

DISSERTAÇÃO DE MESTRADO Nº 1031

**MÉTODOS DE OTIMIZAÇÃO HIPERPARAMÉTRICA: UM ESTUDO
COMPARATIVO UTILIZANDO ÁRVORES DE DECISÃO E FLORESTAS
ALEATÓRIAS NA CLASSIFICAÇÃO BINÁRIA**

Wagner José de Alvarenga Júnior

DATA DA DEFESA: 06/02/2018

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

MÉTODOS DE OTIMIZAÇÃO HIPERPARAMÉTRICA: UM ESTUDO COMPARATIVO UTILIZANDO ÁRVORES DE DECISÃO E FLORESTAS ALEATÓRIAS NA CLASSIFICAÇÃO BINÁRIA

Wagner José de Alvarenga Júnior

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. André Paim Lemos

Belo Horizonte - MG

Fevereiro de 2018

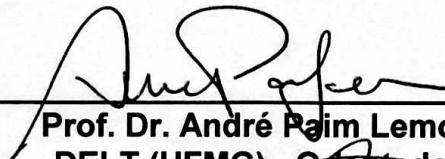
**"Métodos de Otimização Hiperparamétrica:
Um Estudo Comparativo Utilizando Árvores de Decisão e
Florestas Aleatórias na Classificação Binária"**

Wagner José de Alvarenga Júnior

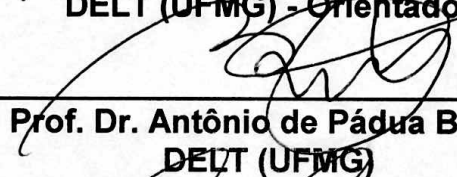
Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 06 de fevereiro de 2018.

Por:



**Prof. Dr. André Paim Lemos
DELT (UFMG) - Orientador**



**Prof. Dr. Antônio de Pádua Braga
DELT (UFMG)**



**Prof. Dr. Cristiano Leite de Castro
DEE (UFMG)**

Este trabalho é dedicado à minha mãe.

"O Dedo de Deus a tocou
e ela adormeceu".

Agradecimentos

Agradeço a Deus pela minha existência, por tudo que existe ao meu redor e por minhas metas estarem em seus Planos.

Agradeço ao professor Paim pela sua orientação e por ter, indiretamente, motivado-me no aprendizado de uma nova linguagem de programação.

Tomarei o que disse Guimarães Rosa ("é junto dos bão que a gente fica mió") para expressar minha gratidão ao laboratório D!FCOM e ao seus integrantes. Meus sinceros agradecimentos os colegas: Antoniel, Fúlvia, Guilherme, Heitor, Luciana, Luiz, Matheus, Pedro C., Pedro Q., Ramon, Rodrigo, Rosileide, Sajad e Tiago, que vivenciaram comigo o cotidiano, compartilhando suas experiências e conhecimentos. Ao colega Klenilmar, agradeço por tudo isto e também pelo auxílio com as configurações de máquinas virtuais. I would like to thank Thomas for his friendship and for the great time spent together. E aos professores: Reinaldo, Leo Torres e Fernando, por suas companhias e cuidados.

Agradeço aos professores Braga, Campelo, Cristiano, Paim, Renato (PPGCC-UFMG) e Rodney pelos fundamentos ensinados durante as disciplinas que cursei.

Agradeço os cuidados de minha mãe, que foram dados em um momento em que era ela quem mais precisava de atenção. Ao meu pai agradeço por sua dedicação, não deixando faltar nenhum zelo. Às minhas irmãs, agradeço pelo amor e carinho. Agradeço à minha avó, pelo incentivo à firmeza. Agradeço também a minha segunda família: Antônio, Teresa, Rafael, Luciano, Bruna e Leonardo.

Sem a perseverança e a lealdade da Livia, este trabalho não teria sido possível. Agradeço por todo o seu amor e sua paciência. Sua existência é motivo para meu desenvolvimento intelectual.

Agradeço à Ione por seu conhecimento e consideração, que agem como um catalisador para a minha evolução pessoal.

Ao Constantino, agradeço por incentivar o meu desejo de retornar à universidade.

Agradeço a dona Vera, a Jaqueline e a Maria Antônia, pelos seus cuidados concedidos sempre com alegria.

