# UM ALGORITMO DE NUVEM DE PARTÍCULAS PARA COMBINAÇÃO DE CLASSIFICADORES EM APRENDIZADO MULTI-VISÃO

ZILTON JOSÉ MACIEL CORDEIRO JUNIOR

# UM ALGORITMO DE NUVEM DE PARTÍCULAS PARA COMBINAÇÃO DE CLASSIFICADORES EM APRENDIZADO MULTI-VISÃO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Gisele Lobo Pappa

Belo Horizonte

Março de 2011

Cordeiro Junior, Zilton José Maciel
C794a Um Algoritmo de Nuvem de Partículas para Combinação de Classificadores em Aprendizado Multi-Visão / Zilton José Maciel Cordeiro Junior. — Belo Horizonte, 2011
xxii, 82 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientadora: Gisele Lobo Pappa

1. Computação - Teses. 2. Sistemas de recuperação da informação - Teses. I. Título.

CDU 519.6*73 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# FOLHA DE APROVAÇÃO

Um algoritmo de nuvem de partículas para combinação de classificadores em aprendizado multi-visão

**ZILTON JOSÉ MACIEL CORDEIRO JUNIOR**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROFA. GISELE LOBO PAPPA - Orientadora
Departamento de Ciência da Computação - UFMG

PROF. LUIS ENRIQUE ZÁRATE
Departamento de Ciência da Computação - PUC-MG

PROF. ADRIANO ALONSO VELOSO
Departamento de Ciência da Computação - UFMG

PROF. MARCOS ANDRÉ GONÇALVES
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 25 de março de 2011.

# Agradecimentos

Primeiramente à Deus, que é a força que nos guia.

Aos meus pais, Zilton e Ruth por serem os pilares da minha formação como pessoa.

Aos meus irmãos Paulo, Cássia e Joice, que sempre me apoiaram.

Aos meus eternos amigos da graduação, que sempre torceram pelo meu sucesso.

À minha orientadora e Professora Gisele L. Pappa, que me auxiliou, ensinou e estimulou em todas as etapas do mestrado. Meu muito obrigado!

À Universidade Federal de Minas Gerais, pela estrutura.

Ao Departamento de Ciência da Computação - DCC, pela oportunidade de concluir mais essa fase em minha vida.

Ao laboratório SPEED pela estrutura.

Aos meus antigos professores da graduação na UESC, pelos ensinamentos de toda a vida profissional.

Aos professores e funcionários do DCC pelos ensinamentos e ajuda.

Às agências financiadoras do projeto de pesquisa que se insere esse mestrado.

*"A simplicidade é o último grau de sofisticação."*
(Leonardo da Vinci)

# Resumo

O aprendizado multi-visão ou multi-modalidade está se tornando cada vez mais popular, por fornecer diferentes representações de um problema a partir das quais se pode aprender. Dada a tarefa de classificação de vídeo, por exemplo, o som, a imagem e as legendas poderiam ser consideradas visões.

A ideia principal do aprendizado multi-visão é que, ao aprender a partir dessas representações separadamente, pode-se obter previsões melhores do que ao agregar todas essas visões em uma única base de dados. Porém, um modelo de classificação é criado para cada visão, e as saídas disponibilizadas por cada um deles deve ser combinada para fornecer uma classe final para cada instância.

Esta dissertação propõe um algoritmo de Otimização por Nuvem de Partícula (PSO) para combinar as saídas advindas de diferentes classificadores. O PSO trabalha em dois contextos: o primeiro leva em consideração somente a classe/confiança atribuída por um classificador na categorização de uma instância, aplicando pesos a cada visão. O segundo, além de atribuir pesos às visões, atribui também pesos a cada classe.

Experimentos foram feitos em duas bases de dados, com três visões cada, e comparados com três diferentes métodos existentes na literatura: o voto da maioria, o algoritmo de *Borda Count* e a teoria de Dempster-Shafer. Além desses, foi feita uma comparação com a abordagem que utiliza todas as visões juntas em uma única base de dados. Na grande maioria dos experimentos, o PSO obteve resultados estatisticamente melhores que as outras abordagens avaliadas.

**Palavras-chave:** Classificação, Aprendizado Multi-Visão, Otimização por Nuvem de Partículas, Combinação de Classificadores.

# Abstract

The multi-view or multi-modality learning approach is becoming popular for providing different representations of a problem from which classifiers can learn from. Given the task of video classification, for example, the sound, the image and the subtitles may be considered views.

The main idea behind multi-view learning is that learning from these representations separately can lead to better gains than merging them into a single dataset. Hence, a classification model is created for each view, and outputs provided by each of them must be combined to provide a final class for each example.

This dissertation proposes a Particle Swarm Optimization (PSO) algorithm to combine the outputs coming from different views. The PSO works in two contexts: the first takes into account only the class/confidence assigned by a classifier in the categorization of an instance, applying weights to each view. The second, besides assigning weights to views, also assigns weights to each class.

Experiments were performed in two datasets, each one with three views, and compared with three different methods from the literature: the majority vote, the Borda Count algorithm and the Dempster-Shafer theory. Then, a comparison was made with the approach that uses all views together into a single dataset. The PSO obtained statistically better results than the other approaches evaluated in the majority of the experiments.

**Keywords:** Classification, Multi-view Learning, Particle Swarm Optimization, Combining Classifiers.

# Lista de Figuras

# Lista de Tabelas

# Lista de Algoritmos

# Sumário

# Capítulo 1

# Introdução

A quantidade de dados armazenados e disponíveis, principalmente na Web, vem aumentando cada vez mais, com cerca de 295 milhões de Terabytes de dados *online* em 2009. Esse crescimento inviabiliza a compreensão, indexação e busca de informações úteis de forma trivial, gerando a necessidade de formas automatizadas de análise de dados. Essa grande quantidade de dados permite, também, que uma mesma informação seja representada de maneiras distintas. Por exemplo, hoje no *YouTube* podemos encontrar vídeos e gerar automaticamente as legendas, gerando uma nova representação dos dados. Da mesma forma, diferentes corporações possuem conjuntos de dados referentes a domínios distintos a respeito de um cliente, como por exemplo saúde, crédito, lugares frequentados, etc. Essas bases são conhecidas como bases de dados complexas, i.e., são produzidas por processos distintos de obtenção e manipulação dos dados.

Nessa direção, a demanda e importância de métodos capazes de lidar de forma efetiva com esses diferentes conjuntos de dados, representando um mesmo problema, aumentou consideravelmente. Técnicas de Aprendizado de Máquina (AM) [Mitchell, 1997] tradicionais resolvem esse problema unindo os dados de diferentes representações em uma única base de dados, e depois de uma fase de pré-processamento (que pode incluir seleção de atributos), utilizam um algoritmo para aprender a partir desse conjunto de instâncias. Porém, existe uma forma alternativa de aprender nesses dados.

Em contraste com os métodos de aprendizado tradicionais, que aprendem a partir de uma base de dados única, algoritmos de aprendizado multi-visão [Culp & Michailidis, 2009] ou aprendizado multi-modalidade [Tong et al., 2005] têm se tornado populares por aprenderem a partir de diferentes representações de um problema. O aprendizado multi-visão é comumente utilizado em contextos semi-supervisionados para resolver problemas de classificação [Wang & Zhou, 2008] [Blum & Mitchell, 1998]. Um problema de classificação tem como principal objetivo escolher, a partir de um conjunto de classes

pré-definidas, a que melhor caracteriza a instância sendo analisada. Problemas semi-supervisionados, por sua vez, são aqueles em que parte das instâncias já possuem a classe conhecida (são rotulados) enquanto a grande maioria das instâncias possui classe desconhecida.

A principal ideia do aprendizado multi-visão (AMV) é caracterizar um problema utilizando diferentes representações dos dados (visões), onde podemos aprender nessas representações separadamente e depois combiná-las. Já foi demonstrado que dessa forma podemos obter maiores ganhos que ao utilizar as informações agrupadas em uma única base de dados [Nigam et al., 2000]. Considere, por exemplo, uma aplicação de categorização de vídeos, como a ilustrada pela Figura 1.1. Na classificação pode-se utilizar pelo menos três representações distintas: o áudio, a legenda e a imagem. O princípio do aprendizado multi-visão afirma que aprender em um modelo de áudio, em um modelo de legenda, em um modelo de imagem e, então, combiná-los é mais eficaz do que gerar um modelo único considerando todos os atributos ao mesmo tempo [Nigam et al., 2000] [Wang et al., 2010].

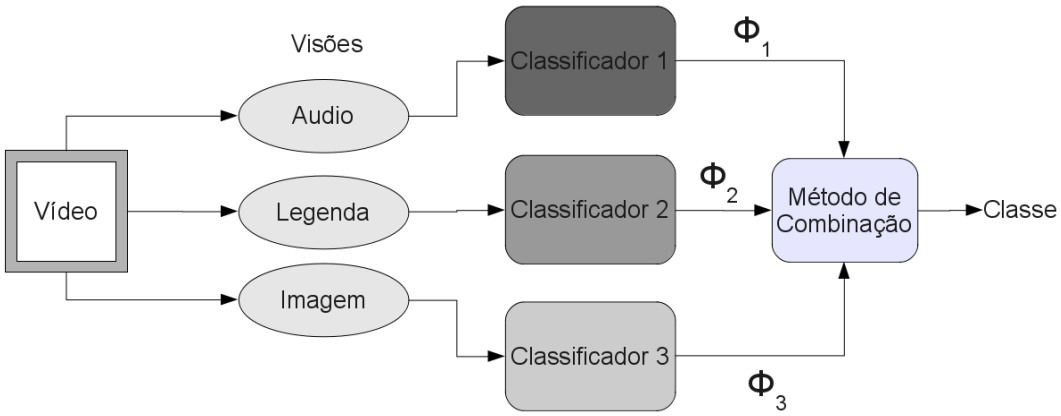



**Figura 1.1.** Classificação em aprendizado multi-visão para um vídeo $e$.

Utilizar o aprendizado multi-visão traz claras vantagens sobre o método mono-visão tradicional. Primeiro, ele permite aos classificadores trabalhar com dados em dimensionalidade reduzida, pois $n$ atributos são divididos em um conjunto de $v$ visões. Segundo, diferentes algoritmos de classificação podem ser utilizados para o aprendizado em diferentes visões, permitindo ao usuário selecionar o algoritmo que melhor se ajuste às características dos dados em cada visão. Além disso, problemas complexos em diversas áreas de aplicação, como anotação de funções de gene em bioinformática e sumarização de vídeo têm demonstrado grandes ganhos a partir de técnicas que consideram mais de um tipo de dados [Fujibuchi & Kato, 2007]. Finalmente, diferentes informações sobre uma mesma instância podem ajudar a resolver outros problemas em

aprendizado de máquina, como transferência de aprendizado [Baxter, 1998] e aprendizado semi-supervisionado [Blum & Mitchell, 1998].

Além da utilização do aprendizado multi-visão por meio de abordagens semi-supervisionadas, diversos trabalhos também focam em aprendizado não-supervisionado [Ghani, 2002] e supervisionado [Ng & Cardie, 2003] [Wang et al., 2010] [Junior et al., 2010] [Junior & Pappa, 2011]. Essa dissertação trata do aprendizado multi-visão supervisionado, aquele em que a classe de todas as instâncias é conhecida. Com relação ao último, classificar as diferentes visões não é um problema complexo, já que cada uma delas pode ser vista como um problema de classificação mono-visão, e qualquer um dos algoritmos de aprendizado consolidados na literatura, como por exemplo, o SVM [Vapnik, 1995], podem ser facilmente aplicados aos dados. Porém, o maior problema em aprendizado multi-visão está em como combinar as decisões provenientes dos classificadores, distintos ou não, criados a partir de cada visão. Aqui, propomos resolver esse problema utilizando um algoritmo de Otimização por Nuvem de Partículas (PSO) [Kennedy et al., 2001].

O PSO é um algoritmo inspirado no comportamento social de um bando de pássaros, e é formado por uma população de indivíduos, chamados de partículas. Cada partícula representa uma solução candidata para o problema a ser resolvido. A nuvem se move em um espaço de soluções através da *cooperação* e *competição* entre suas partículas, que se beneficiam de sua própria experiência e da experiência de outros membros da nuvem durante a busca por uma melhor solução.

A escolha do PSO para resolver o problema de combinação de decisões, provenientes de diferentes visões e classificadores, está baseado na combinação de técnicas de aprendizado local e global que este algoritmo permite. A partícula, por se deixar influenciar tanto pelos vizinhos (global) quanto por si própria (local), ao se mover no espaço de soluções, pode trazer vantagens com relação a métodos que consideram apenas o conhecimento local ou global. Além disso, embora o PSO tenha sido inicialmente proposto para otimização de valores contínuos para funções, ao longo do tempo, ele foi refinado e melhorado, sendo especialmente bem sucedido em tarefas de classificação [Mohemmed et al., 2009] [Wen et al., 2011] [Zhou et al., 2010] [Khan et al., 2010] [Cao & Liu, 2010] [Evans & Zhang, 2008].

Esse trabalho apresenta dois métodos baseados em PSO para combinação de visões: PSO-WV( *PSO-Weight View*) e PSO-WC (*PSO-Weight Classes*). O PSO-WV é utilizado para ponderar $v$ classificadores, heterogêneos ou homogêneos, em diferentes visões de uma base de dados, e pondera a decisão de cada classificador de acordo com seu rendimento naquela visão. Já o PSO-WC leva em consideração, além dos $v$ classificadores, o *ranking* das classes disponibilizado pelos algoritmos de classificação,

atribuindo pesos a cada uma delas. Essa abordagem permite capturar o erro inerente a cada algoritmo de classificação em cada visão dos dados, e considera que algumas visões podem ser melhores na predição de determinadas classes que outras, e portanto devem ser mais confiáveis.

Experimentos foram realizados em dois contextos: classificação automática de documentos de texto [Salles et al., 2010] e classificação de vídeos em redes sociais [Benevenuto et al., 2009], cada base com três visões dos dados. Resultados experimentais foram comparados com aqueles produzidos por outros métodos de combinação estado da arte na literatura, incluindo o voto da maioria, o algoritmo de *Borda Count* e um método baseado na teoria da evidência de Dempster-Shafer. Na grande maioria dos casos, o PSO foi considerado estatisticamente melhor que os métodos supracitados. Além disso, o PSO-WC, que pondera tanto as visões quanto as classes, obteve resultados melhores que o PSO-WV.

Além disso, todos os métodos utilizados na fase de experimentação dessa dissertação estão aglutinados na ferramenta MultiViL [Junior et al., 2011], disponível gratuitamente na *Web*[1]. A MultiViL é a única ferramenta disponível para experimentação em bases de dados multi-visão, além de permitir, também, a classificação tradicional em uma única base de dados. Através da MultiViL é possível fazer uma análise da cooperação e concordância das visões executadas em cada algoritmo de classificação, como aqueles utilizados na fase experimental desta dissertação.

## 1.1 Contribuições

Podemos sumarizar as principais contribuições deste trabalho como segue:

1. Apresentação de diferentes modelos existentes para classificação automática utilizando aprendizado multi-visão;

2. Revisão bibliográfica, apresentando diversos trabalhos em aprendizado multi-visão no contexto de classificação. Com relação ao PSO, também são apresentados diversos trabalhos da literatura em classificação de dados;

3. Adequação de técnicas consolidadas na literatura (voto da maioria, *Borda Count* e teoria da evidência de Dempster-Shafer) para combinação de classificadores no contexto do aprendizado multi-visão;

4. Apresentação de um novo método de combinação e resolução de conflito dos classificadores, em cada visão, utilizando o método PSO;

[1] http://multivil.sourceforge.net/

5. Avaliação e validação da técnica proposta por meio de métricas como a acurácia, realizadas em dois cenários de aplicação real, avaliando como a predição sob a perspectiva multi-visão melhora a classificação realizada; e

6. Disponibilização de uma ferramenta completa para aprendizado multi-visão (MultiViL).

## 1.2 Organização da Dissertação

Esta dissertação é organizada como se segue:

- O Capítulo 2 dispõe a tarefa de classificação nos contexto mono e multi-visão, abordando principalmente métodos utilizados na literatura para a combinação de classificadores;

- O Capítulo 3 apresenta o algoritmo de Otimização por Nuvem de Partículas em sua concepção original;

- O Capítulo 4 descreve o algoritmo PSO implementado neste trabalho para resolver o problema da combinação de $v$ visões de uma base de dados classificadas em $n$ algoritmos, que gerou os métodos PSO-WV e PSO-WC;

- O Capítulo 5 apresenta a MultiViL, ferramenta fruto deste trabalho, construída especificamente para classificação multi-visão, além de permitir também uma classificação mono-visão;

- O Capítulo 6 disponibiliza os resultados experimentais, apresentando uma análise com relação à concordância e cooperação entre as visões, os algoritmos e ambos em duas bases de dados multi-visão; e finalmente

- O Capítulo 7 apresenta as conclusões deste trabalho e delineia os trabalhos futuros.

# Capítulo 2

# Aprendizado Mono e Multi-visão

A área de aprendizado de Máquina (AM) pesquisa métodos computacionais relacionados à aquisição automática de novos conhecimentos, novas habilidades e novas formas de organizar o conhecimento já existente [Mitchell, 1997]. No contexto desta dissertação, podemos definir como métodos de AM aqueles capazes de aprender a partir de um conjunto de instâncias, denominado conjunto de treinamento. No conjunto de treinamento, cada instância é descrita por um conjunto de atributos. De acordo com o tipo de informação disponível nessas instâncias, três tipos de aprendizado podem ser realizados:

1. Supervisionado: Em algoritmos de aprendizado supervisionado, o rótulo da classe a qual as instâncias pertencem é conhecido;

2. Não-supervisionado: Em algoritmos de aprendizado não-supervisionado, as instâncias fornecidas não são rotulados, e o algoritmo agrupa-as por meio de alguma métrica de similaridade, formando agrupamentos (*clusters*); e

3. Semi-supervisionado: São aqueles algoritmos que aprendem utilizando uma combinação das faces oferecidas pelo aprendizado supervisionado e o não-supervisionado. Esses algoritmos trabalham com uma grande quantidade de dados não rotulados, em conjunto com uma pequena parcela de dados rotulados.

Independente da abordagem utilizada, o aprendizado tradicional utiliza apenas uma base de dados, que engloba todos os atributos disponíveis para realização da tarefa. Aqui, esse tipo de aprendizado é referido como mono-visão (onde a visão corresponde a base de dados sem nenhuma separação dos atributos). Porém, nos últimos anos, com

a crescente disponibilidade de dados, pesquisas nas áreas de aprendizado multi-visão ou multi-modalidade começaram a se destacar.

Esse trabalho está focado em aprendizado multi-visão supervisionado, e tem particular interesse na tarefa de classificação. A Seção 2.1 discute a classificação sob uma perspectiva mono-visão, enquanto a Seção 2.2 detalha o aprendizado multi-visão, e os métodos mais comumente utilizados nesta abordagem.

## 2.1 Classificação sob uma Perspectiva Mono-visão

A tarefa de classificação [Tan et al., 2005] tem como principal objetivo aprender a classificar instâncias em conjuntos de classes pré-definidas, baseado em um conjunto de atributos comuns a elas.

Seja $X$ uma matriz contendo informações referentes aos valores de $p$ atributos para $n$ instâncias, nas quais cada elemento $x_{ij}$ representa o valor do $j-$ésimo atributo para a $i-$ésima instância. Para cada instância, tem-se $x_i = \langle(x_{i1}, ..., x_{ip})\rangle$, $C_i$, onde $C_i$ identifica a classe a que pertence a instância. Como mencionado anteriormente, os classificadores são treinados utilizando um conjunto de treinamento. Por treinamento, define-se a busca de relações entre os atributos da base e sua respectiva classe. Os classificadores aprendidos são então utilizados para classificar novas amostras, representadas por um conjunto de teste.

Na literatura referente ao problema de classificação por meio do aprendizado supervisionado [Tan et al., 2005] [Witten & Frank, 2005], são tradicionalmente utilizados modelos de classificação que tratam o referido problema sob a perspectiva de uma única visão dos dados. Diversos classificadores são utilizados para essa tarefa. Aqui descrevemos quatro deles, cuja escolha foi baseada nos classificadores estado da arte e também aqueles indicados para a aplicação inicialmente considerada neste trabalho, classificação de texto [Baeza-Yates & Ribeiro-Neto, 2010]. Esses classificadores são:

1. SVM (*Support Vector Machines*) [Vapnik, 1995]: Consiste em um método utilizado para o reconhecimento de padrões definidos em um espaço vetorial de alta dimensionalidade usando uma função *kernel*, que transforma o espaço de dados de forma que ele se torne linearmente separável. Com isso, para este novo espaço linear, cria-se um hiperplano ótimo separando as classes. Assim, o problema resume-se a encontrar uma superfície de decisão que melhor separe os dados em duas classes distintas (classificação binária). Esse modelo binário foi estendido para suportar regressão e problemas de classificação de múltiplas classes. Uma solução é ótima no sentido de que a margem entre o hiperplano e os vetores de

características mais próximos das duas classes é máxima. Assim, dadas duas classes e um conjunto de pontos que pertencem a essas classes, o SVM determina o hiperplano que separa os pontos de forma a colocar o maior número de pontos da mesma classe do mesmo lado, enquanto maximiza a distância de cada classe a esse hiperplano. A distância de uma classe a um hiperplano é a menor distância entre ele e os pontos dessa classe e é chamada de margem de separação. Os vetores de características que são mais próximos do hiperplano são chamados de vetores de suporte (*support vectors*), significando que a posição dos outros vetores não afetam o hiperplano. Atualmente, o SVM representa o estado da arte em classificação.

2. *Naïve Bayes* [Lewis, 1998]: É um dos métodos mais simples de classificação existentes, sendo o principal representante do grupo de classificadores probabilísticos. Este classificador se baseia no teorema de Bayes, que assume forte independência entre os termos ("*Naïve Bayes assumption*"). Em outras palavras, o *Naïve Bayes* assume que a ocorrência ou ausência de um termo não possui nenhuma relação com a ocorrência ou ausência de outros termos.

3. *Rocchio* [Rocchio, 1971]: Seu modelo de classificação baseia-se em vetores protótipos ou centroides para cada classe $c_i$ da base de dados. O *Rocchio* estima esses centroides utilizando a soma dos vetores das instâncias de treinamento da categoria. Posteriormente, uma instância é rotulada de acordo com a sua similaridade com o respectivo centroide das classes, como, por exemplo, a similaridade do cosseno.

4. *KNN (K-Nearest Neighbor)* [Cover & Hart, 1967]: O KNN tradicional, é um algoritmo que decide sobre a classe de uma instância procurando as $k$ instâncias do conjunto de treinamento mais semelhantes a ela, realizando uma análise da frequência das classes. As $k$ instâncias mais próximas são identificadas com base em uma métrica de similaridade, que pode ser uma simples distância Euclidiana.

A utilização de qualquer um dos classificadores acima segue o modelo padrão de classificação treinamento/teste. Porém, hoje em dia uma das abordagens mais robustas de classificação são aquelas baseadas em comitês de classificadores [Dietterich, 2000]. Embora esses métodos estejam fora do escopo deste trabalho, é importante que sejam citados, pois eles também requerem a combinação de resultados de classificadores, como tratado na Seção 2.3. Porém, note que enquanto comitês trabalham com diferentes amostras de uma mesma base de dados, i.e., mesmo conjunto de atributos, este trabalho discute como combinar resultados de conjuntos de dados, com o mesmo número de

instâncias e diferentes números de atributos. Como nosso interesse está nos métodos de combinação e não nos comitês em si, esses serão revisados na Seção 2.3. Porém, note que os métodos de combinação utilizados por comitês são mais pertinentes no contexto desse trabalho, por sua comprovada eficácia e capacidade de combinar mais de dois classificadores, que podem ser distintos ou não.

Com isso, uma das áreas mais ativas de pesquisa em aprendizado supervisionado é o estudo de métodos para construir bons comitês de classificadores, uma vez que comitês são, muitas vezes, mais precisos do que os classificadores individuais que os compõem [Katakis et al., 2010]. Isso acontece somente se os classificadores individuais discordam entre si [Hansen & Salamon, 1990]. Como exemplo, tomemos um comitê com três classificadores $\{\varphi_1, \varphi_2, \varphi_3\}$, e considere uma nova instância $e$. Se os três classificadores são idênticos, então, quando $\varphi_1(e)$ está errado, $\varphi_2(e)$ e $\varphi_3(e)$ também estarão errados. No entanto, se os erros cometidos pelos classificadores são não-correlacionados, quando $\varphi_1(e)$ estiver errado, $\varphi_2(e)$ e $\varphi_3(e)$ podem estar corretos, assim, um simples voto da maioria classificaria corretamente $e$.

