

Aprendizado Transdutivo Baseado em Teoria da Informação e Teoria do Aprendizado Estatístico

Thiago Turchetti Maia

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica

Universidade Federal de Minas Gerais

Julho 2007

© Thiago Turchetti Maia, 2007

Resumo

O problema de aprendizado de máquinas normalmente é proposto e resolvido dentro do paradigma da inferência indutiva, através de princípios clássicos como a minimização do risco empírico (ERM) ou a minimização do risco estrutural (SRM). Estes princípios se baseiam na criação de regras através de uma quantidade finita de dados, e assumem duas premissas fundamentais. Primeiro, que há pontos suficientes nos dados de treinamento e estes são amostrados i.i.d. (independentemente e identicamente distribuídos) em relação à distribuição do problema original. Segundo, que é necessário criar uma regra geral para estimar pontos desconhecidos em qualquer parte do espaço do problema.

O modelo de aprendizado transdutivo desafia essas premissas, quebrando o conceito de indução de uma regra genérica que é então utilizada para deduzir pontos desconhecidos de uma dada função. Em uma abordagem transdutiva, valores de pontos desconhecidos são estimados diretamente, sem a necessidade de se estimar a própria função genérica e, implicitamente, todas as suas possíveis avaliações no espaço contínuo do problema. Além disso, o aprendizado transdutivo geralmente utiliza a informação contida nos próprios pontos de interesse para melhorar sua estimativa das distribuições de probabilidades do problema e aumentar seu desempenho. De acordo com o princípio da Navalha de Occam, o aprendizado transdutivo não só evita a solução de um problema mais difícil para o qual pode-se não haver informações suficientes, mas ele também ine-

rentemente utiliza toda informação disponível *a priori*, incluindo a contida nos próprios pontos de interesse.

Neste trabalho, uma nova abordagem para o aprendizado transdutivo é proposta baseada em uma variação da teoria de aprendizado estatístico, onde funcionais risco são diretamente estimados em um arcabouço de aprendizado semi-supervisionado e transdutivo. A partir desta abordagem, um novo método livre de parâmetros, o classificador entrópico essencialmente transdutivo (ETEC), é proposto utilizando conceitos da teoria da informação, onde o problema de aprendizado é resolvido como um problema de otimização. Os resultados apresentados indicam que este método supera outros métodos indutivos e transdutivos usuais, como máquinas de vetores de suporte (SVMs) e máquinas de vetores de suporte transdutivas (TSVMs), para problemas de classificação binária em contextos selecionados, identificados através de indicadores de informação propostos.

Abstract

The machine learning problem is most frequently proposed and solved under the inductive inference paradigm, based on classical inductive principles such as the Empirical Risk Minimization (ERM) and Structural Risk Minimization (SRM). These principles are based on the attempt to create a learning rule given some finite training data. They rely on two assumptions: first, that there are enough observations in the training data and that these data are drawn independently and identically distributed (i.i.d.) with respect to the original data distribution of the problem, and second, that one actually needs to determine a general rule able to estimate points anywhere in the space of the problem.

Transductive learning challenges these assumptions, breaking the concept of inducing a general rule and using it to deduce unknown values of a functional dependence. In a transductive setting, one estimates particular values of a functional dependence directly, without necessarily having to estimate the general function itself and, implicitly, all of its possible evaluations in a continuum of points. Furthermore, transductive learning generally takes advantage of the information available at the points of interest to enhance its estimate of the problem's underlying probability distributions and improve its accuracy. Following Occam's Razor principle, transductive learning not only avoids solving a harder problem when there may not be enough information to do so, but it also inherently uses all information available *a priori*, including that given by the points of

interest.

In this work, we propose a new approach to transductive learning based on a variation of statistical learning theory, where risk functionals are directly estimated in a semi-supervised transductive framework. From that, a new parameter-free method, the essentially-transductive entropic classifier (ETEC), is proposed using concepts from information theory, where the learning problem is solved as a constrained optimization problem. In selected contexts identifiable using proposed information indicators, we show that our method outperforms other popular inductive and transductive methods for binary classification problems, such as support vector machines (SVMs) and transductive support vector machines (TSVMs).

Résumé

Le problème d'apprentissage de machines est normalement proposé et résolu sous le paradigme de l'inférence inductive, basée sur des principes classiques comme la minimisation du risque empirique (ERM) ou la minimisation du risque structurale (SRM). Ces principes sont basés sur la création des règles par une quantité finie de données, et supposent deux prémisses fondamentales. Premièrement, qu'y a des points suffisants dans les données d'apprentissage et ceux-ci sont tirés i.i.d. (identiquement et indépendamment distribués) concernant la distribution du problème original. En second, qu'il est nécessaire de créer une règle générale pour estimer des points inconnus en n'importe quel partie de l'espace du problème.

Le modèle d'apprentissage transductif défie ces prémisses, en cassant le concept d'induction d'une règle générique qui alors est utilisée pour déduire des points inconnus d'une dépendance fonctionnelle. Dans un abordage transductif, les valeurs de points inconnus sont estimés directement, sans devoir nécessairement estimer la fonction générique elle-même et, implicitement, toutes ses évaluations possibles dans l'espace continu du problème. Et puis, l'apprentissage transductif généralement utilise les informations contenues aux points d'intérêt eux-mêmes pour améliorer son estimation des distributions de probabilités du problème et augmenter sa performance. Conformément au principe du Rasoir d'Occam, l'apprentissage transductif n'évite pas seulement la solution d'un

problème plus difficile pour lequel il ne peut pas y avoir d'information suffisantes, mais il utilise aussi toutes les informations disponibles *a priori*, y compris celles données par les points d'intérêt.

Dans ce travail, nous avons proposé un nouvel abordage à l'apprentissage transductif basé sur une variation de la théorie d'apprentissage statistique, où les fonctionnels risque sont directement estimés dans un cadre d'apprentissage transductif et semi-supervisé. A partir de cela, on a proposé une nouvelle méthode exempte des paramètres, le classificateur entropic essentiellement transductif (ETEC), en utilisant des concepts de la théorie de l'information, où le problème d'apprentissage est résolu comme un problème d'optimisation. Les résultats présentés indiquent que cette méthode dépasse d'autres méthodes inductives et transductifs usuelles, comme des machines à vecteurs de support (SVMs) et des machines à vecteurs de support transductifs (TSVMs), pour des problèmes de classement binaire dans des contextes sélectionnés, identifiés à travers des indicateurs d'informations proposés.

Sumário

Resumo	ii
Abstract	iv
Résumé	vi
Sumário	viii
Lista de Tabelas	xi
Lista de Figuras	xii
Navalha de Occam	xiv
1 Introdução	1
1.1 O Problema de Aprendizado Computacional	1
1.2 O Problema de Aprendizado	4
1.3 Contribuição Destes Trabalho	6
1.4 Organização da Tese	8
2 Teoria de Aprendizado Estatístico	9
2.1 O Problema de Aprendizado Indutivo	9

2.2	O Problema de Aprendizado Transdutivo	11
3	Máquinas de Vetores de Suporte	14
3.1	Máquinas de Vetores de Suporte Indutivas	14
3.2	Máquinas de Vetores de Suporte Transdutivas	16
3.3	O Problema Minimax para o Aprendizado Transdutivo	19
4	Construindo um Classificador Transdutivo	22
4.1	Transdutividade e Regras de Aprendizado	22
4.2	Uma Abordagem Essencialmente Transdutiva para o Problema de Aprendizado	24
4.3	Estimativas Baseadas em Teoria da Informação	26
4.4	Divergência de Kullback-Leibler	28
4.5	O Problema de Aprendizado como um Problema de Otimização com Restrições	30
4.6	Indicadores	32
4.6.1	Indicadores de Primeira Ordem: Entropia de Conjuntos	33
4.6.2	Indicadores de Segunda Ordem: Entropia de Conjuntos Compostos	33
4.6.3	Indicadores de Terceira Ordem: Medida de Transdutividade	34
5	Experimentos e Resultados	36
5.1	Bases de Dados	36
5.2	Amostragem de Dados	37
5.2.1	Amostragem Polarizada e Não Polarizada	37
5.2.2	Tamanho dos Conjuntos de Treinamento e Trabalho	38
5.3	Métodos de Referência	39
5.4	Ajuste de Parâmetros	40

5.5	Incorporando Informações Adicionais	41
5.6	Resultados de Experimentos	43
5.6.1	Resultados para bcw	43
5.6.2	Resultados para hepatitis	44
5.6.3	Resultados para ionosphere	46
5.6.4	Resultados para musk	47
5.6.5	Resultados para pgs	49
5.6.6	Resultados para pid	50
5.6.7	Resultados para wpbc	52
5.7	Comparação Gráfica	53
6	Conclusão	69
	Referências Bibliográficas	72

Lista de Tabelas

5.1	Bases de dados e suas partições utilizadas em experimentos.	37
5.2	Resultados do procedimento de ajuste de parâmetros, utilizados nos ex- perimentos com SVM e TSVM.	41
5.3	Resultados de ETEC para bcw , distribuição polarizada.	43
5.4	Resultados de ETEC para bcw , distribuição não polarizada.	44
5.5	Resultados de ETEC para hepatitis , distribuição polarizada.	45
5.6	Resultados de ETEC para hepatitis , distribuição não polarizada.	45
5.7	Resultados de ETEC para ionosphere , distribuição polarizada.	46
5.8	Resultados de ETEC para ionosphere , distribuição não polarizada. . . .	47
5.9	Resultados de ETEC para musk , distribuição polarizada.	48
5.10	Resultados de ETEC para musk , distribuição não polarizada.	48
5.11	Resultados de ETEC para pgs , distribuição polarizada.	49
5.12	Resultados de ETEC para pgs , distribuição não polarizada.	50
5.13	Resultados de ETEC para pid , distribuição polarizada.	51
5.14	Resultados de ETEC para pid , distribuição não polarizada.	51
5.15	Resultados de ETEC para wdbc , distribuição polarizada.	52
5.16	Resultados de ETEC para wdbc , distribuição não polarizada.	53

Lista de Figuras

2.1	Modelo de aprendizado baseado na inferência indutiva, onde a função que melhor aproxima a dependência funcional desconhecida é selecionada de um conjunto de funções de acordo com os dados de treinamento.	10
2.2	Modelo de aprendizado baseado na inferência transdutiva. Uma máquina de aprendizado modificada recebe os conjuntos de treinamento e trabalho como entrada, e o avaliador é capaz apenas de estimar valores presentes em um destes conjuntos.	12
5.1	Resultados gráficos para bcw , amostragem polarizada.	55
5.2	Resultados gráficos para bcw , amostragem não polarizada.	56
5.3	Resultados gráficos para hepatitis , amostragem polarizada.	57
5.4	Resultados gráficos para hepatitis , amostragem não polarizada.	58
5.5	Resultados gráficos para ionosphere , amostragem polarizada.	59
5.6	Resultados gráficos para ionosphere , amostragem não polarizada.	60
5.7	Resultados gráficos para musk , amostragem polarizada.	61
5.8	Resultados gráficos para musk , amostragem não polarizada.	62
5.9	Resultados gráficos para pgs , amostragem polarizada.	63
5.10	Resultados gráficos para pgs , amostragem não polarizada.	64
5.11	Resultados gráficos para pid , amostragem polarizada.	65

5.12	Resultados gráficos para pid , amostragem não polarizada.	66
5.13	Resultados gráficos para wpbc , amostragem polarizada.	67
5.14	Resultados gráficos para wpbc , amostragem não polarizada.	68

Entia non sunt multiplicanda praeter necessitatem.

(Entidades não devem ser multiplicadas além da necessidade.)

Lex parsimoniae, ou *Navalha de Occam*,
por William de Occam, frade Franciscano e filósofo do século XIV.

Capítulo 1

Introdução

1.1 O Problema de Aprendizado Computacional

O problema de ensinar uma máquina como adquirir novo conhecimento tem sido explorado desde a introdução do computador moderno. O primeiro modelo de neurônio artificial foi criado por McCulloch e Pitts em 1943 [MP43], em seu trabalho clássico que descreve o que deveria ser a unidade básica de uma rede de neurônios artificiais inspirada no cérebro humano. O foco desse trabalho foi muito mais sobre a capacidade computacional do modelo proposto, do que em métodos pragmáticos para aquisição de conhecimento. Mais tarde em 1949, o aprendizado destas redes foi explorado por Hebb [Heb49], que propôs um modelo de ajuste de pesos nas entradas de cada neurônio. Este modelo se baseou na teoria de que neurônios biológicos aprendem baseados no reforço de suas ligações sinápticas para outros neurônios excitados. Widrow e Hoff [WH60] refinaram ainda mais as idéias de Hebb em 1960, incorporando o método do gradiente descendente para minimizar o erro de saída de cada neurônio. A primeira tentativa de abordar o problema de reconhecimento de padrões a partir de redes de neurônios foi feita por Rosenblatt em 1958 [Ros58], onde ele introduziu o modelo *Perceptron*. O Perceptron

é uma simples rede capaz de resolver problemas de classificação linearmente separáveis no qual o espaço de saída é dividido em regiões complementares que correspondem a categorias.

Apesar desta abordagem conexionista ter recebido grande atenção devido ao trabalho de Rosenblatt, a comunidade de aprendizado de máquina foi desencorajada por Minsky e Papert em 1969 [MP69] a continuar seu desenvolvimento. Minsky e Papert argumentaram que a limitação de não ser capaz de resolver problemas não-linearmente separáveis era muito grave para ser ignorada, além de apontarem outros problemas como o crescimento explosivo do consumo de recursos no tempo e espaço, além da falta de uma regra de aprendizado para redes multi-camadas. Salvo raras exceções, a década de 1970 foi marcada pela falta de interesse da comunidade na área. Apenas após o trabalho de Hopfield em 1982 [Hop82] e o desenvolvimento do algoritmo *backpropagation* em 1986 [RHW86] que redes de neurônios artificiais ¹ novamente ganharam interesse geral. Hopfield explorou as relações entre redes associativas recorrentes e modelos físicos, o que iniciou uma série de outros trabalhos que utilizaram teorias da Física para descrever o comportamento de máquinas de aprendizado [Hop82]. Finalmente, muita atenção foi dada ao trabalho de Rumelhart et al. [RHW86], que criaram um algoritmo, o *backpropagation*, para treinar redes com múltiplas camadas baseado no ajuste de pesos de cada camada de acordo com a retro-propagação de erros de saída.

Esse ressurgimento da escola conexionista na década de 1980 foi seguido por uma nova onda de pesquisas no início da década de 1990 que exploraram novas teorias e algoritmos focados não só na construção e treinamento de máquinas de aprendizado poderosas, mas também na reorganização do ambiente ao seu redor visando o aumento de seu desempenho. Vapnik [Vap95] se refere a este período como o retorno às origens

¹Redes de neurônios artificiais são geralmente chamadas pela comunidade de aprendizado de máquina brasileira de *redes neurais artificiais*, ou ainda *redes neuronais artificiais*.

da teoria de aprendizado estatístico. Trabalhos pioneiros como os de Munro [Mun92], Schapire [Sch92], Drucker et al. [DSS93], e Cachin [Cac94] serviram de base para outros como Breiman [Bre94], que introduziram o conceito de *Bagging* para o treinamento e combinação de múltiplas cópias de um algoritmo de aprendizado, e finalmente Freund e Schapire, que em seguida introduziram a família de algoritmos AdaBoost. O termo *Boosting* foi apresentado por Schapire em 1990 [Sch90], no trabalho que mostrou pela primeira vez que qualquer algoritmo de tempo polinomial que gera hipóteses cujos erros são arbitrariamente melhores que uma hipótese aleatória pode ser transformado em um algoritmo de tempo polinomial com erro arbitrariamente pequeno.

Em paralelo com o desenvolvimento de modelos de aprendizado e metodologias para a melhoria do desempenho de algoritmos já existentes, este retorno às origens também motivou a criação de diversas novas máquinas de aprendizado. Entre elas, destacam-se as de Vapnik e colaboradores que ficaram sendo conhecidas como *Support Vector Machines* (SVMs), ou máquinas de vetores de suporte. Baseados em outros trabalhos de Vapnik e Chervonenkis de 1965 [VC71], nos quais métodos de hiperplanos de separação ótimos foram desenvolvidos, Vapnik et al. criaram a idéia de máquina de vetores de suporte em 1992 [Vap95]. Desde então, o crescente interesse da comunidade de aprendizado de máquina motivou o desenvolvimento de diversos algoritmos de treinamento especializados na resolução dos problemas de otimização com restrições convexas e quadráticos sobre os quais SVMs são formuladas.

Desde os anos 1990, muita atenção foi dada à teoria do aprendizado computacional, ou teoria do aprendizado estatístico [Vap95]. Dentre todos os tópicos que atraíram pesquisadores, a inferência transdutiva se destaca por ainda estar em seus estágios preliminares de desenvolvimento, mesmo tendo sido introduzida nos anos 1970 por Vapnik [VC71, Vap79]. Muitas de suas questões, dentre elas seu potencial para produzir mode-

los e aplicações de aprendizado de máquina mais eficientes, ainda não foram totalmente respondidas.

1.2 O Problema de Aprendizado

Considere o problema de aprendizado como um problema genérico de estimação de funções. Podemos resolvê-lo dentro do paradigma de *inferência indutiva* através de princípios indutivos clássicos, como o princípio de Minimização do Risco Empírico (ou ERM, *Empirical Risk Minimization*) e o princípio de Minimização do Risco Estrutural (ou SRM, *Structural Risk Minimization*). Independentemente de seus detalhes de funcionamento, o objetivo de ambos princípios é o mesmo: criar uma regra de aprendizado que minimize o risco para uma massa fixa de dados de treinamento. De acordo com a teoria estatística, uma regra de baixo risco produz, com grande probabilidade, baixas médias de erro quando aplicada sobre uma grande quantidade de dados de validação. Uma vez que estas regras são criadas, elas podem ser utilizadas para estimar dependências funcionais em quaisquer pontos fornecidos posteriormente.

Estes princípios indutivos se baseiam em duas premissas. Primeiro, que há observações suficientes nos dados de treinamento e que estas observações são amostradas independentemente e identicamente distribuídas (*i.i.d.*) em relação à distribuição original do problema. Em outras palavras, que os dados de treinamento fornecem cobertura suficiente do espaço do problema e que eles contêm informações suficientes para estimar valores em pontos arbitrários dados posteriormente pelos dados de validação. Segundo, que para estimar valores nestes pontos, precisa-se realmente criar uma regra genérica capaz de estimar valores em qualquer ponto do espaço do problema.

De forma interessante, a maioria das situações práticas é bem diferente. Dados de

treinamento são normalmente escassos, e podem facilmente falhar em descrever bem o espaço do problema. Além disso, normalmente estamos interessados em estimar valores apenas em um conjunto fixo de pontos que está disponível antes da solução do problema, ou seja, uma regra genérica é desnecessária.

No caso onde o objetivo é estimar valores de dependências funcionais em pontos desconhecidos, o paradigma indutivo requer que dois problemas sejam resolvidos. Primeiro, deve-se estimar uma regra genérica que contemple todo o espaço do problema. Segundo, deve-se estimar os valores dos pontos de interesse através desta regra. Note que o problema de estimar uma dependência funcional é muito mais difícil, uma vez que implica na utilização de pontos de treinamento limitados para inferir todos os possíveis valores no espaço do problema.

Podemos definir estes dois passos através da filosofia clássica. O primeiro é um passo *indutivo*, onde dados específicos dos pontos de treinamento são transformados em informações genéricas que descrevem a dependência funcional estimada. O segundo é um passo *dedutivo*, onde a regra genérica é utilizada para estimar valores específicos de pontos de interesse.

O principal conceito da chamada *inferência transdutiva* é dar um passo de dados específicos para dados específicos, isto é, estimar valores de dependências funcionais desconhecidas em dados pontos de interesse sem necessariamente ter que estimar as funções propriamente ditas. Note que como consequência, estamos não só resolvendo um problema mais simples, mas também utilizando toda informação disponível nos pontos de interesse, informação esta que é negligenciada pela inferência indutiva. Se interpretarmos o princípio da *Navalha de Occam* para o processo de aprendizado de máquina, o passo de generalização imposto pela inferência indutiva é desnecessário para gerar a mesma informação dada pela inferência transdutiva.

De acordo com Vapnik, o problema de estimar os valores de dependências funcionais desconhecidas é análogo à questão de qual é a base da inteligência humana: o conhecimento de leis (regras) ou a cultura de acesso direto à verdade (intuição, inferência ad-hoc). Baseado neste questionamento filosófico, Vapnik diz que *o modelo de aprendizado transdutivo pode produzir a maior contribuição aos 2000 anos de discussões sobre a essência do raciocínio humano* [Vap95].

1.3 Contribuição Destes Trabalho

Neste trabalho nós examinamos a teoria, métodos e aplicações da inferência transdutiva. Nós iniciamos a discussão pela avaliação da teoria clássica de aprendizado estatístico, assim como sua aplicabilidade ao problema transdutivo. Nós identificamos questões que tornam difícil o uso da teoria atual como base de novas máquinas de aprendizado transdutivo. Particularmente, nós mostramos como os funcionais risco definidos para o aprendizado transdutivo não podem ser diretamente computados, ao contrário dos funcionais risco propostos para o aprendizado indutivo. Em seguida, nós definimos claramente o que é inferência indutiva, inferência transdutiva, aprendizado supervisionado, aprendizado não-supervisionado e aprendizado semi-supervisionado. Apesar destes conceitos serem frequentemente utilizados na literatura, eles muitas vezes são confundidos e trocados uns pelos outros. Baseados nestas definições, nós propomos variações da teoria clássica para descrever cada tipo de problema. Estas variações oferecem funcionais risco que podem ser efetivamente calculados.

A fim de estimar os funcionais risco propostos para o aprendizado transdutivo, foi utilizado o método da Informação Discriminante Mínima (MDI, *Minimum Discrimination Information*), baseado na divergência de Kullback-Leibler entre duas distribuições de pro-

bilidade. Foram utilizadas as definições contínuas da divergência de Kullback-Leibler e de outras medidas de entropia, uma vez que o espaço de busca produzido por heurísticas contínuas, ao contrário de heurísticas discretas, é normalmente mais fácil de ser percorrido. Através de premissas de independência e gaussianidade dos dados, chegamos a expressões de risco elegantes e computacionalmente eficientes. Além disso, utilizamos uma matriz de covariância ponderada para parametrizar nossas distribuições multivariáveis, assinalando um peso a cada amostra do conjunto de dados. Nós então propomos a solução do problema como um problema de otimização com restrições que busca o risco mínimo no espaço formado pelos pesos associados a cada amostra. O problema é então solucionado através de um método Quasi-Newton de programação quadrática sequencial. Este trabalho, portanto, introduz um novo método, livre de parâmetros, para problemas transdutivos de classificação binária.

O método proposto foi testado em 14 problemas diferentes, amostrados de 7 bases de dados usualmente utilizadas na literatura. Os resultados obtidos foram comparados aos resultados gerados por outros dois métodos bem conhecidos, a máquina de vetores de suporte clássica (SVM), e a chamada máquina de vetores de suporte transdutiva (TSVM). Nossos resultados indicam que nosso método tem desempenho superior em contextos selecionados. A fim de identificar estes contextos, propomos um conjunto de indicadores de informação que revelam quais problemas de aprendizado podem ser resolvidos através do método proposto. Por fim, este trabalho também introduz uma metodologia capaz de avaliar a aplicabilidade de máquinas transdutivas a um dado problema de aprendizado.

1.4 Organização da Tese

Esta tese está organizada da seguinte forma. O Capítulo 1 traz a introdução, que se encerra nesta seção. O Capítulo 2 faz uma breve revisão da teoria de aprendizado estatístico, enquanto o Capítulo 3 revê máquinas de vetores de suporte. Com base nos conceitos introduzidos nos capítulos anteriores, o Capítulo 4 traz as propostas e métodos introduzidos neste trabalho. O Capítulo 5 traz os experimentos realizados e os resultados obtidos, enquanto o Capítulo 6 apresenta as conclusões do trabalho.

Capítulo 2

Teoria de Aprendizado Estatístico

Este capítulo traz a revisão das formulações dos problemas de aprendizado de acordo com a teoria de aprendizado estatístico.

2.1 O Problema de Aprendizado Indutivo

De acordo com a teoria do aprendizado estatístico, o problema de aprendizado pode ser formulado como um problema de estimação de funções. Apesar de sua saída ser diferente de acordo com cada domínio, nós assumimos que todas as entradas de qualquer problema de aprendizado têm o formato de um vetor de números reais em um dado espaço $X \subset \mathbb{R}^n$. Baseados nesta definição genérica, podemos determinar um modelo de aprendizado genérico para resolver o problema de aproximação de uma dependência funcional desconhecida.

O modelo possui quatro componentes (Figura 2.1). Há um gerador de dados (DG), que seleciona amostras aleatórias $x \in X \subset \mathbb{R}^n$, amostradas independentemente de uma distribuição de probabilidades desconhecida mas fixa $F(x)$. Há um oráculo – ou supervisor – (O), que retorna um valor de saída $y \in Y$ para qualquer amostra de entrada

$\mathbf{x}$ de acordo com uma distribuição condicional fixa e desconhecida $F(y|\mathbf{x})$. Note que Y é arbitrário para o problema genérico de aprendizado. Existe um avaliador E capaz de implementar um conjunto de funções $f(\mathbf{x}, \alpha), \alpha \in \Lambda$, onde Λ é um conjunto arbitrário de parâmetros que controla o comportamento da função. Note que ao fazer Λ arbitrário, $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ pode ser qualquer conjunto de funções. Por fim, há uma máquina de aprendizado (LM), que deve selecionar a função do conjunto $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ que melhor aproxima a saída do oráculo de acordo com o conjunto de treinamento. Este conjunto de treinamento é um conjunto finito de amostras

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \quad (2.1)$$

amostradas *i.i.d.* por DG em relação a $F(\mathbf{x}, y) = F(\mathbf{x})F(y|\mathbf{x})$.

Uma vez que a máquina é treinada, o valor $\alpha_0 \in \Lambda$ é determinado e produz a função $f(\mathbf{x}, \alpha_0)$ através de E que melhor aproxima a dependência funcional desconhecida. Esta aproximação pode então ser utilizada por E para estimar os valores desconhecidos $\hat{y}^*$ em pontos arbitrários $\mathbf{x}^* \in X$. Este modelo é similar ao modelo clássico descrito pela teoria de aprendizado, apenas exposto em mais detalhe para distinguir entre LM e E .

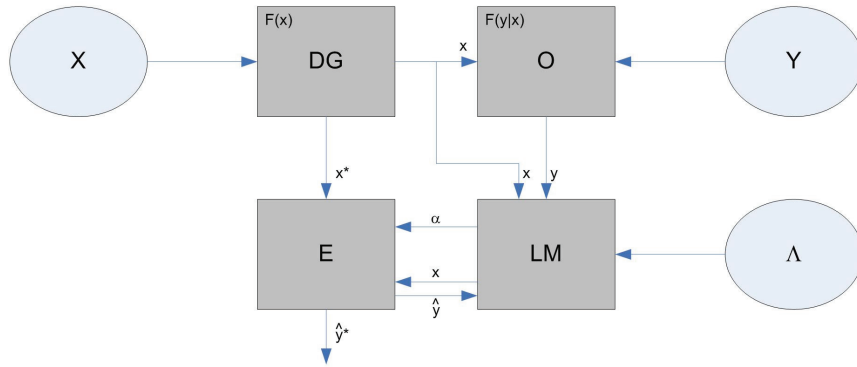


Figura 2.1: Modelo de aprendizado baseado na inferência indutiva, onde a função que melhor aproxima a dependência funcional desconhecida é selecionada de um conjunto de funções de acordo com os dados de treinamento.

Note que, em qualquer momento do treinamento, LM pode apresentar qualquer

amostra $\mathbf{x} \in X$ a E e esperar uma resposta $\hat{y} = f(\mathbf{x}, \alpha)$. Para amostras $\mathbf{x}$ no conjunto (2.1), $\hat{y}$ pode ser comparado à resposta y dada pelo oráculo O . Esta comparação, a base do mecanismo de seleção que determina a aproximação da dependência funcional, é feita por uma função de perda ou discrepância $L(y, f(\mathbf{x}, \alpha))$. Uma vez que o problema indutivo assume que E é capaz de estimar valores $\hat{y}^*$ da dependência funcional desconhecida em pontos de interesse arbitrários $\mathbf{x}^* \in X$, note que $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ deve ser definido para todos $\mathbf{x}^* \in X \subset \mathbb{R}^n$. Frequentemente, o conjunto X é contínuo, em muitos casos simplesmente $X = R^m \subset R^n, m < n$. A não ser que X seja finito tal que $X \subset R^m \subset R^n$, precisaríamos que $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ fosse continuamente definido em X .

Mesmo não tendo definido uma função de perda, podemos definir o valor esperado da perda dado pelo funcional risco

$$R(\alpha) = \int L(y, f(\mathbf{x}, \alpha)) dF(\mathbf{x}, y), \quad (2.2)$$

conhecido com risco esperado.

A solução do problema de aprendizado, implementada por LM , pode portanto ser descrita como a busca da função $f(\mathbf{x}, \alpha_0)$ que minimiza o risco esperado. Em outras palavras, a solução é a busca da função $f(\mathbf{x}, \alpha_0)$ que minimiza o funcional risco $R(\alpha)$ sobre a classe de funções $f(\mathbf{x}, \alpha), \alpha \in \Lambda$, onde $F(\mathbf{x}, y)$ é desconhecido e os únicos dados disponíveis estão no conjunto de treinamento (2.1).

2.2 O Problema de Aprendizado Transdutivo

Examinamos agora a versão transdutiva do problema de aprendizado. Observe as duas principais motivações para o uso da inferência transdutiva, dadas no Capítulo 1. Primeiro, que na maioria das vezes estamos apenas interessados em estimar o valor de dependências funcionais desconhecidas em um conjunto finito de pontos de interesse.

Segundo, que desejamos incorporar a informação contida nestes pontos de interesse à resposta, uma vez que o conjunto de treinamento da Equação (2.1) pode não fornecer informações suficientes do espaço do problema.

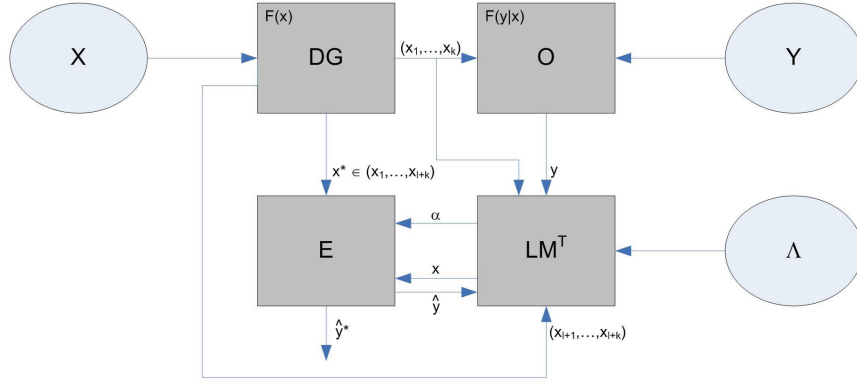


Figura 2.2: Modelo de aprendizado baseado na inferência transdutiva. Uma máquina de aprendizado modificada recebe os conjuntos de treinamento e trabalho como entrada, e o avaliador é capaz apenas de estimar valores presentes em um destes conjuntos.

Assim como na inferência indutiva, somos capazes de criar um modelo de aprendizado genérico para inferência transdutiva baseado nos mesmos quatro componentes (Figure 2.2). Três destes componentes, DG , O e E , são os mesmos do modelo indutivo. A diferença está na máquina de aprendizado LM^T , que agora recebe tanto o conjunto de treinamento original da Equação (2.1) quanto um conjunto de trabalho

$$(\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}), \quad (2.3)$$

também amostrado *i.i.d.* por DG . Este conjunto representa os k pontos de interesse onde desejamos estimar os valores $\hat{y}^* = f(\mathbf{x}^*, \alpha^*)$ da dependência funcional desconhecida, sendo $\mathbf{x}^* \in (\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k})$. Por fim, a comunicação entre DG e E fica restrita, uma vez que E , mesmo sendo idêntico à sua versão indutiva, agora só é capaz de estimar os valores de pontos presentes tanto no conjunto de treinamento (2.1) ou no conjunto de trabalho (2.3). Note que, ao contrário do caso indutivo, não é necessário que as funções

do conjunto $f(\mathbf{x}, \alpha)$, $\alpha \in \Lambda$, $\mathbf{x} \in X$ sejam continuamente definidas em X .

Na seção 2.1, nós descrevemos o problema indutivo de aprendizado como a busca pela função que minimiza o risco esperado dado pelo funcional risco (2.2). Para a inferência transdutiva, a solução do problema, implementada por LM^T , pode ser descrita como a busca pela função que minimiza o *risco total*. Mais formalmente, LM^T deve encontrar a função $f(\mathbf{x}, \alpha^*)$ que, com probabilidade $1 - \eta$, minimiza o risco total de estimar valores $\hat{y}^* = f(\mathbf{x}^*, \alpha^*)$ para $\mathbf{x}^* \in (\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k})$. Isto é análogo a minimizar o funcional

$$R_{\Sigma}(\alpha) = \frac{1}{k} \sum_{i=1}^k L(y_{\ell+i}, f(\mathbf{x}_{\ell+i}, \alpha)), \quad (2.4)$$

onde $L(y, f(\mathbf{x}, \alpha))$ é uma função de perda genérica.

Note como a medida de risco difere entre as versões indutiva e transdutiva do problema de aprendizado. No caso indutivo, o funcional risco (2.2) mede o risco em relação à distribuição conjunta $F(\mathbf{x}, y)$ sobre todo o espaço da dependência funcional desconhecida. No caso transdutivo, o funcional risco (2.4) mede apenas o risco em relação aos pontos de interesse. No Capítulo 1 foi colocado que o problema de estimar valores em pontos de interesse é mais fácil para uma máquina do que estimar uma regra genérica que aproxima uma dependência funcional. A diferença entre as medidas de risco de (2.2) e (2.4) confirma isto [Vap98].

Capítulo 3

Máquinas de Vetores de Suporte

Neste capítulo descrevemos as versões indutiva e transdutiva mais usuais de máquinas de vetores de suporte (SVM).

3.1 Máquinas de Vetores de Suporte Indutivas

Dentre as muitas formulações possíveis de máquinas de vetores de suporte, descrevemos a seguir uma das definições mais usuais, que utiliza o princípio de funções de kernel para induzir o aprendizado em espaços de Hilbert, como descrito por Cristianini e Shawe-Taylor [CST00] e Haykin [Hay94]. Considere o seguinte problema de otimização para a formulação desta SVM, particularmente o caso onde a flexibilização da margem é dada pela norma-1 do vetor de pesos:

$$\begin{aligned} \text{minimize}_{\xi, \mathbf{w}, b} \quad & \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^l \xi_i, \\ \text{sujeito a} \quad & y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \dots, l, \\ & \xi_i \geq 0, i = 1, \dots, l. \end{aligned} \tag{3.1}$$

onde $\mathbf{w}$ é o vetor de pesos, $\mathbf{x}$ é o vetor de entrada, C é o parâmetro de regularização

entre o erro de treinamento e a margem, e ξ é o vetor responsável pela flexibilização do hiperplano de separação.

O Lagrangiano para o problema de otimização descrito na Equação (3.1) é:

$$L_P(\mathbf{w}, b, \xi, \alpha, \mathbf{r}) = \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle + 2 \sum_{i=1}^l \xi_i - \sum_{i=1}^l \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^l r_i \xi_i, \quad (3.2)$$

onde $\alpha_i \geq 0$ e $r_i \geq 0$. São então calculadas as derivadas parciais para L_P :

$$\frac{\partial L_P(\mathbf{w}, b, \xi, \alpha, \mathbf{r})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i = \mathbf{0}, \quad (3.3)$$

$$\frac{\partial L_P(\mathbf{w}, b, \xi, \alpha, \mathbf{r})}{\partial \xi_i} = C - \alpha_i - r_i = 0, \quad (3.4)$$

$$\frac{\partial L_P(\mathbf{w}, b, \xi, \alpha, \mathbf{r})}{\partial b} = \sum_{i=1}^l y_i \alpha_i = 0. \quad (3.5)$$

Ao substituir estas relações na forma primal, a seguinte forma dual do problema é obtida:

$$L_D(\mathbf{w}, b, \xi, \alpha, \mathbf{r}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \quad (3.6)$$

As condições de complementaridade de Karush-Kuhn-Tucker para o problema são portanto:

$$\begin{aligned} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] &= 0, \quad i = 1, \dots, l, \\ \xi_i (\alpha_i - C) &= 0, \quad i = 1, \dots, l. \end{aligned} \quad (3.7)$$

A proposição 3.1.1 a seguir formaliza todo o problema.

Proposição 3.1.1. *Considere a classificação de um exemplo de treinamento:*

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)),$$

através do espaço implicitamente induzido pelo kernel $K(\mathbf{x}, \mathbf{z})$, e suponha que os parâmetros α' resolvem o seguinte problema de otimização quadrática:

$$\begin{aligned} \text{maximize} \quad & L_D(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \\ \text{sujeito a} \quad & \sum_{i=1}^l y_i \alpha_i = 0, \\ & C \geq \alpha_i \geq 0, i = 1, \dots, l. \end{aligned}$$

Considere $f(\mathbf{x}) = \sum_{i=1}^l y_i \alpha'_i K(\mathbf{x}_i, \mathbf{x}) + b'$, onde b' é escolhido tal que $y_i f(\mathbf{x}_i) = 1$ para qualquer i com $C > \alpha'_i > 0$. Esta regra de decisão, dada por $\text{sign}(f(\mathbf{x}))$, é equivalente ao hiperplano no espaço induzido implicitamente definido pelo kernel $K(\mathbf{x}, \mathbf{z})$ que é resolvido pelo problema de otimização (3.1), onde as variáveis de folga (ξ) são definidas relativas à margem geométrica:

$$\gamma = \left(\sum_{j \in \{sv\}} y_i y_j \alpha'_i \alpha'_j K(\mathbf{x}_i, \mathbf{x}_j) \right)^{-\frac{1}{2}}.$$

3.2 Máquinas de Vetores de Suporte Transdutivas

Descrevemos agora a formulação de máquinas de vetores de suporte transdutivas, como introduzidas em [Vap99].

Considere um problema de reconhecimento de padrões com os seguintes dados de treinamento

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \quad \mathbf{x} \in X \subset \mathbb{R}, y \in Y = \{-1, 1\} \quad (3.8)$$

e dados de validação

$$\mathbf{x}_{\ell+1}^*, \dots, \mathbf{x}_{\ell+k}^*. \quad (3.9)$$

O objetivo, no caso linear, é encontrar a função do conjunto de todas as funções lineares

$$y = (\mathbf{w} \cdot \mathbf{x}) + b \quad (3.10)$$

que minimiza o número de erros resultantes de um dado conjunto de validação (3.9). No caso separável, uma boa solução para o problema também classifica os padrões de validação com rótulos

$$y_{\ell+1}^*, \dots, y_{\ell+k}^*. \quad (3.11)$$

Além disso, a completa sequência de dados de treinamento e validação

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), (\mathbf{x}_{\ell+1}^*, y_{\ell+1}^*), \dots, (\mathbf{x}_{\ell+k}^*, y_{\ell+k}^*) \quad (3.12)$$

é separada com margem máxima por esta solução.

Em outras palavras, o objetivo do problema é encontrar classificações (3.11) dos vetores de validação (3.9) para os quais o hiperplano ótimo

$$y = (\mathbf{w}_0^* \cdot \mathbf{x}) + b_0 \quad (3.13)$$

maximiza a margem quando este separa ambos dados de treinamento e testes (3.12), onde $\mathbf{w}_0^*$ denota o hiperplano ótimo que classifica corretamente os dados de validação (3.9) de acordo com (3.11), isto é,

$$\mathbf{w}_0^* = \mathbf{w}_0(y_{\ell+1}^*, \dots, y_{\ell+k}^*). \quad (3.14)$$

Mais formalmente, o objetivo é encontrar classificações $y_{\ell+1}^*, \dots, y_{\ell+k}^*$ que minimizam o funcional

$$\Phi(\mathbf{w}_0(y_{\ell+1}^*, \dots, y_{\ell+k}^*)) = \min_{\mathbf{w}_0^*} \frac{1}{2} \|\mathbf{w}_0^*\|^2 \quad (3.15)$$

respeitando as seguintes inequações:

$$y_i[(\mathbf{x}_i \cdot \mathbf{w}_0^*) + b] \geq 1, \quad i = 1, \dots, \ell, \quad (3.16)$$

$$y_j^*[(\mathbf{x}_j^* \cdot \mathbf{w}_0^*) + b] \geq 1, \quad j = \ell+1, \dots, \ell+k. \quad (3.17)$$

Para o caso não separável, devemos introduzir variáveis de folga ao funcional (3.15) (onde C e C^* são constantes não negativas que controlam a flexibilidade da margem):

$$\Phi(\mathbf{w}_0(y_{\ell+1}^*, \dots, y_{\ell+k}^*), \xi, \xi^*) = \min_{\mathbf{w}_0^*, \xi, \xi^*} \left[\frac{1}{2} \|\mathbf{w}_0^*\|^2 + C \sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=\ell+1}^{\ell+k} \xi_j^* \right]. \quad (3.18)$$

Como no caso indutivo, podemos reescrever o problema na forma de um problema de otimização com restrições:

$$\begin{aligned}
& \text{minimize} \quad \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=\ell+1}^{\ell+k} \xi_j^*, \\
& \text{sujeito a} \quad y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \dots, \ell, \\
& \quad y_j^* (\langle \mathbf{w} \cdot \mathbf{x}_j^* \rangle + b) \geq 1 - \xi_j^*, j = \ell + 1, \dots, \ell + k, \\
& \quad \xi_i \geq 0, i = 1, \dots, \ell, \\
& \quad \xi_j^* \geq 0, j = \ell + 1, \dots, \ell + k.
\end{aligned} \tag{3.19}$$

De forma análoga, resolvemos o problema com multiplicadores de Lagrange e introduzimos espaços induzidos por funções de kernel (note que os índices transdutivos j foram deslocados de $(\ell + 1, \dots, \ell + k)$ para $(1, \dots, k)$ para facilitar a leitura):

$$\begin{aligned}
& \text{maximize} \quad L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \sum_{i=1}^{\ell} \alpha_i + \sum_{j=1}^k \alpha_j^* \\
& \quad - \frac{1}{2} \sum_{i,r=1}^{\ell} y_i y_r \alpha_i \alpha_r K(\mathbf{x}_i, \mathbf{x}_r) \\
& \quad - \frac{1}{2} \sum_{j,r=1}^k y_j^* y_r^* \alpha_j^* \alpha_r^* K(\mathbf{x}_j^*, \mathbf{x}_r^*) \\
& \quad - \sum_{j=1}^{\ell} \sum_{r=1}^k y_j y_r^* \alpha_j \alpha_r^* K(\mathbf{x}_j, \mathbf{x}_r^*), \\
& \text{sujeito a} \quad 0 \leq \alpha_i \leq C, i = 1, \dots, \ell \\
& \quad 0 \leq \alpha_j^* \leq C^*, j = \ell + 1, \dots, \ell + k \\
& \quad \sum_{i=1}^{\ell} \alpha_i y_i + \sum_{j=\ell+1}^{\ell+k} \alpha_j^* y_j^* = 0.
\end{aligned} \tag{3.20}$$

Note que, apesar de ser uma configuração transdutiva, os parâmetros encontrados pela solução do problema (3.20) podem ser utilizados para construir uma regra de decisão, assim como no caso indutivo:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot K(\mathbf{x}, \mathbf{x}_i) + \sum_{j=1}^k y_j^* \alpha_j^* \cdot K(\mathbf{x}, \mathbf{x}_j^*) + b \right). \tag{3.21}$$

3.3 O Problema Minimax para o Aprendizado Transdutivo

Nesta seção revemos alguns algoritmos de treinamento para máquinas de vetores de suporte indutivas. No caso indutivo, o treinamento de máquinas de vetores de suporte é equivalente à solução de um problema de otimização convexo com restrições. Há diversos algoritmos de otimização na literatura que podem ser utilizados com este propósito, onde alguns se destacam por utilizarem diferentes técnicas para explorar as particularidades do problema. De qualquer forma, mesmo no caso de algoritmos iterativos, uma vez que o problema dual é resolvido e todas as restrições foram respeitadas, os parâmetros resultantes são válidos e podem ser utilizados para construir uma regra de decisão equivalente a um hiperplano ótimo de separação.

Entretanto, no caso transdutivo não estamos interessados em estimar uma dependência funcional, mas apenas seus valores em dados pontos de interesse. Apesar desta tarefa não parecer tão complexa quanto treinar uma máquina de vetores de suporte indutiva, há o fator complicador de se considerar o conjunto de trabalho $(x_{\ell+1}, \dots, x_{\ell+k})$ no processo de aprendizado.

De acordo com [Vap98], o problema de estimar rótulos $(y_{\ell+1}^*, \dots, y_{\ell+k}^*)$ para o conjunto de trabalho é análogo a um problema minimax, onde o máximo é dado pela solução de problemas de otimização quadrática e o mínimo é dado por todas as possíveis classificações do conjunto de trabalho. Portanto, para o caso de classificação binária, existem 2^k possíveis conjuntos de rótulos que devem ser exaustivamente examinados para garantir que a melhor solução seja encontrada.

É claro que uma busca exaustiva pela solução ótima não é possível quando k é muito maior que algumas poucas amostras. A alternativa, então, é utilizar rotinas

heurísticas que vasculhem este espaço de soluções com menos recursos computacionais, portanto fazendo com que o treinamento de máquinas de vetores de suporte transdutivas seja possível. Assim como no problema de otimização com restrições do caso indutivo, existem várias heurísticas na literatura para resolver o problema minimax. [RN95] faz a revisão de algumas destas técnicas para a solução de problemas minimax genéricos.

Também de forma análoga com o caso indutivo, as heurísticas de treinamento de destaque para o caso transdutivo são aquelas que consideram algumas das particularidades do problema. Entre elas, o trabalho de Joachims [Joa99] foi um dos primeiros a empregar máquinas de vetores de suporte transdutivas em problemas reais. Baseado em problemas de classificação de textos montados a partir de informações de frequência de palavras, Joachims introduziu o algoritmo TSVM baseado na mesma configuração dada pela Equação (3.20), com exceção do uso de diferentes constantes de flexibilização de margens C_- e C_+ para amostras negativas e positivas, respectivamente. Assim como a maioria das heurísticas transdutivas, o TSVM começa a partir de uma solução indutiva do problema. A partir daí, ele incrementa iterativamente as constantes de flexibilização de margens. A medida que C_- e C_+ crescem, diferentes rótulos são permutados, onde cada mudança resulta na solução de um novo problema de otimização. Apesar de não apresentar provas de convergência, Joachims relata que o TSVM é capaz de treinar uma máquina de vetores de suporte com mais de 10.000 pontos de interesse.

Chen et al. introduziram uma modificação ao algoritmo do TSVM, chamado máquina de vetores de suporte transdutiva progressiva (PTSVM), eliminando a necessidade de se estimar a proporção entre amostras positivas e negativas no conjunto de trabalho [CWD03]. Os resultados do PTSVM foram melhores que os do TSVM em alguns casos, tanto em termos de acurácia quanto em termos de tempo de treinamento. Como ressaltado pelos autores, isto sugere que mais pesquisas em diferentes heurísticas baseadas na

Equação (3.20) podem resultar em métodos com desempenho ainda melhor.

Capítulo 4

Construindo um Classificador Transdutivo

Neste capítulo descrevemos as variações da teoria do aprendizado estatístico introduzidas, assim como sua utilização no desenvolvimento de um novo método de aprendizado transdutivo.

4.1 Transdutividade e Regras de Aprendizado

Os conceitos de transdutividade e aprendizado semi-supervisionado nem sempre são uniformemente empregados na literatura. Uma vez que suas diferenças, ou até sua coexistência, são sutis, há trabalhos que confundem os dois.

Vamos inicialmente examinar como a distinção entre aprendizado não supervisionado, supervisionado e semi-supervisionado se dá em função da disponibilidade de dados *a priori*. No aprendizado não supervisionado, deseja-se estimar rótulos associados a um conjunto de dados sem conhecimento passado qualquer sobre os rótulos deste conjunto. No aprendizado supervisionado, há um conjunto de treinamento com rótulos

disponíveis para criar (ou induzir) uma regra de aprendizado, a partir da qual quaisquer pontos de interesse poderão ser avaliados (ou deduzidos) no futuro. No aprendizado semi-supervisionado, existe não só um conjunto de treinamento com rótulos *a priori*, mas também um conjunto de trabalho contendo amostras sem rótulo.

Em seguida, examinamos como a distinção entre aprendizado indutivo e transdutivo depende da criação de uma regra de aprendizado genérica. No aprendizado indutivo, uma regra é induzida a partir de dados disponíveis *a priori*. Mais tarde, esta regra é utilizada para novas amostras disponíveis apenas *a posteriori*. Se considerarmos que o objetivo de solução do problema é a classificação de pontos ainda sem rótulo, a criação de uma regra de aprendizado é a alternativa lógica para resolver o problema quando não há informação adicional sobre estes pontos *a priori*. No aprendizado transdutivo, por outro lado, todos os pontos de interesse devem estar disponíveis *a priori*, o que automaticamente implica que uma solução transdutiva não precisa generalizar para outros pontos a não ser aqueles inicialmente propostos.

Apesar de não serem unânimes, estas definições são geralmente reconhecidas pela maioria da comunidade de teoria do aprendizado. Baseado nelas, é fácil ver que transdutividade e aprendizado semi-supervisionado não podem ser diretamente comparados. Além disso, podemos constatar que pode haver aprendizado semi-supervisionado indutivo assim como aprendizado semi-supervisionado transdutivo. A diferença entre os dois está exatamente na imposição do primeiro em criar uma regra genérica para ser utilizada na avaliação de pontos desconhecidos, enquanto o segundo está apenas interessado na estimativa imediata de pontos de interesse.

A motivação para se utilizar métodos de aprendizado semi-supervisionado, indutivos ou transdutivos, está na sua habilidade de incorporar toda a informação disponível *a priori* para aumentar a acurácia da solução. A informação contida na disposição dos

pontos sem rótulo pode fornecer uma melhor estimativa da distribuição de probabilidades original do problema, o que levaria o modelo a ter uma acurácia superior e portanto melhores soluções.

Note que no aprendizado transdutivo semi-supervisionado, deve-se explicitamente estimar o valor dos pontos de interesse. No aprendizado indutivo semi-supervisionado, por outro lado, a saída do problema é uma regra de aprendizado capaz de avaliar quaisquer amostras no espaço do problema. Isto significa que modelos de aprendizado indutivos semi-supervisionados não necessariamente precisam estimar o valor dos pontos de interesse. Ao invés disso, assim que a regra genérica é induzida com o auxílio das informações adicionais fornecidas pelos pontos sem rótulo, esta regra é utilizada para classificar estes mesmos pontos. Esta estratégia de dois passos é adotada pela maioria dos métodos indutivos semi-supervisionados da literatura.

4.2 Uma Abordagem Essencialmente Transdutiva para o Problema de Aprendizado

Muitas vezes encontramos referências na literatura que, de acordo com as definições dadas na Seção 4.1, confundem aprendizado indutivo semi-supervisionado por aprendizado transdutivo, geralmente porque descrevem métodos que utilizam dados sem rótulos para aumentar sua acurácia. Alguns destes métodos, na verdade, se baseiam em máquinas de aprendizado indutivo, arranjadas em torno de um problema de meta-otimização onde cada passo em direção à solução é dado após a indução de uma regra genérica completa. Por exemplo, muitas máquinas de vetores de suporte transdutivas são construídas em torno de SVMs indutivas tradicionais, como descrito na Seção 3.2.

Neste trabalho, como já citado neste capítulo, reconhecemos que há uma distinção

clara entre aprendizado indutivo semi-supervisionado e aprendizado transdutivo semi-supervisionado, que é exatamente o requisito para a indução de uma regra genérica de aprendizado ao invés da simples avaliação de pontos de interesse. Esta é a mesma definição dada por Vapnik [Vap98], que sugere, como descrito na Seção 2.2, que resolver um problema de aprendizado transdutivo é mais fácil que resolver um problema indutivo, uma vez que não precisamos induzir uma regra genérica para todo o espaço do problema. De acordo com o princípio da Navalha de Occam [Tho18], Vapnik [Vap95] também diz que:

Se há uma quantidade limitada de informações disponível, não resolva um problema particular através da solução de um problema genérico.

Por fim, Vapnik [Vap98] explicitamente argumenta contra a solução de um problema de classificação onde os pontos de interesse já são conhecidos através desta estratégia de dois passos, indução e dedução. Ele cita que a solução de um problema relativamente simples de se estimar um conjunto finito de números (i.e. os valores desconhecidos) torna-se dependente de uma solução substancialmente mais complexa que é estimar uma função, o que implica estimar todos os valores no contínuo de pontos que contém os pontos de interesse.

Note que esta complexidade adicional é manifestada de duas formas. Primeiro, que quando uma quantidade pequena ou insuficiente de dados é utilizada para inferir informações de ordem mais alta, como a indução de uma regra genérica a partir de pontos no espaço, o modelo final alcançado pode não descrever bem o contexto do problema. Segundo, que devido à complexidade dos espaços de busca envolvidos e à complexidade dos próprios métodos, aproximações destes espaços ou a utilização heurísticas que envolvem compromissos de qualidade da resposta devem frequentemente ser utilizadas para tornar a solução do problema computacionalmente possível.

A fim de claramente distinguir nossa terminologia entre aprendizado transdutivo e aprendizado indutivo semi-supervisionado, adotaremos o termo *aprendizado essencialmente transdutivo* para denotar o aprendizado transdutivo semi-supervisionado.

4.3 Estimativas Baseadas em Teoria da Informação

Como base da proposta de nosso novo método de classificação transdutivo, adotaremos alguns princípios da estatística e da teoria da informação. Iniciaremos nossa discussão pelo princípio da Informação Discriminante Mínima (MDI, *Minimum Discrimination Information*), que diz que a divergência de Kullback-Leibler entre duas distribuições de probabilidades pode ser utilizadas como informação de discriminação a partir da introdução de novos fatos. Considere um problema de classificação binária com classes A e B , onde as distribuições de probabilidade associadas são P_A e P_B , respectivamente. Suponha que P_A e P_B são conhecidas. Para cada nova amostra $\mathbf{x}_i$ a ser classificada, podemos tentar incorporar $\mathbf{x}_i$ em cada uma das classes e termos $A' = \{A \cup \mathbf{x}_i\}$ e $B' = \{B \cup \mathbf{x}_i\}$. Em sequência, podemos calcular $D_{KL}(A||A')$ e $D_{KL}(B||B')$, incorporando permanentemente a nova amostra à classe que teve a menor perturbação, ou seja, que apresentou a menor divergência em relação à sua versão estendida. Formalmente, podemos utilizar a divergência de Kullback-Leibler para classificar novas amostras da seguinte forma:

$$classe(\mathbf{x}_i) = \begin{cases} A & \text{if } D_{KL}(A||\{A \cup \mathbf{x}_i\}) < D_{KL}(B||\{B \cup \mathbf{x}_i\}), \\ B & \text{if } D_{KL}(A||\{A \cup \mathbf{x}_i\}) > D_{KL}(B||\{B \cup \mathbf{x}_i\}), \\ \text{indeterminado} & \text{if } D_{KL}(A||\{A \cup \mathbf{x}_i\}) = D_{KL}(B||\{B \cup \mathbf{x}_i\}), \end{cases} \quad (4.1)$$

onde os valores indeterminados obtidos na equação podem ser polarizados para A ou B .

A fim de resolver o problema de uma forma computacionalmente tratável, propomos que o espaço de busca seja contínuo, não discreto. Desta forma, podemos utilizar

diferentes técnicas de otimização contínua, geralmente mais eficientes que técnicas de otimização discreta. Assim, ao invés de determinar discretamente se cada nova amostra está associada à classe A ou à classe B , introduzimos um vetor real de pesos contidos no intervalo $[0, 1]$. Em um problema de classificação de k amostras, a solução fica reduzida à busca do ponto de mínimo risco no espaço $[0, 1]^k$. A solução final do problema pode então ser descrita de forma similar à Equação (4.1):

$$classe(\mathbf{x}_i) = \begin{cases} A & \text{if } \phi(\mathbf{x}_i) > 0.5, \\ B & \text{if } \phi(\mathbf{x}_i) < 0.5, \\ \text{indeterminado} & \text{if } \phi(\mathbf{x}_i) = 0.5, \end{cases} \quad (4.2)$$

onde $\phi(\mathbf{x}_i)$ é o peso associado à amostra i , e os valores indeterminados podem ser polarizados para A ou B .

Em seguida, definimos S como um vetor de pesos contidos no intervalo $[0, 1]$. S contem $(\ell + k)$ elementos, ou seja, um peso associado a cada uma das amostras do conjunto de treinamento, assim como um peso associado a cada uma das amostras do conjunto de trabalho. Os valores associados ao conjunto de treinamento são fixos em 0 e 1, de acordo com a classe das amostras correspondentes. Os valores associados ao conjunto de trabalho serão calculados pelo método, e serão a resposta final do problema.

Em seguida, introduzimos $\bar{S}$, o complemento de S . $\bar{S}$ tem os mesmos elementos de S , mas cada peso é igual ao seu complemento em S , ou seja, $(1 - \phi(\mathbf{x}_i))$. Agora, de acordo com o princípio MDI, podemos variar os pesos do conjunto de trabalho e avaliar a divergência entre S e seu complemento $\bar{S}$, onde quanto maior a divergência, melhor a solução final. Ou seja, a solução do problema essencialmente transdutivo se resume ao vetor de pesos que maximiza $D_{KL}(S \parallel \bar{S})$.

4.4 Divergência de Kullback-Leibler

A fim de calcular $D_{KL}(S\|\bar{S})$, utilizaremos uma propriedade interessante da divergência de Kullback-Leibler. Se assumirmos um formato conhecido para as distribuições de probabilidade avaliadas pela divergência de Kullback-Leibler diferencial, obtemos uma forma fechada de representação.

O formato assumido para as distribuições de probabilidade é o de uma distribuição Gaussiana multivariável. Esta premissa é justificável com base em duas premissas da estatística. A primeira, pelo método da entropia máxima, que mostra que a função de distribuição de probabilidades que produz a maior entropia é a Gaussiana [LG94]. A segunda, pelo método da máxima verossimilhança, que mostra que as estimativas com máxima verossimilhança para a média e a matriz de covariância de uma distribuição Gaussiana multivariável são iguais à média e matriz de covariância amostrais [HAH01].

A fim de incorporar os pesos de S e $\bar{S}$ para efetivamente calcular $D_{KL}(S\|\bar{S})$, uma vez que seu formato foi definido como uma Gaussiana multivariável, precisamos das médias e variâncias das distribuições para caracterizá-las. Assim, utilizamos uma matriz de covariância ponderada, similar às matrizes de covariância da teoria de portfólios financeiros, calculada a partir dos vetores de peso e dos próprios pontos das amostras dos conjuntos de treinamento e trabalho. De forma análoga, utilizamos a média ponderada dos mesmos elementos para calcular o centro das distribuições.

Por fim, mostramos a forma fechada da divergência de Kullback-Leibler para distribuições Gaussianas multivariáveis:

$$D_{KL}(P\|Q) = \frac{1}{2} \left(\log \left(\frac{|\Sigma_Q|}{|\Sigma_P|} \right) + \text{tr}(\Sigma_Q^{-1}\Sigma_P) + (\mu_Q - \mu_P)^T \Sigma_Q^{-1} (\mu_Q - \mu_P) - N \right). \quad (4.3)$$

Note que, para a solução do problema, será necessário inverter uma das matrizes ponderadas de covariância e calcular o determinante de ambas as matrizes, operações

que podem comprometer a computabilidade do método. Além de serem computacionalmente caras, estas operações podem adicionar excessivos erros numéricos às soluções, comprometendo a qualidade da resposta.

A fim de resolver estes problemas, assumimos que há independência entre as dimensões dos dados, ou seja, que as matrizes de covariância são diagonais. Esta premissa facilita diversas operações, como descrito a seguir.

Primeiro, o cálculo de inversão das matrizes se resume à inversão individual de cada elemento original da diagonal principal, ou seja:

$$A^{-1} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix} \quad (4.4)$$

Segundo, que o traço de uma matriz diagonal A é simplesmente a soma dos termos de sua diagonal principal:

$$tr(A) = tr(diag(A)) = \sum_{i=1}^n a_{ii}. \quad (4.5)$$

Terceiro, que o determinante de uma matriz diagonal é o produto dos termos de sua diagonal principal:

$$|A| = |diag(A)| = \prod_{i=1}^n a_{ii}. \quad (4.6)$$

Por fim, substituímos estas identidades na Equação (4.3), e obtemos a seguinte expressão:

$$D_{KL}(P||Q) = \frac{1}{2} \left(\log \frac{\prod_{i=1}^n q_{ii}}{\prod_{i=1}^n p_{ii}} + \frac{\sum_{i=1}^n p_{ii}}{\sum_{i=1}^n q_{ii}} + \sum_{i=1}^n \frac{(\mu_Q - \mu_P)^2}{q_{ii}} - n \right). \quad (4.7)$$

A fim de evitar possíveis problemas numéricos, utilizamos uma identidade logarítmica, $\log \prod_{i=1}^n a_i = \sum_{i=1}^n \log a_i$, e chegamos à seguinte expressão:

$$D_{KL}(P\|Q) = \frac{1}{2} \left(\sum_{i=1}^n \log \frac{q_{ii}}{p_{ii}} + \frac{\sum_{i=1}^n p_{ii}}{\sum_{i=1}^n q_{ii}} + \sum_{i=1}^n \frac{(\mu_Q - \mu_P)^2}{q_{ii}} - n \right). \quad (4.8)$$

Note que esta expressão, além de muito mais simples de ser avaliada em relação à forma fechada original da divergência de Kullback-Leibler, também traz o benefício adicional de ser consideravelmente mais fácil para ser diferenciada.

4.5 O Problema de Aprendizado como um Problema de Otimização com Restrições

Utilizando todas as definições descritas neste capítulo, finalmente chegamos à formulação do método de classificação essencialmente transdutivo baseado em entropia (ETEC). Considere um problema de classificação binário, onde temos um conjunto de treinamento com ℓ amostras $\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$, $\theta \in (X, Y)$, um conjunto de trabalho com k amostras $\tau = \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}$, $\tau \in X$, onde X é o espaço do problema, e Y é o conjunto de possíveis classificações (ou rótulos). O objetivo do problema é estimar os valores $y_{\ell+1}, \dots, y_{\ell+k}$ nos pontos de interesse. Uma vez que o problema é de classificação binária, assume-se que $Y = \{-1, +1\}$, o que implica que cada estimativa $y_{\ell+1}, \dots, y_{\ell+k}$ será feita dentro do conjunto $\{-1, +1\}$.

Em seguida, colocamos o problema como um problema de otimização com restrições, onde o valor dos pesos $\alpha_{\ell+1}, \dots, \alpha_{\ell+k}$ que resultam no menor valor de $-Dkl(S\|\bar{S})$ produz a resposta. Os pesos $\alpha_{\ell+1}, \dots, \alpha_{\ell+k}$ podem variar apenas dentro do intervalo $[0, 1]$, as únicas restrições necessárias. A resposta final do problema, isto é, as classificações para cada amostra do conjunto de trabalho, são determinadas para cada um

dos dois possíveis rótulos de acordo com seus pesos correspondentes serem maiores ou menores que 0.5, isto é:

$$\arg \min_{\{\alpha_{\ell+1}, \dots, \alpha_{\ell+k}\}} -D_{KL}(S\|\bar{S}),$$

$$y_i = \begin{cases} +1 & \text{if } \alpha_i \geq 0.5 \\ -1 & \text{if } \alpha_i < 0.5 \end{cases}, \quad \forall (\ell+1) \leq i \leq (\ell+k) \quad (4.9)$$

$$0 \leq \alpha_i \leq 1,$$

onde explicitamente polarizamos a resposta para casos onde pesos são iguais a 0.5.

Em seguida, definimos $D_{KL}(S\|\bar{S})$, que é resultado da expansão da Equação (4.8):

$$\begin{aligned} D_{KL}(S\|\bar{S}) = & 0.5 \left(\sum_{i=1}^n \log \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2}{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2} \right. \\ & - \sum_{i=1}^n \log \frac{\left(\ell+k - \sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} x_{j,i}^2 - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} x_{j,i} - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2}{\left(\ell+k - \sum_{j=1}^{\ell+k} \alpha_j \right)^2} \\ & + \sum_{i=1}^n \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\frac{\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}}{\sum_{j=1}^{\ell+k} \alpha_j} - \frac{\sum_{j=1}^{\ell+k} x_{j,i} - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i}}{\ell+k - \sum_{j=1}^{\ell+k} \alpha_j} \right)^2}{\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2} \\ & + \sum_{i=1}^n \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\ell+k - \sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} x_{j,i}^2 - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right)}{\left(\ell+k - \sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2 \right)} \\ & \left. - \sum_{i=1}^n \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\sum_{j=1}^{\ell+k} x_{j,i} - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2}{\left(\ell+k - \sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2 \right)} - n \right) \quad (4.10) \end{aligned}$$

onde $x_{j,i}$ denota o valor da j -ésima dimensão do i -ésimo vetor $\mathbf{x}$ no conjunto de dados.

É interessante notar que, em contraste com outros métodos da literatura, o método proposto não tem nenhum parâmetro ajustável, o que aumenta sua aplicabilidade e facilidade de uso.

A fim de resolver o problema de otimização com restrições, utilizamos um método Quasi-Newton de programação quadrática sequencial. As derivadas de primeira ordem da Equação (4.10) foram calculadas em relação a uma amostra genérica e foram explicitamente utilizadas pelo método. O algoritmo utilizado durante a otimização foi fornecido pelo *Optimization Toolbox* do MATLAB, versão 7.4.0 (R2007a).

4.6 Indicadores

Uma vez que propusemos um método baseado em conceitos da teoria da informação, talvez seja possível utilizar outras medidas de informação para inferir propriedades de problemas de aprendizado. Se um dado problema de aprendizado contem muita informação de treinamento, mas pouca informação de trabalho, é provável que um método transdutivo não vá desempenhar melhor que um método indutivo. Por outro lado, se há dados escassos de treinamento, a informação contida no conjunto de trabalho pode ser utilizada por um método transdutivo para aumentar a acurácia da resposta. Portanto, estamos interessados em avaliar as quantidades de informação contidas em cada um destes conjuntos em um problema de aprendizado transdutivo.