Ao PPGEE e seu funcionários agradeço pelo suporte acadêmico. Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), agradeço pelo suporte financeiro.

E por fim, agradeço aos membros da banca examinadora deste trabalho pela avaliação do presente documento.

Resumo

O Aprendizado de Máquina se tornou uma ferramenta fundamental para algumas áreas do conhecimento. Muito de sua robustez reside na existência de diferentes níveis de complexidade para um modelo, que podem vir a serem escolhidos, assim como as possíveis configurações do algoritmo de treinamento. Estes ajustes impactam diretamente no erro de generalização do modelo. Hiperparâmetros são as variáveis que controlam estas funções, e que precisam de uma definição de valor antes mesmo que o treinamento do modelo seja realizado. Por isto, a estimação do valor ótimo para o(s) hiperparâmetro(s) é fundamental para a obtenção de um modelo com melhor desempenho de predição.

O presente trabalho possui o objetivo de comparar o desempenho dos métodos de otimização hiperparamétrica: Busca em Grade, Busca Aleatória e otimização Bayesiana (com o uso do Processo Gaussiano), na sintonia de hiperparâmetros dos modelos de aprendizado: Árvore de Decisão e Floresta Aleatória. Nestes modelos, são testados hiperparâmetros que controlam o crescimento de uma árvore, e que define o grau de descorrelação entre as árvores de uma floresta aleatória. Estes dois algoritmos de aprendizado são empregados em problemas de Classificação binária, utilizando diferentes conjuntos de dados.

Os resultados obtidos mostram que para um mesmo número de treinamentos, a técnica de otimização Bayesiana proporciona melhores resultados que os outros dois métodos testados.

Palavras-chave: Otimização Hiperparamétrica, Busca em Grade, Busca Aleatória, otimização Bayesiana, Árvore de Decisão, Floresta Aleatória.

Abstract

Machine Learning has become a fundamental tool for some areas of knowledge. Much of its strength lies in the existing of different levels of complexity for a model and existence of adjustments for its training algorithm. These settings allow the achievement of models with lower generalization error. Hyperparameter is a type of variable that controls these functions and needs to be set even before the training procedure is carried out. Therefore the estimation of their optimum values is crucial to obtain a fine model.

This work aims to compare the performance of the following hyperparametric optimization methods: Grid Search, Random Search and Bayesian optimization (using Gaussian Process). These three techniques are applied on tuning of hyperparameters from two types of learning models: Decision Tree and Random Forest. For such comparisons, hyperparameters related to tree depth control and decorrelation level between predictors of a random forest were chosen. These two learning algorithms are applied on binary classification problems, using different datasets.

The results obtained show that for a same number of model trainings, the Bayesian optimization technique provides better results comparing to the other methods.

Keywords: Hyperparameter optimization, Grid Search, Random Search, Bayesian optimization, Decision Tree, Random Forest.

Sumário

Lista de Tabelas	iv
Lista de Figuras	vi
Símbolos	vii
Abreviaturas	x
1 Introdução	1
1.1 Motivação	1
1.2 Definição do Problema	2
1.3 Objetivos	2
1.4 Estrutura do Texto	3
2 Modelos de Aprendizado Supervisionado	5
2.1 Introdução	5
2.2 Risco Esperado	6
2.3 Risco Empírico	6
2.4 Dilema entre o Viés e a Variância	7
2.5 Avaliação e Seleção de Modelos	8
2.5.1 Métricas de Avaliação	9
2.5.2 Validação Cruzada	11
2.6 Árvores de Decisão	13
2.6.1 Introdução	13
2.6.2 Definição de uma Árvore de Decisão para Classificação	14
2.6.3 Estrutura de uma Árvore de Decisão	15
2.6.4 Métricas de Impureza	16