## 2.2 Classificação sob uma Perspectiva Multi-visão

Essa seção revisa os fundamentos do aprendizado multi-visão, e apresenta diversos trabalhos relacionados a essa área, além de trabalhos relacionados a comitês de classificadores, pois, esses, também utilizam diversos métodos de combinação de algoritmos de classificação, os quais se aplicam, em muitos casos, à combinação de visões de uma base de dados, como os métodos descritos na Seção 2.3.

Como mencionado na Seção 2.1, o método mais tradicional de classificação é aquele onde apenas uma visão dos dados é explorada. Esta seção, em contrapartida, descreve o aprendizado multi-visão e discute suas vantagens em relação ao aprendizado mono-visão.

Ao fornecer a um algoritmo de classificação diversas representações de um mesmo problema, esse pode aprender diferentes modelos e depois combiná-los para obter um modelo de classificação/previsão único e mais robusto [Nigam et al., 2000]. Conforme descrito em [Wang et al., 2010], o ponto máximo das pesquisas em aprendizado multi-visão está em como aprender a partir de diferentes atributos, de uma mesma base de dados, através de múltiplas visões. Com isso, em diversas pesquisas (Tabela 2.1), tem sido provada a superior generalização do aprendizado multi-visão com relação ao método usual de mono-visão.

A principal vantagem da utilização do aprendizado multi-visão é que os algorit-

mos de classificação podem ser aplicados a conjuntos disjuntos de atributos. Com isso, é possível aprender diferentes modelos em diferentes classificadores, aplicando cada algoritmo de classificação à visão que o permita uma classificação mais acurada. Em outras palavras, como cada classificador manipula o espaço de dados de maneira distinta, pode-se aplicar os classificadores às visões que melhor permitam a separação das classes no espaço manipulado. Após a classificação em cada visão, os resultados devem ser combinados, fazendo com que as visões cooperem e, por fim, concordem com relação à classe de uma determinada instância.

Diversos trabalhos utilizam o aprendizado multi-visão para adicionar maiores e melhores informações aos algoritmos de classificação (Tabela 2.1), permitindo que esses forneçam modelos classificatórios mais satisfatórios. As pesquisas nessa área são recentes e diversos algoritmos têm sido propostos, por meio da utilização dos aprendizados de máquina supervisionado, não-supervisionado e semi-supervisionado.

A seguir, são definidos os fundamentos do aprendizado multi-visão.

### 2.2.1 Fundamentos

Em problemas de aprendizado multi-visão, uma instância $e$ é descrito por uma série de atributos distintos em cada visão. Considere, por exemplo, a tarefa de classificação, tratada nesta dissertação. Em um domínio com $v$ visões $V_1, V_2, ..., V_v$, uma instância rotulada pode ser vista como uma tupla $\langle\ e_1, e_2, ..., e_v, l\ \rangle$, onde $l$ é um rótulo e $[x_1], [x_2], ..., [x_v]$ são conjuntos de dados em $v$ visões. Definidas as visões, um modelo é criado de forma independente para cada visão, sendo necessária uma posterior combinação desses modelos para predição das classes de novas instâncias. Como mostrado em [Nigam et al., 2000], a utilização de $v$ visões dos dados leva os classificadores a acertarem mais do que os métodos que utilizam apenas uma visão. Mesmo que esses pressupostos tenham sido originados do aprendizado semi-supervisionado, eles podem facilmente ser extrapolados para o contexto do aprendizado supervisionado, conforme mostrado em [Wang et al., 2010].

Existem diversos pesquisadores trabalhando com aprendizado multi-visão. Porém, a maioria dos estudos é aplicado no contexto semi-supervisionado. O termo aprendizado multi-visão tornou-se aparente com os trabalhos de [de Sa, 1993], mas as pesquisas nessa área se tornaram mais fortes com os trabalhos de [Blum & Mitchell, 1998], onde o aprendizado se aplica a problemas que têm um divisão natural do espaço de características (visões), e cada uma é suficiente para aprender um conceito alvo.

Dadas as definições acima, os fundamentos do aprendizado multi-visão assumem que as visões são compatíveis e não-correlatas [Muslea et al., 2002]. Um problema

tem visões compatíveis se todas as instâncias são rotulados identicamente em cada visão. As visão são ditas não-correlatas se, dado um rótulo de qualquer instância, suas descrições em cada visão são independentes. Assim, para as visões serem compatíveis e não-correlatas, para qualquer instância $\langle e_{11}, e_{12}, ..., e_{1v}, l \rangle$, $e_{11}$ e $e_{12}$ são independentes, dado o rótulo $l$. A Figura 2.1 exemplifica a divisão de uma base de dados multi-visão.

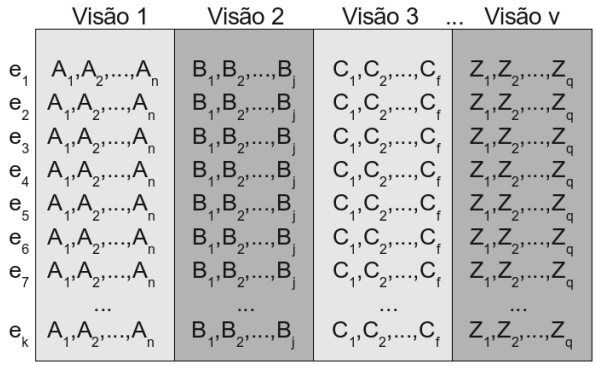



**Figura 2.1.** Divisão de uma base de dados em multi-visão.

### 2.2.2 Decisão dos Classificadores

Dada a divisão apresentada na Figura 2.1, em um contexto com $v$ classificadores e $m$ classes, cada classificador $\varphi_i$, onde $1 \leq i \leq v$, disponibiliza, como saída da sua classificação, uma lista de tamanho $m$, que é ordenada em ordem decrescente pelo valor da confiança de $\varphi$ em atribuir uma classe $c$ a uma instância $e$. Assim, uma classe $c_j$ que obteve uma maior confiança em classificar $e$ torna-se a decisão da classificação para essa instância. Como a métrica de confiança para cada classificador varia de acordo com o classificador utilizado, as confianças $\phi$ são normalizadas, para cada um dos $v$ classificadores em cada uma das $m$ classes, pertencem ao intervalo $[0, 1]$, tal que, $(\phi_{v_{i1}} + \phi_{v_{i2}} + ... + \phi_{v_{im}}) = 1$. A Equação 2.1 define a normalização das confianças atribuídas às $m$ classes para uma instância e.

$$\phi_i(e) = \frac{\phi_i}{\sum_{j=1}^{|m|} \phi_{ij}} \tag{2.1}$$

Com isso, no contexto da classificação por multi-visão (Seção 2.2), sejam $v$ classificadores $(\varphi_1, \varphi_2, ..., \varphi_v)$ em $v$ visões dos dados $(V_1, V_2, ..., V_v)$, a decisão dos classificadores para uma instância $e$ pode ser organizada em uma matriz de decisão (MD), ilustrada pela Figura 2.2.

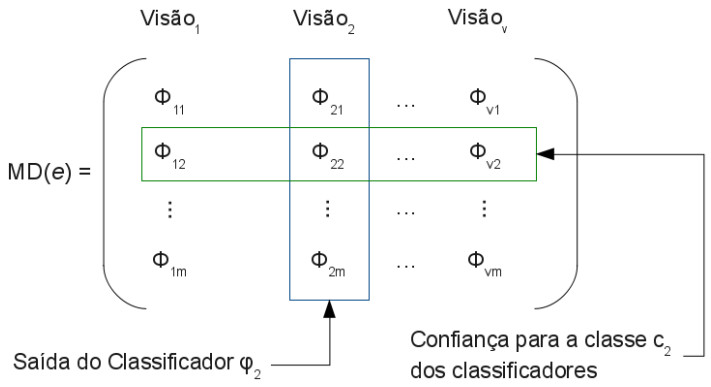



**Figura 2.2.** Matriz de decisão para uma instância $e$.

A matriz de decisão é utilizada por todos os métodos de combinação de classificadores apresentados nesta dissertação.

### 2.2.3 Trabalhos Relacionados

As seções anteriores revisaram os conceitos básicos de aprendizado multi-visão. Essa seção apresenta trabalhos relacionados, ressaltando os métodos de combinação mais utilizados. No quadro comparativo apresentado na Tabela 2.1 é possível identificar que a grande maioria dos trabalhos utilizam duas visões dos dados. Isso porque grande parte desses algoritmos são derivados do *Co-training* (aprendizado semi-supervisionado), que é um algoritmo que, em sua versão original, trabalha com dois classificadores Naïve Bayes cooperando para rotular as instâncias de uma base de dados. Os trabalhos que utilizam mais de duas visões, são, em sua grande maioria, derivados dos aprendizados supervisionado e não-supervisionado. Além disso, observe que grande parte deles se relaciona a classificação de texto.

Considere os dois primeiros trabalhos da Tabela 2.1, que utilizam o aprendizado supervisionado, e são mais pertinentes para esta dissertação. [Wang et al., 2010] desenvolveu um novo método de aprendizado multi-visão que leva em consideração uma única fonte de dados. Concretamente, para uma única fonte de dados, primeiro é feito um mapeamento em um espaço de características $M$ dimensional, por $M$ diferentes funções de *kernel*. Com isso, cada espaço de característica gerado é associado a um algoritmo de Regularização Discriminativa, que, sintetiza $M$ Regularizações Discriminativas em um único processo de aprendizado obtendo um novo *Multi-view Discriminative Regularization* (MVDR). No MVDR, cada Regularização Discriminativa pode ser tomada como uma visão. Ou seja, nesse trabalho, a partir do aprendizado regularizado[1], a

[1]É visto como um importante método para melhorar a performance de classificadores [Chen &

**Tabela 2.1.** Comparativo entre diferentes abordagens utilizando aprendizado multi-visão.

| **Autores** | **Visões** | **Tarefa resolvida** | **Algoritmo de Combinação** | **Tipo de aprendizado** | **Modelo** | **Aplicação** |
|---|---|---|---|---|---|---|
| [Wang et al., 2010] | $N$ visões | Classificação | *MVDR* | Supervisionado | Vetores | Texto e Vídeo |
| [Hinrichs et al., 2011] | $N$ visões | Classificação | *Multi-Kernel Learning* (MKL) | Supervisionado | Vetores | Base de dados de medicina |
| [Junior et al., 2010] | Três visões | Classificação | *PSO* | Supervisionado | Vetores | Texto e Rede social de vídeo |
| [Mihalcea, 2004] | Duas visões | Classificação | *Co-training* e *Self-training* | Semi-supervisionado | Vetores | Texto |
| [Hovelynck & Chidlovskii, 2010] | Duas visões | Classificação | *Mapping Co-Convergence* | Semi-Supervisionado | Vetores e Grafos | Texto e Rede Social |
| [Laguna & Lopes, 2009] | Duas visões | Classificação | *Co-training* | Semi-supervisionado | Vetores e Grafos | Texto |
| [Yu et al., 2008] | Duas visões | Classificação e agrupamento | *Co-training* | Semi-supervisionado | Vetores | Páginas da *Web* e imagens |
| [Christoudias et al., 2008] | Duas visões | Classificação e agrupamento | *Co-training* | Semi-supervisionado | Vetores | Áudio e vídeo |
| [Wang & Zhou, 2008] | Duas visões | Classificação | *Co-training* | Semi-supervisionado | Vetores | Páginas da *Web* |
| [Zhang et al., 2008] | Duas visões | Classificação | *Co-training e Self-training* | Não-supervisionado | Vetores | Movimento de objetos em vídeos |
| [Guo & Viktor, 2006] | multi-visões correlatas | Classificação | *Multi-view Relational Classification (MRC)* | Não-supervisionado | Banco de dados | Financeiras; biológicas; e base da ECML |
| [Brefeld et al., 2005] | Duas visões | Classificação | Derivação do *multi-view Hidden Markov (HM) perceptron* | Semi-supervisionado | Vetores | Texto |
| [Tong et al., 2005] | Duas visões | Classificação | *Co-training* | Semi-supervisionado e Não-supervisionado | Grafos | Páginas da *Web*; Imagens; e Imagens da *Web* |
| [Guo & Viktor, 2005] | Cinco visões | Classificação | *Multi-View Classification Algorithm* | Semi-supervisionado | Banco de dados | Banco de dados financeiro |
| [Denis et al., 2003] | Duas visões | Classificação | *Co-training* | Não-supervisionado | Vetores | Texto |
| [Blum & Mitchell, 1998] | Duas visões | Classificação | *Co-training* | Semi-supervisionado | Vetores | Páginas da *Web* |

Haykin, 2002].

base de dados é particionada por funções de *kernel* e $M$ visões dos dados são obtidas e, posteriormente classificadas, obtendo ao final um resultado único. Note que o MVDR é responsável por criar as visões a partir de um conjunto inicial de dados, e não segue o modelo tradicional do multi-visão. Nesse trabalho, o aprendizado multi-visão foi utilizado para resolver o problema da classificação multi-rótulo, i.e., quando uma instância pode pertencer a mais de uma classe ao mesmo tempo.

Similarmente, o trabalho de [Hinrichs et al., 2011] utiliza um algoritmo de *Multi-Kernel Learning* (MKL), onde uma única fonte de dados é particionada por diferentes funções de *kernel*, classificadas e combinadas pelo algoritmo MKL. Nesse trabalho, cuja aplicação é a classificação de imagens médicas, fica evidente a superioridade da utilização do aprendizado multi-visão ou multi-modalidade para classificação automática.

Diferentemente dos trabalhos de [Wang et al., 2010] e [Hinrichs et al., 2011], que são supervisionados, os trabalhos de [Guo & Viktor, 2006], [Brefeld et al., 2005] e [Guo & Viktor, 2005] que não são diretamente derivados do algoritmo *Co-training*, mas trabalham no contexto semi-supervisionado e, além disso, as visões dos dados já foram previamente separadas antes de serem aplicadas aos algoritmos de classificação. Além disso, note que nesses trabalhos, as visões foram separadas de maneira intuitiva, não havendo um estudo da relação dos atributos ou aplicação de alguma função ao espaço de dados para uma futura separação dos atributos em visões, como foi feito por [Wang et al., 2010].

Já os demais trabalhos apresentados na Tabela 2.1 utilizam o *Co-training*, que é um algoritmo que trabalha com apenas com apenas duas visões por vez, fazendo com que dois classificadores Naïve Bayes cooperem para rotular os dados em aprendizado semi-supervisionado, ou propõem variações desse algoritmo como o *Self-training*.

## 2.3 Métodos de Combinação de Classificadores

Essa seção descreve os principais métodos utilizados na literatura para combinação de classificadores, especialmente aqueles utilizados com comitês. Diversos trabalhos relacionados a combinação de classificadores são apresentados na Tabela 2.2.

Tecendo alguns comentários a respeito dos principais trabalhos apresentados na Tabela 2.2, especialmente àquele apresentado na primeira linha da Tabela 2.2, que é baseado em PSO. Portanto, em [Kausar et al., 2010] foi encontrado o trabalho mais similar a esta dissertação, onde um PSO é utilizado para ponderar um voto da maioria em um comitê de classificadores. Porém, diversos fatores apontam diferenças entre os trabalhos. Além do fato de focarmos em aprendizado multi-visão e não com comitê de

**Tabela 2.2.** Comparativo entre diferentes trabalhos no contexto da combinação de comitês de classificadores.

| **Autores** | **Tarefa resolvida** | **Quantidade de Classificadores** | **Método de Combinação** | **Aplicação** |
|---|---|---|---|---|
| [Kausar et al., 2010] | Classificação | 4 | PSO com Voto da Maioria em um comitê de classificadores | Repositório UCI |
| [Tsai et al., 2011] | Classificação | 3 | Voto da Maioria | Base de Transações Econômicas |
| [Quost et al., 2011] | Classificação | 4 | Voto da Maioria e Teoria de Dempster-Shafer | Oitos base de dados |
| [Perez et al., 2011] | Classificação | 2 | *Borda Count* | Reconhecimento de face em imagens |
| [Langbehn et al., 2010] | Classificação | 1 | *Borda Count* | Rede social de vídeo |
| [Sun, 2010] | Classificação | 2 | Voto da Maioria Ponderado | Nove contextos distintos |
| [Huynh et al., 2010] | Classificação | 3 - 6 | Combinação ponderada usando a Teoria de Dempster-Shafer | Texto |
| [Re & Valentini, 2010] | Classificação | 1 | Fusão de Kernels e Voto da Maioria Ponderado | Tráfego em Rede (Ataque) |
| [Perdisci et al., 2009] | Classificação e Agrupamento | 3 - 11 | Variações do Voto da Maioria | Tráfego em Rede (Ataque) |
| [Bi et al., 2008] | Classificação | 2 - 8 | Voto da Maioria ponderado | Diversos contextos não descritos |
| [Bi et al., 2008] | Classificação | 2 - 8 | Voto da Maioria e Teoria de Dempster-Shafer | Diversos contextos não descritos |
| [Guo et al., 2006] | Classificação | 5 | Voto da Maioria e Teoria de Dempster-Shafer | Diversos contextos não descritos |
| [Bi et al., 2006] | Classificação | 4 | Voto da Maioria e Teoria de Dempster-Shafer | Texto |
| [Bell et al., 2005] | Classificação | 5 | Teoria de Dempster-Shafer | Texto |
| [Al-Ani & Deriche, 2002] | Classificação | 5 - 9 | Teoria de Dempster-Shafer | Texto, imagens e Identificação de fala |

classificadores, a tarefa adotada neste trabalho difere em dois pontos principais: (*i*) o voto da maioria é ponderado, i.e., os pesos gerados pelo PSO são atribuídos à confiança

de cada classificador em classificar uma instância $e$ em uma classe $c$. Após isso, cada classificador tem uma nova confiança ponderada, e o voto da maioria é feito, ou seja, para os classificadores que concordam com relação à classe de $e$, as novas confianças ponderadas são somadas e então, a classe a ser atribuída à instância $e$ será aquela que obteve o maior valor após o somatório das confianças para as respectivas classes. Note que esse trabalho leva em consideração somente a primeira classe prevista por cada classificador, que é a confiança do classificador para uma dada classe $c$; $(ii)$ já nesta dissertação, além de se ponderar as decisões dos classificadores em cada visão dos dados, é possível levar em consideração todo o *ranking* de classes em cada visão. Em [Kausar et al., 2010] os autores ponderam somente o voto da maioria com relação à classe prevista. Com a abordagem aqui proposta, é possível reforçar ou não a predição, caso o classificador seja melhor ou não em uma determinada visão.

Note que o trabalho supracitado não leva em consideração a força de cada classificador independentemente. Esse pode ser um ponto fraco pois, ao se trabalhar com classificadores distintos, sabe-se, por definição, que enquanto um classificador é bom em um dado contexto, outro pode não ter o mesmo desempenho. Assim, aqui neste trabalho os classificadores são ponderados de acordo com sua força classificatória em cada visão de uma base de dados. Ou seja, são atribuídos pesos às confianças de cada classificador, em sua respectiva visão, com relação à classe prevista para uma dada instância $e$. Além disso, para combinação das visões, tenta-se capturar o erro inerente a cada classificador, levando em consideração todo o *ranking* de classes disponibilizado pelos classificadores para cada instância.

Os demais trabalhos apresentados na Tabela 2.2 utilizam, no contexto de comitês, os métodos de combinação de classificadores utilizados nesta dissertação, em comparação com o método proposto, e que são amplamente estudados e consolidados por diversos trabalhos na literatura, os quais são descritos a seguir.

### 2.3.1 Voto da Maioria

O voto da maioria é um método de combinação simples, que tem sido muito estudado por diversos pesquisadores [Ruta & Gabrys, 2005] [Brown & Kuncheva, 2010]. Em um voto da maioria, cada classificador tem a mesma importância. Assim, para resolver o problema da combinação de $v$ classificadores, onde cada classificador $\varphi_v$ atribui uma classe $c$ a uma instância $e$, temos: seja $m$ a quantidade de classes de uma dado problema de classificação e $num(e_c)$ o número de classificadores que atribuíram a classe $c$ a uma instância $e$. Com isso, a classe $t$ a ser atribuída a uma instância $e$, pelo voto da maioria,

é definida pela Equação 2.2.

$$t(e) = argmax_v(num(e_c)) \tag{2.2}$$

### 2.3.2 Borda Count

No contexto da teoria da decisão em grupo, um mapeamento a partir de um conjunto individual de *rankings*, para um *ranking* combinado, é referenciado como uma função de consenso de grupo. Uma função de consenso de grupo útil é o *Borda Count* [Black, 1958]. A magnitude do *Borda Count*, para cada classe $c$, mensura a força com que os classificadores concordam que uma instância $e$, de entrada, pertença a uma classe $c$. Para um problema com duas classes, o *Borda Count* é equivalente a um simples voto da maioria.

A função de *Borda Count* assume independência aditiva entre as contribuições dos classificadores individuais. Usando esse método, um classificador é redundante se ele sempre reforça os erros cometidos pelos outros, ou seja, se todas as classes escolhidas estão sempre contidas na escolha de algum outro classificador.

O *Borda Count* para uma classe $c$ é a soma do número de instâncias rotuladas por cada classificador para a classe $c$. O *ranking* de consenso é dado pelo arranjo das classes de modo que sua contagem fique em ordem decrescente.

O método de *Borda Count* é simples de implementar e não necessita de treinamento. No entanto, ele não leva em conta as diferenças nas capacidades de classificação individual. Todos os classificadores são tratados igualmente, o que pode não ser preferível, ainda mais quando se sabe que certos classificadores têm uma maior probabilidade de estarem corretos com relação a outros.

**Tabela 2.3.** Exemplo fictício- matrizes de decisão para três instâncias.

| **Classes** | **Matrizes de Decisão** | | | | | |
|---|---|---|---|---|---|---|
| | **$e_1$** | | **$e_2$** | | **$e_3$** | |
| | $Visão_1$ | $Visão_2$ | $Visão_1$ | $Visão_2$ | $Visão_1$ | $Visão_2$ |
| $c_1$ | **0.724** | 0.060 | **0.900** | **0.500** | **0.800** | 0.050 |
| $c_2$ | 0.184 | **0.688** | 0.030 | 0.300 | 0.090 | **0.600** |
| $c_3$ | 0.050 | 0.044 | 0.060 | 0.050 | 0.070 | 0.250 |
| $c_4$ | 0.042 | 0.208 | 0.010 | 0.150 | 0.040 | 0.100 |

Dadas as três matrizes de decisão apresentadas na Tabela 2.3, o método de *Borda Count* funciona da seguinte forma. Primeiro construímos um *ranking* para cada classe com todas as instâncias. No caso desse exemplo são três instâncias, classificadas em duas visões, para cada uma é gerado um *ranking* (Tabela 2.4). O valor de cada instância

no *ranking* de cada visão depende da confiança que foi atribuída pelo classificador à classe $c_i$, i.e., a menor confiança recebe a posição 1 no *ranking* e a maior a posição $n$, que representa a $n-$ésima instância. Computando o *Borda Count* para as classes desse exemplo temos, pela Tabela 2.4, os *rankings* para as respectivas visões e suas pontuações nas referidas classes. O valor da pontuação para cada classe é obtido pela soma das posições de cada *ranking* para a instância $e_i$, em cada visão. Dessa forma, pela Tabela 2.4, observamos que para a instância $e_1$, com relação à classe $c_1$, a pontuação é dada pela soma de suas posições nos *rankings* relativos às visões 1 e 2, obtendo pontuação de valor 3.

**Tabela 2.4.** Exemplo - quadro de pontuações para categorização de instâncias (V = Visão, Pont. = Pontuação e Inst. = Instâncias).

| **Inst.** | **Classes** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathbf{c_1}$ | | | $\mathbf{c_2}$ | | | $\mathbf{c_3}$ | | | $\mathbf{c_4}$ | | | **Categoria atribuída** |
| | $V_1$ | $V_2$ | **Pont.** | $V_1$ | $V_2$ | **Pont.** | $V_1$ | $V_2$ | **Pont.** | $V_1$ | $V_2$ | **Pont.** | |
| $e_1$ | 1 | 2 | **3** | 3 | 3 | **6** | 1 | 1 | **2** | 3 | 3 | **6** | $c_2$ |
| $e_2$ | 3 | 3 | **6** | 1 | 1 | **2** | 2 | 2 | **4** | 1 | 2 | **3** | $c_1$ |
| $e_3$ | 2 | 1 | **3** | 2 | 2 | **4** | 3 | 3 | **6** | 2 | 1 | **3** | $c_3$ |

Para categorizar as instâncias pela combinação das visões, observa-se a classe que obteve a maior pontuação. No caso de empate, como ocorreu nesse exemplo, para a instância $e_1$ com relação às classes $c_2$ e $c_4$, a decisão é feita pela visão que obteve maior valor no *ranking*, no caso de um novo empate, como nesse exemplo, onde cada visão obteve valor igual a 3, atribui-se, à instância, aquela classe com maior confiança, de acordo com a matriz de decisão. A última coluna da Tabela 2.4 representa a classe atribuída às respectivas instâncias pelo algoritmo de *Borda Count*.