Neste capítulo, introduzimos 8 indicadores baseados na teoria da informação. Uma vez que adotamos a forma contínua de medidas de entropia, utilizamos como unidade o *nat*. Ainda, utilizamos como notação a expressão $h(A)$ para denotar a entropia diferencial de uma distribuição de probabilidades (assumida Gaussiana multivariável) por trás de um conjunto de dados A . Para as seções a seguir, assumimos um problema

de aprendizado transdutivo onde há um conjunto de treinamento dividido em dois sub-conjuntos A e B , cada um representando uma classe, e há um conjunto de trabalho T .

4.6.1 Indicadores de Primeira Ordem: Entropia de Conjuntos

Indicadores de primeira ordem são dados pela entropia dos conjuntos imediatamente disponíveis em um problema de aprendizado. Os 3 indicadores de primeira ordem são, portanto:

$h(A)$ – Entropia do conjunto de amostras da classes A no conjunto de treinamento;

$h(B)$ – Entropia do conjunto de amostras da classes B no conjunto de treinamento;

$h(T)$ – Entropia do conjunto de amostras do conjunto de trabalho.

4.6.2 Indicadores de Segunda Ordem: Entropia de Conjuntos Compostos

Indicadores de segunda ordem são definidos pela entropia da união de dois ou mais conjuntos. Os 4 indicadores de segunda ordem são:

$h(A + B)$ – Entropia de todo o conjunto de treinamento;

$h(A + T)$ – Entropia do conjunto de amostras da classes A combinado com o conjunto de trabalho;

$h(B + T)$ – Entropia do conjunto de amostras da classes B combinado com o conjunto de trabalho;

$h(A + B + T)$ – Entropia do conjunto de treinamento combinado com o conjunto de trabalho.

Estas combinações são úteis para avaliar se a informação contida em dois conjuntos é similar ou não. Note que os 3 primeiros indicadores variam sempre que a proporção entre dados de treinamento e validação mudam. O último indicador é fixo para quaisquer distribuições de dados entre os conjuntos, medindo a informação total contida nos dados do problema.

4.6.3 Indicadores de Terceira Ordem: Medida de Transdutividade

Por fim, propomos um indicador de terceira ordem, calculado não diretamente a partir dos dados, mas a partir da combinação de dois outros indicadores. Este indicador pode ser considerado como a medida de transdutividade do problema, sendo definido como:

$$\frac{h(T)}{h(A + B)} - 1. \quad (4.11)$$

Note que como o indicador é definido a partir da proporção entre duas medidas de entropia em nats, ele não tem unidade, sendo portanto exibido como um valor percentual.

Quando este indicador é positivo, ele mostra que o problema contém mais informações em seu conjunto de trabalho que em seu conjunto de treinamento. Esta é uma indicação que o problema pode ser favorecido por um método transdutivo. De forma análoga, se o indicador é negativo, é provável que um método indutivo também tenha um bom desempenho.

Este indicador não deve ser considerado como uma medida direta da aplicabilidade de métodos transdutivos. Primeiro, ele não avalia medidas de informação suficientes em relação aos conjuntos de treinamento e trabalho. Por exemplo, se há um

problema com conjuntos de treinamento e trabalho igualmente dispersos, o valor do indicador será próximo de zero se ambos conjuntos forem distantes um do outro no espaço do problema. Segundo, há um certo nível de aproximação envolvido nas estimativas de indicadores baseados em entropia diferencial, onde algumas das propriedades dos conjuntos de dados podem passar despercebidas.

Esta medida de transdutividade deve ser utilizada como uma referência, ao contrário de uma medida direta, da aplicabilidade de métodos transdutivos. Além disso, ela deve ser preferencialmente analisada juntamente com outros indicadores de primeira e segunda ordem, que exibem informações adicionais sobre um dado problema.

Capítulo 5

Experimentos e Resultados

Este capítulo traz os experimentos e resultados obtidos neste trabalho.

5.1 Bases de Dados

Todos os experimentos deste trabalho foram baseados em bases de dados publicamente disponíveis. Foram selecionadas 7 bases largamente utilizadas para testes de métodos de classificação. Todas estão disponíveis no *UCI Repository of Machine Learning* [BM98]. As bases selecionadas foram **bcw**, **hepatitis**, **ionosphere**, **musk**, **pgs**, **pidand** e **wpbc**. Seu tamanho e dimensões são descritos na Tabela 5.1.

Todas as bases de dados têm saídas em $\{-1, +1\}$ e entradas em $[-\infty, +\infty]$, portanto satisfazendo os requisitos para os experimentos de classificação binária. Algumas bases tinham dimensões arbitrárias (não numéricas) que foram convertidas para o domínio $\{-1, +1\}$.

Tabela 5.1: Bases de dados e suas partições utilizadas em experimentos.

conjunto	dimensões	número de amostras
bcw	9	699
hepatitis	19	155
ionosphere	34	351
musk	166	476
pgs	114	106
pid	8	768
wpbc	33	198

5.2 Amostragem de Dados

Todos os experimentos conduzidos neste trabalho foram repetidos 10 vezes a fim de se evitar aberrações estatísticas. Para cada experimento, os conjuntos de treinamento e trabalho foram amostrados das bases de dados originais, descritas na Seção 5.1, de acordo com dois critérios. Estes critérios, a distribuição de probabilidades das quais os dados são amostrados e a proporção entre dados de treinamento e trabalho, são descritos nas seções a seguir.

5.2.1 Amostragem Polarizada e Não Polarizada

Como descrito na Seção 2.1, uma das premissas sobre a definição geral do problema de aprendizado é que os dados são amostrados i.i.d. em relação a uma data distribuição, mesmo que esta seja desconhecida. A fim de testar o comportamento do método proposto

e de métodos indutivos quando esta premissa é violada, todos os experimentos foram repetidos com duas configurações:

Distribuições Não Polarizadas Onde todos os dados foram amostrados i.i.d. em relação a uma distribuição de probabilidades uniforme fixa.

Distribuições Polarizadas Onde todos os dados foram amostrados de uma distribuição unimodal centrada em uma primeira amostra escolhida aleatoriamente, com pesos de cada amostra proporcionais à sua distância do centro.

Os resultados de todos os experimentos são relatados para ambas as configurações, possibilitando a comparação do efeito da estratégia de amostragem sobre cada método testado.

5.2.2 Tamanho dos Conjuntos de Treinamento e Trabalho

Os tamanhos dos conjuntos de treinamento e trabalho são muito importantes para definir o comportamento de modelos de aprendizado indutivos e transdutivos. Quanto mais dados um método de aprendizado recebe em seu conjunto de treinamento, mais informações ele terá sobre as distribuições de probabilidade do problema. Em problemas de aprendizado supervisionado, esta é toda a informação que os métodos terão para produzir uma solução final. Em problemas de aprendizado semi-supervisionado, entretanto, o conjunto de trabalho também aumenta as informações conhecidas pelo método sobre estas distribuições.

Além disso, o tamanho do conjunto de trabalho não influencia o comportamento de métodos indutivos (em aprendizado indutivo, o conjunto de trabalho é chamado conjunto de validação, ou conjunto de testes). Este não é o caso dos métodos transdutivos, onde o número de amostras no conjunto de trabalho é igual ao número de respostas

que devem ser dadas como solução do problema. Em outras palavras, apesar de quanto maior for o conjunto de trabalho, maior será quantidade de informações disponível para o método, isto também implica que será cada vez mais difícil resolver o problema.

Neste trabalho, todos os experimentos foram feitos com 9 diferentes proporções entre o tamanho do conjunto de treinamento e o tamanho do conjunto de trabalho. Iniciamos nossos experimentos com 10% de todos os dados das bases originais assinalados ao conjunto de treinamento, e os 90% restantes assinalados ao conjunto de trabalho. Gradualmente invertemos as proporções em passos de 10%, até que 90% estejam no conjunto de treinamento e os 10% restantes no conjunto de trabalho.

5.3 Métodos de Referência

Nesta seção iniciamos nossos experimentos com as bases de dados descritas na Seção 5.1. Antes dos experimentos com o método proposto no Capítulo 4, descreveremos os experimentos para as mesmas bases de dados com outros dois algoritmos de aprendizado bem conhecidos:

SVM Máquina de vetores de suporte indutiva tradicional. O algoritmo de treinamento utilizado foi o SVM^{light} , introduzido em [Joa98]. A implementação foi disponibilizada pelo próprio autor, versão 6.01, escrita em C.

TSVM Máquina de vetores de suporte transdutiva. O algoritmo de treinamento utilizado também foi o SVM^{light} , que incorpora um algoritmo transdutivo iterativo conhecido como TSVM, como descrito em [Joa99]. A implementação foi disponibilizada pelo próprio autor, versão 6.01, escrita em C.

Note que ao invés de utilizar um algoritmo de treinamento de máquinas de vetores de suporte exclusivamente indutivo, como o SMO, selecionamos o mesmo algoritmo de

treinamento para ambos as máquinas, indutiva (SVM) e transdutiva (TSVM). Assim, procuramos evitar discrepâncias entre SVM e TSVM ocasionadas pela diferença de estratégia e desempenho de cada algoritmo de treinamento.

5.4 Ajuste de Parâmetros

Ao contrário do método sem parâmetros proposto no Capítulo 4, ambos SVM e TSVM dependem de parâmetros específicos para cada problema de aprendizado. Mais especificamente, eles precisam de um parâmetro de flexibilização da margem C , que regula o viés entre erro de treinamento e generalização, e parâmetros correspondentes à forma e tamanho da função de kernel.

Para selecionar os parâmetros de cada base de dados, foram executados testes exaustivos com uma SVM indutiva tradicional. Todas bases de dados foram reduzidas a 100 amostras, em seguida divididas com 30% de dados treinamento e 70% de dados de validação. As SVMs foram treinadas e avaliadas com todas as combinações possíveis de parâmetros da função de kernel e C . Cada combinação foi testada 10 vezes, cada uma com amostragens diferentes dos conjuntos de treinamento e testes.

O parâmetro de controle da margem C foi escolhido do conjunto $\{10^{-3}, 10^0, 10^3\}$. O formato da função de kernel foi escolhido entre *linear*, *polinomial* ou *função de base radial* (RBF). Cada parâmetro de tamanho das funções de kernel foi selecionado do mesmo conjunto $\{10^{-3}, 10^0, 10^3\}$.

Após 10 execuções com cada combinação, a configuração de parâmetros que resultou na melhor acurácia para cada base de dados foi selecionada. Os resultados, descritos na Tabela 5.2, foram utilizados em todos os experimentos de referência descritos a seguir. O procedimento de ajuste de parâmetros utilizado é descrito em maior detalhe em

[MBdC07].

Tabela 5.2: Resultados do procedimento de ajuste de parâmetros, utilizados nos experimentos com SVM e TSVM.

conjunto	C	kernel	σ
bcw	0.001	linear	-
hepatitis	1000	rbf	1
ionosphere	1	rbf	1
musk	1000	rbf	1000
pgs	1000	linear	-
pid	0.001	linear	-
wpbc	1000	rbf	1

5.5 Incorporando Informações Adicionais

Além dos dados contidos nos conjuntos de treinamento e trabalho de cada experimento, foram incorporadas informações adicionais a estes conjuntos através da inclusão de 4 novas dimensões. Este procedimento pode ser interpretado como sendo parte do próprio método transdutivo, uma vez que seu cálculo, baseado em teoria da informação, depende da disponibilidade do conjunto de trabalho.

Considere um problema de aprendizado transdutivo onde o conjunto de treinamento é dividido em dois conjuntos A e B , cada um representando uma classe, e existe um conjunto de trabalho não classificado T . Neste passo de pré-processamento, se os dados dos conjuntos das bases originais têm n dimensões, cada uma das amostras receberá 4 novas características, portanto resultando em dados com $(n + 4)$ dimensões.

Para cada conjunto A , B e T , estas dimensões têm diferentes definições. Para o conjunto parcial de treinamento A , as dimensões adicionadas a cada amostra $x \in A$ foram:

Dimensão $(n + 1)$, $x \in A$: $H(A)$ (entropia de A)

Dimensão $(n + 2)$, $x \in A$: $H(B \cup x)$ (entropia de B' estendido (com $x \in B'$))

Dimensão $(n + 3)$, $x \in A$: $H(A) - H(B \cup x)$

Dimensão $(n + 4)$, $x \in A$: $H(A)/H(B \cup x)$

Para amostras na parte restante do conjunto de treinamento, B :

Dimensão $(n + 1)$, $x \in B$: $H(B)$ (entropia de B)

Dimensão $(n + 2)$, $x \in B$: $H(A \cup x)$ (entropia de A' estendido (com $x \in A'$))

Dimensão $(n + 3)$, $x \in B$: $H(A \cup x) - H(B)$

Dimensão $(n + 4)$, $x \in B$: $H(A \cup x)/H(B)$

Para amostras no conjunto de trabalho T :

Dimensão $(n + 1)$, $x \in T$: $H(A \cup x)$ (entropia de A' estendido (com $x \in A'$))

Dimensão $(n + 2)$, $x \in T$: $H(B \cup x)$ (entropia de B' estendido (com $x \in B'$))

Dimensão $(n + 3)$, $x \in T$: $H(A \cup x) - H(B \cup x)$

Dimensão $(n + 4)$, $x \in T$: $H(A \cup x)/H(B \cup x)$

Para amostras no conjunto de treinamento, estas novas características medem o efeito na entropia de se ter cada amostra pertencendo à classe oposta. Para amostras no conjunto de trabalho, elas medem a perturbação causada em cada classe quando cada amostra é individualmente incorporada ao conjunto.

5.6 Resultados de Experimentos

As próximas seções mostram os resultados experimentais do método proposto com cada uma das bases de dados utilizadas nestes trabalho.

5.6.1 Resultados para bcw

Os resultados do método proposto (ETEC) para a base de dados **bcw**, com amostragem polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.3 e 5.4.

Tabela 5.3: Resultados de ETEC para **bcw**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	95.27% $\pm$ 1.50%
20%	80%	96.36% $\pm$ 0.82%
30%	70%	96.75% $\pm$ 0.48%
40%	60%	96.62% $\pm$ 0.74%
50%	50%	96.43% $\pm$ 0.90%
60%	40%	96.57% $\pm$ 0.91%
70%	30%	96.48% $\pm$ 1.01%
80%	20%	95.43% $\pm$ 2.43%
90%	10%	89.57% $\pm$ 8.17%

Tabela 5.4: Resultados de ETEC para **bcw**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	95.65% $\pm$ 1.52%
20%	80%	96.25% $\pm$ 0.73%
30%	70%	96.49% $\pm$ 0.53%
40%	60%	96.55% $\pm$ 0.56%
50%	50%	96.63% $\pm$ 0.84%
60%	40%	96.57% $\pm$ 0.54%
70%	30%	95.71% $\pm$ 1.65%
80%	20%	92.64% $\pm$ 5.10%
90%	10%	87.00% $\pm$ 3.95%

5.6.2 Resultados para hepatitis

Os resultados do método proposto (ETEC) para a base de dados **hepatitis**, com amostragens polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.5 e 5.6.

Tabela 5.5: Resultados de ETEC para **hepatitis**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	79.21% $\pm$ 2.74%
20%	80%	79.35% $\pm$ 3.14%
30%	70%	77.22% $\pm$ 3.41%
40%	60%	73.91% $\pm$ 4.38%
50%	50%	68.69% $\pm$ 10.56%
60%	40%	63.33% $\pm$ 10.95%
70%	30%	59.32% $\pm$ 13.40%
80%	20%	47.86% $\pm$ 18.15%
90%	10%	45.83% $\pm$ 16.31%

Tabela 5.6: Resultados de ETEC para **hepatitis**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	77.71% $\pm$ 6.08%
20%	80%	79.52% $\pm$ 3.02%
30%	70%	77.13% $\pm$ 4.17%
40%	60%	68.37% $\pm$ 19.55%
50%	50%	73.29% $\pm$ 6.80%
60%	40%	59.83% $\pm$ 15.66%
70%	30%	57.73% $\pm$ 12.13%
80%	20%	46.79% $\pm$ 16.19%
90%	10%	41.67% $\pm$ 22.22%

5.6.3 Resultados para ionosphere

Os resultados do método proposto (ETEC) para a base de dados **ionosphere**, com amostragem polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.7 e 5.8.

Tabela 5.7: Resultados de ETEC para **ionosphere**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	79.05% $\pm$ 7.11%
20%	80%	83.54% $\pm$ 6.16%
30%	70%	85.62% $\pm$ 4.99%
40%	60%	86.01% $\pm$ 5.38%
50%	50%	84.19% $\pm$ 7.31%
60%	40%	87.94% $\pm$ 4.08%
70%	30%	87.40% $\pm$ 5.13%
80%	20%	75.78% $\pm$ 17.09%
90%	10%	73.93% $\pm$ 16.33%

Tabela 5.8: Resultados de ETEC para **ionosphere**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	84.68% $\pm$ 4.71%
20%	80%	84.79% $\pm$ 4.46%
30%	70%	87.87% $\pm$ 2.84%
40%	60%	87.59% $\pm$ 3.39%
50%	50%	89.82% $\pm$ 2.14%
60%	40%	90.66% $\pm$ 2.73%
70%	30%	88.20% $\pm$ 7.04%
80%	20%	79.84% $\pm$ 13.57%
90%	10%	59.29% $\pm$ 20.57%

5.6.4 Resultados para musk

Os resultados do método proposto (ETEC) para a base de dados **musk**, com amostragens polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.9 e 5.10.

Tabela 5.9: Resultados de ETEC para **musk**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	57.23% $\pm$ 5.62%
20%	80%	56.85% $\pm$ 5.58%
30%	70%	63.27% $\pm$ 5.54%
40%	60%	67.44% $\pm$ 6.45%
50%	50%	69.11% $\pm$ 4.98%
60%	40%	73.49% $\pm$ 5.18%
70%	30%	71.63% $\pm$ 5.74%
80%	20%	69.35% $\pm$ 7.45%
90%	10%	69.11% $\pm$ 10.65%

Tabela 5.10: Resultados de ETEC para **musk**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	56.32% $\pm$ 6.96%
20%	80%	57.17% $\pm$ 4.06%
30%	70%	64.17% $\pm$ 5.52%
40%	60%	67.79% $\pm$ 8.92%
50%	50%	68.73% $\pm$ 9.64%
60%	40%	72.33% $\pm$ 7.30%
70%	30%	72.34% $\pm$ 7.82%
80%	20%	69.46% $\pm$ 6.09%
90%	10%	64.45% $\pm$ 8.64%

5.6.5 Resultados para pgs

Os resultados do método proposto (ETEC) para a base de dados **pgs**, com amostragem polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.11 e 5.12.

Tabela 5.11: Resultados de ETEC para **pgs**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	50.31% $\pm$ 4.42%
20%	80%	60.71% $\pm$ 13.37%
30%	70%	72.57% $\pm$ 16.17%
40%	60%	76.51% $\pm$ 9.84%
50%	50%	74.62% $\pm$ 9.92%
60%	40%	73.17% $\pm$ 9.13%
70%	30%	60.00% $\pm$ 11.11%
80%	20%	61.58% $\pm$ 14.48%
90%	10%	70.00% $\pm$ 22.97%

Tabela 5.12: Resultados de ETEC para **pgs**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	48.75% $\pm$ 1.61%
20%	80%	61.06% $\pm$ 11.69%
30%	70%	62.16% $\pm$ 9.81%
40%	60%	68.09% $\pm$ 9.80%
50%	50%	65.58% $\pm$ 10.78%
60%	40%	60.49% $\pm$ 11.77%
70%	30%	54.67% $\pm$ 13.53%
80%	20%	56.31% $\pm$ 8.96%
90%	10%	67.50% $\pm$ 18.82%

5.6.6 Resultados para **pid**

Os resultados do método proposto (ETEC) para a base de dados **pid**, com amostragens polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.13 e 5.14.

Tabela 5.13: Resultados de ETEC para **pid**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	72.31% $\pm$ 2.30%
20%	80%	70.62% $\pm$ 6.38%
30%	70%	73.12% $\pm$ 2.13%
40%	60%	73.27% $\pm$ 2.63%
50%	50%	75.00% $\pm$ 2.59%
60%	40%	71.89% $\pm$ 3.29%
70%	30%	69.39% $\pm$ 3.44%
80%	20%	69.08% $\pm$ 4.46%
90%	10%	68.82% $\pm$ 7.05%

Tabela 5.14: Resultados de ETEC para **pid**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	71.82% $\pm$ 3.16%
20%	80%	72.06% $\pm$ 4.25%
30%	70%	70.98% $\pm$ 9.41%
40%	60%	72.56% $\pm$ 6.79%
50%	50%	72.34% $\pm$ 3.53%
60%	40%	70.69% $\pm$ 4.06%
70%	30%	65.65% $\pm$ 3.30%
80%	20%	66.08% $\pm$ 3.87%
90%	10%	66.45% $\pm$ 4.53%

5.6.7 Resultados para wpbc

Os resultados do método proposto (ETEC) para a base de dados **wpbc**, com amostragem polarizada e não polarizada, são respectivamente descritos nas Tabelas 5.15 e 5.16.

Tabela 5.15: Resultados de ETEC para **wpbc**, distribuição polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	70.06% $\pm$ 6.45%
20%	80%	65.53% $\pm$ 5.73%
30%	70%	64.68% $\pm$ 9.39%
40%	60%	65.46% $\pm$ 7.21%
50%	50%	65.96% $\pm$ 4.74%
60%	40%	69.62% $\pm$ 4.70%
70%	30%	73.56% $\pm$ 5.60%
80%	20%	73.08% $\pm$ 5.57%
90%	10%	74.21% $\pm$ 10.36%

Tabela 5.16: Resultados de ETEC para **wpbc**, distribuição não polarizada.

Conjunto de treinamento (relativo a todos os dados)	Conjunto de trabalho (relativo a todos os dados)	Acurácia média (10 amostragens distintas)
10%	90%	56.14% $\pm$ 14.40%
20%	80%	57.23% $\pm$ 9.71%
30%	70%	60.58% $\pm$ 7.24%
40%	60%	64.62% $\pm$ 10.12%
50%	50%	64.14% $\pm$ 8.15%
60%	40%	69.37% $\pm$ 8.29%
70%	30%	69.32% $\pm$ 5.27%
80%	20%	77.69% $\pm$ 4.37%
90%	10%	74.74% $\pm$ 8.15%

5.7 Comparação Gráfica

Nesta seção exibimos todos os resultados obtidos neste trabalho de forma gráfica. Os resultados de experimentos com métodos de referência foram integrados aos resultados de experimentos com o método proposto (ETEC), todos exibidos nos mesmos gráficos para que seu desempenho fosse comparado. Além disso, os indicadores de informação para todas as bases de dados também são exibidos logo abaixo de cada gráfico de desempenho para cada uma das variações entre os tamanhos dos conjuntos de treinamento e trabalho.

Nos gráficos dos indicadores, note que os 3 indicadores de primeira ordem e os 4 indicadores de segunda ordem têm a mesma unidade, o *nat*. O indicador de terceira ordem, que mede a transdutividade do problema, é uma proporção mostrada como um percentual, portanto sem unidade. A fim de simplificar a demonstração, este indicador

é exibido no mesmo gráfico dos outro 7 sobre um segundo eixo y que foi omitido para preservar o espaço das figuras.

Ao longo das próximas 14 páginas, exibimos as Figuras de 5.1 a 5.14, que trazem os resultados e indicadores para todos os experimentos conduzidos com cada uma das 8 bases de dados e com as 2 estratégias de amostradas utilizadas.

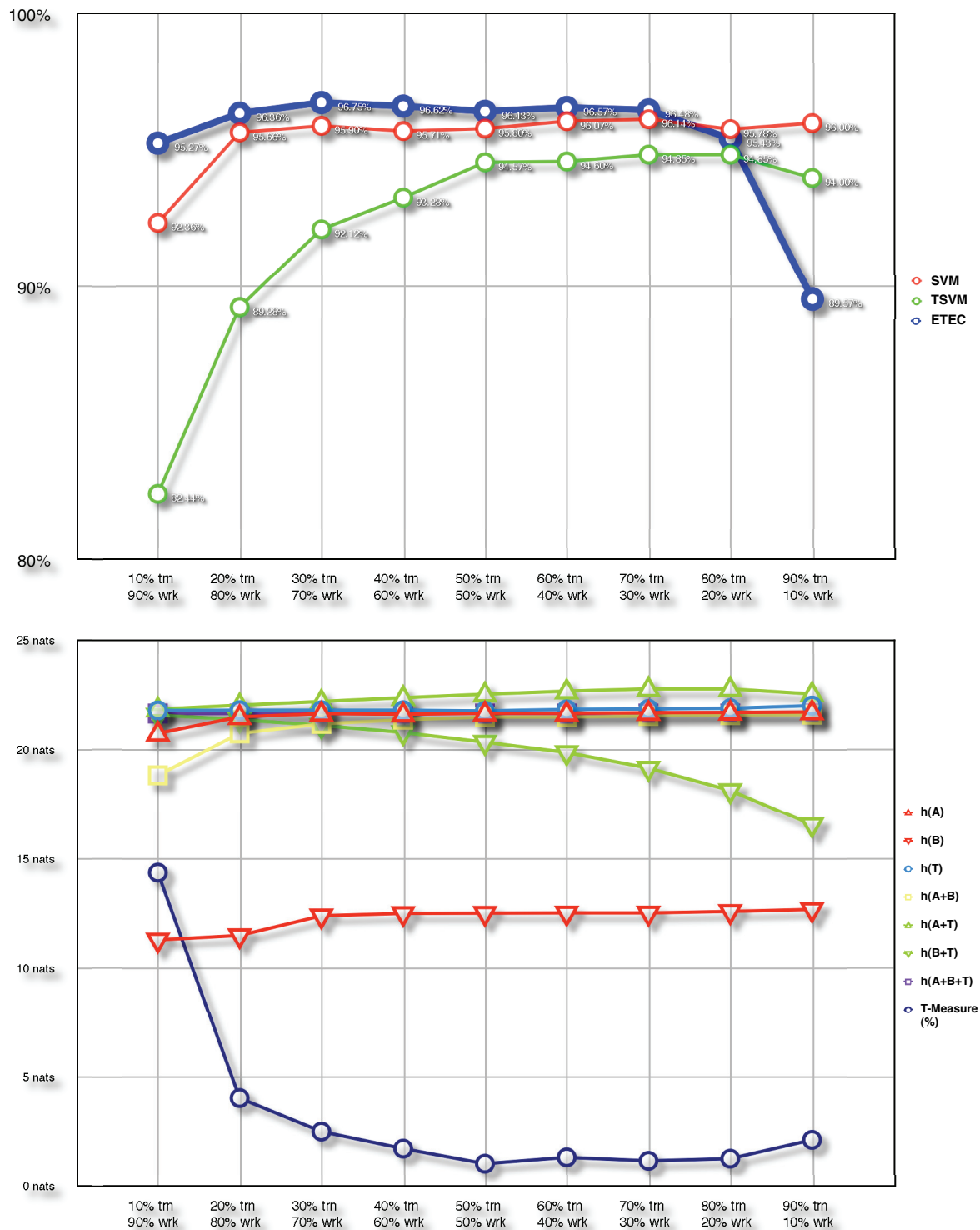


Figura 5.1: Resultados gráficos para **bcw**, amostragem polarizada.

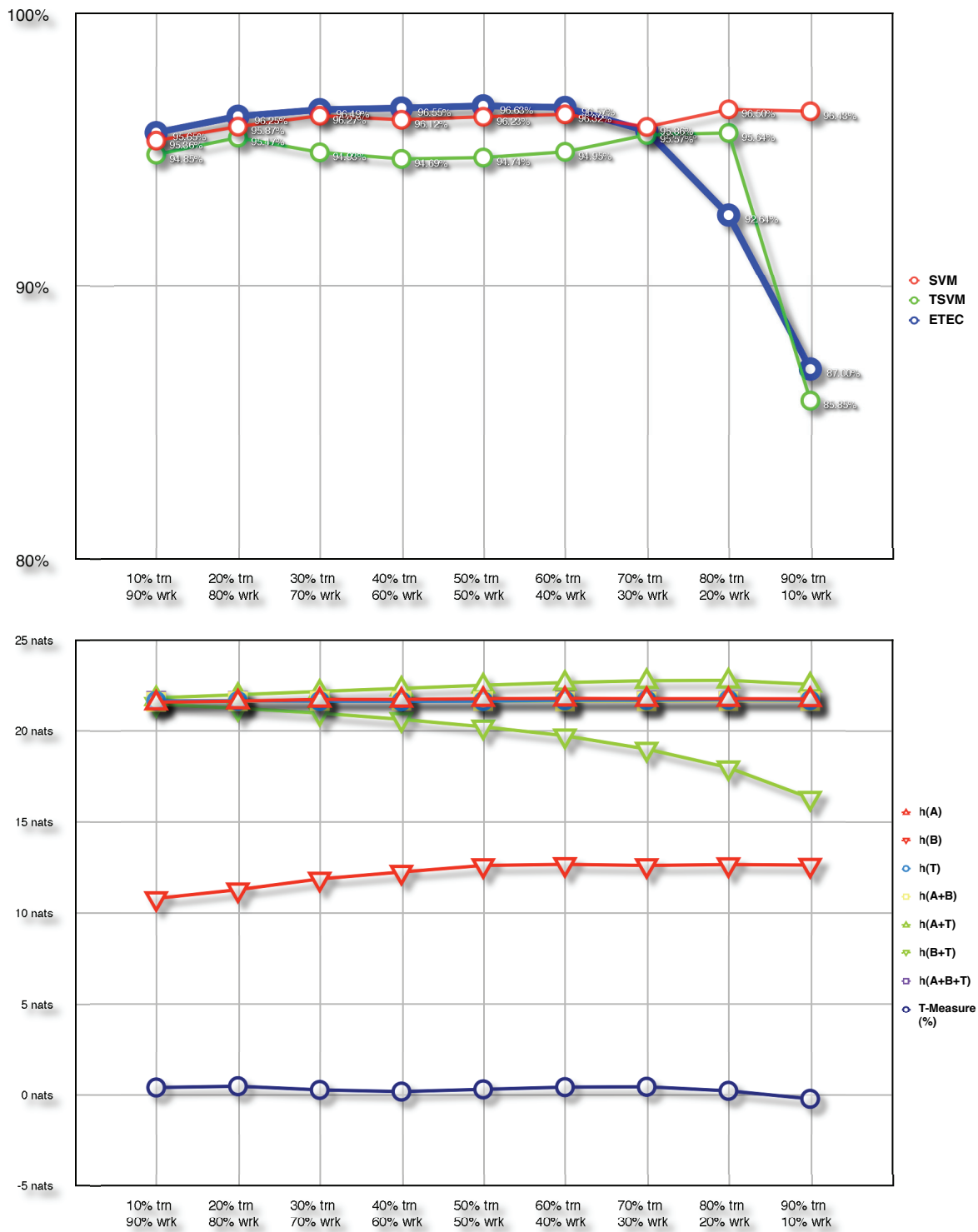


Figura 5.2: Resultados gráficos para **bcw**, amostragem não polarizada.

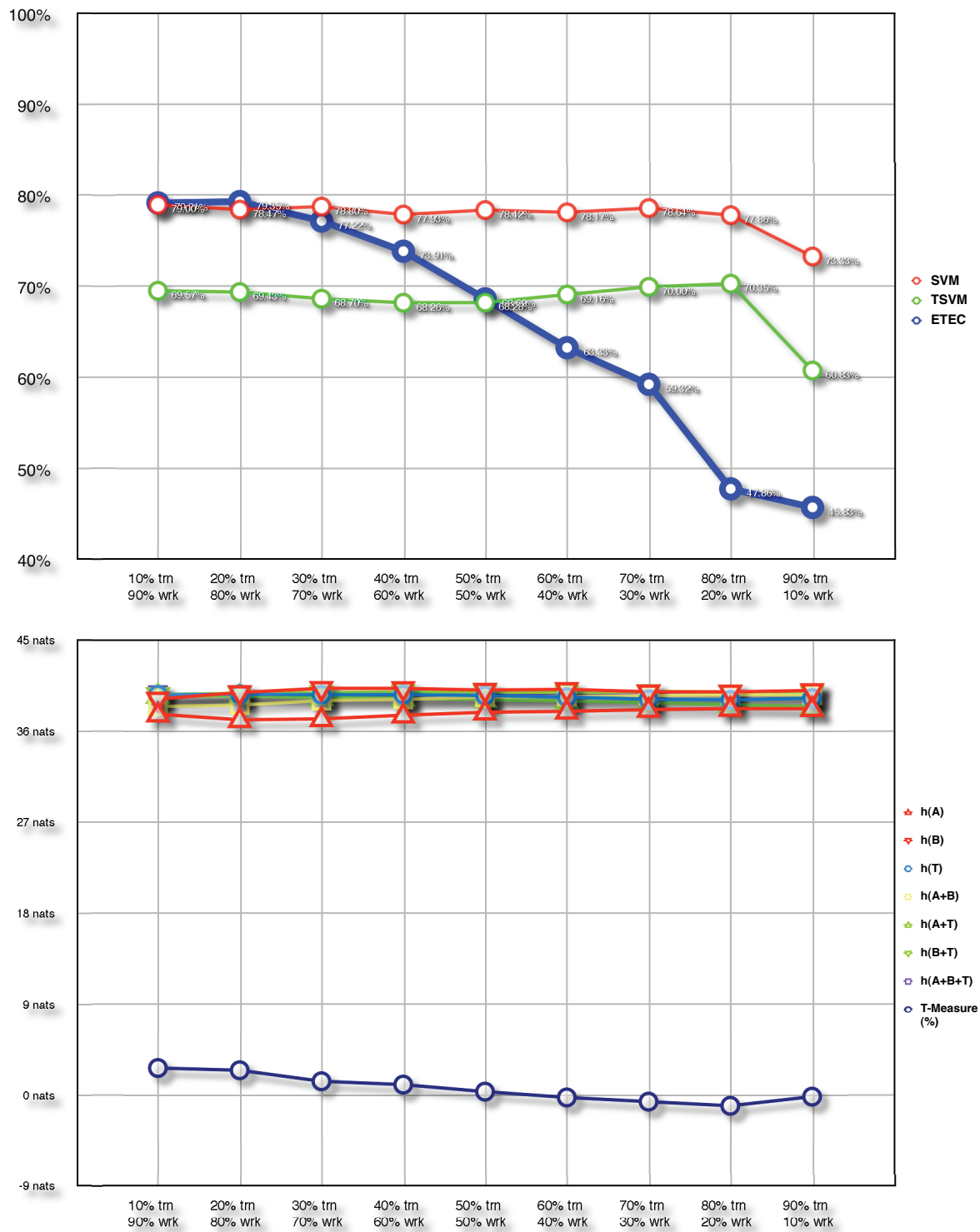


Figura 5.3: Resultados gráficos para **hepatitis**, amostragem polarizada.

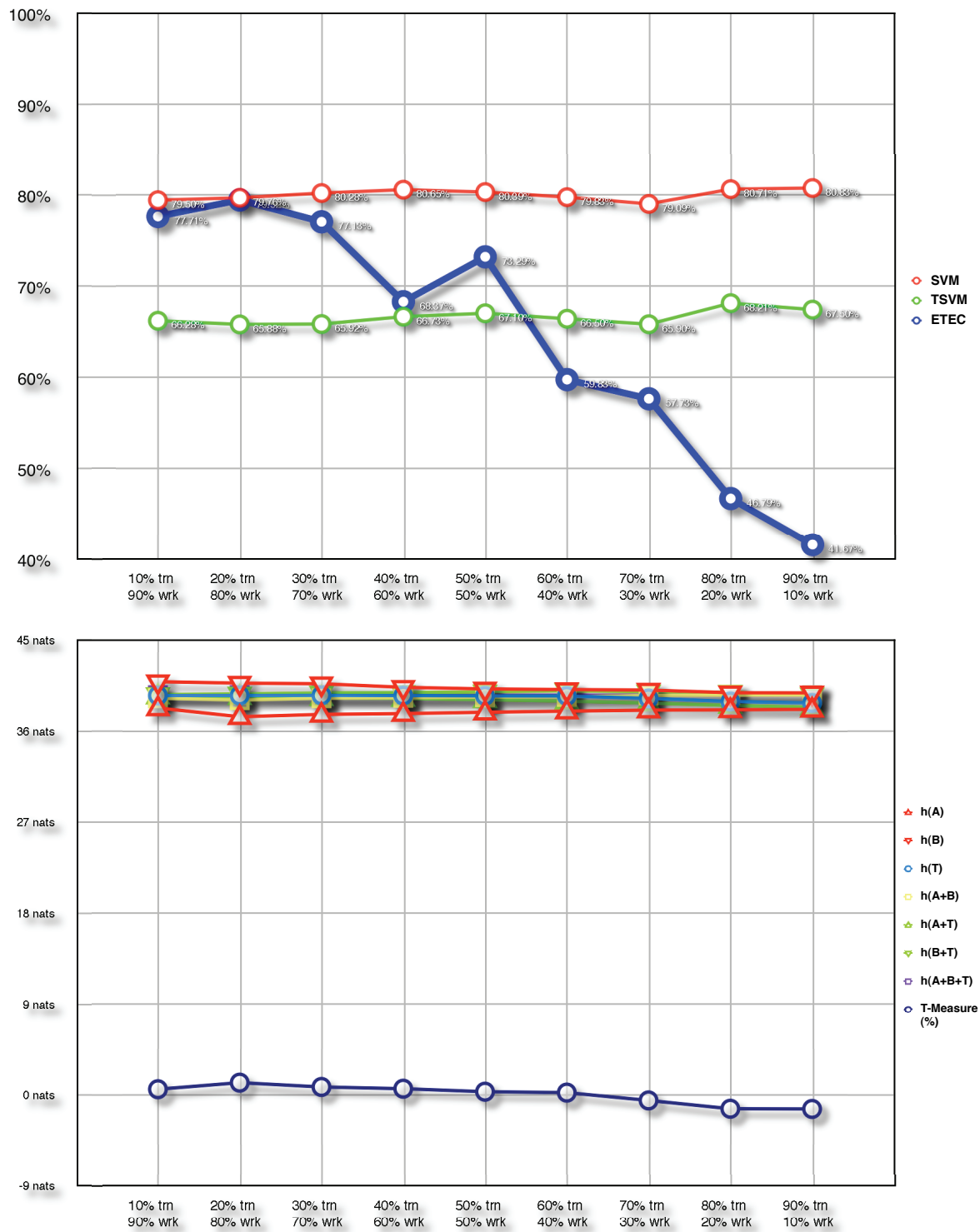


Figura 5.4: Resultados gráficos para **hepatitis**, amostragem não polarizada.

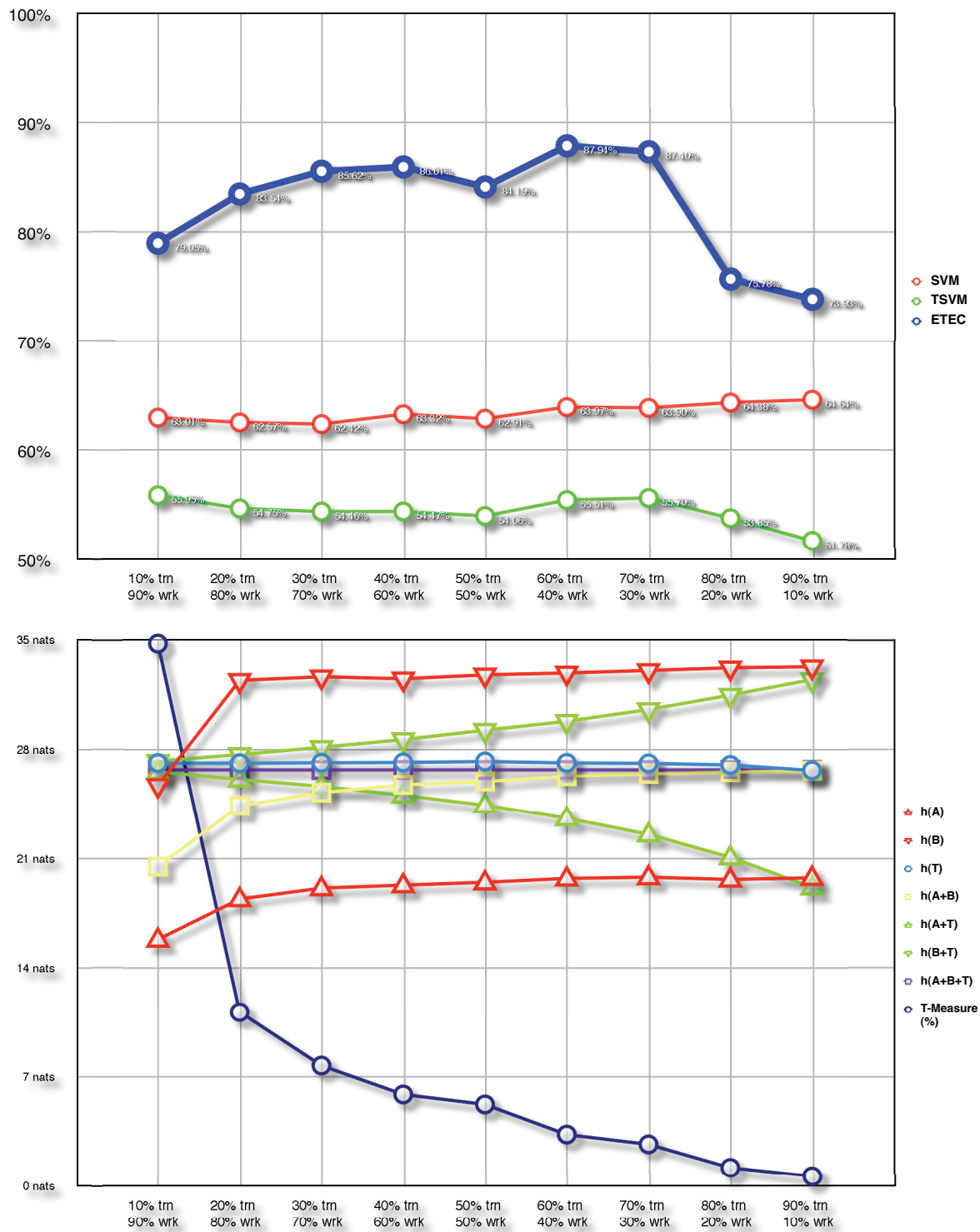


Figura 5.5: Resultados gráficos para **ionosphere**, amostragem polarizada.

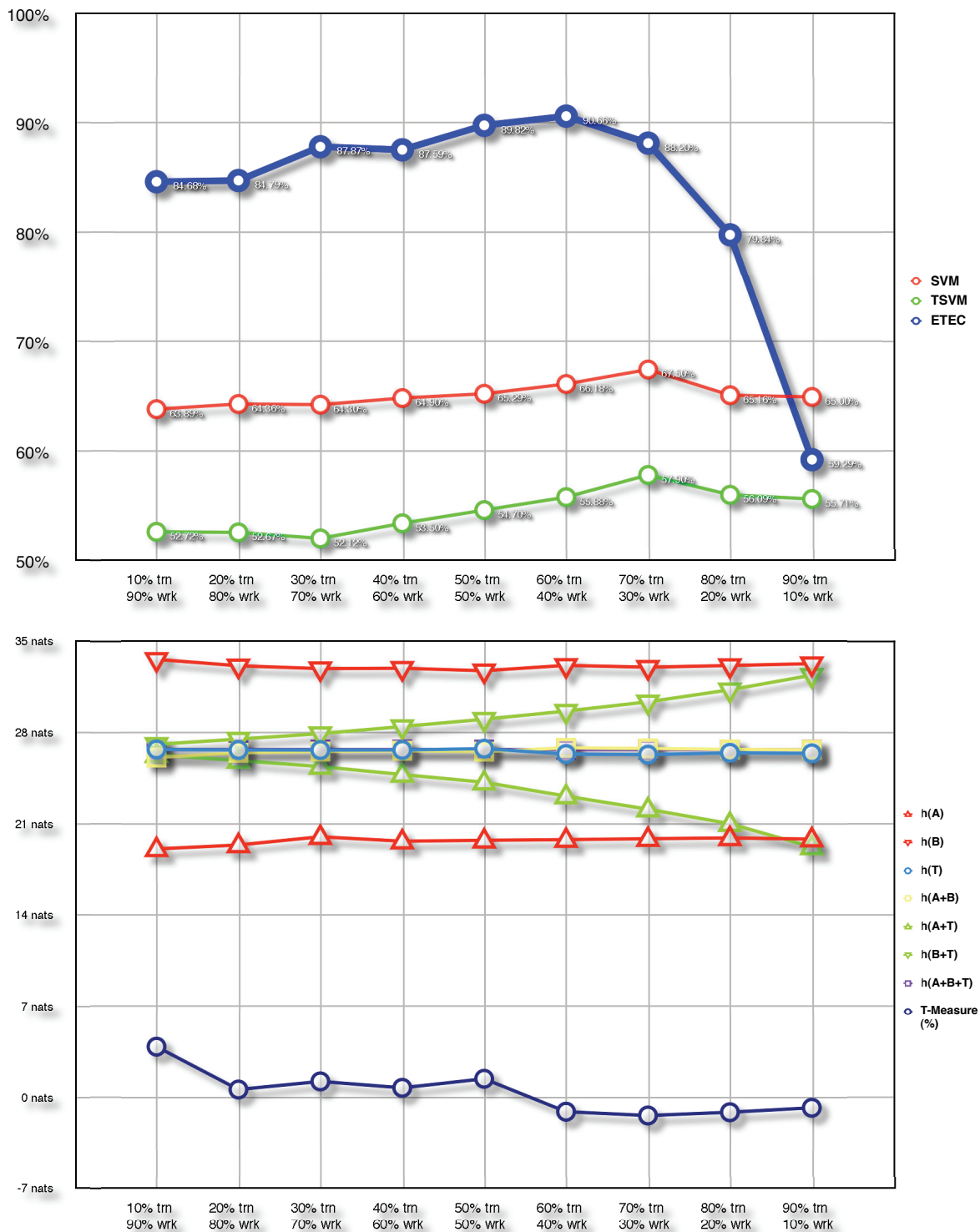


Figura 5.6: Resultados gráficos para **ionosphere**, amostragem não polarizada.

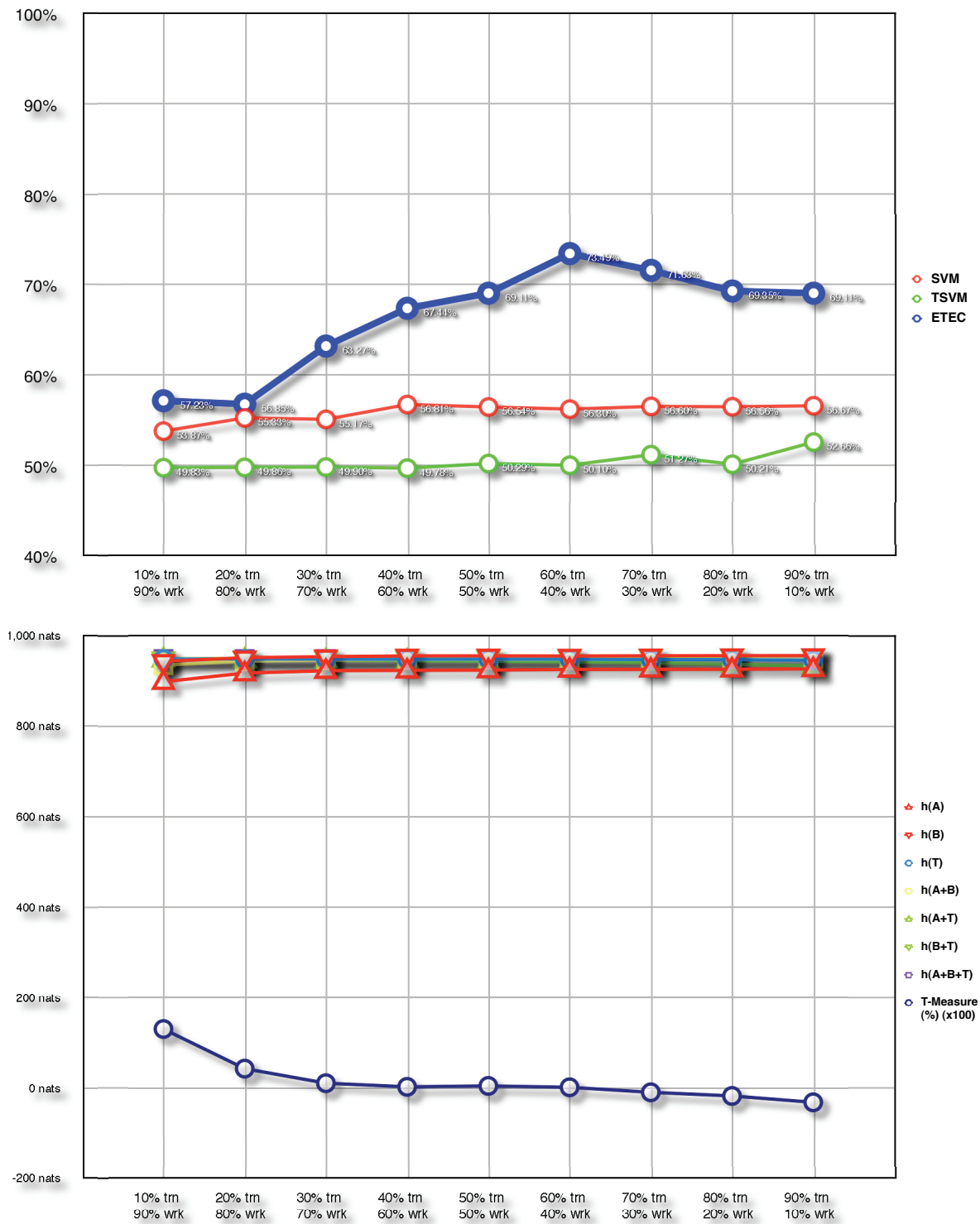


Figura 5.7: Resultados gráficos para **mus**k, amostragem polarizada.

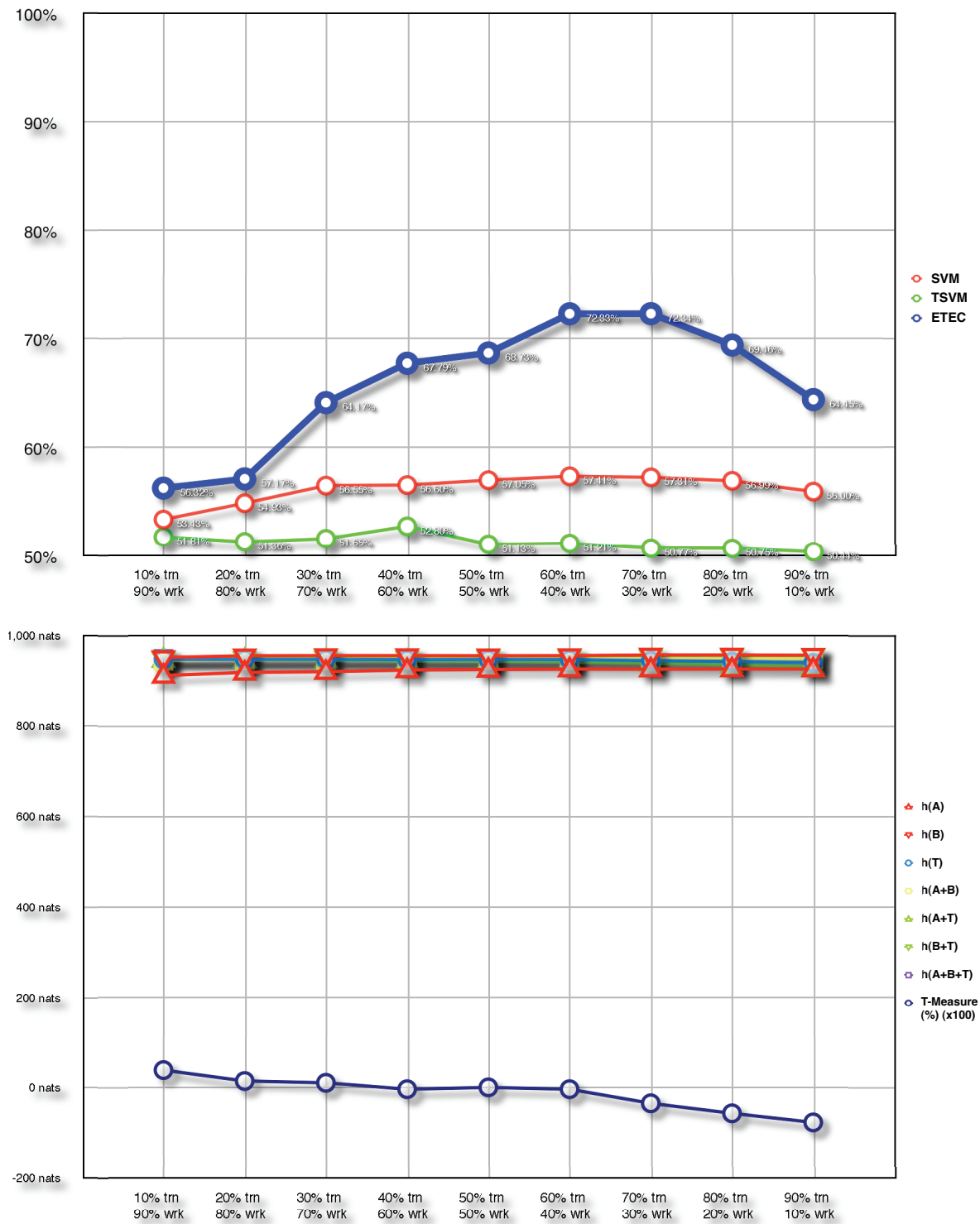


Figura 5.8: Resultados gráficos para **musk**, amostragem não polarizada.

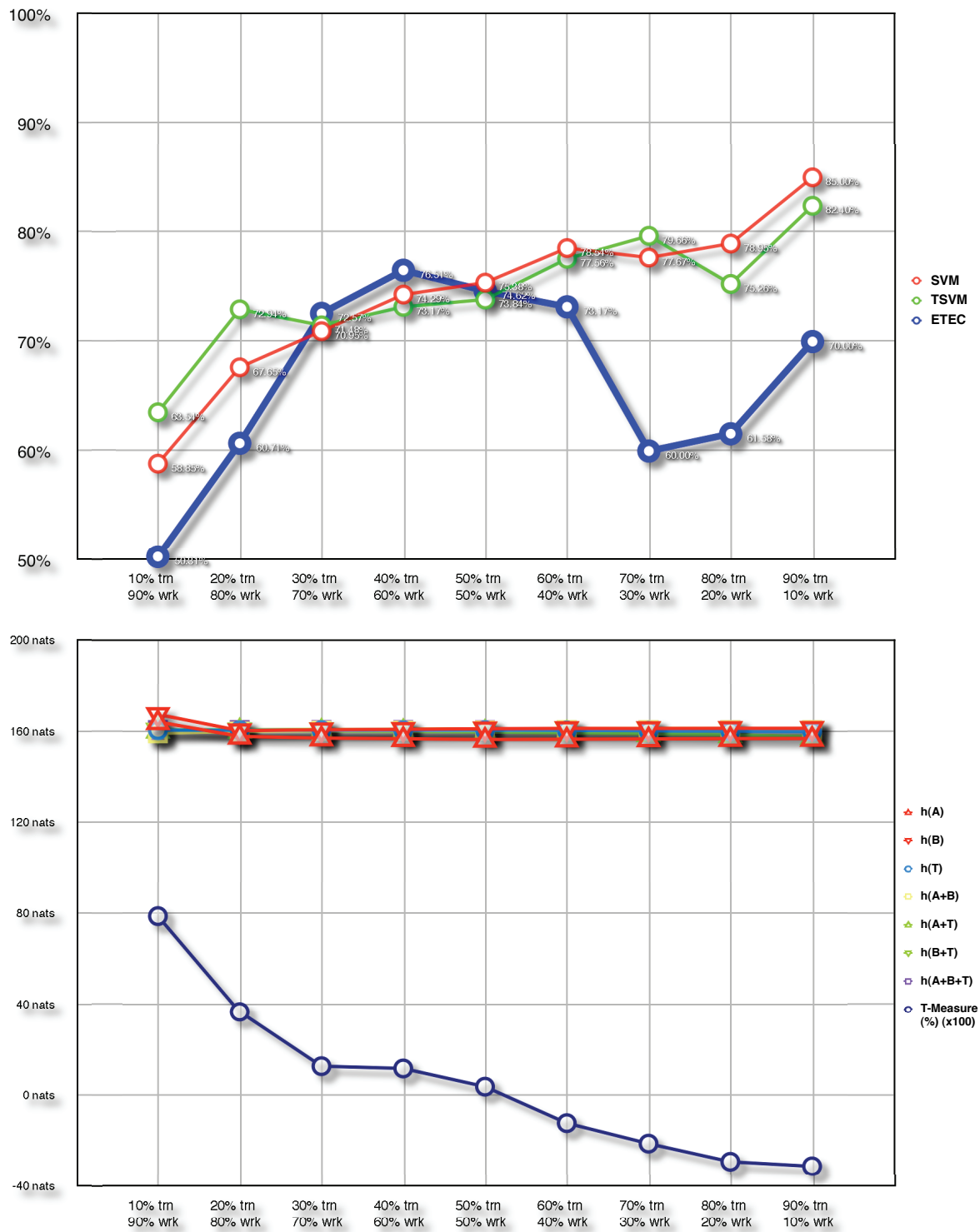


Figura 5.9: Resultados gráficos para **pgs**, amostragem polarizada.

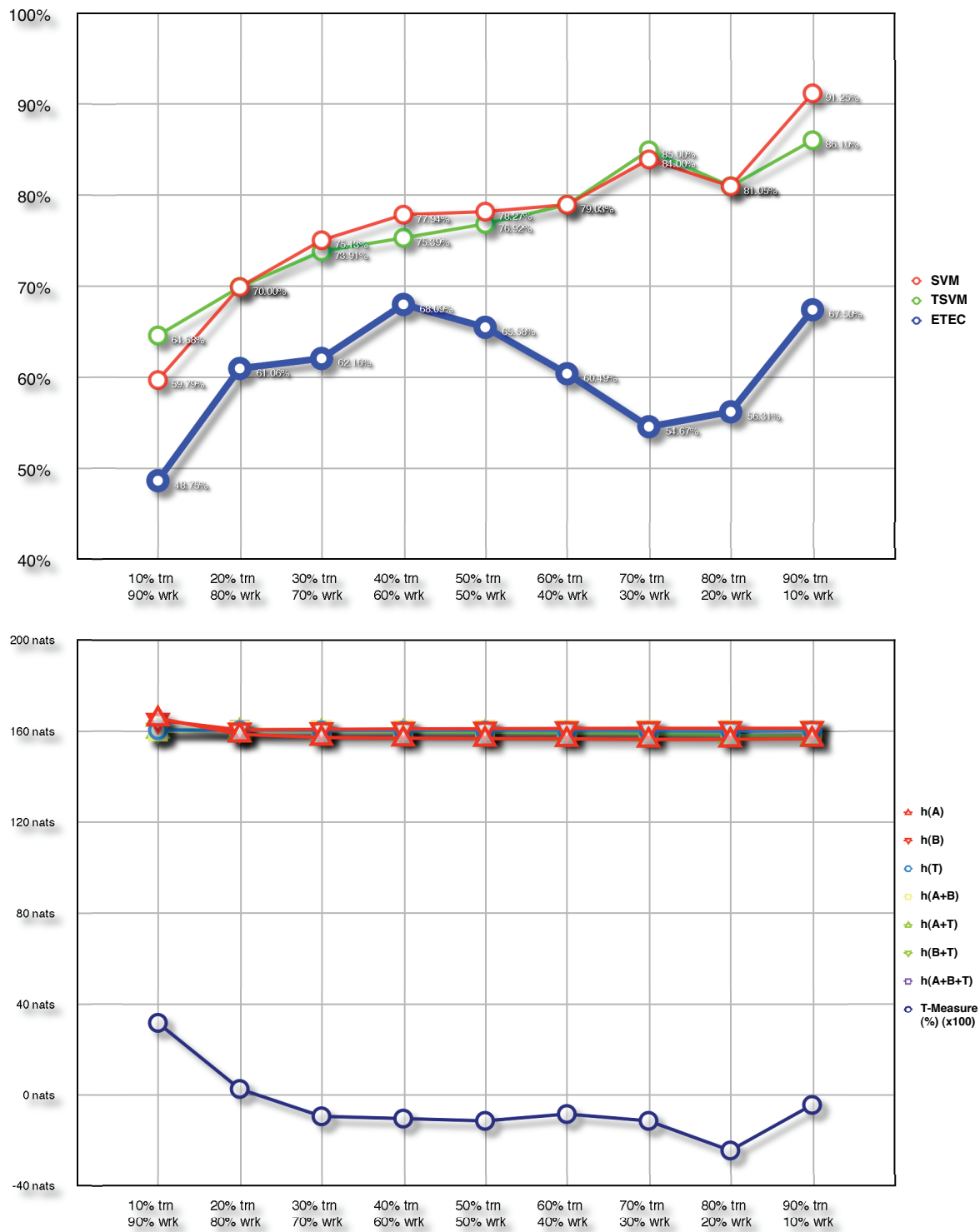


Figura 5.10: Resultados gráficos para **pgs**, amostragem não polarizada.

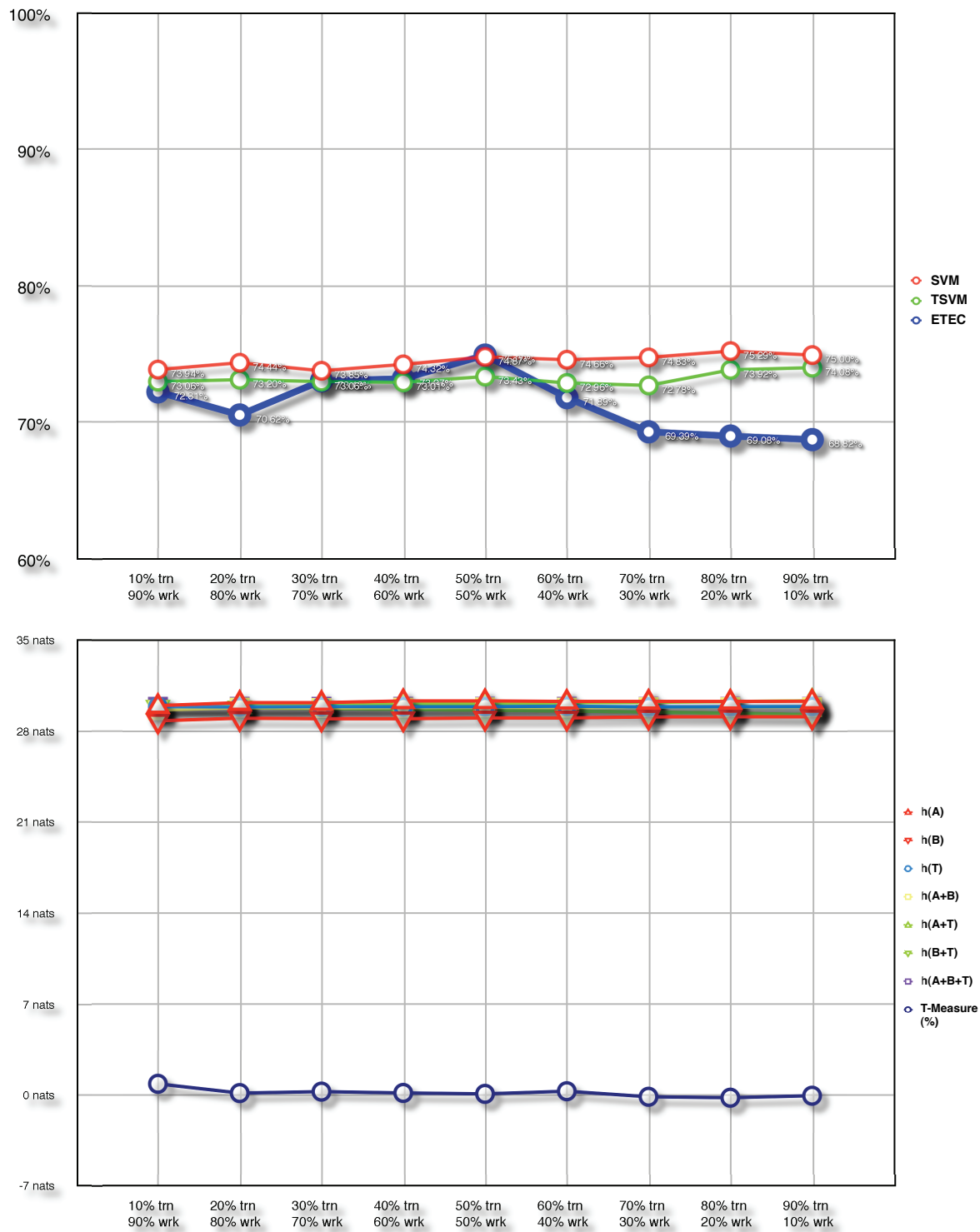


Figura 5.11: Resultados gráficos para **pid**, amostragem polarizada.

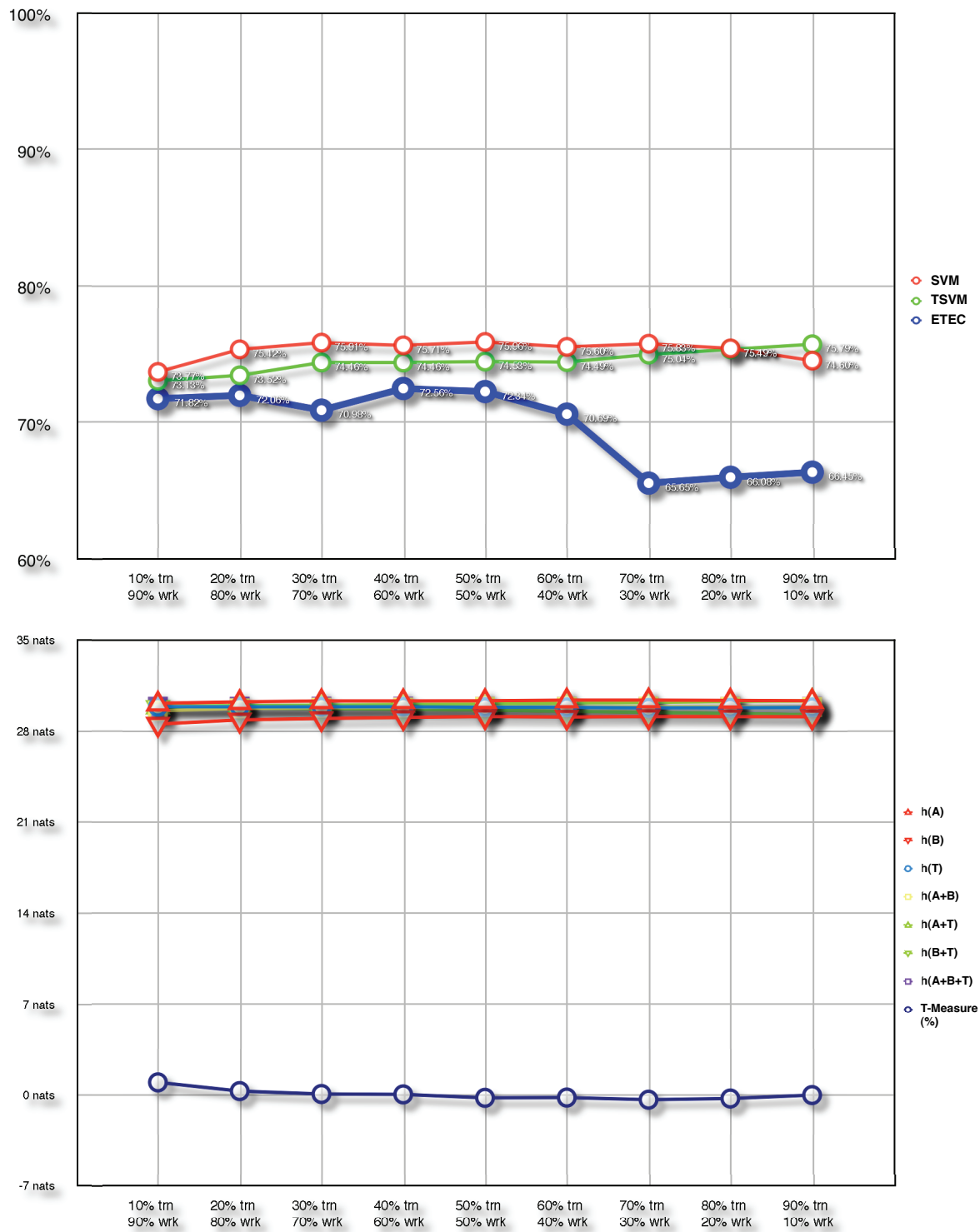


Figura 5.12: Resultados gráficos para **pid**, amostragem não polarizada.