2.6.5	Indução de um Árvore de Decisão	17
2.6.6	Poda e Hiperparâmetros	18
2.7	Floresta Aleatória	22
2.7.1	Introdução	22
2.7.2	Bootstrap	22
2.7.3	Bagging	23
2.7.4	Definição de uma Floresta Aleatória	23
2.7.5	Características e Hiperparâmetros de uma Floresta Aleatória	24
2.7.6	Construção de uma Floresta Aleatória	25
3	Otimização Hiperparamétrica	26
3.1	Introdução	26
3.2	Revisão Bibliográfica	27
3.3	Desafios da Busca Hiperparamétrica	29
3.4	Métodos de Busca Hiperparamétrica	31
3.4.1	Otimização via Busca em Grade	31
3.4.2	Otimização via Busca Aleatória	32
3.4.3	Otimização Bayesiana	34
4	Experimentos e Resultados	45
4.1	Introdução	45
4.2	Conjunto de Dados	45
4.3	Recursos Computacionais e Ambiente de Programação	46
4.4	Metodologia	46
4.5	Resultados	50
4.5.1	Comparações Utilizando Árvores de Decisão	50
4.5.2	Comparações Utilizando Florestas Aleatórias	53
4.6	Discussão	56
5	Conclusões	58
5.1	Conclusões	58
5.2	Trabalhos Futuros	59
	Referências Bibliográficas	61

Lista de Tabelas

3.1	Hiperparâmetros e parâmetros ordinários para alguns modelos de aprendizado.	27
4.1	Conjunto de dados utilizados nos experimentos. A tabela mostra nome, número de instâncias e atributos, e a razão de balanceamento entre as duas classes, para cada conjunto.	46
4.2	Intervalo de busca para os hiperparâmetros do modelo de árvore de decisão, e respectivos valores intermediários (utilizados apenas na busca em grade).	47
4.3	Hiperparâmetros sintonizados com os métodos de busca em grade, aleatória e otimização Bayesiana, nos experimentos com o modelo de Floresta Aleatória. A tabela mostra os limites inferior e superior, e valores intermediários. Estes últimos são utilizados apenas na busca em grade.	50
4.4	Resultados para os métodos de busca em grade, busca aleatória e otimização Bayesiana, na sintonia de hiperparâmetros de árvores de decisão. Valores apresentados são média±desvio padrão da métrica AUC, para as 30 repetições dos experimentos, para cada conjunto de dados.	52
4.5	Resultados para os métodos de busca em grade, busca aleatória e otimização Bayesiana, na sintonia de hiperparâmetros de árvores de decisão. Valores apresentados são média±desvio padrão da métrica AUC, para as 30 repetições dos experimentos, para cada conjunto de dados.	55

Lista de Figuras

2.1	Esquema com cenário do aprendizado supervisionado e seus três componentes: gerador de dados, supervisor e máquina de aprendizado	6
2.2	Esquema para o dilema viés-variância.	7
2.3	Esquema com os tipos de validação cruzada: k-fold e LOOCV.	12
2.4	Gráficos com conjunto de dados sintéticos representando um problema de classificação binária com 2 atributos (<i>à esquerda</i>), e particionamento feito por uma árvore de decisão, com o respectivo valor de AUC atingindo ao se usar 60% dos dados para treinamento e 40% para teste (<i>à direita</i>). A classificação é dada pela cor que pinta cada partição, sendo a tonalidade uma escala degradê que reflete a estratificação das classes.	15
2.5	Árvore de Decisão CART construída com o conjunto de dados sintéticos da Figura 2.4. Na imagem, cada nó possui: o atributo considerado no <i>split</i> e seu valor de corte, o valor de pureza (gini), o número de amostras, a estratificação para as classes e a classe dominante. O caminho à esquerda de um nó representa valores verdadeiros (em relação ao critério de <i>split</i>), e à direita os falsos.	16
2.6	Árvores de decisão (CART) construídas com o conjunto de dados da Figura 2.4. Partindo do quadrante superior direito (no sentido horário) cada modelo fez uso do seguinte hiperparâmetro: N_{min} , N_{folha} , β e d_{max} , respectivamente.	20
2.7	Os gráficos mostram a classificação do conjunto de dados da Figura 2.4, feito por cada árvore de decisão que está ilustrada na Figura 2.6.	21
2.8	Ilustração da técnica bootstrap, na criação de B conjuntos reamostrados, com reposição, a partir do conjunto original contendo 3 instâncias.	22
2.9	Relação entre número de árvores e erro de predição (utilizando a métrica AUC), de uma floresta aleatória empregada na classificação dos dados da Figura 2.4, cujo hiperparâmetro $d_{max} = 5$ foi utilizado. Os dados apresentados no gráfico são média e desvio padrão de 10 execuções.	24
3.1	Esquema com o espaço de configuração hiperparamétrico de uma rede MLP. . .	31
3.2	Exemplos de grades hiperparamétricas para (a) uma, (b) duas e (c) três dimensões. . .	32
3.3	Otimização hiperparamétrica com 9 pontos de configuração, utilizando Busca em Grade e Busca Aleatória. Imagem inspirada em (Bergstra e Bengio, 2012) . .	34
3.4	Exemplos de <i>kernels</i> de família matérn, para $\nu = 3/2$ e $\nu = 5/2$, e o caso particular que resulta na Exponencial Quadrático (SE) quando o $\nu \rightarrow \infty$	37
3.5	Gráficos mostram comportamentos variados para o modelo <i>a posteriori</i> , em função do tipo de <i>kernel</i> (matérn $\nu = 3/2$ e $\nu = 5/2$ e exponencial quadrático) e diferentes valores para seus hiperparâmetros.	38