### 2.3.3 Teoria da Evidência de Dempster-Shafer

A teoria da evidência de Dempster-Shafer (DS) [Shafer, 1976] é uma forma de representar a incerteza do conhecimento. Essa teoria é vista como uma generalização da teoria probabilística Bayesiana, por prover uma coerente representação para a ignorância (falta de evidência). A teoria de DS é adaptada a uma série de atividades de tomada de decisão, pois formula um processo de raciocínio como elementos de prova e hipóteses. Esse método foi escolhido por representar uma poderosa ferramenta para combinar medidas de evidências e ser amplamente utilizado na literatura [Al-Ani & Deriche, 2002] [Bi et al., 2006] [Bi et al., 2008] [Gromisz & Zadrożny, 2010] [Huynh et al., 2010].

A teoria de DS é formulada em termos de funções de evidência e ignorância. Essas funções podem ser funções de massa, funções de confiança e funções de plausibilidade

[Shafer, 1976]. Uma definição formal dessa teoria pode ser encontrada em [Bell et al., 2005].

**Definição 1** *Seja $\Theta$ um conjunto finito e não-vazio de possíveis hipóteses, chamado* ***quadro de discernimento****. Seja $[0,1]$ um conjunto de valores numéricos, e $m$ uma função de mapeamento $2^{\Theta} \rightarrow [0,1]$. Com isso, $m$ é chamada de* ***função de massa*** *se satisfaz:*

$$m(\emptyset) = 0, \qquad \sum_{X \subseteq \Theta} m(X) = 1 \tag{2.3}$$

*Uma* ***função de massa*** *é uma atribuição básica de probabilidade (abp) para todos subconjuntos $X$ de $\Theta$. Um subconjunto $A$ de um quadro $\Theta$ é dito focal de uma* ***função de massa*** *$m$ em $\Theta$ se $m(A) > 0$, e $A$ é chamado de único se ele é um subconjunto de um elemento focal. Tomando-se a matriz de decisão, que é a representação geral das saídas dos classificadores utilizados nessa dissertação (Subseção 2.2.2), a seguir define-se uma* ***função de massa****.*

**Definição 2** *Seja $C$ um* ***quadro de discernimento****, onde cada escolha $C_i \in C$ é uma proposição que a instância $e$ é classificada na classe $c_i$. Seja $\phi(e) = \{\phi_1, \phi_2, ..., \phi_{|C|}\}$ uma lista de confianças, onde no caso desta dissertação cada $\phi_i$ corresponde à saída de um classificador em uma visão. Uma* ***função de massa*** *é definida como um mapeamento $m$: $2^C \rightarrow [0,1]$, i.e., um abp para $c_i \in C$ para $1 \leq i \leq |C|$, como segue:*

$$m(\{c_i\}) = \frac{\phi_i}{\sum_{j=1}^{|C|} \phi_j} \tag{2.4}$$

*Essa* ***função de massa*** *expressa os graus de liberdade de escolha de uma classe para uma instância. Pela Equação 2.4, pode-se reescrever $\phi(e)$ como $\phi(e) = \{m(\{c_1\}), m(\{c_2\}), ..., m(\{c_{|C|}\})\}$, referido como uma lista de decisões - uma parte da evidência.*

**Definição 3** *Sejam $m_1$ e $m_2$ duas funções de massa em um* ***quadro de discernimento*** *$\Theta$, e para qualquer subconjunto $A \subseteq \Theta$, a soma ortogonal $\oplus$ de duas funções de massa $A$ é definida como:*

$$(m_1 \oplus m_2)(A) = \frac{\sum_{X \cap Y} m_1(X) \times m_2(Y)}{N} \tag{2.5}$$

*onde* $N = 1 - \sum_{X \cap Y = \emptyset} m_1(X) \times m_2(Y)$ *e* $K = \frac{1}{N}$ *é chamado de normalização constante da soma ortogonal* $m_1 \oplus m_2$. *Essa soma é comumente chamada de regra combinatória de Dempster. Existem duas condições para assegurar a existência da soma ortogonal:*

1. $N \neq 0$ - *Se* $N = 0$, *então duas funções de massa* $m_1$ *e* $m_2$ *são totalmente contraditórias.*

2. *Duas funções de massa devem ser totalmente independentes uma da outra, ou seja, representarem opiniões independentes relativas a um mesmo* ***quadro de discernimento****.*

A seguir é definida uma função de massa tripla, que é uma estrutura que particiona uma lista de decisões para uma instância $e$ rotulada por um classificador $\varphi$, e é utilizada para combinação de $n$ classificadores quaisquer, de acordo com a matriz de decisão.

#### 2.3.3.1 Estrutura Tripla

Em uma estrutura tripla, uma lista de decisões $\phi(e)$ é particionada em três subconjuntos. A seguir são apresentadas definições a respeito da formação de uma estrutura tripla.

**Definição 4** *Seja* $C$ *um* ***quadro de discernimento*** *e* $\phi(e) = \{m(\{c_1\}), m(\{c_2\}), ..., m(\{c_{|C|}\})\}$ *onde* $\phi(e) \geq 2$, *uma tripla é definida como uma expressão na forma* $Y = \langle A_1, A_2, A_3 \rangle$, *onde* $A_1, A_2 \subseteq C$ *são únicos, e* $A_3$ *é todo o conjunto* $C$ *restante.*

**Definição 5** *Dado um* ***quadro de discernimento*** $C$, *uma* ***função de massa*** $m$ *é chamada de* ***função de massa*** *tripla se ela não tem outros elementos focais além de dois únicos* $\{x\}, \{y\}$ *e todo o conjunto* $C$, *onde* $x, y \in C$ *tal como a Equação 2.6.*

$$m(\{x\}) + m(\{y\}) + m(\{C\}) = 1 \tag{2.6}$$

*Para obter uma* ***função de massa*** *tripla, uma operação focal é definida em um contexto geral, chamada de regra excepcional. As operações focais são definidas pelas relações enumerativas da subseção seguinte.*

**Definição 6** *Seja $C$ um quadro de discernimento, e $m$ uma* ***função de massa*** *com elementos focais $\{x_1\}, \{x_2\}, ..., \{x_n\} \subseteq C$, $n \geq 2$, e $n \leq |C|$, então uma regra excepcional é definida como uma operação focal $\sigma$ em $m$, denotada por $m^\sigma$ da seguinte forma: $m^\sigma(\{u\}) + m^\sigma(\{v\}) + m^\sigma(\{\Theta\}) = 1$*

*Onde:*

$$\{u\} = argmax(\{m(\{x_1\}), m(\{x_2\}), ..., m(\{x_n\})\})$$

$$\{v\} = argmax(\{m(\{x_1\})|x \in \{x_1, x_2, ..., x_n\} - \{u\}\})$$

*Ignorância*$(A) = m^\sigma(C) = 1 - m^\sigma(\{u\}) - m^\sigma(\{v\})$

*Assim, para um problema com $v$ classificadores, temos a seguinte equação*

$$\varphi_i(d) = \{m^\sigma(\{u\}), m^\sigma(\{v\}), m^\sigma(\{C\})\}, 1 \leq i \leq M \tag{2.7}$$

*e $A_1 = \{u\}, A_2 = \{v\}$ e $A_3 = \{C\}$.*

#### 2.3.3.2 Computando duas Funções de Massa Tripla

Baseado no número de decisões únicas, uma tripla pode ser referenciada como uma estrutura de dois pontos focais, chamada **função de massa** associada, uma **função de massa** de dois pontos.

Supondo-se duas triplas $\langle\{x_1\}, \{y_1\}, C\rangle$ e $\langle\{x_2\}, \{y_2\}, C\rangle$ onde $\{x_i\} \subseteq C, \{y_i\} \subseteq C$ $(i = 1, 2)$, e as funções de massa associadas $m_1$ e $m_2$. As relações enumerativas entre quaisquer dois pares de elementos focais $x_1$, $y_1$ e $x_2$, $y_2$ são demonstradas abaixo.

1. Se $\{x_1\} = \{x_2\}$ e $\{y_1\} = \{y_2\}$, então $\{x_1\} \cap \{y_2\} = \emptyset$ e $\{y_1\} \cap \{x_2\} = \emptyset$, e a combinação de duas funções triplas envolve três diferentes elementos focais (dois pontos focais iguais).

2. Se $\{x_1\} = \{x_2\}$ e $\{y_1\} \neq \{y_2\}$, então $\{x_1\} \cap \{y_2\} = \emptyset$ e $\{y_1\} \cap \{x_2\} = \emptyset$ e $\{y_1\} \cap \{y_2\} = \emptyset$ ou se $\{x_1\} \neq \{x_2\}$ e $\{y_1\} = \{y_2\}$, então $\{x_1\} \cap \{y_2\} = \emptyset$ e $\{x_2\} \cap \{y_1\} = \emptyset$ e $\{x_1\} \cap \{x_2\} = \emptyset$, então a combinação de duas funções triplas envolve quadro diferentes elementos focais (um ponto focal igual).

3. Se $\{x_1\} \neq \{x_2\}$ e $\{y_1\} \neq \{y_2\}$ e $\{x_1\} \neq \{y_2\} = \emptyset$ e $\{y_1\} \neq \{x_2\} = \emptyset$, então $\{x_1\} \cap \{x_2\} = \emptyset$ e $\{y_1\} \cap \{y_2\} = \emptyset$ e $\{x_1\} \cap \{y_2\} = \emptyset$ e $\{y_1\} \cap \{x_2\} = \emptyset$, então a combinação de duas funções triplas envolve cinco diferentes elementos focais (pontos focais totalmente diferentes).

Assim, a partir dessas relações enumerativas, é necessário combinar as funções de massa de acordo com cada relação. Em sequência, são mostrados os teoremas respectivos a cada uma das três relações enumerativas.

#### 2.3.3.3 Dois Pontos Focais Iguais

Supondo-se duas funções de massa tripa $m_1$ e $m_2$, com dois pares de dois elementos focais $\{x_1\}$, $\{y_1\}$ e $\{x_2\}$, $\{y_2\}$ e $x_1 = x_2$, $y_1 = y_2$ $(x_1 \neq y_1)$, têm-se:

$$m_1(\{x_1\}) + m_1(\{y_1\}) + m_1(C) = 1$$

$$m_2(\{x_2\}) + m_2(\{y_2\}) + m_2(C) = 1$$

Primeiro, é necessário mostrar que, em qualquer condição, a combinação de $m_1 \oplus m_2$ existe. Posteriormente será mostrado como computar sua combinação.

**Teorema 1** *Seja $C$ um **quadro de discernimento**, seja $m_1$ e $m_2$ duas funções de massa tripla em $C$, e também seja $\{x\}$, $\{y\}$ e $\{x\}$, $\{y\}$ $(x \neq y)$ serem dois pares de dois-pontos elementos focais, com a condição de,*

$$m_1(\{x\}) + m_1(\{y\}) + m_1(C) = 1, 0 \leq m_1(\{x\}), m_1(\{y\}), m_1(C) \leq 1$$

$$m_2(\{x\}) + m_2(\{y\}) + m_2(C) = 1, 0 \leq m_2(\{x\}), m_2(\{y\}), m_2(C) \leq 1$$

*Então,*

$$K = 1 - m_1(\{x\}) \times m_2(\{y\}) - m_1(\{y\}) \times m_2(\{x\})$$

*e $m_1$, $m_2$ são combináveis se e somente se*

$$0 \leq m_1(\{x\}) \times m_2(\{y\}) + m_1(\{y\}) \times m_2(\{x\}) < 1$$

De acordo com o Teorema 1, é possível obter a fórmula de combinação de duas funções de massa tripla $m_1 \oplus m_2$, de acordo com as Equações 2.8, 2.9 e 2.10.

$$m_1 \oplus m_2(\{x\}) = K(m_1(\{x\}) \times m_2(\{x\}) + m_1(\{x\}) \times m_2(C)) + m_1(C) \times m_2(\{x\}) \quad (2.8)$$

$$m_1 \oplus m_2(\{y\}) = K(m_1(\{y\}) \times m_2(\{y\}) + m_1(\{y\}) \times m_2(C)) + m_1(C) \times m_2(\{y\}) \quad (2.9)$$

$$m_1 \oplus m_2(C) = K(m_1(\Theta) \times m_2(C) \tag{2.10}$$

#### 2.3.3.4 Um Ponto Focal Igual

Dadas duas funções de massa tripla $m_1$ e $m_2$, um elemento focal em uma tripla é igual a um em outra tripla. O Teorema 2 revela que essas duas funções de massa são combináveis.

**Teorema 2** *Seja $C$ um **quadro de discernimento**, $m_1$ e $m_2$ duas funções de massa tripla em $C$, e também seja $\{x\}$, $\{y\}$ e $\{x\}$, $\{z\}$ $(y \neq z)$ serem dois pares de elementos focais conforme a condição seguinte:*

$$m_1(\{x\}) + m_1(\{y\}) + m_1(\Theta) = 1, 0 \leq m_1(\{x\}), m_1(\{y\}), m_1(\Theta) \leq 1$$

$$m_2(\{x\}) + m_2(\{z\}) + m_2(\Theta) = 1, 0 \leq m_2(\{x\}), m_2(\{y\}), m_2(\Theta) \leq 1$$

*Então,*

$$K = 1 - m_1(\{x\}) \times m_2(\{y\}) - m_1(\{y\}) \times m_2(\{z\}) - m_1(\{x\}) \times m_2(\{z\})$$

*$m_1$, $m_2$ são combináveis se e somente se a seguinte restrição é mantida:*

$$m_1(\{x\}) \times m_2(\{y\}) + m_1(\{y\}) \times m_2(\{z\}) + m_1(\{x\}) \times m_2(\{z\}) < 1$$

Pelo Teorema 2 e a soma ortogonal, uma nova **função de massa** pode ser obtida a partir de duas funções de massa tripla. As fórmulas gerais para se computar as novas funções de massa são dadas a seguir:

$$m_1 \oplus m_2(\{x\}) = K(m_1(\{x\}) \times m_2(\{x\}) + m_1(\{x\}) \times m_2(C)) + m_1(C) \times m_2(\{x\}) \tag{2.11}$$

$$m_1 \oplus m_2(\{y\}) = K(m_1(\{y\}) \times m_2(C) \tag{2.12}$$

$$m_1 \oplus m_2(\{z\}) = K(m_1(\Theta) \times m_2(\{z\}) \tag{2.13}$$

$$m_1 \oplus m_2(C) = K(m_1(C) \times m_2(C) \tag{2.14}$$

Onde,

$$K = 1 - \sum_{X \cap Y = \emptyset} m_1(X) \times m_2(Y) = 1 - m_1(X) \times m_2(Z) - m_1(Y) \times m_2(X) \tag{2.15}$$

#### 2.3.3.5 Pontos Focais Totalmente Diferentes

Nesse caso, não existe nenhum ponto focal em comum. Como indicado anteriormente, a combinação de tais funções de massa tripla irá envolver cinco diferentes elementos focais. Primeiro, será fornecido um teorema para garantir que essas duas funções de massa tripla são combináveis.

**Teorema 3** *Seja $\Theta$ um **quadro de discernimento**, sejam $m_1$ e $m_2$ duas funções de massa tripla, e $\{x\}, \{y\}$ e $\{u\}, \{v\}$ $(x \neq y, x \neq u, y \neq v)$ serem dois pares de elementos focais, obedecendo as seguintes condições:*

$$m_1(\{x\}) + m_1(\{y\}) + m_1(\Theta) = 1, 0 \leq m_1(\{x\}), m_1(\{y\}), m_1(\Theta) \leq 1$$

$$m_2(\{u\}) + m_2(\{v\}) + m_2(\Theta) = 1, 0 \leq m_2(\{u\}), m_2(\{v\}), m_2(\Theta) \leq 1$$

*Então,*

$$K = 1 - m_1(\{x\}) \times m_2(\{u\}) - m_1(\{x\}) \times m_2(\{v\}) - m_1(\{y\}) \times m_2(\{u\}) - m_1(\{y\}) \times m_2(\{v\})$$

*e $m_1$, $m_2$ são combináveis se e somente se a seguinte restrição é mantida:*

$$0 \leq m_1(\{x\}) \times m_2(\{u\}) + m_1(\{y\}) \times m_2(\{v\}) + m_1(\{y\}) \times m_2(\{u\}) m_1(\{y\}) \times m_2(\{v\}) < 1$$

Dado esse teorema, é preciso saber como a combinação das funções de massa pode ser feita. Suponha a seguinte expressão:

$$m_1 \oplus m_2(\{x\}) = K(m_1(\{x\}) \times m_2(\Theta) \tag{2.16}$$

$$m_1 \oplus m_2(\{y\}) = K(m_1(\{y\}) \times m_2(\Theta) \tag{2.17}$$

$$m_1 \oplus m_2(\{u\}) = K(m_1(\Theta) \times m_2(\{u\}) \tag{2.18}$$

$$m_1 \oplus m_2(\{v\}) = K(m_1(\Theta) \times m_2(\{v\}) \tag{2.19}$$

Onde,

$$\begin{gathered} K = 1 - \sum_{X \cap Y = \emptyset} m_1(X) \times m_2(Y) = 1 - m_1(x) \times m_2(\{u\}) - \\ m_1(\{x\}) \times m_2(\{v\}) - m_1(\{y\}) \times m_2(\{u\}) - m_1(\{y\}) \times m_2(\{v\}) \end{gathered} \tag{2.20}$$

A combinação de $m_1$, $m_2$ não é mais uma **função de massa** tripla, pois envolve cinco elementos focais $\{x\}, \{y\}, \{u\}, \{v\}, C$, portanto, mais combinações com funções triplas são inválidas nesse contexto. Para se obter uma nova **função de massa** tripla, há a necessidade de se aplicar a regra pendente para combinar os resultados. Mais especificamente, pela Definição 6, é possível obter uma nova função $(m_1 \oplus m_2)^\sigma$ da seguinte forma:

$$(m_1 \oplus m_2)^\sigma(\{X'\}) + (m_1 \oplus m_2)^\sigma(\{Y'\}) + (m_1 \oplus m_2)^\sigma(C) = 1$$

Então, para um elemento focal $\{X'\}$ tem-se

$$(m_1 \oplus m_2)^\sigma(\{X'\}) = f(Y'), \tag{2.21}$$

onde $\{X'\} = argmax(f(x), f(y), f(u), f(v))$.
Para um elemento focal $\{X'\}$ têm-se

$$(m_1 \oplus m_2)^\sigma(\{Y'\}) = f(Y'), \tag{2.22}$$

onde $\{Y'\} = argmax(\{f(t)|t \in (\{x, y, u, v\} - \{X'\})\})$.
Finalmente, para um elemento focal $C$, tem-se

$$(m_1 \oplus m_2)^\sigma(C) = 1 - f(X') - f(Y'). \tag{2.23}$$

Nas definições e teoremas anteriores, foi validado como duas funções de massa tripla quaisquer são combinadas, além de serem estabelecidas as equações de combi-

nação dessas funções. Por repetitividade, a aplicação da regra excepcional em cada passo da combinação de duas funções de massa tripla, o resultado pode ser transformado em uma nova **função de massa** tripla. Supondo-se $M$ funções de massa tripla $m_1, m_2, ..., m_M$, as quais podem ser combinadas em qualquer ordem devido à regra de Dempster, tendo essa as propriedades comutativa e associativa. A Equação 2.24 é uma soma ortogonal para combinar qualquer número de funções de massa tripla e a decisão final é a seleção da confiança máxima para todas as classes.

Para estabelecer a classe a que pertence uma dada instância é necessário a utilização de um limiar para ignorância, ou seja, a massa alocada para todo o conjunto (**Quadro de discernimento**) não deve ser maior do que esse limiar. Conforme definido em [Bell et al., 2005] o limiar pode ser definido como 10% de ignorância e esse valor é adotado aqui neste trabalho. O Algoritmo 1 mostra como funções de massa tripla são combinadas.

$$m = m^1 \oplus m^2 \oplus ... \oplus m^M = [...[[m^1 \oplus m^2] \oplus ... \oplus m^M]] \tag{2.24}$$

**Algoritmo 1:** Combinando múltiplas funções de massa tripla

```
Seja T um conjunto de funções de massa tripla
ct : o resultado da combinação de funções de massa tripla
ct ← t' ∈ T
para cada t ∈ T {t'} faça
    se dois focais iguais em t e ct então
        ct ← ct ⊕ t                    /* Combinar pelas Equações 2.8 - 2.10 */
    senão se um focal igual em t e ct então
        ct ← ct ⊕ t                    /* Combinar pelas Equações 2.11 - 2.13 */
        ct ← ct^σ   /* Transforma em uma nova função de massa tripla pelas Equações 2.11 - 2.15
        */
    senão
        ct ← ct ⊕ t                    /* Combinar pelas Equações 2.16 - 2.20 */
        ct ← ct^σ   /* Transforma em uma nova função de massa tripla pelas Equações 2.21 - 2.23
        */
retorna ct
```

Similarmente, é possível considerar quatro e cinco pontos focais em termos de quartetos e quintetos. Os quartetos e quintetos são conceitualmente simples e eles têm adicionado propriedades que podem ser utilizadas para lidar com decisões mais separadas a partir de listas ordenadas de decisões (*ranking* de classes disponibilizado por cada classificador). Maiores definições com relação aos quartetos e quintetos podem ser encontradas em [Bi et al., 2008].

Para um maior entendimento do algoritmo baseado na teoria de Dempster-Shafer, um exemplo simples, para computação de funções de massa tripla, é apresentado a seguir. Na Tabela 2.5 as classes destacadas foram aquelas previstas pelos classificadores para categorizar a instância $e$ em suas respectivas visões.

**Tabela 2.5.** Exemplo de aplicação - matriz de decisão para uma instância $e$.

| | $e$ | |
|---|---|---|
| Classes | Visão$_1$ | Visão$_2$ |
| $\mathbf{C_1}$ | **0.724** | 0.060 |
| $\mathbf{C_2}$ | 0.184 | **0.688** |
| $C_3$ | 0.050 | 0.044 |
| $C_4$ | 0.042 | 0.208 |

Através da matriz de decisão disposta na Tabela 2.5, são extraídas duas triplas $\langle A_1, A_2, C\rangle$ e $\langle B_1, B_2, C\rangle$, que representam os resultados correspondentes às visões 1 e 2, i.e.,

$$\langle A_1, A_2, C\rangle = \langle\{c_1\}, \{c_2\}, \{c_3, c_4\}\rangle$$

$$\langle B_1, B_2, C\rangle = \langle\{c_2\}, \{c_4\}, \{c_1, c_3\}\rangle,$$

ou seja, a combinação dessas duas visões envolve quatro diferentes elementos focais $\langle\{c_1\}, \{c_2\}, \{c_4\}, C\rangle$.

Assim, existem três possibilidades para categorizar o resultado combinado, que são: $A_1, A_2, B_2$. Com isso, de acordo com a matriz de decisão apresentada na Tabela 2.5, obtém-se duas funções de massa, uma para cada visão:

- Visão$_1$

$$m_1(A_1) = m_1(\{c_1\}) = 0.724$$

$$m_1(A_2) = m_1(\{c_2\}) = 0.184$$

$$m_1(C) = 0.092$$

- Visão$_2$

$$m_2(B_1) = m_2(\{c_2\}) = 0.688$$

$$m_2(B_2) = m_2(\{c_4\}) = 0.208$$

$$m_2(C) = 0.104$$

Como pode ser observado, nesse exemplo, existe apenas um ponto focal igual, com isso, a computação dessas duas funções de massa é regida pelo Teorema 2. Dessa forma, a soma ortogonal dessas duas funções de massa é feita pelas Equações 2.11 a 2.14.

Após isso, pelas respectivas equações, têm-se:

$$(m_1 \oplus m_2)(\{c_1\}) = 0.24$$

$$(m_1 \oplus m_2)(\{c_2\}) = 0.67$$

$$(m_1 \oplus m_2)(\{c_4\}) = 0.06$$

$$(m_1 \oplus m_2)(C) = 0.03$$

Assim, a **função de massa tripla associada** é obtida pelo maior valor da soma ortogonal para as classes, nesse caso é:

$$m(\{c_2\}) = 0.67$$

$$m(\{c_1\}) = 0.24$$

$$m(C) = 1 - 0.67 - 0.24 = 0.09$$

Após essas computações e através do limiar estabelecido para ignorância, que, aqui nesse trabalho, conforme já definido, foi estabelecido em 0.1, ou seja, a massa alocada para todo o conjunto restante é de no máximo 10%. Logo, como $m(C) \leq 0.1$, a classe atribuída à instância $e$, desse exemplo, foi $c_2$, pois essa classe obteve a confiança máxima para todas as classes. Caso $m(C)$ tivesse obtido um valor maior que o limiar definido, a classe que categorizaria a instância $e$ seria $c_1$, pois ela obteve a segunda maior confiança após a computação da função de massa associada.