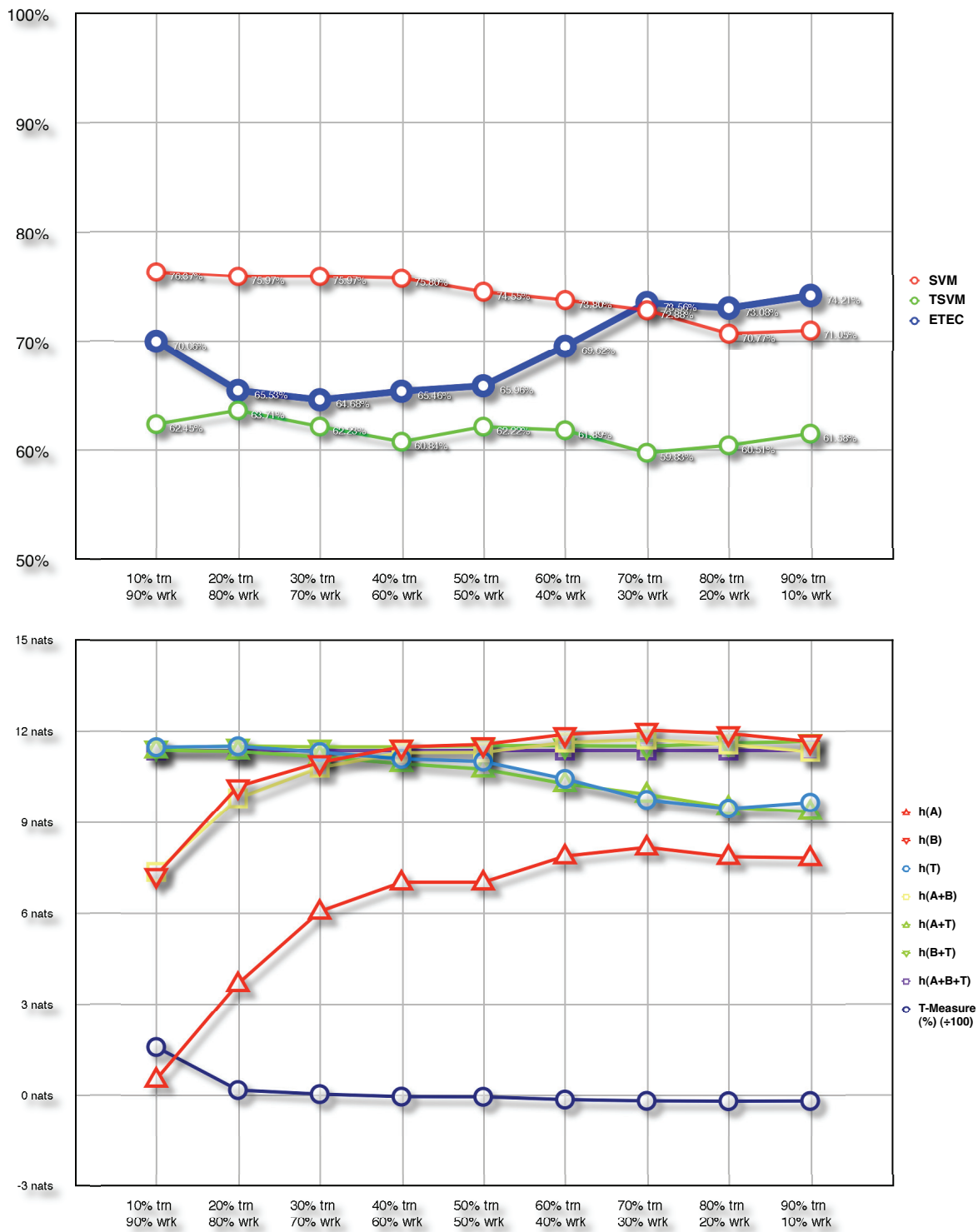


Figura 5.13: Resultados gráficos para **wpbc**, amostragem polarizada.

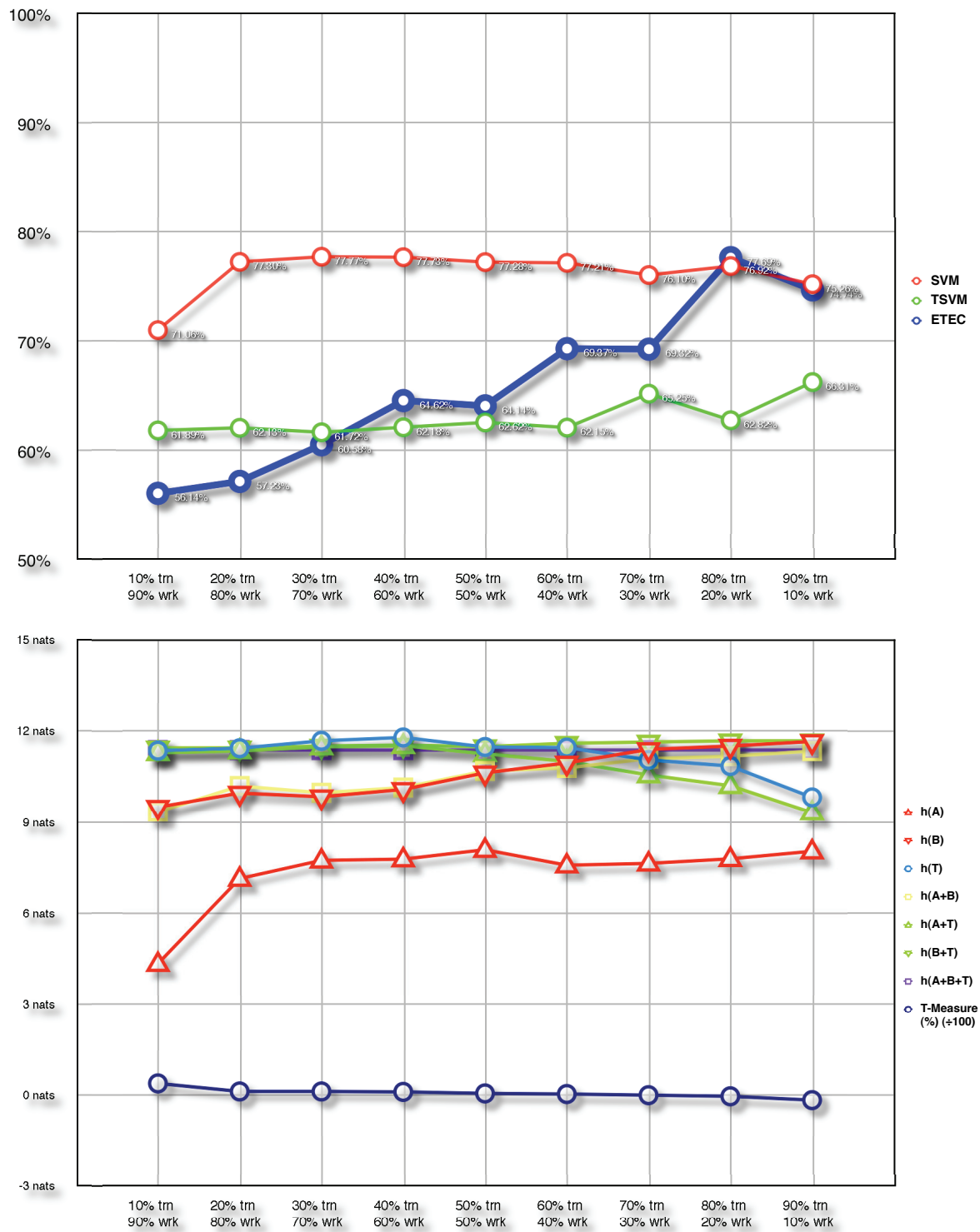


Figura 5.14: Resultados gráficos para **wpbc**, amostragem não polarizada.

Capítulo 6

Conclusão

Neste capítulo tecemos nossas conclusões finais para este trabalho. Existem três tópicos que abordamos: a teoria, o método proposto e os resultados obtidos.

Começamos pela teoria. Neste trabalho, analisamos a teoria clássica de aprendizado estatístico para o problema transdutivo. Identificamos questões específicas que fazem com que esta teoria não seja utilizada como base de métodos de aprendizado, de forma similar ao que ocorre com o aprendizado indutivo. De acordo com outros autores, acreditamos que o aprendizado transdutivo ainda não tem uma teoria completa e difundida, ao contrário de outros tipos de inferência. A medida que o estudo do problema transdutivo evolui na comunidade de aprendizado de máquina, esperamos que uma nova e detalhada teoria seja eventualmente desenvolvida.

Neste trabalho foram dados alguns passos nesta direção. Para superar algumas dificuldades apresentadas pela teoria clássica, apresentamos variações que mais tarde foram utilizadas como a base para construirmos um novo método transdutivo. Dado o sucesso da implementação e os resultados deste método, acreditamos que o conceito de estimativa direta de riscos, possível com estas variações, pode ser utilizado para a construção de outros métodos. Além disso, acreditamos também que esta nova teoria,

apenas proposta de forma abstrata neste trabalho, pode ser aprofundada em diferentes vertentes, por exemplo na análise de consistência do aprendizado transdutivo, assim como na sua convergência e comparação com o aprendizado indutivo.

Talvez um dos maiores obstáculos que impediu o desenvolvimento do aprendizado transdutivo até hoje tenha sido a quantidade de teorias e métodos eficientes disponíveis para o aprendizado indutivo. Um dos sintomas deste viés pode ser observado em alguns trabalhos encontrados na literatura, onde nem o próprio conceito de transdutividade é uniformemente reconhecido. A mudança de paradigma é, sem dúvida, notável. Parte deste legado do aprendizado indutivo deve ser substituído por novos conceitos que descrevem totalmente um processo transdutivo, efetivamente abraçando a idéia de não indução de uma regra de aprendizado genérica utilizada para deduzir respostas.

Sobre o método proposto, acreditamos que uma de suas maiores qualidades seja sua sólida base teórica. Através de uma variação da teoria original como ponto de partida, demos uma sequência de passos para construir o método, cada um embasado por princípios da Estatística e Teoria da Informação. Como resultado, chegamos ao método transdutivo final, que teve bom resultado em problemas de classificação binária desde o começo. Este fato, combinado com sua base teórica, nos leva a crer que o método está apenas em seus estágios preliminares de desenvolvimento, e que pesquisas futuras têm o potencial de aumentar seu desempenho e aplicabilidade.

Junto com o método propriamente dito, nossas proposições de indicadores de informação também podem ser desenvolvidas. Durante o trabalho, notamos como eles descreveram bem algumas das propriedades dos problemas que influenciaram o desempenho do método proposto. De forma análoga ao próprio método, estes indicadores têm o potencial de se tornarem medidas genéricas para problemas de aprendizado transdutivo.

Por fim, podemos fazer algumas observações sobre os resultados experimentais obtidos. Primeiro, não só o método proposto teve bom desempenho nas bases de dados selecionadas, mas os indicadores propostos conseguiram identificar os contextos nos quais o método apresentou ganhos em relação aos métodos indutivos. Segundo, algumas de nossas suspeitas foram confirmadas, por exemplo que métodos indutivos têm seu desempenho degradado quando a suposição de dados amostrados i.i.d. é quebrada. O método transdutivo proposto não só teve desempenho melhor que os métodos indutivos nestes cenários, mas em alguns casos conseguiu capturar e aproveitar as tendências implícitas nos dados. Além disso, os indicadores foram muito bem sucedidos em detectar polarizações e desequilíbrios nos problemas testados.

Como última observação, pudemos testemunhar empiricamente como o problema de ajuste de parâmetros pode afetar o desempenho de um modelo de aprendizado. Apesar de submetermos nossos experimentos de referência a um processo de ajuste de parâmetros, alcançar uma configuração ótima para cada base de dados é um grande desafio. Enquanto imaginávamos o efeito de parâmetros ruins em nossos experimentos, pudemos observar as vantagens do método proposto ser livre de parâmetros.

Referências Bibliográficas

- [BGL⁺99] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, Jr. M. Ares, and D. Haussler. Support vector machine classification of microarray gene expression data. Technical report, University of California, Santa Cruz, 1999.
- [BM98] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.
- [Bre94] Leo Breiman. Bagging predictors. Technical report, University of California at Berkeley, 1994.
- [Cac94] Christian Cachin. Pedagogical pattern selection strategies. *Neural Networks*, 7(1):175–181, 1994.
- [CKB87] G. Cestnik, I. Kononenko, and I. Bratko. Assistant-86: A knowledge-elicitation tool for sophisticated users. *Progress in Machine Learning*, pages 31–45, 1987.
- [CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.
- [CWD03] Yisong Chen, Guoping Wang, and Shihai Dong. Learning with progressive transductive support vector machine. *Pattern Recogn. Lett.*, 24(12):1845–1855, 2003.
- [DLLP97] Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.
- [DSS93] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705 – 719, 1993.

- [HAH01] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, Inc., Upper Saddle River, New Jersey, 2001.
- [Hay94] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 866 Third Avenue, New York, New York 10022, 1994.
- [Heb49] D. O. Hebb. *The Organization of Behavior*. Wiley, 1949.
- [Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective properties. In *Proceedings of the National Academy of Sciences*, 79, pages 2554–2558, 1982.
- [HR87] C. Harley and R. Reynolds. Analysis of e. coli promoter sequences. *Nucleic Acids Research*, 15:2343–2361, 1987.
- [Joa98] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT Press, 1998.
- [Joa99] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.
- [KK01] Michihiro Kuramochi and George Karypis. Gene classification using expression profiles: A feasibility study. In *IEEE International Conference on Bioinformatics and Biomedical Engineering*, pages 191–200, 2001.
- [LG94] Alberto Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, second edition, 1994.
- [MBdC07] Thiago Turchetti Maia, Antonio Pádua Braga, and André F. de Carvalho. Hybrid classification algorithms based on boosting and support vector machines. *Kybernetes*, 2007. to appear.
- [MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [MP69] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry*. MIT Press, 1969.

- [MSW95] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.
- [Mun92] P. W. Munro. Repeat until bored: A pattern selection strategy. *Advances in neural information processing systems*, 4:1001–1008, 1992.
- [OC89] M. C. O’Neill and F. Chiafari. Escherichia coli promoters. ii. a spacing-class dependent promoter search protocol. *Journal of Biological Chemistry*, 264:5531–5534, 1989.
- [O’N89] M. O’Neill. Escherichia coli promoters: Consensus as it relates to spacing class specificity, repeat substructure, and threedimensional organization. *Journal of Biological Chemistry*, 264:5522–5530, 1989.
- [Por98] Noah Porter, editor. *Webster’s Revised Unabridged Dictionary*. C. & G. Merriam Co., 1998.
- [Qui92] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. The Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann, October 1992.
- [RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- [Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65:386–408, 1958.
- [Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [Sch92] R. E. Schapire. *The Design and Analysis of Efficient Learning Algorithms*. MIT Press, 1992.
- [SED⁺88] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*, pages 261–265. IEEE Computer Society Press, 1988.
- [SMW95] W. N. Street, O. L. Mangasarian, and W. H. Wolberg. An inductive learning approach to prognostic prediction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 522–530. Morgan Kaufmann, 1995.

- [SWHB89] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.
- [Tho18] William M. Thorburn. The myth of occam’s razor. *Mind*, 27:345–353, 1918.
- [TSN90] G. Towell, J. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, 1990.
- [Vap79] Vladimir. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. (in Russian), Nauka, Moscow, 1979. (English translation: Vapnik, V. N. 1982. *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York).
- [Vap95] Vladimir. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
- [Vap98] Vladimir. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.
- [Vap99] Vladimir. N. Vapnik. *Three Remarks on the Support Vector Method of Function Estimation*, chapter 3, pages 25–41. The MIT Press, Cambridge, Massachusetts, 1999.
- [VC71] Vladimir N. Vapnik and A. J. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.
- [WH60] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *Institute of Radio Engineers, Western Electronic Show and Convention*, 1960.
- [WM90] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, pages 9193–9196, 1990.
- [Zha92] J. Zhang. Selecting typical instances in instance-based learning. In *Proceedings of the Ninth International Machine Learning Conference*, pages 470–479. Morgan Kaufmann, 1992.

Transductive Learning Based On Information Theory and Statistical Learning Theory

by

Thiago Turchetti Maia

B.Sc., Universidade Federal de Minas Gerais, Brazil, 1999;

M.Sc., Universidade Federal de Minas Gerais, Brazil, 2003.

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy in Electrical Engineering

Universidade Federal de Minas Gerais

July 2007

© Thiago Turchetti Maia, 2007

This thesis is dedicated to my family and their example of hard work and dedication
that kept me going during all these years.

Acknowledgments

I would like to thank my family for all their support and encouragement. Without them, this work would have become a burden too heavy to bear.

I would like to especially thank my supervisor, Prof. Braga, who has granted me with endless patience and understanding for so many years. Without his motivation and advice, this work would never have existed.

Thiago Turchetti Maia

Belo Horizonte, July 2007

Abstract

The machine learning problem is most frequently proposed and solved under the inductive inference paradigm, based on classical inductive principles such as the Empirical Risk Minimization (ERM) and Structural Risk Minimization (SRM). These principles are based on the attempt to create a learning rule given some finite training data. They rely on two assumptions: first, that there are enough observations in the training data and that these data are drawn independently and identically distributed (i.i.d.) with respect to the original data distribution of the problem, and second, that one actually needs to determine a general rule able to estimate points anywhere in the space of the problem.

Transductive learning challenges these assumptions, breaking the concept of inducing a general rule and using it to deduce unknown values of a functional dependence. In a transductive setting, one estimates particular values of a functional dependence directly, without necessarily having to estimate the general function itself and, implicitly, all of its possible evaluations in a continuum of points. Furthermore, transductive learning generally takes advantage of the information available at the points of interest to enhance its estimate of the problem's underlying probability distributions and improve its accuracy. Following Occam's Razor principle, transductive learning not only avoids solving a harder problem when there may not be enough information to do so, but it also inherently uses all information available *a priori*, including that given by the points of

interest.

In this work, we propose a new approach to transductive learning based on a variation of statistical learning theory, where risk functionals are directly estimated in a semi-supervised transductive framework. From that, a new parameter-free method, the essentially-transductive entropic classifier (ETEC), is proposed using concepts from information theory, where the learning problem is solved as a constrained optimization problem. In selected contexts identifiable using proposed information indicators, we show that our method outperforms other popular inductive and transductive methods for binary classification problems, such as support vector machines (SVMs) and transductive support vector machines (TSVMs).

Resumo

O problema de aprendizado de máquinas normalmente é proposto e resolvido dentro do paradigma da inferência indutiva, através de princípios clássicos como a minimização do risco empírico (ERM) ou a minimização do risco estrutural (SRM). Estes princípios se baseiam na criação de regras através de uma quantidade finita de dados, e assumem duas premissas fundamentais. Primeiro, que há pontos suficientes nos dados de treinamento e estes são amostrados i.i.d. (independentemente e identicamente distribuídos) em relação à distribuição do problema original. Segundo, que é necessário criar uma regra geral para estimar pontos desconhecidos em qualquer parte do espaço do problema.

O modelo de aprendizado transdutivo desafia essas premissas, quebrando o conceito de indução de uma regra genérica que é então utilizada para deduzir pontos desconhecidos de uma dada função. Em uma abordagem transdutiva, valores de pontos desconhecidos são estimados diretamente, sem a necessidade de se estimar a própria função genérica e, implicitamente, todas as suas possíveis avaliações no espaço contínuo do problema. Além disso, o aprendizado transdutivo geralmente utiliza a informação contida nos próprios pontos de interesse para melhorar sua estimativa das distribuições de probabilidades do problema e aumentar seu desempenho. De acordo com o princípio da Navalha de Occam, o aprendizado transdutivo não só evita a solução de um problema mais difícil para o qual pode-se não haver informações suficientes, mas ele também inerentemente

utiliza toda informação disponível *a priori*, incluindo a contida nos próprios pontos de interesse.

Neste trabalho, uma nova abordagem para o aprendizado transdutivo é proposta baseada em uma variação da teoria de aprendizado estatístico, onde funcionais risco são diretamente estimados em um arcabouço de aprendizado semi-supervisionado e transdutivo. A partir desta abordagem, um novo método livre de parâmetros, o classificador entrópico essencialmente transdutivo (ETEC), é proposto utilizando conceitos da teoria da informação, onde o problema de aprendizado é resolvido como um problema de otimização. Os resultados apresentados indicam que este método supera outros métodos indutivos e transdutivos usuais, como máquinas de vetores de suporte (SVMs) e máquinas de vetores de suporte transdutivas (TSVMs), para problemas de classificação binária em contextos selecionados, identificados através de indicadores de informação propostos.

Résumé

Le problème d'apprentissage de machines est normalement proposé et résolu sous le paradigme de l'inférence inductive, basée sur des principes classiques comme la minimisation du risque empirique (ERM) ou la minimisation du risque structurale (SRM). Ces principes sont basés sur la création des règles par une quantité finie de données, et supposent deux prémisses fondamentales. Premièrement, qu'y a des points suffisants dans les données d'apprentissage et ceux-ci sont tirés i.i.d. (identiquement et indépendamment distribués) concernant la distribution du problème original. En second, qu'il est nécessaire de créer une règle générale pour estimer des points inconnus en n'importe quel partie de l'espace du problème.

Le modèle d'apprentissage transductif défie ces prémisses, en cassant le concept d'induction d'une règle générique qui alors est utilisée pour déduire des points inconnus d'une dépendance fonctionnelle. Dans un abordage transductif, les valeurs de points inconnus sont estimés directement, sans devoir nécessairement estimer la fonction générique elle-même et, implicitement, toutes ses évaluations possibles dans l'espace continu du problème. Et puis, l'apprentissage transductif généralement utilise les informations contenues aux points d'intérêt eux-mêmes pour améliorer son estimation des distributions de probabilités du problème et augmenter sa performance. Conformément au principe du Rasoir d'Occam, l'apprentissage transductif n'évite pas seulement la solution d'un

problème plus difficile pour lequel il ne peut pas y avoir d'information suffisantes, mais il utilise aussi toutes les informations disponibles *a priori*, y compris celles données par les points d'intérêt.

Dans ce travail, nous avons proposé un nouvel abordage à l'apprentissage transductif basé sur une variation de la théorie d'apprentissage statistique, où les fonctionnels risque sont directement estimés dans un cadre d'apprentissage transductif et semi-supervisé. A partir de cela, on a proposé une nouvelle méthode exempte des paramètres, le classificateur entropique essentiellement transductif (ETEC), en utilisant des concepts de la théorie de l'information, où le problème d'apprentissage est résolu comme un problème d'optimisation. Les résultats présentés indiquent que cette méthode dépasse d'autres méthodes inductives et transductifs usuelles, comme des machines à vecteurs de support (SVMs) et des machines à vecteurs de support transductifs (TSVMs), pour des problèmes de classement binaire dans des contextes sélectionnés, identifiés à travers des indicateurs d'informations proposés.

Contents

Dedication	ii
Acknowledgments	iii
Abstract	iv
Resumo	vi
Résumé	viii
Contents	x
Abbreviation List	xviii
Symbology	xix
List of Tables	xxi
List of Figures	xxv
Occam's Razor	xxvii

I	Introduction	1
1	Introduction	2
1.1	A History of the Learning Problem	2
1.2	The Learning Problem	4
1.3	Contributions of This Work	7
1.4	Thesis Outline	9
II	Review	12
2	Statistical Learning Theory	13
2.1	The Learning Problem	14
2.2	Theoretical Foundations	17
2.2.1	PAC Learning	17
2.2.2	Vapnik-Chervonenkis Theory	18
2.3	Inductive Learning Principles	21
2.3.1	Empirical Risk Minimization	21
2.3.2	Structural Risk Minimization	23
2.4	The Transductive Learning Problem	25
2.5	Transductive Learning Principle	27
3	Support Vector Machines	29
3.1	Linear Learning Machines	29
3.1.1	Linear Classification	30
3.1.2	Dual Representation	31
3.2	Kernel-Induced Feature Spaces	32
3.2.1	Mapping into Feature Spaces	32

3.2.2	Building Kernels	34
3.2.3	Examples of Kernels	36
3.3	Building Support Vector Machines	37
3.3.1	Linear Support Vector Machines	38
3.3.2	Non-Linear Support Vector Machines	41
3.3.3	Soft Margin Optimization	44
3.3.4	Support Vector Regression	50
3.4	Building Transductive Support Vector Machines	53
3.4.1	Transductive Pattern Recognition	53
3.4.2	Transductive Regression	55
3.5	Training Methods	57
3.5.1	General Techniques	58
3.5.2	Decomposition and Chunking	60
3.5.3	Other Approaches	62
3.5.4	Sequential Minimal Optimization	62
3.6	The Minimax Problem for Transductive Learning	67
4	Information Theory	70
4.1	Overview	70
4.2	Units of Information	71
4.3	Information Measures	72
4.3.1	Self-information	72
4.3.2	Entropy	73
4.3.3	Differential Entropy	73
4.3.4	Joint Entropy	75
4.3.5	Conditional Entropy	77

4.3.6	Mutual Information	78
4.3.7	Cross Entropy	79
4.3.8	Kullback-Leibler Divergence	80
5	Sample Statistics	85
5.1	Variance	85
5.2	Covariance	86
5.3	Covariance Matrix	86
5.4	Weighted Covariance Matrix	87
III	Development	89
6	Interpreting Statistical Learning Theory	90
6.1	Transductivity and Learning Rules	90
6.2	An Essentially-Transductive Approach to the Learning Problem	92
6.3	Theory Variations	94
6.3.1	Unsupervised Inductive Learning	96
6.3.2	Unsupervised Transductive Learning	96
6.3.3	Supervised Inductive Learning	97
6.3.4	Supervised Transductive Learning	98
6.3.5	Semi-supervised Inductive Learning	98
6.3.6	Semi-supervised (or Essentially-) Transductive Learning	99
6.4	Existence of the Separating Margin	100
7	Building an Essentially-Transductive Method	104
7.1	Information-Theoretic Estimation of the Transductive Risk	105
7.2	Moving into the Continuous	107

7.3	The Problem of Density Estimation is Ill-Posed	109
7.4	Gaussian Assumption	110
7.5	Independence Assumption	112
7.6	Modified Weighted Covariance	114
8	Essentially-Transductive Entropic Classifier	116
8.1	Learning as a Constrained Optimization Problem	116
8.2	Solving the Optimization Problem	119
9	Information Indicators	120
9.1	Overview	120
9.2	First-Order Indicators: Entropy of Sets	122
9.3	Second-Order Indicators: Entropy of Compound Sets	122
9.4	Third-Order Indicators: Transductivity Measure	123
IV	Results	125
10	Data Sets	126
10.1	Description	126
10.1.1	Wisconsin Breast Cancer Database (bcw)	127
10.1.2	Hepatitis Domain (hepatitis)	128
10.1.3	Johns Hopkins University Ionosphere Database (ionosphere) . . .	129
10.1.4	Musk Database (musk)	130
10.1.5	Escherichia coli promoter gene sequences (DNA) (pgs)	130
10.1.6	Pima Indians Diabetes Database (pid)	131
10.1.7	Wisconsin Prognostic Breast Cancer (wpbc)	132
10.2	Sampling	133

10.2.1 Biased and Unbiased Sampling	133
10.2.2 Training and Working Set Sizes	134
10.3 Information Indicators	134
11 Benchmark Results	135
11.1 Benchmark Methods	135
11.2 Parameter Tuning	136
11.3 Reference Results	137
12 Experimental Results	138
12.1 Incorporating Additional Information	138
12.2 Results	140
12.2.1 Results for bcw	140
12.2.2 Results for hepatitis	142
12.2.3 Results for ionosphere	143
12.2.4 Results for musk	145
12.2.5 Results for pgs	146
12.2.6 Results for pid	148
12.2.7 Results for wpbc	149
12.3 Graphical Comparison	151
12.4 Discussion of Results	166
12.4.1 Analysis for bcw	166
12.4.2 Analysis for hepatitis	168
12.4.3 Analysis for ionosphere	169
12.4.4 Analysis for musk	170
12.4.5 Analysis for pgs	170

12.4.6 Analysis for pid	171
12.4.7 Analysis for wpbc	171
V Conclusion	173
13 Conclusion	174
14 Future Work	178
14.1 Performance Improvement	178
14.2 Consistency of Transductive Learning Processes	179
14.3 Indeterminacy of the Separating Margin	180
14.4 Multi-class Classification	181
14.5 Multi-label Classification	182
14.6 Regression Problems	182
14.7 Sparseness: Support Vector Entropy Concept	183
VI Appendices	186
Appendix A Information Indicators	187
A.1 Indicators for bcw	187
A.2 Indicators for hepatitis	190
A.3 Indicators for ionosphere	192
A.4 Indicators for musk	194
A.5 Indicators for pgs	196
A.6 Indicators for pid	198
A.7 Indicators for wpbc	200

Appendix B	Reference Results	202
B.1	Biased Sampling	202
B.1.1	Reference Results for Biased bcw	202
B.1.2	Reference Results for Biased hepatitis	204
B.1.3	Reference Results for Biased ionosphere	205
B.1.4	Reference Results for Biased musk	207
B.1.5	Reference Results for Biased pgs	208
B.1.6	Reference Results for Biased pid	210
B.1.7	Reference Results for Biased wdbc	211
B.2	Unbiased Sampling	213
B.2.1	Reference Results for Unbiased bcw	213
B.2.2	Reference Results for Unbiased hepatitis	214
B.2.3	Reference Results for Unbiased ionosphere	216
B.2.4	Reference Results for Unbiased musk	217
B.2.5	Reference Results for Unbiased pgs	219
B.2.6	Reference Results for Unbiased pid	220
B.2.7	Reference Results for Unbiased wdbc	222
Appendix C	Derivatives	224
Appendix D	Notes on Performance Measures	229
Appendix E	Notes on Typesetting	230
Bibliography		231

Abbreviation List

These are the abbreviations most commonly used throughout the text, most of which widely adopted in the literature.

SVM	Support Vector Machine
TSVM	Transductive Support Vector Machine
ETEC	Essentially-Transductive Entropic Classifier
ERM	Empirical Risk Minimization
SRM	Structural Risk Minimization
SMOR	Structural Minimization of Overall Risk
DG	Data Generator
O	Oracle (or Supervisor)
E	Evaluator
LM	Learning Machine
PAC	Probably Approximately Correct
MCE	Minimum Cross Entropy
MDI	Minimum Discrimination Information
MLE	Maximum Likelihood Estimation
ME	Maximum Entropy Method
i.i.d.	Identically and independently distributed

Symbology

The following mathematical symbols are uniformly used throughout the text with the same meaning. Symbols in bold denote vectors. Specific symbols are described in selected scopes.

X	Input domain of learning problems, usually $X \in \mathbb{R}^n$;
Y	Output domain of learning problems, where $Y \in \{-1, 1\}$ or $Y \in \{0, 1\}$ for pattern recognition, $Y \in \mathbb{R}$ for regression, among other possible configurations;
$F(\mathbf{x})$	Probability distribution function from which all data $\mathbf{x} \in X$ are drawn;
$F(y \mathbf{x})$	Conditional probability distribution function from which data $y \in Y$ are drawn given $\mathbf{x}$;
Λ	Set of all possible parameters that govern the behavior of a given algorithm;
α	Set of parameters that govern the behavior of a given algorithm, usually $\alpha \in \Lambda$;
$f(\mathbf{x}, \alpha)$	Arbitrary set of functions in Y that approximate the unknown functional dependence being sought;

$L(y, f(\mathbf{x}, \alpha))$	Loss or discrepancy function that measures how fit is the approximation of y given by $f(\mathbf{x}, \alpha)$;
$R(\alpha)$	Expected risk of the learning problem;
$R_{emp}(\alpha)$	Empirical risk of the learning problem;
$R_{\Sigma}(\alpha)$	Overall risk of the transductive learning problem;
$I(x)$	Self-information associated with x ;
$H(X)$	Entropy of random variable X ;
$h(X)$	Differential entropy of random variable X ;
$H(X, Y)$	Joint entropy of random variables X and Y ;
$H(X Y)$	Conditional entropy of random variable X given Y ;
$I(X; Y)$	Mutual information between random variables X and Y ;
$\check{H}(X, Y)$	Cross entropy between random variables X and Y ;
$D_{KL}(X\ Y)$	Kullback-Leibler divergence, or relative entropy, between random variables X and Y .

List of Tables

3.1	Summary of commonly used inner-product kernel functions.	37
4.1	Continuous entropy expressions for known probability distributions. . . .	76
10.1	Datasets and their partitions for experiments described in this section. .	127
11.1	Results from parameter tuning procedure, adopted for experiments with SVM and TSVM.	137
12.1	ETEC results for bcw , biased distribution.	141
12.2	ETEC results for bcw , unbiased distribution.	141
12.3	ETEC results for hepatitis , biased distribution.	142
12.4	ETEC results for hepatitis , unbiased distribution.	143
12.5	ETEC results for ionosphere , biased distribution.	144
12.6	ETEC results for ionosphere , unbiased distribution.	144
12.7	ETEC results for musk , biased distribution.	145
12.8	ETEC results for musk , unbiased distribution.	146
12.9	ETEC results for pgs , biased distribution.	147
12.10	ETEC results for pgs , unbiased distribution.	147
12.11	ETEC results for pid , biased distribution.	148

12.12	ETEC results for pid , unbiased distribution.	149
12.13	ETEC results for wpbc , biased distribution.	150
12.14	ETEC results for wpbc , unbiased distribution.	150
A.1	Primary indicators for bcw , biased distribution.	188
A.2	Compound and transductivity indicators for bcw , biased distribution. . .	188
A.3	Primary indicators for bcw , unbiased distribution.	189
A.4	Compound and transductivity indicators for bcw , unbiased distribution. .	189
A.5	Primary indicators for hepatitis , biased distribution.	190
A.6	Compound and transductivity indicators for hepatitis , biased distribution.	190
A.7	Primary indicators for hepatitis , unbiased distribution.	191
A.8	Compound and transductivity indicators for hepatitis , unbiased distribution.	191
A.9	Primary indicators for ionosphere , biased distribution.	192
A.10	Compound and transductivity indicators for ionosphere , biased distribution.	192
A.11	Primary indicators for ionosphere , unbiased distribution.	193
A.12	Compound and transductivity indicators for ionosphere , unbiased distribution.	193
A.13	Primary indicators for musk , biased distribution.	194
A.14	Compound and transductivity indicators for musk , biased distribution. .	194
A.15	Primary indicators for musk , unbiased distribution.	195
A.16	Compound and transductivity indicators for musk , unbiased distribution. .	195
A.17	Primary indicators for pgs , biased distribution.	196
A.18	Compound and transductivity indicators for pgs , biased distribution. . .	196
A.19	Primary indicators for pgs , unbiased distribution.	197

A.20	Compound and transductivity indicators for pgs , unbiased distribution. .	197
A.21	Primary indicators for pid , biased distribution.	198
A.22	Compound and transductivity indicators for pid , biased distribution. . .	198
A.23	Primary indicators for pid , unbiased distribution.	199
A.24	Compound and transductivity indicators for pid , unbiased distribution. .	199
A.25	Primary indicators for wpbc , biased distribution.	200
A.26	Compound and transductivity indicators for wpbc , biased distribution. .	200
A.27	Primary indicators for wpbc , unbiased distribution.	201
A.28	Compound and transductivity indicators for wpbc , unbiased distribution.	201
B.1	SVM results for bcw , biased distribution.	203
B.2	TSVM results for bcw , biased distribution.	203
B.3	SVM results for hepatitis , biased distribution.	204
B.4	TSVM results for hepatitis , biased distribution.	205
B.5	SVM results for ionosphere , biased distribution.	206
B.6	TSVM results for ionosphere , biased distribution.	206
B.7	SVM results for musk , biased distribution.	207
B.8	TSVM results for musk , biased distribution.	208
B.9	SVM results for pgs , biased distribution.	209
B.10	TSVM results for pgs , biased distribution.	209
B.11	SVM results for pid , biased distribution.	210
B.12	TSVM results for pid , biased distribution.	211
B.13	SVM results for wpbc , biased distribution.	212
B.14	TSVM results for wpbc , biased distribution.	212
B.15	SVM results for bcw , unbiased distribution.	213
B.16	TSVM results for bcw , unbiased distribution.	214

B.17	SVM results for hepatitis , unbiased distribution.	215
B.18	TSVM results for hepatitis , unbiased distribution.	215
B.19	SVM results for ionosphere , unbiased distribution.	216
B.20	TSVM results for ionosphere , unbiased distribution.	217
B.21	SVM results for musk , unbiased distribution.	218
B.22	TSVM results for musk , unbiased distribution.	218
B.23	SVM results for pgs , unbiased distribution.	219
B.24	TSVM results for pgs , unbiased distribution.	220
B.25	SVM results for pid , unbiased distribution.	221
B.26	TSVM results for pid , unbiased distribution.	221
B.27	SVM results for wpbc , unbiased distribution.	222
B.28	TSVM results for wpbc , unbiased distribution.	223

List of Figures

1.1	<i>Inductive inference</i> , from particular to general. <i>Deductive inference</i> , from general to particular. <i>Transductive inference</i> , from particular to particular.	6
2.1	Learning model based on inductive inference, where a function that best approximates the unknown functional dependence is selected from the set of all possible functions based on the given training data.	15
2.2	Learning model based on transductive inference. A modified learning machine takes both training and working sets as inputs, and the evaluator is only able to estimate values on either of these sets.	25
12.1	Graphical results and indicators for bcw , biased sampling.	152
12.2	Graphical results and indicators for bcw , unbiased sampling.	153
12.3	Graphical results and indicators for hepatitis , biased sampling.	154
12.4	Graphical results and indicators for hepatitis , unbiased sampling.	155
12.5	Graphical results and indicators for ionosphere , biased sampling.	156
12.6	Graphical results and indicators for ionosphere , unbiased sampling.	157
12.7	Graphical results and indicators for musk , biased sampling.	158
12.8	Graphical results and indicators for musk , unbiased sampling.	159
12.9	Graphical results and indicators for pgs , biased sampling.	160

12.10	Graphical results and indicators for pgs , unbiased sampling.	161
12.11	Graphical results and indicators for pid , biased sampling.	162
12.12	Graphical results and indicators for pid , unbiased sampling.	163
12.13	Graphical results and indicators for wpbc , biased sampling.	164
12.14	Graphical results and indicators for wpbc , unbiased sampling.	165

Entia non sunt multiplicanda praeter necessitatem.

(Entities should not be multiplied beyond necessity.)

Lex parsimoniae, or *Occam's Razor* principle,
by William of Occam, 14th-century logician and Franciscan friar.

Part I

Introduction

Chapter 1

Introduction

1.1 A History of the Learning Problem

The problem of teaching a machine how to acquire knowledge has been explored ever since modern computer architectures came to be. The first model of an artificial neuron is due to McCulloch and Pitts back in 1943 [MP43], in their classical work of describing what was to be the basic unit of a network of neurons that would resemble the human brain. The focus of their work was much less in the methods for inducing knowledge into such networks, but instead in their computing abilities [BLC00]. Later in 1949, the learning of these networks of neurons was explored by Hebb [Heb49], who proposed a model based on the adjustment of weights at the neurons' inputs. This model stood on the theory that biological neurons learn based on the reinforcement of synaptic links to other excited neurons. Widrow and Hoff [WH60] further refined Hebb's ideas in 1960, incorporating the method of gradient descent to minimize the output error of each neuron. The first attempt to tackle pattern recognition problems with neural networks is due to Rosenblatt in 1958 [Ros58] in which he introduced the *Perceptron* model. The Perceptron is a simple neural network able to solve linearly separable classification problems in

which the output space is divided into complementary regions that correspond to each category.

Though this connectionist approach¹ received great attention after the work of Rosenblatt, the machine learning community was discouraged to pursue it any further by Minsky and Papert in 1969 [MP69]. Minsky and Papert argued that the limitation of not being able to solve non-linearly separable problems was too great to be ignored, along with other restrictions such as explosive growth in space and time, and the lack of a learning rule to multi-layered networks. The 1970's were hence marked by the lack of interest from the community in this area, except for a few scattered researchers. It was not until the works of Hopfield in 1982 [Hop82] and the development of the *back-propagation* algorithm in 1986 [RHW86] that artificial neural networks regained general interest. Hopfield explored the relation between recurrent associative networks and physical models, which paved the way for using Physics theories to describe the behavior of learning machines [Hop82]. Much attention was then drawn to the work of Rumelhart et al. [RHW86], in which they managed to create an algorithm for training multi-layered neural networks, *back-propagation*, based on the adjustment of weights in each layer according to the retro-propagation of their output errors.

This rebirth from ashes of the connectionist school in the 1980's was followed by a new wave of research starting in the 1990's that explored new theories and algorithms focused not only in constructing and training powerful learning machines, but also in reorganizing the environment around them in order to improve their performance. Vapnik [Vap95] refers to this period as the return to the origins of statistical learning theory. Pioneer works such as those from Munro [Mun92], Schapire [Sch92], Drucker et al. [DSS93], and Cachin [Cac94] served as basis for others like Breiman [Bre94], who in-

¹Other terms that have been used to name this field include *connectionism*, *parallel distributed processing*, *neural computation*, *adaptive networks*, and *collective computation* [RN95].

troduced the Bagging method for training and combining multiple copies of a learning algorithm, and also Freund and Schapire, who later introduced the AdaBoost family of algorithms. The term *Boosting* itself is due to Schapire in 1990 [Sch90].

In parallel with the development of learning models and methodologies for improving the performance of previous algorithms, this return to the origins also motivated the creation of several new learning machines. Among these machines, we highlight those of Vapnik and coworkers which became known as Support Vector Machines. Relying on the previous 1965 work from Vapnik and Chervonenkis [VC71], who developed the method of the optimal separating hyperplane, Vapnik et al. created the support vector method in 1992 [Vap95]. Since then, the growing interest of the machine learning community motivated the development of several specialized training algorithms for solving the constrained convex quadratic optimization problem with which SVMs are formulated.

Since the 1990s, we have seen a lot of attention dedicated to the advanced parts of computational (or statistical) learning theory, what was once referred to as *returning to the origin* [Vap95]. Among the topics that have attracted researchers, transductive inference is still on its preliminary stages, despite having been initially proposed by Vapnik in the 1970's [VC71, Vap79]. Most of its issues, including the crucial one of understanding its full potential to inspire better machine learning models and applications, are yet to be answered.

1.2 The Learning Problem

Consider the learning problem as a generic function estimation problem for now. We are able to solve this problem in the usual *inductive inference* paradigm using classical inductive principles, such as the Empirical Risk Minimization (ERM) and Structural Risk

Minimization (SRM) principles. Regardless of their individual strategies, their goal is the same: to create learning rules with minimal risk given some fixed amount of training data. According to the statistics of large samples, a low-risk rule yields, with high probability, low error means when used over a large test set. Once these rules are created, they may then be used to estimate the unknown functional dependence at indefinitely as many later given points.

These inductive principles rely on two key assumptions. First, that there are enough observations in the training data and that these data are drawn independently and identically distributed (*i.i.d.*) with respect to the original data distribution of the problem. In other words, that the training data provides sufficient coverage of the space of the problem's dataset and that it contains enough information to estimate values at arbitrary points later given by the test data. Second, that in order to estimate values at these points, one actually needs to determine a general rule able to estimate points anywhere in the space of the problem.

Interestingly, the usual situation in machine learning practical problems is somewhat different. Training data are often limited, and it may easily fail to resemble well the space of the dataset. Also, one is often interested in estimating values only at fixed unknown points available before solving the problem, that is, a general rule is unnecessary.

In the case where the objective is to estimate values of the unknown functional dependence at given points, one is forced to solve two independent problems. First, one must estimate a general rule that resembles the unknown functional dependence. Later, one must evaluate the values of this function at these points. Notice that the problem of estimating a functional dependence is much more difficult, as it implies that from a limited dataset one must infer the values of the function at all possible points of its space.

We may define these two steps on the basis of classical philosophy (Figure 1.1). One

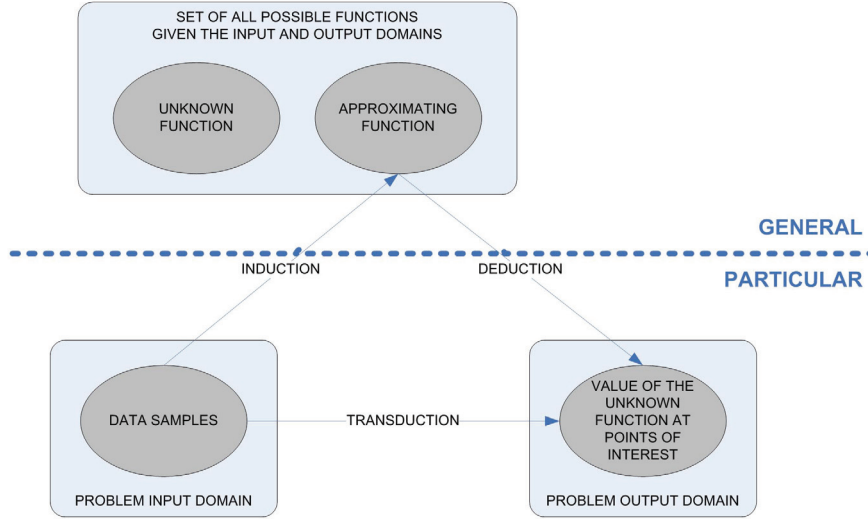


Figure 1.1: *Inductive inference*, from particular to general. *Deductive inference*, from general to particular. *Transductive inference*, from particular to particular.

first takes an *inductive* step to estimate an unknown functional dependence, moving from the particular information of the function values at the points of the training set to the general information of the function description. Later, one takes a *deductive* step to estimate specific values of this function, moving from the general function information back to the particular information of its values at the points of interest.

The main concept of the so-called *transductive inference* is to move directly from particular to particular, that is, to estimate values of unknown functional dependencies at given points without necessarily having to estimate the function itself. Notice that by doing so, not only is one solving an easier problem, but one is also inherently using the information available in the given points of interest themselves, information which is neglected by inductive inference. Especially in cases of limited training sets, this information is used to better describe the space of the problem at hand. If we interpret *Occam's Razor* principle [Tho18] with respect to the learning process, the generalization step imposed by inductive inference is unnecessary to convey the same information given

by transductive inference.

According to Vapnik, the problem of estimating the values of unknown functional dependencies at given points is analogous to the question of what is the basis of human intelligence: the knowledge of laws (rules) or the culture of direct access to the truth (intuition, ad-hoc inference). Based on this philosophical questioning, he states that *this [transductive] model can provide the strongest contribution to the 2000 years of discussions about the essence of human reason* [Vap95].

1.3 Contributions of This Work

In this work we examine the theory, methods, and applications of transductive inference. We start our discussion by evaluating the classical statistical learning theory, as well as its applicability to the transductive learning problem. We identify issues that make it difficult for the current theory to be directly used as a foundation to build new transductive learning machines. Particularly, we show how the risk functionals defined for transductive learning may not be directly evaluated, as opposed to the risk functionals proposed for inductive learning. Next, we clearly define what are inductive inference and transductive inference, as well as supervised, unsupervised, and semi-supervised learning models. Although these concepts are frequently used in the literature, they are sometimes mistaken for one another. Based on these definitions, we propose variations on the classical theory to describe each type of problem. These variations overcome the issue of not offering risk functionals that may be directly estimated. Together with this analysis of transductive inference and the definition of an alternative theory, this work also raises theoretical comparisons and questions that, with future research, may yield not only those but other theoretical contributions as well.

Based on the proposed variations of learning theory, we explore the semi-supervised transductive learning problem. In order to explicitly differentiate this setting of the learning problem, we coin the term and refer to it as *essentially-transductive* learning. We propose a way of directly estimating the risk functional of the essentially-transductive learning problem using concepts from information theory, thus bridging the gap between learning methods and transductive learning theory. Our principle is based on the Minimum Discrimination Information method, which relies on the Kullback-Leibler divergence between two probability distributions. We use the continuous definitions of Kullback-Leibler divergence and other entropic measures, as we show how the method benefits from a continuous search space as opposed to a discrete, and inherently hard to traverse, space. Backed by the Maximum-Likelihood and the Maximum Entropy methods, we assume that the underlying probability distributions of the problem are multivariate Gaussians. We also assume the independence amongst the dimensions of these multivariate distributions, which leads to an elegant and computationally-friendly risk expression. Additionally, we employ the concept of weighted covariance matrices to parametrize our multivariate distributions, where every sample in the problem is assigned to an individual weight. We then pose the evaluation of our learning problem as the optimization of a constrained optimization problem, which seeks the minimal risk in the search space formed by the weights assigned to data samples. The optimization problem is finally solved using a Quasi-Newton sequential quadratic programming algorithm. In terms of learning methods, this work contributes with the introduction of a new, parameter-free, binary classification method for the essentially-transductive learning problem, using as its foundation the suggested alternative theory as well as principles from information theory and statistics.

The proposed classification method, the essentially-transductive entropic classifier

(ETEC), was tested in 14 different problems, drawn from 7 datasets commonly used in the literature. The results obtained were compared to those obtained with two well-known methods, the standard inductive support vector machine (SVM) and the so-called transductive support vector machine (TSVM). Our results indicate that our method outperforms both the SVM and the TSVM in selected contexts. In order to easily identify these contexts, we propose a set of information indicators that reveal which learning problems are likely to be better solved using a transductive (or essentially-transductive) learning method. In addition to contributions to the theory and methods for transductive learning, this work also introduces a methodology capable of assessing the applicability of transductive machines to a given learning problem.

1.4 Thesis Outline

This thesis is organized as follows. The main text is divided into five parts, with an additional part for the appendices. Chapter 1, in Part I, contains the introduction, which ends after this section.

Part II brings the bibliographical review of the thesis. Chapter 2 presents inductive and transductive learning theories, often drawing comparisons between the two. Concepts such as learning principles and the VC (Vapnik-Chervonenkis) dimension are described, together with the classical formalizations of the inductive and transductive learning problems.

Chapter 3 introduces concepts and formulations of support vector machines, the learning method selected as benchmark in this work. We examine formulations of different machines, evolving from simple linear SVMs into non-linear and non-linear soft-margin SVMs, along with different approaches used to train them. We also describe the

so-called transductive support vector machine, an extension of the original SVM proposition capable of incorporating unlabeled data from a working set.

Chapter 4 reviews the concepts from information theory used in this work, most notably some information measures such as information entropy and the Kullback-Leibler divergence. Ending Part II, Chapter 5 brings a short review of concepts from sample statistics, namely variance, covariance, covariance matrices, and weighted covariance matrices.

Part III contains the main propositions and contributions resulting from this work. It starts with Chapter 6, where theoretical analyses on transductivity and statistical learning theory are given. The definition of the learning problem is revisited, this time with the introduction of variations of the classical learning theory.

Chapter 7 then describes each step necessary to build an essentially-transductive learning method, starting from its foundations in the proposed variation of the theory. We show how to employ the Minimum Discrimination Information principle to directly estimate the functional risk, along with other building blocks and assumptions made.

Chapter 8 concludes the proposition of the new method, the essentially-transductive entropic classifier (ETEC). It also describes how the method is based on the solution of a constrained optimization problem, and the algorithm used to solve it.

Chapter 9 ends Part III with the description of information indicators that reveal the applicability of transductive learning methods to a given problem.

Part IV describes the results from all experiments in this work. It starts with the description of the datasets and sampling strategies in Chapter 10, which also shows the information indicators for each of the problems experimented with. Chapter 11 brings the results for the two benchmark algorithms selected as references for inductive and transductive learning machines, namely SVM and TSVM. Chapter 12 show the results

obtained with the proposed ETEC method, and compares them to benchmark results.

Part V ends the main body of the text. Chapter 13 summarizes the results obtained in this work, and highlights the conclusions we reached from them. Chapter 14 looks into the future and suggests new paths towards the continuity of this research.

Part VI brings the appendices. For the sake of good aesthetics, Appendix C shows complete formulae for the derivatives of the objective function of the ETEC method. Ending this part, Appendix D describes the execution environment used in the experiments, and Appendix E brings remarks about the typesetting of the text.

The bibliography immediately follows the appendices and ends this thesis.

Part II

Review

Chapter 2

Statistical Learning Theory

This chapter reviews different topics of learning theory, including inductive and transductive learning principles. Vapnik [Vap95] divides learning theory into four parts:

- Theory of consistency of learning processes;
- Non-asymptotic theory of the rate of convergence of learning processes;
- Theory of controlling the generalization ability of learning processes;
- Theory of constructing learning algorithms.

In this chapter we shall examine concepts regarding the first three parts, where we establish the foundations to explore the forth part when we discuss the intricacies of support vector machines. Some relevant concepts on learning theory, necessary to a more general understanding of support vector machines (SVM), are introduced in this chapter in Section 2.1. A brief introduction to statistical learning theory is given in Section 2.3, taking into account important definitions such as empirical risk minimization and structural risk minimization principles. The PAC model and the Vapnik-Chervonenkis (VC) dimension concept are also presented in Section 2.2. The VC dimension can be

considered as a measure of the classification power of a family of functions and provides bounds to the empirical risk. Finally, in Sections 2.4 and 2.5 we examine the transductive learning problem and transductive learning principle, respectively.

2.1 The Learning Problem

According to computational learning theory, the learning problem may be formulated as a function estimation problem. Although its output may differ according to each domain, we assume that all inputs of any learning problem have the format of real feature vectors in some space $X \subset \mathbb{R}^n$. Based on this general definition, we are able to determine a generic learning model to solve the problem of approximating an unknown functional dependence.

The model contains four components (Figure 2.1). There is a data generator (DG), which selects random samples $\mathbf{x} \in X \subset \mathbb{R}^n$, drawn independently from an unknown but fixed probability distribution function $F(\mathbf{x})$. There is an oracle – or supervisor – (O), which returns an output value $y \in Y$ to every input sample $\mathbf{x}$ with respect to an unknown but fixed conditional distribution function $F(y|\mathbf{x})$. Recall that Y is arbitrary for the general learning problem. There is an evaluator (E) capable of implementing a set of functions $f(\mathbf{x}, \alpha), \alpha \in \Lambda$, where Λ is an arbitrary set of parameters that governs the behavior of the function. Notice that by making Λ arbitrary, $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ may actually be any set of functions. Finally, there is a learning machine (LM), which must select from the set of functions $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ the one that best approximates the oracle's output based on a training set. This training set is a finite set of samples

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \tag{2.1}$$

drawn *i.i.d.* by DG with respect to $F(\mathbf{x}, y) = F(\mathbf{x})F(y|\mathbf{x})$.

Once the machine is trained, the value $\alpha_0 \in \Lambda$ is determined and produces the function $f(\mathbf{x}, \alpha_0)$ through E that best approximates the unknown functional dependence. This approximation may then be used by E to estimate unknown values $\hat{y}^*$ at arbitrary points $\mathbf{x}^* \in X$. This model is similar to the classical model described by computational learning theory, only it has been further detailed to distinguish between LM and E .

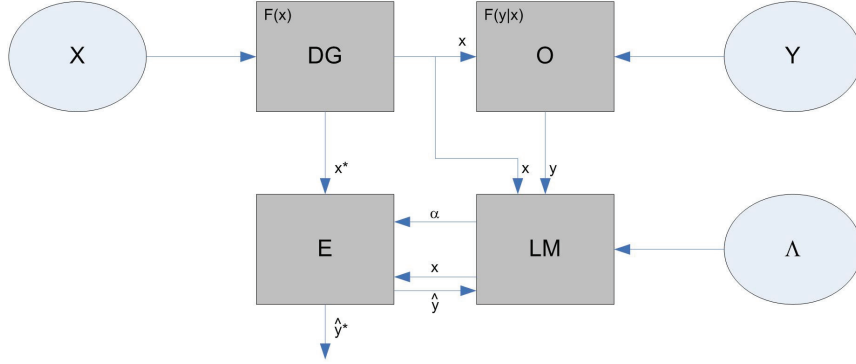


Figure 2.1: Learning model based on inductive inference, where a function that best approximates the unknown functional dependence is selected from the set of all possible functions based on the given training data.

Notice that, anytime during training, LM may present any sample $\mathbf{x} \in X$ to E and expect a response $\hat{y} = f(\mathbf{x}, \alpha)$. For samples $\mathbf{x}$ in the set (2.1), $\hat{y}$ may be compared to the answer y provided by the oracle O . This comparison is the basis of the selection mechanism that determines the approximation to the functional dependence, and it is measured by a loss or discrepancy function $L(y, f(\mathbf{x}, \alpha))$. Since the inductive problem assumes that E will be able to estimate values $\hat{y}^*$ of the unknown functional dependence at arbitrary points of interest $\mathbf{x}^* \in X$, notice that $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ must be defined for all $\mathbf{x}^* \in X \subset \mathbb{R}^n$. Frequently, the set X is continuous, in most cases simply $X = R^m \subset R^n, m < n$. Unless X is finite such that $X \subset R^m \subset R^n$, this requires $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ to be continuously defined over X .

Even not having defined the loss function, we are able to define the expected value

of the loss given by the risk functional

$$R(\alpha) = \int L(y, f(\mathbf{x}, \alpha)) dF(\mathbf{x}, y), \quad (2.2)$$

referred to as the expected risk.

The solution of the learning problem, implemented by LM , may hence be described as finding the function $f(\mathbf{x}, \alpha_0)$ which minimizes the expected risk. In other words, find the function $f(\mathbf{x}, \alpha)$ that minimizes the risk functional $R(\alpha)$ over the class of functions $f(\mathbf{x}, \alpha), \alpha \in \Lambda$, where $F(\mathbf{x}, y)$ is unknown and the only data available are in the training set (2.1).

From this generic function estimation problem described so far, we may derive the three main problems of machine learning, namely pattern recognition, regression estimation, and density estimation.

For the pattern recognition (or classification) problem, the oracle's output may only take two values, $Y = \{0, 1\}$, and $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ is referred as the set of indicator functions. The loss function may be defined as:

$$L(y, f(\mathbf{x}, \alpha)) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}, \alpha), \\ 1 & \text{if } y \neq f(\mathbf{x}, \alpha). \end{cases} \quad (2.3)$$

In this case, the risk functional (2.2) determines the probability of having different answers from the oracle and the learning machine.

For the regression estimation problem, the oracle's output may be a real value, $Y \subset \mathbb{R}$. Let $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ be a set of real functions that contain the regression function

$$f(\mathbf{x}, \alpha_0) = \int y dF(y|\mathbf{x}).$$

The loss function may be defined as:

$$L(y, f(\mathbf{x}, \alpha)) = (y - f(\mathbf{x}, \alpha))^2, \quad (2.4)$$

which is called quadratic loss function.

For the density estimation problem, one wishes to approximate a probability density function from a set of densities $p(\mathbf{x}, \alpha), \alpha \in \Lambda$. The loss function may be defined as:

$$L(p(\mathbf{x}, \alpha)) = -\log p(\mathbf{x}, \alpha). \quad (2.5)$$

For all of these three variations, the same principle for the learning problem holds: solving the problem is analogous to minimizing the risk functional (2.2), where the only information available is given by the training set (2.1).

2.2 Theoretical Foundations

2.2.1 PAC Learning

We now describe Valiant's Probably Approximately Correct (PAC) model, or *distribution free* learning model. We shall now revisit some of these issues and introduce new concepts that together serve as basis for the Vapnik-Chervonenkis theory, which we will use later to describe support vector machines.

According to Cristianini and Shawe-Taylor [CST00], one of the key assumptions of the PAC model is that training and testing data in a learning procedure are generated independently and identically (*i.i.d.*) according to an unknown yet fixed distribution D . There are refinements of the model that consider the distribution D changing over time, or even cases where the learning algorithm may influence the selection of data. These refinements will not be considered in this discussion, in which we assume the *i.i.d.* case.

Since data samples are drawn according to D , we may introduce an error measure for a generic binary classification function f to define the probability of misclassification

of a random example:

$$\text{err}_D(f) = D\{(\mathbf{x}, y) : f(\mathbf{x}) \neq y\}, \quad (2.6)$$

where $\mathbf{x} \in X$ and $y \in Y = \{-1, +1\}$. We refer to this error measure as *risk functional*. The purpose of this analysis is to assert bounds on the error of the hypothesis, for instance in terms of the number of training examples required to obtain a particular level of error. This concept is known as the *sample complexity* of the learning problem.

Consider the selection mechanism of a hypothesis function f from a class F , such that the selection must be based on a set $S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N))$ of N training examples chosen i.i.d. according to D . We may use the PAC model to examine the generalization bounds of this selection mechanism, the learning algorithm itself, through the bound $\epsilon = \epsilon(N, F, \delta)$, where δ is a parameter specified by the algorithm. This bound asserts that with probability of at least $1 - \delta$ over the training sets in S , the generalization error of f is bounded as follows:

$$\text{err}_D(f) \leq \epsilon(N, F, \delta), \quad (2.7)$$

or, in other words, is probably approximately correct [CST00]. Put differently, we may state this inequality in terms of the small probability that the training set gives rise to a hypothesis function with large error:

$$D^n\{S : \text{err}_D(f) > \epsilon(N, F, \delta)\} < \delta. \quad (2.8)$$

2.2.2 Vapnik-Chervonenkis Theory

Given a finite set of hypotheses, we may obtain a bound in the form of inequality (2.7). Assume the selection of f is consistent with the training examples in S . The probability that all N independent examples are consistent with a hypothesis f such that $\text{err}_D(f) > \epsilon$

is bounded by:

$$D^N\{S : f \text{ consistent and } \text{err}_D(f) > \epsilon\} \leq (1 - \epsilon)^N \leq e^{-\epsilon N}. \quad (2.9)$$

We have now that the probability of one single hypothesis in F being consistent with S , even if all $|F|$ hypotheses have large error, is at most $|F|e^{-\epsilon N}$ by the union bound on the probability that one of several events occurs [CST00]. We may now bound the probability of any consistent hypothesis f_S , as in inequality (2.8):

$$D^N\{S : f \text{ consistent and } \text{err}_D(f_S) > \epsilon\} < |F|e^{-\epsilon N} \quad (2.10)$$

In order to keep $|F|e^{-\epsilon N} < \delta$, we have that:

$$\epsilon = \epsilon(N, F, \delta) = \frac{1}{N} \ln \frac{|F|}{\delta}. \quad (2.11)$$

Equation (2.11) demonstrates how the complexity of the function class F , measured by its cardinality, has a direct effect on the error bound, where a $|F|$ too large may lead to overfitting. This result shows that a property relating the true error to the empirical error holds for all hypotheses in F , for which reason it is said to demonstrate *uniform convergence*. According to Cristianini and Shawe-Taylor [CST00], learning theory relies on bounding the difference between empirical and true estimates of error uniformly over the set of hypotheses and conditions that can arise, where the major contribution of the theory by Vapnik and Chervonenkis is to extend such analysis to infinite sets of hypotheses.

The bounding over an infinite set of functions is done by bounding the probability of inequality (2.8) by twice the probability of having zero error on the training examples but, high error on a second random sample $\hat{S}$:

$$\begin{aligned} D^N\{S : \exists f \in F : \text{err}_S(f) = 0, \text{err}_S(f) > \epsilon\} &\leq \\ 2D^{2N}\{S\hat{S} : \exists f \in F : \text{err}_S(f) = 0, \text{err}_{\hat{S}}(f) > \epsilon N/2\}. \end{aligned} \quad (2.12)$$

This bound is an application of Chernoff bounds, provided that $N > 2/\epsilon$ [CST00]. In order to obtain a union bound on the overall probability of the right hand side of inequality (2.12), we define what Vapnik [Vap95] called one of the three milestones of learning theory, the *growth function*:

$$B_F(N) = \max_{(\mathbf{x}_1, \dots, \mathbf{x}_N) \in X^N} |\{(f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)) : f \in F\}|. \quad (2.13)$$

It is interesting to observe that this function may not exceed 2^N since the sets over which the maximum is searched are all subsets of the set of binary sequences of length N . A set of points $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ is said to be *shattered* by F if:

$$\{(h(\mathbf{x}_1), \dots, h(\mathbf{x}_N)) : f \in F\} = \{-1, +1\}^N. \quad (2.14)$$

The growth function is equal to 2^N for all N if there are sets of any size which can be shattered. Finally, consider the case where the largest size of shattered set is h , and where the growth function can be bounded as follows for $N > h$:

$$B_F(N) \leq \left(\frac{eN}{h}\right)^h, \quad (2.15)$$

giving polynomial growth with exponent h . The value h is known as the Vapnik-Chervonenkis (VC) dimension of the class F . The VC dimension is a measure of the richness or flexibility of the function class, and it is often referred to as its capacity [CST00]. We may now rewrite inequality (2.8) as:

$$D^n\{S : \exists f \in F : \text{err}_S(f) = 0, \text{err}_D(f) > \epsilon\} < 2 \left(\frac{2eN}{h}\right)^N 2^{-\epsilon N/2}, \quad (2.16)$$

which results in a PAC bound for any consistent hypothesis f_S :

$$\text{err}_D(f) \leq \epsilon(N, F, \delta) = \frac{1}{N} \left(N \log \frac{2eN}{h} + \log \frac{2}{\delta} \right). \quad (2.17)$$

This demonstrates the *fundamental theorem of learning* due to Vapnik and Chervonenkis [VC71].