3.6	Exemplo de distribuição <i>a posteriori</i> de um processo Gaussiano de dimensão 1, com 3 pares (\mathbf{x}, \mathbf{f}) . A linha azul é a função desconhecida, a linha laranja mostra a média <i>a posteriori</i> (aproximação), e a área sombreada a variância.	40
3.7	Influência das funções de aquisição PI, UCB e EI, para diferentes valores de seus hiperparâmetros. Em azul, a função verdadeira $f(x) = (e^{-(x-2)^2} + e^{-(x-6)^{2/10}} + 1/(x^2 + 0.5))/2$. Este exemplo utiliza o algoritmo L-BFGS-B, como método de otimização (com 13 avaliações), e o <i>kernel</i> matérn ($\nu = 5/2$).	42
4.1	Resultados dos conjuntos Breast Cancer, Cardiotocography, Ionosphere, Spambase e Vertebral Column, para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de árvore de decisão.	51
4.2	Resultados dos conjuntos Musk-2, Parkinsons, Pima Indian Diabetes, Qsar e Sonar, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de árvore de decisão.	51
4.3	Resultados dos conjuntos Diabetic, German Credit, Liver e Musk-1, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de árvore de decisão.	52
4.4	Resultados dos conjuntos Breast Cancer, Cardiotocography, Ionosphere, Musk-2 e Spambase, com a métrica AUC para as 10 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de floresta aleatória.	53
4.5	Resultados dos conjuntos Musk-1, Parkinsons, Qsar, Sonar e Vertebral Column, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de floresta aleatória.	54
4.6	Resultados dos conjuntos Diabetic, German Credit, Liver e Pima Indian Diabetes, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de floresta aleatória.	54
4.7	Resultado dos testes com o método de otimização Bayesiana na estimação dos hiperparâmetros N_{folha} , β e m de um modelo de floresta aleatória, para os conjuntos: Ionosphere, Liver e Musk-1. Em preto o maior valor da métrica AUC, até o ponto em questão. Desvio padrão em vermelho para 5 repetições.	56