Extendendo-se esse exemplo para um problema com mais de duas visões, por definição, a função de massa associada seria computada com a função de massa advinda dessa terceira visão e a classe final para a instância $e$ seria obtida da mesma forma como apresentado acima, observando-se a quantidade de pontos focais, como apresentado no Algoritmo 1.

# Capítulo 3

# Algoritmo de Otimização por Nuvem de Partículas

A utilização de algoritmos técnicas de Inteligencia Computacional (IC) para aprendizado e otimização de sistemas é uma área de pesquisa emergente e abrange um campo bastante vasto e multidisciplinar. Entre esses algoritmos, destacam-se, inicialmente, os de computação evolucionária, como algoritmos genéticos e programação genética [Banzhaf et al., 1998], que são motivados pela teoria da evolução.

As pesquisas ligadas à Computação Evolutiva continuam em grande ascensão, e os recentes estudos ligados a essa área levaram à criação de novos algoritmos (meta-heurísticas), como aqueles baseados em Inteligência de Enxames. Entre eles, podemos citar os algoritmos de Colônia de Formigas [Dorigo & Blum, 2005] [Wang & Cheng, 2010] e Nuvem de Partículas (PSO) [Kennedy et al., 2001] [Ng et al., 2009] [Rani & Deepa, 2010] [Tang et al., 2010], que têm uma forte motivação biológica.

Resultados promissores têm sido alcançados através da utilização de técnicas de IC, tanto no meio acadêmico quanto no campo comercial, devido às várias pesquisas nas mais diversas áreas do conhecimento. Esse sucesso deve-se, principalmente, às características inerentes aos algoritmos inteligentes, tais como capacidade de aprendizado, inferência, adaptação, dedução, reconhecimento de padrões, entre outras. Este trabalho explora a técnica de nuvens de partículas, descrita nas seções seguintes.

## 3.1 Descrição Geral

Fundamentados nas técnicas de computação evolucionária, *Kennedy* e *Eberhart* desenvolveram um método estocástico, inspirado na simulação do comportamento social de um bando de pássaros em revoada, com movimento localmente aleatório, mas global-

mente determinado [Eberhart & Kennedy, 1995] [Eberhart et al., 1996]. Esse algoritmo é denominado de *Particle Swarm Optimization* (PSO) ou Otimização por Nuvem (ou Enxame) de Partículas.

O PSO é formado por uma população de indivíduos, chamados de partículas, que ao invés de utilizarem operadores genéticos, evoluem através da *cooperação* e *competição* entre si. Em outras palavras, as partículas se beneficiam de sua própria experiência e da experiência de outros membros da nuvem durante a busca por uma melhor solução, da mesma forma que o ser humano se espelha em sua própria experiência e dos indivíduos mais bem sucedidos.

Em sua versão original, o PSO proposto por [Eberhart & Kennedy, 1995] foi criado para otimizar funções contínuas não-lineares, tendo como base de estudos a modelagem de grupos sociais simplificados. O foco dos criadores da técnica era descobrir uma descrição da dinâmica de movimento que envolvia os grupos de animais que, mesmo em grande quantidade de indivíduos, apresentavam sincronismo ao mudar de direção, espalhar-se, reagrupar-se, etc.

O algoritmo é composto por um conjunto de partículas movendo-se em um espaço de busca $d$-dimensional, sendo cada partícula uma solução potencial para o problema. Por moverem-se no espaço, cada partícula $i$ que compõe a nuvem é representada por uma posição no espaço $d$-dimensional, dada pela Equação 3.1.

$$X_i = (X_{i_1}, X_{i_2}, ..., X_{i_d}) \tag{3.1}$$

Para um melhor entendimento, considere o problema de obter as raízes de uma função simples do segundo grau, definida por: $f(x) = ax^2 + bx + c$, a qual possui no máximo dois valores que tornem $f(x) = 0$. Assim, deseja-se obter dois valores reais (raízes) $x_1$ e $x_2$, que satisfaçam a função $f(x)$. Note que esse não é um problema difícil o suficiente para o uso do PSO, já que existe uma equação matemática para obtenção das raízes, mas será apresentado aqui apenas para ilustrar o método.

O PSO para resolução desse problema seria modelado utilizando-se partículas compostas por duas dimensões, conforme a Equação 3.1. Cada uma dessas dimensões armazenaria um valor referente às raízes $x_1$ e $x_2$.

Para que as partículas se movam no espaço, é utilizada uma taxa da mudança de posição, que é chamada de velocidade e, para a $i$-ésima partícula, é representada pela Equação 3.2.

$$V_i = (V_{i_1}, V_{i_2}, ..., V_{i_d}) \tag{3.2}$$

O fluxo de execução de um PSO, segue, em suma, a descrição da Figura 3.1.

Inicialmente, todas as partículas são inicializadas aleatoriamente, i.e., suas posições e velocidade. O desempenho de cada partícula é medido de acordo com uma função de aptidão pré-definida (*fitness*), relacionada ao problema a ser resolvido. No caso do exemplo aqui utilizado, a *fitness* de cada partícula é calculada sobre a própria função quadrática $f(x)$ definida. Assim, como os valores ótimos a serem encontrados são aqueles que fazem $f(x) = 0$, quanto mais próximo de zero o valor da *fitness*, melhor será a solução da partícula. Porém, como neste problema, especificamente, seriam obtidos dois valores de *fitness*, um para cada raiz da função, o valor da *fitness* de cada partícula $p$ pode ser dado por $fitness_p = |fitness_{x_1}| + |fitness_{x_2}|$. Ou seja, quanto menor o valor de $fitness_p$, melhor a qualidade da solução da partícula. Esse processo se repete até que um critério de parada, que normalmente se refere a o número máximo de iterações, seja satisfeito. A seguir a posição e velocidade da partícula são atualizadas. Esse processo de atualização a cada iteração depende da definição de uma vizinhança, cuja obtenção é descrita na Seção 3.3. Na Seção 3.4 são descritos os parâmetros de um algoritmo PSO, incluindo o número máximo de iterações.

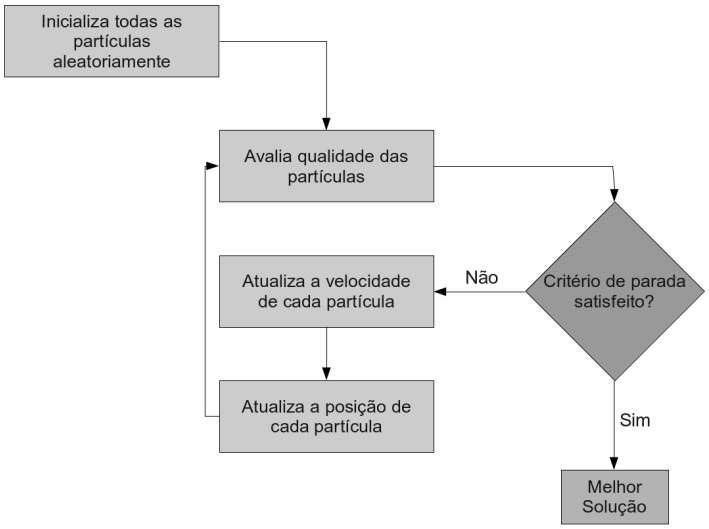



**Figura 3.1.** Fluxograma de um PSO.

## 3.2 Atualizando Velocidades e Posições

A movimentação de cada partícula é baseada em três fatores (Equação 3.3):

- Fator de Inércia (FI): delimita o movimento, uma vez que esse é direcional e determinado.

- Fator de Cognitividade Individual (FCI): determina a atração da partícula a sua melhor posição;

- Fator de Sociabilidade (FS): determina a atração das partículas para a melhor posição descoberta por qualquer elemento da nuvem; e

Devido a esses fatores, ao longo das iterações do algoritmo, cada partícula mantém o rastro de suas coordenadas no espaço de busca do problema, guardando a melhor posição em que aparece até o momento (*pbest*). Dado o exemplo de encontrar os zeros de uma função quadrática, a melhor posição é aquela que possibilita um valor de $f(x)$ mais próximo de zero. Outro valor utilizado pelas partículas é o melhor valor obtido por uma partícula em uma determinada vizinhança (*gbest*), como descrito na Seção 3.3.

Para calcular a nova velocidade da partícula $i$ na dimensão $d$, utiliza-se a Equação 3.3. A nova posição da partícula é determinada pela Equação 3.4.

$$V_{i_d} = \underbrace{W \times V_{i_d}}_{FI} + \underbrace{C_1 \times R_1 \times (P_{i_d} - X_{i_d})}_{FCI} + \underbrace{C_2 \times R_2 \times (P_{g_d} - X_{i_d})}_{FS} \tag{3.3}$$

$$X_{i_d} = X_{i_d} + V_{i_d} \tag{3.4}$$

Na Equação 3.3, define-se:

- $W$ - é o fator de inércia que determina a diversificação ou intensificação das partículas.

- $C_1$ e $C_2$ - são duas constantes positivas que correspondem às componentes cognitivas individual e social.

- $R_1$ e $R_2$ - são duas funções aleatórias no intervalo [0,1].

- $P_{i_d}$ - é a melhor posição encontrada pela partícula ($p_{best}$).

- $P_{g_d}$ - é a melhor posição global encontrada por todas as partículas ($g_{best}$).

- $X_{i_d}$ - é a posição atual da partícula.

Com a associação desses fatores, a partícula movimenta-se no espaço de busca sofrendo influência de sua memória (melhor posição encontrada pela partícula em sua vida - $p_{best}$), inércia (impele a partícula em uma direção idêntica a que ela vinha seguindo) e

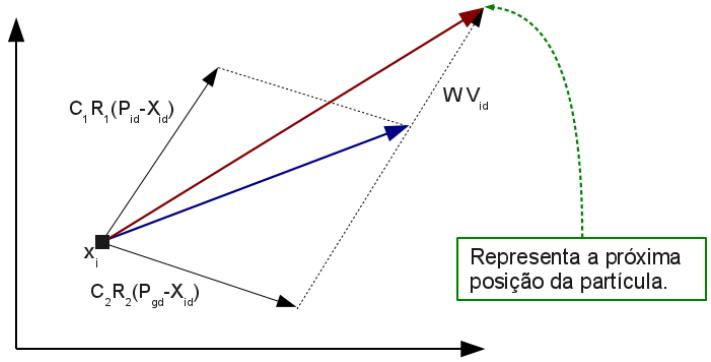



**Figura 3.2.** Movimentação de uma partícula.

comparação (melhor ponto descoberto pela nuvem - $g_{best}$). Esse deslocamento da partícula é demonstrado na Figura 3.2, que exemplifica uma partícula que se encontrava na posição $X_j$ e desloca-se para posição $X_i$ fazendo o uso dos vetores mencionados.

Dessa forma, a Equação 3.3 é usada para calcular a nova velocidade da partícula de acordo com sua velocidade anterior e as distâncias entre sua posição atual, sua melhor posição e a melhor posição do grupo. Com isso, a partícula desloca-se para uma nova posição de acordo com Equação 3.4.

Contudo, existe a necessidade do controle da velocidade da partícula em cada dimensão. Com isso, define-se uma velocidade máxima, o que previne a explosão das partículas. Este controle é feito em cada dimensão $d$ de cada partícula $i$, logo:

$$Se\ (V_{id} > V_{max})\ então\ V_{id} = V_{max}$$

$$Senão\ se\ (V_{id} < -V_{max})\ então\ V_{id} = -V_{max}$$

O mesmo pode acontecer com relação à posição da partícula, pois em muitos casos, o espaço de busca das partículas deve ser limitado, devendo cada posição $X_{id}$ de uma partícula $i$ pertencer a um intervalo $[-X_{max}, X_{max}]$. No caso do nosso exemplo, em que desejamos encontrar duas raízes $x_1$ e $x_2$ que torne $f(x) = 0$, o espaço de busca das partículas é ilimitado, ou seja, $X_{id} \in \Re$. Assim, neste caso, o intervalo ao qual pertence cada posição da partícula é $[-\infty, +\infty]$.

## 3.3 Vizinhança em um PSO

Como mencionado anteriormente, na atualização da velocidade, algoritmos PSO utilizam modelos de vizinhança. A vizinhança de uma partícula pode ser encontrada seguindo-se duas ópticas distintas. A primeira permite que uma vizinhança seja definida com base na qualidade da solução apresentada por cada partícula, ou seja, o

quão boa é sua *fitness*. Essa vizinhança é dita social, pois não importa a disposição das partículas no espaço, mas sim a qualidade de sua solução para o problema. Já a segunda permite que os vizinhos de uma partícula sejam encontrados com base nos valores de suas posições no espaço de busca, i.e., com base em suas informações geográficas. Além disso, uma partícula na nuvem também é considerada sua própria vizinha. A Figura 3.3 exemplifica esses modelos de vizinhança.

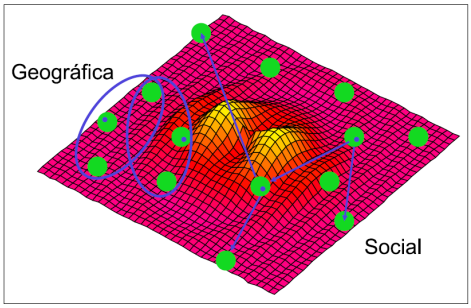



**Figura 3.3.** Vizinhança em um PSO.

Na grande maioria dos casos, o PSO utiliza uma vizinhança global - topologia em estrela (Figura 3.4), i.e., todos as partículas são influenciadas pela melhor partícula da nuvem. Isso acontece porque quando uma vizinhança é restrita - topologia em anel (Figura 3.5), ou seja, é formada por grupos de até $n$ partículas, torna-se mais demorada a transmissão da melhor posição através da nuvem e, consequentemente, a convergência é mais lenta. Ao mesmo tempo, essa vizinhança provê uma maior diversidade entre partículas, o que pode ser bom para que o espaço de busca seja explorado de forma sistemática.

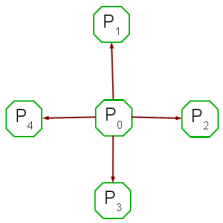



**Figura 3.4.** Topologia em Estrela (Vizinhança global).

Uma vizinhança do tipo anel (Figura 3.5), pode ser utilizada quando se deseja encontrar ótimos locais, *lbest*. Assim, o fator social da partícula passa a ser a experiência da vizinhança a qual ela pertence em conjunto com seu próprio conhecimento adquirido. Neste caso, o termo $P_{g_d}$ da Equação 3.3 remete à melhor posição da melhor partícula dentro da vizinhança local, ou seja, aquela que obteve uma melhor valor de

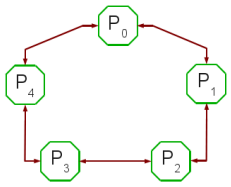



**Figura 3.5.** Topologia em Anel (Vizinhança local de tamanho igual a três).

*fitness*. No decorrer de $i$ iterações, as partículas podem mudar de vizinhos, fazendo parte de diferentes vizinhanças que são formadas por conta de sua movimentação no espaço de busca ou pela qualidade de suas soluções, de acordo com métricas definidas na modelagem do PSO.

Já em uma vizinhança do tipo estrela (Figura 3.4), é levada em consideração a melhor dentre todas as partículas em conjunto com seu próprio conhecimento adquirido. Neste caso, o termo $P_{gd}$ da Equação 3.3 remete à melhor posição da melhor partícula da nuvem ($g_{best}$). No decorrer de $i$ iterações, a melhor partícula global pode mudar e ao final das iterações as outras partículas tendem a ter soluções próximas à melhor global, na maioria dos casos.

## 3.4 Parâmetros de um PSO

Esta seção descreve os parâmetros de um algoritmo PSO, como número de partículas e iterações e, aqueles encontrados na equação de atualização da velocidade de uma partícula (Equação 3.3), que são os que definem a velocidade máxima, composta pelo peso de inércia ($W$); o fator cognitivo individual, composto pela componente cognitiva individual ($C_1$) e uma função de aleatoriedade ($R_1$); e também aqueles que definem o fator de sociabilidade de uma partícula, que é composto por um fator social ($C_2$) e uma função de aleatoriedade ($R_2$).

O número de partículas em um espaço de busca e a quantidade de iterações do algoritmo são claramente conhecidos como fatores importantes na probabilidade de encontrar o ótimo. Quanto maior o número de partículas em um determinado espaço, mais alta será a probabilidade de se chegar à solução ótima ou próxima a essa. O mesmo acontece com relação ao número de iterações, que podemos entender como o tempo de aprendizagem das partículas. Porém, reciprocamente, um número maior de partículas e iterações resultará no aumento de pontos individuais que serão testados, aumentando assim o tempo de computação.

Cada problema, em sua especificidade, requer uma nuvem com quantidades distintas de partículas e, consequentemente, diferentes quantidades de iterações. Com

isso, após um determinado tempo (iterações) em que uma certa quantidade $n$ de partículas está cooperando para solucionar um dado problema, a curva de aprendizado tende a estagnar, conforme pode ser observado pela Figura 3.6.


**Figura 3.6.** Curva de aprendizagem.

Outro fator importante é o peso de inércia $W$, que é empregado para controlar o impacto da velocidade anterior na velocidade atual, influenciando as habilidades de exploração global e local das partículas. Um peso de inércia maior facilita a exploração global (procurando novas áreas), enquanto um peso de inércia menor tende a facilitar exploração local para refinar a área de procura atual. A seleção satisfatória do peso de inércia $W$ pode prover um equilíbrio entre essas habilidades de exploração, e requer menos iterações para encontrar os solução ótima.

Os parâmetros de confiança $c_1$ e $c_2$ representam a ponderação dos termos estocásticos de aceleração que empurram as partículas em direção às posições $p_{best}$ e $g_{best}$ [Eberhart et al., 1996]. Esses parâmetros indicam o quanto as partículas confiam em sua própria solução ($c_1$) - parte cognitiva individual; e na nuvem ($c_2$) - parte cognitiva social. Na maioria das vezes, os valores para esses parâmetros de confiança são iguais.

Os fatores $R_1$ e $R_2$ são duas funções aleatórias utilizadas para manter a diversidade da população e são distribuídos uniformemente no intervalo $[0, 1]$.

## 3.5 PSO em Classificação

Na Tabela 3.1, é apresentado um quadro comparativo entre vários trabalhos que utilizam o algoritmo PSO em diversos contextos de classificação, incluindo a combinação de classificadores em comitês. Em todos estes trabalhos, o PSO foi considerado um bom método para resolver o problema a que foi submetido.

Nesses diversos trabalhos apresentados na Tabela 3.1, o PSO foi um método utilizado com sucesso para resolver diversas tarefas, que abordam a ponderação de atributos para redes neuronais, parâmetros para algoritmos de classificação e, por fim, no trabalho de [Mohemmed et al., 2009] foi proposto um algoritmo PSO para classificação

**Tabela 3.1.** Comparativo entre diferentes abordagens utilizando PSO.

| Autores | Tarefa Resolvida | Modo de Resolução do Problema | Aplicação |
|---|---|---|---|
| [Mohemmed et al., 2009] | Classificação | Voto da maioria em um comitês de classificadores | Dez problemas da UCI-ML |
| [Wen et al., 2011] | Classificação | PSO para ponderação de parâmetros para o classificador SVM | Base de dados de glicogênio e carbono |
| [Junior et al., 2010] | Classificação | PSO para ponderação de visões | Texto e Rede social de vídeo |
| [Zhou et al., 2010] | Classificação | PSO como classificador | Três contextos distintos |
| [Zou et al., 2010] | Classificação | Seleção de atributos | Base de combustível e informações de qualidade da carne |
| [Khan et al., 2010] | Classificação | Um algoritmo PSO para induzir regras de classificação | Cinco bases de dados distintas |
| [Cao & Liu, 2010] | Classificação | Algoritmo KNN baseado em um PSO para ponderação de atributos | Dez bases de dados distintas |
| [Khurshid & Gokhale, 2009] | Classificação | PSO para ajustar parâmetros em uma rede neuronal utilizada como classificador | Seis esquemas distintos de sinais digitais |
| [Escalante et al., 2009] | Classificação | Seleção de atributos e ponderar variáveis do LS-SVM[1] | Cinco contextos distintos |
| [Holden & Freitas, 2008] | Classificação | Um algoritmo PSO híbrido para descobrir regras de classificação | Base de dados UCI |
| [Evans & Zhang, 2008] | Classificação | Particionamentos dos atributos e ponderação das classes com decisão final por um voto da maioria | Base de imagens |

[1] *Least-Squares Support Vector Machine.*

baseado em um comitê de classificadores. Porém, o modelo classificatório utilizado é baseado em centroides. Assim, o PSO tenta encontrar o melhor centroide para cada classe. Duas abordagens são utilizadas: a primeira codifica cada partícula como um vetor $n$-dimensional, onde os centroides para cada classe são otimizados. Já na segunda abordagem, cada partícula otimiza o centroide para apenas uma classe. Com isso, a **classificação é feita pelo PSO** e ao final os resultados são combinados por um simples voto da maioria.

# Capítulo 4

# Método PSO para Combinação de Classificadores

Um método heurístico é aquele utilizado quando várias abordagens para solução de um problema são conhecidas, mas não existe um algoritmo para resolver o problema de modo consistente [Michalewicz & Fogel, 2000]. O método heurístico examina o problema e tenta aplicar cada uma das abordagens possíveis de resolução, sendo capaz de julgar, após a tentativa, se o problema está próximo ou não da solução. Dessa forma, sem oferecer garantias, o algoritmo tem como objetivo resolver problemas complexos para encontrar soluções de boa qualidade. Em casos como o problema da combinação de $v$ visões após a classificação dos dados, em que há mais de uma solução para o problema, os métodos heurísticos são bastante indicados, pois são eficazes em tais situações.

O PSO é uma boa alternativa para resolver o problema da combinação de $v$ visões após a classificação dos dados, devido a sua combinação de técnicas de aprendizado local e global. Essa combinação é feita através das influências sofridas pela vizinhança e pela experiência da própria partícula. Além disso, embora o PSO tenha sido inicialmente proposto para otimização de valores contínuos para funções, ao longo do tempo, foi refinado e melhorado, com diversas variações propostas [Kennedy et al., 2001]. Hoje o PSO é um algoritmo utilizado para resolver diversos problemas [Guo & Gao, 2009] [Vaisakh et al., 2009] [Horng et al., 2010] [Tang et al., 2010] [Uğ Andur et al., 2010] [Rani & Deepa, 2010] nas mais diversas áreas, como, por exemplo, a tarefa de classificação (Seção 3.5).

Assim, dado o problema da combinação de classificadores, esse trabalho apresenta dois métodos distintos, baseados em PSO, para resolver o problema. Essas duas versões propostas permitem abordar o problema da combinação de $v$ visões dos dados

de maneiras distintas. O primeiro método (PSO-WV) a ser descrito será o PSO para ponderar $v$ classificadores, sejam esses classificadores heterogêneos ou homogêneos, em diferentes visões de uma base de dados (*Subseção* 4.2), o que permite fazer inferências com relação ao rendimento de cada classificador em cada visão. Posteriormente, é apresentado o segundo método (PSO-WC), que objetiva tratar o problema da combinação de $v$ classificadores levando em consideração o *ranking* das classes disponibilizado pelos algoritmos de classificação, atribuindo pesos a cada uma das $m$ classes, em cada visão (*Subseção* 4.3). Essa abordagem permite capturar o erro inerente a cada algoritmo de classificação, em cada visão dos dados, além de possibilitar uma análise do rendimento dos classificadores em cada visão dos dados.

## 4.1 Descrição Geral do PSO

O algoritmo PSO que foi modelado para o problema de aprendizado multi-visão segue uma abordagem um pouco diferente do PSO clássico descrito no Capítulo 3. Aqui, é utilizada uma variação da Equação 3.3, que calcula a velocidade. Conforme apresentado por [Kennedy et al., 2001], os termos da fórmula referentes a *pbest* e *gbest* podem ser recolhidos para um único termo ($\rho$), sem que ocorra perda de qualquer informação, onde $\rho$ representa a média ponderada desses dois melhores valores. Para o cálculo do valor de $\rho$ são levados em consideração os fatores cognitivo individual e social das partículas, que são representados, respectivamente, por $\varphi_1$ e $\varphi_2$.