Theorem 2.2.1. (Vapnik and Chervonenkis) Let F be a hypothesis space having VC dimension h . For any probability distribution D on $X \times \{-1, +1\}$, with probability $1 - \delta$ over N random examples S , any hypothesis $f \in F$ that is consistent with S has error no more than:

$$\text{err}_D(f) \leq \epsilon(N, F, \delta) = \frac{1}{N} \left(h \log \frac{2eN}{h} + \log \frac{2}{\delta} \right),$$

provided $h \leq N$ and $N > 2/\epsilon$.

VC theory not only provides a distribution free bound on the generalization of a consistent hypothesis, but also shows that the bound is tight up to log factors. This property is shown in the following theorem.

Theorem 2.2.2. Let F be a hypothesis space with finite VC dimension $h \geq 1$. Then for any learning algorithm there exist distributions such that with probability at least δ over N random examples, the error of the hypothesis f returned by the algorithm is at least:

$$\max \left(\frac{h-1}{32N}, \frac{1}{N} \ln \frac{1}{\delta} \right).$$

2.3 Inductive Learning Principles

2.3.1 Empirical Risk Minimization

Now that we have defined the learning problem, we must define learning principles with which one may build learning machines. We start with the Empirical Risk Minimization (ERM) inductive principle. Consider the general form of the risk functional (2.2):

$$R(\alpha) = \int Q(z, \alpha) dF(z), \quad \alpha \in \Lambda, \quad (2.18)$$

where the probability measure $F(z)$ is unknown but the samples $(z_1, \dots, z_\ell)$ are given, each z corresponds to a tuple (x, y) , and the loss function $Q(z, \alpha)$ may take the form of any of the loss functions already described in Equations (2.3), (2.4), and (2.5).

The general risk functional (2.18) is then transformed in the empirical risk functional by making it discrete with respect to the training set (2.1):

$$R_{emp}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(z_i, \alpha), \quad \alpha \in \Lambda. \quad (2.19)$$

Note that there are no probability distributions in Equation (2.19). $R_{emp}(\alpha)$ is fixed for a particular choice of α and for a particular training set (2.1) [Bur98].

Assume that α_0 and α_1 are the optimal parameter sets for the empirical functional risk and functional risk, respectively. We have that $R_{emp}(\alpha_0)$ and $R_{emp}(\alpha_1)$ will converge to the same value as the number of samples N tends to infinity. Moreover, for a given α' and some value η such that $0 \leq \eta \leq 1$, according to Vapnik [Vap95], the following bound holds with probability $1 - \eta$:

$$R(\alpha') \leq R_{emp}(\alpha') + \sqrt{\frac{h(\log(2\ell/h) + 1) - \log(\eta/4)}{\ell}}, \quad (2.20)$$

where h is the Vapnik-Chervonenkis dimension of the class F . Burges [Bur98] refers to inequality (2.20) as *risk bound*, and to the second term on its right hand side as *VC confidence*.

The ERM principle states that, in order to find the function $Q(z, \alpha_0)$ that minimizes the general risk functional (2.18) with an unknown distribution function $F(z)$, one must in fact approximate the function $Q(z, \alpha_0)$ by the function $Q(z, \alpha_\ell)$ that minimizes the empirical risk functional (2.19). Notice that by making the functional (2.19) independent of the unknown distribution function $F(z)$, learning machines built by the ERM principle may hence be used to solve general problems.

The necessary and sufficient conditions for consistency (convergence in probability to the best possible result) of the ERM principle as well as its non-asymptotic bounds on the rate of convergence may be found in [Vap98].

2.3.2 Structural Risk Minimization

Despite being theoretically sound, the ERM principle is not well suited to deal with small data populations. This motivated the development of the so-called Structural Risk Minimization (SRM) inductive principle. The SRM principle, as the ERM principle, also attempts to minimize the empirical risk. The SRM principle, though, simultaneously attempts to minimize the confidence interval given by the bounds on the risk functional [Vap95]. As a result, the SRM principle is able to consider both the quality of the approximation of the data as well as the complexity of the approximating function.

We start the description of the SRM principle based on the concept of Vapnik Chervonienkis (VC) dimension [VC71]. The VC dimension of a set of functions is a measure of the richness or flexibility of this set, and it is often referred to as its capacity [CST00]. The VC dimension of a set of indicator functions $Q(z, \alpha), \alpha \in \Lambda$ is the maximum number h of vectors $z_1, \dots, z_h$ that may be *shattered*, that is, that may be separated in two classes in all 2^h possible ways using functions from the set. The VC dimension of a set of real bounded functions $A \leq Q(z, \alpha) \leq B, \alpha \in \Lambda$ is the VC dimension of the set of corresponding indicators $I(z, \alpha, \beta) = \theta\{Q(z, \alpha) - \beta\}, \alpha \in \Lambda, \beta \in (A, B)$, where θ is the step-function (or sign function).

Consider a set S of (not necessarily indicator) functions $Q(z, \alpha), \alpha \in \Lambda$. Let the *structure* of the set be given by nested subsets $S_k = \{Q(z, \alpha), \alpha \in \Lambda_k\}$, where $S_1 \subset S_2 \subset \dots \subset S_n \dots$. The VC dimension h_k of each set S_k must be finite and $h_1 \leq h_2 \leq \dots \leq h_n \dots$. For a given set of training data $(z_1, \dots, z_\ell)$, the SRM principle will select a function $Q(z, \alpha_\ell^k)$ from the set S_k that minimizes the empirical risk, where k is determined with respect to minimizing both empirical risk and confidence interval. As k decreases, so does the confidence interval while the empirical risk increases. As k increases, the opposite occurs. The SRM principle will set k to the value where the empirical risk and confidence

interval intersect, therefore minimizing the summation of both, which is the bound on the risk.

The VC theory outlined so far applies only when hypotheses are consistent with the training data. This theory may be adapted to allow for a number of errors on the training set by counting the permutations which leave no more errors on the left hand side of the inequality in Theorem (2.2.1). The resulting bound on the generalization error is given in the following theorem.

Theorem 2.3.1. *Let F be a hypothesis space having VC dimension h . For any probability distribution D on $X \times \{-1, +1\}$, with probability $1 - \delta$ over N random examples S , any hypothesis $f \in F$ that makes k errors on the training set S has error no more than:*

$$\text{err}_D(f) \leq \epsilon(N, F, \delta) = \frac{2k}{N} + \frac{4}{N} \left(h \log \frac{2eN}{h} + \log \frac{4}{\delta} \right),$$

provided $h \leq N$.

This theorem suggests that a learning algorithm for a hypothesis class F must minimize the number of training errors, since everything else in the bound has to be fixed by the choice of F . This is the inductive principle known as empirical risk minimization, discussed in Section 2.3.1, since it seeks to empirically minimize the value of the risk functional.

Let us extend the application of Theorem 2.3.1 to a nested sequence of hypothesis classes $F_1 \subset F_2 \subset \dots \subset F_M$ by using δ/M , hence making the probability of any one of the bounds failing to hold to be less than δ . If an ideal hypothesis f_i with minimum training error is searched in each class F_i , then the number of errors k_i over the fixed training set S will satisfy the monotonic decreasing inequality $k_1 \geq k_2 \geq \dots \geq k_M$. Similarly, the VC dimension for each class H_i will also satisfy a monotonic decreasing inequality $h_1 \geq h_2 \geq \dots \geq h_M$. The bound of Theorem 2.3.1 may be used to choose

on the same four components for transductive inference (Figure 2.2). Three of these components, DG , O , E , are identical to the inductive case. The difference lies on the learning machine LM^T , which now not only takes as input the original training set in Equation (2.1), but also a so-called working set

$$(\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}), \quad (2.21)$$

also drawn *i.i.d.* by DG . This set represents the k points of interest where one wishes to estimate the values $\hat{y}^* = f(\mathbf{x}^*, \alpha^*)$ of the unknown functional dependence, where $\mathbf{x}^* \in (\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k})$. Finally, the communication between DG and E becomes restricted, since E , despite being the same as in the inductive model, is now only able to properly estimate values either on the training set in (2.1) or the working set in (2.21). Notice that, as opposed to the inductive case, it is not necessary that the functions of the set $f(\mathbf{x}, \alpha)$, $\alpha \in \Lambda$, $\mathbf{x} \in X$ be continuously defined over X .

In Section 2.1, based on inductive inference, we stated that the solution to the learning problem may be described as finding the function which minimizes the expected risk given by the risk functional (2.2). For transductive inference, the solution to the learning problem, implemented by LM^T , may be described as finding the function which minimizes the *overall risk*. More formally, LM^T must find the function $f(\mathbf{x}, \alpha^*)$ that, with probability $1 - \eta$, minimizes the overall risk of estimating values $\hat{y}^* = f(\mathbf{x}^*, \alpha^*)$ for $\mathbf{x}^* \in (\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k})$. This is analogous to minimizing the functional

$$R_{\Sigma}(\alpha) = \frac{1}{k} \sum_{i=1}^k L(y_{\ell+i}, f(\mathbf{x}_{\ell+i}, \alpha)), \quad (2.22)$$

where $L(y, f(\mathbf{x}, \alpha))$ is a generic loss function.

Notice how the measurement of risk differs from the inductive to the transductive settings of the learning problem. In the inductive case, the risk functional (2.2) assesses the risk with respect to the joint distribution function $F(\mathbf{x}, y)$ over the whole space of the

unknown functional dependence. In the transductive case, the risk functional (2.22) only assesses the risk with respect to the given points of interest. In Chapter 1 we stated that the problem of estimating values at given points was easier for the learning machine than estimating a general rule that approximates a functional dependence. This difference in risk measurement between (2.2) and (2.22) asserts that [Vap98].

2.5 Transductive Learning Principle

Given the transductive learning problem described in Section 2.4, we must now introduce a compatible transductive learning principle in the form of those described in Section 2.3 for the inductive case. The method of structural minimization of the overall risk is, in fact, the same as the inductive method of structural risk minimization. The key difference lies in the *a priori* availability of the given points of interest where one must estimate the values of the unknown functional dependence:

$$\mathbf{x}_1, \dots, \mathbf{x}_\ell, \mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}. \quad (2.23)$$

When estimating functions, one must determine the structure on a (possibly infinite) set of functions $Q(z, \alpha), \alpha \in \Lambda$ given the set itself and the domain of definition of its members. When estimating values at given points, a structure $S_1 \subset S_2 \subset \dots \subset S_n$ on the set $Q(z, \alpha), \alpha \in \Lambda$ may be defined on the equivalence classes of the set. A set is decomposed by a finite number of equivalence classes through a set of indicator functions, where two indicator functions are said to be equivalent when they classify elements of the complete sample given in (2.23) in the same way.

According to [Vap98], given the set of indicator functions with VC dimension h , the additive bound

$$R_\Sigma(\alpha) \leq R_{emp}(\alpha) + \Omega(\ell, k, h, \eta) \quad (2.24)$$

is valid with probability $1 - \eta$. Similarly, for the set of arbitrary functions with VC dimension h , the multiplicative bound

$$R_{\Sigma}(\alpha) \leq R_{emp}(\alpha) \Omega^*(\ell, k, h, \eta) \tag{2.25}$$

is valid with probability $1 - \eta$.

Chapter 3

Support Vector Machines

This chapter contains a brief revision on support vector machines. In Section 3.1 we use a trivial linear discriminant to introduce a basic learning framework later used to describe support vector machines from the bottom up. Section 3.2 introduces the concepts of inner-product kernel functions and learning in kernel-induced feature spaces, which are some of the key features behind the representation power of SVMs. In Section 3.3 we examine formulations of different machines, evolving from simple linear SVMs into non-linear and non-linear soft-margin SVMs. In Section 3.4 we examine transductive formulations of support vector machines. In Section 3.5 we describe different approaches used to train these inductive and transductive learning machines. Ending the chapter, in Section 3.6 we make some considerations about transductive support vector machines.

3.1 Linear Learning Machines

In order to describe advanced non-linear learning machines such as support vector machines, in this section we first explore the basic principles of linear learning machines. These simple concepts form the very basis of many learning algorithms, and historically

have initiated and motivated the field. Simple models such as the McCulloch and Pitts [MP43] model of an artificial neuron are examples of this pioneer work, which was later followed by others such as Hebb's [Heb49], Rosenblatt's [Ros58], and Widrow and Hoff's [WH60].

We start our discussion with a simple display of a linear classification model, equivalent to that known as the McCulloch and Pitts neuron. Later, we extend this model to its dual representation, which will be used in the description of SVMs themselves.

3.1.1 Linear Classification

Consider a binary classification problem with a real input domain $X \in \mathbb{R}^n$, and an output domain $Y \in \{-1, +1\}$. We shall use a classification function $f : X \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ such that the input $\mathbf{x}$ is assigned to the positive class if $f(\mathbf{x}) \geq 0$, and to the negative class otherwise. Considering that f is a linear function of $\mathbf{x} \in X$, we may define f as:

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \\ &= \sum_{i=1}^n w_i x_i + b, \end{aligned} \tag{3.1}$$

where $(\mathbf{w}, b) \in \mathbb{R}^n \times \mathbb{R}$ are the parameters governing f . In the context of the McCulloch and Pitts neuron, $\mathbf{w}$ is known as the weight vector, and b as the threshold value. Since the output solely depends upon an inner product between two vectors exceeding or not the threshold value, this learning machine may only describe a linearly separable problem [DHS01, Hay94]. Geometrically, f may be interpreted as defining a separating hyperplane of dimension $n - 1$ between classes in an n -dimensional space.

Among the many possible ways to determine the optimal values for $(\mathbf{w}, b)$, that is, for training the learning machine, the first iterative algorithm to do so was Rosenblatt's Perceptron [Ros58]. The Perceptron algorithm works by starting with arbitrary initial values for the weight vector and the threshold value. Then, following repeated iterations,

these values are adjusted based on the error values defined as the difference between the output value of the machine and the ideal value expected.

3.1.2 Dual Representation

Strang [Str86] exemplifies duality in a three-dimensional space, where he states that *the minimum distance to the points on a line is equal to the maximum distance to planes through that line*. We refer to the former representation of the same concept as its primal form, and to the latter as its dual form. Dual representations are most useful in optimization problems, where one often attempts to minimize a primal form while at the same time attempting to maximize a dual form. If the problem obeys the strong duality theorem, described in [CST00], both forms will equal once their optimum value are found, that is, there will be no duality gap on the solution.

Consider the linear classification machine and its training algorithm, the Perceptron, described in Section 3.1.1. Cristianini and Shawe-Taylor [CST00] demonstrate, without loss of generality, that if the initial weight vector and threshold value are the zero vector and zero, respectively, the final hypothesis output by the algorithm will be a linear combination of the training points:

$$\hat{\mathbf{w}} = \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i, \quad (3.2)$$

where $\hat{\mathbf{w}}$ is the augmented weight vector $\mathbf{w}$ which incorporates the original threshold value b , and l is the number of iterations run by the algorithm. We may now rewrite the decision function in Equation (3.1) using its dual representation:

$$\begin{aligned} f'(\mathbf{x}) &= \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \\ &= \langle \sum_{i=1}^l \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} \rangle + b \\ &= \sum_{i=1}^l \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b \end{aligned} \quad (3.3)$$

Cristianini and Shawe-Taylor [CST00] describe the primal and dual forms of the Perceptron algorithm pseudo-code, along with an interesting analysis of bounds from learning theory.

3.2 Kernel-Induced Feature Spaces

This section describes what may be seen as one of the main building blocks of support vector machines [CST00], which is the use of kernel-induced feature spaces for transforming the original input space. Haykin [Hay94] also highlights the concept, where he states that the basic idea of a SVM is summarized by the non-linear mapping of an input vector into a high-dimensional feature space that is hidden from both the input and output, where an optimal separating hyperplane is sought later.

3.2.1 Mapping into Feature Spaces

The difficulty of a learning task varies with the complexity of the target function to be learned, which in turn depends on how this function is represented. A common strategy in machine learning is the preprocessing of data, where their representation is changed as follows:

$$\mathbf{x} = (x_1, \dots, x_n) \rightarrow \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_N(\mathbf{x})). \quad (3.4)$$

It is equivalent to say that the input space X was mapped into a new space $F = \{\phi(\mathbf{x}) | \mathbf{x} \in X\}$. The space F is referred to as *feature space*, where the process of finding the most suitable representation for the data is known as *feature selection*. A suitable representation is characterized for making learning easier than in the original input space, for instance by mapping a non-linear problem into a linear one.

There are several approaches for feature selection. Common techniques seek, for

instance, eliminating features which are irrelevant to the problem. Others, such as principal component analysis, attempt to map the original input space into a feature space with less dimensions, thus trying to avoid the so-called *curse of dimensionality* [CST00, Hay94].

Consider the linear learning machine described in Section 3.1.1. We may rewrite Equation (3.1) incorporating mapping into a feature space:

$$f(\mathbf{x}) = \sum_{i=1}^N w_i \phi_i(\mathbf{x}) + b. \quad (3.5)$$

As shown in Section 3.1.2, an important property of linear learning machines is the ability to represent them in their dual form. Hence, we may rewrite Equation (3.3) incorporating mapping into a feature space, which is analogous to the dual form of Equation (3.5):

$$f'(\mathbf{x}) = \sum_{i=1}^l \alpha_i y_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b. \quad (3.6)$$

We now define the *kernel* function, which is a way of directly computing the inner product $\langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle$. Using a kernel function, we are thus able to transform a linear learning machine into a non-linear learning machine. A kernel is a function K , such that for all $\mathbf{x}, \mathbf{z} \in X$:

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle. \quad (3.7)$$

There are interesting remarks about using a kernel function in learning machines. First, a consequence of the dual representation is that the dimension of the feature space need not affect the computation of the kernel. Furthermore, we do not need to know the underlying feature map to be able to learn in the feature space. Also, since feature vectors are not explicitly represented, potential computational problems inherent in evaluating the feature map are automatically overcome.

3.2.2 Building Kernels

Since the number of operations required to compute the inner product evaluated by a kernel function is not necessarily proportional to the number of features, using kernel functions as feature maps is often very interesting in terms of computational resources. The first step in order to use such mapping is to determine the most appropriate kernel function. Even though it may seem more intuitive to build a kernel function starting with building its corresponding feature space, in practice, the approach taken is to define the kernel function directly, hence implicitly defining the feature space. We now state the necessary conditions to ensure that a given function K is a kernel for some feature space.

First of all, a kernel function $K(\mathbf{x}, \mathbf{z})$ must be symmetric:

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle = \langle \phi(\mathbf{z}) \cdot \phi(\mathbf{x}) \rangle = K(\mathbf{z}, \mathbf{x}) \quad (3.8)$$

Also, it should satisfy the inequalities that follow from the Cauchy-Schwarz inequality, that is ¹:

$$\begin{aligned} K(\mathbf{x}, \mathbf{z})^2 &= \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle^2 \leq \|\phi(\mathbf{x})\|^2 \|\phi(\mathbf{z})\|^2 \\ &= \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{x}) \rangle \langle \phi(\mathbf{z}) \cdot \phi(\mathbf{z}) \rangle^2 = K(\mathbf{x}, \mathbf{x}) K(\mathbf{z}, \mathbf{z}) \end{aligned} \quad (3.9)$$

Finally, it may obey Mercer's theorem, which provides a way for characterizing a kernel function, described as follows. Consider a finite input space $X = \{\mathbf{x}_1, \dots, \mathbf{x}_1\}$, and suppose $K(\mathbf{x}, \mathbf{z})$ is a symmetric function on X . Consider the matrix $\mathbf{K}$:

$$\mathbf{k} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n. \quad (3.10)$$

We have that $\mathbf{K}$ is symmetric and there is an orthogonal matrix $\mathbf{V}$ such that $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}'$, where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_k of $\mathbf{K}$ and the corre-

¹Notice that during the text, whenever we write $\|\mathbf{x}\|$, we mean the 2-norm, or Euclidean norm, of vector $\mathbf{x}$. We use $\|\mathbf{x}\|_1$ to refer to the 1-norm of vector $\mathbf{x}$.

sponding eigenvectors $\mathbf{v}_t = (v_{ti})_{i=1}^n$, which are the columns of $\mathbf{V}$. Assuming all eigenvalues are non-negative, consider the feature mapping:

$$\phi : \mathbf{x}_i \rightarrow \left(\sqrt{\lambda_t} v_{ti} \right)_{t=1}^n \in \mathbb{R}^n, i = 1, \dots, n. \quad (3.11)$$

We have now that:

$$\langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle = \sum_{t=1}^n \lambda_t v_{ti} v_{tj} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}'_{ij} = \mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j), \quad (3.12)$$

which implies that $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function with the corresponding feature mapping ϕ . The proof that the eigenvalues λ_i must be non-negative is presented by Cristianini and Shawe-Taylor [CST00], given by contradiction based on their effect on the geometry of the space.

We may generalize an inner product in a Hilbert space by introducing weights α_i for each dimension, hence allowing infinite dimension in the feature vectors:

$$\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}) \quad (3.13)$$

According to Mercer's theorem, a continuous symmetric function $K(\mathbf{x}, \mathbf{z})$ may be represented as:

$$\sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}), \quad (3.14)$$

where all λ_i are non-negative, which is equivalent to $K(\mathbf{x}, \mathbf{z})$ being an inner product in the feature space $F \supseteq \psi(X)$, where F is the l_2 space of all sequences $\Psi = (\psi_1, \dots, \psi_i, \dots)$ for which $\sum_{i=1}^{\infty} \lambda_i \psi_i^2 < \infty$.

This will induce a space defined by the feature vector, and as a consequence a linear decision function. We represent both primal and dual forms of this decision function as follows, where the term on left hand side is the primal and the term on the right hand side is the dual:

$$f(\mathbf{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i \phi_i(\mathbf{x}) + b = \sum_{j=1}^l \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b \quad (3.15)$$

Although we do not address the issue in detail here, the contribution from functional analysis comes from the study of the eigenvalue problem for integral equations of the form:

$$\int_K K(\mathbf{x}, \mathbf{z}) \phi(\mathbf{z}) d\mathbf{z} = \lambda \phi(\mathbf{x}), \quad (3.16)$$

where $K(\mathbf{x}, \mathbf{z})$ is a bounded, symmetric, and positive kernel function and X is a compact space. The details of this analysis are presented by Cristianini and Shawe-Taylor [CST00], along with many other interesting considerations about building kernels that are beyond the scope of this text, such as reproducing kernel Hilbert spaces, building kernels from kernels, and building kernels from features.

We conclude the discussion by stating Mercer's theorem.

Theorem 3.2.1. *(Mercer) Let X be a compact subset of $\mathbb{R}^n$. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(X) \rightarrow L_2(X)$,*

$$(T_K f)(\cdot) = \int_K K(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x},$$

is positive, that is,

$$\int_{X \times X} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0,$$

for all $f \in L_2(X)$. Then we can expand $K(\mathbf{x}, \mathbf{z})$ in a uniformly convergent series (on $X \times X$) in terms of T_K 's eigen-functions $\phi_j \in L_2(X)$, normalized in such a way that $\|\phi_j\|_{L_2} = 1$, and positive associated eigenvalues $\alpha_j > 0$,

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{z}).$$

3.2.3 Examples of Kernels

There are many possible functions that satisfy all requirements necessary to be used as kernels. One of the steps of building a support vector machine, as will be described in

Table 3.1: Summary of commonly used inner-product kernel functions.

Name	$K(\mathbf{x}, \mathbf{x}_i), i = 1, \dots, N$	Comments
Polynomial	$(\mathbf{x} \cdot \mathbf{x}_i + 1)^p$	Power p is a parameter determined <i>a priori</i> .
Radial-basis function (RBF)	$\exp\left(-\frac{\ \mathbf{x}-\mathbf{x}_i\ ^2}{2\sigma^2}\right)$	Width σ^2 is a parameter determined <i>a priori</i> .
Two-layer Perceptron	$\tanh(\beta_0 \mathbf{x} \cdot \mathbf{x}_i + \beta_1)$	Mercer's theorem is satisfied only for some values of β_0 and β_1 .

Section 3.3, is the selection of the most appropriate kernel for the problem in hand. Table 3.1 presents some of the most commonly used kernels in the literature [Bur98, CST00].

3.3 Building Support Vector Machines

In this section we combine the concepts introduced in the previous sections of this chapter to build learning machines known as support vector machines. We start by describing the principles of building a simple SVM for linearly separable problems. Then, using this basic machine as a basis, we incorporate inner-product kernels into the model to make learning possible on the corresponding induced feature spaces. As we shall see later, this learning machine based on finding a separating hyperplane in an induced feature space has the ability to tackle problems with non-linear input spaces. Completing our description of classification machines, we add slack variables to this separating hyperplane in order to cope with noisy data. Though the main focus of this chapter is to describe support vector machines for classification problems, we conclude the section by briefly describing the use of SVMs in regression problems.

3.3.1 Linear Support Vector Machines

Consider the linear learning machine and the binary classification problem described in Section 3.1. Assume that the machine is trained on linearly separable data. Considering an input space with n dimensions, the geometrical interpretation for the decision function $f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$ is a separating hyperplane with $n - 1$ dimensions. Notice that the points $\mathbf{x}$ lying on this hyperplane imply that $f(\mathbf{x}) = 0$, where $\mathbf{w}$ is normal to the hyperplane, $|b|/\|\mathbf{x}\|$ is the perpendicular distance from the hyperplane to the origin, and $\|\mathbf{x}\|$ is the Euclidean norm of $\mathbf{w}$. Let us define γ_+ as the shortest distance from the separating hyperplane to the closest positive example, similarly for γ_- and the closest negative example. We define the important concept of *geometric margin* of a separating hyperplane as $\gamma = \gamma_+ + \gamma_-$. For a linearly separable case, the support vector machine attempts to find the separating hyperplane with the largest margin possible.

Consider the margin of the decision function output as the *functional margin*. Cristianini and Shawe-Taylor [CST00] prove that the geometric margin is the functional margin of a normalized weight vector. Therefore, we may optimize the geometric margin by fixing the functional margin to 1 and minimizing the Euclidean norm of the weight vector. Consider a positive example $\mathbf{x}^+$ and a negative example $\mathbf{x}^-$. If the functional margin is fixed to 1, we have that:

$$\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = +1, \quad (3.17)$$

$$\langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1. \quad (3.18)$$

We now compute the geometric margin γ by normalizing $\mathbf{w}$:

$$\begin{aligned} \gamma &= \frac{1}{2} \left(\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}^+ \right\rangle - \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}^- \right\rangle \right) \\ &= \frac{1}{2\|\mathbf{w}\|} (\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle - \langle \mathbf{w} \cdot \mathbf{x}^- \rangle) \\ &= \frac{1}{\|\mathbf{w}\|}. \end{aligned} \quad (3.19)$$

Therefore, we achieve the maximum margin γ by minimizing the norm of the weight vector $\|\mathbf{w}\|$. Burges [Bur98] proves the same result by determining the maximum margin between two parallel hyperplanes that contain the closest positive and negative examples. The formulation of the problem is given as follows.

Proposition 3.3.1. *Given a linearly separable training sample:*

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)),$$

the hyperplane $(\mathbf{w}, b)$ that solves the optimization problem:

$$\begin{aligned} & \text{minimize}_{\mathbf{w}, b} \quad \langle \mathbf{w} \cdot \mathbf{w} \rangle, \\ & \text{subject to} \quad y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \\ & \quad \quad \quad i = 1, \dots, l, \end{aligned}$$

realizes the maximal margin hyperplane with geometric margin $\gamma = 1/\|\mathbf{w}\|$.

In order to maximize the margin, we choose the approach of using the Lagrangian of the problem:

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1], \quad (3.20)$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. Aside from minimizing this primal form, given that this is a convex quadratic programming (QP) problem, we may also solve it by maximizing its dual form. We first differentiate L_P with respect to $\mathbf{w}$ and b :

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i = \mathbf{0}, \quad (3.21)$$

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^l y_i \alpha_i = 0. \quad (3.22)$$

Since these are equality constraints in the dual formulation, we can substitute them

into the primal form from Equation (3.20) to obtain the dual form:

$$\begin{aligned}
L_D(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^l \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1], \\
&= \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle - \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + \sum_{i=1}^l \alpha_i, \quad (3.23) \\
&= \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle.
\end{aligned}$$

Recall that after differentiating the primal form of the Lagrangian of the linear SVM in Equation (3.21), we may write the weight vector as:

$$\mathbf{w} = \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i. \quad (3.24)$$

The formulation of the dual problem uses this definition of $\mathbf{w}$, and is given after Proposition 3.3.1.

Proposition 3.3.2. *Given a linearly separable training sample:*

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)),$$

and suppose the parameters $\boldsymbol{\alpha}'$ solve the following quadratic optimization (QP) problem:

$$\begin{aligned}
&\text{maximize} \quad L_D(\boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle, \\
&\text{subject to} \quad \sum_{i=1}^l y_i \alpha_i = 0, \\
&\quad \alpha_i \geq 0, i = 1, \dots, l.
\end{aligned}$$

Then the weight vector $\mathbf{w}' = \sum_{i=1}^l y_i \alpha'_i \mathbf{x}_i$ realizes the maximal margin hyperplane with geometric margin:

$$\gamma = \frac{1}{\|\mathbf{w}'\|}.$$

Since the threshold value b is not present in the dual problem, we compute b' using the primal constraints:

$$b' = -\frac{\max_{y_i=-1} (\langle \mathbf{w}' \cdot \mathbf{x}_i \rangle) + \min_{y_i=1} (\langle \mathbf{w}' \cdot \mathbf{x}_i \rangle)}{2} \quad (3.25)$$

We now describe the Karush-Kuhn-Tucker (KKT) conditions for constrained optimization problems. All KKT conditions are satisfied at the solution of the constrained optimization problems for support vector machines, since all constraints in this context are linear [Bur98]. The KKT conditions for the primal form L_P from Equation (3.20) are given below:

$$\frac{\partial L_P(w_v, b, \boldsymbol{\alpha})}{\partial w_v} = w_v - \sum_{i=1}^l \alpha_i y_i x_{iv} = 0 \quad v = 1, \dots, n \quad (3.26)$$

$$\frac{\partial L_P(w_v, b, \boldsymbol{\alpha})}{\partial b} = - \sum_{i=1}^l \alpha_i y_i = 0 \quad (3.27)$$

$$y_i (\langle \mathbf{x}_i \cdot \mathbf{w} \rangle + b) - 1 \geq 0 \quad i = 1, \dots, l \quad (3.28)$$

$$\alpha_i \geq 0 \quad \forall i \quad (3.29)$$

$$\alpha_i (y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1) = 0 \quad \forall i \quad (3.30)$$

According to Burges [Bur98], who in turn relies on the results from Fletcher [Fle87], solving the SVM problem is equivalent to finding a solution to the KKT conditions, which are necessary and sufficient for $\mathbf{w}$, b , and $\boldsymbol{\alpha}$ to be a solution. For a complete reference of the subject, Bazaraa [BSS92] brings a thorough survey of most concepts used so far to model SVMs, such as constrained convex optimization problems, KKT conditions, Lagrangian duality, as well as several algorithms for solving optimization problems.

3.3.2 Non-Linear Support Vector Machines

After introducing the basic support vector machine from Section 3.3.1 for solving linearly separable problems, we now move to the support vector machine that Cristianini and Shawe-Taylor [CST00] considered the easiest to understand and yet the basis of many other more complex SVMs. This classifier relies on the same principles of achieving the maximal margin hyperplane as its linear predecessor, only it does so in a kernel-induced

feature space, hence also allowing for non-linear problems.

Consider an optimal solution α' , $(\mathbf{w}', b)$ for the dual problem in Proposition 3.3.2. According to the KKT complementarity condition in Equation (3.30), we have that:

$$\alpha'_i [y_i (\langle \mathbf{w}' \cdot \mathbf{x}_i \rangle + b') - 1] = 0, \forall i. \quad (3.31)$$

This implies that the non-zero values of α_i are only those whose inputs $\mathbf{x}_i$ have functional margin one and therefore lie closest to the hyperplane. These input vectors are referred to as *support vectors*, since they are the ones that determine the solution of the machine. If we consider the dual representation of the separating hyperplane, we may write the optimal separating hyperplane disregarding non-support vectors:

$$\begin{aligned} \sum_{i=1}^l y_i \alpha'_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b' \\ \sum_{i \in \{\text{support vectors}\}} y_i \alpha'_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b'. \end{aligned} \quad (3.32)$$

Also after the KKT complementarity condition in Equation (3.30), we may write the square of the Euclidean norm of the weight vector alternatively:

$$\begin{aligned} \langle \mathbf{w}' \cdot \mathbf{w}' \rangle &= \sum_{i,j=1}^l y_i y_j \alpha'_i \alpha'_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \\ &= \sum_{j \in \{sv\}} y_j \alpha'_j \sum_{i \in \{sv\}} y_i \alpha'_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \\ &= \sum_{j \in \{sv\}} \alpha'_j (1 - y_j b') \\ &= \sum_{j \in \{sv\}} \alpha'_j. \end{aligned} \quad (3.33)$$

We may now extend Proposition 3.3.2 by writing the solution's geometric margin, first defined in Equation (3.19), in terms of the Lagrange multipliers:

$$\begin{aligned} \gamma &= \frac{1}{\|\mathbf{w}'\|} \\ &= \left(\sum_{j \in \{sv\}} \alpha'_j \right)^{-\frac{1}{2}} \end{aligned} \quad (3.34)$$

Consider a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. If we replace the inner product of input vectors $\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$ from Proposition 3.3.2 with this kernel function, we have that the separating

hyperplane is sought in the feature space defined by $K(\mathbf{x}_i, \mathbf{x}_j)$. Notice that the geometric margin of the solution, defined solely in terms of the Lagrange multipliers, is not altered, despite the change in space. We synthesize these results on the following proposition.

Proposition 3.3.3. *Given a training sample:*

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)),$$

where S is linearly separable in the feature space implicitly defined by the kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, suppose the parameters α' and b' solve the following quadratic optimization (QP) problem:

$$\begin{aligned} \text{maximize} \quad & L_D(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \\ \text{subject to} \quad & \sum_{i=1}^l y_i \alpha_i = 0, \\ & \alpha_i \geq 0, i = 1, \dots, l. \end{aligned}$$

Then the decision rule given by $\text{sign}(f(\mathbf{x}))$, where $f(\mathbf{x}) = \sum_{i=1}^l y_i \alpha'_i K(\mathbf{x}_i, \mathbf{x}) + b'$, is equivalent to the maximal margin hyperplane in the feature space implicitly defined by the kernel $K(\mathbf{x}, \mathbf{z})$, and this hyperplane has geometric margin:

$$\gamma = \left(\sum_{j \in \{sv\}} \alpha'_j \right)^{-\frac{1}{2}}.$$

Notice that since the kernel $K(\mathbf{x}, \mathbf{z})$ must satisfy Mercer's theorem, the optimization problem is convex [CST00]. Hence, one of the properties required for a kernel function to define a feature space is that the maximal margin optimization problem has a unique solution, that is, that there is no duality gap.

We now consider the expected generalization error for the support vector machine described so far in this section. It is interesting to notice the effect that the number of support vectors of a given solution has over its expected generalization, where the fewer support vectors, the better generalization. According to Cristianini and Shawe-Taylor [CST00], this is consistent with the principle known as *Occam's Razor*, which may

be interpreted in this context as favoring more compact representations for classification functions as opposed to more complex ones. We next present a theorem given by these same authors on SVM generalization bounds, after Theorem 2.2.1 from Vapnik and Chervonenkis.

Theorem 3.3.4. *Consider the thresholding real-valued linear functions L with unit weight vectors on an inner product space X . For any probability distribution D on $X \times \{-1, +1\}$, with probability $1 - \delta$ over l random examples S , the maximal margin hyperplane has error no more than:*

$$\text{err}_D(f) \leq \frac{1}{l-d} \left(d \log \frac{el}{d} + \log l\delta \right),$$

where d is the number of support vectors.

3.3.3 Soft Margin Optimization

Even though the non-linear maximal margin classifier described in Section 3.3.2 is the basis for many more advanced support vector machines, its direct applicability suffers from its lack of ability to handle noisy data, which is almost always the case in real-world problems. Since the maximal margin classifiers outputs hypothesis with zero training error, the machine's behavior is dictated by their training data, which in turn may include outliers and idiosyncrasies that jeopardize the quality of the final solution. We now explore the concept of *margin distributions*, where we add slack variables to the maximal margin classifier to allow for its margin constraints to be violated.

Recall from Proposition 3.3.1 the primal optimization problem for the maximal margin case. We introduce slack variables that allow the margin constraints to be violated in

order to optimize the margin slack vector:

$$\begin{aligned}
& \text{minimize}_{\mathbf{w}, b} \quad \langle \mathbf{w} \cdot \mathbf{w} \rangle, \\
& \text{subject to} \quad y_i (\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \dots, l, \\
& \quad \quad \quad \xi_i \geq 0, i = 1, \dots, l.
\end{aligned} \tag{3.35}$$

Cristianini and Shawe-Taylor [CST00] demonstrate the bounds on the generalization error in terms of the 2-norm of the margin slack vector, also called 2-norm *soft margin*, which contains ξ_i scaled by the norm of the weight vector. Their result suggests an optimal choice for C in the optimization problem, as follows.

$$\begin{aligned}
& \text{minimize}_{\xi, \mathbf{w}, b} \quad \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^l \xi_i^2, \\
& \text{subject to} \quad y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \dots, l, \\
& \quad \quad \quad \xi_i \geq 0, i = 1, \dots, l.
\end{aligned} \tag{3.36}$$

The same approach may be adapted for the 1-norm case, where the optimization problem minimizes another combination of the norm of weights and the 1-norm of slack variables. Once there is a value of C that corresponds to the optimal choice of $\|\mathbf{w}\|$, C will give the optimal bound as it will correspond to achieving the minimum of $\|\xi_i\|_1$ with the given value of $\|\mathbf{w}\|$.

$$\begin{aligned}
& \text{minimize}_{\xi, \mathbf{w}, b} \quad \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^l \xi_i, \\
& \text{subject to} \quad y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \dots, l, \\
& \quad \quad \quad \xi_i \geq 0, i = 1, \dots, l.
\end{aligned} \tag{3.37}$$

Consider the case for 2-norm soft margins. We have that the primal Lagrangian for the problem in Equation (3.36) is:

$$L_P(\mathbf{w}, b, \xi, \alpha) = \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle + \frac{C}{2} \sum_{i=1}^l \xi_i^2 - \sum_{i=1}^l \alpha_i [y_i (\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i]. \tag{3.38}$$

If we calculate the partial derivatives for L_P we have:

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i = \mathbf{0}, \quad (3.39)$$

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial \boldsymbol{\xi}} = C \boldsymbol{\xi} - \boldsymbol{\alpha} = \mathbf{0}, \quad (3.40)$$

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^l y_i \alpha_i = 0. \quad (3.41)$$

Since these are equality constraints in the dual formulation, we can substitute them into the primal form from Equation (3.38) to obtain the dual form:

$$L_D(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle - \frac{1}{2C} \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \quad (3.42)$$

Hence, maximizing the dual representation above is equivalent to Proposition 3.3.5, where δ_{ij} is the Kronecker δ defined to be 1 if $i = j$ and 0, and the corresponding KKT complementarity conditions are:

$$\alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] = 0, i = 1, \dots, l. \quad (3.43)$$

Proposition 3.3.5. *Consider classifying a training sample:*

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)),$$

using the feature space implicitly defined by the kernel $K(\mathbf{x}, \mathbf{z})$, and suppose the parameters $\boldsymbol{\alpha}'$ solve the following quadratic optimization (QP) problem:

$$\begin{aligned} & \text{maximize} \quad L_D(\boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j (K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{C} \delta_{ij}), \\ & \text{subject to} \quad \sum_{i=1}^l y_i \alpha_i = 0, \\ & \quad \alpha_i \geq 0, i = 1, \dots, l. \end{aligned}$$

Let $f(\mathbf{x}) = \sum_{i=1}^l y_i \alpha'_i K(\mathbf{x}_i, \mathbf{x}) + b'$, where b' is chosen so that $y_i f(\mathbf{x}_i) = 1 - \alpha'_i / C$ for any i with $\alpha'_i \neq 0$. Then the decision rule given by $\text{sign}(f(\mathbf{x}))$ is equivalent to the hyperplane

in the feature space implicitly defined by the kernel $K(\mathbf{x}, \mathbf{z})$ which solves the optimization problem (3.36), where the slack variables are defined relative to the geometric margin:

$$\gamma = \left(\sum_{j \in \{sv\}} \alpha'_j - \frac{1}{C} \langle \boldsymbol{\alpha}' \cdot \boldsymbol{\alpha}' \rangle \right)^{-\frac{1}{2}}.$$

Proof. The value of b' is chosen using the relation $\alpha_i = C\xi_i$ and by reference to the primal constraints which by the KKT complementarity conditions:

$$\alpha_i [y_i (\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] = 0, i = 1, \dots, l,$$

must be equalities for non-zero α_i . It remains to compute the norm of $\mathbf{w}'$ which defines the size of the geometric margin.

$$\begin{aligned} \langle \mathbf{w}' \cdot \mathbf{w}' \rangle &= \sum_{i,j=1}^l y_i y_j \alpha'_i \alpha'_j K(\mathbf{x}_i \cdot \mathbf{x}_j) \\ &= \sum_{j \in \{sv\}} y_j \alpha'_j \sum_{i \in \{sv\}} y_i \alpha'_i K(\mathbf{x}_i \cdot \mathbf{x}_j) \\ &= \sum_{j \in \{sv\}} \alpha'_j (1 - \xi'_i - y_j b') \\ &= \sum_{j \in \{sv\}} \alpha'_j - \sum_{j \in \{sv\}} \alpha'_j \xi'_j \\ &= \sum_{j \in \{sv\}} \alpha'_j - \frac{1}{C} \langle \boldsymbol{\alpha}' \cdot \boldsymbol{\alpha}' \rangle. \end{aligned}$$

□

According to Cristianini and Shawe-Taylor [CST00], this quadratic optimization problem is equivalent to a simple change in the function:

$$K'(\mathbf{x}, \mathbf{z}) = K(\mathbf{x}, \mathbf{z}) + \frac{1}{C} \delta_{\mathbf{x}}(\mathbf{z}). \quad (3.44)$$

Consider now the case for 1-norm soft margin. We have that the Lagrangian for the 1-norm soft margin optimization problem is:

$$L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \frac{1}{2} \langle \mathbf{w} \cdot \mathbf{w} \rangle + 2 \sum_{i=1}^l \xi_i - \sum_{i=1}^l \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^l r_i \xi_i, \quad (3.45)$$

with $\alpha_i \geq 0$ and $r_i \geq 0$. We then calculate the partial derivatives for L_P :

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l y_i \alpha_i \mathbf{x}_i = \mathbf{0}, \quad (3.46)$$

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r})}{\partial \xi_i} = C - \alpha_i - r_i = 0, \quad (3.47)$$

$$\frac{\partial L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r})}{\partial b} = \sum_{i=1}^l y_i \alpha_i = 0. \quad (3.48)$$

We substitute these relations into the primal form from Equation (3.45) to obtain the dual form:

$$L_D(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle, \quad (3.49)$$

which is identical to that for the maximal margin, except that the constraints $C - \alpha_i - r_i = 0$ and $r_i \geq 0$ enforce $\alpha_i \leq C$, while $\xi_i \neq 0$ only if $r_i = 0$ and therefore $\alpha_i = C$. The corresponding KKT complementarity conditions are therefore:

$$\begin{aligned} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] &= 0, \quad i = 1, \dots, l, \\ \xi_i (\alpha_i - C) &= 0, \quad i = 1, \dots, l. \end{aligned} \quad (3.50)$$

Notice that these KKT conditions imply that there may only be non-zero slack variables if $\alpha_i = C$. Points with non-zero slack variables have geometric margin less than $1/\|\mathbf{w}\|$, where points for which $0 < \alpha_i < C$ lie at distance $1/\|\mathbf{w}\|$ from the separating hyperplane. Proposition 3.3.6 formalizes the problem.

Proposition 3.3.6. *Consider classifying a training sample:*

$$S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)),$$

using the feature space implicitly defined by the kernel $K(\mathbf{x}, \mathbf{z})$, and suppose the parameters $\boldsymbol{\alpha}'$ solve the following quadratic optimization problem:

$$\begin{aligned} \text{maximize} \quad & L_D(\boldsymbol{\alpha}) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \\ \text{subject to} \quad & \sum_{i=1}^l y_i \alpha_i = 0, \\ & C \geq \alpha_i \geq 0, i = 1, \dots, l. \end{aligned}$$

Let $f(\mathbf{x}) = \sum_{i=1}^l y_i \alpha'_i K(\mathbf{x}_i, \mathbf{x}) + b'$, where b' is chosen so that $y_i f(\mathbf{x}_i) = 1$ for any i with $C > \alpha'_i > 0$. Then the decision rule given by $\text{sign}(f(\mathbf{x}))$ is equivalent to the hyperplane in the feature space implicitly defined by the kernel $K(\mathbf{x}, \mathbf{z})$ which solves the optimization problem (3.37), where the slack variables are defined relative to the geometric margin:

$$\gamma = \left(\sum_{j \in \{sv\}} y_i y_j \alpha'_i \alpha'_j K(\mathbf{x}_i, \mathbf{x}_j) \right)^{-\frac{1}{2}}.$$

Proof. The value of b' is chosen using the KKT complementarity which imply that if $C > \alpha'_i > 0$ both $\xi'_i = 0$ and:

$$\alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] = 0.$$

The norm of $\mathbf{w}'$ is clearly given by the expression:

$$\begin{aligned} \langle \mathbf{w}' \cdot \mathbf{w}' \rangle &= \sum_{i,j=1}^l y_i y_j \alpha'_i \alpha'_j K(\mathbf{x}_i \cdot \mathbf{x}_j) \\ &= \sum_{j \in \{sv\}} \sum_{i \in \{sv\}} y_i y_j \alpha'_i \alpha'_j K(\mathbf{x}_i \cdot \mathbf{x}_j). \end{aligned}$$

□

Notice that all α_i are upper bounded by C , which brings up the name *box constraint* since the vector α must lie inside a box with length C in the positive orthant. C is hence a regularization parameter that adjusts the trade-off between accuracy and generalization. Another interesting outcome is that this constraint also ensures that the feasible region is bounded and hence the primal always has a non-empty feasible region. The concepts around box constraints with 1-norm soft margins are explicitly explored by the Sequential Minimization Optimization algorithm, later described in Section 3.5.4.

Pontil and Verri [PV98] explored the effects of the regularization parameter C over the solution of the SVM. They concluded that some support vectors whose Lagrange multipliers are smaller than C , referred to as *margin vectors*, play a distinct role in the composition of the solution. Given the optimal separating hyperplane described by a

decision surface, this hyperplane can be written as a sum of two orthogonal terms, the first depending only on the margin vectors, the second proportional to the regularization parameter. Therefore, this result from Pontil and Verri enables us to better predict the effects of changes in C over the solution.

3.3.4 Support Vector Regression

Although initially proposed for classification problems, support vector machines may also be used on regression problems. The approach of using support vectors, so far only described for the classification case, can be adapted to the regression case as well. This adapted approach is based on the same concepts of learning a non-linear function with a linear learning machine in a kernel-induced feature space. The main difference lies on the loss indicator function. For classification, this function was responsible for controlling the margins of the problem by distinguishing between two classes, but in regression, it must do the same except the margin moves along with the function under regression. For complete references on using SVMs in regression problems, please refer to Burges [Bur98] or Cristianini and Shawe-Taylor [CST00].

Like the classification approach, in regression we seek to optimize the generalization bounds according to a loss function that ignores errors within a certain distance of the correct value. This loss function is referred to as ϵ -insensitive loss function, where there are variables ξ_i that measure the error for each of the training points, and $\xi_i = 0$ if vector i is within the so-called ϵ -insensitive band. There are many choices for this function, for instance, the linear ϵ -insensitive loss and the quadratic ϵ -insensitive loss, which correspond to the 1-norm and 2-norm of the loss vector, respectively.

Let us consider first the linear ϵ -insensitive loss case. In general, loss functions do not penalize training vectors within the limits of the margins. The linear ϵ -insensitive loss

function presents a linear penalty on vectors outside these margins. The function may be given as:

$$L(y, f(\mathbf{x}, \mathbf{w})) = \begin{cases} 0, & \text{if } |y - f(\mathbf{x}, \mathbf{w})| \leq \epsilon \\ |y - f(\mathbf{x}, \mathbf{w})| - \epsilon, & \text{if } |y - f(\mathbf{x}, \mathbf{w})| > \epsilon \end{cases} \quad (3.51)$$

This function will be represented as $|y - f(\mathbf{x}, \mathbf{w})|_\epsilon$, where ϵ corresponds to the insensitive band.

Consider now the quadratic ϵ -insensitive loss case:

$$L(y, f(\mathbf{x}, \mathbf{w})) = \begin{cases} 0, & \text{if } |y - f(\mathbf{x}, \mathbf{w})| \leq \epsilon \\ (|y - f(\mathbf{x}, \mathbf{w})| - \epsilon)^2, & \text{if } |y - f(\mathbf{x}, \mathbf{w})| > \epsilon \end{cases} \quad (3.52)$$

Lets first consider the quadratic ϵ -insensitive and describe the corresponding primal optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{w}\|^2 + C \sum_{i=1}^l (\xi_i^2 + \hat{\xi}_i^2), \\ \text{subject to} \quad & (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - y_i \leq \epsilon + \xi_i, i = 1, \dots, l, \\ & y_i - (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq \epsilon + \hat{\xi}_i, i = 1, \dots, l, \\ & \xi, \hat{\xi}_i \geq 0, i = 1, \dots, l, \end{aligned} \quad (3.53)$$

where two slack variables are introduced, one for exceeding the target value by more than ϵ , and another for being more than ϵ below the target. The dual problem is derived using the same approach as in the classification case, only now taking into account that $\xi_i \hat{\xi}_i = 0$ and therefore that $\alpha_i \hat{\alpha}_i$ holds for the corresponding Lagrange multipliers:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^l y_i (\hat{\alpha}_i - \alpha_i) - \epsilon \sum_{i=1}^l (\hat{\alpha}_i + \alpha_i) \\ & - \frac{1}{2} \sum_{i,j=1}^l (\hat{\alpha}_i - \alpha_i) (\hat{\alpha}_j - \alpha_j) (\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + \frac{1}{C} \delta_{ij}) \\ \text{subject to} \quad & \sum_{i=1}^l (\hat{\alpha}_i - \alpha_i) = 0, \\ & \hat{\alpha}_i \geq 0, \alpha_i \geq 0, i = 1, \dots, l, \end{aligned} \quad (3.54)$$

where the KKT complementarity conditions are:

$$\begin{aligned}
\alpha_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b - y_i - \epsilon - \xi_i) &= 0, \quad i = 1, \dots, l, \\
\hat{\alpha}_i (y_i - \langle \mathbf{w} \cdot \mathbf{x}_i \rangle - b - \epsilon - \hat{\xi}_i) &= 0, \quad i = 1, \dots, l, \\
\xi_i \hat{\xi}_i &= 0, \alpha_i \hat{\alpha}_i, \quad i = 1, \dots, l.
\end{aligned} \tag{3.55}$$

Consider now the linear ϵ -insensitive loss case. We may write the the corresponding primal optimization problem as follows:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l (\xi_i + \hat{\xi}_i), \\
\text{subject to} \quad & (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - y_i \leq \epsilon + \xi_i, i = 1, \dots, l, \\
& y_i - (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq \epsilon + \hat{\xi}_i, i = 1, \dots, l, \\
& \xi_i, \hat{\xi}_i \geq 0, i = 1, \dots, l.
\end{aligned} \tag{3.56}$$

The corresponding dual problem is:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^l y_i (\hat{\alpha}_i - \alpha_i) - \epsilon \sum_{i=1}^l (\hat{\alpha}_i + \alpha_i) \\
& - \frac{1}{2} \sum_{i,j=1}^l (\hat{\alpha}_i - \alpha_i) (\hat{\alpha}_j - \alpha_j) \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \\
\text{subject to} \quad & \sum_{i=1}^l (\hat{\alpha}_i - \alpha_i) = 0, \\
& \hat{\alpha}_i \leq C, \alpha_i \geq 0, i = 1, \dots, l,
\end{aligned} \tag{3.57}$$

where the KKT complementarity conditions are:

$$\begin{aligned}
\alpha_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b - y_i - \epsilon - \xi_i) &= 0, \quad i = 1, \dots, l, \\
\hat{\alpha}_i (y_i - \langle \mathbf{w} \cdot \mathbf{x}_i \rangle - b - \epsilon - \hat{\xi}_i) &= 0, \quad i = 1, \dots, l, \\
\xi_i \hat{\xi}_i &= 0, \alpha_i \hat{\alpha}_i, \quad i = 1, \dots, l. \\
(\alpha_i - C) \xi_i &= 0, (\hat{\alpha}_i - C) \hat{\xi}_i = 0, \quad i = 1, \dots, l.
\end{aligned} \tag{3.58}$$

Notice that if we replace $\langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle$ by $K(\mathbf{x}_i, \mathbf{x}_j)$ in the dual forms for both the quadratic and linear ϵ -insensitive loss cases above, we have machines that learn in the corresponding kernel induced feature space. This is the same technique used before to map input spaces to kernel induced feature spaces in classification problems.

There are different approaches for support vector regression other than using ϵ -insensitive loss functions. One of these approaches explores the case where $\epsilon = 0$ for the quadratic loss function, which corresponds to least squares regression with a weight decay factor. This approach is referred to as ridge regression, and it is thoroughly described by many authors including Cristianini and Shawe-Taylor [CST00].

3.4 Building Transductive Support Vector Machines

In this section we show the extensions of the inductive formulations of support vector machines given in previous sections to build their transductive counterparts. We start by examining the case of pattern recognition, or classification, which is the focus of this work. Later, we overview the use of transductive SVMs in the case of regression estimation.

3.4.1 Transductive Pattern Recognition

We now describe the formulation of transductive support vector machines, as first presented in [Vap99].

Consider a pattern recognition learning problem with the following training data

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \quad \mathbf{x} \in X \subset \mathbb{R}, y \in Y = \{-1, 1\} \quad (3.59)$$

and test data

$$\mathbf{x}_{\ell+1}^*, \dots, \mathbf{x}_{\ell+k}^*. \quad (3.60)$$

The goal, in the linear case, is to find a function from the set of all linear functions

$$y = (\mathbf{w} \cdot \mathbf{x}) + b \quad (3.61)$$

that minimizes the number of errors on the given test set (3.60). In the separable case, a good solution to this problem also classifies the test patterns with labels

$$y_{\ell+1}^*, \dots, y_{\ell+k}^*. \quad (3.62)$$

Furthermore, the complete sequence of training and test data

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), (\mathbf{x}_{\ell+1}^*, y_{\ell+1}^*), \dots, (\mathbf{x}_{\ell+k}^*, y_{\ell+k}^*) \quad (3.63)$$

is separated with the maximal margin by this solution.

In other words, the goal of the problem is to find classifications (3.62) of the test vectors (3.60) for which the optimal hyperplane

$$y = (\mathbf{w}_0^* \cdot \mathbf{x}) + b_0 \quad (3.64)$$

maximizes the margin when it separates both training and test data (3.63), where $\mathbf{w}_0^*$ denotes the optimal hyperplane that correctly classifies the test data (3.60) according to (3.62), that is,

$$\mathbf{w}_0^* = \mathbf{w}_0(y_{\ell+1}^*, \dots, y_{\ell+k}^*). \quad (3.65)$$

More formally, the goal is to find classifications $y_{\ell+1}^*, \dots, y_{\ell+k}^*$ that minimize the functional

$$\Phi(\mathbf{w}_0(y_{\ell+1}^*, \dots, y_{\ell+k}^*)) = \min_{\mathbf{w}_0^*} \frac{1}{2} \|\mathbf{w}_0^*\|^2 \quad (3.66)$$

while respecting the following inequalities:

$$y_i[(\mathbf{x}_i \cdot \mathbf{w}_0^*) + b] \geq 1, \quad i = 1, \dots, \ell, \quad (3.67)$$

$$y_j^*[(\mathbf{x}_j^* \cdot \mathbf{w}_0^*) + b] \geq 1, \quad j = \ell+1, \dots, \ell+k. \quad (3.68)$$

For the non-separable case, we must introduce slack variables to the functional (3.66) (where C and C^* are non-negative constants that control the flexibility of the margin):

$$\Phi(\mathbf{w}_0(y_{\ell+1}^*, \dots, y_{\ell+k}^*), \xi, \xi^*) = \min_{\mathbf{w}_0^*, \xi, \xi^*} \left[\frac{1}{2} \|\mathbf{w}_0^*\|^2 + C \sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=\ell+1}^{\ell+k} \xi_j^* \right]. \quad (3.69)$$

As in the inductive case, we may rewrite the problem in terms of a constrained optimization problem:

$$\begin{aligned}
& \text{minimize} && \langle \mathbf{w} \cdot \mathbf{w} \rangle + C \sum_{i=1}^{\ell} \xi_i + C^* \sum_{j=\ell+1}^{\ell+k} \xi_j^*, \\
& \text{subject to} && y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, i = 1, \dots, \ell, \\
& && y_j^* (\langle \mathbf{w} \cdot \mathbf{x}_j^* \rangle + b) \geq 1 - \xi_j^*, j = \ell + 1, \dots, \ell + k, \\
& && \xi_i \geq 0, i = 1, \dots, \ell, \\
& && \xi_j^* \geq 0, j = \ell + 1, \dots, \ell + k.
\end{aligned} \tag{3.70}$$

Analogously, we solve the problem using Lagrange multipliers and introduce kernel-induced feature spaces (notice that transductive indices j have been shifted from $(\ell + 1, \dots, \ell + k)$ to $(1, \dots, k)$ to ease the reading):

$$\begin{aligned}
& \text{maximize} && L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \sum_{i=1}^{\ell} \alpha_i + \sum_{j=1}^k \alpha_j^* \\
& && - \frac{1}{2} \sum_{i,r=1}^{\ell} y_i y_r \alpha_i \alpha_r K(\mathbf{x}_i, \mathbf{x}_r) \\
& && - \frac{1}{2} \sum_{j,r=1}^k y_j^* y_r^* \alpha_j^* \alpha_r^* K(\mathbf{x}_j^*, \mathbf{x}_r^*) \\
& && - \sum_{j=1}^{\ell} \sum_{r=1}^k y_j y_r^* \alpha_j \alpha_r^* K(\mathbf{x}_j, \mathbf{x}_r^*), \\
& \text{subject to} && 0 \leq \alpha_i \leq C, i = 1, \dots, \ell \\
& && 0 \leq \alpha_j^* \leq C^*, j = \ell + 1, \dots, \ell + k \\
& && \sum_{i=1}^{\ell} \alpha_i y_i + \sum_{j=\ell+1}^{\ell+k} \alpha_j^* y_j^* = 0.
\end{aligned} \tag{3.71}$$

Despite being a transductive setting, the parameters found by solving the problem (3.71) may be used to build a decision rule, much like the inductive case:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot K(\mathbf{x}, \mathbf{x}_i) + \sum_{j=1}^k y_j^* \alpha_j^* \cdot K(\mathbf{x}, \mathbf{x}_j^*) + b \right). \tag{3.72}$$

3.4.2 Transductive Regression

For the case of function regression, transductive inference has an even greater appeal compared to inductive inference. Recall that, in order to solve a learning problem with

transduction, one uses a transductive step instead of both inductive and a deductive steps otherwise necessary with inductive inference. This is the case for either pattern recognition or function regression problems. Notice, however, that for function regression, the inductive step necessarily means one has to estimate the values of an unknown functional dependence in the continuum points of the domain. Therefore, one is trying to solve a far more general problem given limited training data. For function regression, not only is the domain of the unknown functional dependence continuum, but so is its image.

One of the possible approaches for transductive function regression is ridge regression and the leave-one-out procedure. Let us review this approach for the inductive case. Consider training data in the form of (2.1). In order to minimize the loss functional (2.2), one must minimize the following ridge regression functional:

$$R_{emp}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - f(\mathbf{x}_i, \alpha))^2 + \gamma \|\alpha\|^2, \quad \alpha \in \Lambda, \quad (3.73)$$

where γ is fixed non-negative regularization parameter. The problem is to determine the value of γ that minimizes (3.73) as well as the risk functional (2.2) with loss function (2.4).

Since this optimal value of γ cannot be directly determined, one uses the leave-one-out procedure. It consists of removing tuples $(x_i, y_i) \forall (1, \dots, \ell)$ from the training set (2.1), constructing ℓ decision rules with the remaining data, and testing the removed element on each decision rule created. The minimum γ obtained corresponds to final solution. Details of ridge regression and the leave-one-out procedure are given in [CST00].

The transductive version of ridge regression is based on the extended data set in the form of (3.63) with $y \in \mathbb{R}$. The empirical risk functional is then given by taking the

unknown values $y_{\ell+1}^*, \dots, y_{\ell+k}^*$ as variables:

$$R_{emp}(\alpha|y_1^*, \dots, y_k^*) = \frac{1}{\ell + k} \sum_{i=1}^{\ell} (y_i - f(\mathbf{x}_i, \alpha))^2 + \sum_{i=\ell+1}^{\ell+k} (y_i^* - f(\mathbf{x}_i, \alpha))^2 + \gamma \|\alpha\|^2. \quad (3.74)$$

One then uses the leave-one-out procedure, analogously to the inductive case, to find γ that minimizes (3.74). The formalization of the transductive version of the leave-one-out procedure is slightly more complex than the inductive case, as it introduces a new parameter γ^* . These details are given in [CVW99], where the authors also report experiments comparing both inductive and transductive versions. The transductive version is shown to consistently outperform its inductive counterpart for two benchmark problems.

3.5 Training Methods

In previous sections, we described how the problem of training an inductive or transductive support vector machine may be reduced to a constrained optimization problem. Solving constrained optimization problems has been widely studied in the literature, especially for the convex case. In this section we describe methods for training SVMs, starting with well known generic optimization methods, shifting to enhanced methods that take advantage of specific SVM properties.

Despite having a wide range of methods for training SVMs from optimization literature, the problem has particular characteristics that suggest the use of customized approaches. For instance, large training sets impose serious resource restrictions due to the quadratic complexity of the problem in space. The dimensionality of the kernel matrix grows with the square of the training set size, thus potentially generating a substantial number of variables to be optimized. Furthermore, SVM problems are rather sparse, therefore computational resource consumption and performance may be substantially improved if the algorithms being used are able to deal efficiently with sparseness.

3.5.1 General Techniques

We shall now describe some techniques based on generic optimization methods for solving the problem of training classification support vector machines. Consider the problem stated in Proposition 3.3.6, where $C = \infty$ gives the hard margin case, an adaptation of the kernel as described in Equation (3.44) gives 2-norm soft margin optimization, and $C < \infty$ gives the 1-norm soft margin case. We also have that the Karush-Kuhn-Tucker complementarity conditions for the 2-norm and 1-norm soft margin cases are given by Equations (3.43) and (3.50), respectively. Recall that the convexity of the problem ensures that a solution can always be found efficiently, where satisfying all KKT conditions is necessary and sufficient for finding a solution.

There are many numerical approaches to this problem, where algorithms iteratively increase the value of the dual objective function without leaving the feasible region, until a stop criterion is met. Cristianini and Shawe-Taylor [CST00] describe three different stopping criteria derived from the properties of the problem, as we describe next.

1. One of the simplest stop criterion possible is monitoring the growth of the dual objective function. Since there are no local maxima in the problem, the function will reach its maximum value at the solution. Therefore, one may monitor the fractional rate of increase between iterations to fall below a given threshold, and hence stop the iterative training.
2. Since meeting all KKT conditions is necessary and sufficient for a solution, these conditions provide a natural stopping criteria. For the 1-norm soft margin case, we

have that:

$$\begin{aligned}
& 0 \leq \alpha_i \leq C, \\
& y_i f(\mathbf{x}_i) \begin{cases} \geq 1 & \text{for points with } \alpha_i = 0, \\ = 1 & \text{for points with } 0 < \alpha_i < C, \\ \leq 1 & \text{for points with } \alpha_i = C. \end{cases} \quad (3.75)
\end{aligned}$$

For the 2-norm soft margin case, we have that:

$$\begin{aligned}
& \alpha_i \geq 0, \\
& y_i f(\mathbf{x}_i) \begin{cases} \geq 1 & \text{for points with } \alpha_i = 0, \\ = 1 - \alpha_i/C & \text{for points with } \alpha_i > 0, \end{cases} \quad (3.76)
\end{aligned}$$

where slack variables are implicitly defined by $\xi_i = \alpha_i/C$.

3. Considering that the duality gap at the solution is zero in convex quadratic optimization problems, one may monitor progress by assessing the difference between the primal and dual objective functions. The following ratio may be computed and checked against a threshold value as it converges toward zero:

$$\frac{L_P - L_D}{L_P - 1}, \quad (3.77)$$

where L_P is the primal objective function and L_D is the dual objective function.

The simplest method for achieving a numerical solution for a convex optimization problem is using gradient ascent, also known as steepest ascent algorithm. It works by iteratively shifting the solution toward the direction of the steepest ascent by steps s of fixed length, factored using a learning rate η , similarly to many neural network training algorithms [BLC00, Hay94]. The update of Lagrange multipliers at time t is given by:

$$\alpha_i^{t+1} = \alpha_i^t + \eta_i s_i. \quad (3.78)$$

Though very simple to implement, this approach suffers from the general problems associated with naive gradient ascent, such as convergence speed and oscillation before convergence. Cristianini and Shawe-Taylor [CST00] introduce the topic of training SVMs with a thorough description of this method, which is also explored by Almeida et al. [ABB01b].

Another iterative method similar to gradient ascent is that of projected conjugate gradients (PCG). Although the concepts behind both methods are essentially the same, where the solution is achieved by climbing the convex region up to its maximum, the direction of each step is computed in a more sophisticated and efficient way.

Yet another iterative algorithm was proposed by Mangasarian and Musicant [MM98, MM99], called Successive Over Relaxation (SOR). It consists of relaxing some of the equality constraints in the problem formulation, leaving it with only inequality constraints. One of the interesting properties of SOR is its ability to deal with massive datasets [MM99, BMM99]. According to Cristianini and Shawe-Taylor [CST00], SOR is equivalent to stochastic gradient ascent combined with sample selection heuristics such as those described by Platt [Pla98a].

3.5.2 Decomposition and Chunking

One of the drawbacks of the iterative algorithms described in Section 3.5.1 is their computational resource requirements, more specifically the necessity to store the data in the form of a kernel matrix, thus implying quadratic complexity in space. New algorithms emerged attempting to make support vector machine training efficient even for large data sets. One of the techniques derived from this effort is the so-called *active set* or *working set* method in optimization, which consists of discarding inactive constraints to simplify the problem, where selection heuristics are used to determine which constraints

are active and which are not.