Símbolos

α	Nível de significância estatística;
α_{cut}	Hiperparâmetro da poda critério de custo-complexidade;
$\alpha_{l-bfgs-b}$	Hiperparâmetro do método L-BFGS-B;
β	Hiperparâmetro relacionado à profundidade de uma T ;
$\Gamma(v)$	Função gama igual a $(v - 1)!$;
γ_j	Valor de saída atribuído a uma R_j ;
θ	Vetor contendo os parâmetros a serem estimados pela MLE;
θ_0	Vetor com valores iniciais dos parâmetros estimados pela MLE;
$\hat{\theta}$	Vetor contendo a estimação que maximiza a verossimilhança;
$\Theta = \{R_j, \gamma_j\}_1^J$	Parâmetros de uma T ;
Θ^*	Vetor que minimiza $R(\Theta)$;
κ	Hiperparâmetro da função de aquisição UCB;
λ_i	Representação de um hiperparâmetro;
λ_H	Vetor contendo os H hiperparâmetros de um algoritmo;
λ^*	Vetor contendo os hiperparâmetros que maximizam o desempenho de predição de um modelo;
λ_{t+1}	Próximo vetor candidato no processo de otimização hiperparamétrica;
Λ	Espaço de busca de um hiperparâmetro;
Λ	Espaço de busca para um vetor de hiperparâmetros;
μ_t	Média a posteriori;
ξ	Hiperparâmetro das funções de aquisição PI e EI;
ζ	Função genérica de Densidade (Probabilidade);
σ_0^2	Hiperparâmetro da função de covariância que controla a quantidade de variância global da função;
σ_t^2	Incerteza a posteriori;
ν	Hiperparâmetro da função de covariância matérn;
Υ	Subespaço de busca para o método MLE;
Φ	Função de distribuição acumulada normal;
ϕ	Função de distribuição de probabilidade normal;
Ψ	Matriz diagonal contendo hiperparâmetros l_i do <i>kernel</i> ;
Ω	Espaço de busca genérico;
A	Modelo de aprendizado;
A_λ	Modelo de aprendizado configurado com o vetor de hiperparâmetros λ ;
\mathcal{A}	Conjunto de algoritmos de aprendizado;
B	Hiperparâmetro de uma floresta aleatória que define o número de T ;
$CV_{(n)}$	Validação cruzada do tipo LOOCV;
$C_{\alpha_{cut}}(T)$	Critério de custo-complexidade da poda de uma T ;

d_{max}	Hiperparâmetro relacionado à profundidade de uma T ;
\mathcal{D}	Conjunto de dados $(\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$;
\mathcal{D}_{train}	Conjunto de dados de treinamento;
\mathcal{D}_{valid}	Conjunto de dados de validação;
\mathbf{D}	Conjunto de dados contendo os pares (λ, \mathbf{f}) ;
$E[\cdot]$	Operador de Esperança;
f	Função genérica;
$\hat{f}_{bag}(\mathbf{x}_i)$	Função de Predição para um modelo de Bagging;
$\hat{f}_{fa}^B(\mathbf{x}_i)$	Função de Predição para um modelo de floresta aleatória;
\mathbf{f}	Resultado de uma avaliação de desempenho de predição de um modelo de aprendizado;
F_q	Estatística do teste de Quade;
FP_{taxa}	Taxa de Falsos Positivos;
g	Número de Equações de Verossimilhança em uma MLE;
$g_{[\cdot]}^l$	Graus de liberdade;
$\mathcal{GP}(m(x), k_{[\cdot]}(x, x'))$	Função que define o Processo Gaussiano;
H	Número total de hiperparâmetros de um A ;
\mathbf{H}	Matriz Hessiana;
$I(\cdot)$	Função indicadora que retorna 0 ou 1;
$i(j)$	Grau de impureza (pureza) de uma R_j ;
J	Número total de regiões disjuntas determinadas por uma T ;
j_F	Saída Falsa de um nó em uma T ;
j_T	Saída Verdadeira de um nó em uma T ;
K	Número total de classes em um problema de reconhecimento de padrões;
k_{cv}	Número de partições da CV;
$k_{[\cdot]}(x, x')$	Função genérica de covariância;
$k_{SE}(x, x')$	Função de covariância Exponencial Quadrática;
$k_{Matern}(x, x')$	Função de covariância Matern;
K_ν	Função de Bessel modificada;
\mathbf{k}	Vetor de covariância entre x_{t+1} e elementos do vetor \mathbf{x} ;
\mathbf{K}	Matriz de covariância dos elementos de um vetor \mathbf{x} (termo a termo);
$KL(Q P)$	Divergência de Kullback-Leibler;
l	Hiperparâmetro comprimento de escala de um <i>kernel</i> ;
$l(\boldsymbol{\theta}, \mathbf{x})$	Função log-Verossimilhança;
$L(\boldsymbol{\theta}, \mathbf{x})$	Função de Verossimilhança;
\mathcal{L}	Função de Perda (<i>loss function</i>);
M_p	Modelo de Superfície de Resposta;
m	Hiperparâmetro de uma floresta aleatória que controla a decorrelação de suas $T(s)$;
$m(x)$	Função Média do processo gaussiano;
$m_{l-bfgs-b}$	Hiperparâmetro do método L-BFGS-B;
\mathbf{m}	Hiperparâmetro da função média $m(x)$;
n	Número total de amostras de um conjunto;
n_{train}	Número total de amostras de treinamento;
N	Número total de amostras avaliadas pelo modelo;
N_{folha}	Hiperparâmetro relacionado à profundidade de uma T ;
N_m	Total de pontos de treinamento contidos em uma folha de uma T ;
N_{min}	Hiperparâmetro relacionado à profundidade de uma T ;