A partir disso, a cada iteração o valor de $\varphi_1$ diminui e o de $\varphi_2$ aumenta, fazendo com que as partículas, ao longo do tempo (iterações), passem a confiar mais na nuvem de partículas do que em suas próprias soluções. Em uma analogia, podemos comparar esse fatores com um grupo de pessoas ou animais, em que todos os integrantes são desconhecidos entre si. No início, pelo fato da falta de afinidade e conhecimento, os membros do grupo não confiam uns nos outros. Porém, com o tempo e a convivência, passam a se conhecer e a confiar nas decisões de seus semelhantes. Isso acontece com todos animais que vivem em sociedade. Portanto, essa metáfora pode ser trazida ao PSO, que é, como descrito anteriormente, um algoritmo derivado do comportamento social de grupos (bandos e/ou enxames) de animais.

Esse PSO utiliza a noção de vizinhança social do tipo estrela (Figura 3.4), onde a melhor partícula da nuvem, na iteração $i$, compartilha seu conhecimento com todo o grupo. Esse tipo de vizinhança foi utilizado pois, para este problema, existe a necessidade de que os pesos encontrados sejam generalizáveis, com o intuito de evitar *overfitting* [Kohavi & Sommerfield, 1995]. As próximas seções apresentam o PSO-WV

e PSO-WC.

## 4.2 PSO para Ponderação das Visões (PSO-WV)

O algoritmo PSO para ponderar $v$ visões, classificadas por $v$ classificadores, distintos ou não, teve suas partículas modeladas com $v$ dimensões, onde cada dimensão corresponde a um peso $W_v$ para cada uma das $\phi_v$ saídas dos classificadores (confianças), sendo $v$ a quantidade de visões do problema, conforme matriz de decisão, apresentada na Subseção 2.2.2. Cada dimensão das partículas é formada por um valor real positivo, sendo sua soma igual a 1. Com isso, para cada classificador $\varphi_v$, um peso $W_v$ é gerado.

Assim, dado um problema com $v$ visões e $m$ classes, e a confiança $\phi$ do classificador em predizer a classe de uma instância $e$, o PSO como método de combinação irá decidir a classe de $e$ baseado na $max(\phi_{e_c})$, onde $\phi_{vm}$ é a confiança do classificador $v$ em classificar uma instância $e$ como pertencente à classe $c$, $\forall\ c \in m$, e é definido pela Equação 4.1.

Ao longo de $i$ iterações, as partículas trocam informações e experiências, encontrando os melhores pesos para cada classificador/visão, a fim de obter uma classificação final com $\mu f1$ maior (métrica descrita na Seção 4.4). A Figura 4.1 mostra a decodificação de uma partícula do PSO-WV. Seu funcionamento está descrito no *Algoritmo* 2.

$$max(\phi_{e_c}) = argmax_v(W_v \times argmax_m(\phi_{vm})) \tag{4.1}$$

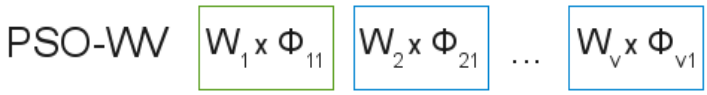



**Figura 4.1.** "Decodificação" de uma partícula: como o PSO-WV utiliza as informações da matriz de decisão para classificar uma nova instância $e$.

## 4.3 PSO para Ponderação das Classes (PSO-WC)

No segundo tipo de PSO implementado, diferentemente do primeiro, todo o *ranking* de $m$ classes gerado por um classificador $C$ em cada uma das $v$ visões é considerado. Para cada classe $c_v$ é gerado um peso $W_{vm}$. Então, a representação de uma partícula é um vetor $v$-dimensional, e cada uma das $d_v$ dimensões contém um vetor $m$-dimensional que armazena os pesos respectivos a cada uma das $\phi_{vm}$ confianças de cada classificador $C_v$ para as $m$ classes.

**Algoritmo 2:** PSO-WV

```
para cada treino f faça
    Inicializa população
    Aplica pesos iniciais às decisões dos classificadores
    para cada Iteração i faça
        para cada Partícula p faça
            Atualiza informação local de p (pbest)
            Atualiza informação global da nuvem (gbest)
            Calcula a fitness de p
            para cada Visão v faça
                Atualiza fator individual (φ1)          /* Eq. 4.7 */
                Atualiza fator social (φ2)              /* Eq. 4.8 */
                Atualiza velocidade das dimensões de p  /* Eq. 4.10 */
                Atualiza posição das dimensões de p
                Aplica o peso W_v à saída φ_v
retorna W_gbest_f
Calcula a fitness para o teste f usando os pesos de gbest
```

Assim, dado um problema com $v$ visões e $m$ classes, e a confiança $\phi$ do classificador à classe $m$ de uma instância $e$, o PSO, como método de combinação, irá decidir, baseado na $max(\phi_{e_{vc}})$, onde $\phi_{vm}$ é a confiança do classificador $v$ em classificar uma lista de classes referentes a uma instância $e$, e é definido pela Equação 4.2.

Ao longo de $i$ iterações as partículas trocam informações e experiências, encontrando os melhores pesos para cada classe em cada classificador, a fim de obter uma classificação final com um $\mu f1$ maior. A Figura 4.2 mostra a decodificação de uma partícula do PSO-WC. Seu funcionamento está descrito no *Algoritmo* 3.

Dessa forma, ponderar as classes, permite capturar o erro inerente a cada classificador em sua respectiva visão, assim, é possível fazer uma avaliação do ganho de cada classe em cada visão dos dados e em cada classificador. Uma avaliação empírica é feita no Capítulo 6.

$$max(\phi_{e_{vc}}) = argmax_v(argmax_m(W_{vm} \times \phi_{vm})) \tag{4.2}$$

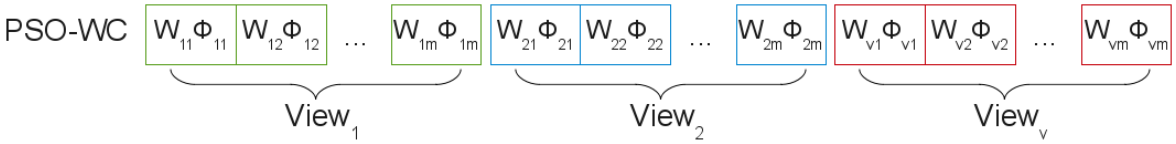



**Figura 4.2.** "Decodificação" de uma partícula: como o PSO-WC utiliza as informações do perfil de decisão para classificar uma nova instância $e$.

**Algoritmo 3:** PSO-WC

```
para cada treino f faça
    Inicializa população
    Aplica pesos iniciais às saídas
    para cada Iteração i faça
        para cada Partícula p faça
            Atualiza informação local de p (pbest)
            Atualiza informação global da nuvem (gbest)
            Calcula a fitness de p
            para cada Visão v faça
                para cada Classe m faça
                    Atualiza fator individual (φ1)          /* Eq. 4.7 */
                    Atualiza fator social (φ2)              /* Eq. 4.8 */
                    Atualiza velocidade das dimensões de p  /* Eq. 4.10 */
                    Atualiza posição das dimensões de p
                    Aplica os pesos W_{v_m} à saída φ_{v_m}
retorna W_{gbest_f}
Calcula a fitness para o teste f usando os pesos de gbest
```

## 4.4 Avaliação das Partículas

Para este trabalho, a métrica de avaliação escolhida foi a Micro F1 ($\mu_{f1}$), obtida a partir da multiplicação dos $v$ pesos encontrados pela nuvem de partículas a cada iteração $i$. As métricas de avaliação do PSO são as mesmas utilizadas para os algoritmos de classificação mono-visão. Abaixo são apresentadas as principais medidas de classificação utilizadas na literatura para a avaliação de classificadores.

Nas equações seguintes, define-se:

- Verdadeiro positivo (VP): número de instâncias positivas e classificadas como tal;
- Falso positivo (FP): número de instâncias negativas, porém classificadas positivas; e
- Falso negativo (FN): número de instâncias positivas, porém classificadas negativas.

1. Precisão ($Pr$): taxa com que todas as instâncias classificadas como positivas são realmente positivas. Nenhum exemplo negativo é incluído;

$$Pr = \frac{VP}{VP + FP} \tag{4.3}$$

2. Revocação ($Re$): taxa com que classifica como positivas todas as instâncias que são positivas. Nenhuma instância positiva é deixada de fora. Apresenta uma

indicação do quanto do total de informação relevante foi recuperada;

$$Re = \frac{VP}{VP + FN} \tag{4.4}$$

3. Macro-média F1 ($\eta f1$): parte do princípio que cada classe $i$ é igualmente importante e, por isso, atribui-lhes pesos iguais, independente da quantidade de instâncias contidas em cada classe, onde $M$ é o total de classes. Além de ser também baseada na precisão e revocação; e

$$\eta_{f1} = \frac{\sum_{i=1}^{M} F_i}{M}, \qquad F_i = \frac{2 \times Pr \times Re}{Pr + Re} \tag{4.5}$$

4. Micro-média F1 ($\mu_{f1}$): pondera os F1 de cada classe com base na representatividade da classe na coleção de acordo com o número de instâncias em cada classe, ou seja, é uma média harmônica entre a precisão e a revocação. Equivale à acurácia, quando existe previsão para todas as instâncias.

$$\mu_{f1} = \frac{2 \times Pr \times Re}{Pr + Re} \tag{4.6}$$

Em termos qualitativos, a macro-média F1 parte do princípio que cada classe é igualmente importante, e, por isso, atribui-lhes pesos iguais, independente da quantidade de instâncias contidos em cada uma. Por outro lado, a micro-média F1 parte da premissa que cada instância é igualmente importante. Como resultado, se a maioria das classes em uma base de dados contiver, proporcionalmente, poucas instâncias em relação ao todo, então a macro-média F1 é uma métrica tipicamente mais relevante, pois são raros os casos em que é adequado subestimar a importância de uma vasta diversidade de classes. Caso contrário, a micro-média F1 é uma métrica tipicamente mais significativa.

Assim, para cada iteração $i$ em um conjunto com $v$ partículas, são gerados $v$ valores de *fitness*. Quanto maior o valor alcançado, melhor a solução.

Na MultiViL, ferramenta descrita no Capítulo 5, é possível escolher qualquer umas das outras métricas como formas de avaliação das partículas.

## 4.5 Atualização da Velocidade e Posição

Como já mencionado, a variação do PSO utilizada, aqui, une os fatores cognitivos individual e social em uma única variável ($\rho$). As Equações 4.7 e 4.8 definem esses

fatores cognitivos individual e social, respectivamente. Assim, podemos simplificar a fórmula da velocidade (Equação 3.3) para a fórmula definida na Equação 4.10, onde $\varphi$ é um valor aleatório $\in [0, 1]$. A equação que define a nova posição da partícula permanece a mesma, conforme a Equação 3.4, descrita anteriormente.

$$\varphi_1 = C_{MAX} - (\frac{(C_{MAX} - C_{MIN}).iter}{NumMaxIter}) \tag{4.7}$$

$$\varphi_2 = C_{MIN} + (\frac{(C_{MAX} - C_{MIN}).iter}{NumMaxIter}) \tag{4.8}$$

$$\rho = \frac{\varphi_1.pbest + \varphi_2.gbest}{\varphi_1 + \varphi_2} \tag{4.9}$$

$$V = V + \varphi.(\rho - X) \tag{4.10}$$

Nas Equações 4.7 e 4.8, define-se,

- $C_{MIN} = 0.5$;
- $C_{MAX} = 4$;
- $NumMaxIter$ é o número máximo de iterações; e
- $iter$ é a iteração corrente.

Desta forma, pelas Equações 4.7 e 4.8, a cada iteração, $\varphi_1 + \varphi_2 = 4$, o que faz com que as partículas percorram o espaço de busca de forma sistemática, conforme mostrado em [Kennedy et al., 2001].

Através da Equação 4.9, não é necessário o controle da posição máxima e posição mínima, pois $\rho$ terá sempre um valor válido, sendo este estipulado pelas posições *pbest* e *gbest*.

# Capítulo 5

# A Ferramenta MultiViL

O nome MultiViL vem do inglês ***Multi-View Learning*** ou Aprendizado Multi-Visão. Essa é uma ferramenta criada dentro deste trabalho de mestrado para realizar o aprendizado multi-visão, e está disponível gratuitamente, na *Web*[1]. A MultiViL integra quatro algoritmos de classificação, são eles SVM-Perf [Vapnik, 1995], Naïve Bayes [McCallum & Nigam, 1998], Rocchio [Rocchio, 1971], KNN (*K-Nearest Neighbor*) [Zhang & Zhou, 2007]. Além disso, a MultiViL também permite a combinação dos resultados de $v$ visões, e inclui um esquema de visualização da cooperação entre as visões utilizando diagramas de *Venn* e gráficos.

Além dos algoritmos de classificação supracitados, a MultiViL disponibiliza quatro métodos de combinação de resultados: o voto da maioria, o algoritmo de *Borda Count*, o algoritmo baseado na Teoria de Dempster-Shafer [Bi et al., 2008] e a ponderação das visões pelo PSO proposto nesta dissertação. O método PSO é descrito no Capítulo 4 e os demais métodos de combinação são descritos na Seção 2.3. A Figura 5.1 mostra um esquema de execução da MultiViL, que vai desde a base de dados até a visualização final dos resultados.

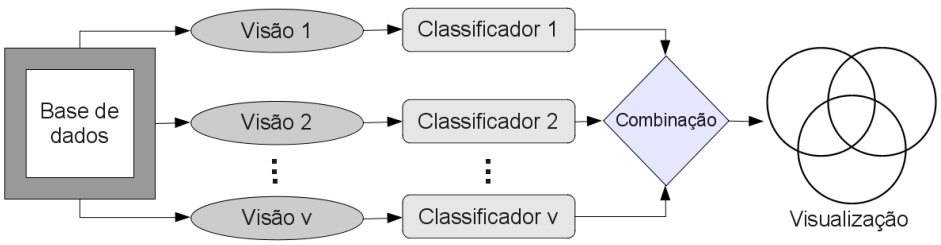



**Figura 5.1.** Um esquema para aprendizado multi-visão.

A seguir são apresentadas as principais telas da MultiViL.

[1]http://multivil.sourceforge.net/

A Figura 5.2 mostra a tela inicial da MultiViL. Nessa tela, é possível realizar a classificação de qualquer base de dados em qualquer um dos algoritmos de classificação acima citados, seja a base dividia em multi-visões ou não.

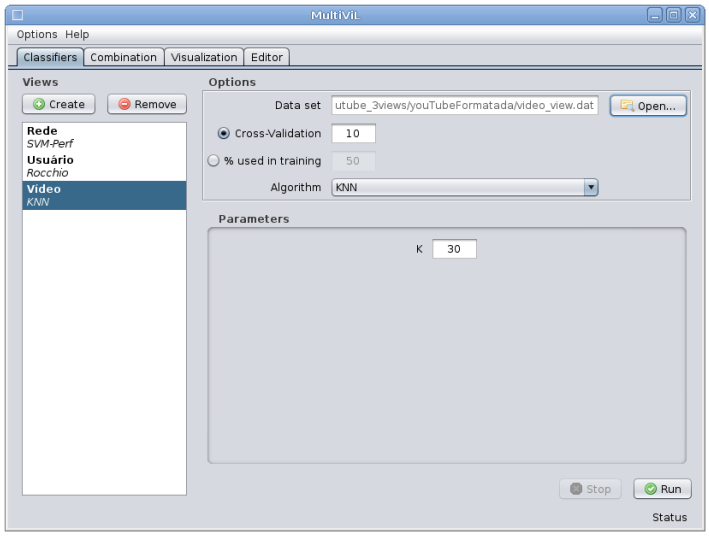



**Figura 5.2.** Tela de classificação da MultiViL.

A MultiViL permite que o usuário a utilize tanto em modo gráfico como por linha de comando (terminal). A Figura 5.3 refere-se à tela dos métodos de combinação, nela temos um exemplo do PSO para combinar três visões de uma base de dados. Essa tela disponibiliza, também, os demais métodos de combinação descritos anteriormente.

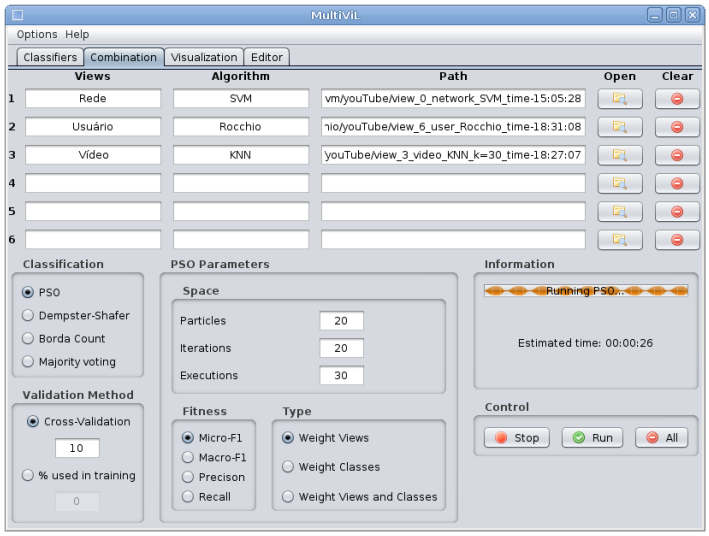



**Figura 5.3.** Tela de combinação de classificadores/visões pela MultiViL.

Para visualização dos resultados individuais dos classificadores e também após a combinação, um simples editor de texto é disponibilizado pela MultiViL (Figura 5.4).

O editor permite, além da avaliação dos resultados, que o usuário acesse e altere as bases de dados a serem classificadas.

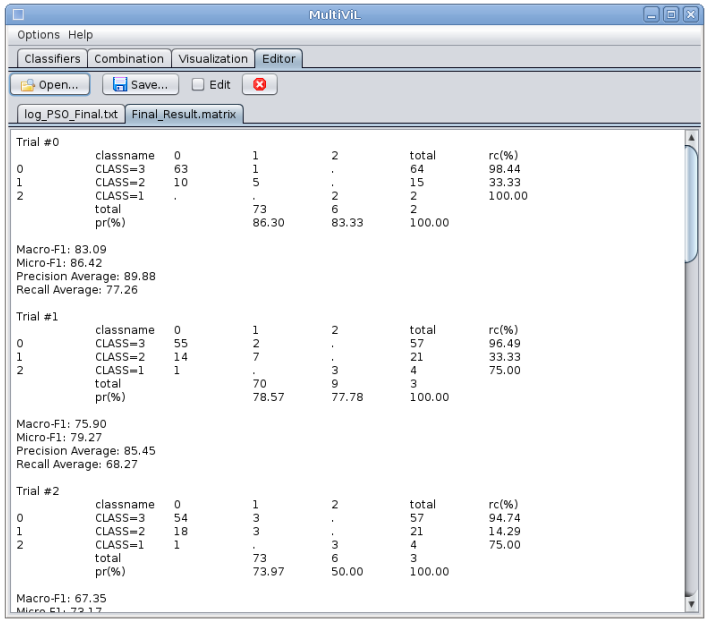



**Figura 5.4.** Tela do editor de texto da MultiViL

Após a combinação dos resultados, é de grande interesse do usuário saber a contribuição de cada visão na classificação. Desta forma, a MultiViL disponibiliza um gráfico (Figura 5.5) com o rendimento em porcentagem de instâncias que cada visão dos dados contribuiu com a tarefa de classificação por multi-visão, ou seja, permite que o usuário tenha acesso às informações com relação à cooperação entre as visões, facilitando, assim, a análise do resultado final da classificação. Nessa mesma tela é possível verificar, através do diagrama de Venn (Figura 5.6), o quanto que as visões concordam antes e após a combinação. Esse modo de análise está disponível somente para bases de dados com quatro visões ou menos, devido à dificuldade em se "desenhar e visualizar" um diagrama de Venn para uma quantidade maior de visões.

Assim, a MultiViL torna-se uma importante e alternativa ferramenta para trabalhar no contexto de classificação e combinação de classificadores. Nessa primeira versão é disponibilizado o aprendizado supervisionado, mas, posteriormente, no decorrer de sua evolução, novos tipos de aprendizado poderão ser adicionados, além de novos algoritmos de classificação e agrupamento, além de novas formas de visualização dos dados e dos resultados.

É importante notar que a MultiViL foi construída de forma que a adição de novos classificadores e/ou métodos de combinação seja simples. A codificação da MultiViL foi dividida da seguinte forma: todos os métodos de combinação de classificadores e


**Figura 5.5.** Tela de visualização da cooperação entre as visões na MultiViL.

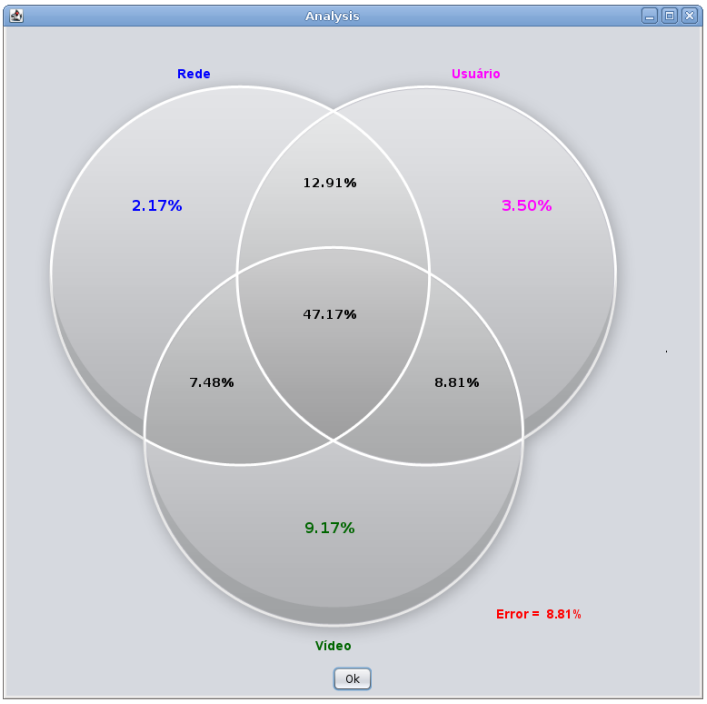



**Figura 5.6.** Tela de análise da concordância entre as visões pelo diagrama de Venn na MultiViL.

a interface com o usuário foram desenvolvidos na linguagem de programação Java; os algoritmo de classificação presentes na MultiViL foram implementados nas linguagens C e C++ (KNN, Naïve Bayes e Rocchio), sendo o SVM codificado nessas duas últimas.

# Capítulo 6

# Resultados Experimentais

Este capítulo apresenta os resultados obtidos pelo PSO-WV e PSO-WC em duas bases de dados, com diferentes domínios de aplicação. Na Seção 6.1 essas bases de dados são descritas, sendo que a primeira trata da classificação de documentos de texto e a segunda da classificação de vídeos em uma rede social. Na Seção 6.2 é apresentada a metodologia utilizada nesta dissertação, com a descrição de todas as etapas experimentais. Já na Seção 6.3 são apresentados os resultados sob a perspectiva mono-visão, mostrando a concordância das visões antes da combinação. Na Seção 6.4 são apresentados os resultados da abordagem multi-visão, com uma análise do comportamento do PSO e dos resultados obtidos pela combinação das visões pelo PSO-WV e PSO-WC e, também, pelos demais métodos de combinação de classificadores utilizados para comparação, que são descritos na Seção 2.3. Além disso, foi feita uma análise dos pesos gerados pelo algoritmo PSO, pois, a partir deles, pode-se entender melhor a relação entre as visões de cada base e o rendimento de cada algoritmo de classificação em cada uma delas, podendo, assim, ser possível identificar que algoritmos melhor se adaptam a cada visão.

## 6.1 Bases de Dados

Em multi-visão, existe uma grande dificuldade em se encontrar bases de dados preparadas. Embora, seja intuitivo enxergar diversas visões, principalmente no contexto das redes sociais e aprendizado utilizando imagens, a preparação dessas bases requer um esforço grandioso. Como esse não é o objetivo deste trabalho, foram utilizadas duas bases de dados, que estão inseridas em dois contextos bastante distintos. A primeira é relativa à classificação de textos, da qual foram preparadas e extraídas cada uma das três visões dos dados. A segunda, é uma base de dados que está inserida no contexto

das redes sociais, utilizada em [Benevenuto et al., 2009].

As redes sociais têm, em seus dados, características bastante distintas, permitindo uma divisão natural dos atributos. Com isso, e também com a crescente importância dessas redes na *Web*, é de extrema importância pesquisas em bases de dados nesse domínio, sobretudo com a utilização do aprendizado multi-visão. As bases de dados utilizadas nesse trabalho são descritas a seguir.