One of these heuristics is called *chunking*, where the algorithm starts by training the SVM using an arbitrary subset of the data, also called active or working set. Chunking is analogous to the *divide and conquer* strategy for algorithms, where a problem is broken down into smaller more tractable sub-problems. The SVM is repeatedly trained with new data subsets, where each subset is chosen as to keep the support vectors from the previous iterations together with the points that most violate KKT conditions. The algorithm iterates until a stop criterion is found, where the values of the Lagrange multipliers yield the final solution. This algorithm does not change the complexity requirements of SVM training, where the kernel matrix must only be stored in memory for the working set currently being processed.

Inspired by chunking techniques, Joachims [Joa98] introduced SVM^{light} , which selects the Lagrange multipliers of its active set positioned on the steepest feasible descent direction, determined based on a first order approximation of the objective function. Chunking techniques do not have theoretical convergence proofs, though in practice, they are remarkably efficient for dealing with large data sets.

Another more advanced heuristic based on chunking is sometimes referred to as *decomposition*, introduced by Osuna et al. [OFG97]. Instead of focusing in finding the optimal data set over which the optimizer must be executed, decomposition attempts to optimize the global problem by processing small chunks of data at a time. Osuna et al. proved that the algorithm asymptotically converges if at least one point violating KKT conditions is added to the working set. Osuna et al. also suggest that the number of Lagrange multipliers should be fixed, where for each multiplier added to the working set, another has to be removed.

3.5.3 Other Approaches

Hybrid training approaches have also been tried in the literature. Onoda [ORM00] describes an algorithm based on Boosting [Bre94, FS99, Sch99, Sch02], where a boosted SVM was compared to the classical K Nearest Neighbors algorithm, RBF and MLP neural networks. Maia also discusses different strategies to integrate SVM training to Boosting algorithms in [Mai03].

Other related works also rely on instance pre-selection schemes, for instance those from Almeida et al. [ABB00, ABB01a]. These authors first used an *a priori* cluster selection strategy to accelerate the training of classification SVMs [ABB00]. Later, the same authors created a new training algorithm based on gradient ascent and error-dependent repetition techniques [Mun92, Cac94], in which KKT conditions need not be evaluated. Interestingly, they report generalization errors equal to or better than those from SMO for a couple of problems [ABB01a].

3.5.4 Sequential Minimal Optimization

We now describe the so-called *Sequential Minimal Optimization* (SMO) algorithm [Pla98a, Pla98b], based on the ideas by Osuna et al. [OFG97] about decomposing a quadratic programming (QP) optimization problem into a series of smaller problems in order to decrease the resources needed to execute the algorithm. Platt extended this concept, where the original problem is divided in the smallest possible QP problems, which allows them to be solved analytically rather than iteratively. This way, SMO requires a linear amount of memory and a linear-to-quadratic time in the training set size, whereas other chunking methods, such as projected conjugate gradients chunking, require linear-to-cubic times [Pla98a]. The efficiency of SMO compared to previous SVM training algorithms is regarded as one of the factors that triggered the recent interest that SVMs have received in

different application domains, often outperforming many other traditional techniques.

Recall the problem stated in Proposition 3.3.6, where $C = \infty$ gives the hard margin case, an adaptation of the kernel as described in Equation (3.44) gives 2-norm soft margin optimization, and $C < \infty$ gives the 1-norm soft margin case. The KKT conditions for both 2-norm and 1-norm soft margin cases are given by Equations (3.43) and (3.50), respectively. We have that Lagrange multipliers α_i and α_j must obey a linear equality constraint. Therefore, it follows that the smallest possible QP sub-problem derived from the original problem must optimize two Lagrange multipliers at a time. The main advantage of SMO is that solving the problem for two Lagrange multipliers can be done analytically, thus avoiding expensive numerical iterative algorithms. Also, since the maximum number of multipliers in the working set is two, only a two-by-two matrix is required to store the kernel matrix.

The SMO algorithm may be broken down into three parts: an analytical method for solving the optimization problem for two Lagrange multipliers, a selection heuristic to choose which two Lagrange multipliers to optimize, and a method for computing the threshold value of the SVM. We now describe each one of these components.

Consider the linear constraint relating any two Lagrange multipliers α_1 and α_2 chosen to be optimized such that $\sum_{i=1}^l \alpha_i y_i = 0$ holds. It follows that α_1 and α_2 are constrained in a box defined by $0 \leq \alpha_1, \alpha_2 \leq C$ according to this condition, where α_1 and α_2 must lie in a diagonal line such that:

$$\alpha_1 y_1 + \alpha_2 y_2 = \text{constant}. \quad (3.79)$$

This geometric display offers another means for explaining the minimum number of Lagrange multipliers to be selected, where the algorithm would not be able to fulfill all constraints if only one multiplier was to be updated at a time.

Without loss of generality, the algorithm first computes the second Lagrange multi-

plier α_2 and computes the ends of the diagonal line segment in terms of α_2 . If $y_1 \neq y_2$, then the following bounds apply to α_2 :

$$L = \max(0, \alpha_2^{old} - \alpha_1^{old}), H = \min(C, C + \alpha_2^{old} - \alpha_1^{old}). \quad (3.80)$$

If $y_1 = y_2$, then these bounds become:

$$L = \max(0, \alpha_1^{old} + \alpha_2^{old} - C), H = \min(C, \alpha_1^{old} + \alpha_2^{old}). \quad (3.81)$$

Let E_i be the difference between the machine's output $f(\mathbf{x})$ and the target classification on training points $\mathbf{x}_1$ and $\mathbf{x}_2$, such that:

$$E_i = f(\mathbf{x}_i) - y_i = \left(\sum_{j=1}^l \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + b \right) - y_i, i = 1, 2, \quad (3.82)$$

where $\phi(\cdot)$ is the mapping to the feature space corresponding to the kernel function $K(\cdot, \cdot)$. Consider also the second derivative of the objective function along the diagonal line:

$$\eta = K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2) = \|\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2)\|^2. \quad (3.83)$$

We must now compute the location of the constrained maximum of the objective function while allowing only two Lagrange multipliers to change. Under normal circumstances, there will be a maximum lying along the direction of the linear equality constraint and $\eta < 0$. In this case, we have that the following unconstrained maximum:

$$\alpha_2^{new} = \alpha_2^{old} - \frac{y_2(E_1 - E_2)}{\eta}. \quad (3.84)$$

Next, we find the constrained maximum by clipping the unconstrained maximum to the ends of the line segment:

$$\alpha_2^{new, clipped} = \begin{cases} H & \text{if } \alpha_2^{new} \geq H \\ \alpha_2^{new} & \text{if } L < \alpha_2^{new} < H \\ L & \text{if } \alpha_2^{new} \leq L \end{cases} \quad (3.85)$$

We now compute the first Lagrange multiplier, α_1 , based on $\alpha_2^{new,clipped}$:

$$\alpha_1^{new} = \alpha_1^{old} + y_1 y_2 \left(\alpha_2^{old} - \alpha_2^{new,clipped} \right). \quad (3.86)$$

The second derivative η will not be negative under normal circumstances, though SMO does not fail should this happen. Notice that η may be equal to zero if more than one training point has the same input vector.

Cristianini and Shawe-Taylor [CST00] present the analytic solution for two Lagrange multipliers as a theorem, which they prove based on the partial derivative of the objective function with respect to α_2 and the geometric properties of the problem.

Next, we describe the selection heuristics used by SMO to determine the pairs of Lagrange multipliers to optimize at each iteration. According to Osuna et al. [OFG97], a chunking algorithm will asymptotically converge if points that violate KKT conditions are always added to the working set. It follows that the algorithm tends to converge faster if the points that most violate KKT conditions are selected, since they have larger contributions to the gap between the current and optimal solutions [CST00]. Although always selecting the two points that most violate KKT conditions ensures that the algorithm will converge with less iterations, evaluating these conditions for all points is computationally expensive. Therefore, selection heuristics must be used to select two points with a reasonable contribution toward the solution, though yet computationally cheap to be sought.

Platt uses two nested heuristics for selecting Lagrange multipliers, where the first heuristic selects α_1 , and the second selects α_2 based on the first choice.

First Choice Heuristic The first choice heuristic provides the outer loop of the SMO algorithm, which scans through all points determining those that violate KKT conditions. When found, these points are selected for optimization and handled over

to the second choice heuristic. In order to speed up training, the algorithm only scans points whose Lagrange multipliers α_i satisfy $0 < \alpha_i < C$, that is, those points not on the boundary of the feasible solution. Only after all such points do not violate KKT conditions, the algorithm resumes scanning through the complete list of points, including those on the boundary of the feasible solution.

Second Choice Heuristic The selection of the second Lagrange multiplier must take into consideration the selection of the first, where the pair that yields the greater contribution toward the solution is ideal. In order to quickly determine this pair, SMO evaluates the step size $|E_1 - E_2|$ and selects the pair with its largest value, where error values $E_i \forall i \mid 0 < \alpha_i < C$ are cached by the algorithm to avoid repeated evaluations of the kernel function. It follows that if E_1 is negative, SMO chooses an example with minimum error E_2 , and if E_1 is positive, it chooses an example with maximum E_2 . If this choice of α_2 fails to deliver a significant increase in the dual objective, SMO attempts scans through every non-bound example, and if this new choice still fails to provide a relevant contribution, SMO searches through the complete set.

Notice that all condition checks against KKT conditions are made within a margin ϵ . According to Platt [Pla98a], this loose checking of KKT conditions inhibit the incidence of numerical errors, empirically shortening the convergence time depending on its value.

The error cache used in the heuristics is kept for all values E_i whose Lagrange multipliers lie within the feasible region, that is, $0 < \alpha_i < C$. When a non-bound point is selected for optimization, that is, a point not lying over the bounds of the feasible region, its error value is set to zero. After each optimization step, the error values for every

non-bound multipliers α_k are updated as follows:

$$E_k^{new} = E_k^{old} + y_1 (\alpha_1^{new} - \alpha_1^{old}) K(\mathbf{x}_1, \mathbf{x}_k) + y_2 (\alpha_2^{new,clipped} - \alpha_2^{old}) K(\mathbf{x}_2, \mathbf{x}_k) + b^{old} - b^{new}, \quad (3.87)$$

where b is the threshold value of the SVM. Finally, after solving for all Lagrange multipliers in the optimization problem, we must determine how to compute b . After every step, b is re-evaluated so that KKT conditions are fulfilled for both optimized example selections. A threshold value b_1 is valid when α_1^{new} is not at the bounds, since it forces the SVM to output y_1 given the input vector $\mathbf{x}_1$:

$$b_1 = E_1 + y_1 (\alpha_1^{new} - \alpha_1^{old}) K(\mathbf{x}_1, \mathbf{x}_1) + y_2 (\alpha_2^{new,clipped} - \alpha_2^{old}) K(\mathbf{x}_1, \mathbf{x}_2) + b^{old}. \quad (3.88)$$

Similarly, a threshold value b_2 is valid when α_2^{new} is not at the bounds, since it forces the SVM to output y_2 given the input vector $\mathbf{x}_2$:

$$b_2 = E_2 + y_1 (\alpha_1^{new} - \alpha_1^{old}) K(\mathbf{x}_1, \mathbf{x}_2) + y_2 (\alpha_2^{new,clipped} - \alpha_2^{old}) K(\mathbf{x}_2, \mathbf{x}_2) + b^{old}. \quad (3.89)$$

Whenever both b_1 and b_2 are valid, we have that $b_1 = b_2$. When both α_1 and α_2 lie over the bounds and $L \geq H$, the interval between b_1 and b_2 provides infinite possible thresholds that are consistent the KKT, where SMO sets b to the halfway point $b_1 + \frac{b_2 - b_1}{2}$.

3.6 The Minimax Problem for Transductive Learning

In this section we review several different training algorithms for inductive support vector machine. In the inductive case, training a support vector machine is equivalent to solving a constrained convex optimization problem. There are many algorithms from optimization literature that may be used for this purpose, and the ones that are outstanding use different techniques to take advantage of the particular characteristics of the problem. Nevertheless, even in the case of iterative algorithms, once the the dual problem

(3.42) has been solved and all given constraints have been respected, the resulting parameters are valid and may be used to build a decision rule equivalent to the optimal separating hyperplane.

In the transductive case, though, one is not interested in estimating a functional dependence, but only its values at given points of interest. Although this task does not seem as complex as training an inductive support vector machine, there is the complicating factor of taking into account the working set $(x_{\ell+1}, \dots, x_{\ell+k})$ in the training process.

According to [Vap98], the problem of estimating the labels $(y_{\ell+1}^*, \dots, y_{\ell+k}^*)$ for the working set is analogous to a minimax problem, where the maximum is given by solving quadratic optimization problems and the minimum is taken over all admissible classifications of the working set. Therefore, for the pattern recognition case, there are 2^k possible sets of labels that must be exhaustively examined in order to assure that the exact solution has been found.

It is clear that an exhaustive search for the optimal solution is not possible when k is larger than a few samples. The alternative, then, is to use heuristic procedures to search this solution space with lesser resource consumption, thus making the training of transductive support vector machines possible. As with the constrained convex optimization problem in the inductive case, there are many heuristic techniques in the literature to solve the minimax problem. [RN95] reviews several of these techniques with respect to solving general minimax problems.

Also analogously with the inductive case, the outstanding training heuristics for the transductive case are those that take into account some specific properties of the problem. Among those, the work by Joachims [Joa99] was one of the first to employ transductive support vector machines on a real problem. Based on a text classification problem built with word frequency information, Joachims introduced the TSVM training algo-

rithm based on the same transductive setting given in Equation (3.71), except for the use different margin flexibility constants C_- and C_+ for negative and positive examples, respectively. As most transductive heuristics do, TSVM starts with the inductive solution to the problem. It then iteratively increases the margin flexibility constants. As C_- and C_+ increase, different labels are switched based on these cost factors, and for each change a new convex optimization problem is solved. Despite not presenting convergence proofs, Joachims reports that TSVM is able to successfully train a transductive support vector machine with more than 10,000 points of interest.

Chen et al. introduced a modification to the TSVM algorithm, called progressive transductive support vector machine (PTSVM), eliminating the need to estimate the ratio of positive and negative examples in the working set [CWD03]. Interestingly, PTSVM results presented better performance than TSVM in selected contexts both in terms of accuracy and training time. As the authors point out in their work, this suggests that further research into different heuristics based on the setting in Equation (3.71) may yield new methods with even better performance.

Chapter 4

Information Theory

This chapter briefly reviews Information Theory. It is not the goal of the following sections the thorough revision of Information Theory and all of its many ramifications. Instead, the text covers the concepts used further ahead in this work, in sufficient depth for them to be recognized and acknowledged when employed. Details on information theory may be found in several references, including [Yeu02], [Mac03], and [Hay94]. We start in Section 4.1 with an overview of information theory, followed in Section 4.2 with remarks about units for information quantification. The chapter ends with Section 4.3, which brings a description of several measures of information used in this work.

4.1 Overview

Information theory is one of the foundations of modern communications. It is generally considered to have been born after the work of Claude Shannon in 1948 [Sha48], which tackled the problem of transmitting information over a noisy channel. This work introduced concepts such as *information entropy*, the measure of uncertainty associated with a random variable, used in the chapters ahead. Also, it presented two theorems widely

used in many disciplines, namely the source coding theorem, which defines the number of *bits* required to represent the result of an uncertain event, and the noisy-channel coding theorem, which states that error-free communication is possible in a noisy channel up to a given maximum rate.

Information theory has evolved much since its introduction, and its applications are vastly present in our daily lives, from digital encoding and data communication to applications in statistics, cryptography, computer science, among other fields. Specifically in relation with this work, we have seen an increasing number of applications of information theory in machine learning. Many different approaches to the learning problem are based on entropic measures of information, for instance [GB05], [EP01], and [NLM99].

4.2 Units of Information

The most commonly used unit of information is the *bit* (for **binary digit**), which may only take values 0 or 1. The *bit* is the smallest possible amount of information addressed by a digital computer. It was introduced by Shannon in 1948 [Sha48]. Due to the information technology revolution seen in the last decades, the *bit* and its applications have become extremely popular, and the *bit* has been generally acknowledged as the fundamental unit of modern computing.

The logarithmic base of the formulae in information theory determines the unit of information entropy. Applications based on the *bit* are based on logarithms of base 2 and powers of 2. Although the *bit* is definitely the most used unit, there are others. The *hartley* (or *ban*), named after Ralph Hartley for his work on the transmission of information [Har28], is based on base 10 logarithms and powers of 10. The *nat* (for **natural digit**, sometimes also *nit*), is the unit based on natural logarithms and powers

of e . For reference, one *nat* is approximately equal to 1.44 *bits* ($\frac{1}{\ln 2}$) or 0.43 *hartleys* ($\frac{1}{\ln 10}$). The *nat* is the natural unit for information entropy, corresponding to Boltzmann's definition of thermodynamic entropy $S = k \ln \Omega$ [HRK96, Cro03].

In this work, the *nat* is the chosen unit for all measures of information. The following sections present formulae for information measures in their generic form – with no bases in logarithms. In their specific application, the natural base should be considered.

4.3 Information Measures

There are many entropic measures of information, that is, generalized concepts which quantify the amount of information contained in given data as well as relationships between two or more sets of data. In the next sections we review the most fundamental measures of information, including those used in this work.

For these next sections, assume that X and Y are two random variables, $p(x)$ and $p(y)$ are their respective marginal distributions, and $p(x, y)$ is their joint distribution.

4.3.1 Self-information

Self-information is the amount of information associated with a probabilistic event x , that is, the outcome of a random variable X being equal to x . Self-information depends only on the probability of occurrence of this event $P(X = x)$, as shown by its definition:

$$I(x) = -\log p(x), \quad (4.1)$$

where $p(x) = P(X = x)$.

The self-information of two mutually independent events is the sum of the self-information of both events. Self-information is sometimes referred to as *surprisal*, as it represents the quantification of the surprise of having a given outcome.

4.3.2 Entropy

The information entropy of a random variable X is the expectation of its self-information over all of its possible outcomes:

$$H(X) = \mathbb{E}_X\{I(x)\}. \quad (4.2)$$

For a discrete random variable with n possible outcomes $x_1, \dots, x_n$, the information entropy is defined as:

$$H(X) = \sum_{x \in X} p(x) I(x) = - \sum_{x \in X} p(x) \log p(x). \quad (4.3)$$

Notice that, according to this definition, the entropy of a discrete random variable may never be negative. Furthermore, it is symmetric about the order of possible outcomes $x_1, \dots, x_n$ in X .

The entropy of a discrete random variable is a quantification of the uncertainty associated with its outcome. The more uncertain the outcome of a random variable, the larger its entropy. The case where all possible outcomes of a random variable are equiprobable is the case where entropy is maximized: $H(X) = - \sum_{i=1}^n \frac{1}{n} \log \frac{1}{n} = \log n$. The Jensen inequality follows from this property:

$$H(X) = \mathbb{E}_X \left\{ \log \left(\frac{1}{p(X)} \right) \right\} \leq - \log (\mathbb{E} \{p(X)\}) \leq \log(n). \quad (4.4)$$

4.3.3 Differential Entropy

The original information entropy proposed by Shannon [Sha48] only considered finite outcomes for a given probabilistic event. It is possible to extend the definition from Equation (4.3) to the continuous case, therefore yielding the so-called differential entropy, or continuous entropy:

$$h(X) = - \int_X p(x) \log p(x) dx. \quad (4.5)$$

This definition is reached by partitioning the range of X into equal bins of length Δ with its associated samples points $i\Delta$ within the bins, and then making $\Delta \rightarrow 0$. The quantized version of X may be defined as $X_\Delta = i\Delta$ if $i\Delta \leq X \leq (i+1)\Delta$. There is a value x_i in each bin such that

$$p(x_i)\Delta = \int_{i\Delta}^{(i+1)\Delta} p(x)dx. \quad (4.6)$$

We then approximate the integral of $p(x)$ using the Riemann integral:

$$\int_{-\infty}^{\infty} p(x)dx = \lim_{\Delta \rightarrow 0} \sum_{i=-\infty}^{\infty} p(x_i)\Delta \quad (4.7)$$

where we inherit the property that $\Delta \rightarrow 0$.

The next step is to calculate the continuous estimate of H , denoted H^Δ , based on Equation (4.3):

$$\begin{aligned} H^\Delta &= - \sum_{i=-\infty}^{\infty} \Delta p(x_i) \log \Delta p(x_i) \\ &= - \sum_{i=-\infty}^{\infty} \Delta p(x_i) \log p(x_i) - \sum_{i=-\infty}^{\infty} p(x_i)\Delta \log \Delta p(x_i). \end{aligned} \quad (4.8)$$

As $\Delta \rightarrow 0$, we have that

$$\sum_{i=-\infty}^{\infty} p(x_i)\Delta \rightarrow \int p(x)dx = 1, \quad (4.9)$$

which implies

$$\sum_{i=-\infty}^{\infty} \Delta p(x_i) \log \Delta p(x_i) \rightarrow \int p(x) \log p(x)dx. \quad (4.10)$$

Notice, however, that $\log \Delta \rightarrow -\infty$ as $\Delta \rightarrow 0$. Therefore, we finally define the differential entropy as:

$$h(X) = \lim_{\Delta \rightarrow 0} (H^\Delta + \log \Delta) = - \int_{-\infty}^{\infty} p(x) \log p(x)dx. \quad (4.11)$$

This shows an important qualitative property of differential entropy. It is not defined, nor can it be seen, as the limit on the discrete entropy from Equation (4.3) as $n \rightarrow \infty$.

Instead, the differential entropy, or continuous entropy, is an information measure for continuous random variables.

Differential entropy differs somewhat from its originating discrete cousin. Among its cons, it may take negative values, which are of little significance to quantify information. Nevertheless, differential entropy has another outstanding qualitative property: given its definition from Equation (4.5), the differential entropy of any continuous random variable may be calculated solely based on its underlying probability distribution. This means that if we know the probability density function behind any data, we may calculate its entropy regardless of the data themselves. Table 4.1 brings a list of some of the most common continuous probability distributions, along with their probability density functions and associated entropy definitions.

This important property is used further ahead in this work, where we use the properties of the underlying distributions behind training and working data to calculate their entropic measures.

4.3.4 Joint Entropy

The joint entropy of two discrete random variables X and Y is simply the entropy of pair (X, Y) , that is, the joint entropy over the set of possible outcomes (x, y) for which the probability $p(x, y)$ is defined. Analogously, the joint entropy is the expectation of the self-information over all possible outcomes:

$$H(X, Y) = \mathbb{E}_{X,Y}[-\log p(x, y)] = - \sum_{x,y} p(x, y) \log p(x, y). \quad (4.12)$$

It follows that $H(X, Y) \geq 0$, as in the entropy for a single discrete random variables. Furthermore, the following property holds:

$$H(X, Y) \leq H(X) + H(Y) \quad (4.13)$$

Table 4.1: Continuous entropy expressions for known probability distributions. [Wik07]

Distribution	Probability Density Function	Entropy (in nats)
Uniform	$f(x) = \frac{1}{b-a}$ for $a \leq x \leq b$	$\ln(b-a)$
Normal	$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\ln(\sigma\sqrt{2\pi}e)$
Exponential	$f(x) = \lambda \exp(-\lambda x)$	$1 - \ln \lambda$
Rayleigh	$f(x) = \frac{x}{b^2} \exp\left(-\frac{x^2}{2b^2}\right)$	$1 + \ln \frac{\beta}{\sqrt{2}} + \frac{\gamma}{2}$
Beta	$f(x) = \frac{x^{p-1}(1-x)^{q-1}}{B(p,q)}$ for $0 \leq x \leq 1$	$\ln B(p,q) - (p-1)[\psi(p) - \psi(p+q)] - (q-1)[\psi(q) - \psi(p+q)]$
Cauchy	$f(x) = \frac{\lambda}{\pi} \frac{1}{\lambda^2 + x^2}$	$\ln(4\pi\lambda)$
Chi	$f(x) = \frac{2}{2^{n/2}\sigma^n\Gamma(n/2)} x^{n-1} \exp\left(-\frac{x^2}{2\sigma^2}\right)$	$\ln \frac{\sigma\Gamma(n/2)}{\sqrt{2}} - \frac{n-1}{2}\psi\left(\frac{n}{2}\right) + \frac{n}{2}$
Chi-squared	$f(x) = \frac{1}{2^{n/2}\sigma^n\Gamma(n/2)} x^{\frac{n}{2}-1} \exp\left(-\frac{x}{2\sigma^2}\right)$	$\ln 2\sigma^2\Gamma\left(\frac{n}{2}\right) - \left(1 - \frac{n}{2}\right)\psi\left(\frac{n}{2}\right) + \frac{n}{2}$
Erlang	$f(x) = \frac{\beta^n}{(n-1)!} x^{n-1} \exp(-\beta x)$	$(1-n)\psi(n) + \ln \frac{\Gamma(n)}{\beta} + n$
F	$f(x) = \frac{\frac{n_1}{2} \frac{n_2}{2}}{B\left(\frac{n_1}{2}, \frac{n_2}{2}\right)} \frac{x^{\frac{n_1}{2}-1}}{(n_2+n_1x)^{\frac{n_1+n_2}{2}}}$	$\ln \frac{n_1}{n_2} B\left(\frac{n_1}{2}, \frac{n_2}{2}\right) + \left(1 - \frac{n_1}{2}\right)\psi\left(\frac{n_1}{2}\right) - \left(1 + \frac{n_2}{2}\right)\psi\left(\frac{n_2}{2}\right) + \frac{n_1+n_2}{2}\psi\left(\frac{n_1+n_2}{2}\right)$
Gamma	$f(x) = \frac{x^{\alpha-1} \exp(-\frac{x}{\beta})}{\beta^\alpha \Gamma(\alpha)}$	$\ln(\beta\Gamma(\alpha)) + (1-\alpha)\psi(\alpha) + \alpha$
Laplace	$f(x) = \frac{1}{2\lambda} \exp\left(-\frac{ x-\theta }{\lambda}\right)$	$1 + \ln(2\lambda)$
Logistic	$f(x) = \frac{e^{-x}}{(1+e^{-x})^2}$	2
Lognormal	$f(x) = \frac{1}{\sigma x \sqrt{2\pi}} \exp\left(-\frac{(\ln x - m)^2}{2\sigma^2}\right)$	$m + \frac{1}{2} \ln(2\pi e \sigma^2)$
Maxwell-Boltzmann	$f(x) = 4\pi^{-\frac{1}{2}} \beta^{\frac{3}{2}} x^2 \exp(-\beta x^2)$	$\frac{1}{2} \ln \frac{\pi}{\beta} + \gamma - 1/2$
Generalized normal	$f(x) = \frac{2\beta^{\frac{\alpha}{2}}}{\Gamma(\frac{\alpha}{2})} x^{\alpha-1} \exp(-\beta x^2)$	$\ln \frac{\Gamma(\alpha/2)}{2\beta^{\frac{\alpha}{2}}} - \frac{\alpha-1}{2}\psi\left(\frac{\alpha}{2}\right) + \frac{\alpha}{2}$
Pareto	$f(x) = \frac{ak^a}{x^{a+1}}$	$\ln \frac{k}{a} + 1 + \frac{1}{a}$
Student's t	$f(x) = \frac{(1+x^2/n)^{-\frac{n+1}{2}}}{\sqrt{n} B\left(\frac{1}{2}, \frac{n}{2}\right)}$	$\frac{n+1}{2}\psi\left(\frac{n+1}{2}\right) - \psi\left(\frac{n}{2}\right) + \ln \sqrt{n} B\left(\frac{1}{2}, \frac{n}{2}\right)$
Triangular	$f(x) = \begin{cases} \frac{2x}{a} & 0 \leq x \leq a \\ \frac{2(1-x)}{1-a} & a \leq x \leq 1 \end{cases}$	$\frac{1}{2} - \ln 2$
Weibull	$f(x) = \frac{c}{\alpha} x^{c-1} \exp\left(-\frac{x^c}{\alpha}\right)$	$\frac{(c-1)\gamma}{c} + \ln \frac{\alpha^{1/c}}{c} + 1$
Multivariate normal	$f_X(x_1, \dots, x_N) = \frac{1}{(2\pi)^{N/2} \Sigma ^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$	$\frac{1}{2} \ln\{(2\pi e)^N \det(\Sigma)\}$

where this inequality becomes an equality if and only if X and Y are independent random variables.

The same concept of differential entropy may be applied to calculate the joint entropy of two continuous random variables. The rationale is the same as described in Section 4.3.3, extended to the joint probability between two variables. The definition of the differential joint entropy is hence given by:

$$h(X, Y) = - \int_Y \int_X p(x, y) \log p(x, y) dx dy. \quad (4.14)$$

4.3.5 Conditional Entropy

The conditional entropy $H(X|Y)$ (sometimes called equivocation), is a measure of the remaining entropy of X given that Y is known. Its definition is analogous to the Shannon entropy, except that it is based on the conditional probability of x given y . We start with the definition of a given value $y \in Y$, that is:

$$H(X|y) = E_{X|y}[-\log p(x|y)] = - \sum_{x \in X} p(x|y) \log p(x|y), \quad (4.15)$$

where $p(x|y) = \frac{p(x, y)}{p(y)}$.

The definition of conditional entropy immediately follows:

$$\begin{aligned} H(X|Y) &= E_Y\{H(X|y)\} = - \sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log p(x|y) \\ &= \sum_{x, y} p(x, y) \log \frac{p(y)}{p(x, y)}. \end{aligned} \quad (4.16)$$

From this definition and Bayes' theorem we have that:

$$H(X|Y) = H(X, Y) - H(Y). \quad (4.17)$$

This property illustrates well the concept of conditional entropy, where the information contained in $H(X|Y)$ is equal to the information of both variables, expressed by

the joint entropy $H(X, Y)$, minus the information learned from variable Y , which is its entropy $H(Y)$. Notice that $H(X|Y) = 0$ if and only if the value of X is completely determined by Y . Analogously, we have that $H(X|Y) = H(X)$ if and only if X and Y are independent random variables.

The concept of conditional entropy may also be extended to continuous random variables. As in the discrete case, we start by defining:

$$h(X|y) = - \int_X p(x|y) \log p(x|y) dx. \quad (4.18)$$

The definition of differential conditional entropy then follows:

$$\begin{aligned} h(X|Y) &= - \int_Y p(y) \left(\int_X p(x|y) \log p(x|y) dx \right) dy \\ &= \int_Y \int_X p(x, y) \log \frac{p(y)}{p(x, y)} dx dy. \end{aligned} \quad (4.19)$$

4.3.6 Mutual Information

Mutual information, or transinformation, is a measure of dependency between two random variables. In other words, mutual information measures the amount of information that two random variables share. Therefore, if two random variables X and Y are independent, we have that $I(X; Y) = 0$. Analogously, we have that if $X = Y$, $I(X; Y) = H(X) = H(Y)$. Mutual information may be expressed as other information quantities, for instance:

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X, Y). \end{aligned} \quad (4.20)$$

From these properties, it follows that $I(X; Y) \geq 0$. It also follows that mutual information is symmetric, that is, $I(X; Y) = I(Y; X)$.

Mutual information may be explicitly defined as:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}, \quad (4.21)$$

where $p(x, y)$ is the joint probability distribution of X and Y , and $p(x)$ and $p(y)$ their corresponding marginal probability distributions.

As with other information quantities, the concept of mutual information may be extended to the continuous case, where we define the differential mutual information as:

$$I(X; Y) = \int_Y \int_X p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy. \quad (4.22)$$

Finally, mutual information may be defined in terms of the Kullback-Leibler divergence, described in Section 4.3.8, between a given joint probability distribution $p(x, y)$ and its corresponding marginal distributions $p(x)$ and $p(y)$, so that

$$I(X; Y) = D_{KL}(p(x, y) \| p(x)p(y)). \quad (4.23)$$

4.3.7 Cross Entropy

Cross entropy is a measure of the information required to identify an event from a set of all possible events, given that the coding scheme used is based on a different probability distribution q , rather than the correct probability distribution p . Cross entropy may be defined as:

$$\check{H}(p, q) = \mathbb{E}_p[-\log q] = - \sum_{x \in X} p(x) \log q(x). \quad (4.24)$$

From this definition, it follows that cross entropy may be expressed in terms of the Kullback-Leibler divergence, described ahead in Section 4.3.8:

$$\check{H}(p, q) = H(p) + D_{KL}(p \| q) \quad (4.25)$$

The continuous extension of cross entropy also exists, and can be expressed as:

$$\check{h}(p, q) = - \int_X p(x) \log q(x) dx. \quad (4.26)$$

In the literature, the cross entropy is often referred to as $H(p, q)$, which is the same notation generally accepted for the joint entropy. In order to avoid ambiguity, in this work is adopt the $\check{H}(p, q)$ notation.

Among the many applications of cross entropy, we highlight the so-called cross entropy (CE) method, a general Monte Carlo approach to combinatorial and continuous multi-extremal optimization and importance sampling [dBKMR05, MPR05]. Furthermore, there is the principle of Minimum Cross Entropy (MCE), or Minxent, which has been extensively used in this work. In the literature, the MCE principle is sometimes called the principle of Minimum Discrimination Information Divergence [Wik07], based on the minimization of the Kullback-Leibler divergence, which is slightly different. We shall examine this principle in more detail ahead in Section 4.3.8, where we define the Kullback-Leibler divergence.

4.3.8 Kullback-Leibler Divergence

The Kullback-Leibler divergence (also referred to as information divergence, information gain, or relative entropy), measures the difference between a *true* probability distribution P and an arbitrary probability distribution Q . Notice that it is a *divergence*, as opposed to a distance, since it is not symmetric nor does it obey the triangle inequality.

The Kullback-Leibler divergence has been extensively used in this work to assess the difference between two probability distributions, in our case those underlying the two classes of data from binary classification problems. We therefore describe it in a little more detail as we did for the other information quantities previously examined.

Definitions and Properties

Suppose a discrete random variable with two probability distributions P and Q , where P is the *true* distribution, and Q is an arbitrary one. The definition of Kullback-Leibler divergence is then given by:

$$D_{KL}(P\|Q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}. \quad (4.27)$$

Notice that P usually represents a reference distribution or empirical data, whereas Q represents an approximation of P or a different model.

The Kullback-Leibler divergence is also defined for continuous random variables:

$$D_{KL}(P\|Q) = \int_X p(x) \log \frac{p(x)}{q(x)} dx. \quad (4.28)$$

As described in the previous sections, the Kullback-Leibler divergence is intimately related to other quantities in information theory. First, it may be defined in terms of the cross entropy between P and Q , and the entropy of P , so that

$$D_{KL}(P\|Q) = \check{H}(P, Q) - H(P). \quad (4.29)$$

Also, it follows that the Kullback-Leibler is always non-negative, which is expressed by the so-called Gibbs' inequality:

$$D_{KL}(P\|Q) \geq 0. \quad (4.30)$$

Notice that $D_{KL}(P\|Q) = 0$ if and only if $P = Q$, that is, the entropy $H(P)$ is the minimum value that the cross entropy $\check{H}(P, Q)$ may take.

The Kullback-Leibler divergence may be used to express the mutual information:

$$\begin{aligned} I(X; Y) &= D_{KL}(P(X, Y) \| P(X)P(Y)) \\ &= \mathbb{E}_X \{ D_{KL}(P(Y|X) \| P(Y)) \} \\ &= \mathbb{E}_Y \{ D_{KL}(P(X|Y) \| P(X)) \}. \end{aligned} \quad (4.31)$$

Similarly, it may define entropy itself based on the divergence between the true distribution and a uniform distribution:

$$H(X) = \log N - D_{KL}(P(X) \| P_U(X)). \quad (4.32)$$

Finally, it may define conditional entropy in terms of the expectation of the divergence between the true distribution and a uniform distribution as follows:

$$\begin{aligned} H(X|Y) &= \log N - D_{KL}(P(X, Y) \| P_U(X)P(Y)) \\ &= \log N - D_{KL}(P(X, Y) \| P(X)P(Y)) - D_{KL}(P(X) \| P_U(X)) \\ &= H(X) - I(X; Y) \\ &= \log N - \mathbb{E}_Y \{ D_{KL}(P(X|Y) \| P_U(X)) \}. \end{aligned} \quad (4.33)$$

Multivariate Gaussian

Recall from Section 4.3.3 that the differential entropy of distributions may be calculated if we know their probability density function. For instance, as given in Table 4.1, we have that the closed-form differential entropy expression for a continuous multivariate Gaussian distribution is given by:

$$h(P) = \frac{1}{2} \log \{ (2\pi e)^N |\Sigma| \}, \quad (4.34)$$

where $|\Sigma|$ is the determinant of the covariance matrix Σ , and N is the number of dimensions of the multivariate distribution.

Similarly, the Kullback-Leibler divergence $D_{KL}(P \| Q)$ may also be calculated based on two probability density functions $p(x)$ and $q(x)$ for P and Q respectively. If we assume that $p(x)$ and $q(x)$ are Gaussian, we have the following closed-form expression for the Kullback-Leibler divergence between two continuous multivariate Gaussian distributions:

$$D_{KL}(P \| Q) = \frac{1}{2} \left(\log \left(\frac{|\Sigma_Q|}{|\Sigma_P|} \right) + \text{tr}(\Sigma_Q^{-1} \Sigma_P) + (\mu_Q - \mu_P)^T \Sigma_Q^{-1} (\mu_Q - \mu_P) - N \right), \quad (4.35)$$

where $P \sim N_N(\mu_P, \Sigma_P)$ and $Q \sim N_N(\mu_Q, \Sigma_Q)$ are two N -dimensional Gaussian distributions, and tr denotes the trace operation. This result is widely used throughout this work, as detailed further ahead in the text.

Principle of Minimum Discrimination Information

The principle of Minimum Discrimination Information (MDI) was introduced by Kullback [Kul72] in order to use the divergence between two distributions as discrimination information. Suppose new facts about a distribution P have been discovered. We may calculate the new distribution P' to that which minimizes the Kullback-Leibler divergence $D_{KL}(P\|P')$.

In this work, we use the principle of Minimum Discrimination Information to discriminate between the distributions underlying classes of training and working data. The general idea is to assess the divergences $D_{KL}(P\|P')$ and $D_{KL}(Q\|Q')$, where P' and Q' incorporate newly discovered facts. These facts are then classified and permanently incorporated into P or Q depending on which distribution yielded the smallest divergence. This procedure shall be covered in detail in Section 7.1.

The principle of Minimum Discrimination Information is sometimes mistaken by the principle of Minimum Cross Entropy (MCE) in the literature. We shall recall that $\check{H}(p, q) = H(p) + D_{KL}(p\|q)$. If one has fixed p and is minimizing the cross entropy $\check{H}(p, q)$ in order to find q , the two principles are equivalent since the entropy $H(p)$ is invariant. On the other hand, if q is invariant and p is being sought, the two principles are not equivalent. According to [Wik07], this ambiguity has led to some inconsistencies, where authors attempted to redefine cross entropy as $D_{KL}(p\|m)$ rather than $\check{H}(p, m)$.

Symmetric Kullback-Leibler Divergence

The Kullback-Leibler divergence can be made symmetric using the following mathematical trick, which is how Kullback and Leibler originally defined it:

$$D_{KL}^{symmetric}(P\|Q) = D_{KL}(P\|Q) + D_{KL}(Q\|P). \quad (4.36)$$

Another generalized alternative is the λ -divergence, which incorporates the probabilities λ and $(1 - \lambda)$ about P and Q respectively:

$$D_{\lambda}(P\|Q) = \lambda D_{KL}(P\|\lambda P + (1 - \lambda)Q) + (1 - \lambda) D_{KL}(Q\|\lambda P + (1 - \lambda)Q) \quad (4.37)$$

Notice that for $\lambda = 0.5$, we have the so-called Jensen-Shannon divergence, so that

$$D_{JS} = \frac{1}{2} D_{KL} \left(P \left\| \frac{P+Q}{2} \right\| \right) + D_{KL} \left(Q \left\| \frac{P+Q}{2} \right\| \right). \quad (4.38)$$

Chapter 5

Sample Statistics

In this chapter we make a short review of weighted covariance matrices, along with an equally-short review of sample variance and sample covariance. In this work, we generalize the standard definition of weighted covariance in Part III. Before that, we state the classical definitions for variance in Section 5.1, covariance in Section 5.2, covariance matrix in Section 5.3, and finally for the weighted covariance matrix in Section 5.4.

5.1 Variance

The variance of a random variable is a measure of its statistical dispersion around the expected value. Let the mean of a random variable X be defined as $\mu = E(X)$. We have that the variance is the expectation of the square of the difference between the values and the mean, so that

$$\text{var}(X) = E[(X - \mu)^2]. \quad (5.1)$$

The standard definition of sample mean is given by

$$\mu = E(X) = \frac{1}{n} \sum_{i=1}^n x_i. \quad (5.2)$$

The classical definition of sample variance follows:

$$\text{var}(X) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2. \quad (5.3)$$

Notice that, for a discrete random variable with probability mass function $p_1, \dots, p_n$, we would have a different mean given by

$$\mu = E(X) = \sum_{i=1}^n x_i p_i. \quad (5.4)$$

Analogously, we would have the weighted variance

$$\text{var}(X) = \sum_{i=1}^n (x_i - \mu)^2 p_i. \quad (5.5)$$

5.2 Covariance

The covariance is a generalization of the variance which measures how two random variables vary together. Consider two random variables X and Y , with respective expected values $E(X) = \mu$ and $E(Y) = \nu$.

The standard definition of covariance between X and Y is then

$$\text{cov}(X, Y) = E[(X - \mu)(Y - \nu)]. \quad (5.6)$$

Some interesting properties follow from this definition, namely $\text{cov}(X, X) = \text{var}(X)$, which shows how the covariance is a generalization of the variance, and the symmetry property $\text{cov}(X, Y) = \text{cov}(Y, X)$.

5.3 Covariance Matrix

The concept of covariance may be extended to higher dimensions, creating the so-called covariance matrix. The covariance matrix Σ may be defined as

$$\Sigma = E[(\mathbf{X} - E[\mathbf{X}])(\mathbf{Y} - E[\mathbf{Y}])^T]. \quad (5.7)$$

The covariance matrix Σ can be expanded and seen as

$$\begin{bmatrix} E[(X_1 - \mu_1)(X_1 - \mu_1)] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ E[(X_2 - \mu_2)(X_1 - \mu_1)] & E[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & E[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & E[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & E[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}. \quad (5.8)$$

The sample covariance matrix Σ above may be generalized to $\Sigma = [\sigma_{ij}]$, where each entry is given by

$$\sigma_{ij} = \frac{1}{n-1} \sum_{k=1}^n (x_{ik} - \mu_i)(x_{jk} - \mu_j) \quad (5.9)$$

The sample mean from Equation (5.2) and the sample covariance matrix from Equation (5.9) are unbiased estimates of the mean and the covariance matrix of the random variable $\mathbf{X}$.

5.4 Weighted Covariance Matrix

Finally, consider a discrete random variable with probability mass function $p_1, \dots, p_n$, where each vector $\mathbf{x}_k$ from $\mathbf{X}$ has an assigned probability p_i . Consider the weighted mean given in Equation (5.4), and assume $p_k \geq 0$ and $\sum_{k=1}^n p_k = 1$. We may now replace the expansion of the covariance matrix Σ given by Equation (5.9) with a weighted covariance term, such that

$$\sigma_{ij} = \frac{\sum_{k=1}^n p_k (x_{ik} - \mu_i)(x_{jk} - \mu_j)}{1 - \sum_{k=1}^n p_k^2}. \quad (5.10)$$

Notice that if all vectors are equiprobable, that is, $p_k = \frac{1}{n}$, the weighted mean and covariance are equivalent to the sample mean and covariance respectively.

Being some of the most fundamental tools from statistics, there certainly are many applications for sample variance, sample covariance, and the sample covariance matrix in practically all fields of science. The applications for weighted covariance matrices are somewhat more limited, though. There are some scattered applications reported in research papers in statistics, physics, bioinformatics, machine learning, among other fields. There are even fewer generally-acknowledged, widespread applications. One of these few, the use of weighted covariance matrices to assess the risk of a financial portfolio, inspired this work. In this application, the weights of the covariance matrix reflect the relative size of each security contained in a portfolio. The risk is then assessed based on the weighted covariance matrix, and may even be minimized in order to find the optimal portfolio composition [EGBG03]. Although this application from portfolio theory lies in a totally different field from machine learning, its principles are the same as those used in this work, where we also minimize a measure of risk dependent of the covariance matrix while we adjust its weights.

Part III

Development

Chapter 6

Interpreting Statistical Learning Theory

In this chapter, we examine once again some concepts in statistical learning theory and the definition of transductivity. Specifically for the transductive learning problem, we show how some well established concepts in inductive learning are still missing from the current theory. We start in Section 6.1 with a discussion on the roots of transductivity and give general definitions for different learning problems. In Section 6.2 we define essentially-transductive learning, an alternative, differentiating domination for semi-supervised transductive learning. In Section 6.3 we introduce variations on the current theory to better describe different learning problems. We end the chapter in Section 6.4, where we introduce the question of existence and indeterminacy of a separating margin in a transductive learning problem. This question is not fully answered, and its further study is suggested as future work in Section 14.3.

6.1 Transductivity and Learning Rules

The concepts of transductive and semi-supervised learning are not always properly disambiguated in the literature. Since their differences, or even their co-existence, are

somewhat subtle, there is often confusion between the two. Let us examine each one separately.

First, the distinction between unsupervised, supervised, and semi-supervised learning ultimately has to do with the availability of data *a priori*. In unsupervised learning, one tries to estimate labels for a data set with no prior knowledge of any labels in this set. In supervised learning, there is a training set with labels available to create (or induce) a learning rule, from which points of interest given later may be evaluated (or deduced). In semi-supervised learning, not only there is a labeled training set *a priori*, but there also exists a working set containing unlabeled samples.

Second, the distinction between inductive and transductive learning has to do with the creation of general learning rules. In inductive learning, one induces a learning rule from data available *a priori*. Later, one uses this rule to evaluate new samples only available *a posteriori*. If we consider that one's goal is to estimate unknown labels at given points of interest, creating a learning rule is the logical alternative to solve the problem when no information about such points is available *a priori*. In transductive learning, on the other hand, all points of interest must be available *a priori*, which automatically implies that a transductive solution does not need to generalize to other points except those initially provided.

Although not unanimously, these definitions are generally acknowledged by most of the learning theory community. Based on them, it is easy to realize that transduction and semi-supervised learning may not be directly compared, much like apples and oranges. Furthermore, one also realizes that there may be semi-supervised inductive learning as well as semi-supervised transductive learning. The difference between the two lies exactly on the requirement of the first to create a learning rule to be later evaluated over arbitrary unknown samples, where the second is only interested on the immediate esti-

mate of the unknown values at the given points.

The motivation for using semi-supervised learning methods, inductive or transductive, is the ability to use all *a priori* information available to increase the accuracy of solutions. The information contained in the disposition of the given unlabeled samples may give a better estimate of the distribution underlying the data of the problem, which ultimately would lead to improved accuracy and better solutions.

Notice that in transductive semi-supervised learning, one must explicitly estimate the unknown values at the given points. In inductive semi-supervised learning, on the other hand, the outcome of the problem is a learning rule capable of evaluating any sample in the range of the problem. This means that inductive semi-supervised learning does not need to necessarily evaluate the values at these given points. Instead, after the learning rule has been induced with the aid of the additional information provided by the given unlabeled samples, this rule is used to evaluate and label these same samples. This two-step strategy is adopted by the majority of inductive semi-supervised methods.

6.2 An Essentially-Transductive Approach to the Learning Problem

We often find references in the literature that, according to the definitions given in Section 6.1, mistake semi-supervised inductive learning by transductive learning, generally because they make use of unlabeled data to improve the accuracy of their methods. Some so-called transductive learning machines found in the literature actually rely on underlying inductive machines, often posing a meta-optimization problem where each step towards the solution is taken after inducing a full-scale general learning rule. For instance, several transductive support vector machines are built around traditional in-

ductive SVM heuristics, such as the maximum margin to the separating hyperplane, as reviewed in Section 3.4. The necessity of this meta-optimization approach is consistent with the striking scene given by the *a priori* availability of the points of interest, which reveals the combinatorial nature of the transductive problem.

In this work, as postulated in this chapter, we acknowledge that there is a clear distinction between semi-supervised inductive learning and semi-supervised transductive learning, which is exactly the requirement of inducing a general learning rule instead of only evaluating unknown values at points of interest. This is the same definition given by Vapnik [Vap98], who suggests, as shown in Section 2.4, that solving a transductive learning problem is easier than solving an inductive problem, as one is moving from particular to particular instead of inducing a general learning rule defined over the whole range of the problem. Following Occam’s Razor principle [Tho18], Vapnik [Vap95] also states that:

If you are limited to a restricted amount of information, do not solve the particular problem you need by solving a more general problem.

Finally, Vapnik [Vap98] explicitly argues against solving the problem of labeling a given set of points of interest via the two-step strategy of first inducing a learning rule and then using it to estimate the values of these points. He justifies that the solution to the relatively simple problem of estimating a finite set of numbers (i.e. the unknown values) becomes dependent of the solution of a substantially more complex problem of estimating a function, which implicitly means estimating all the values in the continuum of points containing the original points of interest.

Notice that this added complexity can be manifested in two fronts. First, when one uses little or insufficient data to infer higher-order information, such as using sample data to induce a learning rule, and the final model fails to properly describe the intended

context. Second, on the imposed complexity of the search spaces and of the methods themselves, although heuristic procedures and approximations of these spaces are often, if not always, necessary to make the problem computationally feasible.

In order to clearly distinguish our terminology for transductive learning and semi-supervised inductive learning, we shall introduce the term **essentially-transductive** to refer to what we defined as semi-supervised transductive learning, that is, learning with no induction of learning rules and with the direct estimation of unknown values at points of interest.

6.3 Theory Variations

We have qualitatively outlined several variations of the learning problem in Section 6.2. In this section, we shall revisit each of these settings and develop new formulations based on a slight variation of the classical statistical learning theory.

Before that, let us briefly comment on some concepts from statistical learning theory, introduced by Vapnik since the 1970's [Vap95]. As already described in Chapter 2, the classical theory, originally created to describe the inductive setting of the learning problem, has an extension that also covers the transductive learning problem. The so-called principle of structural minimization of the overall risk is based on the risk functional given by Equation (2.22), now repeated:

$$R_{\Sigma}(\alpha) = \frac{1}{k} \sum_{i=1}^k L(y_{\ell+i}, f(\mathbf{x}_{\ell+i}, \alpha)). \quad (6.1)$$

This risk functional is an extension of the risk functional from the Empirical Risk Minimization (ERM) principle given below, where the individual risk of each sample assessed by a loss function L is taken over the k samples in the working set, as opposed

to the ℓ samples in the training set.

$$R_{emp}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, f(\mathbf{x}_i, \alpha)), \quad \alpha \in \Lambda. \quad (6.2)$$

In the inductive setting of the learning problem, one is interested in finding the value for $\alpha^* \in \Lambda$ that minimizes the functional risk at $R_{emp}(\alpha^*)$. Notice that Λ is the generic set of parameters for $f(\mathbf{x}_i, \alpha)$, which makes $f(\mathbf{x}_i, \alpha), \alpha \in \Lambda$ the set of all admissible functions from which a learning rule will be selected. The selection of a function from the set $f(\mathbf{x}_i, \alpha), \alpha \in \Lambda$ is based on the training data available *a priori*, and corresponds to the induction of a learning rule. This rule is then used *a posteriori* to deduce the value of arbitrary points. This way, we may rewrite Equation (6.2) for the inductive learning problem, where the solution is exactly the parameter α^* that minimizes the risk functional R_{emp} , as shown below:

$$\alpha^* = \arg \min_{\alpha} \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, f(\mathbf{x}_i, \alpha)), \quad \alpha \in \Lambda. \quad (6.3)$$

Notice that in order to assess the empirical risk of a given parameter $\alpha \in \Lambda$, the loss function L in Equation (6.2) relies on both the current estimate of $f(\mathbf{x}_i, \alpha), \alpha \in \Lambda$ as well as the label y_i associated with each training sample $\mathbf{x}_i$. While the set of labels $(y_{\ell+1}, \dots, y_{\ell+k})$ is necessarily available for L when solving a supervised learning problem, this is not the case for semi-supervised transductive or unsupervised learning problems. On the contrary, the set $(y_{\ell+1}, \dots, y_{\ell+k})$ is actually the answer to the problem. If we recall Equation (6.1) for the overall risk, we realize that it does require this information, creates a gap between the theory and its applicability as the basis of new learning machines.

Over the next sections, we use the same notation as Equation (6.3) to characterize all combinations of the learning problem previously identified. In the next chapters, we shall see how this variation of the classical theory does serve as the foundation for a learning machine, where the value risk functional is directly estimated and minimized to

solve the essentially-transductive learning problem. Notice that the notation used in this section can be extended to the continuous case, where the summations are replaced by integrals. Although this is not shown for each of the variations described, it is done in Chapter 7 for the essentially-transductive problem.

6.3.1 Unsupervised Inductive Learning

We start with the unsupervised inductive learning problem. In this type of problem, one does not have any information about the labels of the samples. Instead, only the set $\theta = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$ is available. The result of the problem is a general learning rule determined by α , which can be used later to estimate $f(\mathbf{x}', \alpha)$ at any arbitrary point $\mathbf{x}'$. This problem can be characterized by the following expression.

$$\arg \min_{\alpha} \quad \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(f(\mathbf{x}_i, \alpha), \theta), \quad \alpha \in \Lambda \quad (6.4)$$

$$\theta = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\},$$

In this case, the loss function Φ must consider only the dispersion of the samples in θ to best estimate a general learning rule. Also, the set of functions $f(\mathbf{x}, \alpha), \alpha \in \Lambda$ must be defined for the whole range of the problem.

6.3.2 Unsupervised Transductive Learning

The unsupervised transductive learning problem is similar to its inductive variation in the sense that all information available is the set $\theta = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$. But instead of inducing a general learning rule, this setting of the problem is only interested in estimating the

values $y_1, \dots, y_\ell$ for the samples in θ :

$$\arg \min_{\{y_1, \dots, y_\ell\}} \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(f(\mathbf{x}_i), \theta), \quad (6.5)$$

$$\theta = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}.$$

This setting of unsupervised learning problem is, in theory, less complex than the inductive setting, as it does not require inferring general information using only particular data.

6.3.3 Supervised Inductive Learning

This is the classical setting of the learning problem for supervised learning. In this type of problem, one is interested in estimating a general learning rule determined by α , which can be used later to estimate $f(\mathbf{x}', \alpha)$ at any arbitrary point $\mathbf{x}'$. The data available, given by the set $\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$, contain not only information about the dispersion of points, but also about their associated labels:

$$\arg \min_{\alpha} \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(f(\mathbf{x}_i, \alpha), \theta), \quad \alpha \in \Lambda \quad (6.6)$$

$$\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}.$$

Equation (6.6) is equivalent to Equation (6.3), only put in different terms. In particular, the direct comparison done by the loss function L between the estimate of the function $f(\mathbf{x}_i, \alpha)$ and its corresponding label y_i for each training sample is replaced by the implicit risk assessment of loss function Φ , capable of considering θ to both compare y_i to $f(\mathbf{x}_i, \alpha)$ as well as incorporate the remaining information in θ to help guide the search for a learning rule α^* that minimizes the risk functional.

6.3.4 Supervised Transductive Learning

The supervised transductive variation of the learning problem is somewhat contradictory. As the problem is transductive, one is interested in only estimating the values of an unknown function at given points of interest. On the other hand, as the problem is also supervised, one may only use labeled samples. The contradiction lies in the fact that the information contained in the points of interest may not be used to find the solution, nor is one interested in estimating a general rule to deduce the values of arbitrary points *a posteriori*.

The resulting problem is equivalent to the two-step strategy of solving the transductive problem that was discussed in Section 6.2, where one does induce a general learning rule, only to used to deduce the values at points of interest. The resulting formulation is the following, where we explicitly change the output of the problem to reflect the necessity to induce a learning rule when no unlabeled samples may be considered (i.e. supervised learning):

$$\arg \min_{\{f(\mathbf{x}_{\ell+1}, \alpha), \dots, f(\mathbf{x}_{\ell+k}, \alpha)\}} \frac{1}{\ell} \sum_{i=1}^{\ell} \Phi(f(\mathbf{x}_i, \alpha), \theta), \quad \alpha \in \Lambda \quad (6.7)$$

$$\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{\ell}, y_{\ell})\}.$$

This variation of the problem is of little practical value, but it is useful to illustrate a possible setting of the learning problem where the range of $f(\mathbf{x}_i, \alpha), \alpha \in \Lambda$ does not need to be defined anywhere but for the given points of interest.

6.3.5 Semi-supervised Inductive Learning

The semi-supervised inductive learning problem, as discussed in Section 6.1, is often mistaken to be a transductive problem. As it is interested in inducing a general learning rule α , it is clearly inductive. Even so, it uses both the labeled data given by set $\theta =$

$\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$ and the unlabeled data given by set $\tau = \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}$ as input, making it semi-supervised:

$$\begin{aligned} \arg \min_{\alpha} \quad & \frac{1}{k} \sum_{i=\ell+1}^{\ell+k} \Phi(f(\mathbf{x}_i, \alpha), \theta, \tau), \quad \alpha \in \Lambda \\ \theta = & \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}, \\ \tau = & \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}. \end{aligned} \tag{6.8}$$

As in the supervised inductive setting, the risk assessment of each sample is given by loss function Φ , which is capable of considering θ and τ to compare y_i to $f(\mathbf{x}_i, \alpha)$ for each sample in θ , as well as incorporate the information contained in τ in the search for a α^* that minimizes the risk functional.

6.3.6 Semi-supervised (or Essentially-) Transductive Learning

Finally, we examine what we called an essentially-transductive learning problem, that is, a semi-supervised transductive variation. First of all, as in the semi-supervised inductive variation, all information available is employed, namely the training set $\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$, and the working set $\tau = \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}$. Second, the problem only aims at estimating the values $y_{\ell+1}, \dots, y_{\ell+k}$ at given points of interest. Following Occam's Razor principle, since the data conveying the points of interest are available *a priori*, we do not attempt to infer higher-order information in the format of a learning rule using particular sample data:

$$\begin{aligned} \arg \min_{\{y_{\ell+1}, \dots, y_{\ell+k}\}} \quad & \frac{1}{k} \sum_{i=\ell+1}^{\ell+k} \Phi(f(\mathbf{x}_i), \theta, \tau), \\ \theta = & \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}, \\ \tau = & \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}. \end{aligned} \tag{6.9}$$

Notice how $f(\mathbf{x}_i)$, the estimate of the value of the unknown function at point $\mathbf{x}_i$, is independent of any explicit parameters, where the loss function Φ is responsible for directly searching the values of $f(\mathbf{x}_i)$. During this search, Φ must also take into consideration all the information in θ and τ .

6.4 Existence of the Separating Margin

When one considers the essentially-transductive learning problem, one may conjecture that the underlying probability distributions and the placement of working set data may implicitly define a separating margin between classes, even if the method used to solve the problem fails to explicitly infer it.

Consider the binary classification problem, where there are two classes, A and B , whose corresponding underlying probability distributions P_A and P_B have been used to draw all working set samples *i.i.d.*. If p_A and p_B are the probability density functions for P_A and P_B respectively, one could immediately infer that there are pseudo-optimal (i.e. with minimum risk) classifications of samples $\mathbf{x} \in X$, where X is the range of the problem, given by:

$$class(\mathbf{x}_i) = \begin{cases} A & \text{if } p_A(\mathbf{x}_i) > p_B(\mathbf{x}_i), \\ B & \text{if } p_A(\mathbf{x}_i) < p_B(\mathbf{x}_i), \\ \text{undetermined} & \text{if } p_A(\mathbf{x}_i) = p_B(\mathbf{x}_i), \end{cases} \quad (6.10)$$

where the undetermined values obtained at the equality may be biased to either A or B .

If we fragment X into slices of length δ (or hypercubes of side δ in the n -dimensional case where $X \in \mathbb{R}^n$) and make $\delta \rightarrow 0$, we obtain a minimum-risk separating margin on the continuum of points at the equality of Equation (6.10). Therefore, under the

assumption that P_A and P_B are known (not approximated), we may infer the underlying separating margin of the problem.

The essence of the learning problem, though, is that we must (even if implicitly) estimate P_A and P_B from sample data. The question then becomes whether or not one is able to determine a separating margin exclusively from sample data. The setting of the learning problem plays an important role to answer this question, as it determines which samples will be available to estimate the margin.

Let us first examine the supervised inductive variation. In this case, there is a certain amount of training data from which a general learning rule α is induced. If we consider that we may evaluate $f(\mathbf{x}, \alpha)$ over the whole range of X , the result will yield a separating margin (even if not continuous) which is the *estimate* of the minimum-risk margin given the method used.

Even though it is straight forward to see the resulting separating margin obtained from an induced learning rule, transductive variations of the learning problem suffer from the indeterminacy of the margin. In the inductive setting, a finite number of training samples were promptly available to induce a fixed learning rule, which was then infinitely evaluated to yield the separating margin. In the transductive setting, though, every new sample that is added to the working set possibly influences the classification of other samples, which directly affects the format of the margin.

Therefore, if we extend the analysis of the binary classification problem just proposed to the transductive setting, we see that regardless of making $\delta \rightarrow 0$, the resulting margin will be different than that of the original problem with a finite working set. In fact, every new sample added to the working set is likely to change the margin, unless there already is an identical sample in the set and the method employed ignores duplicates. Although it is possible to estimate a skewed version of the margin, one cannot estimate the margin

itself without affecting and changing it.

The problem of indeterminacy of the separating margin in the transductive setting of the learning problem is similar to other indeterminacy problems from natural sciences. In quantum mechanics, Heisenberg's Principle of Uncertainty, shown below, states that one can only determine the position and the momentum of a particle with limited precision:

$$\Delta x \Delta p \geq \frac{\hbar}{2}, \quad (6.11)$$

where $\hbar$ is the reduced Planck constant $\hbar = h/2\pi$, Δx is the standard deviation of the position of the particle, and Δp is the standard deviation of its momentum. If one wishes to precisely determine the position of an object, for instance, the lower bound on the standard deviation of its momentum will increase, and vice-versa. Heisenberg's principle has been generalized by Schrödinger, Bohr, Dirac, among others, to other natural phenomena, such as between the angular position and angular momentum of objects, between the number of electrons in a superconductor and the phase of its Ginzburg-Landau order parameter, between the energy and time of quantum systems, and many other relations.

When trying to determine the separating margin, a similar phenomenon occurs. As one attempts to determine the value of the unknown function at a new point in X to later estimate the margin, one can only add this point to the working set, since the problem is transductive.

It follows that the presence of the value $\mathbf{x}_{\ell+k+1}$ in the working set necessarily changes the working sets's cardinality, and very likely also changes its mean, variance, entropy, and other measures over the set of samples. Should any property of the working set change, including its cardinality, so must its estimates, from the set $\{y_{\ell+1}, \dots, y_{\ell+k}\}$ to $\{y'_{\ell+1}, \dots, y'_{\ell+k}\}$. The deviation between $y_{\ell+k+1}$ and $y'_{\ell+k+1}$ is a measure of the skewness of the assessed margin. Each measure of skewness depends on the learning method

being used. Although this skewness may not be directly computed as $y_{\ell+k+1}$ is unknown, it may be generically given by:

$$\int_X L(y'_{\ell+k+i}, y_{\ell+k+i}) dx, \quad (6.12)$$

where L is a loss function, for instance the squared loss function $L(y'_{\ell+i}, y_{\ell+i}) = (y'_{\ell+i} - y_{\ell+i})^2$.

As a result, the assessed margin is deviated from the original one, much like trying to determine the momentum of an electron at a fixed position. Whether there is a bound governing the relationship between the standard deviation of the granularity of the range and the standard deviation of, say, the norm of the margin, is yet to be determined.

Chapter 7

Building an Essentially-Transductive Method

In this chapter we show definitions and assumptions made in the construction of an essentially-transductive learning machine. We start in Section 7.1, where we show how to directly estimate the functional risk for the transductive learning problem by employing the minimum discrimination information principle. Next, in Section 7.2, we show how this direct estimation is easier when done using the continuous definition of the functional risk, along with its implications. In Section 7.3 we review how the problem of density estimation is ill-posed. In Section 7.4 we justify our assumption of underlying Gaussian probability distributions based on the maximum-likelihood and maximum entropy methods. In Section 7.5 we show how our independence assumption greatly assists the solution of the problem. Finally, in Section 7.6 we modify the weighted covariance matrix reviewed in Section 5.4 to further simplify the solution of the problem.

7.1 Information-Theoretic Estimation of the Transductive Risk

We start our definition of an essentially-transductive method with the formulation of the essentially-transductive learning problem from Section 6.3.6, which can be stated as the following minimization problem:

$$\begin{aligned} \arg \min_{\{y_{\ell+1}, \dots, y_{\ell+k}\}} \quad & \frac{1}{k} \sum_{i=\ell+1}^{\ell+k} \Phi(f(\mathbf{x}_i), \theta, \tau), \\ \theta = & \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}, \\ \tau = & \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}, \end{aligned} \tag{7.1}$$

where there is a training set $\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$, a working set $\tau = \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}$, X is the range of the problem, and its goal is to estimate the values $y_{\ell+1}, \dots, y_{\ell+k}$ associated with each point in the working set.

The same expression can be generalized to the continuous case, where the overall risk is assessed over the whole range of the problem:

$$\begin{aligned} \arg \min_{\{y_{\ell+1}, \dots, y_{\ell+k}\}} \quad & \int_X \Phi(f(\mathbf{x}), \theta, \tau) d\mathbf{x}, \\ \theta = & \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}, \\ \tau = & \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}. \end{aligned} \tag{7.2}$$

In this continuous version, the transductive risk is not assessed at each of the given points of interest, but over the whole range of the problem. Notice that the problem still retains its transductivity properties, as the loss function Φ interprets and takes into consideration θ and τ , which remain discrete. Therefore, the estimates for $f(\mathbf{x})$, defined only for the points in τ (where $\tau \in X$), are given by Φ considering both θ and τ , where the solution is achieved based on the general minimum risk over the whole range of the problem.

Next, we recall the Minimum Discrimination Information (MDI) method, described in Section 4.3.8. The MDI principle states that the Kullback-Leibler divergence between two probability distributions may be used as discrimination information given new facts. Consider a binary classification problem with classes A and B , whose underlying probability distributions are P_A and P_B , respectively. Suppose P_A and P_B are known. For every new sample $\mathbf{x}_i$ to be classified, one may attempt to incorporate $\mathbf{x}_i$ in each class by having $A' = \{A \cup \mathbf{x}_i\}$ and $B' = \{B \cup \mathbf{x}_i\}$. Later, one may calculate $D_{KL}(A\|A')$ and $D_{KL}(B\|B')$, having the class where the sample caused the lesser disturbance, that is, the one with the smallest divergence to its extended version, permanently incorporating the new sample. Put formally, one may use the Kullback-Leibler divergence to classify new samples in the following way:

$$class(\mathbf{x}_i) = \begin{cases} A & \text{if } D_{KL}(A\|\{A \cup \mathbf{x}_i\}) < D_{KL}(B\|\{B \cup \mathbf{x}_i\}), \\ B & \text{if } D_{KL}(A\|\{A \cup \mathbf{x}_i\}) > D_{KL}(B\|\{B \cup \mathbf{x}_i\}), \\ \text{undetermined} & \text{if } D_{KL}(A\|\{A \cup \mathbf{x}_i\}) = D_{KL}(B\|\{B \cup \mathbf{x}_i\}), \end{cases} \quad (7.3)$$

where the undetermined values obtained at the equality may be biased to either A or B . Notice that Equation (7.3) is fairly similar to Equation (6.10), where the Kullback-Leibler divergence, instead of the direct comparison between the two probability distributions, conveys the discriminatory information.