1. ACM-DL[1]: É um subconjunto com informações de artigos da *Digital Library of the Association for Computing Machinery*, composta por 6.681 documentos, divididos em 8 classes, distribuídos da seguinte forma: 702 A, 658 B, 1848 C, 420 D, 197 E, 1554 F, 1065 G, 237 H. Foram extraídas três visões dessa base:

   - **Termos**: corresponde aos termos presentes nos *abstracts* dos artigos;
   - **Rede de Co-autoria**: representa as relações de co-autoria entre os autores dos artigos, ou seja, se um indivíduo é autor de dois trabalhos distintos, uma aresta é criada entre esses dois artigos; e
   - **Rede de Citações**: representa as citações entre artigos, ou seja, quando artigos sitam o mesmo artigo uma aresta é criada entre esses dois.

   Essa base teve um tratamento especial, pois foram considerados somente os artigos que, na visão de Citações, apresentava no mínimo três citações. Foram utilizadas técnicas de pré-processamento de textos na visão de termos, como a remoção de sufixos e prefixos (*stemming*) e *stop words* [Baeza-Yates & Ribeiro-Neto, 2010].

2. YouTube [Benevenuto et al., 2009]: Contém informações sobre 829 usuários do YouTube[2], divididos em três classes, distribuídos da seguinte forma: 641 legítimos, 157 *spammers* e 31 *promoters*. Contém, também, três visões:

   - **Vídeo**: propriedades dos vídeos postados pelos usuário;
   - **Rede**: relação social entre usuário de vídeo resposta; e
   - **Usuário**: características de comportamento dos usuários.

É importante ressaltar que a divisão dessas duas bases de dados em três visões foi intuitiva. Talvez uma outra divisão levasse a resultados melhores.

[1] http://portal.acm.org/dl.cfm
[2] http://www.youtube.com

## 6.2 Metodologia Experimental

A fase experimental desta dissertação foi dividida em três etapas, que são descritas a seguir.

1. Abordagem mono-visão

    - Na primeira etapa, foram utilizados os quatro classificadores descritos na Seção 2.1. Esses classificadores foram executados independentemente em cada visão das duas bases de dados, descritas na seção anterior. É importante salientar que o SVM foi executado com o *kernel* linear [Carpineto et al., 2009] [Lodhi et al., 2002] e não houve uma otimização dos parâmetros desse algoritmo. Com relação ao KNN, foi utilizado um $K$ de valor 30. Nesse trabalho, não foi feita um estudo e otimização dos parâmetros para nenhum dos algoritmos de classificação, pois não estamos interessado em mostrar o poder de classificação individual, mas sim os ganhos que podem ser obtidos após a combinação das visões dos dados. A Figura 6.1 descreve essa etapa de classificação, onde as visões são divididas em 10 conjuntos de treino e teste e, ao final, tem-se cada uma das visões classificadas. Os classificadores também foram utilizados para categorizar a base de dados única, i.e., aquela contendo as informações de todas as visões juntas.

2. Abordagem multi-visão (homogêneo)

    - Em uma segunda etapa, os algoritmos PSOs propostos foram utilizados para combinar as visões, em cada base de dados, usando uma combinação homogênea, i.e., o mesmo classificador em todas as visões. Essa etapa foi de extrema importância para a fase experimental seguinte, pois, a partir dela, foi possível identificar qual o melhor classificador para cada uma das visões.

3. Abordagem multi-visão (heterogêneo)

    - A terceira etapa inicia-se com a combinação de classificadores heterogêneos, i.e., os classificadores utilizados em cada visão são distintos. E, como essa etapa é posterior àquela que identifica o melhor classificador em cada visão - combinação homogênea, foi possível verificar qual dos dois tipos de combinação obteve o melhor rendimento para cada uma das bases de dados. Para isso, foram feitas todas as combinações possíveis, dos quatro classificadores, nas três visões, em cada uma das bases de dados utilizadas. Nessa combinação, é desejado obter resultados a partir de algoritmos totalmente distintos

em cada visão, não permitindo que um mesmo algoritmo fosse utilizado em duas visões como no caso do KNN que obteve melhor rendimento para as visões de Vídeo e Usuário. Caso combinações de dois algoritmos em três visões fossem feitas, existiria uma gama de combinações a serem testadas, tornando-se inviável por questões de tempo. Esses resultados são apresentados nas seções seguintes. Nessa etapa, também foram executados os outros métodos de combinação de classificadores utilizados para comparação com o PSO. É importante salientar que, tanto para fase anterior quanto nesta, os algoritmos PSO - por serem não-deterministicos, foram executados 30 vezes com sementes distintas.

Todo o processo de combinação de classificadores aqui apresentado utiliza a matriz de decisão, descrita na Subseção 2.2.2 desta dissertação. Dessa forma, é possível combinar quaisquer classificadores, contanto que esses sigam as definições de uma matriz de decisão.

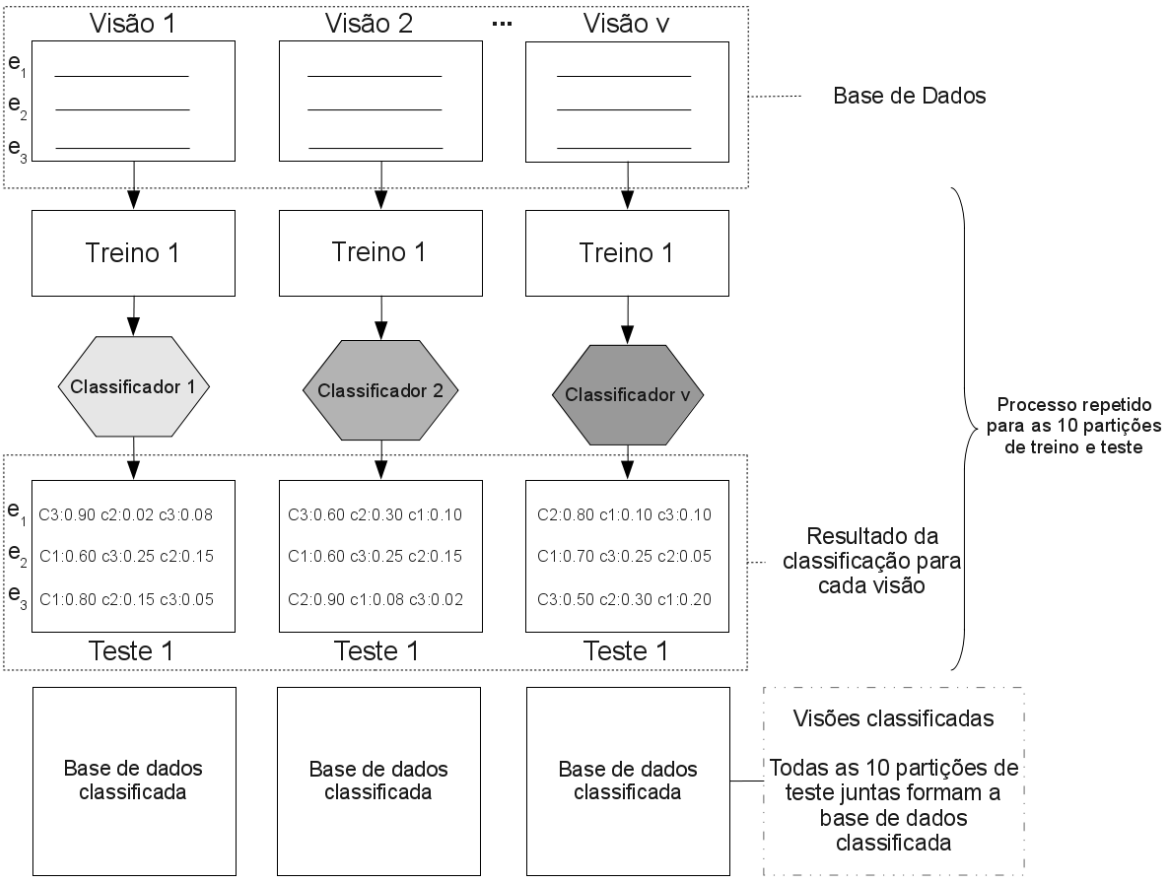



**Figura 6.1.** Primeira etapa dos experimentos - classificação das visões de cada uma das bases de dados.

O PSO modelado para resolver o problema da combinação das decisões de classificadores, trabalha com um conjunto de treino e outro de teste, onde o algoritmo aprende e evolui, gerando o conjunto de pesos que possibilitou um melhor resultado da classificação após combinar as saídas dos classificadores (fase de treinamento). Após essa etapa de aprendizado, o melhor conjunto de pesos obtido no treinamento (melhor partícula global) é diretamente aplicado ao conjunto de teste, onde é possível observar se o algoritmo conseguiu aprender e generalizar o conhecimento adquirido.

Como validação estatística é utilizada a validação cruzada, a qual divide um conjunto de instâncias em $k$ subconjuntos ou partições de treino e teste. A cada iteração do algoritmo, uma partição é utilizada para teste e as demais para o treinamento. Esse processo é repetido tantas vezes quanto for o número de partições. Neste trabalho, utilizamos a validação cruzada estratificada, que utiliza a mesma proporção de exemplos de cada classe do problema para formar as partições, fazendo com que cada uma delas tenha a mesma distribuição de instâncias por classe que o conjunto original [Browne, 2000]. Foram utilizados 10 conjuntos distintos de treino e teste, tanto para a categorização pelos algoritmos de classificação, quanto para o PSO.

Assim, o PSO gera dez conjuntos distintos de pesos, e aqueles que possibilitaram um maior valor de *fitness*, dentre os dez conjuntos de treinamento, são generalizado para todas as partições de teste. Dessa forma, é possível validar, estatisticamente, que o algoritmo consegue aprender e generalizar. Em um fluxo de execução, a Figura 6.1 mostra desde a primeira etapa de experimentos, onde as visões das bases de dados são classificadas utilizando a validação cruzada em cada um dos respectivos classificadores, até o terceira etapa, que é apresentada pela Figura 6.2 e demonstra o processo de combinação das visões pelo PSO-WV e/ou PSO-WC. Esse processo é o mesmo utilizado para combinações dos resultados pelos outros métodos de combinação utilizados neste trabalho. Vale salientar que os demais métodos de combinação de classificadores aqui utilizados não necessitam de uma fase de treinamento.

## 6.3 Resultados sob uma Perspectiva Mono-Visão

Após a categorização de cada visão pelos algoritmos de classificação descritos na Seção 2.1 (KNN, Naïve Bayes, Rocchio e SVM), é necessária uma comparação do rendimento das visões, ou seja, o quanto elas concordam antes da combinação pelos dois algoritmos propostos neste trabalho (PSO-WV e PSO-WC), e dos métodos de combinação de classificadores presentes na literatura e descritos na Seção 2.3 (voto da maioria, *Borda Count* e teoria de Dempster-Shafer). Essa etapa de visualização da concordância entre

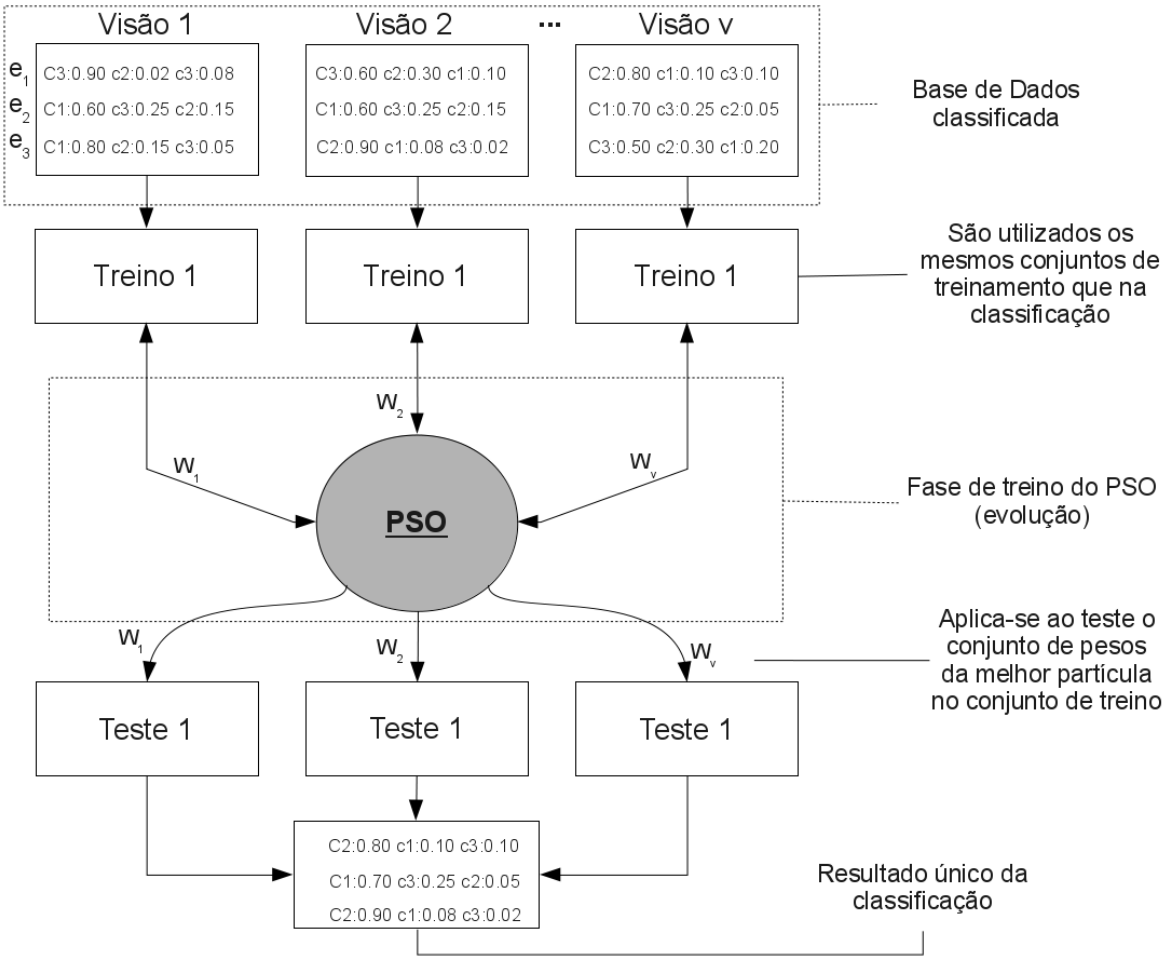
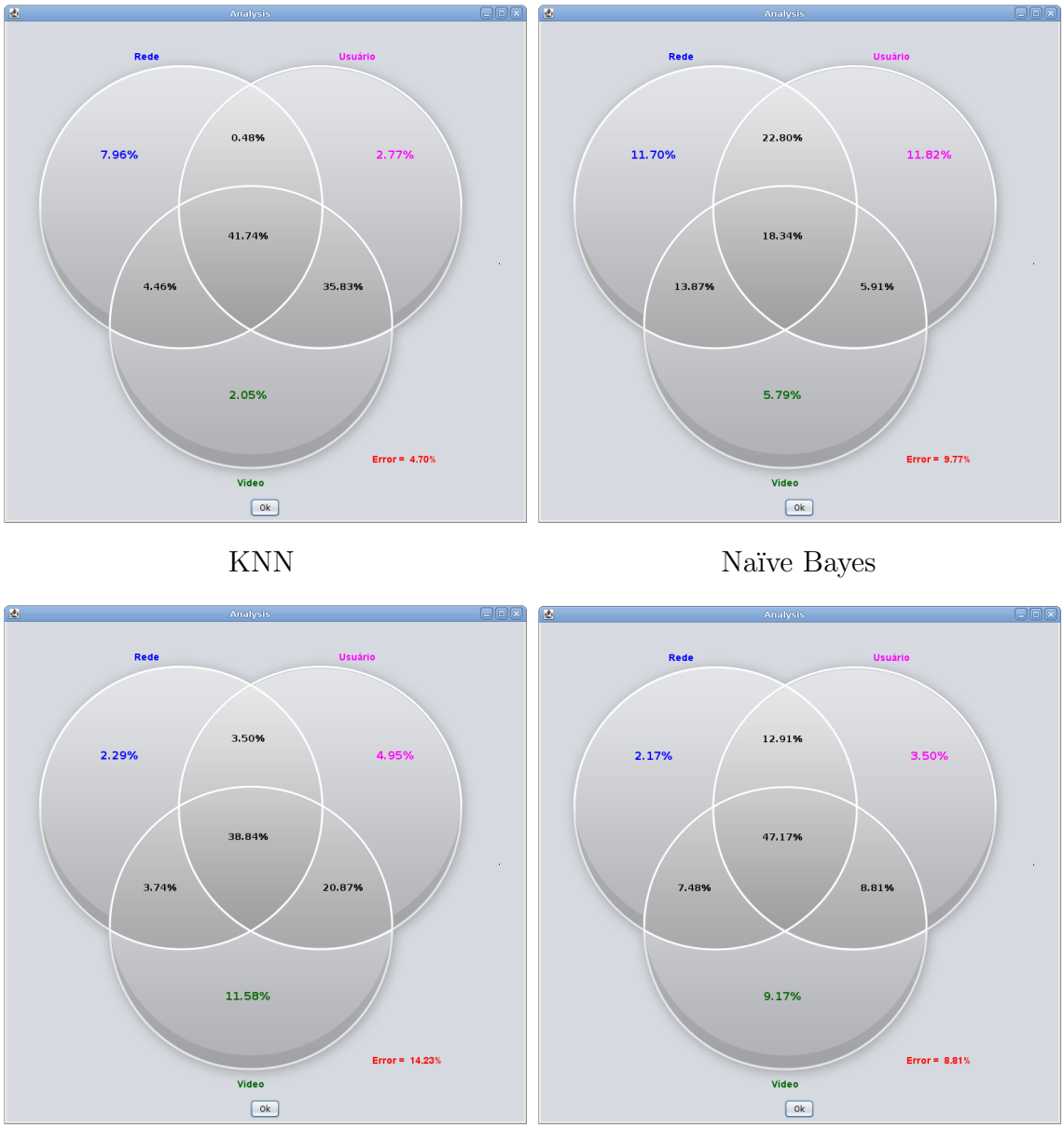



**Figura 6.2.** Segunda e terceira etapa dos experimentos - Combinação das visões de cada umas das bases de dados.

as visões, assim como todas as outras descritas nesta dissertação, estão disponíveis na MultiViL, ferramenta descrita no Capítulo 5.

Com isso, através do diagrama de Venn, é possível saber o rendimentos das visões separadamente ou em relação às outras presentes na base de dados analisada. Nas Figuras 6.3 e 6.4 cada uma das partições do diagrama de Venn representa, em porcentagem, o quanto cada visão acerta sozinha (locais sem intersecção) , em conjunto com outra visão (intersecção de duas visões) ou as três visões em conjunto (intersecção das três visões). A Figura 6.3 apresenta a concordância entre as visões em uma classificação homogênea para a base do YouTube.

Como é possível perceber pela Figura 6.3, o algoritmo que obteve melhor rendimento foi o KNN, com o menor erro. Já o Rocchio foi aquele que obteve o pior rendimento, ou seja, em 14.23% da classificação feita por esse algoritmo, nenhuma das visões acertou a classe das instâncias. Outra observação importante é que em todos os



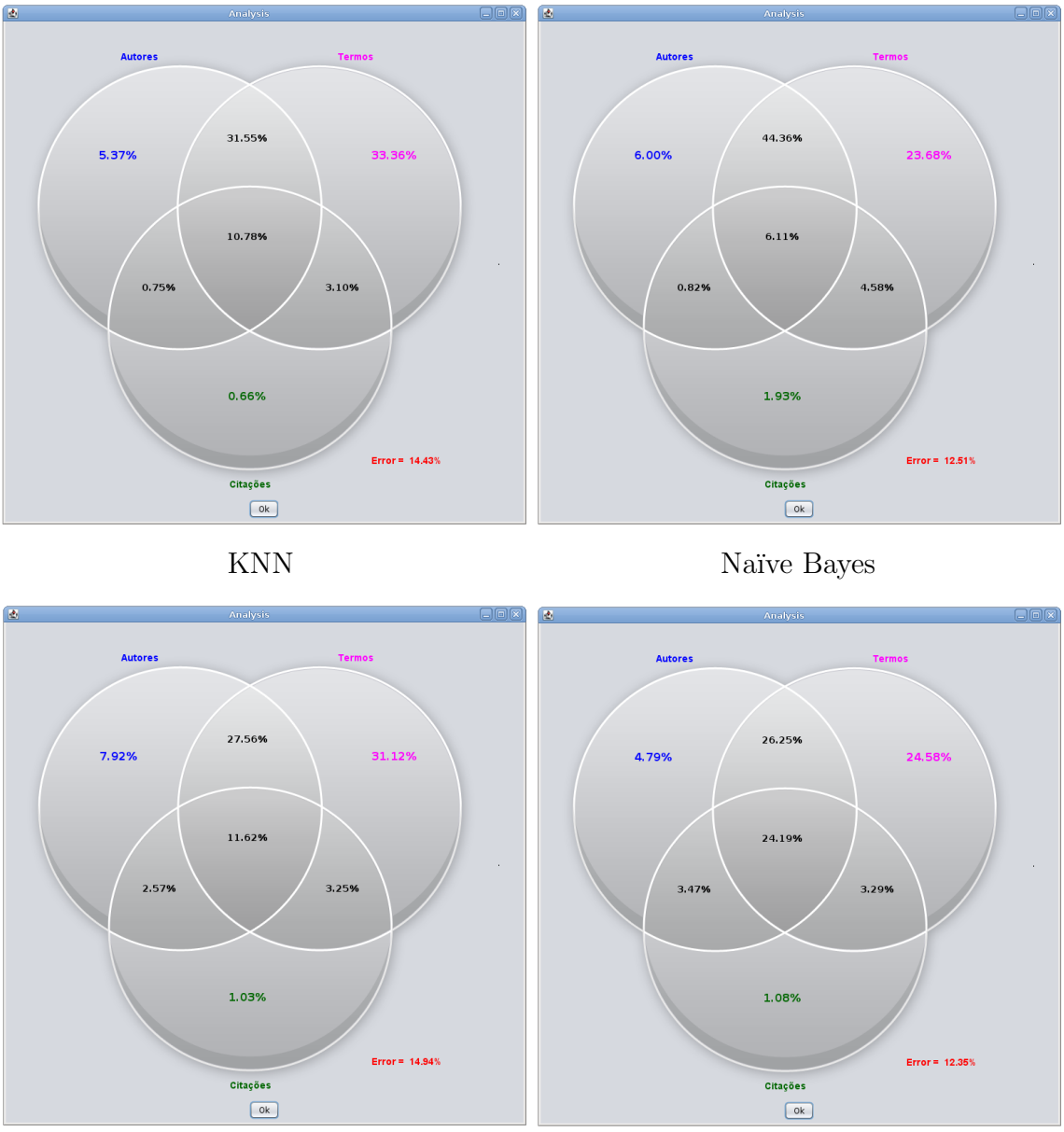

KNN Naïve Bayes

Rocchio SVM-Perf

**Figura 6.3.** Concordância entre as visões da base de dados YouTube.

algoritmos, com exceção do Naïve Bayes, as três visões concordam bastante, como, por exemplo, o SVM-Perf que obteve uma concordância em 47.17% das instâncias. Além disso, em todos os algoritmos, cada visão, independentemente, tem algo a acrescentar à classificação, em uma posterior combinação das visões.

Fazendo uma análise totalmente independente algoritmo/visão, é possível perceber que as visões de Vídeo e Usuário tiveram um melhor rendimento quando classificadas pelo algoritmo KNN. Já a visão de Rede obteve um maior ganho pelo SVM.




KNN Naïve Bayes

Rocchio SVM-Perf

**Figura 6.4.** Concordância entre as visões da base de dados ACM-DL.

Esse rendimento pode ser comprovado pela Tabela 6.2, onde é mostrada a combinação heterogênea dos classificadores.

Em uma análise da base de dados ACM-DL, pela Figura 6.4 é possível perceber que os algoritmos que obtiveram o melhor rendimento foram o Naïve Bayes e o SVM, com os menores erros dentre todos, onde em 12,51% e 12,35%, respectivamente, nenhuma instância foi classificada corretamente. O KNN e o Rocchio tiveram um erro muito próximo, sendo esses piores que os outros dois algoritmos. Aqui, diferentemente

da base de dados do YouTube, as três visões não concordam tanto, devido a características inerentes à base de dados da ACM-DL. A característica mais importante é que a visão de Citações é muito esparsa Já a visão de Termos é a mais robusta. Isso pode ser observado na Figura 6.4, pois em todos os algoritmos, a visão de Citações, quando observada independentemente, contribui pouco para uma futura combinação. Já a visão de Termos é aquela que mais agrega informação, tendo uma grande concordância com a visão de Autores. Da mesma forma que na coleção do YouTube, a ACM-DL pode ter seu rendimento comprovado pela Tabela 6.1, onde é mostrada a combinação heterogênea e todos os algoritmos são também distintos em cada visão. Com isso, temos, como melhor configuração, o Naïve Bayes na visão de Autores, o KNN na visão de Termos e o SVM com um rendimento superior a todos os outros algoritmos na visão de Citações, com exceção do KNN que teve um rendimento equivalente estatisticamente ao SVM.