Finally, notice also that the Kullback-Leibler divergence has a discrete definition as well as continuous one, both described in Section 4.3.8 and expressed by Equations (4.27) and (4.28), respectively. Since we also have a discrete and a continuous formulation of the essentially-transductive problem given by Equations (7.1) and (7.2), either one may be used to build a learning method based on the MDI principle, where we must select the corresponding Kullback-Leibler divergence definition.

7.2 Moving into the Continuous

In order to build an essentially-transductive method, we will try to directly estimate the transductive risk from one of the settings proposed in the last section. Since solving the problem can be reduced to solving a risk minimization problem, we shall try to find an expression for the loss function Φ that best reflects the problem and that, once minimized, should result in a good solution.

In Section 4.3.8 and again in Section 7.1, we have seen how one may employ the MDI principle to select a class for a given unknown sample. Let us extend this concept to the whole set of samples contained in the working set.

First, one may iteratively traverse through all k samples in the working set and assign classes based on the simple criterion from Equation (7.3). Notice that by doing so, one is not using all information contained in the working set to solve the learning problem, as new samples are just incorporated gradually, and the first samples are classified based only on the information contained in the training set.

Second, one may try reclassification after all samples in the working set have been assigned to a class, thus considering all information available at this point. This reclassification scheme, though, reveals the combinatorial nature of the transductive learning problem, as discussed in Section 3.6. Every time a new sample is reclassified, other samples may be influenced by its new value, which may lead to a chain effect that does not necessarily converge to a unique solution.

In order to solve the essentially-transductive learning problem by directly minimizing the transductive risk, one must clearly employ a more tractable heuristic to cope with the complexity of the search space. Ideally, we would like to use the MDI principle over the whole set of samples in the working set, without necessarily assessing all discrete combinations of hypotheses possible. Instead of making discrete assignments for each

sample to either class, A and B , we could move the problem into a continuous search space that, say, accepts real weighted assignments in the interval $[0, 1]$. If we have k samples to classify, the problem is then reduced to finding the point with minimum risk in the space $[0, 1]^k$. The final solution may then be taken similarly to Equation (7.3), by:

$$class(\mathbf{x}_i) = \begin{cases} A & \text{if } \phi(\mathbf{x}_i) > 0.5, \\ B & \text{if } \phi(\mathbf{x}_i) < 0.5, \\ \text{undetermined} & \text{if } \phi(\mathbf{x}_i) = 0.5, \end{cases} \quad (7.4)$$

where $\phi(\mathbf{x}_i)$ is the weight associated with sample i , and the undetermined values obtained at the equality may be biased to either A or B .

Next, we must integrate the MDI principle with the weighted assignment search space. For that, we shall generalize the concept of weighted probability distribution. In the discrete expression, we evaluated the underlying probability distribution for the set A , as well as its extension $A' = \{A \cup \mathbf{x}_i\}$, to measure $D_{KL}(A||A')$. Now, we shall evaluate the underlying probability distribution over an enlarged set S , containing both training set and working set such that $S = \theta \cup \tau$. The weights for the elements from θ , the training set, may take values 0 or 1, according to the previously available labels associated with each sample. The weights for the elements from τ will be sought by the method, and in the end will result in the final answer to the problem.

Finally, we introduce $\bar{S}$, the complement of S . $\bar{S}$ has the same elements as S , but every weight is equal to its complement from S , that is, $(1 - \phi(\mathbf{x}_i))$. Notice how the samples in the training set are uniformly described between the sets, taking values 0 or 1 at either S or $\bar{S}$. We may, then, evaluate the divergence between S and its complement $\bar{S}$, where the larger the divergence, according to the MDI principle, the better the final solution. That is, we may maximize the value of $D_{KL}(S||\bar{S})$ in order to find the solution to the essentially-transductive problem.

In order to select between the two Kullback-Leibler formulations from Equations (4.27) and (4.28) to define $D_{KL}(S\|\bar{S})$, we refer to Section 4.3.3, where we showed how the differential entropy of distributions may be calculated in closed form if we know their probability density functions. Similarly, in Section 4.3.8, we showed the same property for the differential Kullback-Leibler divergence, which has a closed form if we assume the shapes of the distributions and their parameters. Since we have already established that solving the problem is equivalent to maximizing the Kullback-Leibler divergence given by $D_{KL}(S\|\bar{S})$, we choose the continuous definition, and take advantage of its closed form.

At this point, we may extend the continuous version of the essentially-transductive problem formulation, where the original integral in Equation (7.2) is replaced by the integral from Equation (4.28), yielding:

$$\arg \min_{\{y_{\ell+1}, \dots, y_{\ell+k}\}} \int_X p_S(\mathbf{x}) \log \frac{p_S(\mathbf{x})}{p_{\bar{S}}(\mathbf{x})} d\mathbf{x}, \quad (7.5)$$

where p_S and $p_{\bar{S}}$ are the probability density functions of the general weighted probability distribution S and its complement $\bar{S}$, respectively.

There is still an open issue, though, which is exactly how to define the general concept of a weighted probability distribution, and consequently its probability density function. This is shown next in Sections 7.3, 7.4, and 7.6.

7.3 The Problem of Density Estimation is Ill-Posed

In order to evaluate the explicit risk expression from Equation (7.5), one must estimate the probabilities P_S and $P_{\bar{S}}$, which may not at all be an easy task. Estimating probability densities from samples is a well studied ill-posed problem from Statistics. The problem consists of approximating a density function $\hat{p}(x)$ from data samples whose correspond-

ing distribution function is

$$F(x) = P(X \leq x) = \int_{-\infty}^x p(t)dt. \quad (7.6)$$

The density estimation problem is analogous to solving the linear operator equation to the approximation of $F(x)$

$$\int_{-\infty}^{\infty} \theta(x-t)\hat{p}(t)dt = \hat{F}(x), \quad (7.7)$$

where we are given data samples $x_1, \dots, x_\ell$, drawn independently and identically distributed with respect to the unknown distribution function $F(x)$ (approximated by $\hat{F}(x)$), and θ is the sign function:

$$\theta(x) = \begin{cases} -1 & x < 0, \\ 0 & x = 0, \\ 1 & x > 0. \end{cases} \quad (7.8)$$

Not only is the problem ill-posed, in our application to machine learning it becomes particularly expensive. Since we often deal with samples with dozens, sometimes hundreds or even thousands of dimensions, estimating joint distributions may be impractical [SS04].

7.4 Gaussian Assumption

Recall from Section 4.3.8 that the continuous Kullback-Leibler divergence expression may have a closed form if the shape of the probability distributions are known *a priori*. Since estimating the probability density functions of P_S and $P_{\bar{S}}$ is an ill-posed problem, we shall make an assumption about the shape of the distributions.

Therefore, we assume that both P_S and $P_{\bar{S}}$ are multivariate Gaussian distributions.

By doing so, we have the immediate benefit of relying on the closed form of the Kullback-Leibler divergence, in the case of multivariate Gaussians, given by Equation (4.35).

The Gaussian assumption may be well justified by two statistical principles. First, the *maximum entropy* method. Consider a given random variable with unknown probability distribution function, where the mean and variance of the random variable are known. If one is trying to estimate the probability distribution function, there is not enough information in the mean and variance to uniquely estimate the function. By the maximum entropy method, one selects the probability distribution that yields the maximum entropy, which is a Gaussian distribution. Details of the method, as well as the proof of the Gaussian distribution being that with the largest entropy, are given in [LG94].

Second, the *maximum-likelihood method*, which, for an assumed fixed probability density function, finds the set of parameters that maximizes the likelihood of generating the given sample data. It follows that for a multivariate Gaussian density function, the maximum-likelihood estimates for the mean and covariance matrices are the sample mean and sample covariance matrices themselves. Therefore, since we do not know any other information about the given probability functions, the Gaussian assumption has the benefit of using the sample mean and sample covariances as the direct parameters of the distributions. More details about the maximum-likelihood method may be found in [HAH01].

In addition to these two principles, the Gaussian assumption also benefits from the definition of the weighted probability distribution. Since the distributions are now assumed to be Gaussians, then we must weight the input parameters of each of the Gaussian distribution, namely their means and variances.

In Chapter 5, we have seen how the weighted mean of set of samples may be given

by the straightforward expression:

$$\mu = \mathbb{E}(X) = \sum_{i=1}^n x_i p_i. \quad (7.9)$$

Additionally, we have seen an expression for the weighted covariance matrix, borrowed from the theory of financial portfolios, in Equation (5.10). The original expression, however, assumes that weights are a probability distribution, hence $\sum_{i=1}^k p_i = 1$. In our model, though, we have defined each weight in the interval $[0, 1]$. Therefore, we need a generalization of the original weighted covariance matrix which does not make this weight assumption, which is derived ahead in Section 7.6.

By providing weighted means and covariance matrices to the Gaussian distributions from Equation (7.5), we have solved the issue of how to define the weighted probability distributions for P_S and $P_{\bar{S}}$.

7.5 Independence Assumption

Before we finish the definition of our essentially-transductive method, one last assumption becomes convenient. Since we have settled for the continuous version of the Kullback-Leibler divergence assuming Gaussian distributions, let us examine its closed form given by Equation (4.35) and repeated below:

$$D_{KL}(P\|Q) = \frac{1}{2} \left(\log \left(\frac{|\Sigma_Q|}{|\Sigma_P|} \right) + \text{tr}(\Sigma_Q^{-1} \Sigma_P) + (\mu_Q - \mu_P)^T \Sigma_Q^{-1} (\mu_Q - \mu_P) - N \right). \quad (7.10)$$

Let us assume that μ_Q and μ_P are the sample weighted means, and that Σ_Q and Σ_P are sample weighted covariance matrices. In order to evaluate $D_{KL}(P\|Q)$, we see that some non-trivial operations are necessary, namely the inversion of one of the covariance matrices and the evaluation of the determinant of both covariance matrices. These operations are not only computationally expensive, but may add numerical error to the final

method when it is implemented on a computer. Moreover, the expression in Equation (7.10) is not easily differentiable as it stands.

In order to improve on these points, therefore, we assume the independence of data along the n dimensions given by the problem. Formally, the independence assumption means that the covariance matrices Σ_Q and Σ_P are diagonal matrices, that is, their only non-zero terms lie along their main diagonal.

This independence assumption has an immediate effect over Equation (7.10). First, the inversion of a diagonal matrix becomes far simpler, where the result is another diagonal matrix whose the terms along the main diagonal are equal to the individual inversion of the original terms, such that:

$$A^{-1} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix} \quad (7.11)$$

Second, the trace of a diagonal matrix A is simply the sum of terms along its main diagonal:

$$tr(A) = tr(diag(A)) = \sum_{i=1}^n a_{ii}. \quad (7.12)$$

Third, the determinant of a diagonal matrix is the product of terms along its main diagonal:

$$|A| = |diag(A)| = \prod_{i=1}^n a_{ii}. \quad (7.13)$$

Finally, we substitute all these identities in Equation (7.10) under the assumption

that both Σ_Q and Σ_P are diagonal, and obtain the following expression:

$$D_{KL}(P\|Q) = \frac{1}{2} \left(\log \frac{\prod_{i=1}^n q_{ii}}{\prod_{i=1}^n p_{ii}} + \frac{\sum_{i=1}^n p_{ii}}{\sum_{i=1}^n q_{ii}} + \sum_{i=1}^n \frac{(\mu_Q - \mu_P)^2}{q_{ii}} - n \right). \quad (7.14)$$

As a last refinement, we eliminate the possible numerical problems resulting from the log of products in the first term using the basic logarithmic identity $\log \prod_{i=1}^n a_i = \sum_{i=1}^n \log a_i$:

$$D_{KL}(P\|Q) = \frac{1}{2} \left(\sum_{i=1}^n \log \frac{q_{ii}}{p_{ii}} + \frac{\sum_{i=1}^n p_{ii}}{\sum_{i=1}^n q_{ii}} + \sum_{i=1}^n \frac{(\mu_Q - \mu_P)^2}{q_{ii}} - n \right). \quad (7.15)$$

Notice how this expression is not only simpler to evaluate, but it is also much simpler to differentiate, as shall be done ahead.

7.6 Modified Weighted Covariance

The definition of the modified weighted covariance matrix starts from the definition of weighted variances given in Equation (5.5). We may extend this definition and drop the requirement that the sum of weights equals 1:

$$\text{var}(X) = \frac{\sum_{i=1}^n (x_i - \mu)^2 p_i}{\sum_{i=1}^n p_i}. \quad (7.16)$$

Notice that, given the independence assumption, we are only interest in estimating the main diagonal of the covariance matrix, that is, the weighted variances along each dimension. From Chapter 5 we also have the definition from Equation (5.9) which defines each element of the sample covariance matrix. Together with Equation (7.16) we may then define the general format of each element along the main diagonal of the generalized covariance matrix, where every other element is assumed to be 0:

$$\begin{aligned} \sigma_{ii} &= \frac{\sum_{d=1}^n (x_{id} - \mu_i)^2 p_i}{\sum_{d=1}^n p_d}, \\ \sigma_{ij} &= 0 \quad \forall i \neq j. \end{aligned} \quad (7.17)$$

Next, we can add the definition of weighted mean to the expression, and, after final algebraic transformations, reach the final expression for the terms of the weighted covariance matrix:

$$\begin{aligned}
\sigma_{ii} &= \frac{\sum_{d=1}^n (x_{id} - \mu_i)^2 p_i}{\sum_{d=1}^n p_d}, \\
&= \frac{\sum_{d=1}^n (x_{id} - \frac{\sum_{e=1}^n p_e x_{ie}}{\sum_{e=1}^n p_e})^2 p_i}{\sum_{d=1}^n p_d}, \\
&= \frac{\sum_{d=1}^n p_d x_{id}^2}{\sum_{d=1}^n p_d} - \left(\frac{\sum_{d=1}^n p_d x_{id}}{\sum_{d=1}^n p_d} \right)^2, \\
\sigma_{ij} &= 0 \quad \forall i \neq j.
\end{aligned} \tag{7.18}$$

Chapter 8

Essentially-Transductive Entropic Classifier

This chapter shows the resulting essentially-transductive entropic classifier based on the theory from Chapter 6 and the definitions and assumptions from Chapter 7. We start in Section 8.1 by proposing the transductive learning problem as a constrained optimization problem. Then, we show how we solved the optimization problem using a well-known constrained optimization algorithm in Section 8.2.

8.1 Learning as a Constrained Optimization Problem

Using all definitions described throughout Chapter 7, we may at last define the so-called Essentially-Transductive Entropic Classifier (or ETEC). Consider a binary classification essentially-transductive learning problem, where we have a training set with ℓ samples $\theta = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)\}$, $\theta \in (X, Y)$, a working set with k samples $\tau = \{\mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+k}\}$, $\tau \in X$, X is the range of the problem, and Y is the set of possible classification labels. The goal of the problem is to estimate values $y_{\ell+1}, \dots, y_{\ell+k}$ at given points of interest.

Since this is a binary classification problem, assume $Y = \{-1, +1\}$, which implies that each estimated $y_{\ell+1}, \dots, y_{\ell+k}$ may take values in $\{-1, +1\}$.

We then pose the learning problem as a constrained optimization problem, where the values of the weights $\alpha_{\ell+1}, \dots, \alpha_{\ell+k}$ that result in the minimum value for $-D_{KL}(S\|\bar{S})$ yield the answer. The weights $\alpha_{\ell+1}, \dots, \alpha_{\ell+k}$ are allowed to vary in the closed interval $[0, 1]$, the only constraints necessary. The final answer to the problem, that is, the labels for each of the samples in the working set, are set to either possible label should the corresponding weights be greater than or lesser than 0.5, that is,

$$\begin{aligned} \arg \min_{\{\alpha_{\ell+1}, \dots, \alpha_{\ell+k}\}} & -D_{KL}(S\|\bar{S}), \\ y_i = & \begin{cases} +1 & \text{if } \alpha_i \geq 0.5 \\ -1 & \text{if } \alpha_i < 0.5 \end{cases}, \quad \forall (\ell+1) \leq i \leq (\ell+k) \\ & 0 \leq \alpha_i \leq 1, \end{aligned} \tag{8.1}$$

where we explicitly biased the answer of weights equal to 0.5 towards the +1 label.

Next, we define $D_{KL}(S\|\bar{S})$. As our method assumes multivariate Gaussian distributions for their data, the shortened notation $D_{KL}(S\|\bar{S})$ shall denote the divergence between the multivariate Gaussian probability distributions obtained from the weighted sample mean and weighted sample covariance matrix of the sets S and $\bar{S}$, where each set has its own weight vector α^S and $\alpha^{\bar{S}}$. The sample data in S and $\bar{S}$ are the same, namely $\theta \cup \tau$, that is, all data available in the problem. The weight vector α^S for S is defined as follows:

Weights $\alpha_1^S, \dots, \alpha_\ell^S$ corresponding to training set samples:

$$\alpha_i^S = \begin{cases} 0 & \text{if sample } x_i \text{ belongs to class } -1, \\ 1 & \text{if sample } x_i \text{ belongs to class } +1, \end{cases} \quad 1 \leq i \leq \ell$$

Weights $\alpha_{\ell+1}^S, \dots, \alpha_{\ell+k}^S$ corresponding to working set samples:

$\alpha_i^S \in_{(\ell+1) \leq i \leq (\ell+k)} [0, 1]$, where weights are estimated to solve the optimization problem.

The weight vector for the complement $\bar{S}$ is given by $\alpha^{\bar{S}} = 1 - \alpha^S$, which means that $\alpha^{\bar{S}}$ is contained in the same interval $[0, 1]$. Since the weight vector for the complement is given by this simple relation, we shall simplify the notation by considering a single weight vector α , equivalent to α^S , with elements $\alpha_1, \dots, \alpha_\ell, \alpha_{\ell+1}, \dots, \alpha_{\ell+k}$.

Finally, the expression for $D_{KL}(S \parallel \bar{S})$ is given by:

$$\begin{aligned}
D_{KL}(S \parallel \bar{S}) = & 0.5 \left(\sum_{i=1}^n \log \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2}{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2} \right. \\
& - \sum_{i=1}^n \log \frac{\left(\ell + k - \sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} x_{j,i}^2 - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} x_{j,i} - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2}{\left(\ell + k - \sum_{j=1}^{\ell+k} \alpha_j \right)^2} \\
& + \sum_{i=1}^n \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\frac{\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}}{\sum_{j=1}^{\ell+k} \alpha_j} - \frac{\sum_{j=1}^{\ell+k} x_{j,i} - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i}}{\ell + k - \sum_{j=1}^{\ell+k} \alpha_j} \right)^2}{\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2} \\
& + \sum_{i=1}^n \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\ell + k - \sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} x_{j,i}^2 - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right)}{\left(\ell + k - \sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2 \right)} \\
& \left. - \sum_{i=1}^n \frac{\left(\sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\sum_{j=1}^{\ell+k} x_{j,i} - \sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2}{\left(\ell + k - \sum_{j=1}^{\ell+k} \alpha_j \right)^2 \left(\left(\sum_{j=1}^{\ell+k} \alpha_j \right) \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i}^2 \right) - \left(\sum_{j=1}^{\ell+k} \alpha_j x_{j,i} \right)^2 \right)} - n \right) \quad (8.2)
\end{aligned}$$

where $x_{j,i}$ denotes the value of the j -th dimension of the i -th vector $\mathbf{x}$ in the data set.

Remarkably – and in contrast with other methods from the literature –, the method does not have any tunable parameters, which enhances its applicability and ease of use.

8.2 Solving the Optimization Problem

In order to solve the essentially-transductive learning problem posed as an optimization problem in Section 8.1, we may use several constrained optimization algorithms available in the literature. A key factor to the selection of this algorithm is the availability of derivatives of the objective function.

In our problem, as the objective function is $f(\alpha) = -D_{KL}(S\|\bar{S})$, we must find an expression $\frac{\partial f(\alpha)}{\partial \alpha_k}$ for the partial derivatives of the objective for any given k . Since the function is continuous and differentiable, this partial derivative exists and is available. Since the expression for $\frac{\partial f(\alpha)}{\partial \alpha_k}$ is somewhat large, it is omitted here and shown in Appendix C. The second-order derivatives of $f(\alpha)$ have not been used, as the expression for its Hessian matrix becomes far too large to be effectively tractable or even efficiently computable.

In this work, we used a Quasi-Newton sequential quadratic programming (SQP) algorithm to solve the optimization problem. The first-order derivatives were explicitly available via the expression in Equation (C.2), and second-order derivatives were approximated by the method.

The actual algorithm implementation used was provided by the Optimization Toolbox shipped with MATLAB version 7.4.0 (R2007a).

Chapter 9

Information Indicators

This chapter introduces dataset indicators based on information theory. These indicators are used to measure information quantities in all sets used in the experiments described in Part IV. We start with Section 9.1 where we give a general description of the motivation and assumptions behind the indicators proposed. In Section 9.2 we then describe first-order indicators, defined as measures taken over the standard sets available in a given learning problem. In Section 9.3, we describe second-order indicators, based on measures taken over the union of two or more of these sets. We end the chapter with Section 9.4, where we show a third-order indicator based on the ratio between two other indicators.

9.1 Overview

Since we have proposed a learning method based on concepts from information theory, it may be possible to use other information measures to infer properties from learning problems. If a given learning problem contains plenty of training information, but very little working data, it is likely that a transductive method will not perform better than

an inductive one to solve it. On the other hand, if we have limited training data, the information contained in the working set may enhance the accuracy of a transductive method by providing it with better estimates of the underlying probability distribution of the problem. Therefore, we are interested in assessing the quantities of information contained in each set of the transductive learning problem, expecting that these measures will indicate whether a transductive or essentially-transductive learning method will yield good results.

In this chapter we introduce 8 information indicators that we use later when we analyze the benchmark and experimental results from the ETEC method proposed. Notice that these indicators are solely based on general information measures. As suggested in Section 14.1, it is likely that other (not necessarily entropic) measures may represent better the applicability of transductive methods to given learning problems. Nevertheless, we introduce our entropic indicators in this work, and set the search for new non-entropic indicators as a possible future work.

The most straight-forward entropic indicators that we may produce for a given learning problem are the measures of the entropy of the sets of the problem. Considering the Gaussian and independence assumptions made in Sections 7.4 and 7.5 respectively, we may easily assess the differential entropy of each of these sets. Recall that the differential entropy defined by Equation (4.5) is given in closed form for multivariate Gaussian distributions by Equation (4.34). Notice that the Gaussian differential entropy only depends on the covariance matrix of the distribution, not even requiring its mean value. Given the independence assumption, this covariance matrix becomes diagonal, containing only the variances for each of the dimensions. As described in Section 4.2, the unit for these entropy measures is the *nat*, which, for simplicity, we shall omit when we publish our indicators.

In this chapter and whenever we refer to information indicators, we shall use the short notation $h(A)$ to denote the multivariate Gaussian differential entropy of the probability distribution underlying dataset A . For the remaining sections, consider a transductive learning problem where the training set is divided into two sets A and B , each one representing a class, and there is a working set T .

9.2 First-Order Indicators: Entropy of Sets

First-order indicators are given by the entropy of the sets readily available in a learning problem. The 3 first-order indicators are, therefore:

$h(A)$ – Entropy of the set of samples of class A in the training set;

$h(B)$ – Entropy of the set of samples of class B in the training set;

$h(T)$ – Entropy of the set of unknown samples in the working set.

9.3 Second-Order Indicators: Entropy of Compound Sets

Second-order indicators are defined by the entropy of the union of two or more sets. The 4 second-order indicators are:

$h(A + B)$ – Entropy of the complete training set;

$h(A + T)$ – Entropy of the set of samples of class A combined with the set of unknown samples;

$h(B + T)$ – Entropy of the set of samples of class B combined with the set of unknown samples;

$h(A + B + T)$ – Entropy of the complete training set combined with the working set.

These combinations are useful to evaluate if the information contained in two sets is similar or not. For instance, if the value for $h(A)$ is close to the value for $h(B)$, A and B may have the same dispersion, but they may not necessarily be overlapped. If $h(A)$ is close to $h(B)$ and also to $h(A + B)$, though, we can see that the amount of information obtained by adding one set to the other does not significantly change, which suggests that A and B overlap.

Notice that the first 3 indicators will vary whenever we change the proportions between training and working data. The last indicator, though, is fixed for any distribution between sets, as it measures the total information of the problem.

9.4 Third-Order Indicators: Transductivity Measure

At last, we propose a third-order indicator, which is not directly assessed from the data, but obtained from the combination of two other indicators. This indicator may be seen as a transductivity measure of a problem, as it is defined as

$$\frac{h(T)}{h(A + B)} - 1. \quad (9.1)$$

Notice that being a ratio between two entropy measures in nats, this indicator has no units, and it is therefore shown as a percentage.

When this indicator is positive, it essentially means that the problem contains more information in its working set than in its training set. This is an indication that this problem may be well suited to a transductive method. Analogously, when the indicator is negative, it is likely that an inductive method will perform just as well.

This indicator must not be taken as a direct measure of the applicability of transductive learning. First, it does not assess enough information measures to convey precise

information with respect to the training and working sets. For instance, if there is a problem with similarly dispersed training and working sets, this indicator will be close to zero even if the sets are far apart in the space of the problem. Second, there is a certain degree of approximation involved with estimating indicators based on differential entropy of sets, where some of these sets' properties may fail to be identified.

This transductivity measure, or *t-measure*, may then be used as a reference, as opposed to a direct measure, for the applicability of transductive learning methods. In addition, it should preferably be used accompanied by the other second-order and first-order indicators, which convey more information about the problem at hand.

Part IV

Results

Chapter 10

Data Sets

This chapter describes the data used in all experiments conducted in this work. Section 10.1 describes each one of the 7 datasets adopted. Section 10.2 shows how these datasets were sampled to obtain training and working sets for each experiment, including the two sampling variations used. Finally, Section A brings the application of the information indicators described in Chapter 9 to each of datasets sampled.

10.1 Description

All experiments in this work were based on publicly available datasets. We selected 7 datasets commonly used for testing classification methods. All of them are available from the UCI Repository of Machine Learning [BM98]. They are **bcw**, **hepatitis**, **ionosphere**, **musk**, **pgs**, **pid**, and **wpbc**. Their dimensions and number of samples are described in Table 10.1.

All datasets have binary outputs in $\{-1, +1\}$ and inputs in $[-\infty, +\infty]$, thus meeting all requirements for the binary classification experiments conducted. Some databases had arbitrary (non-numeric) binary dimensions which were converted to the $\{-1, +1\}$

domain.

Table 10.1: Datasets and their partitions for experiments described in this section.

set	dimensions	dataset size
bcw	9	699
hepatitis	19	155
ionosphere	34	351
musk	166	476
pgs	114	106
pid	8	768
wpbc	33	198

The following sections describe each database used, bringing details about the data and literature references when available.

10.1.1 Wisconsin Breast Cancer Database (bcw)

This breast cancer database was obtained from the UCI Repository of Machine Learning Databases [BM98]. It was originally created at the University of Wisconsin Hospitals, Madison, by Dr. William H. Wolberg. It describes a binary classification problem that labels tissue samples with cancer as *benign* or *malignant*. Each tissue sample underwent a series of cytological tests that described its cells' characteristics, creating feature-vectors with 9 dimensions of continuous values that form the input of the problem. There are 699 samples, 458 (65.5%) being benign, 241 (34.5%) being malignant.

A subset of this database has been studied before by Wolberg and Mangasarian [WM90], who used a multi-surface method of separating hyperplanes pairs to divide

the multidimensional input space in two halves. The subset had only 369 instances then, which were all tissue samples collected so far. They report the result to be consistent with 50% of the dataset when using two pairs, and 93.5% accurate on the remaining 50%. When using three pairs, they report the result to be consistent with 67% of the dataset, and 95.9% accurate on the remaining 33%.

Zhang also reports experiments with this same database subset [Zha92], to which he applied a nearest-neighbors classifier. The average of ten executions of the 1-nearest-neighbor algorithm yielded 93.7% accuracy, where the training set was a random sample with 54% of the instances, and the testing set was the remaining 46%.

This database originally contained 16 missing values in one of its input dimensions. These missing values are due to a change in the way data were collected during the base construction. Since most methods are not tolerant to missing data, each missing value has been replaced by zero for the purpose of conducting the experiments described in this work.

10.1.2 Hepatitis Domain (hepatitis)

This database was obtained from the UCI Repository of Machine Learning Databases [BM98]. It describes a binary classification problem that distinguishes patients with hepatitis between those who died, and those who managed to survive the disease. Each sample consists of 13 binary and 6 continuous dimensions that describe patients' characteristics, symptoms and clinical test results. There are 155 instances of patients, of whom 32 died and 123 lived.

This prediction problem has been studied before by Cestnik et al. [CKB87]. They proposed a new machine learning algorithm called Assistant-86, which performed with 83% accuracy over the database.

This database contained 167 missing values scattered across different input dimensions in a few samples. These missing values are due to lack of specific information about patients during the base construction. Since Support Vector Machines are not tolerant to missing data, each missing value has been replaced by zero for the purpose of conducting experiments. All binary dimension representations in the database have been converted to the $\{-1, +1\}$ set, which is particularly useful when representing data blanks as zeros.

10.1.3 Johns Hopkins University Ionosphere Database (ionosphere)

Sigillito et al. investigated the problem of finding evidence of structure in the ionosphere using radar signals [SWHB89]. The data were collected by a system in Goose Bay, Labrador. This system consisted of a phased array of 16 high-frequency antennas with a total transmitted power of around 6.4 kilowatts. Targets were free electrons in the ionosphere. Good radar returns were those showing evidence of some type of structure in the ionosphere, and bad returns were those whose signals passed through the ionosphere. The learning problem was consisted of determining whether a reading is *good* or *bad*, based on a 34-dimensional real-valued feature vector. The database contains 351 instances of readings, where 126 are labeled as bad and 225 are labeled as good.

Sigillito et al. report investigating the problem with back-propagation and Perceptron neural networks. Using the first 200 equally-balanced instances for training, they found 90.7% accuracy with a linear Perceptron, 92% with a non-linear Perceptron, and 96% with a back-propagation-trained multi-layer Perceptron network.

10.1.4 Musk Database (musk)

This dataset describes a set of molecules of which some are judged by human experts to be musks and the remaining to be non-musks¹. The data is described as a binary classification problem that uses 166 real-valued features to describe these molecules according to their shape and conformation. There are 476 samples of molecules, of which 207 are found to be musks and 269 are found to be non-musks. This database was obtained from the UCI Repository of Machine Learning Databases [BM98].

Dietterich et al. have studied this problem with axis-parallel rectangles in [DLLP97], where they also report results on a second musk dataset. Using 10-fold cross-validation, they describe average, minimum and maximum generalization results for five of their proposed algorithms, 92.4% [87.0%–97.8%], 91.3% [85.5%–97.1%], 90.2% [84.2%–96.3%], 83.7% [76.2%–91.2%], 80.4% [72.3%–88.5%], respectively. These results are also compared to those of a back-propagation neural network with 127 hidden units, 75.0% [66.2%–83.8%], and to those of C4.5 (pruned), 68.5% [40.9%–61.3%].

10.1.5 Escherichia coli promoter gene sequences (DNA) (pgs)

The problem of determining promoter gene sequences is one of the classic problems of bioinformatics. Many studies about the bacterium Escherichia coli have been published, both in the light of biological sciences and in the light of machine learning. Harley and Reynolds compiled and analyzed 263 promoters with known transcriptional start points for Escherichia coli DNA genes in [HR87], which served as basis for many other developments. Based on their results, Towell et al. used a 106-instance compiled database to

¹According to the Webster's Revised Unabridged Dictionary [Por98], *musk* is a substance of a reddish brown color, and when fresh of the consistence of honey, obtained from a bag being behind the navel of the male musk deer. It has a slightly bitter taste, but is especially remarkable for its powerful and enduring odor. It is used in medicine as a stimulant antispasmodic. The term is also applied to secretions of various other animals, having a similar odor.

evaluate a hybrid learning algorithm that uses examples to inductively refine preexisting knowledge [TSN90]. Using a leave-one-out methodology, they obtained generalization errors as low as (4/106), which they compared against other machine learning techniques such as a back-propagation neural network (8/106), nearest-neighbor algorithm (13/106), Quinlan’s decision tree builder (19/106) [Qui92], and O’Neill (12/106), which is an ad hoc technique from the biological literature [O’N89, OC89].

The database contains 106 instances, of which 53 are promoters and 53 are not promoters. For each sample, the original database contained a sequence of 57 bases, each being either adenine (A), cytosine (C), guanine (G), or thymine (T). These bases were naively translated from the $\{A, G, T, C\}$ input space to the $\{-1, +1\}^2$ space, each base having a unique pair in $\{-1, +1\}^2$, hence losing all sequence information in purpose. Each base originated two independent dimensions, thus forming a 114-dimensional binary input space. Finally, the output space was also translated to the $\{-1, +1\}$ binary domain. The original sequence database derived by [HR87] and used by [TSN90] was obtained from the UCI Repository of Machine Learning Databases [BM98]. Other encoding examples of gene classification problems may be found in [BGL⁺99, KK01].

10.1.6 Pima Indians Diabetes Database (pid)

This database describes another case of disease diagnosis problem, this time of diabetes mellitus on a specific population of Pima American Indians. Diagnostics were binary, determining whether each patient showed signs of diabetes according to World Health Organization criteria, that is, if their 2-hour post-load plasma glucose was at least 200 mg/dl at any survey examination, or if found during routine medical care. There are 768 samples of patients, 500 of which tested negative for diabetes and the remaining 278 tested positive. Each sample contains 8 real-valued features describing patients’ blood

test results, body mass, and age. This database was obtained from the UCI Repository of Machine Learning Databases [BM98].

Smith et al. studied this problem using the ADAP algorithm [SED⁺88]. This algorithm makes continuous predictions in $[0, 1]$, which were discretized using a cutoff function. Using 75% of the data as a training set, they report 76% success in predicting the remaining testing set.

10.1.7 Wisconsin Prognostic Breast Cancer (wpbc)

A follow-up study of the breast cancer problem described in Sections 10.1.1 consists of determining cancer prognostics, that is, whether or not patients experienced recurrence of the disease after there had been no further symptoms for a predetermined amount of time (24 months). Wolberg, Street and Mangasarian used the same 30 features from their previous work [MSW95], together with 3 new features that described properties of the recurrent disease [SMW95]. This binary problem is therefore made of a 33-dimensional input space of real values, and a binary output that corresponds to labels *recurred* and *did not recur*. This database, as described in [SMW95], is available from the UCI Repository of Machine Learning Databases [BM98].

Wolberg, Street and Mangasarian report results using the same technique of separating hyperplanes in a restricted 4-dimensional subspace of the original 33-dimensional input space. They obtained 86.3% accuracy over the 198 input instances, of which 151 were labeled *did not recur* and 47 were labeled *recurred*.

10.2 Sampling

All experiments conducted in this work have been repeated 10 times to avoid statistical flukes. For each experiment, training set data and working set data have been drawn from the original datasets, described in Section 10.1, according to two criteria. These criteria, namely the probability distribution from which the data were drawn and the proportion between training and working data, are described in the sections ahead.

10.2.1 Biased and Unbiased Sampling

As shown in Section 2.1, one of the assumptions about the general definition of the learning problem is that data are drawn i.i.d. (identically and independently distributed) with respect to a given distribution. In order to test the behavior of our method, as well as others, when this assumption is broken, we have repeated every experiment with two setups:

Unbiased distributions Where all data sampled to assemble training and working sets were drawn i.i.d. with respect to a given fixed uniform probability distribution.

Biased distributions Where all data sampled were drawn probabilistically with respect to a unimodal distribution centered at the first randomly-chosen sample and with weights proportional to the distance of each sample to the center.

Results for all experiments are reported for each of these two setups, so we may evaluate some of the effects of the sampling criteria on the behavior of learning methods.

10.2.2 Training and Working Set Sizes

The sizes of the training set and the working set are very important to define the behavior of inductive and transductive learning methods. The more data a learning method receives in its training set, the more information it will have about the underlying probability distributions of the problem. In supervised problems, that is all the information the methods will have to come up with a solution. In semi-supervised learning problems, though, the working set data also enhance the method's knowledge about these distributions.

Additionally, the size of the working set has no influence on the behavior of inductive methods (in an inductive setup, the working set is referred to as testing set). That is not the case for transductive methods, though, where the number of samples in the working set equal the amount of answers that must be contained in the solution of the problem. In other words, although the larger the working set the more information about the method receives, it also means the harder it is to solve the learning problem.

In this work, all experiments were conducted with 9 different proportions between training set size and working set size. We started with experiments where 10% of all data in the original dataset were drawn and assigned to the training set, and the remaining 90% were assigned to the working set. Using 10% intervals, we worked our way up until 90% of data were drawn and assigned to the training set, and the remaining 10% were assigned to the working set.

10.3 Information Indicators

Appendix A shows the indicators introduced in Chapter 9, evaluated for all combinations of sampling strategy and set sizes.

Chapter 11

Benchmark Results

This chapter brings the results of benchmark experiments conducted with all datasets sampled in this work, so we may later compare these results to those of our proposed method. These benchmark experiments were done with well-known inductive and transductive algorithms, described in Section 11.1. In order to tune the parameters for these algorithms, we followed the procedure described in Section 11.2. All benchmark results are shown in Section B.1 and B.2 with biased and unbiased sampling, respectively.

11.1 Benchmark Methods

In this section we start our experiments with the datasets described in Chapter 10. Before the experiments with the method proposed in Part III, we examine results for these datasets, in all configurations, for two well known learning algorithms:

SVM Standard inductive SVM. The training algorithm used was SVM^{light} , described in Section 3.5.2 and introduced in [Joa98]. The implementation used was provided by the author, version 6.01, implemented in C.

TSVM Transductive SVM. The training algorithm used was also SVM^{light} , which incorporates the iterative transductive algorithm known as TSVM, described in [Joa99]. The implementation used was provided by the author, version 6.01, implemented in C.

Notice that instead of using an exclusively inductive training algorithm, such as SMO, we selected the same training algorithm for both SVM and TSVM, namely SVM^{light} . This way, we shall avoid discrepancies in results between SVM and TSVM occurred due to differences in the strategy of each training algorithm.

11.2 Parameter Tuning

As opposed to the method proposed in Part III, both SVM and TSVM depend on specific parameters for each learning problem. More specifically, they require a trade-off parameter C , which regulates the bias between the training error and the margin, and an inner-product kernel function, where one may select the function itself as well as its governing parameters. Please refer to Section 3.2 for a review on kernel functions and Section 3.3.3 for a review on soft-margin SVMs and the regularization parameter.

In order to select the parameters for each dataset, we ran exhaustive tests using a standard inductive SVM. All datasets were cropped to 100 samples, then split into training sets and testing sets, the first with 70% and the latter with 30% of the samples. We then trained and evaluated the SVM with all possible kernel and kernel parameter combinations. All combinations were tested 10 times, each one with new samplings of training and testing sets.

For the trade-off parameter C , the possible values were chosen from $\{10^{-3}, 10^0, 10^3\}$. The kernel parameter was either *linear*, *polynomial*, or *radial-basis function* (RBF). Each

of the parameters of these kernel functions were selected from the same set of possible values $\{10^{-3}, 10^0, 10^3\}$.

After 10 executions for each combination, the parameter setup with best accuracy for a given dataset was determined. The results, shown in Table 11.1, were used in all benchmark experiments described in Section B.1. This parameter tuning procedure is the same as described in [MBdC07].

Table 11.1: Results from parameter tuning procedure, adopted for experiments with SVM and TSVM.

set	C	kernel	σ
bcw	0.001	linear	-
hepatitis	1000	rbf	1
ionosphere	1	rbf	1
musk	1000	rbf	1000
pgs	1000	linear	-
pid	0.001	linear	-
wpbc	1000	rbf	1

11.3 Reference Results

Appendix B shows benchmark results for each of the datasets used in this work with biased and unbiased sampling. Complete comparative analyses of these results are given in Section 12.4.

Chapter 12

Experimental Results

This chapter brings the results of all experiments conduct with the ETEC method proposed in Chapter 8. We start in Section 12.1 where we describe the additional information automatically incorporated to the datasets prior to the solution of the optimization problem, which may be interpreted as being part of the proposed method itself. Next, in Section 12.2 we describe the results obtained with all datasets, for both their biased and unbiased versions. Section 12.3 shows all results, including benchmark results and indicators, in graphical format. We conclude the chapter in Section 12.4, where we analyze the results from Sections 12.2 and 12.3.

12.1 Incorporating Additional Information

In addition to the actual sample data contained in the training and working sets drawn for each experiment, we have incorporated additional information to these original sets by including 4 new dimensions before feeding the ETEC algorithm. This procedure may be interpreted as part of the method itself, as these dimensions are also based on information theory. Nevertheless, we chose not to describe it with the ETEC definition from

Chapter 8, but as part of the experimental methodology.

Consider the familiar transductive learning problem where the training set is divided into two sets A and B , each one representing a class, and there is an unlabeled working set T . In this preprocessing step, if the data in these sets contain n dimensions, each one of the samples will receive 4 additional features, hence yielding data with $(n + 4)$ dimensions. This extended dataset is the one actually given as input to the ETEC algorithm.

For each set A , B , and T , these dimensions have different definitions. For the partial training set A , the dimensions added to each sample $x \in A$ were the following:

Dimension $(n + 1)$, $x \in A$: $H(A)$ (entropy of A)

Dimension $(n + 2)$, $x \in A$: $H(B \cup x)$ (entropy of extended B' (with $x \in B'$))

Dimension $(n + 3)$, $x \in A$: $H(A) - H(B \cup x)$

Dimension $(n + 4)$, $x \in A$: $H(A)/H(B \cup x)$

For samples in the remaining part of the training set, B :

Dimension $(n + 1)$, $x \in B$: $H(B)$ (entropy of B)

Dimension $(n + 2)$, $x \in B$: $H(A \cup x)$ (entropy of extended A' (with $x \in A'$))

Dimension $(n + 3)$, $x \in B$: $H(A \cup x) - H(B)$

Dimension $(n + 4)$, $x \in B$: $H(A \cup x)/H(B)$

For samples in the working set T :

Dimension $(n + 1)$, $x \in T$: $H(A \cup x)$ (entropy of extended A' (with $x \in A'$))

Dimension $(n + 2)$, $x \in T$: $H(B \cup x)$ (entropy of extended B' (with $x \in B'$))

Dimension $(n + 3), x \in T : H(A \cup x) - H(B \cup x)$

Dimension $(n + 4), x \in T : H(A \cup x)/H(B \cup x)$

For samples in the training set, these features measure the entropy effects of having each sample belonging to the opposite class. For samples in the working set, they measure the disturbance caused in each class when samples are individually incorporated to its set.

12.2 Results

The next sections show the experimental results for each of the datasets used in this work. Complete comparative analyses of these results are given in Section 12.4.

12.2.1 Results for bcw

ETEC results for the **bcw** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.1 and 12.2 respectively.

Table 12.1: ETEC results for **bcw**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	95.27% $\pm$ 1.50%
20%	80%	96.36% $\pm$ 0.82%
30%	70%	96.75% $\pm$ 0.48%
40%	60%	96.62% $\pm$ 0.74%
50%	50%	96.43% $\pm$ 0.90%
60%	40%	96.57% $\pm$ 0.91%
70%	30%	96.48% $\pm$ 1.01%
80%	20%	95.43% $\pm$ 2.43%
90%	10%	89.57% $\pm$ 8.17%

Table 12.2: ETEC results for **bcw**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	95.65% $\pm$ 1.52%
20%	80%	96.25% $\pm$ 0.73%
30%	70%	96.49% $\pm$ 0.53%
40%	60%	96.55% $\pm$ 0.56%
50%	50%	96.63% $\pm$ 0.84%
60%	40%	96.57% $\pm$ 0.54%
70%	30%	95.71% $\pm$ 1.65%
80%	20%	92.64% $\pm$ 5.10%
90%	10%	87.00% $\pm$ 3.95%

12.2.2 Results for hepatitis

ETEC results for the **hepatitis** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.3 and 12.4 respectively.

Table 12.3: ETEC results for **hepatitis**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$79.21\% \pm 2.74\%$
20%	80%	$79.35\% \pm 3.14\%$
30%	70%	$77.22\% \pm 3.41\%$
40%	60%	$73.91\% \pm 4.38\%$
50%	50%	$68.69\% \pm 10.56\%$
60%	40%	$63.33\% \pm 10.95\%$
70%	30%	$59.32\% \pm 13.40\%$
80%	20%	$47.86\% \pm 18.15\%$
90%	10%	$45.83\% \pm 16.31\%$

Table 12.4: ETEC results for **hepatitis**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$77.71\% \pm 6.08\%$
20%	80%	$79.52\% \pm 3.02\%$
30%	70%	$77.13\% \pm 4.17\%$
40%	60%	$68.37\% \pm 19.55\%$
50%	50%	$73.29\% \pm 6.80\%$
60%	40%	$59.83\% \pm 15.66\%$
70%	30%	$57.73\% \pm 12.13\%$
80%	20%	$46.79\% \pm 16.19\%$
90%	10%	$41.67\% \pm 22.22\%$

12.2.3 Results for ionosphere

ETEC results for the **ionosphere** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.5 and 12.6 respectively.

Table 12.5: ETEC results for **ionosphere**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	79.05% $\pm$ 7.11%
20%	80%	83.54% $\pm$ 6.16%
30%	70%	85.62% $\pm$ 4.99%
40%	60%	86.01% $\pm$ 5.38%
50%	50%	84.19% $\pm$ 7.31%
60%	40%	87.94% $\pm$ 4.08%
70%	30%	87.40% $\pm$ 5.13%
80%	20%	75.78% $\pm$ 17.09%
90%	10%	73.93% $\pm$ 16.33%

Table 12.6: ETEC results for **ionosphere**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	84.68% $\pm$ 4.71%
20%	80%	84.79% $\pm$ 4.46%
30%	70%	87.87% $\pm$ 2.84%
40%	60%	87.59% $\pm$ 3.39%
50%	50%	89.82% $\pm$ 2.14%
60%	40%	90.66% $\pm$ 2.73%
70%	30%	88.20% $\pm$ 7.04%
80%	20%	79.84% $\pm$ 13.57%
90%	10%	59.29% $\pm$ 20.57%

12.2.4 Results for musk

ETEC results for the **musk** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.7 and 12.8 respectively.

Table 12.7: ETEC results for **musk**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$57.23\% \pm 5.62\%$
20%	80%	$56.85\% \pm 5.58\%$
30%	70%	$63.27\% \pm 5.54\%$
40%	60%	$67.44\% \pm 6.45\%$
50%	50%	$69.11\% \pm 4.98\%$
60%	40%	$73.49\% \pm 5.18\%$
70%	30%	$71.63\% \pm 5.74\%$
80%	20%	$69.35\% \pm 7.45\%$
90%	10%	$69.11\% \pm 10.65\%$

Table 12.8: ETEC results for **musk**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	56.32% $\pm$ 6.96%
20%	80%	57.17% $\pm$ 4.06%
30%	70%	64.17% $\pm$ 5.52%
40%	60%	67.79% $\pm$ 8.92%
50%	50%	68.73% $\pm$ 9.64%
60%	40%	72.33% $\pm$ 7.30%
70%	30%	72.34% $\pm$ 7.82%
80%	20%	69.46% $\pm$ 6.09%
90%	10%	64.45% $\pm$ 8.64%

12.2.5 Results for pgs

ETEC results for the **pgs** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.9 and 12.10 respectively.

Table 12.9: ETEC results for **pgs**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	50.31% $\pm$ 4.42%
20%	80%	60.71% $\pm$ 13.37%
30%	70%	72.57% $\pm$ 16.17%
40%	60%	76.51% $\pm$ 9.84%
50%	50%	74.62% $\pm$ 9.92%
60%	40%	73.17% $\pm$ 9.13%
70%	30%	60.00% $\pm$ 11.11%
80%	20%	61.58% $\pm$ 14.48%
90%	10%	70.00% $\pm$ 22.97%

Table 12.10: ETEC results for **pgs**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	48.75% $\pm$ 1.61%
20%	80%	61.06% $\pm$ 11.69%
30%	70%	62.16% $\pm$ 9.81%
40%	60%	68.09% $\pm$ 9.80%
50%	50%	65.58% $\pm$ 10.78%
60%	40%	60.49% $\pm$ 11.77%
70%	30%	54.67% $\pm$ 13.53%
80%	20%	56.31% $\pm$ 8.96%
90%	10%	67.50% $\pm$ 18.82%

12.2.6 Results for pid

ETEC results for the **pid** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.11 and 12.12 respectively.

Table 12.11: ETEC results for **pid**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$72.31\% \pm 2.30\%$
20%	80%	$70.62\% \pm 6.38\%$
30%	70%	$73.12\% \pm 2.13\%$
40%	60%	$73.27\% \pm 2.63\%$
50%	50%	$75.00\% \pm 2.59\%$
60%	40%	$71.89\% \pm 3.29\%$
70%	30%	$69.39\% \pm 3.44\%$
80%	20%	$69.08\% \pm 4.46\%$
90%	10%	$68.82\% \pm 7.05\%$

Table 12.12: ETEC results for **pid**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$71.82\% \pm 3.16\%$
20%	80%	$72.06\% \pm 4.25\%$
30%	70%	$70.98\% \pm 9.41\%$
40%	60%	$72.56\% \pm 6.79\%$
50%	50%	$72.34\% \pm 3.53\%$
60%	40%	$70.69\% \pm 4.06\%$
70%	30%	$65.65\% \pm 3.30\%$
80%	20%	$66.08\% \pm 3.87\%$
90%	10%	$66.45\% \pm 4.53\%$

12.2.7 Results for **wpb**

ETEC results for the **wpb** dataset, drawn with biased and unbiased sampling, are shown in Tables 12.13 and 12.14 respectively.

Table 12.13: ETEC results for **wpbc**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	70.06% $\pm$ 6.45%
20%	80%	65.53% $\pm$ 5.73%
30%	70%	64.68% $\pm$ 9.39%
40%	60%	65.46% $\pm$ 7.21%
50%	50%	65.96% $\pm$ 4.74%
60%	40%	69.62% $\pm$ 4.70%
70%	30%	73.56% $\pm$ 5.60%
80%	20%	73.08% $\pm$ 5.57%
90%	10%	74.21% $\pm$ 10.36%

Table 12.14: ETEC results for **wpbc**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	56.14% $\pm$ 14.40%
20%	80%	57.23% $\pm$ 9.71%
30%	70%	60.58% $\pm$ 7.24%
40%	60%	64.62% $\pm$ 10.12%
50%	50%	64.14% $\pm$ 8.15%
60%	40%	69.37% $\pm$ 8.29%
70%	30%	69.32% $\pm$ 5.27%
80%	20%	77.69% $\pm$ 4.37%
90%	10%	74.74% $\pm$ 8.15%

12.3 Graphical Comparison

In this section we show all the results obtained in this work in graphical fashion. We integrate the benchmark results from Chapter 11 with all experimental results in Chapter 12 in the same charts, so the difference in performance between the SVM, TSVM, and ETEC methods may be evaluated.

Additionally, we also plot all indicators collected in Chapter 10 just below each of the performance charts, so one may visualize the influence of the quantities of information contained in the sets of the problems over the accuracy of their solution, for each of the variations between training and working sets.

When we plot our indicators, notice that all 3 first-order and 4 second-order indicators share the same unit, the nat. The third-order indicator, the transductivity measure (or t-measure, for short), is a ratio shown as a percentage, therefore with no units. For simplicity's sake, this indicator is plotted in the same chart as the other indicators, but on a secondary y-axis which is omitted to save space for the figure. The scale used is the same as the one for the other indicators, but read as a percentage. For instance, should a t-measure point lie over a 10-nat line, one should read 10%. In specific plots, as always indicated by the legends, this indicator may be multiplied or divided by 100 to improve its visibility, so one would read 1000% or 0.10%, respectively, for the same example.

Over the next 14 pages, we show Figures 12.1 to 12.14, which describe the results and indicators for all experiments conducted with each of the 7 datasets with the 2 sampling strategies used. Later, in Section 12.4, we shall refer again to these charts as well as the tables from which they draw their data, to analyze and comment on these results.

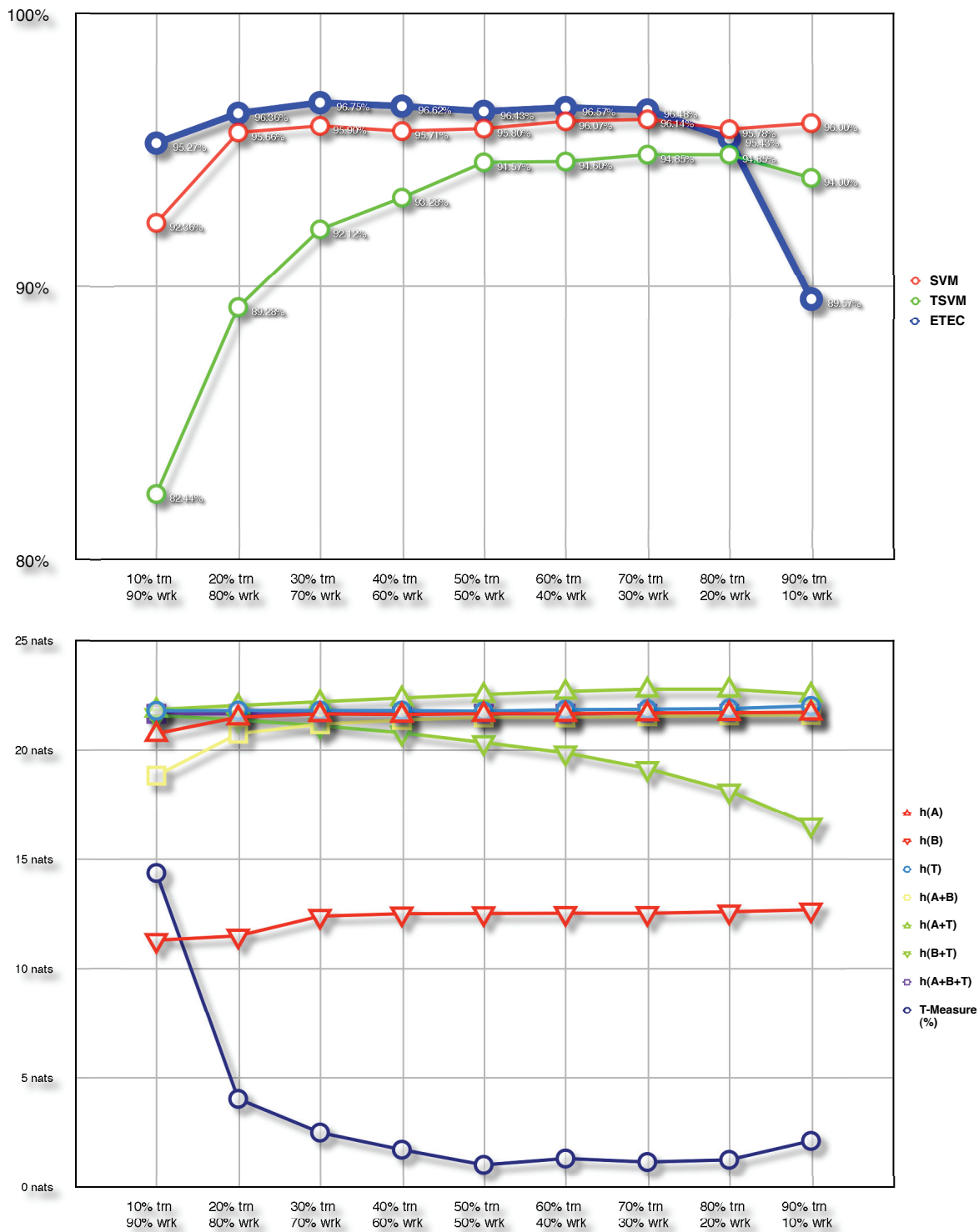


Figure 12.1: Graphical results and indicators for **bcw**, biased sampling.

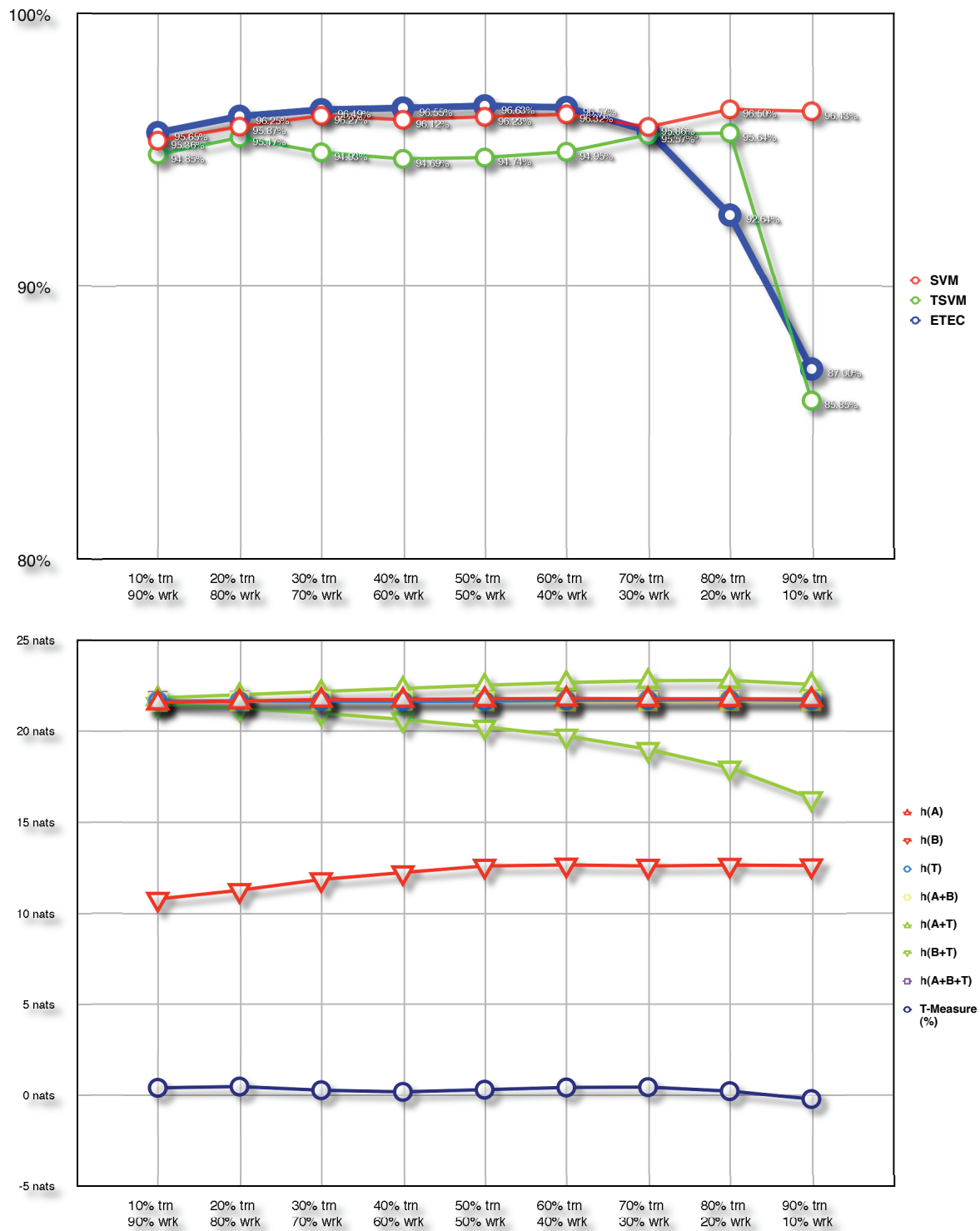


Figure 12.2: Graphical results and indicators for **bcw**, unbiased sampling.

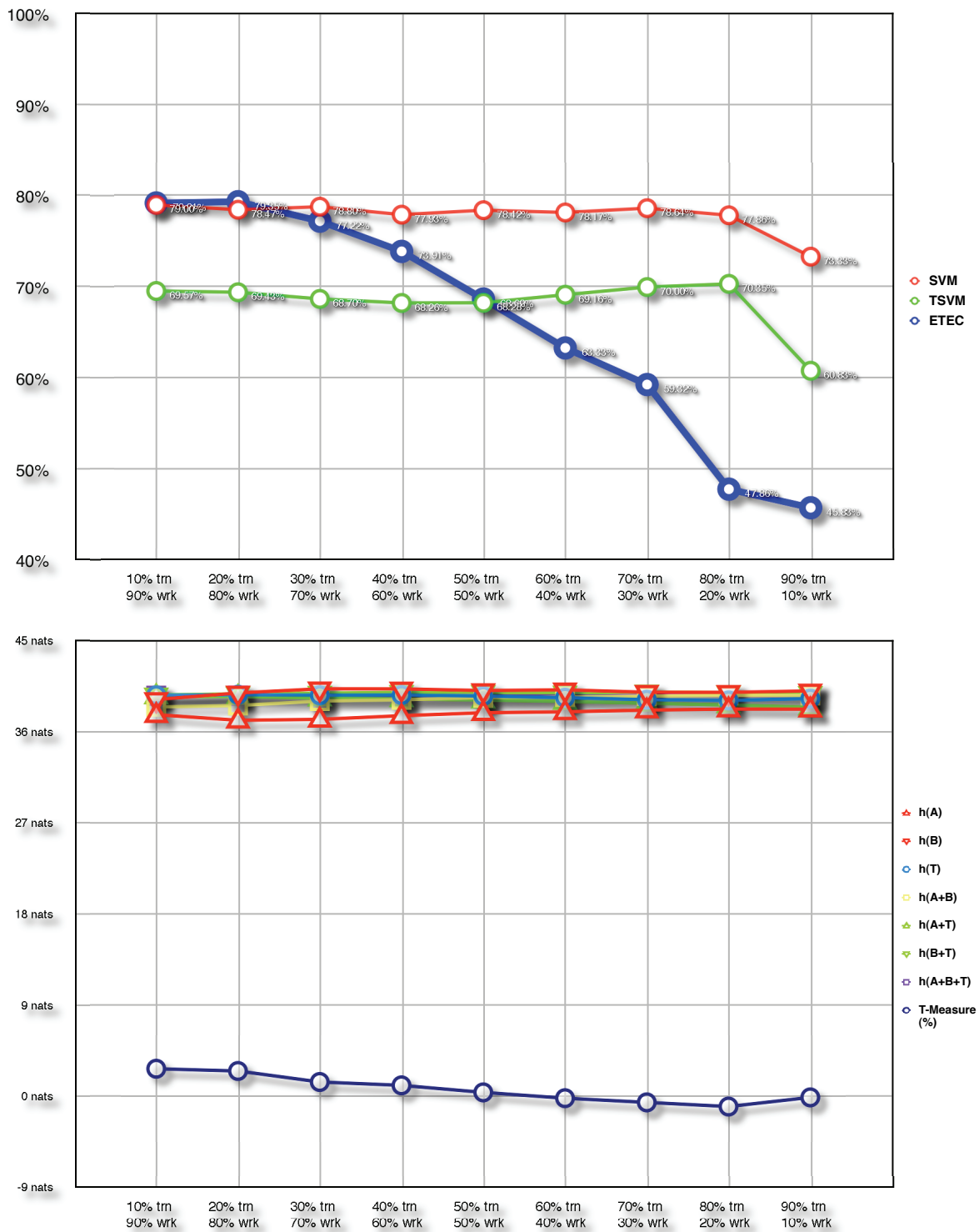


Figure 12.3: Graphical results and indicators for **hepatitis**, biased sampling.

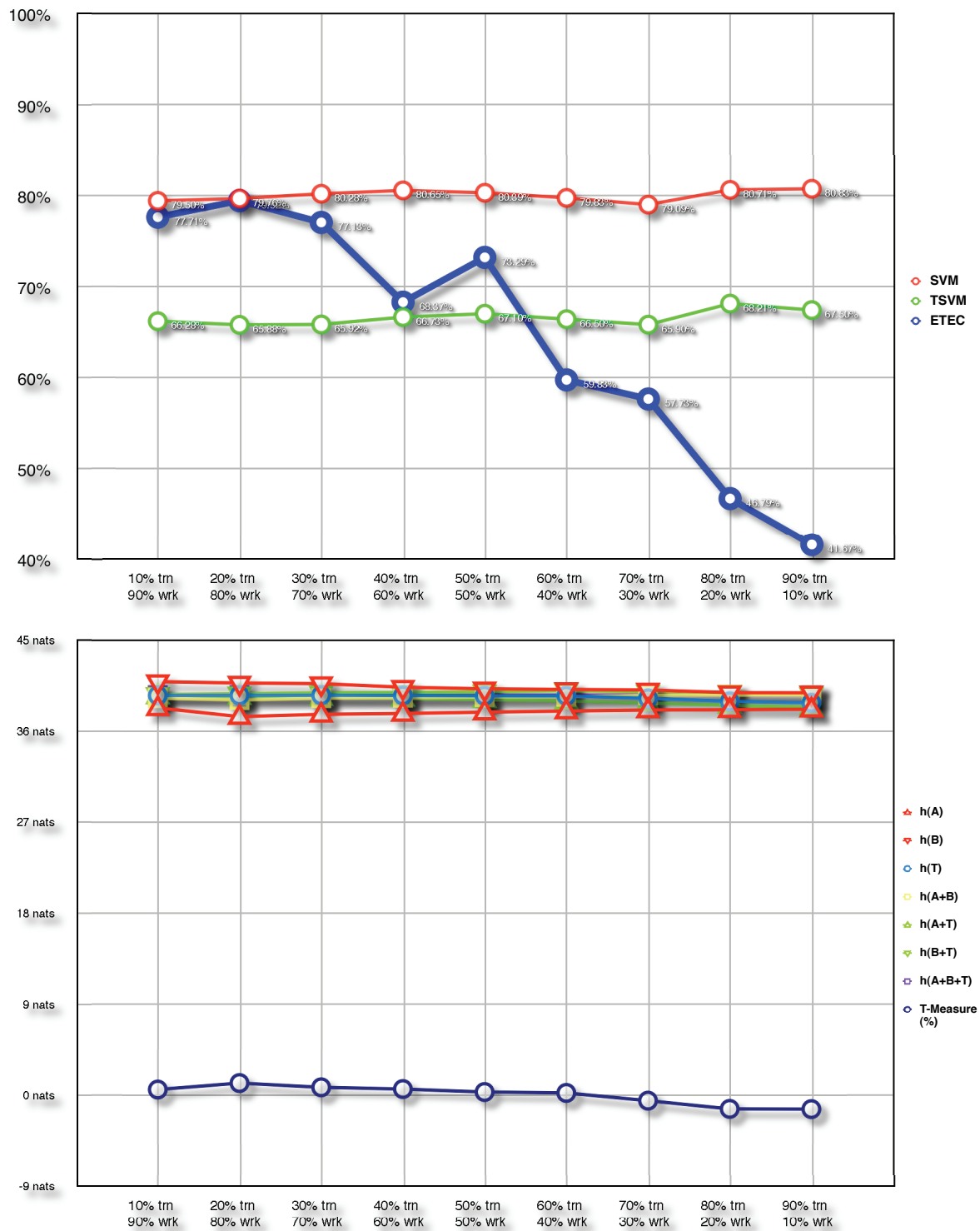


Figure 12.4: Graphical results and indicators for **hepatitis**, unbiased sampling.

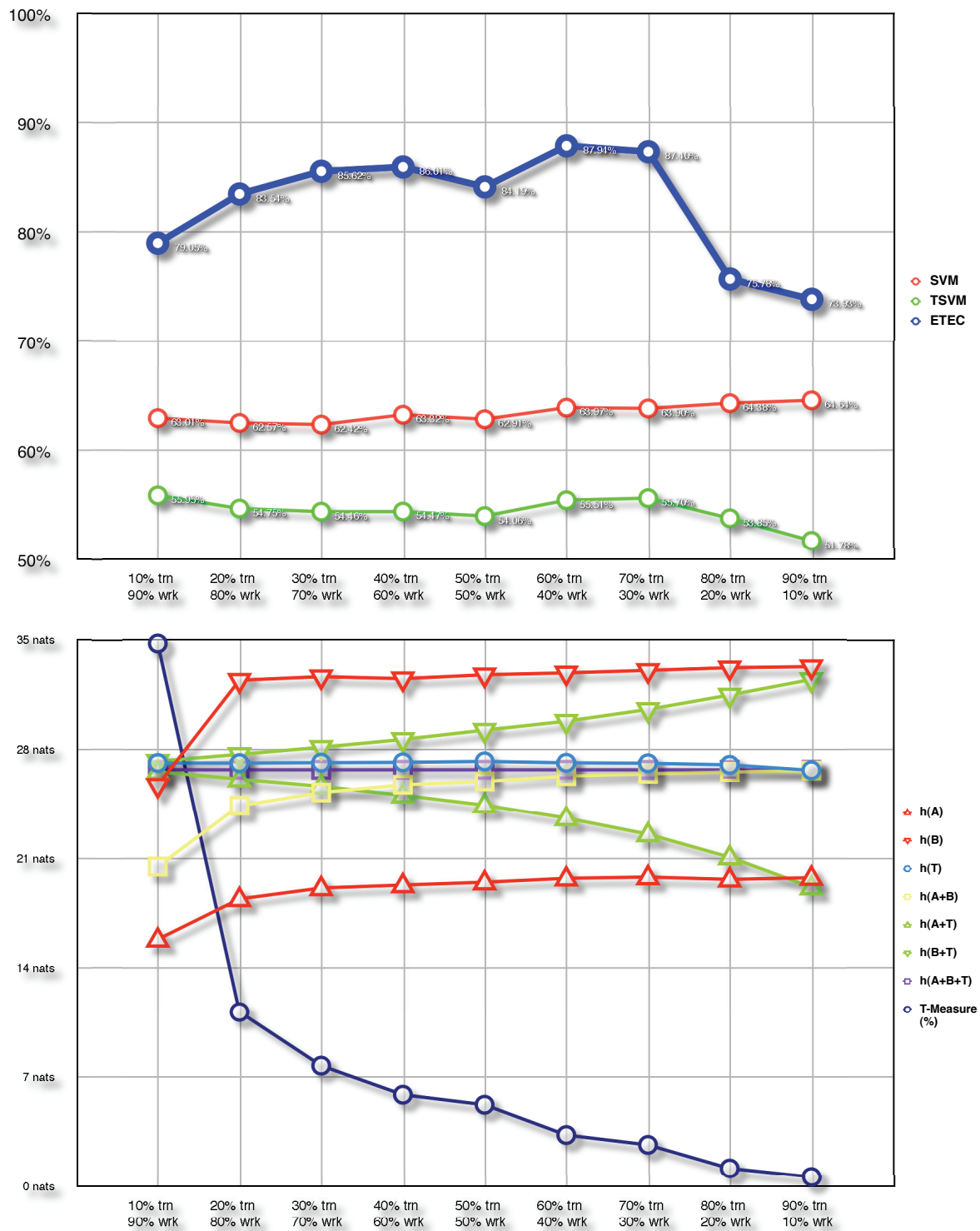


Figure 12.5: Graphical results and indicators for **ionosphere**, biased sampling.

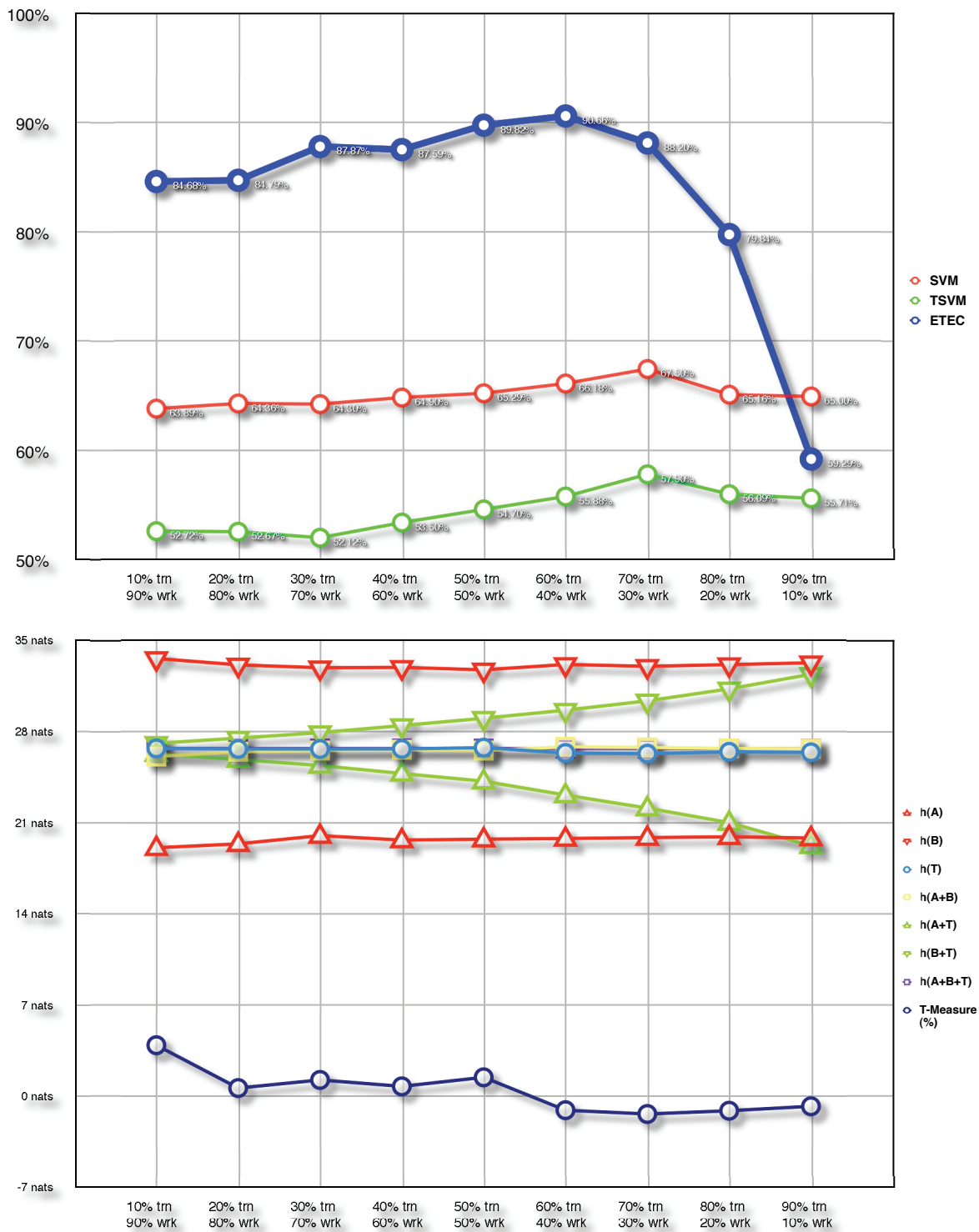


Figure 12.6: Graphical results and indicators for **ionosphere**, unbiased sampling.

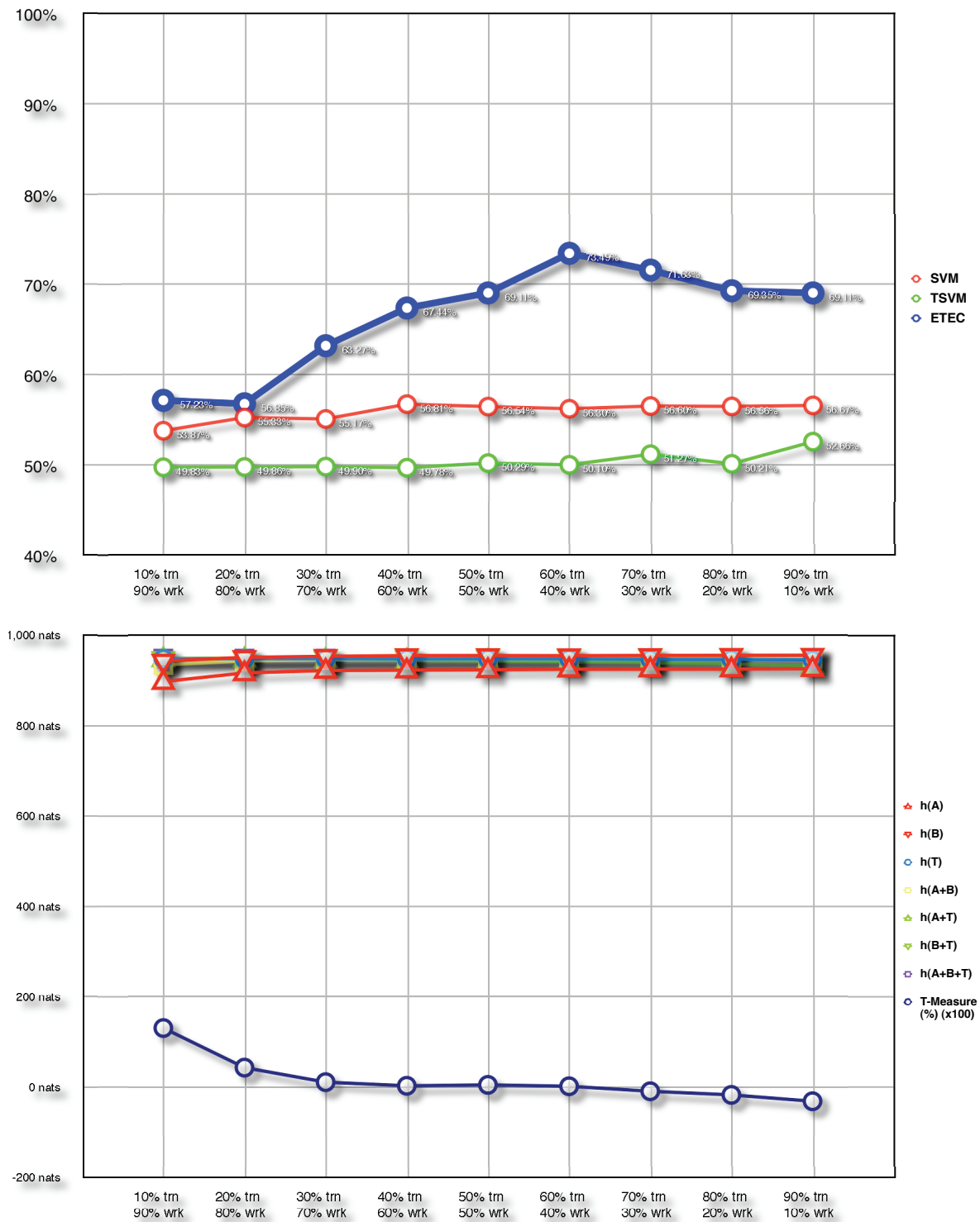


Figure 12.7: Graphical results and indicators for **musk**, biased sampling.

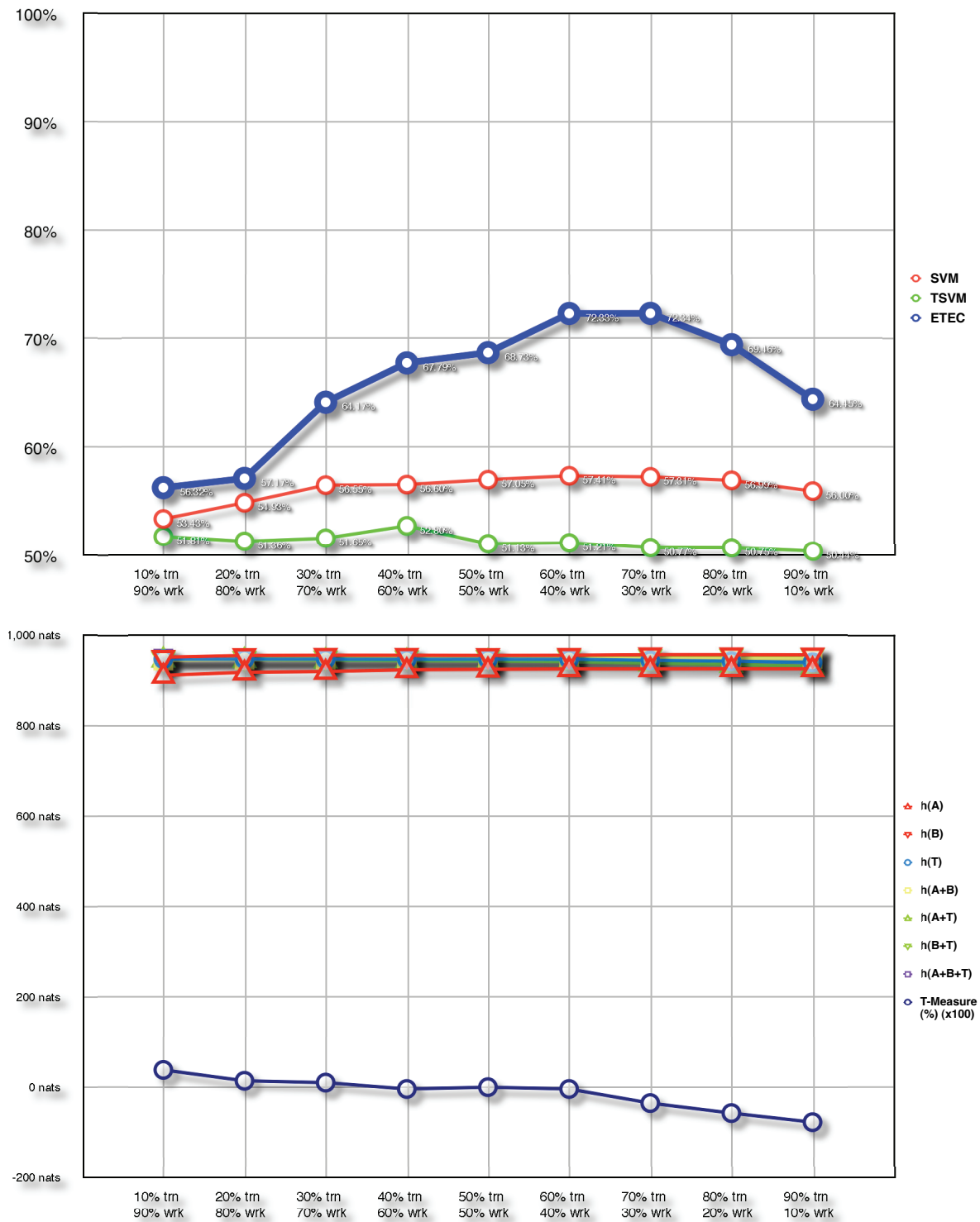


Figure 12.8: Graphical results and indicators for **musk**, unbiased sampling.

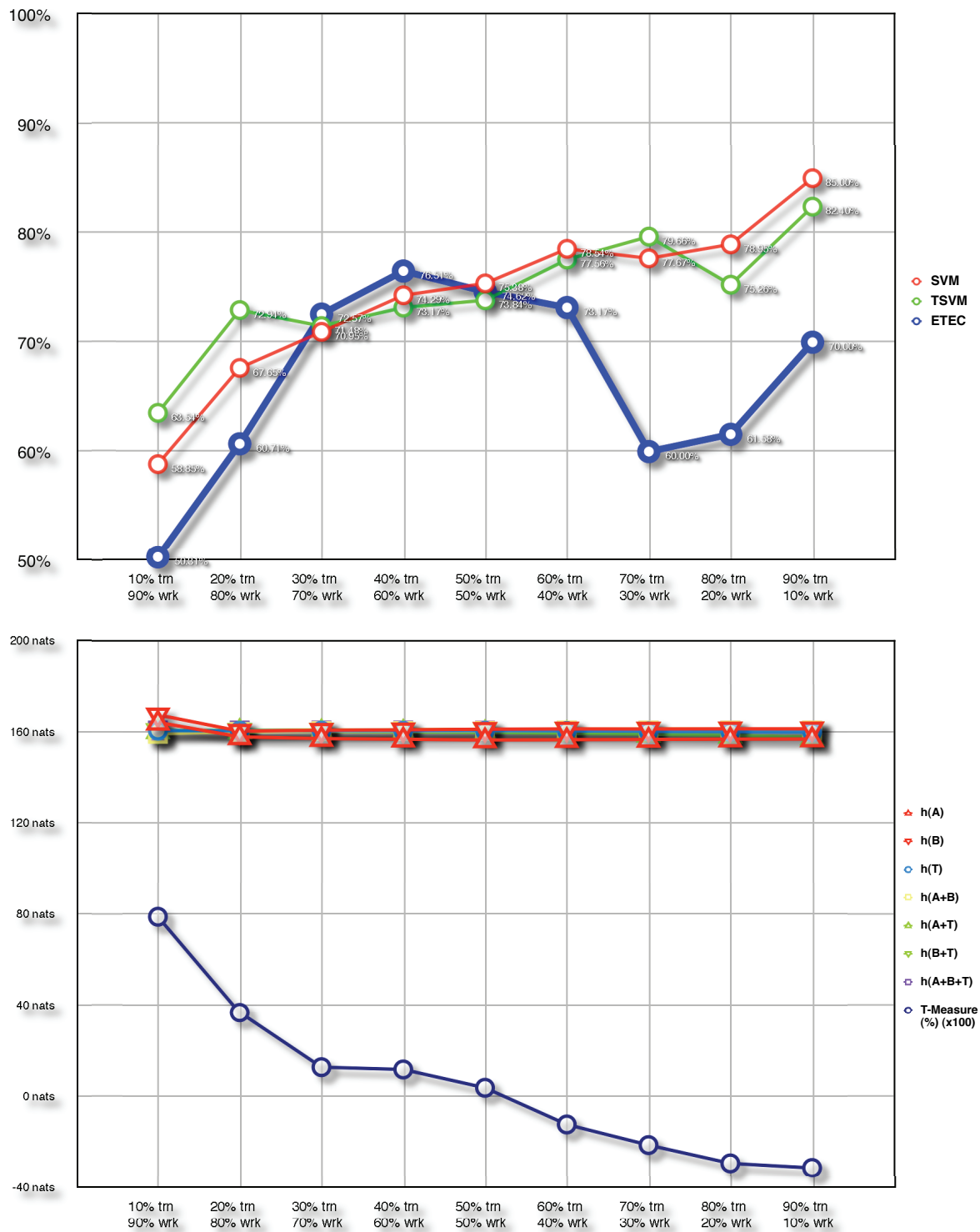


Figure 12.9: Graphical results and indicators for **pgs**, biased sampling.

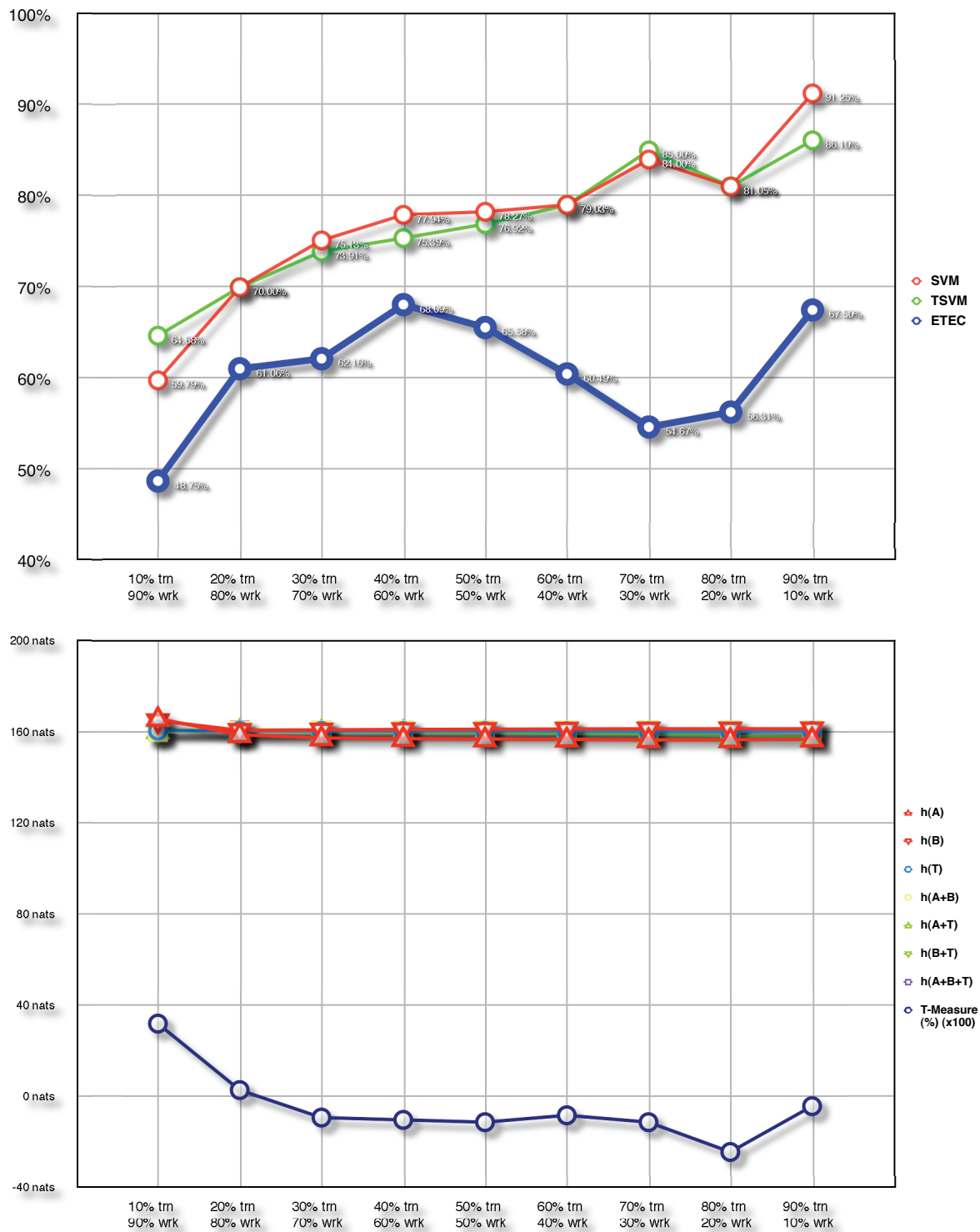


Figure 12.10: Graphical results and indicators for **pgs**, unbiased sampling.

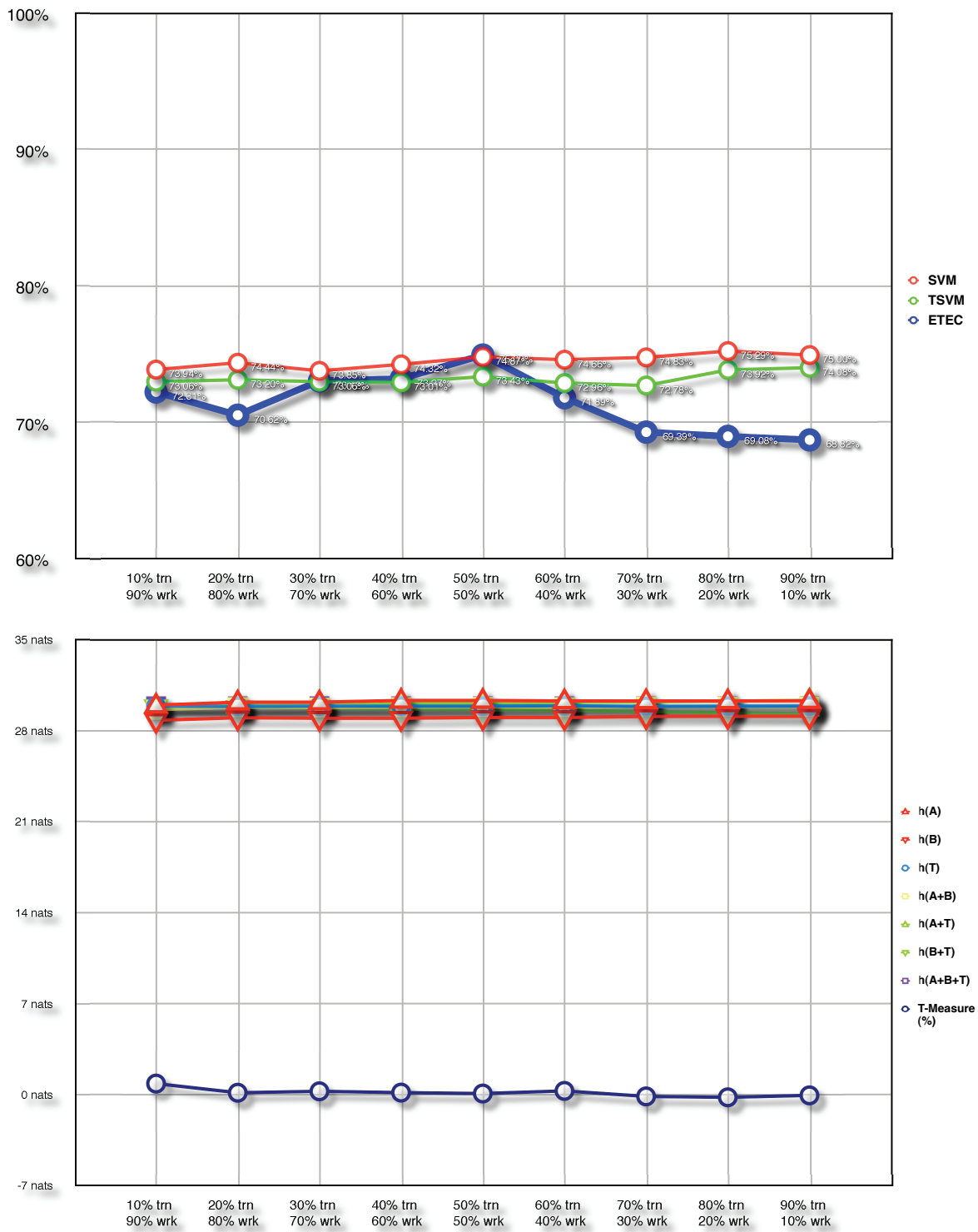


Figure 12.11: Graphical results and indicators for **pid**, biased sampling.

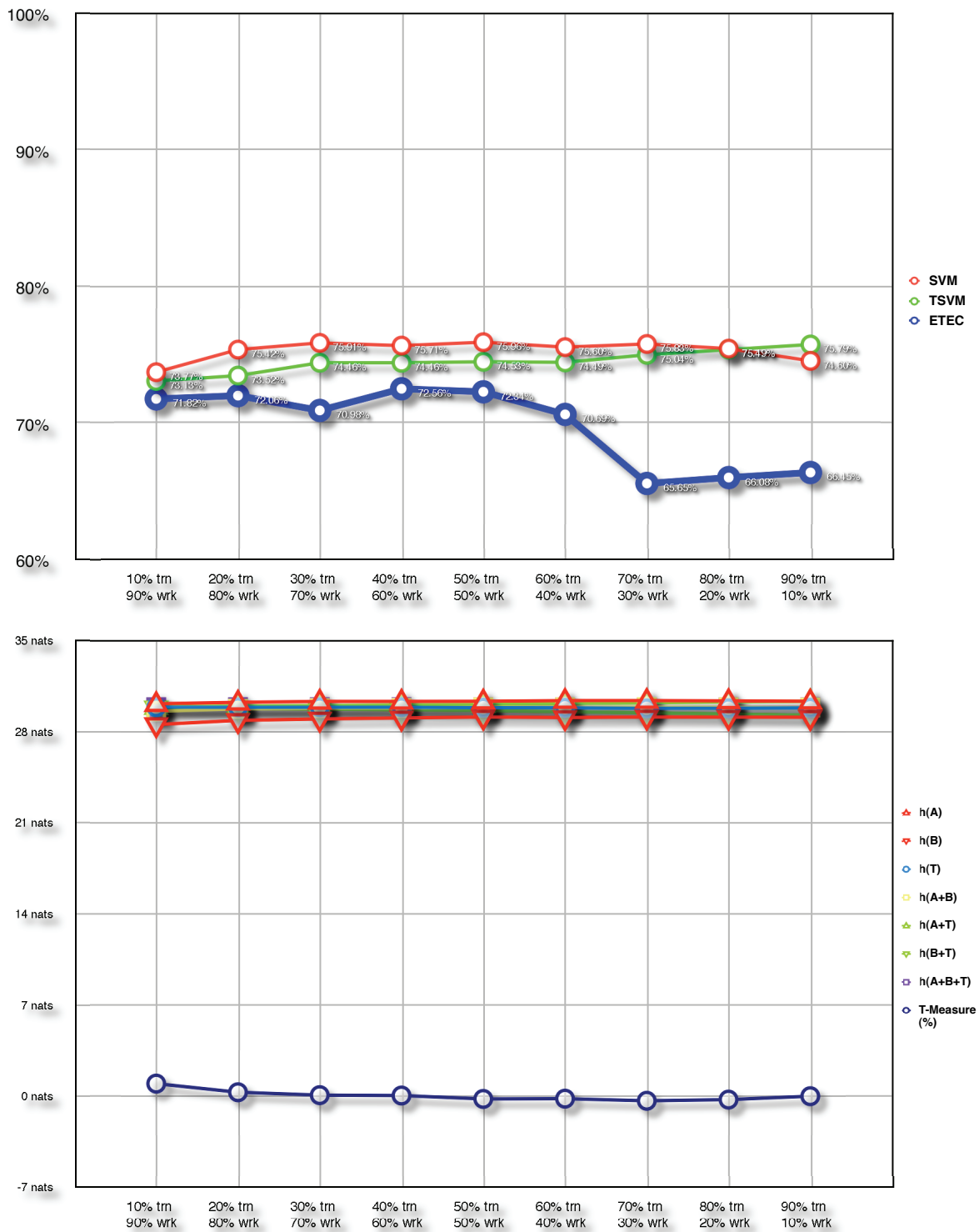


Figure 12.12: Graphical results and indicators for **pid**, unbiased sampling.

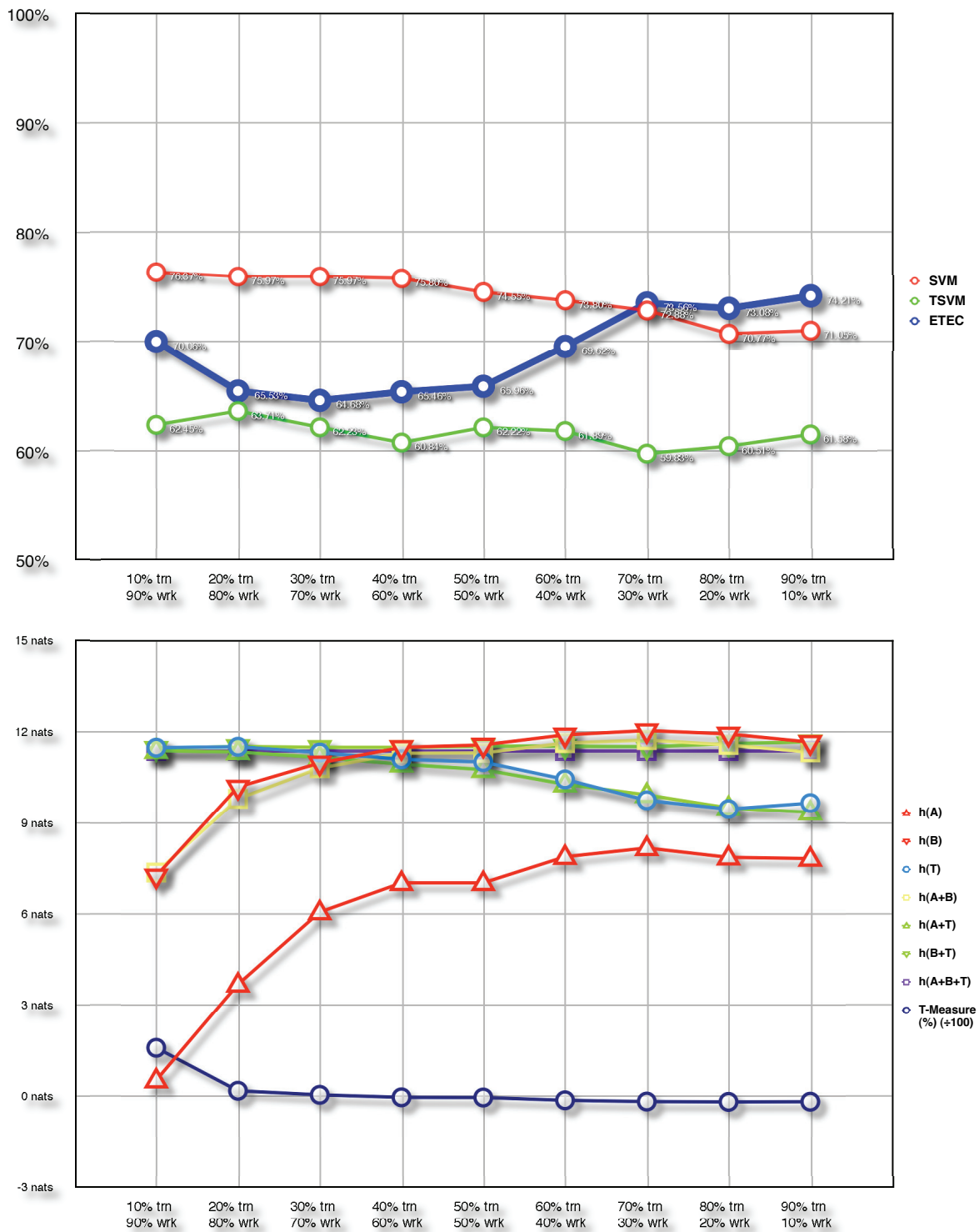


Figure 12.13: Graphical results and indicators for **wpbc**, biased sampling.

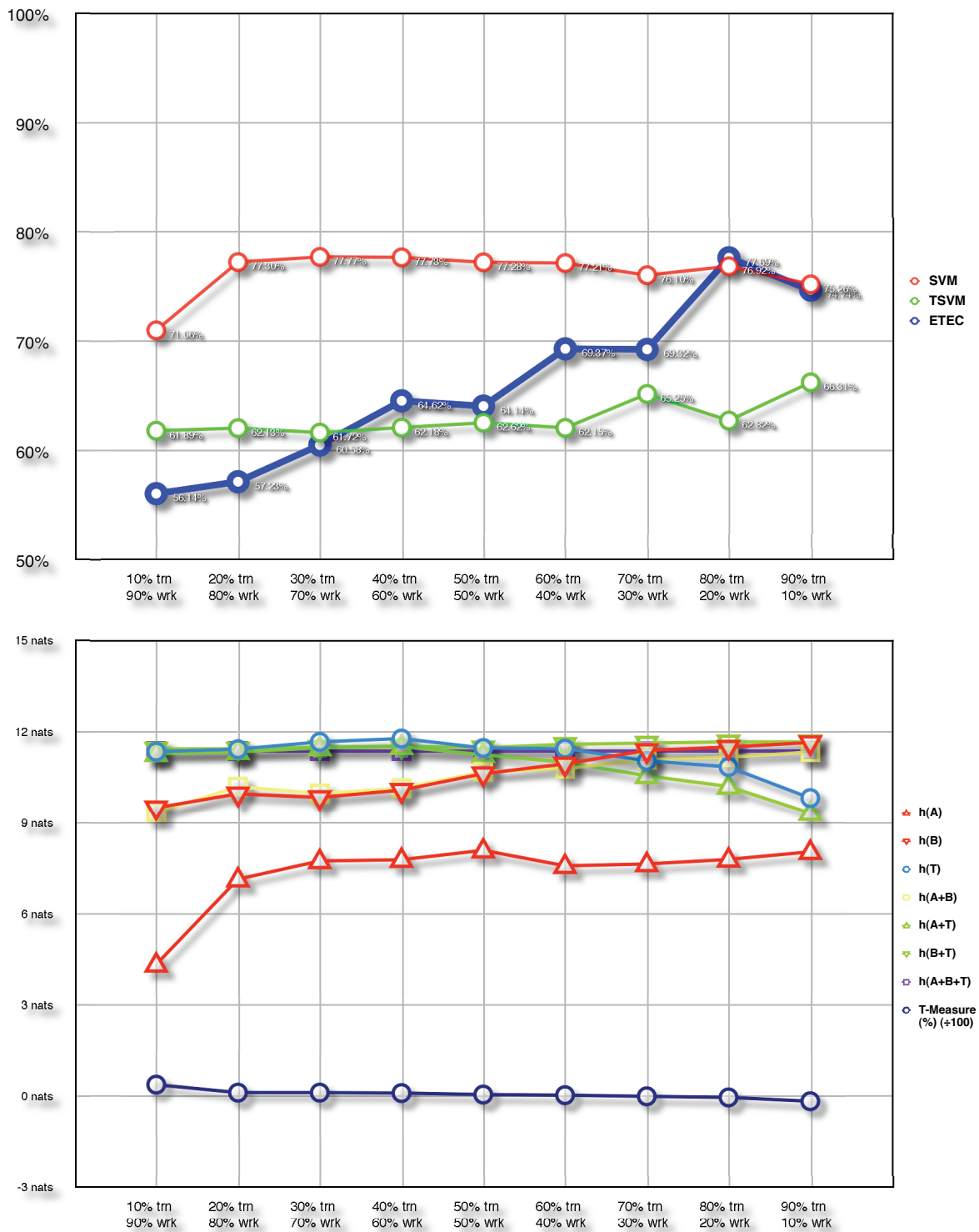


Figure 12.14: Graphical results and indicators for **wpbc**, unbiased sampling.

12.4 Discussion of Results

After selecting 7 datasets with 2 sampling variations for each and showing the proposed information indicators in Chapter 10, after describing benchmark results for two commonly-used methods from the literature in Chapter 11, after showing the results for the proposed ETEC method in Chapter 12, and after presenting all of these results in graphical fashion in Section 12.3, we finally analyze and comment on all results obtained in this work.

Over the next sections, we examine the results for the 7 datasets, where each set led to two experiments, referred to as setups in the analyses, one with each of the 2 sampling strategies. Most analyses are made with direct references to the charts in Section 12.3, which summarize all results. When needed, we refer to the tables previously presented to inspect numerical values in detail.

12.4.1 Analysis for bcw

We start our analysis of the experiments with the **bcw** dataset by examining the performance of the methods tested. We notice that, for both biased and unbiased samplings, the performance of ETEC was superior to that of SVM and TSVM, except in the two rightmost cases, the ones with the most samples in the training set. Interestingly, ETEC's performance was not stable as the training set increased. Instead, it declined when the training set exceeded 80% of the samples. If we take a deeper look at this decline, though, we find that in these two last configurations, the method had to classify the 139 and 69 samples in the working set, and failed to correctly classify an average of 8 samples for both. So, although the average rate of correctly classified samples dropped, it only dropped because the working set got smaller, and a few outstanding samples refused to

be properly estimated. Notice that both SVM and TSVM did not behave like this. As the training set increased, so did their performance, with the exception of TSVM in the unbiased setup, which hit trouble and had a performance lower than that of ETEC.

In both setups with the **bcw** dataset, we see that the overall accuracy of all methods was reasonably high, in most cases above 90%. Both SVM and TSVM showed classical inductive behavior, where the accuracy increased almost-monotonically as the training set size increased. Although this is expected for the SVM, it shows that TSVM did not take advantage of the use of unlabeled data. In fact, TSVM exhibited poor behavior for the **bcw** dataset, being outperformed by the standard SVM in all cases, and by ETEC in all but two cases, one in each sampling setup.

Now, let us examine the indicators for these experiments. First, the two first-order indicators $h(A)$ and $h(B)$ smoothly increased as the training set got bigger. We may interpret this smoothness as a result of the data of the problem being evenly distributed in its space. This hypothesis is reinforced by the behavior of $h(T)$, which decreases also very smoothly, and the behavior of $h(A + B + T)$, which is nearly the same as $h(A)$ and $h(T)$. Notice that in all cases, $h(A)$ is substantially larger than $h(B)$, which suggests that the samples in A carry more information than those in B . This may be due to the fact that there are more samples in A than in B , that the samples in A are more dispersed than those in B , or both.

In both sampling setups, the behavior of first-order indicators was approximately the same. One second-order indicator, $h(A + B)$, did catch the bias, though. In the unbiased setup, $h(A + B)$ was nearly the same as $h(A)$, which suggests that B was contained in A , as the union of the sets added little additional information to B . In the biased setup, $h(A + B)$ was slightly smaller than $h(A)$ in the three leftmost cases, suggesting that in these cases with small training sets, there really was a bias towards a given region of the

space where samples were more densely packed, hence yielding a smaller entropy.

Last, we look at the t-measure, which clearly shows the distinction between the biased and unbiased setups. In the unbiased setup, the t-measure is almost flat and near zero, suggesting that the performance of ETEC would not be favored. In fact, this is exactly what is observed, as ETEC only slightly outperforms the inductive SVM in all but 2 cases. In the biased setup, the t-measure shows a bold increase at the beginning, suggesting that ETEC would perform well. Again, this is what is observed, as ETEC performs well in these cases, while both SVM and TSVM perform significantly worse.

12.4.2 Analysis for hepatitis

In both sampling setups for the **hepatitis** dataset, we notice right away that all first-order and second-order indicators are practically flat. If we inspect Tables A.5, A.6, A.7, and A.8, we confirm that these values not only had little variation between cases with different set sizes, but they also had small standard variations. This suggests that not only are the data within each set well distributed, there is also an even distribution between classes A and B . This scenario is not favorable for transductive methods, as there is little information gain from the unlabeled data. This is confirmed by the t-measure, which lies almost flat near zero in both sampling setups.

As expected, the performance of ETEC suffered in this problem. We notice that in both setups, it decreased at a near-constant rate, with the exception of a single bump in the unbiased setup. This means that for all cases of set sizes, ETEC's performance decreased as the number of samples in the working set also decreased, that is, it failed to correctly classify a roughly-fixed number of samples. This behavior is justified by the fact that in an extremely well-distributed dataset, as the indicators point out, ETEC has practically the same information at all times, regardless if all samples have labels or not.

Again for this dataset, the inductive SVM performed consistently better than TSVM in all cases, where both methods reported a constant accuracy rate, as expected from the even distributions.

12.4.3 Analysis for ionosphere

The results for the **ionosphere** dataset show great correlation between the performance of ETEC and the t-measure. In the biased setup, ETEC significantly outperformed both SVM and TSVM. This is consistent with the t-measure, which was quite high at the beginning, decreasing towards the end but nonetheless always keeping above zero. Notice that, although the performance was still good, the accuracy rate did decrease as the working set also decreased, a phenomenon similar to that observed for other datasets, where a nearly-fixed number of samples are consistently classified incorrectly.

In the unbiased setup, the t-measure kept above zero in the first 5 cases, but became negative in the last 4. As a result, we observe how the performance of ETEC dropped in these cases, far more than the usual drop observed and just commented.

Once again, the performance of SVM and TSVM slightly increased as the working set size also increased, while SVM consistently outperformed TSVM in both setups.

The indicators for the unbiased setup show a very neat figure. Indicators $h(A + B)$ and $h(T)$ remained constant, approximately equal to $h(A + B + T)$. $h(A)$ and $h(B)$ also remained constant, while $h(B)$ was significantly larger than $h(A)$. Meanwhile, $h(A + T)$ and $h(B + T)$ converged smoothly from $h(T)$ to $h(A)$ and $h(B)$. This suggests that the data in the problem are very well distributed, although there is a discrepancy between classes A and B . As no variation of $h(A + B)$ suggests that A and B were differently dispersed, it is likely that there are simply more samples in B than in A .

Indicators for the biased setup were very similar to those for the unbiased setup,

especially as the training set got smaller. In the leftmost cases, the indicators once again were able to identify the bias.

12.4.4 Analysis for musk

All first-order and second-order indicators for both setups of the **musk** dataset are practically constant. Interestingly, the amount of information contained in the set is far more than in other datasets, with indicators ranging around 900 nats. One reason for such is the fact that samples in this dataset have 166 dimensions, therefore increasing the entropy of its sets. As expected, the t-measure in both setups, shown multiplied by 100, is also nearly flat and close to zero.

As a result of these indicators, it would be expected that the ETEC method would perform poorly for this problem. Instead, it consistently outperforms both SVM and TSVM, where the SVM again outperformed TSVM.

The reason for such may not be explained by the good performance shown by ETEC. Instead, it is likely that both SVM and TSVM were not properly tuned. It may be the case that the tuning procedure used was not capable of reaching a reasonable configuration, especially for a problem with 166 dimensions. As ETEC is parameter-free, it did not suffer from a similar problem.

12.4.5 Analysis for pgs

As seen in other datasets, all first-order and second-order indicators for both setups of the **pgs** dataset are practically constant. Also, the t-measure, shown again multiplied by 100, is almost flat and around zero.

This time, the SVM and TSVM were apparently well tuned. As expected, both SVM and TSVM consistently outperformed ETEC in the two sampling setups, with minor dif-

ferences between the performance of the SVM and TSVM.

12.4.6 Analysis for pid

The analysis for the **pid** dataset is the same as the one for **pgs**. Flat indicators confirm that both SVM and TSVM outperform ETEC, with minor differences between these two.

12.4.7 Analysis for wpbc

If we look at the t-measure for both setups of the **wpbc** dataset, we notice a phenomenon not observed in any other dataset, as it presented a great variation between the cases with different proportions between training and working data. Although this may not be clear by looking at Figures 12.13 and 12.14, it is so much so that t-measure's scale had to be reduced, and it is shown divided by 100. If we resort to Tables A.25, A.26, A.27, and A.28, we not only confirm this variation, but also observe how the standard variation of the t-measure is high.

In this odd scenario, instead of relying on the t-measure to indicate how ETEC will perform, we first identify why it has varied so much. The reason lies in the indicator $h(A + B)$, which assumed values both much higher and much lower than $h(T)$, hence producing an extremely volatile t-measure. Also, we notice that $h(B)$ is significantly higher than $h(A)$, and that $h(A + B)$ is very similar to $h(B)$. This suggests a great unbalance between classes A and B , not only in the number of samples, but also in their dispersion, as $h(A + T)$ fails to converge to $h(A)$, and as $h(B + T)$ tags along $h(T)$ and gradually converges to $h(B)$. In fact, if we look in Section 10.1.7, we see that this dataset is composed of a total 198 samples, 47 in A and 151 in B , which confirms the indicators. Also, we have that $h(A + T)$ is nearly the same as $h(T)$ and that the value of $h(A)$ is very low (reaching near zero in the biased setup), which suggests that the samples in A are

more densely packed than those in B .

As a result of this unbalanced dataset, ETEC did not perform well, especially in the unbiased setup. In the biased setup, it was still better than TSVM, but in 6 out of 9 cases it was worse than SVM. Interestingly, the usual slight performance increases obtained with ETEC in biased setups with other datasets was much higher in this case with the **wpsc** dataset.

Part V

Conclusion

Chapter 13

Conclusion

In this chapter we draw our final conclusions for this work. There are three particular subjects which we address, namely the theory, the method proposed, and the experimental results obtained.

We start with the theory. In this work, we have analyzed the classical statistical learning theory for the transductive problem. We have identified specific issues that kept this theory from being used as the basis of learning methods, similarly to what we have with inductive learning theory. In accordance with other authors, we believe that the transductive learning problem lacks a comprehensive and generally acknowledged theory, like those available for other types of inference. As the study of the transductive problem evolves in the machine learning community, we expect that a new, detailed theory may eventually be fully developed.

In this work we have taken a few steps in that direction. To overcome some of the difficulties present in the classical theory, we have proposed variations that later were used as the foundation to construct a new transductive method. Given the success of the implementation and results of this method, we believe that the concept of direct estimation of risks, possible with these proposed variations, may be further used on

the construction of other methods, which may, in turn, give new contributions to the field. Additionally, we also believe that this new theory, only very generally proposed in this work, may be further developed in many senses, for instance on the analysis of consistency of transductive learning, and on its convergence and comparison to inductive learning.

Perhaps, one of the greatest obstacles that has kept transductive learning from being more extensively explored is the amount of well-known theories and efficient methods available for inductive learning. One of the symptoms of this bias may be observed in some papers found in the literature, where, as we described, not even the concept of transduction is fully acknowledged. The paradigm change is indeed outstanding. Even important general concepts such as generalization stop making sense with transductive inference. Part of this legacy from inductive learning must be replaced by new concepts that fully describe a transductive process, effectively embracing the idea of not inducing a learning rule and later using it to deduce answers.

As for the method proposed, we acknowledge that one of its greatest strengths is its strong theoretical foundation. Using a theory variation as a basis, we have taken a series of steps to build the method, each one backed by solid principles from statistics and information theory. As a result, we reached our essentially transductive learning method which performed well on several binary classification problems right from the start. This fact, combined with its underlying foundations, leads us to think that the proposed method is only in its preliminary stages, and that further development has the potential to greatly increase its performance and applicability, two things we strongly believe.

Along with the method itself, our propositions of information indicators may also be further developed. In our discussion of the experimental results, we noticed how

well they described some properties from learning problems that largely influenced the performance of the method proposed. Analogously, these indicators have the potential to become generally acknowledged measures for learning algorithms, much like we have in other fields such as the F-measure from information retrieval.

Finally, there are many interesting observations we may do with respect to the experimental results obtained. The first one is that not only did the method proposed perform well on the datasets selected, the indicators proposed also identified the contexts which were favorable to our method as well as those where we did not expect any gain over inductive methods. Still, the same remark made about the potential of the method also applies to these results, where we strongly believe that further development is very likely to yield even better figures for these same experiments.

Also, some suspicions that we had were empirically confirmed, for instance that other inductive methods struggle when the i.i.d. assumption over the training and working data is broken. Meanwhile, our transductive method not only performs better than inductive ones in these scenarios, but it may even be favored by some biased environments. In addition, our information indicators were very successful in detecting bias and imbalance in the learning problems we experimented with.

As our last observation, we empirically witnessed how the problem of parameter tuning may affect the performance of a learning method. Although we submitted our benchmark methods to a parameter tuning procedure, reaching the right set of parameters that draw the ultimate performance from each method is quite a challenge. While we wondered about the effects of poor parameters on the performance of benchmark methods, we also observed our proposed parameter-free method excel with no further hassle.

Based on these conclusions and the results obtained in this work, in Chapter 14 we

propose several follow-up topics for new research that may continue the development of the ideas introduced here.

Chapter 14

Future Work

The results and conclusions of this work give rise to a series of issues that suggest the further deepening of the research conducted so far. The following sections describe some of the topics that may be further investigated in the near future.

14.1 Performance Improvement

The first and foremost issue that may be addressed following this work is the overall improvement of the performance of the methods proposed. This enhancement may come in several flavors.

First, one may work on the improvement of the accuracy of the ETEC method. As the method is well founded, there are many attempts that could be tried, without changing its structure, that could still result in significant enhancements in its classification performance. For instance, the data may be further preprocessed, possibly adding new dimensions with additional information, as is already done, or even normalizing data in a way to favor the separability of the classes in the weight space. Furthermore, given the independence assumption, one may perform a principal component analysis on the data

to ensure that the lack of covariance terms does not hide potential information in the dataset.

Additionally, one could also use a different objective function for the optimization problem. For instance, we have used the function $f(\alpha) = -D_{KL}(S\|\bar{S})$ to define the ETEC method. We have, however, experimented with the function $f(\alpha) = -(D_{KL}(S\|\bar{S}) + D_{KL}(\bar{S}\|S))$, which used the symmetrized version of the Kullback-Leibler divergence. Although this specific attempt did not yield better results, similar functions equally based on the Minimum Discrimination Information principle may pay off, even other parametrized expressions for the symmetric Kullback-Leibler divergence.

Together with the classification performance of the method, one may further develop the information indicators proposed. All indicators have been proposed based on simple measures there were implicitly calculated by the method, where no additional effort was made to collect specific measures from the probability distributions. For example, not even the mean of the distributions is considered by the indicators. If one was to extend the same concept of information indicators proposed in this work, it is possible that new, more precise, indicators would be found, therefore enhancing the prediction of applicability of the transductive method.

14.2 Consistency of Transductive Learning Processes

Classical inductive learning theory addresses issues such as the consistency of learning processes, their rate of convergence, and methods to control this rate of convergence. For the transductive learning problem, though, the theory must be revised. Vapnik analyzes the uniform relative deviation of frequencies for transductive problems in [Vap98], but he does not define the necessary and sufficient conditions for consistency of a transduc-

tive learning process.

Until an alternative theory is consolidated, several open questions remain unanswered, for instance:

- Is there a relationship between inductive learning and transductive learning?
- Can one determine an asymptotic theory on the number of training samples, similarly to the one for inductive learning? If so, do the bounds of the two theories eventually connect?
- Can one determine an asymptotic theory on the information contained in the problem, as opposed to the traditional sample-based theories?
- What is the definition of consistency for a transductive learning process?
- Is there, and if yes, what is the definition of rate of convergence for a transductive learning process?
- How does one handle the bias-variance dilemma in a transductive problem? What is the transductive extension to the concept of generalization?

14.3 Indeterminacy of the Separating Margin

In Section 6.4 we raised the question of whether a separating margin between classes does exist in a transductive problem, and if it could be determined using an essentially-transductive method. As we only introduced the subject qualitatively, one may extend this theoretical investigation and attempt to prove if such phenomenon of indeterminacy of the margin does exist. Furthermore, one could make quantitative analyses of the phenomenon, perhaps assessing the degree of indeterminacy of the margin, as suggested

in Equation (6.12). The study of this indeterminacy may determine if it has any impact on the quality of solutions of the learning problem. If confirmed, this result would reflect on the learning bounds of transductive learning theory.

14.4 Multi-class Classification

A natural extension of the standard binary classification problems we explored in this work would be the replacement of the output domain $\{-1, +1\}$ by a finite set of n labels $\{1, \dots, n\}$. In a transductive setting, multi-class classification means that the search space sought by the training algorithm is much larger. For instance, if we see the space of a binary classification problem as a binary tree with 2^k leaf nodes, the space of a multi-class classification problem shall have n^k leaf nodes.

There are many inductive support vector machines techniques for transforming a multi-class problem into a set of binary problems. One such example is the *one versus all* (OVA) pairwise comparison procedure. There are two ways to extend the method proposed to multi-class problems. First, one could change the method itself to be able to output more than two labels. This would involve changing the search space of weights sought by the optimization algorithm, and as the case with traditional support vector machines, such change would deeply impact the performance of the method and may not be worthwhile. Second, one could use a decomposition technique such as *one versus all*, keeping the original method and building a meta-algorithm to feed and control it. This technique is widely adopted in the literature, including applications to extend SVMs to multi-class problems [RTR⁺01].

14.5 Multi-label Classification

Another extension that follows the multi-class classification described in Section 14.4 is multi-label classification. Given a problem with a finite output set with n labels $\{1, \dots, n\}$, multi-label classification would assign confidence levels to each of the n labels when presented a new data sample.

As with multi-class classification, support vector machines also have extensions to cope with confidence measures. One of the most straight forward confidence measures for SVMs is the Euclidean distance of a sample point to the separating hyperplane. A similar approach has been explored in [RTR⁺01] for choosing between results of machines trained in *one versus all* pairwise comparisons.

The extension to yield confidence levels for each of the n labels may be as hard to implement in the method proposed as its multi-class extension, as described in Section 14.4. For $n = 2$, though, as the method is based on searches in a weight space, the value of the weights at the solution point may be used to calculate confidence levels.

14.6 Regression Problems

Aside from exploring different input spaces, we may also tackle problems with different output spaces other than the discrete ones we have so far described. One of the most interesting applications of support vector machines is on regression problems, that is, problems with output space in $[-\infty, +\infty]$ [CST00]. Many recent advances have also widened the applicability range of SVMs in regression problems, such as the work of Flake and Lawrence [FL01] to build a new version of SMO [Pla98a] for real-value output problems. Due to its structural risk minimization principle, SVMs are often found to outperform other regression machines.

Notice that in a transductive setting, similarly to the effect described in Section 14.4, the search space will theoretically be infinite. If we consider that most digital computer architectures deal with numbers in $\mathbb{R}$ with discrete approximations of m possible finite states, we can, in abstract terms, consider the resulting search space as having m^n leaf nodes. Regardless of the abstraction and of the size of m , which is quite large for modern floating-point and fixed-point technologies, any transductive algorithm dealing with this problem will heavily depend on its search heuristics to efficiently traverse the search space.

14.7 Sparseness: Support Vector Entropy Concept

Traditional inductive support vector machines are built around a constrained quadratic optimization problem solved with the introduction of Lagrange multipliers $\alpha_i \geq 0$. At the saddle point, one of the conditions that must hold is

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0, \quad (14.1)$$

Therefore, the solution vector given by

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \quad (14.2)$$

is made of the expansion of training samples whose associated Lagrange multipliers are non-zero, the so-called support-vectors.

With the introduction of soft separating hyperplanes controlled by a regularization parameter C , we have the familiar inequality

$$0 \leq \alpha_i \leq C, \quad (14.3)$$

where larger values of C allow more Lagrange multipliers to be non-zero, thus increasing the number of support vectors as well as the flexibility of the margin.

Concurrently, the regularization parameter controls the amount of information contained in the margin, as a direct result of how many support vectors are required for its calculation. Notice that by providing a mechanism to regulate how many training samples are considered, support vectors enable sparse solutions to be computed efficiently, regardless of the size of the training set.

A similar approach may be used in our entropic machine, enabling it to, first, control the amount of information considered using a regularization parameter, and second, compute (hopefully more efficient) sparse solutions.

In the traditional SVM, the regularization parameter controls if a given sample is a support vector (and thus part of the solution) based on its geometric location. If the sample is close enough to the margin (where how close is up to the parameter), it is considered a support vector. Analogously, a sample far from the margin would likely to be discarded, as it would not be of any value to the solution.

In our entropic world, a regularization parameter would also be used to control the amount of information, except that there are no geometric margins. Instead, a sample should be elected to be a support vector or not based on the amount of information that it adds to the set of support vectors, or some other set of vectors in the same space. For instance, suppose we have a set of samples X , where $x_1 \in X$. If we add another sample x_2 to X , where x_2 lies close together to x_1 in the space, we would not add much information to the set X . On the other hand, if we add a sample x_3 to X where x_3 is far away from any other sample in X , this would substantially increase in the information contained in X . The amount of information to be considered or discarded is exactly what the regularization parameter would control.

There are many entropic measures that are well suited to determine a criteria for support vector selection, for instance mutual information or the Kullback-Leibler divergence.

The entropic constraint analogous to Equation (14.3) could be written

$$0 \leq D_{KL}(X \| X') \leq C, \quad (14.4)$$

or

$$0 \leq I(X; X') \leq C, \quad (14.5)$$

where X' is the extended distribution of the set containing the sample being evaluated, similarly to the mechanism described in Section 7.1.

Finally, notice that the regularization parameter would directly affect the entropic measures of each set, as it would control the total amount of information considered in each case. Hopefully, this control would enable variations in the ratio amongst the measures used to solve the transduction problem, resulting an effect similar to that found in the standard SVM. In essence, this would be yet another approach to the bias-variance dilemma.

Part VI

Appendices

Appendix A

Information Indicators

The next sections show the indicators introduced in Chapter 9, evaluated for all combinations of sampling strategy and set sizes, where the data used in each combination are the same data used in all experiments in Chapters 11 and 12.

A.1 Indicators for **bcw**

Primary indicators for the **bcw** dataset drawn with biased sampling are shown in Table A.1. Transductivity and compound indicators are shown in Table A.2.

Table A.1: Primary indicators for **bcw**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	20.79 ± 0.62	11.34 ± 1.82	21.83 ± 0.13	18.88 ± 1.38
20%	80%	21.54 ± 0.41	11.54 ± 1.71	21.84 ± 0.15	20.79 ± 0.72
30%	70%	21.69 ± 0.31	12.44 ± 1.04	21.84 ± 0.15	21.22 ± 0.45
40%	60%	21.66 ± 0.19	12.55 ± 0.97	21.84 ± 0.17	21.41 ± 0.30
50%	50%	21.70 ± 0.14	12.56 ± 0.78	21.81 ± 0.25	21.53 ± 0.26
60%	40%	21.70 ± 0.13	12.57 ± 0.63	21.88 ± 0.29	21.53 ± 0.23
70%	30%	21.73 ± 0.07	12.57 ± 0.45	21.90 ± 0.26	21.58 ± 0.14
80%	20%	21.74 ± 0.05	12.64 ± 0.29	21.93 ± 0.26	21.62 ± 0.09
90%	10%	21.76 ± 0.05	12.73 ± 0.20	22.05 ± 0.50	21.64 ± 0.07

Table A.2: Compound and transductivity indicators for **bcw**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	21.89 ± 0.04	21.62 ± 0.09	21.69 ± 0.00	$14.41\% \pm 9.77\%$
20%	80%	22.07 ± 0.05	21.41 ± 0.11	21.69 ± 0.00	$4.08\% \pm 4.05\%$
30%	70%	22.24 ± 0.03	21.15 ± 0.13	21.69 ± 0.00	$2.55\% \pm 2.99\%$
40%	60%	22.41 ± 0.03	20.83 ± 0.15	21.69 ± 0.00	$1.77\% \pm 2.29\%$
50%	50%	22.57 ± 0.03	20.39 ± 0.23	21.69 ± 0.00	$1.08\% \pm 2.37\%$
60%	40%	22.71 ± 0.02	19.92 ± 0.27	21.69 ± 0.00	$1.37\% \pm 2.43\%$
70%	30%	22.81 ± 0.03	19.21 ± 0.25	21.69 ± 0.00	$1.21\% \pm 1.83\%$
80%	20%	22.81 ± 0.03	18.19 ± 0.24	21.69 ± 0.00	$1.31\% \pm 1.67\%$
90%	10%	22.58 ± 0.06	16.61 ± 0.37	21.69 ± 0.00	$2.17\% \pm 2.16\%$

Primary indicators for the **bcw** dataset drawn with unbiased sampling are shown in Table A.3. Transductivity and compound indicators are shown in Table A.4.

Table A.3: Primary indicators for **bcw**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	21.62 ± 0.48	10.85 ± 2.11	21.69 ± 0.07	21.64 ± 0.67
20%	80%	21.68 ± 0.41	11.33 ± 1.30	21.69 ± 0.12	21.67 ± 0.47
30%	70%	21.77 ± 0.28	11.92 ± 1.14	21.68 ± 0.14	21.69 ± 0.34
40%	60%	21.77 ± 0.15	12.30 ± 0.80	21.67 ± 0.16	21.70 ± 0.26
50%	50%	21.81 ± 0.08	12.66 ± 0.61	21.68 ± 0.25	21.68 ± 0.27
60%	40%	21.83 ± 0.07	12.72 ± 0.33	21.72 ± 0.21	21.66 ± 0.16
70%	30%	21.81 ± 0.09	12.66 ± 0.20	21.73 ± 0.24	21.67 ± 0.10
80%	20%	21.81 ± 0.08	12.71 ± 0.18	21.75 ± 0.27	21.68 ± 0.07
90%	10%	21.80 ± 0.03	12.68 ± 0.18	21.73 ± 0.45	21.68 ± 0.05

Table A.4: Compound and transductivity indicators for **bcw**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	21.86 ± 0.02	21.51 ± 0.06	21.69 ± 0.00	$0.45\% \pm 3.40\%$
20%	80%	22.03 ± 0.02	21.29 ± 0.11	21.69 ± 0.00	$0.52\% \pm 2.37\%$
30%	70%	22.21 ± 0.03	21.02 ± 0.13	21.69 ± 0.00	$0.32\% \pm 1.92\%$
40%	60%	22.38 ± 0.03	20.68 ± 0.16	21.69 ± 0.00	$0.23\% \pm 1.58\%$
50%	50%	22.55 ± 0.03	20.27 ± 0.23	21.69 ± 0.00	$0.35\% \pm 2.09\%$
60%	40%	22.70 ± 0.03	19.78 ± 0.20	21.69 ± 0.00	$0.47\% \pm 1.48\%$
70%	30%	22.80 ± 0.02	19.05 ± 0.22	21.69 ± 0.00	$0.49\% \pm 1.33\%$
80%	20%	22.82 ± 0.02	18.03 ± 0.24	21.69 ± 0.00	$0.27\% \pm 1.56\%$
90%	10%	22.60 ± 0.04	16.36 ± 0.32	21.69 ± 0.00	$-0.16\% \pm 1.93\%$

A.2 Indicators for hepatitis

Primary indicators for the **hepatitis** dataset drawn with biased sampling are shown in Table A.5. Transductivity and compound indicators are shown in Table A.6.