## 6.4 Resultados sob uma Perspectiva Multi-Visão

Nesta seção são apresentados e avaliados o resultados obtidos pelo PSO ao combinar as visões das bases de dados. O objetivo do PSO é fazer com que as visões não apenas concordem, mas cooperem entre si. Tomando-se como exemplo a base de dados do YouTube, pela Figura 6.3, podemos ver que, em todos os algoritmos de classificação, as visões têm contribuições a serem feitas separadamente, duas a duas, ou as três juntas. O papel do PSO, ao combinar essas visões, é fazer com que a cooperação seja máxima, ou seja, que as visões de fato possam fortalecer a tarefa de classificação, contribuindo com seus respectivos acertos para a classificação final.

### 6.4.1 Análise do Comportamento do PSO

Dentre todos os parâmetros descritos na Seção 3.4, foram utilizados valores padrões da literatura para a maioria, concentrando nossos esforços em encontrar a melhor configuração possível para o número de partículas e iterações. Em testes preliminares, esses dois parâmetros foram variados e os valores que possibilitaram bons resultados em um tempo satisfatório foi com a utilização de 20 partículas e 20 iterações do algoritmo. Isso pode ser comprovado pelas Figuras 6.5 e 6.6, que mostram a dispersão das partículas, no espaço de busca, na primeira iteração, e sua convergência no decorrer das iterações. Para cada configuração do PSO (algoritmo/visão), conforme apresentado nas Tabelas 6.4, 6.5, 6.6 e 6.7, foram feitas 30 execuções com sementes distintas, e o desvio padrão entre as soluções de cada execução é baixo, o que comprova a efetividade do algoritmo. Com relação ao tempo de execução, para a base base da ACM-DL, o PSO-WV demora

cerca de de 40 minutos para cada execução. Já para o PSO-WC o tempo é maior, girando em torno de 60 minutos. Em relação à base de dados do YouTube, esse tempo é menor: o PSO-WV demora para cada execução cerca de 20 minutos e o PSO-WC esse tempo é de cerca de 35 minutos.

Além dos parâmetros número de partículas e quantidade de iterações, os valores de $C_{min}$ e $C_{max}$, que aparecem nas Equações 4.7 e 4.8 e, são relativos ao fator individual e social das partículas, foram definidos como 0.5 e 4.0, respectivamente. Esses valores, como mostrado em [Kennedy et al., 2001], fazem com que as partículas percorram o espaço de busca sistematicamente.

Com relação ao método baseado na teoria de Dempster-Shafer, que é utilizado para comparação com o PSO, o parâmetro de ignorância utilizado foi de 0.1, ou seja, a massa alocada para todo o conjunto restante é de no máximo 10%, conforme apresentado no Capítulo 2. Esse valor foi obtido a partir de diversos teste, comprovando, conforme apresentado por [Bell et al., 2005], que esse é o melhor valor para esse parâmetro.

Analisando as Figuras 6.5 e 6.6 é possível avaliar o comportamento do PSO com relação à sua movimentação no espaço de busca. Aqui, levamos em consideração somente o PSO-WV, pois esse possui somente três dimensões, pesos para cada uma das três visões. Já o PSO-WC possui, para cada visão, no subconjunto da ACM-DL-oito classes, e na base de dados do YouTube - três classes. Como cada base de dados tem três visões, torna-se inviável construir uma figura com tantas dimensões para o PSO-WC, sendo $(3 \times 8)$ para a ACM-DL e $(3 \times 3)$ para o YouTube. As figuras mostram o movimento da nuvem com relação ao PSO-WV, em ambas as bases de dados, e são relativas à combinação heterogênea, i.e., aquela que utiliza os melhores classificadores identificados em cada visão dos dados.

Cada eixo dos gráficos corresponde a uma visão, e cada ponto ilustra a posição da partícula (os pesos atribuídos a cada visão em cada dimensão), na primeira, décima e vigésima iteração do PSO-WV. Para ambas as bases de dados, pode-se observar que, inicialmente, as partículas estão dispersas nos espaço de busca, e com o passar das iterações elas tendem a se mover em direção à melhor solução, como de fato deve acontecer. Note que, ao fim das 20 iterações do algoritmo, para o YouTube, as visões de Usuário e Rede são consideradas mais importantes que a visão de Vídeo, com o respectivos pesos: 0.514498, 0.258022 e 0.227480. Já para a ACM-DL, as visões de Autores e Citações são consideradas mais importantes que a visão de Termos, com o respectivos pesos: 0.320916, 0.380513 e 0.298571. Todos esses pesos são apresentados na Tabela 6.3-configuração heterogênea.

As Figuras 6.7 e 6.8 mostram a evolução da *fitness* das partículas no decorrer das

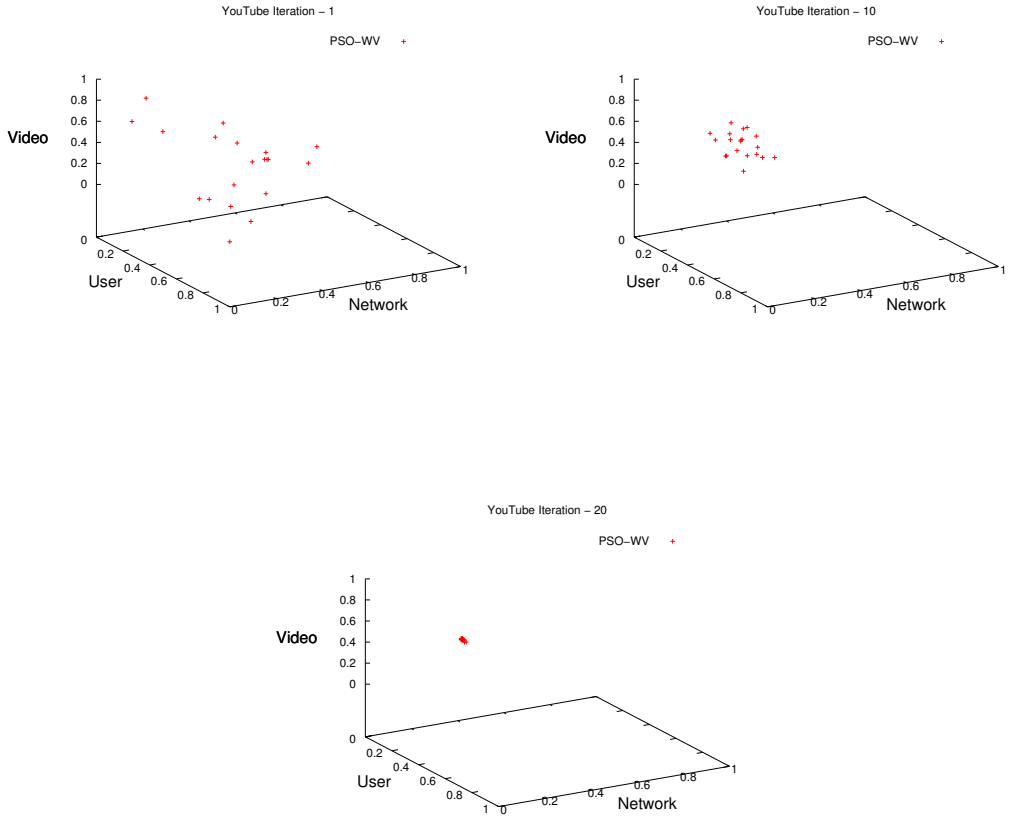



**Figura 6.5.** Movimentação da nuvem para o PSO-WV na base de dados YouTube.

20 iterações, também utilizando o melhor classificador identificado para cada visão. A cada iteração os pesos dos dois algoritmos, PSO-WV e PSO-WC são aplicados às partições de treino e ao final da evolução, os pesos da melhor partícula ($g_{best}$) são aplicados à partição de teste. Em uma primeira análise, é possível perceber que o PSO-WV obteve melhores resultados com relação à base de dados do YouTube (Tabela 6.2). Em contrapartida, o PSO-WC obteve melhores resultados combinando as visões da base de dados da ACM-DL (Tabela 6.1). Isso pode ser explicado pelo fato de que, como a ACM-DL tem oito classes, uma mudança no *ranking* de classes pelo PSO-WC tem menor impacto do que uma mudança em um *ranking* de três classes como no caso do YouTube.

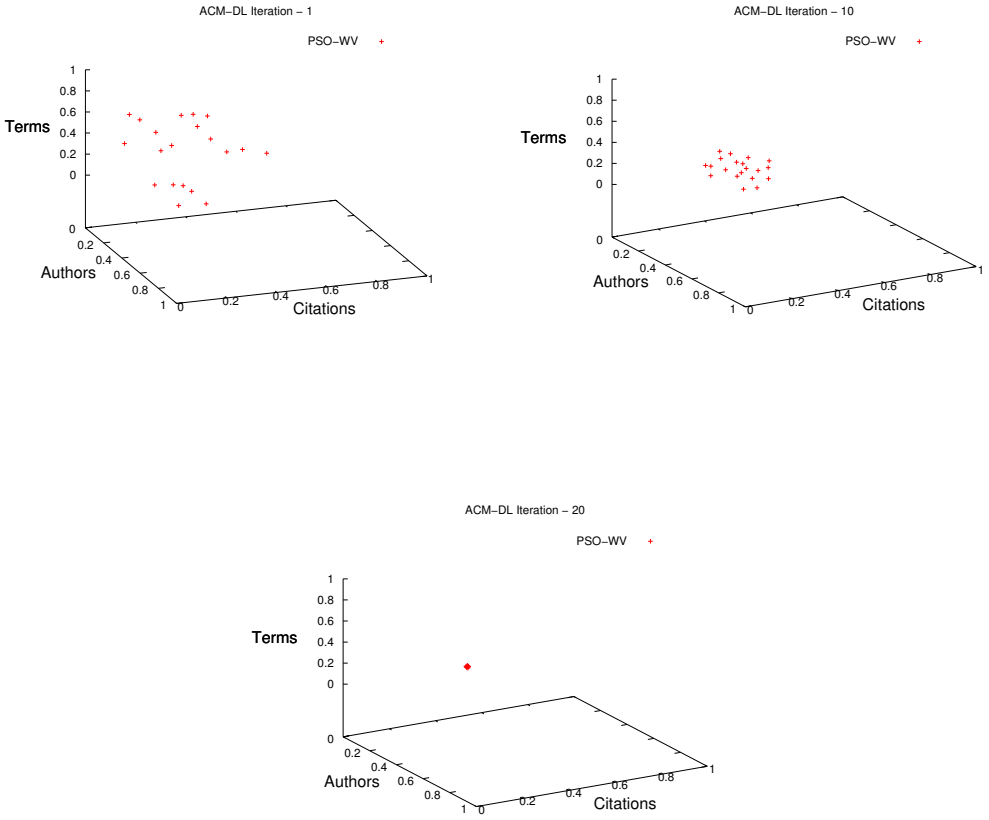



**Figura 6.6.** Movimentação da nuvem para o PSO-WV na base de dados ACM-DL.

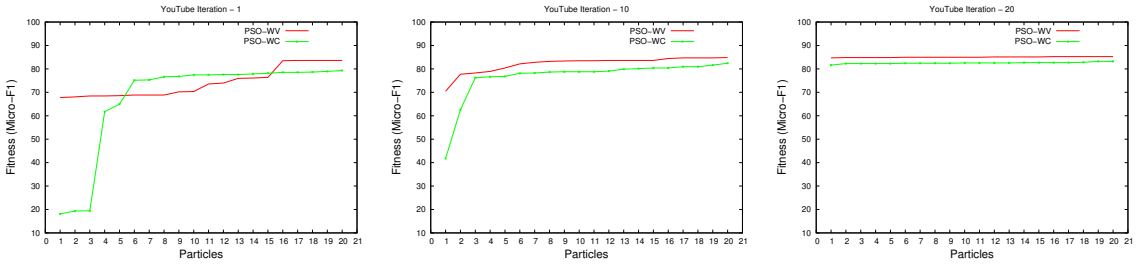



**Figura 6.7.** Evolução da *Fitness* do PSO-WV e PSO-WC na base de dados do YouTube.

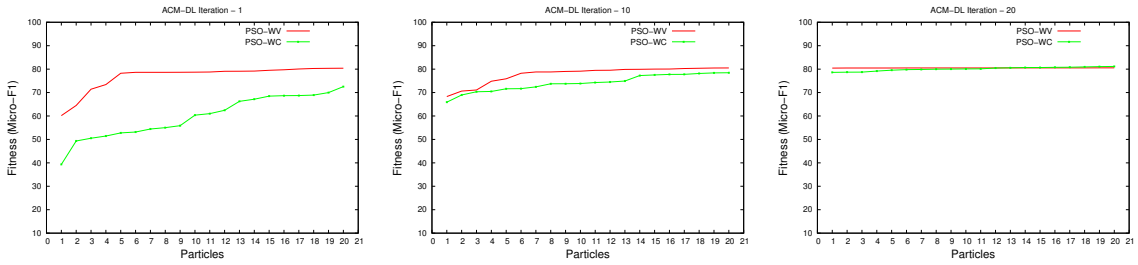



**Figura 6.8.** Evolução da *Fitness* do PSO-WV e PSO-WC na base de dados da ACM-DL.

### 6.4.2 Resultados das Combinações das Visões

Agora, em uma análise dos resultados obtidos pelas combinações do PSO, as Tabelas 6.1 e 6.2 mostram os valores de Micro-F1 obtidos nos experimentos. Para a Tabela 6.1, temos as seguintes abreviações: A = Autores; T = Termos; C = Citações. Já para a Tabela 6.2, as abreviações são as seguintes: R = Rede; U = Usuário; V = Vídeo; Rocc. = Rocchio. Para ambas as tabelas, temos: NB = Naïve Bayes; VM = Voto da Maioria; DS = Dempster-Shafer; BC = *Borda Count*. A primeira coluna dessas tabelas apresenta o algoritmo executado, seguido dos resultados das três visões independentes. As próximas quatro colunas mostram os resultados da execução das três visões juntas, seguidas pelos outros três métodos de combinação utilizados para comparação com o PSO: um esquema de voto da maioria, a teoria da evidência de Dempster-Shafer e o *Borda Count*. As últimas duas colunas trazem os resultados obtidos pelos PSO-WV e PSO-WC.

Como descrito anteriormente, todos os valores obtidos e apresentados nas tabelas deste capítulo foram obtidos baseado em uma validação cruzada com 10 partições. Os resultados dispostos nas Tabelas 6.1 e 6.2 foram comparados usando um Teste-t de *Student* com 95% de confiança. Os resultados dos experimentos são apresentados nas referidas tabelas de duas formas. Primeiro foi feita uma comparação entre os resultados do PSO-WV e PSO-WC. As células dos PSOs foram então marcadas com (•), indicando que os resultados são estatisticamente equivalentes, ou com um (▲), indicando que os resultados nas células marcadas são estatisticamente melhores que o do PSO da célula vizinha. Em uma segunda análise, o PSO marcado na primeira análise é comparado com os outros métodos de combinação, marcando essas células com uma terceira notação (▼), a qual indica que o referido resultado é estatisticamente inferior ao obtido pelo PSO.

A partir disso, uma primeira análise, com relação aos resultados, pode ser feita na base de dados da ACM-DL (Tabela 6.1). Nesse caso, os resultados para o PSO-WV são estatisticamente melhores que o PSO-WC em três casos: a combinação homogênea do KNN, Rocchio e SVM, e estatisticamente equivalente para os resultados obtidos pelo Naïve Bayes. No entanto, quando uma combinação heterogênea é feita, i.e., a combinação dos melhores algoritmos em sua respectiva visão (mostrado destacado), o PSO-WC apresenta os melhores resultados dentre todos, com um Micro-F1 de 81.96. Isso mostra a clara vantagem do aprendizado multi-visão: utilizar diferentes classificadores para diferentes conjuntos disjuntos de atributos é melhor do que todos atributos agrupados em um único conjunto de dados. Pode-se observar na Tabela 6.1 que a visão de Termos é a mais robusta e a de Citações a menos robusta quando consideradas independentemente. Além disso, deve ser levado em consideração que as visões de Autores e Citações são bastante esparsas.

**Tabela 6.1.** Resultados de Micro-F1 para a base de dados da ACM-DL.

| | **Visões** | | | **Métodos de Combinação** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Algorit.** | **A** | **T** | **C** | **A+T+C** | **VM** | **DS** | **BC** | **PSO-WV** | **PSO-WC** |
| KNN | 57.78 | **78.79** | 33.90 | 80.77● | 79.13▼ | 78.56▼ | 71.62▼ | 81.22▲ | 79.85 |
| NB | **57.30** | 78.73 | 13.44 | 68.63▼ | 79.88● | 80.19● | 70.45▼ | 80.72● | 80.45● |
| Rocchio | 49.69 | 73.54 | 18.47 | 74.63▼ | 75.48▼ | 59.19▼ | 66.60▼ | 76.62▲ | 75.29 |
| SVM | 58.70 | 78.31 | **32.03** | 79.09● | 60.22▼ | 66.62▼ | 71.51▼ | 79.15▲ | 59.20 |
| **Heterog.** | **NB** | **KNN** | **SVM** | — | 59.46▼ | 77.47▼ | 71.65▼ | 80.77 | **81.96**▲ |

Já em uma análise da base de dados do YouTube (Tabela 6.2), o PSO-WC é estatisticamente melhor que o PSO-WV em dois casos, combinando os resultados provenientes dos algoritmo KNN e Naïve Bayes, enquanto que nos outros o PSO-WV é superior. No entanto, o melhor resultado dentre todos foi obtido combinando as visões classificadas pelo KNN com um Micro-F1 de 86.37, que é estatisticamente superior à classificação pelo KNN quando todas as visões estão aglutinadas em um único conjunto de dados, cujo Micro-F1 foi de 84.92. Em conclusão, em ambas bases de dados o melhor resultado foi obtido pelo PSO-WC, que pondera não somente as visões, mas todo o conjunto de classes em cada visão. Essa abordagem tem a vantagem de identificar qual classificador e visões são melhores para a predição das classes.

**Tabela 6.2.** Resultados de Micro-F1 para a base de dados do YouTube.

| | **Visões** | | | **Métodos de Combinação** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Algorit.** | **R** | **U** | **V** | **R+U+V** | **VM** | **DS** | **BC** | **PSO-WV** | **PSO-WC** |
| KNN | 77.57 | 80.82 | **84.08** | 84.92▼ | 81.91▼ | 81.76▼ | 71.26▼ | 83.96 | **86.37**▲ |
| NB | 66.71 | 58.87 | 43.90 | 40.54▼ | 66.67▼ | 64.98▼ | 59.66▼ | 66.10 | 74.18▲ |
| Rocc. | 48.38 | **68.17** | 75.04 | 72.26▼ | 69.13▼ | 54.11▼ | 52.54▼ | 73.35▲ | 70.69 |
| SVM | **69.71** | 72.37 | 72.63 | 47.75▼ | 69.12▼ | 71.50▼ | 70.77▼ | 76.12▲ | 74.43 |
| **Heterog.** | **SVM** | **Rocc.** | **KNN** | — | 77.93▼ | 66.79▼ | 74.76▼ | **84.56**▲ | 83.71 |

Todas as combinações possíveis, no contexto heterogêneo, onde cada algoritmo classifica apenas uma única visão, são apresentadas nas Tabelas 6.4 e 6.5, para a base de dados do YouTube, com resultados relativos ao PSO-WV e PSO-WC respectivamente. Já nas Tabelas 6.6 e 6.7 são apresentados os resultado que referenciam, respectivamente, os PSO-WV e PSO-WC para a base de dados da ACM-DL. Em Todas essas tabelas são apresentados os valores de Micro-F1, pois essa foi a métrica utilizada para o cálculo da *fitness* dos PSOs. Nessas tabelas citadas, que contém todas as combinação de quatro algoritmo em três visões, é possível perceber que, para os melhores resultados, os valores são muito próximos e, em muitos existe uma equivalência estatística. Com isso, para uma efetiva comparação entre os dois PSO, coletou-se resultados que foram estatisticamente melhores, seguindo a mesma configuração algoritmo/visão para o PSO-WC e PSO-WV.

Agora, uma última análise, baseada nos pesos, pode fornecer algum entendimento sobre a atuação do PSO. Assim, como o PSO-WC tem um grande número de pesos, a discussão desses torna-se não trivial, principalmente para a base de dados da ACM-DL. Portanto, será focada uma análise dos pesos gerados pelo PSO-WV. Observando-se os resultados da Tabela 6.3, para a ACM-DL, a visão de Citações recebe, na maioria das vezes, um peso menor que os das outras visões. Já a visão de Termos foi considerada pelo Rocchio duas vezes melhor que as outras duas. O mesmo aconteceu com relação ao KNN, onde as visões de Autores e Citações obtiveram praticamente o mesmo peso e a visão de Termos obteve um peso de quase 50%. Isso é devido à forma como o algoritmo de classificação trabalha, não apresentando um bom rendimento ao trabalhar com dados esparsos. Em contraste, para o Naïve Bayes, metade do peso foi atribuído à visão de Autores, seguido pela de Termos. O SVM foi o único classificador que obteve, em todas as visões, pesos similares.

Com isso, uma diferente análise com relação aos pesos pode ser feita. Observando-se os resultados isolados das visões de Termos na ACM-DL e Vídeo no YouTube, essas duas, em suas respectivas bases de dados, obtiveram os melhores resultados, mas uma pergunta pode ser feita. Por que o PSO não atribuiu os maiores pesos a essas visões? Isso se deve ao fato de que o PSO, ao longo se sua evolução (iterações), identifica a contribuição de cada visão para a tarefa de classificação, e ajusta essa contribuição pelos pesos atribuídos a cada visão. Dessa forma, nem sempre uma visão que isoladamente obteve um melhor rendimento ao ser combinada irá receber o maior peso. Isso não acontece, pois, como observado nas Figuras 6.3 e 6.4, todas as visões têm algo a contribuir para a classificação, mesmo que em menor escala como, por exemplo, a visão de Citação. Se uma visão que obteve o melhor resultado receber um peso muito maior do que as outras, após a combinação, a classificação tem uma

grande possibilidade de ficar praticamente com o mesmo rendimento da melhor visão. Da mesma forma, o PSO-WV em uma combinação heterogênea, como observado na Tabela 6.3, atribui à visão de Vídeo o menor dentre todos os pesos para as visões da base de dados do YouTube. Isoladamente, esse foi o classificador/visão com o melhor resultado.

Em uma análise relativa à base de dados do YouTube, é interessante mostrar o contraste dos pesos entre os diferentes classificadores. Pela Tabela 6.3, nota-se que, enquanto a visão de Rede é a mais importante para o KNN, para o Rocchio isso é ignorado, sendo que, esse último, não obteve uma boa performance nessa visão. O Naïve Bayes foi o classificador que obteve os pesos mais balanceados entre as visões, e isso representou o pior resultado pelo PSO-WV.

**Tabela 6.3.** Pesos para as visões nas bases de dados do YouTube e ACM-DL.

| **Classificadores** | **Visões** | | | | | |
|---|---|---|---|---|---|---|
| | **ACM-DL** | | | **YouTube** | | |
| | **Autores** | **Termos** | **Citações** | **Rede** | **Usuário** | **Vídeo** |
| KNN | 0.266029 | 0.471674 | 0.262297 | 0.467758 | 0.206725 | 0.325516 |
| SVM | 0.356605 | 0332062 | 0.311333 | 0.293970 | 0.371428 | 0.334602 |
| Naïve Bayes | 0.509747 | 0.347285 | 0.142968 | 0.364758 | 0.370820 | 0.264423 |
| Rocchio | 0.190283 | 0.560280 | 0.249437 | 0.031204 | 0.479448 | 0.489348 |
| **Heterogêneo** | 0.320916 | 0.298571 | 0.380513 | 0.258022 | 0.514498 | 0.227480 |

Por fim, em uma análise comparativa, entre os cinco métodos de combinação e a classificação em que todas as visões estão aglutinadas em uma única base de dados, é possível perceber pelas Tabelas 6.1 e 6.2 que, tanto o PSO-WV quanto o PSO-WC são dois métodos mais eficazes, nos dois contextos aqui estudados, do que os demais métodos de combinação de classificadores aqui utilizados, que são base para diversos trabalhos existentes na literatura, como aqueles citados nas Tabelas 2.1, 2.2 e 3.1. Dentre todos os experimentos apresentados, houve, em somente um caso, equivalência estatística entre algum dos PSOs e alguns dos outros métodos de combinação, que foi com relação à base de dados da ACM-DL quando rotulada pelo Naïve Bayes. Em todos os demais casos, o PSO-WV e/ou o PSO-WC foram estatisticamente superior aos demais métodos de combinação de classificadores aqui utilizados.