Table A.5: Primary indicators for **hepatitis**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	37.77 ± 1.34	39.26 ± 2.50	39.68 ± 0.09	38.53 ± 1.20
20%	80%	37.22 ± 1.01	39.90 ± 1.96	39.73 ± 0.11	38.67 ± 0.55
30%	70%	37.31 ± 0.99	40.31 ± 1.19	39.68 ± 0.23	39.11 ± 0.66
40%	60%	37.67 ± 0.70	40.30 ± 0.90	39.70 ± 0.25	39.23 ± 0.51
50%	50%	37.96 ± 0.40	40.14 ± 0.70	39.60 ± 0.34	39.44 ± 0.37
60%	40%	38.05 ± 0.44	40.21 ± 0.53	39.44 ± 0.51	39.51 ± 0.38
70%	30%	38.22 ± 0.60	39.97 ± 0.32	39.24 ± 0.89	39.59 ± 0.42
80%	20%	38.32 ± 0.31	39.96 ± 0.24	39.17 ± 0.91	39.65 ± 0.19
90%	10%	38.31 ± 0.24	40.10 ± 0.23	39.35 ± 1.32	39.65 ± 0.14

Table A.6: Compound and transductivity indicators for **hepatitis**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	39.60 ± 0.04	39.71 ± 0.08	39.63 ± 0.00	$2.77\% \pm 3.61\%$
20%	80%	39.54 ± 0.04	39.81 ± 0.12	39.63 ± 0.00	$2.55\% \pm 1.92\%$
30%	70%	39.40 ± 0.11	39.88 ± 0.17	39.63 ± 0.00	$1.47\% \pm 2.31\%$
40%	60%	39.32 ± 0.11	39.96 ± 0.15	39.63 ± 0.00	$1.14\% \pm 2.00\%$
50%	50%	39.20 ± 0.14	39.96 ± 0.14	39.63 ± 0.00	$0.45\% \pm 1.77\%$
60%	40%	39.07 ± 0.18	39.90 ± 0.31	39.63 ± 0.00	$-0.12\% \pm 2.17\%$
70%	30%	38.89 ± 0.22	40.00 ± 0.34	39.63 ± 0.00	$-0.53\% \pm 3.10\%$
80%	20%	38.72 ± 0.16	39.93 ± 0.38	39.63 ± 0.00	$-0.94\% \pm 2.69\%$
90%	10%	38.50 ± 0.14	39.86 ± 0.62	39.63 ± 0.00	$-0.04\% \pm 2.84\%$

Primary indicators for the **hepatitis** dataset drawn with unbiased sampling are shown in Table A.7. Transductivity and compound indicators are shown in Table A.8.

Table A.7: Primary indicators for **hepatitis**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	38.38 ± 0.90	40.96 ± 1.92	39.60 ± 0.11	39.34 ± 0.81
20%	80%	37.52 ± 0.90	40.82 ± 1.16	39.61 ± 0.12	39.15 ± 0.75
30%	70%	37.75 ± 0.80	40.76 ± 0.89	39.62 ± 0.18	39.34 ± 0.45
40%	60%	37.83 ± 0.80	40.41 ± 0.96	39.60 ± 0.26	39.38 ± 0.43
50%	50%	37.96 ± 0.55	40.25 ± 0.68	39.57 ± 0.31	39.45 ± 0.34
60%	40%	38.07 ± 0.47	40.19 ± 0.68	39.56 ± 0.45	39.47 ± 0.36
70%	30%	38.17 ± 0.39	40.14 ± 0.45	39.32 ± 0.55	39.61 ± 0.27
80%	20%	38.20 ± 0.35	39.88 ± 0.51	39.02 ± 0.72	39.63 ± 0.20
90%	10%	38.25 ± 0.25	39.86 ± 0.27	38.87 ± 1.51	39.63 ± 0.16

Table A.8: Compound and transductivity indicators for **hepatitis**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	39.54 ± 0.07	39.70 ± 0.08	39.63 ± 0.00	$0.64\% \pm 2.28\%$
20%	80%	39.48 ± 0.08	39.76 ± 0.11	39.63 ± 0.00	$1.28\% \pm 2.15\%$
30%	70%	39.39 ± 0.10	39.85 ± 0.13	39.63 ± 0.00	$0.86\% \pm 1.44\%$
40%	60%	39.31 ± 0.10	39.89 ± 0.17	39.63 ± 0.00	$0.69\% \pm 1.60\%$
50%	50%	39.19 ± 0.12	39.96 ± 0.20	39.63 ± 0.00	$0.40\% \pm 1.55\%$
60%	40%	39.05 ± 0.15	40.08 ± 0.26	39.63 ± 0.00	$0.31\% \pm 2.01\%$
70%	30%	38.86 ± 0.11	39.96 ± 0.36	39.63 ± 0.00	$-0.45\% \pm 1.91\%$
80%	20%	38.63 ± 0.15	39.93 ± 0.36	39.63 ± 0.00	$-1.26\% \pm 2.31\%$
90%	10%	38.45 ± 0.14	39.68 ± 0.42	39.63 ± 0.00	$-1.28\% \pm 3.86\%$

A.3 Indicators for ionosphere

Primary indicators for the **ionosphere** dataset drawn with biased sampling are shown in Table A.9. Transductivity and compound indicators are shown in Table A.10.

Table A.9: Primary indicators for **ionosphere**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	15.88 ± 4.50	25.78 ± 6.68	27.20 ± 0.26	20.54 ± 3.15
20%	80%	18.47 ± 2.13	32.47 ± 2.59	27.19 ± 0.36	24.46 ± 1.90
30%	70%	19.17 ± 2.17	32.70 ± 1.70	27.22 ± 0.62	25.31 ± 1.74
40%	60%	19.36 ± 1.52	32.57 ± 1.51	27.24 ± 0.87	25.81 ± 1.47
50%	50%	19.54 ± 1.41	32.82 ± 1.10	27.30 ± 1.14	26.02 ± 1.32
60%	40%	19.79 ± 0.74	32.94 ± 0.58	27.21 ± 1.02	26.36 ± 0.74
70%	30%	19.87 ± 0.50	33.10 ± 0.50	27.18 ± 1.09	26.51 ± 0.46
80%	20%	19.72 ± 0.47	33.27 ± 0.47	27.08 ± 1.14	26.62 ± 0.27
90%	10%	19.83 ± 0.18	33.34 ± 0.23	26.73 ± 1.62	26.71 ± 0.14

Table A.10: Compound and transductivity indicators for **ionosphere**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	26.64 ± 0.18	27.31 ± 0.21	26.76 ± 0.00	$34.81\% \pm 22.67\%$
20%	80%	26.16 ± 0.25	27.75 ± 0.21	26.76 ± 0.00	$11.21\% \pm 11.13\%$
30%	70%	25.71 ± 0.40	28.20 ± 0.30	26.76 ± 0.00	$7.77\% \pm 10.49\%$
40%	60%	25.13 ± 0.54	28.70 ± 0.30	26.76 ± 0.00	$5.92\% \pm 9.72\%$
50%	50%	24.50 ± 0.55	29.29 ± 0.39	26.76 ± 0.00	$5.28\% \pm 10.22\%$
60%	40%	23.66 ± 0.46	29.86 ± 0.33	26.76 ± 0.00	$3.35\% \pm 6.91\%$
70%	30%	22.62 ± 0.49	30.61 ± 0.32	26.76 ± 0.00	$2.72\% \pm 5.82\%$
80%	20%	21.14 ± 0.46	31.53 ± 0.29	26.76 ± 0.00	$1.21\% \pm 5.13\%$
90%	10%	19.22 ± 0.45	32.52 ± 0.19	26.76 ± 0.00	$0.66\% \pm 6.23\%$

Primary indicators for the **ionosphere** dataset drawn with unbiased sampling are shown in Table A.11. Transductivity and compound indicators are shown in Table A.12.

Table A.11: Primary indicators for **ionosphere**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	19.16 ± 3.15	33.66 ± 2.21	26.77 ± 0.30	26.15 ± 2.94
20%	80%	19.46 ± 1.94	33.17 ± 1.31	26.73 ± 0.38	26.51 ± 1.51
30%	70%	20.09 ± 1.32	32.95 ± 1.63	26.72 ± 0.60	26.61 ± 1.39
40%	60%	19.75 ± 0.95	32.98 ± 1.03	26.73 ± 0.45	26.67 ± 0.62
50%	50%	19.82 ± 0.62	32.79 ± 0.82	26.83 ± 0.38	26.60 ± 0.36
60%	40%	19.87 ± 0.51	33.20 ± 0.53	26.42 ± 0.73	26.89 ± 0.44
70%	30%	19.94 ± 0.39	33.06 ± 0.50	26.37 ± 0.75	26.84 ± 0.29
80%	20%	20.00 ± 0.27	33.19 ± 0.34	26.51 ± 0.73	26.77 ± 0.17
90%	10%	19.91 ± 0.24	33.33 ± 0.19	26.47 ± 1.12	26.75 ± 0.09

Table A.12: Compound and transductivity indicators for **ionosphere**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	26.36 ± 0.16	27.16 ± 0.17	26.76 ± 0.00	$3.95\% \pm 14.13\%$
20%	80%	25.92 ± 0.17	27.56 ± 0.24	26.76 ± 0.00	$0.68\% \pm 7.55\%$
30%	70%	25.46 ± 0.30	27.98 ± 0.26	26.76 ± 0.00	$1.30\% \pm 7.46\%$
40%	60%	24.85 ± 0.26	28.52 ± 0.24	26.76 ± 0.00	$0.81\% \pm 3.60\%$
50%	50%	24.26 ± 0.28	29.08 ± 0.20	26.76 ± 0.00	$1.50\% \pm 1.27\%$
60%	40%	23.20 ± 0.37	29.70 ± 0.22	26.76 ± 0.00	$-1.01\% \pm 3.95\%$
70%	30%	22.20 ± 0.39	30.41 ± 0.23	26.76 ± 0.00	$-1.30\% \pm 3.82\%$
80%	20%	21.09 ± 0.44	31.33 ± 0.16	26.76 ± 0.00	$-1.05\% \pm 3.29\%$
90%	10%	19.31 ± 0.31	32.45 ± 0.22	26.76 ± 0.00	$-0.71\% \pm 4.47\%$

A.4 Indicators for musk

Primary indicators for the **musk** dataset drawn with biased sampling are shown in Table A.13. Transductivity and compound indicators are shown in Table A.14.

Table A.13: Primary indicators for **musk**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	900.43 ± 29.74	945.29 ± 8.02	950.37 ± 0.93	936.93 ± 9.53
20%	80%	919.39 ± 9.91	953.26 ± 5.94	950.36 ± 1.36	945.47 ± 5.55
30%	70%	924.68 ± 5.04	955.53 ± 4.85	950.09 ± 1.61	948.09 ± 3.64
40%	60%	925.51 ± 4.87	957.19 ± 3.00	949.79 ± 1.38	949.17 ± 1.85
50%	50%	926.02 ± 3.71	957.24 ± 2.08	949.97 ± 1.65	949.23 ± 1.48
60%	40%	927.50 ± 2.60	956.95 ± 2.29	949.88 ± 2.52	949.30 ± 1.62
70%	30%	927.73 ± 2.41	957.42 ± 1.98	949.24 ± 2.45	949.52 ± 1.13
80%	20%	927.85 ± 2.20	957.77 ± 1.62	948.87 ± 4.48	949.59 ± 1.05
90%	10%	928.72 ± 1.35	957.91 ± 0.89	947.99 ± 5.36	949.83 ± 0.47

Table A.14: Compound and transductivity indicators for **musk**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	949.68 ± 0.53	950.82 ± 0.42	950.13 ± 0.00	$1.32\% \pm 1.22\%$
20%	80%	948.91 ± 0.80	951.49 ± 0.54	950.13 ± 0.00	$0.45\% \pm 0.75\%$
30%	70%	947.84 ± 1.13	952.12 ± 0.43	950.13 ± 0.00	$0.13\% \pm 0.52\%$
40%	60%	946.51 ± 0.85	952.89 ± 0.59	950.13 ± 0.00	$0.05\% \pm 0.33\%$
50%	50%	945.31 ± 0.78	953.72 ± 0.60	950.13 ± 0.00	$0.07\% \pm 0.33\%$
60%	40%	943.91 ± 1.42	954.44 ± 0.55	950.13 ± 0.00	$0.04\% \pm 0.43\%$
70%	30%	941.71 ± 1.18	955.36 ± 0.72	950.13 ± 0.00	$-0.07\% \pm 0.34\%$
80%	20%	938.97 ± 1.82	956.44 ± 0.76	950.13 ± 0.00	$-0.15\% \pm 0.50\%$
90%	10%	935.10 ± 1.42	957.37 ± 0.48	950.13 ± 0.00	$-0.29\% \pm 0.46\%$

Primary indicators for the **musk** dataset drawn with unbiased sampling are shown in Table A.15. Transductivity and compound indicators are shown in Table A.16.

Table A.15: Primary indicators for **musk**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A + B)$
10%	90%	914.10 ± 11.57	953.67 ± 11.98	949.54 ± 1.34	946.68 ± 9.76
20%	80%	920.85 ± 7.96	957.52 ± 5.11	949.48 ± 0.99	948.36 ± 4.72
30%	70%	922.54 ± 4.70	957.94 ± 4.35	949.47 ± 1.54	948.81 ± 3.80
40%	60%	926.30 ± 4.01	957.89 ± 3.58	949.08 ± 1.59	949.72 ± 2.44
50%	50%	927.35 ± 2.29	957.62 ± 2.76	949.39 ± 2.02	949.46 ± 1.97
60%	40%	928.27 ± 2.11	957.87 ± 2.51	949.16 ± 2.88	949.62 ± 1.89
70%	30%	928.17 ± 2.38	959.17 ± 1.40	946.87 ± 3.37	950.55 ± 1.20
80%	20%	928.39 ± 1.93	959.13 ± 1.25	945.03 ± 3.69	950.54 ± 0.79
90%	10%	928.31 ± 0.76	958.78 ± 0.44	942.33 ± 2.36	950.35 ± 0.27

Table A.16: Compound and transductivity indicators for **musk**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A + T)$	$h(B + T)$	$h(A + B + T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	948.93 ± 1.18	950.73 ± 0.34	950.13 ± 0.00	$0.40\% \pm 1.09\%$
20%	80%	948.11 ± 0.85	951.45 ± 0.49	950.13 ± 0.00	$0.16\% \pm 0.57\%$
30%	70%	947.19 ± 1.10	952.23 ± 0.52	950.13 ± 0.00	$0.12\% \pm 0.52\%$
40%	60%	946.09 ± 1.19	952.80 ± 0.36	950.13 ± 0.00	$-0.02\% \pm 0.39\%$
50%	50%	945.23 ± 1.31	953.60 ± 0.38	950.13 ± 0.00	$0.02\% \pm 0.40\%$
60%	40%	943.73 ± 1.55	954.44 ± 0.36	950.13 ± 0.00	$-0.02\% \pm 0.49\%$
70%	30%	940.52 ± 1.49	955.38 ± 0.53	950.13 ± 0.00	$-0.33\% \pm 0.48\%$
80%	20%	937.62 ± 1.43	956.33 ± 0.44	950.13 ± 0.00	$-0.55\% \pm 0.50\%$
90%	10%	933.98 ± 0.93	957.36 ± 0.18	950.13 ± 0.00	$-0.75\% \pm 0.38\%$

A.5 Indicators for pgs

Primary indicators for the **pgs** dataset drawn with biased sampling are shown in Table A.17. Transductivity and compound indicators are shown in Table A.18.

Table A.17: Primary indicators for **pgs**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	164.52 ± 6.46	167.78 ± 7.83	160.89 ± 0.07	159.52 ± 1.41
20%	80%	158.19 ± 1.14	160.70 ± 1.38	160.86 ± 0.14	160.32 ± 0.59
30%	70%	157.43 ± 1.02	161.08 ± 0.71	160.88 ± 0.11	160.66 ± 0.43
40%	60%	157.16 ± 0.65	161.27 ± 0.52	160.85 ± 0.23	160.68 ± 0.51
50%	50%	156.83 ± 0.65	161.52 ± 0.35	160.89 ± 0.25	160.80 ± 0.30
60%	40%	156.96 ± 0.54	161.65 ± 0.15	160.79 ± 0.43	160.94 ± 0.25
70%	30%	157.05 ± 0.63	161.62 ± 0.10	160.61 ± 0.55	160.93 ± 0.15
80%	20%	157.21 ± 0.50	161.69 ± 0.10	160.40 ± 0.69	160.91 ± 0.11
90%	10%	157.25 ± 0.31	161.72 ± 0.03	160.27 ± 1.17	160.92 ± 0.09

Table A.18: Compound and transductivity indicators for **pgs**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	160.83 ± 0.03	160.97 ± 0.04	160.90 ± 0.00	$0.79\% \pm 0.94\%$
20%	80%	160.70 ± 0.07	161.05 ± 0.08	160.90 ± 0.00	$0.37\% \pm 0.38\%$
30%	70%	160.55 ± 0.10	161.18 ± 0.06	160.90 ± 0.00	$0.13\% \pm 0.31\%$
40%	60%	160.36 ± 0.17	161.28 ± 0.10	160.90 ± 0.00	$0.12\% \pm 0.44\%$
50%	50%	160.11 ± 0.18	161.41 ± 0.11	160.90 ± 0.00	$0.04\% \pm 0.33\%$
60%	40%	159.72 ± 0.27	161.49 ± 0.11	160.90 ± 0.00	$-0.12\% \pm 0.40\%$
70%	30%	159.30 ± 0.23	161.55 ± 0.12	160.90 ± 0.00	$-0.21\% \pm 0.43\%$
80%	20%	158.86 ± 0.35	161.64 ± 0.08	160.90 ± 0.00	$-0.29\% \pm 0.48\%$
90%	10%	158.03 ± 0.35	161.69 ± 0.04	160.90 ± 0.00	$-0.31\% \pm 0.70\%$

Primary indicators for the **pgs** dataset drawn with unbiased sampling are shown in Table A.19. Transductivity and compound indicators are shown in Table A.20.

Table A.19: Primary indicators for **pgs**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A + B)$
10%	90%	166.15 ± 5.36	165.27 ± 4.89	160.87 ± 0.08	160.36 ± 1.25
20%	80%	159.10 ± 1.63	161.03 ± 1.38	160.84 ± 0.15	160.77 ± 0.38
30%	70%	157.69 ± 1.19	161.23 ± 0.76	160.79 ± 0.14	160.93 ± 0.36
40%	60%	157.39 ± 0.92	161.42 ± 0.42	160.72 ± 0.25	160.93 ± 0.38
50%	50%	157.23 ± 0.49	161.51 ± 0.27	160.71 ± 0.28	160.92 ± 0.33
60%	40%	157.15 ± 0.45	161.62 ± 0.16	160.66 ± 0.39	160.95 ± 0.28
70%	30%	156.87 ± 0.66	161.70 ± 0.16	160.61 ± 0.43	160.93 ± 0.14
80%	20%	156.92 ± 0.48	161.71 ± 0.09	160.42 ± 0.53	160.94 ± 0.09
90%	10%	157.21 ± 0.23	161.72 ± 0.03	160.76 ± 0.91	160.91 ± 0.07

Table A.20: Compound and transductivity indicators for **pgs**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A + T)$	$h(B + T)$	$h(A + B + T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	160.80 ± 0.03	160.97 ± 0.05	160.90 ± 0.00	$0.32\% \pm 0.83\%$
20%	80%	160.67 ± 0.08	161.06 ± 0.09	160.90 ± 0.00	$0.03\% \pm 0.31\%$
30%	70%	160.50 ± 0.10	161.14 ± 0.09	160.90 ± 0.00	$-0.09\% \pm 0.30\%$
40%	60%	160.28 ± 0.13	161.24 ± 0.13	160.90 ± 0.00	$-0.10\% \pm 0.38\%$
50%	50%	160.02 ± 0.20	161.36 ± 0.10	160.90 ± 0.00	$-0.11\% \pm 0.34\%$
60%	40%	159.68 ± 0.28	161.45 ± 0.09	160.90 ± 0.00	$-0.08\% \pm 0.29\%$
70%	30%	159.30 ± 0.20	161.56 ± 0.09	160.90 ± 0.00	$-0.11\% \pm 0.24\%$
80%	20%	158.69 ± 0.21	161.66 ± 0.04	160.90 ± 0.00	$-0.24\% \pm 0.33\%$
90%	10%	158.09 ± 0.23	161.73 ± 0.06	160.90 ± 0.00	$-0.04\% \pm 0.58\%$

A.6 Indicators for pid

Primary indicators for the **pid** dataset drawn with biased sampling are shown in Table A.21. Transductivity and compound indicators are shown in Table A.22.

Table A.21: Primary indicators for **pid**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	30.03 ± 0.47	28.87 ± 0.47	29.95 ± 0.04	29.68 ± 0.34
20%	80%	30.26 ± 0.45	29.07 ± 0.36	29.93 ± 0.07	29.88 ± 0.29
30%	70%	30.26 ± 0.32	29.03 ± 0.28	29.96 ± 0.06	29.85 ± 0.16
40%	60%	30.39 ± 0.28	29.03 ± 0.19	29.95 ± 0.07	29.88 ± 0.12
50%	50%	30.39 ± 0.24	29.09 ± 0.22	29.95 ± 0.11	29.90 ± 0.12
60%	40%	30.35 ± 0.18	29.09 ± 0.16	29.99 ± 0.13	29.88 ± 0.09
70%	30%	30.35 ± 0.14	29.17 ± 0.13	29.93 ± 0.20	29.92 ± 0.08
80%	20%	30.35 ± 0.10	29.18 ± 0.12	29.95 ± 0.30	29.92 ± 0.08
90%	10%	30.37 ± 0.06	29.18 ± 0.07	29.96 ± 0.38	29.92 ± 0.04

Table A.22: Compound and transductivity indicators for **pid**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	29.98 ± 0.02	29.91 ± 0.02	29.94 ± 0.00	$0.91\% \pm 1.29\%$
20%	80%	30.01 ± 0.04	29.86 ± 0.03	29.94 ± 0.00	$0.21\% \pm 1.20\%$
30%	70%	30.07 ± 0.05	29.81 ± 0.04	29.94 ± 0.00	$0.32\% \pm 0.75\%$
40%	60%	30.13 ± 0.05	29.74 ± 0.06	29.94 ± 0.00	$0.22\% \pm 0.62\%$
50%	50%	30.18 ± 0.08	29.68 ± 0.07	29.94 ± 0.00	$0.15\% \pm 0.78\%$
60%	40%	30.25 ± 0.07	29.62 ± 0.07	29.94 ± 0.00	$0.35\% \pm 0.74\%$
70%	30%	30.28 ± 0.09	29.54 ± 0.08	29.94 ± 0.00	$-0.07\% \pm 0.88\%$
80%	20%	30.34 ± 0.10	29.45 ± 0.06	29.94 ± 0.00	$-0.14\% \pm 0.91\%$
90%	10%	30.40 ± 0.07	29.34 ± 0.06	29.94 ± 0.00	$0.01\% \pm 1.35\%$

Primary indicators for the **pid** dataset drawn with unbiased sampling are shown in Table A.23. Transductivity and compound indicators are shown in Table A.24.

Table A.23: Primary indicators for **pid**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	30.21 ± 0.70	28.62 ± 0.42	29.95 ± 0.04	29.63 ± 0.42
20%	80%	30.32 ± 0.43	28.94 ± 0.24	29.94 ± 0.07	29.82 ± 0.30
30%	70%	30.39 ± 0.18	29.05 ± 0.18	29.94 ± 0.07	29.88 ± 0.17
40%	60%	30.39 ± 0.15	29.13 ± 0.18	29.94 ± 0.10	29.89 ± 0.15
50%	50%	30.40 ± 0.17	29.20 ± 0.19	29.91 ± 0.07	29.94 ± 0.07
60%	40%	30.45 ± 0.15	29.16 ± 0.15	29.90 ± 0.09	29.94 ± 0.06
70%	30%	30.45 ± 0.14	29.20 ± 0.08	29.86 ± 0.16	29.95 ± 0.07
80%	20%	30.43 ± 0.12	29.19 ± 0.10	29.87 ± 0.23	29.94 ± 0.05
90%	10%	30.40 ± 0.06	29.18 ± 0.06	29.90 ± 0.41	29.93 ± 0.04

Table A.24: Compound and transductivity indicators for **pid**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	29.99 ± 0.02	29.89 ± 0.02	29.94 ± 0.00	$1.02\% \pm 1.64\%$
20%	80%	30.03 ± 0.03	29.84 ± 0.04	29.94 ± 0.00	$0.36\% \pm 1.23\%$
30%	70%	30.07 ± 0.04	29.80 ± 0.03	29.94 ± 0.00	$0.14\% \pm 0.78\%$
40%	60%	30.12 ± 0.05	29.75 ± 0.04	29.94 ± 0.00	$0.11\% \pm 0.81\%$
50%	50%	30.15 ± 0.06	29.69 ± 0.03	29.94 ± 0.00	$-0.15\% \pm 0.43\%$
60%	40%	30.23 ± 0.06	29.60 ± 0.04	29.94 ± 0.00	$-0.13\% \pm 0.49\%$
70%	30%	30.28 ± 0.06	29.52 ± 0.07	29.94 ± 0.00	$-0.29\% \pm 0.75\%$
80%	20%	30.35 ± 0.08	29.43 ± 0.07	29.94 ± 0.00	$-0.20\% \pm 0.96\%$
90%	10%	30.40 ± 0.05	29.33 ± 0.05	29.94 ± 0.00	$0.06\% \pm 1.41\%$

A.7 Indicators for wpbc

Primary indicators for the **wpbc** dataset drawn with biased sampling are shown in Table A.25. Transductivity and compound indicators are shown in Table A.26.

Table A.25: Primary indicators for **wpbc**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A+B)$
10%	90%	0.55 ± 5.39	7.28 ± 4.65	11.51 ± 0.44	7.39 ± 3.94
20%	80%	3.71 ± 3.85	10.22 ± 2.62	11.54 ± 0.54	9.83 ± 2.09
30%	70%	6.09 ± 1.76	11.01 ± 2.65	11.35 ± 0.97	10.84 ± 2.23
40%	60%	7.05 ± 1.75	11.52 ± 1.53	11.13 ± 0.84	11.33 ± 1.20
50%	50%	7.05 ± 1.54	11.60 ± 1.21	11.04 ± 1.04	11.34 ± 1.07
60%	40%	7.91 ± 1.15	11.93 ± 0.80	10.47 ± 1.24	11.66 ± 0.71
70%	30%	8.20 ± 1.23	12.07 ± 0.54	9.78 ± 1.53	11.76 ± 0.60
80%	20%	7.89 ± 0.94	11.96 ± 0.59	9.49 ± 2.78	11.60 ± 0.60
90%	10%	7.85 ± 0.76	11.69 ± 0.38	9.69 ± 3.94	11.36 ± 0.39

Table A.26: Compound and transductivity indicators for **wpbc**, biased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A+T)$	$h(B+T)$	$h(A+B+T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	11.41 ± 0.41	11.49 ± 0.10	11.40 ± 0.00	$161.21\% \pm 314.37\%$
20%	80%	11.37 ± 0.46	11.55 ± 0.13	11.40 ± 0.00	$20.24\% \pm 33.17\%$
30%	70%	11.19 ± 0.86	11.52 ± 0.13	11.40 ± 0.00	$6.91\% \pm 29.05\%$
40%	60%	10.97 ± 0.68	11.52 ± 0.12	11.40 ± 0.00	$-1.20\% \pm 16.81\%$
50%	50%	10.79 ± 0.74	11.56 ± 0.14	11.40 ± 0.00	$-1.83\% \pm 19.15\%$
60%	40%	10.31 ± 0.84	11.56 ± 0.20	11.40 ± 0.00	$-10.99\% \pm 13.81\%$
70%	30%	9.96 ± 0.80	11.54 ± 0.21	11.40 ± 0.00	$-15.16\% \pm 17.52\%$
80%	20%	9.53 ± 1.22	11.64 ± 0.19	11.40 ± 0.00	$-16.11\% \pm 27.69\%$
90%	10%	9.38 ± 1.10	11.70 ± 0.19	11.40 ± 0.00	$-15.57\% \pm 35.87\%$

Primary indicators for the **wpbc** dataset drawn with unbiased sampling are shown in Table A.27. Transductivity and compound indicators are shown in Table A.28.

Table A.27: Primary indicators for **wpbc**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A)$	$h(B)$	$h(T)$	$h(A + B)$
10%	90%	4.37 ± 4.16	9.54 ± 4.49	11.38 ± 0.42	9.38 ± 4.21
20%	80%	7.17 ± 2.58	10.00 ± 2.61	11.46 ± 0.46	10.23 ± 2.13
30%	70%	7.77 ± 1.76	9.88 ± 1.86	11.70 ± 0.55	10.01 ± 1.51
40%	60%	7.81 ± 1.72	10.12 ± 1.58	11.81 ± 0.73	10.18 ± 1.33
50%	50%	8.12 ± 1.17	10.66 ± 1.78	11.50 ± 1.26	10.72 ± 1.35
60%	40%	7.61 ± 1.17	10.98 ± 1.42	11.48 ± 1.59	10.82 ± 1.20
70%	30%	7.67 ± 1.20	11.42 ± 1.02	11.08 ± 1.90	11.12 ± 0.91
80%	20%	7.82 ± 0.56	11.53 ± 0.72	10.88 ± 2.49	11.20 ± 0.70
90%	10%	8.07 ± 0.54	11.68 ± 0.31	9.86 ± 2.86	11.37 ± 0.31

Table A.28: Compound and transductivity indicators for **wpbc**, unbiased distribution.

$\frac{\text{trn}}{\text{trn}+\text{wrk}}$	$\frac{\text{wrk}}{\text{trn}+\text{wrk}}$	$h(A + T)$	$h(B + T)$	$h(A + B + T)$	$\frac{h(T)}{h(A+B)} - 1$
10%	90%	11.31 ± 0.33	11.47 ± 0.12	11.40 ± 0.00	$40.32\% \pm 88.19\%$
20%	80%	11.37 ± 0.37	11.47 ± 0.18	11.40 ± 0.00	$14.17\% \pm 29.30\%$
30%	70%	11.50 ± 0.43	11.53 ± 0.15	11.40 ± 0.00	$14.19\% \pm 19.35\%$
40%	60%	11.54 ± 0.54	11.56 ± 0.17	11.40 ± 0.00	$12.79\% \pm 19.34\%$
50%	50%	11.28 ± 1.00	11.51 ± 0.19	11.40 ± 0.00	$7.87\% \pm 24.60\%$
60%	40%	11.03 ± 1.16	11.62 ± 0.17	11.40 ± 0.00	$5.87\% \pm 24.79\%$
70%	30%	10.59 ± 1.17	11.66 ± 0.19	11.40 ± 0.00	$2.19\% \pm 25.60\%$
80%	20%	10.24 ± 1.28	11.70 ± 0.20	11.40 ± 0.00	$-1.45\% \pm 28.86\%$
90%	10%	9.36 ± 0.84	11.70 ± 0.19	11.40 ± 0.00	$-13.87\% \pm 26.45\%$

Appendix B

Reference Results

B.1 Biased Sampling

The next sections show the benchmark results for each of the datasets used in this work with biased sampling.

B.1.1 Reference Results for Biased bcw

Results for the inductive SVM and the TSVM are shown in Tables B.1 and B.2 respectively for the **bcw** dataset drawn with biased sampling.

Table B.1: SVM results for **bcw**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$92.36\% \pm 5.14\%$
20%	80%	$95.66\% \pm 0.48\%$
30%	70%	$95.90\% \pm 0.54\%$
40%	60%	$95.71\% \pm 0.84\%$
50%	50%	$95.80\% \pm 0.88\%$
60%	40%	$96.07\% \pm 1.02\%$
70%	30%	$96.14\% \pm 1.24\%$
80%	20%	$95.78\% \pm 1.45\%$
90%	10%	$96.00\% \pm 3.07\%$

Table B.2: TSVM results for **bcw**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$82.44\% \pm 5.09\%$
20%	80%	$89.28\% \pm 4.01\%$
30%	70%	$92.12\% \pm 2.74\%$
40%	60%	$93.28\% \pm 1.83\%$
50%	50%	$94.57\% \pm 2.24\%$
60%	40%	$94.60\% \pm 1.91\%$
70%	30%	$94.85\% \pm 2.23\%$
80%	20%	$94.85\% \pm 2.07\%$
90%	10%	$94.00\% \pm 4.09\%$

B.1.2 Reference Results for Biased hepatitis

Results for the inductive SVM and the TSVM are shown in Tables B.3 and B.4 respectively for the **hepatitis** dataset drawn with biased sampling.

Table B.3: SVM results for **hepatitis**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$79.00\% \pm 0.50\%$
20%	80%	$78.47\% \pm 1.08\%$
30%	70%	$78.80\% \pm 1.19\%$
40%	60%	$77.93\% \pm 1.62\%$
50%	50%	$78.42\% \pm 2.79\%$
60%	40%	$78.17\% \pm 4.27\%$
70%	30%	$78.64\% \pm 4.57\%$
80%	20%	$77.86\% \pm 4.99\%$
90%	10%	$73.33\% \pm 8.61\%$

Table B.4: TSVM results for **hepatitis**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$69.57\% \pm 3.00\%$
20%	80%	$69.43\% \pm 3.32\%$
30%	70%	$68.70\% \pm 2.65\%$
40%	60%	$68.26\% \pm 2.39\%$
50%	50%	$68.28\% \pm 3.07\%$
60%	40%	$69.16\% \pm 4.46\%$
70%	30%	$70.00\% \pm 6.14\%$
80%	20%	$70.35\% \pm 7.35\%$
90%	10%	$60.83\% \pm 10.43\%$

B.1.3 Reference Results for Biased ionosphere

Results for the inductive SVM and the TSVM are shown in Tables B.5 and B.6 respectively for the **ionosphere** dataset drawn with biased sampling.

Table B.5: SVM results for **ionosphere**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$63.01\% \pm 0.84\%$
20%	80%	$62.57\% \pm 0.90\%$
30%	70%	$62.42\% \pm 1.51\%$
40%	60%	$63.32\% \pm 2.27\%$
50%	50%	$62.91\% \pm 2.74\%$
60%	40%	$63.97\% \pm 3.06\%$
70%	30%	$63.90\% \pm 3.60\%$
80%	20%	$64.38\% \pm 4.53\%$
90%	10%	$64.64\% \pm 10.02\%$

Table B.6: TSVM results for **ionosphere**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$55.95\% \pm 2.99\%$
20%	80%	$54.75\% \pm 2.00\%$
30%	70%	$54.46\% \pm 1.80\%$
40%	60%	$54.47\% \pm 2.19\%$
50%	50%	$54.06\% \pm 2.79\%$
60%	40%	$55.51\% \pm 3.86\%$
70%	30%	$55.70\% \pm 4.95\%$
80%	20%	$53.85\% \pm 6.23\%$
90%	10%	$51.78\% \pm 15.08\%$

B.1.4 Reference Results for Biased musk

Results for the inductive SVM and the TSVM are shown in Tables B.7 and B.8 respectively for the **musk** dataset drawn with biased sampling.

Table B.7: SVM results for **musk**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$53.87\% \pm 5.95\%$
20%	80%	$55.33\% \pm 4.82\%$
30%	70%	$55.17\% \pm 5.29\%$
40%	60%	$56.81\% \pm 1.71\%$
50%	50%	$56.54\% \pm 2.28\%$
60%	40%	$56.30\% \pm 2.50\%$
70%	30%	$56.60\% \pm 4.00\%$
80%	20%	$56.56\% \pm 7.17\%$
90%	10%	$56.67\% \pm 8.53\%$

Table B.8: TSVM results for **musk**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$49.83\% \pm 3.17\%$
20%	80%	$49.86\% \pm 3.09\%$
30%	70%	$49.90\% \pm 3.35\%$
40%	60%	$49.78\% \pm 3.71\%$
50%	50%	$50.29\% \pm 2.53\%$
60%	40%	$50.10\% \pm 5.65\%$
70%	30%	$51.27\% \pm 5.89\%$
80%	20%	$50.21\% \pm 4.15\%$
90%	10%	$52.66\% \pm 4.57\%$

B.1.5 Reference Results for Biased pgs

Results for the inductive SVM and the TSVM are shown in Tables B.9 and B.10 respectively for the **pgs** dataset drawn with biased sampling.

Table B.9: SVM results for **pgs**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	58.85% $\pm$ 9.21%
20%	80%	67.65% $\pm$ 6.95%
30%	70%	70.95% $\pm$ 6.78%
40%	60%	74.29% $\pm$ 5.64%
50%	50%	75.38% $\pm$ 6.34%
60%	40%	78.54% $\pm$ 5.73%
70%	30%	77.67% $\pm$ 8.47%
80%	20%	78.95% $\pm$ 10.23%
90%	10%	85.00% $\pm$ 15.37%

Table B.10: TSVM results for **pgs**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	63.54% $\pm$ 7.74%
20%	80%	72.94% $\pm$ 3.68%
30%	70%	71.48% $\pm$ 6.06%
40%	60%	73.17% $\pm$ 5.72%
50%	50%	73.84% $\pm$ 5.96%
60%	40%	77.56% $\pm$ 8.35%
70%	30%	79.66% $\pm$ 6.57%
80%	20%	75.26% $\pm$ 12.42%
90%	10%	82.40% $\pm$ 21.45%

B.1.6 Reference Results for Biased **pid**

Results for the inductive SVM and the TSVM are shown in Tables B.11 and B.12 respectively for the **pid** dataset drawn with biased sampling.

Table B.11: SVM results for **pid**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$73.94\% \pm 1.33\%$
20%	80%	$74.44\% \pm 1.05\%$
30%	70%	$73.85\% \pm 1.22\%$
40%	60%	$74.32\% \pm 1.33\%$
50%	50%	$74.87\% \pm 1.92\%$
60%	40%	$74.66\% \pm 2.61\%$
70%	30%	$74.83\% \pm 3.00\%$
80%	20%	$75.29\% \pm 3.09\%$
90%	10%	$75.00\% \pm 5.78\%$

Table B.12: TSVM results for **pid**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$73.06\% \pm 1.31\%$
20%	80%	$73.20\% \pm 1.15\%$
30%	70%	$73.06\% \pm 1.61\%$
40%	60%	$73.01\% \pm 1.56\%$
50%	50%	$73.43\% \pm 2.01\%$
60%	40%	$72.96\% \pm 2.54\%$
70%	30%	$72.78\% \pm 2.39\%$
80%	20%	$73.92\% \pm 3.27\%$
90%	10%	$74.08\% \pm 4.21\%$

B.1.7 Reference Results for Biased **wabc**

Results for the inductive SVM and the TSVM are shown in Tables B.13 and B.14 respectively for the **wabc** dataset drawn with biased sampling.

Table B.13: SVM results for **w_{pbc}**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	76.37% $\pm$ 0.91%
20%	80%	75.97% $\pm$ 0.83%
30%	70%	75.97% $\pm$ 1.80%
40%	60%	75.80% $\pm$ 2.27%
50%	50%	74.55% $\pm$ 3.08%
60%	40%	73.80% $\pm$ 3.22%
70%	30%	72.88% $\pm$ 5.98%
80%	20%	70.77% $\pm$ 7.66%
90%	10%	71.05% $\pm$ 10.00%

Table B.14: TSVM results for **w_{pbc}**, biased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	62.45% $\pm$ 4.99%
20%	80%	63.71% $\pm$ 2.46%
30%	70%	62.23% $\pm$ 4.80%
40%	60%	60.84% $\pm$ 3.78%
50%	50%	62.22% $\pm$ 3.13%
60%	40%	61.89% $\pm$ 5.42%
70%	30%	59.83% $\pm$ 5.24%
80%	20%	60.51% $\pm$ 8.39%
90%	10%	61.58% $\pm$ 7.87%

B.2 Unbiased Sampling

The next sections show the benchmark results for each of the datasets used in this work with unbiased sampling.

B.2.1 Reference Results for Unbiased bcw

Results for the inductive SVM and the TSVM are shown in Tables B.15 and B.16 respectively for the **bcw** dataset drawn with unbiased sampling.

Table B.15: SVM results for **bcw**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$95.36\% \pm 0.55\%$
20%	80%	$95.87\% \pm 0.36\%$
30%	70%	$96.27\% \pm 0.47\%$
40%	60%	$96.12\% \pm 0.66\%$
50%	50%	$96.23\% \pm 0.64\%$
60%	40%	$96.32\% \pm 0.45\%$
70%	30%	$95.86\% \pm 0.81\%$
80%	20%	$96.50\% \pm 1.86\%$
90%	10%	$96.43\% \pm 2.26\%$

Table B.16: TSVM results for **bcw**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$94.85\% \pm 1.93\%$
20%	80%	$95.47\% \pm 1.41\%$
30%	70%	$94.93\% \pm 1.80\%$
40%	60%	$94.69\% \pm 2.12\%$
50%	50%	$94.74\% \pm 2.07\%$
60%	40%	$94.95\% \pm 1.69\%$
70%	30%	$95.57\% \pm 1.23\%$
80%	20%	$95.64\% \pm 1.80\%$
90%	10%	$85.85\% \pm 30.27\%$

B.2.2 Reference Results for Unbiased hepatitis

Results for the inductive SVM and the TSVM are shown in Tables B.17 and B.18 respectively for the **hepatitis** dataset drawn with unbiased sampling.

Table B.17: SVM results for **hepatitis**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	79.50% $\pm$ 0.48%
20%	80%	79.76% $\pm$ 1.49%
30%	70%	80.28% $\pm$ 2.09%
40%	60%	80.65% $\pm$ 3.07%
50%	50%	80.39% $\pm$ 3.74%
60%	40%	79.83% $\pm$ 4.41%
70%	30%	79.09% $\pm$ 3.83%
80%	20%	80.71% $\pm$ 5.11%
90%	10%	80.83% $\pm$ 7.91%

Table B.18: TSVM results for **hepatitis**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	66.28% $\pm$ 1.42%
20%	80%	65.88% $\pm$ 3.25%
30%	70%	65.92% $\pm$ 2.85%
40%	60%	66.73% $\pm$ 2.91%
50%	50%	67.10% $\pm$ 3.34%
60%	40%	66.50% $\pm$ 4.93%
70%	30%	65.90% $\pm$ 3.86%
80%	20%	68.21% $\pm$ 3.93%
90%	10%	67.50% $\pm$ 9.17%

B.2.3 Reference Results for Unbiased ionosphere

Results for the inductive SVM and the TSVM are shown in Tables B.19 and B.20 respectively for the **ionosphere** dataset drawn with unbiased sampling.

Table B.19: SVM results for **ionosphere**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$63.89\% \pm 1.11\%$
20%	80%	$64.36\% \pm 1.18\%$
30%	70%	$64.30\% \pm 1.51\%$
40%	60%	$64.90\% \pm 1.98\%$
50%	50%	$65.29\% \pm 1.69\%$
60%	40%	$66.18\% \pm 2.64\%$
70%	30%	$67.50\% \pm 3.14\%$
80%	20%	$65.16\% \pm 3.54\%$
90%	10%	$65.00\% \pm 6.26\%$

Table B.20: TSVM results for **ionosphere**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$52.72\% \pm 3.24\%$
20%	80%	$52.67\% \pm 2.20\%$
30%	70%	$52.12\% \pm 1.81\%$
40%	60%	$53.50\% \pm 2.03\%$
50%	50%	$54.70\% \pm 2.59\%$
60%	40%	$55.88\% \pm 3.73\%$
70%	30%	$57.90\% \pm 3.93\%$
80%	20%	$56.09\% \pm 5.38\%$
90%	10%	$55.71\% \pm 4.19\%$

B.2.4 Reference Results for Unbiased musk

Results for the inductive SVM and the TSVM are shown in Tables B.21 and B.22 respectively for the **musk** dataset drawn with unbiased sampling.

Table B.21: SVM results for **musk**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	53.43% $\pm$ 5.72%
20%	80%	54.93% $\pm$ 4.79%
30%	70%	56.55% $\pm$ 1.14%
40%	60%	56.60% $\pm$ 0.84%
50%	50%	57.05% $\pm$ 1.59%
60%	40%	57.41% $\pm$ 2.17%
70%	30%	57.31% $\pm$ 2.83%
80%	20%	56.99% $\pm$ 5.12%
90%	10%	56.00% $\pm$ 7.09%

Table B.22: TSVM results for **musk**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	51.81% $\pm$ 2.00%
20%	80%	51.36% $\pm$ 1.16%
30%	70%	51.65% $\pm$ 1.84%
40%	60%	52.80% $\pm$ 2.90%
50%	50%	51.13% $\pm$ 3.48%
60%	40%	51.21% $\pm$ 4.01%
70%	30%	50.77% $\pm$ 2.51%
80%	20%	50.75% $\pm$ 4.70%
90%	10%	50.44% $\pm$ 4.45%

B.2.5 Reference Results for Unbiased pgs

Results for the inductive SVM and the TSVM are shown in Tables B.23 and B.24 respectively for the **pgs** dataset drawn with unbiased sampling.

Table B.23: SVM results for **pgs**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$59.79\% \pm 6.88\%$
20%	80%	$70.00\% \pm 8.31\%$
30%	70%	$75.13\% \pm 4.78\%$
40%	60%	$77.94\% \pm 5.04\%$
50%	50%	$78.27\% \pm 5.74\%$
60%	40%	$79.03\% \pm 6.00\%$
70%	30%	$84.00\% \pm 8.43\%$
80%	20%	$81.05\% \pm 7.52\%$
90%	10%	$91.25\% \pm 8.44\%$

Table B.24: TSVM results for **pgs**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$64.68\% \pm 6.49\%$
20%	80%	$70.00\% \pm 7.30\%$
30%	70%	$73.91\% \pm 6.98\%$
40%	60%	$75.39\% \pm 7.71\%$
50%	50%	$76.92\% \pm 6.54\%$
60%	40%	$79.02\% \pm 7.20\%$
70%	30%	$85.00\% \pm 7.41\%$
80%	20%	$81.05\% \pm 9.02\%$
90%	10%	$86.10\% \pm 10.93\%$

B.2.6 Reference Results for Unbiased **pid**

Results for the inductive SVM and the TSVM are shown in Tables B.25 and B.26 respectively for the **pid** dataset drawn with unbiased sampling.

Table B.25: SVM results for **pid**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$73.77\% \pm 2.59\%$
20%	80%	$75.42\% \pm 1.42\%$
30%	70%	$75.91\% \pm 0.65\%$
40%	60%	$75.71\% \pm 1.07\%$
50%	50%	$75.96\% \pm 1.41\%$
60%	40%	$75.60\% \pm 1.44\%$
70%	30%	$75.83\% \pm 1.71\%$
80%	20%	$75.49\% \pm 3.28\%$
90%	10%	$74.60\% \pm 4.34\%$

Table B.26: TSVM results for **pid**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$73.13\% \pm 2.70\%$
20%	80%	$73.52\% \pm 1.01\%$
30%	70%	$74.46\% \pm 1.06\%$
40%	60%	$74.46\% \pm 1.45\%$
50%	50%	$74.53\% \pm 1.44\%$
60%	40%	$74.49\% \pm 1.54\%$
70%	30%	$75.04\% \pm 2.01\%$
80%	20%	$75.42\% \pm 3.18\%$
90%	10%	$75.79\% \pm 5.05\%$

B.2.7 Reference Results for Unbiased **wpbc**

Results for the inductive SVM and the TSVM are shown in Tables B.27 and B.28 respectively for the **wpbc** dataset drawn with unbiased sampling.

Table B.27: SVM results for **wpbc**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$71.06\% \pm 17.71\%$
20%	80%	$77.30\% \pm 1.69\%$
30%	70%	$77.77\% \pm 1.87\%$
40%	60%	$77.73\% \pm 2.81\%$
50%	50%	$77.28\% \pm 3.31\%$
60%	40%	$77.21\% \pm 4.38\%$
70%	30%	$76.10\% \pm 4.03\%$
80%	20%	$76.92\% \pm 5.54\%$
90%	10%	$75.26\% \pm 11.38\%$

Table B.28: TSVM results for **w_{pbc}**, unbiased distribution.

Training set size (relative to all data)	Working set size (relative to all data)	Average accuracy (over 10 distinct samplings)
10%	90%	$61.89\% \pm 5.50\%$
20%	80%	$62.13\% \pm 5.57\%$
30%	70%	$61.72\% \pm 3.68\%$
40%	60%	$62.18\% \pm 3.41\%$
50%	50%	$62.62\% \pm 3.69\%$
60%	40%	$62.15\% \pm 5.08\%$
70%	30%	$65.25\% \pm 4.61\%$
80%	20%	$62.82\% \pm 6.65\%$
90%	10%	$66.31\% \pm 10.29\%$

Appendix C

Derivatives

We now show the derivatives for Equation (8.2), which is the expression minimized using a sequential quadratic programming method. This definition has been omitted from the main text and is shown in this appendix for aesthetical purposes.

In order to calculate the partial derivatives for Equation (8.2) with respect to an arbitrary weight α_k , $\frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_k}$, we first rewrite the expression replacing the range of all summations, originally between 1 and $\ell + k$, to the range between 1 and k . This generalization does not influence the final result, as it is done only to simplify the final expression. Next, we isolate α_k from the summations where it appears, making the last term of these summations explicit and reducing their range to between 1 and $(k - 1)$. Notice that the order of terms in the summations is irrelevant. The resulting expression is shown in Equation (C.1).

$$\begin{aligned}
D_{KL}(S||\tilde{S}) = & \\
& 0.5 \left(-n - \sum_{i=1}^n \log \left[\frac{- \left(-\alpha_k x_{k,i} + \sum_{j=1}^k x_{j,i} - \sum_{j=1}^{k-1} \alpha_j x_{j,i} \right)^2 + \left(k - \alpha_k - \sum_{j=1}^{k-1} \alpha_j \right) \left(-\alpha_k x_{k,i}^2 + \sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^{k-1} \alpha_j x_{j,i}^2 \right)}{\left(k - \alpha_k - \sum_{j=1}^{k-1} \alpha_j \right)^2} \right] + \right. \\
& + \sum_{i=1}^n \log \left[\frac{- \left(\alpha_k x_{k,i} + \sum_{j=1}^{k-1} \alpha_j x_{j,i} \right)^2 + \left(\alpha_k + \sum_{j=1}^{k-1} \alpha_j \right) \left(\alpha_k x_{k,i}^2 + \sum_{j=1}^{k-1} \alpha_j x_{j,i}^2 \right)}{\left(\alpha_k + \sum_{j=1}^{k-1} \alpha_j \right)^2} \right] + \\
& + \sum_{i=1}^n \frac{\left(\alpha_k + \sum_{j=1}^{k-1} \alpha_j \right)^2 \left(-\frac{\alpha_k x_{k,i} + \sum_{j=1}^k x_{j,i} - \sum_{j=1}^{k-1} \alpha_j x_{j,i}}{k - \alpha_k - \sum_{j=1}^{k-1} \alpha_j} + \frac{\alpha_k x_{k,i} + \sum_{j=1}^{k-1} \alpha_j x_{j,i}}{\alpha_k + \sum_{j=1}^{k-1} \alpha_j} \right)^2}{\left(\alpha_k x_{k,i} + \sum_{j=1}^{k-1} \alpha_j x_{j,i} \right)^2 + \left(\alpha_k + \sum_{j=1}^{k-1} \alpha_j \right) \left(\alpha_k x_{k,i}^2 + \sum_{j=1}^{k-1} \alpha_j x_{j,i}^2 \right)} + \\
& + \sum_{i=1}^n \frac{\left(\alpha_k + \sum_{j=1}^{k-1} \alpha_j \right)^2 \left(- \left(-\alpha_k x_{k,i} + \sum_{j=1}^k x_{j,i} - \sum_{j=1}^{k-1} \alpha_j x_{j,i} \right)^2 + \left(k - \alpha_k - \sum_{j=1}^{k-1} \alpha_j \right) \left(-\alpha_k x_{k,i}^2 + \sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^{k-1} \alpha_j x_{j,i}^2 \right) \right)}{\left(k - \alpha_k - \sum_{j=1}^{k-1} \alpha_j \right)^2 \left(- \left(\alpha_k x_{k,i} + \sum_{j=1}^{k-1} \alpha_j x_{j,i} \right)^2 + \left(\alpha_k + \sum_{j=1}^{k-1} \alpha_j \right) \left(\alpha_k x_{k,i}^2 + \sum_{j=1}^{k-1} \alpha_j x_{j,i}^2 \right) \right)} \Bigg)
\end{aligned} \tag{C.1}$$

Finally, we compute the derivatives of Equation (C.1) with respect to α_k with the aid of a computer. The final result is shown in Equation (C.2). This expression is used by the sequential quadratic programming method to explicitly evaluate the first derivatives of Equation (8.2).

$$\begin{aligned}
\frac{\partial D_{KL}(S||\vec{S})}{\partial \alpha_k} = & 0.5 \left[- \frac{\left(\sum_{i=1}^n \left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(\frac{2 \left(- \left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \right)}{\left(k - \sum_{j=1}^k \alpha_j \right)^3} \right) \right)}{\left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} \right. \\
& \sum_{i=1}^n \frac{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(\frac{-x_{k,i}^2 \left(k - \sum_{j=1}^k \alpha_j \right) + 2x_{k,i} \left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right) - \sum_{j=1}^k x_{j,i}^2 + \sum_{j=1}^k \alpha_j x_{j,i}^2}{\left(k - \sum_{j=1}^k \alpha_j \right)^2} \right)}{\left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} + \\
& \sum_{i=1}^n \left(\left(\sum_{j=1}^k \alpha_j \right)^2 \left(- \frac{\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i}}{k - \sum_{j=1}^k \alpha_j} + \frac{\sum_{j=1}^k \alpha_j x_{j,i}}{\sum_{j=1}^k \alpha_j} \right)^2 \left(x_{k,i}^2 \sum_{j=1}^k \alpha_j - 2x_{k,i} \sum_{j=1}^k \alpha_j x_{j,i} + \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \right. \\
& \left. - \frac{\left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)^2}{\left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} \right. \\
& \left. 2 \left(\sum_{j=1}^k \alpha_j \right)^2 \left(\frac{x_{k,i}}{k - \sum_{j=1}^k \alpha_j} + \frac{x_{k,i}}{\sum_{j=1}^k \alpha_j} - \frac{\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i}}{\left(k - \sum_{j=1}^k \alpha_j \right)^2} - \frac{\sum_{j=1}^k \alpha_j x_{j,i}}{\left(\sum_{j=1}^k \alpha_j \right)^2} \right) \left(- \frac{\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i}}{k - \sum_{j=1}^k \alpha_j} + \frac{\sum_{j=1}^k \alpha_j x_{j,i}}{\sum_{j=1}^k \alpha_j} \right) \right. \\
& \left. - \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \\
& \left. 2 \left(\sum_{j=1}^k \alpha_j \right) \left(- \frac{\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i}}{k - \sum_{j=1}^k \alpha_j} + \frac{\sum_{j=1}^k \alpha_j x_{j,i}}{\sum_{j=1}^k \alpha_j} \right)^2 \right) + \\
& \left. - \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial D_{KL}(S||\tilde{S})}{\partial \alpha_k} &= \text{(continued...)} \\
&= \sum_{i=1}^n \left(\frac{\left(\sum_{j=1}^k \alpha_j \right)^2 \left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 \left(x_{k,i}^2 \sum_{j=1}^k \alpha_j - 2x_{k,i} \sum_{j=1}^k \alpha_j x_{j,i} + \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} + \right. \\
&\quad \left. - \frac{\left(\sum_{j=1}^k \alpha_j \right)^2 \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \left(x_{k,i}^2 \sum_{j=1}^k \alpha_j - 2x_{k,i} \sum_{j=1}^k \alpha_j x_{j,i} + \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} + \right. \\
&\quad \left. 2 \left(\sum_{j=1}^k \alpha_j \right) \left(- \left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \right) \right. \\
&\quad \left. \frac{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} + \right. \\
&\quad \left. 2 \left(\sum_{j=1}^k \alpha_j \right) \left(- \left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(k - \sum_{j=1}^k \alpha_j \right) \left(\sum_{j=1}^k x_{j,i}^2 - \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \right) \right. \\
&\quad \left. \frac{\left(k - \sum_{j=1}^k \alpha_j \right)^3 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(k - \sum_{j=1}^k \alpha_j \right)^3 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} + \right. \\
&\quad \left. \left(\sum_{j=1}^k \alpha_j \right)^2 \left(-x_{k,i}^2 \left(k - \sum_{j=1}^k \alpha_j \right) + 2x_{k,i} \left(\sum_{j=1}^k x_{j,i} - \sum_{j=1}^k \alpha_j x_{j,i} \right) - \sum_{j=1}^k x_{j,i}^2 + \sum_{j=1}^k \alpha_j x_{j,i}^2 \right) \right. \\
&\quad \left. \frac{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(k - \sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} + \right. \\
&\quad \left. \sum_{i=1}^n \left(\sum_{j=1}^k \alpha_j \right)^2 \left(\frac{x_{k,i}^2 \sum_{j=1}^k \alpha_j - 2x_{k,i} \sum_{j=1}^k \alpha_j x_{j,i} + \sum_{j=1}^k \alpha_j x_{j,i}^2}{\left(\sum_{j=1}^k \alpha_j \right)^2} - \frac{2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(\sum_{j=1}^k \alpha_j \right)^3} \right) \right. \\
&\quad \left. - \frac{\left(\sum_{j=1}^k \alpha_j \right)^2 \left(\frac{x_{k,i}^2 \sum_{j=1}^k \alpha_j - 2x_{k,i} \sum_{j=1}^k \alpha_j x_{j,i} + \sum_{j=1}^k \alpha_j x_{j,i}^2}{\left(\sum_{j=1}^k \alpha_j \right)^2} - \frac{2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)}{\left(\sum_{j=1}^k \alpha_j \right)^3} \right)}{\left(\sum_{j=1}^k \alpha_j \right)^2 \left(- \left(\sum_{j=1}^k \alpha_j x_{j,i} \right)^2 + \left(\sum_{j=1}^k \alpha_j \right) \sum_{j=1}^k \alpha_j x_{j,i}^2 \right)} \right]
\end{aligned} \tag{C.2}$$

Appendix D

Notes on Performance Measures

All experiments conducted in this work were executed in an Apple MacBook Pro machine based on an Intel Core 2 Duo processor running at 2.16 GHz, 2.0 Gb of Double Data Rate 2 (DDR2) SDRAM memory on a bus running at 667 MHz, and a Fujitsu MHW2120BH Serial ATA 120 Gb hard disk.

The machine was installed with Mac OS X 10.4.9 (8P2137), and used the Darwin kernel version 8.9.1, a variation of the BSD kernel. C implementations were compiled with the GNU C Compiler (gcc) version 4.0.1, and executed using all libraries originally shipped with the operating system. MATLAB implementations were executed in MATLAB version 7.4.0.287 (R2007a) for Mac with universal binaries.

Appendix E

Notes on Typesetting

This document has been typeset in A4 paper with narrow margins and extended line length, to better accommodate figures and large formulae. In order to improve the readability of the text, double spaces were used between lines. Also to improve readability, the Computer Modern font was used at 12pt, keeping a maximum of 65 to 75 characters per line. All typesetting was done in $\text{\LaTeX}$.

Bibliography

- [ABB00] M. B. Almeida, A. P. Braga, and J. P. Braga. SVM-KM: speeding SVMs learning with a priori cluster selection and k-means. In *Proceedings of the VIth Brazilian Symposium on Neural Networks*. IEEE Computer Society Press, November 2000.
- [ABB01a] M. B. Almeida, A. P. Braga, and J. P. Braga. Training SVMs with EDR Algorithm. *International Journal of Neural Systems*, 11(3):257–263, 2001.
- [ABB01b] M. B. Almeida, A. P. Braga, and J. P. Braga. Uma introdução a support vector machines. Technical report, Departamentos de Engenharia Eletrônica e Química da Universidade Federal de Minas Gerais, 2001.
- [BGL⁺99] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, Jr. M. Ares, and D. Haussler. Support vector machine classification of microarray gene expression data. Technical report, University of California, Santa Cruz, 1999.
- [BLC00] A. P. Braga, T. B. Ludermir, and A. F. Carvalho. *Redes Neurais Artificiais: Teoria e Aplicações*. LTC, 2000.
- [BM98] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.
- [BMM99] P. S. Bradley, O. L. Mangasarian, and D. R. Musicant. Optimization methods in massive datasets. Technical report, University of Wisconsin in Madison, 1999.
- [Bre94] Leo Breiman. Bagging predictors. Technical report, University of California at Berkeley, 1994.
- [BSS92] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley, second edition, 1992.

- [Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [Cac94] Christian Cachin. Pedagogical pattern selection strategies. *Neural Networks*, 7(1):175–181, 1994.
- [CKB87] G. Cestnik, I. Kononenko, and I. Bratko. Assistant-86: A knowledge-elicitation tool for sophisticated users. *Progress in Machine Learning*, pages 31–45, 1987.
- [Cro03] Benjamin Crowell. *Simple Nature*. Benjamin Crowell, USA, 2003.
- [CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.
- [CVW99] O. Chapelle, V. Vapnik, and J. Weston. Transductive inference for estimating values of functions. *Advances in Neural Information Processing Systems*, 12:421–427, 1999.
- [CWD03] Yisong Chen, Guoping Wang, and Shihai Dong. Learning with progressive transductive support vector machine. *Pattern Recogn. Lett.*, 24(12):1845–1855, 2003.
- [dBKMR05] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross entropy method. *Annals of Operations Research*, 134:19–67, 2005.
- [DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2001.
- [DLLP97] Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.
- [DSS93] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705 – 719, 1993.
- [EGBG03] Edwin J. Elton, Martin J. Gruber, Stephen J. Brown, and William N. Goetzmann. *Modern Portfolio Theory and Investment Analysis*. John Wiley & Sons, Inc., New York, sixth edition, 2003.

- [EP01] Deniz Erdogmus and Jose C. Principe. Entropy minimization algorithm for multilayer perceptrons. In *International Joint Conference on Neural Networks*, pages 3003–3008, 2001.
- [FL01] Gary W. Flake and Steve Lawrence. Efficient SVM regression training with SMO. *Machine Learning*, 2001.
- [Fle87] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Inc., second edition, 1987.
- [FS99] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999. Appearing in Japanese, translation by Naoki Abe.
- [GB05] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in Neural Information Processing Systems*, 17:529–536, 2005.
- [HAH01] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, Inc., Upper Saddle River, New Jersey, 2001.
- [Har28] Ralph V. L. Hartley. Transmission of information. *Bell System Technical Journal*, pages 535–563, July 1928.
- [Hay94] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 866 Third Avenue, New York, New York 10022, 1994.
- [Heb49] D. O. Hebb. *The Organization of Behavior*. Wiley, 1949.
- [Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective properties. In *Proceedings of the National Academy of Sciences*, 79, pages 2554–2558, 1982.
- [HR87] C. Harley and R. Reynolds. Analysis of e. coli promoter sequences. *Nucleic Acids Research*, 15:2343–2361, 1987.
- [HRK96] David Halliday, Robert Resnick, and Kenneth Krane. *Física I, II, III, IV*. LTC – Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, forth edition, 1996.

- [Joa98] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT Press, 1998.
- [Joa99] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.
- [KK01] Michihiro Kuramochi and George Karypis. Gene classification using expression profiles: A feasibility study. In *IEEE International Conference on Bioinformatics and Biomedical Engineering*, pages 191–200, 2001.
- [Kul72] Solomon Kullback. Minimum discrimination information estimation and application. Technical report, Department of Statistics, George Washington University, 1972.
- [LG94] Alberto Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, second edition, 1994.
- [Mac03] David J. C. MacKey. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [Mai03] Thiago Turchetti Maia. Strategies for combining boosting and support vector machines. Master’s thesis, Programa de Pós-Graduação em Engenharia Elétrica, 2003.
- [MBdC07] Thiago Turchetti Maia, Antonio Pádua Braga, and André F. de Carvalho. Hybrid classification algorithms based on boosting and support vector machines. *Kybernetes*, 2007. to appear.
- [MM98] O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. Technical report, University of Wisconsin in Madison, 1998.
- [MM99] O. L. Mangasarian and D. R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.
- [MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

- [MP69] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry*. MIT Press, 1969.
- [MPR05] Shie Mannor, Dori Peleg, and Reuven Rubinstein. The cross entropy method for classification. In *Proceeding of the 22nd International Conference on Machine Learning*, 2005.
- [MSW95] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.
- [Mun92] P. W. Munro. Repeat until bored: A pattern selection strategy. *Advances in neural information processing systems*, 4:1001–1008, 1992.
- [NLM99] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.
- [OC89] M. C. O’Neill and F. Chiafari. Escherichia coli promoters. ii. a spacing-class dependent promoter search protocol. *Journal of Biological Chemistry*, 264:5531–5534, 1989.
- [OFG97] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop*, pages 276–285. IEEE, 1997.
- [O’N89] M. O’Neill. Escherichia coli promoters: Consensus as it relates to spacing class specificity, repeat substructure, and threedimensional organization. *Journal of Biological Chemistry*, 264:5522–5530, 1989.
- [ORM00] Takashi Onoda, Gunnar Rätsch, and Klaus-Robert Muller. Applying support vector machines and boosting to a non-intrusive monitoring system for household electric appliances with inverters. In *Proceedings of Neural Computing 2000*, 2000.
- [Pla98a] John C. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1998.
- [Pla98b] John C. Platt. Sequential minimal optimization: a fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998.
- [Por98] Noah Porter, editor. *Webster’s Revised Unabridged Dictionary*. C. & G. Merriam Co., 1998.

- [PV98] Massimiliano Pontil and Alessandro Verri. Properties of support vector machines. *Neural Computation*, 10:955–974, 1998.
- [Qui92] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. The Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann, October 1992.
- [RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- [Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65:386–408, 1958.
- [RTR⁺01] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, , and T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *PNAS*, 98(26):15149–15154, 2001.
- [Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [Sch92] R. E. Schapire. *The Design and Analysis of Efficient Learning Algorithms*. MIT Press, 1992.
- [Sch99] R. E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.
- [Sch02] R. E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
- [SED⁺88] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Symposium on Computer Applications and Medical Care*, pages 261–265. IEEE Computer Society Press, 1988.
- [Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.
- [SMW95] W. N. Street, O. L. Mangasarian, and W. H. Wolberg. An inductive learning approach to prognostic prediction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 522–530. Morgan Kaufmann, 1995.

- [SS04] D. W. Scott and S. R. Sain. *Multi-Dimensional Density Estimation*, volume 23: Data Mining and Computational Statistics, pages 229–258. Elsevier, Amsterdam, 2004.
- [Str86] Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.
- [SWHB89] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.
- [Tho18] William M. Thorburn. The myth of occam’s razor. *Mind*, 27:345–353, 1918.
- [TSN90] G. Towell, J. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, 1990.
- [Vap79] Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data*. (in Russian), Nauka, Moscow, 1979. (English translation: Vapnik, V. N. 1982. *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York).
- [Vap82] V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, New York, 1982.
- [Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
- [Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.
- [Vap99] Vladimir N. Vapnik. *Three Remarks on the Support Vector Method of Function Estimation*, chapter 3, pages 25–41. The MIT Press, Cambridge, Massachusetts, 1999.
- [VC71] Vladimir N. Vapnik and A. J. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.
- [WH60] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *Institute of Radio Engineers, Western Electronic Show and Convention*, 1960.
- [Wik07] Wikipedia. <http://www.wikipedia.org/>, 2007.

- [WM90] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, pages 9193–9196, 1990.
- [Yeu02] Raymond W. Yeung. *A First Course in Information Theory*. Kluwer Academic/Plenum Publishers, 2002.
- [Zha92] J. Zhang. Selecting typical instances in instance-based learning. In *Proceedings of the Ninth International Machine Learning Conference*, pages 470–479. Morgan Kaufmann, 1992.