**Tabela 6.4.** Todas as combinações dos algoritmos nas visões (R: Rede, U: Usuário, V: Vídeo) para a base de dados do YouTube. São reportados os valores de Micro-F1 obtidos pelo **PSO-WV**.

| | **R-U-V** | **R-V-U** | **U-R-V** | **U-V-R** | **V-R-U** | **V-U-R** |
|---|---|---|---|---|---|---|
| **SVM -Naïve Bayes -Rocchio** | 71.23 | 74.68 | 73.58 | 70.08 | 73.77 | 72.08 |
| **SVM -Naïve Bayes -KNN** | 83.72 | 79.74 | 84.08 | 76.91 | 78.29 | 75.88 |
| **SVM -Rocchio -KNN** | **84.56** | 78.65 | 83.90 | 75.82 | 80.10 | 78.05 |
| **Naïve Bayes -Rocchio -KNN** | 82.69 | 76.97 | 81.66 | 75.51 | 79.62 | 75.22 |

**Tabela 6.5.** Todas as combinações dos algoritmos nas visões (R: Rede, U: Usuário, V: Vídeo) para a base de dados do YouTube. São reportados os valores de Micro-F1 obtidos pelo **PSO-WC**.

| | **R-U-V** | **R-V-U** | **U-R-V** | **U-V-R** | **V-R-U** | **V-U-R** |
|---|---|---|---|---|---|---|
| **SVM-Naïve Bayes-Rocchio** | 73.23 | 79.49 | 72.50 | 75.99 | 75.40 | 78.17 |
| **SVM-Naïve Bayes-KNN** | 73.23 | 81.06 | 80.94 | 79.25 | 79.74 | 77.69 |
| **SVM-Rocchio-KNN** | **83.71** | 77.32 | 81.55 | 74.66 | 81.06 | 80.22 |
| **Naïve Bayes-Rocchio-KNN** | 80.82 | 69.48 | 81.18 | 71.42 | 80.82 | 80.58 |

**Tabela 6.6.** Todas as combinações dos algoritmos nas visões (A: Autores, C: Citações, T: Termos) para a base de dados da ACM-DL. São reportados os valores de Micro-F1 obtidos pelo **PSO-WV**.

| | **A-T -C** | **A-C-T** | **T-A-C** | **T-C-A** | **C-A-T** | **C-T-A** |
|---|---|---|---|---|---|---|
| **SVM -Naïve Bayes -Rocchio** | 80.14 | 73.97 | 80.33 | 79.78 | 77.55 | 80.87 |
| **SVM -Naïve Bayes -KNN** | 80.26 | 79.24 | 80.33 | 79.85 | **80.77** | 80.89 |
| **SVM -Rocchio -KNN** | 75.08 | 79.78 | 80.09 | 80.12 | 80.81 | 76.92 |
| **Naïve Bayes -Rocchio -KNN** | 77.92 | 81.10 | 80.99 | 81.02 | 80.81 | 76.93 |

**Tabela 6.7.** Todas as combinações dos algoritmos nas visões (A: Autores, C: Citações, T: Termos) para a base de dados da ACM-DL. São reportados os valores de Micro-F1 obtidos pelo **PSO-WC**.

| | **A-T -C** | **A-C-T** | **T-A-C** | **T-C-A** | **C-A-T** | **C-T-A** |
|---|---|---|---|---|---|---|
| **SVM -Naïve Bayes -Rocchio** | 79.39 | 65.15 | 69.48 | 65.75 | 75.14 | 80.65 |
| **SVM -Naïve Bayes -KNN** | 79.34 | 79.39 | 69.76 | 66.17 | **81.96** | 80.17 |
| **SVM -Rocchio -KNN** | 74.84 | 79.76 | 71.04 | 72.16 | 79.25 | 74.60 |
| **Naïve Bayes -Rocchio -KNN** | 76.40 | 81.14 | 80.57 | 80.29 | 78.93 | 72.32 |

Com isso, fica evidente a eficácia dos métodos PSO propostos, valendo a ressalva de que, o PSO-WC obteve, para ambas as bases de dados utilizadas, os melhores resultados dentre todos. Esse fato confirma que, levar em consideração todo o *ranking* de classe de cada algoritmo de classificação em cada visão dos dados é um fator importante para a combinação de algoritmos de classificação, distintos ou não, pois, além de possibilitar que o erro inerente a cada classificador seja capturado, com o PSO, é possível, além de combinar os algoritmos, verificar qual visão dos dados melhor representam a base de dados. Atrelado a isso tudo têm-se a MulTiViL, uma ferramenta que possibilita diferentes formas de classificação e posterior análise da iteração entre os algoritmo e/ou métodos de combinação.

# Capítulo 7

# Conclusões

Este trabalho apresentou um novo método que utiliza uma abordagem supervisionada para resolver o problema da classificação automática com aprendizado multi-visão. O problema da classificação automática tem uma enorme importância prática, dado o grande volume de dados disponíveis na *Web* e, dos quais podem ser extraídas visões. Entre esses dados *Web*, destacamos os obtidos através das redes sociais, que nos últimos anos têm alcançado um número grandioso de usuários, provendo diversas visões dos dados.

Dado o contexto do aprendizado multi-visão, este trabalho propôs um algoritmo de Otimização por Nuvem de Partículas (PSO) para ponderar visões em aplicações de classificação multi-visão. Foram introduzidas duas versões do algoritmo. A primeira é utilizada para combinar as decisões advindas de algoritmos de classificação, distintos ou não, em diferentes visões dos dados. A segunda considera, além das visões, o quão bom os algoritmos/visões são para rotular as instâncias em cada classe.

Ambas as abordagem - PSO-WV e PSO-WC, foram experimentalmente testados em duas bases de dados com três visões cada - ACM-DL e YouTube. Quatro diferentes algoritmos de classificação, com diferentes características, foram executados separadamente em cada visão, e então combinados pelo PSO. Os resultados foram comparados com a classificação onde todas as visões estão juntas em uma única base de dados, com um esquema de voto da maioria, um método baseado na teoria de Dempster-Shafer e o algoritmo de *Borda Count*, mostrando que o PSO obtém resultados estatisticamente superiores a esses métodos.

Contudo, o PSO-WC foi o que obteve os melhores resultados, o que enfatiza a importância de se levar em consideração, além das visões, o *ranking* de classes gerado pelos algoritmos de classificação. Dessa forma, é possível que o erro inerente a cada algoritmo de classificação seja capturado, provendo uma classificação com um melhor

rendimento, após a combinação das visões.

Além disso, como consolidação deste trabalho, foi disponibilizada uma ferramenta construída especificamente para realizar o aprendizado multi-visão no contexto da classificação automática. A MultiViL vem para agregar conhecimento e servir como uma importante alternativa em pesquisas tanto no contexto da classificação mono-visão, quanto no contexto multi-visão. Isso se deve pelo fato de que a MultiViL aglutina todos os métodos de combinação de classificadores supracitados, além de ser totalmente auto-contida, i.e., todos os algoritmos utilizados estão acessíveis pela ferramenta.

## 7.1 Trabalhos Futuros

Como trabalhos futuros, seria interessante aplicar essa abordagem em outras bases de dados, como aquelas obtidas a partir de outros contexto de redes sociais, onde a presença de diferentes visões é intuitiva. Outro ponto de pesquisa seria explorar outro algoritmo de computação natural, como o de programação genética, para resolver esse mesmo problema e poder comparar seus resultados com os aqui obtidos, permitindo verificar se essa poderia ser uma outra boa alternativa para o problema da combinação de diferentes visões de uma base de dados.

Com relação à MultiViL, como versões futuras e continuidade das pesquisas em multi-visão, pretende-se incorporar à ferramenta o aprendizado de máquina semi-supervisionado e o não-supervisionado, fechando assim o ciclo de classificação. Em uma mudança de contexto, métodos de agrupamentos, em multi-visão, podem ser estudados e futuramente adicionados à ferramenta.

Seria interessante, também, a comparação deste método com outras abordagens de aprendizado supervisionado. Além disso, testar o PSO proposto em um contexto semi-supervisionado, pode ser de extrema importância para os avanços em pesquisas ligadas à multi-visão, podendo essa se tornar uma alternativa totalmente distinta daquelas baseadas, substancialmente, no algoritmo *Co-training*.

Por fim, estudos mais detalhados de como dividir uma base de dados única em diferentes visões é de extrema relevância, dada a premissa de que várias visões podem levar a resultados melhores que uma única.

# Referências Bibliográficas


Al-Ani, A. & Deriche, M. (2002). A new technique for combining multiple classifiers using the Dempster-Shafer theory of evidence. *J. Artif. Int. Res.*, 17:333–361.

Baeza-Yates, R. A. & Ribeiro-Neto, B. (2010). *Modern Information Retrieval: The Concepts and Technology behind Search*. Pearson Education (US), New Jersey/US, 2nd revised edição.

Banzhaf, W.; Francone, F. D.; Keller, R. E. & Nordin, P. (1998). *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Baxter, J. (1998). *Theoretical models of learning to learn*, pp. 71–94. Kluwer Academic Publishers, Norwell, MA, USA.

Bell, D.; Guan, J. & Bi, Y. (2005). On combining classifier mass functions for text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 17(10):1307 – 1319.

Benevenuto, F.; Rodrigues, T.; Almeida, V.; Almeida, J. & Gonçalves, M. (2009). Detecting spammers and content promoters in online video social networks. In *Proc. of Int'l ACM SIGIR*, Boston, MA, USA.

Bi, Y.; Guan, J. & Bell, D. (2008). The combination of multiple classifiers using an evidential reasoning approach. *Artif. Intell.*, 172:1731–1751.

Bi, Y.; McClean, S. & Anderson, T. (2006). On combining multiple classifiers using an evidential approach. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pp. 324–329. AAAI Press.

Black, D. (1958). *The Theory of Committees and Elections*. Springer, 1 edição.

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with Co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pp. 92–100, New York, NY, USA. ACM.

Brefeld, U.; Buscher, C. & Scheffer, T. (2005). Multi-view discriminative sequential learning. In *Proc. of the European Conference on Machine Learning, 2005*, pp. 60–71.

Brown, G. & Kuncheva, L. (2010). "good" and "bad" diversity in majority vote ensembles. In El Gayar, N.; Kittler, J. & Roli, F., editores, *Multiple Classifier Systems*, volume 5997 of *Lecture Notes in Computer Science*, pp. 124–133. Springer Berlin / Heidelberg.

Browne, M. W. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, 44(1):108 – 132.

Cao, Q. & Liu, Y. (2010). A KNN classifier with PSO feature weight learning ensemble. In *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on*, pp. 110 –114.

Carpineto, C.; Michini, C. & Nicolussi, R. (2009). A concept lattice-based kernel for svm text classification. In Ferré, S. & Rudolph, S., editores, *Formal Concept Analysis*, volume 5548 of *Lecture Notes in Computer Science*, pp. 237–250. Springer Berlin / Heidelberg. 10.1007/978-3-642-01815-2_18.

Chen, Z. & Haykin, S. (2002). On different facets of regularization theory. *Neural Comput.*, 14:2791–2846.

Christoudias, C.; Urtasun, R. & Darrell, T. (2008). Multi-view learning in the presence of view disagreement. In *Proc. of the 24th Annual Conf. on Uncertainty in Artificial Intelligence*.

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Culp, M. & Michailidis, G. (2009). A co-training algorithm for multi-view data with applications in data fusion. *Journal of chemometrics*, 23:294–303.

de Sa, V. R. (1993). Learning classification with unlabeled data. In Cowan, J. D.; Tesauro, G. & Alspector, J., editores, *Proc. NIPS'93, Neural Information Processing Systems*, pp. 112–119, San Francisco, CA. Morgan Kaufmann Publishers.

Denis, F.; Laurent, A.; Gilleron, R. & Tommasi, M. (2003). Text classification and co-training from positive and unlabeled examples. In *Proc. of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*, pp. 80–87.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pp. 1–15, London, UK. Springer-Verlag.

Dorigo, M. & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95: Proc. of the 6th International Symposium on Micro Machine and Human Science, 1995.*, pp. 39 –43.

Eberhart, R.; Simpson, P. & Dobbins, R. (1996). *Computational intelligence PC tools.* Academic Press Professional, Inc., San Diego, CA, USA.

Escalante, H. J.; Montes, M. & Sucar, L. E. (2009). Particle swarm model selection. *J. Mach. Learn. Res.*, 10:405–440.

Evans, H. & Zhang, M. (2008). Particle swarm optimisation for object classification. In *Image and Vision Computing, 2008. IVCNZ 2008. 23rd International Conference*, pp. 1 –6.

Fujibuchi, W. & Kato, T. (2007). Classification of heterogeneous microarray data by maximum entropy kernel. *BMC Bioinformatics*, 8:267+.

Ghani, R. (2002). Combining labeled and unlabeled data for multiclass text categorization. In *ICML '02: Proc. of the 19th International Conference on Machine Learning*, pp. 187–194, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gromisz, M. & Zadrożny, S. (2010). Combining the results in pairwise classification using Dempster-Shafer theory: A comparison of two approaches. In *Artificial Intelligence and Soft Computing*, volume 6113 of *Lecture Notes in Computer Science*, pp. 339–346. Springer Berlin / Heidelberg.

Guo, G.; Neagu, D.; Huang, X. & Bi, Y. (2006). An effective combination of multiple classifiers for toxicity prediction. In Wang, L.; Jiao, L.; Shi, G.; Li, X. & Liu, J., editores, *Fuzzy Systems and Knowledge Discovery*, volume 4223 of *Lecture Notes in Computer Science*, pp. 481–490. Springer Berlin / Heidelberg.

Guo, H. & Viktor, H. L. (2005). Mining relational databases with multi-view learning. In *MRDM '05: Proc. of the 4th international workshop on Multi-relational mining*, pp. 15–24, New York, NY, USA. ACM.

Guo, H. & Viktor, H. L. (2006). Mining relational data through correlation-based multiple view validation. In *KDD '06: Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 567–573, New York, NY, USA. ACM.

Guo, S. & Gao, B. (2009). Path-planning optimization of underwater microrobots in 3-d space by PSO approach. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pp. 1655 –1620.

Hansen, L. & Salamon, P. (1990). Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10):993 –1001.

Hinrichs, C.; Singh, V.; Xu, G. & Johnson, S. C. (2011). Predictive markers for ad in a multi-modality framework: An analysis of mci progression in the adni population. *NeuroImage*, 55(2):574–589.

Holden, N. & Freitas, A. A. (2008). A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *J. Artif. Evol. App.*, 2008:2:1–2:11.

Horng, M.-F.; Chen, Y.-T.; Chu, S.-C.; Pan, J.-S. & Liao, B.-Y. (2010). An extensible particles swarm optimization for energy-effective cluster management of underwater sensor networks. In Pan, J.-S.; Chen, S.-M. & Nguyen, N., editores, *Computational Collective Intelligence. Technologies and Applications*, volume 6421 of *Lecture Notes in Computer Science*, pp. 109–116. Springer Berlin / Heidelberg.

Hovelynck, M. & Chidlovskii, B. (2010). Multi-modality in one-class classification. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 441–450. ACM.

Huynh, V.-N.; Nguyen, T. T. & Le, C. A. (2010). Adaptively entropy-based weighting classifiers in combination using Dempster-Shafer theory for word sense disambiguation. *Comput. Speech Lang.*, 24:461–473.

Junior, Z. C.; Pappa, G.; Arcanjo, F.; Jr., W. M. & Gonçalves, M. A. (2010). Combinando multi-visões através de um algoritmo de nuvens de partículas. In *SBBD 2010Short Papers*.

Junior, Z. C. & Pappa, G. L. (2011). A PSO algorithm for improving multi-view classification. In *IEEE CEC Conference*, New Orleans, EUA.

Junior, Z. C.; Pappa, G. L.; de L. Arcanjo, F.; Albinati, J.; Jr., W. M.; Gonçalves, M. A. & Benevenuto, F. (2011). MultiViL: A Tool for Multi-View Classification. In *ACM SIGIR Conference on Research - Demo Session*, Beijing, China. Artigo submetido.

Katakis, I.; Tsoumakas, G. & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, 22:371–391. 10.1007/s10115-009-0206-2.

Kausar, A.; Ishtiaq, M.; Jaffar, M. A. & Mirza, A. M. (2010). Optimization of ensemble based decision using PSO. In *Proceedings of the World Congress on Engineering*, WCE '10.

Kennedy, J.; Eberhart, R. C. & Shi, Y. (2001). *Swarm Intelligence (The Morgan Kaufmann Series in Evolutionary Computation)*. Morgan Kaufmann, 1st edição.

Khan, N.; Rauf Baig, A. & Iqbal, M. (2010). A new discrete PSO for data classification. In *Information Science and Applications (ICISA), 2010 International Conference on*, pp. 1 –6.

Khurshid, A. & Gokhale, A. (2009). Classification system for digital signal types using neuro fuzzy system and PSO. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 373 –378.

Kohavi, R. & Sommerfield, D. (1995). Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proc. of the 1st Int. Knowledge Discovery and Data Mining (KDD-95)*, p. 192–197, Montreal, Canada.

Laguna, V. & Lopes, A. (2009). A multi-view approach for semi-supervised scientific paper classification. In *In: XXIV SBBD- V Workshop em Algoritmos e Aplicações de Mineração de Dados*, pp. 1–10, Fortaleza, Brasil. WAAMD.

Langbehn, H. R.; Ricci, S. M. R.; Gonçalves, M. A.; Almeida, J. M.; Pappa, G. L. & Benevenuto, F. (2010). A multi-view approach for detecting non-cooperative users in online video sharing systems. *JIDM*, 1(3):313–328.

Lewis, D. (1998). Naïve (bayes) at forty: The independence assumption in information retrieval. In Nédellec, C. & Rouveirol, C., editores, *ECML-98: Machine Learning*,

volume 1398 of *Lecture Notes in Computer Science*, pp. 4–15. Springer Berlin / Heidelberg. 10.1007/BFb0026666.

Lodhi, H.; Saunders, C.; Shawe-Taylor, J.; Cristianini, N. & Watkins, C. (2002). Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419--444.

McCallum, A. & Nigam, K. (1998). A comparison of event models for naïve bayes text classification. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence*, pp. 41 – 48.

Michalewicz, Z. & Fogel, D. B. (2000). *How to solve it: Modern Heuristics*. Springer.

Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In *Proc. of CoNLL-2004*, pp. 33 – 40. Boston, MA, USA.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Education (ISE Editions).

Mohemmed, A.; Johnston, M. & Zhang, M. (2009). Particle swarm optimization based multi-prototype ensembles. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pp. 57–64. ACM.

Muslea, I.; Minton, S. & Knoblock, C. A. (2002). Active + semi-supervised learning = robust multi-view learning. In *ICML '02: Proc. of the Nineteenth International Conference on Machine Learning*, pp. 435–442, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ng, E.; Lim, M.; Maul, T. & Lai, W. (2009). Investigations into particle swarm optimization for multi-class shape recognition. In Köppen, M.; Kasabov, N. & Coghill, G., editores, *Advances in Neuro-Information Processing*, volume 5507 of *Lecture Notes in Computer Science*, pp. 599–606. Springer Berlin / Heidelberg.

Ng, V. & Cardie, C. (2003). Weakly supervised natural language learning without redundant views. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pp. 94–101, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nigam, K.; McCallum, A. K.; Thrun, S. & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Mach. Learn.*, 39(2-3):103–134.

Perdisci, R.; Ariu, D.; Fogla, P.; Giacinto, G. & Lee, W. (2009). Mcpad: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53(6):864–881. Traffic Classification and Its Applications to Modern Networks.

Perez, C. A.; Cament, L. A. & Castillo, L. E. (2011). Methodological improvement on local gabor face recognition based on feature selection and enhanced borda count. *Pattern Recognition*, 44(4):951–963.

Quost, B.; Masson, M.-H. & Denoeux, T. (2011). Classifier fusion in the Dempster-Shafer framework using optimized t-norm based combination rules. *International Journal of Approximate Reasoning*, 52(3):353 – 374. Dependence Issues in Knowledge-Based Systems.

Rani, C. & Deepa, S. (2010). Design of optimal fuzzy classifier system using particle swarm optimization. In *Innovative Computing Technologies (ICICT), 2010 International Conference on*, pp. 1 –6.

Re, M. & Valentini, G. (2010). Noise tolerance of multiple classifier systems in data integration-based gene function prediction. *J. Integrative Bioinformatics*, 7(3).

Rocchio, J. (1971). Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*, pp. 313–323.

Ruta, D. & Gabrys, B. (2005). Classifier selection for majority voting. *Information Fusion*, 6(1):63–81. Diversity in Multiple Classifier Systems.

Salles, T.; Rocha, L.; Pappa, G. L.; Mourão, F.; Gonçalves, M. A. & Jr., W. M. (2010). Temporally-aware algorithms for document classification. In *Proceedings of the International ACM SIGIR Conference on Research & Development of Information Retrieval*, pp. 307–314, Genebra, Switzerland.

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton university press.

Sun, S. (2010). Local within-class accuracies for weighting individual outputs in multiple classifier systems. *Pattern Recognition Letters*, 31(2):119–124.

Tan, P.-N.; Steinbach, M. & Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Tang, X.; Zhuang, L.; Cai, J. & Li, C. (2010). Multi-fault classification based on support vector machine trained by chaos particle swarm optimization. *Knowledge-Based Systems*, 23(5):486 – 490.

Tong, H.; He, J.; Li, M.; Zhang, C. & Ma, W.-Y. (2005). Graph based multi-modality learning. In *MULTIMEDIA '05: Proc. of the 13th annual ACM international conference on Multimedia*, pp. 862–871, New York, NY, USA. ACM.

Tsai, C.-F.; Lin, Y.-C.; Yen, D. C. & Chen, Y.-M. (2011). Predicting stock returns by classifier ensembles. *Appl. Soft Comput.*, 11:2452–2459.

Uğ Andur, S.; Savunma, A. & Arikan, O. (2010). A particle swarm optimization based SAR motion compensation algorithm for target image reconstruction. In *IEEE: Radar Conference, 2010*, pp. 129 –133.

Vaisakh, K.; Sridhar, M. & Linga Murthy, K. (2009). Differential evolution particle swarm optimization algorithm for reduction of network loss and voltage instability. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 391 –396.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag.

Wang, D. & Cheng, B. (2010). An unsupervised classification method of remote sensing images based on ant colony optimization algorithm. In *ADMA (1)*, volume 6440, pp. 294–301. Springer.

Wang, W. & Zhou, Z.-H. (2008). On multi-view active learning and the combination with semi-supervised learning. In *ICML '08: Proc. of the 25th international conference on Machine learning*, pp. 1152–1159, New York, NY, USA. ACM.

Wang, Z.; Chen, S.; Xue, H. & Pan, Z. (2010). A novel regularization learning for single-view patterns: Multi-view discriminative regularization. *Neural Processing Letters*, 31:159–175. 10.1007/s11063-010-9132-2.

Wen, J.-H.; Zhong, K.-J.; Tang, L.-J.; Jiang, J.-H.; Wu, H.-L.; Shen, G.-L. & Yu, R.-Q. (2011). Adaptive variable-weighted support vector machine as optimized by particle swarm optimization algorithm with application of qsar studies. *Talanta*, 84(1):13–18.

Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Yu, S.; Krishnapuram, B.; Rosales, R.; Steck, H. & Rao, B. R. (2008). Bayesian co-training. In Platt, J. C.; Koller, D.; Singer, Y. & Roweis, S., editores, *Advances in Neural Information Processing Systems 20*, pp. 1665–1672. MIT Press, Cambridge, MA.

Zhang, M. L. & Zhou, Z. H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Zhang, T.; Li, S. Z.; Xiang, S.; Zhang, L. & Liu, S. (2008). Co-Training Based Segmentation of Merged Moving Objects. In *The Eighth International Workshop on Visual Surveillance - VS 2008*.

Zhou, X.; Yuyu, J.; Qi, R.; Qian, F. & Wang, Z. (2010). IPSO: An immune based PSO supervised learning system for incremental learning. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pp. 2707 –2711.

Zou, H.-Y.; Wu, H.-L.; Fu, H.-Y.; Tang, L.-J.; Xu, L.; Nie, J.-F. & Yu, R.-Q. (2010). Variable-weighted least-squares support vector machine for multivariate spectral analysis. *Talanta*, 80(5):1698–1701.