Opt	Método genérico de otimização;
p_F	Proporção entre pontos que tomam a direção j_F e total que chega ao nó;
p_T	Proporção entre pontos que tomam a direção j_T e total que chega ao nó;
\hat{p}_{mk}	Proporção de pontos de uma determinada classe que habita a região R_j ;
$p(f)$	Probabilidade <i>a priori</i> do GP;
$p(\mathbf{D} f)$	Função de verossimilhança do GP;
$p(f \mathbf{D})$	Probabilidade <i>a posteriori</i> do GP;
$Q_m(T)$	Métrica de impureza utilizada em T ;
r^2	Operador da função de covariância igual à $(x - x')^T \Psi (x - x')$;
$R[\cdot]$	Risco Esperado;
R_j	Região do espaço de atributos a ser dividida por uma T ;
R_{emp}	Risco Empírico;
s_j	Valor de corte de uma região dividida por uma T ;
S_j	Média j da matriz ponderada do teste de Quade;
t	Atual iteração do processo de otimização hiperparamétrica;
t'	Atual iteração das rodadas iniciais aleatórias do processo de otimização hiperparamétrica;
t_d	Limiar de decisão;
$t_{1-\alpha/2^*,(gl1)(gl2)}$	Valor para a distribuição t de Student para α , gl_1 e gl_2 ;
T	Sub-árvore contida em T_0 ;
T_0	Árvore de decisão cuja profundidade não foi limitada;
$ T $	Total de folhas de uma T ;
\mathbb{T}	Número total de treinamentos realizados por um método de otimização hiperparamétrica;
\mathbb{T}'	Número de treinamentos iniciais realizados com valores de hiperparâmetros aleatórios;
$T(\mathbf{x}; \Theta)$	Função de Predição para uma T ;
u	Número de reinicializações do método MLE;
VP_{taxa}	Taxa de Verdadeiros Positivos;
x	Atributo de um modelo genérico;
x^*	Atributo com o valor que maximiza uma certa função;
x^+	Melhor ponto calculado até a presente iteração pelo GP;
x_{t+1}	Próximo valor de x calculado pelo GP;
\mathbf{x}	Vetor contendo os atributos de um modelo genérico;
\mathcal{X}^d	Espaço genérico de busca para um vetor \mathbf{x} contendo d atributos;
y	Rótulo para um determinado padrão \mathbf{x} ;
\hat{y}	Variável contendo a predição de um modelo de aprendizado;
\mathbf{y}	Vetor contendo as variáveis de saída de um modelo genérico;
Y	Espaço contendo $\{0,1\}$;
\mathbf{Z}	Conjunto de dados criado através do método Bootstrap;

Abreviaturas

Acc	Acurácia;
ANOVA	Análise de Variância (<i>Analysis of variance</i>);
AUC	Área Abaixo da Curva ROC (<i>Area Under the ROC Curve</i>);
C4.5	Algoritmo de árvore de decisão desenvolvido por Quinlan (1993) ;
CART	Algoritmo Árvores de Classificação e Regressão (<i>Classification And Regression Trees</i>);
CASH	Problema Combinado de Seleção de algoritmo e Otimização Hiperparamétrica (<i>Combined Algorithm Selection and Hyperparameter Optimization Problem</i>);
CMA-ES	Estratégia Evolucionária de Adaptação da Matriz de Covariância (<i>Covariance Matrix Adaptation Evolution Strategy</i>);
CNN	Rede Neural Convolutacional (<i>Convolutional Neural Network</i>);
CV	Validação Cruzada (<i>Cross Validation</i>);
EI	Função de aquisição Esperança de Melhoria (<i>Expected Improvement</i>);
Espec	Especificidade;
FN	Falso(s) Negativo(s);
FP	Falso(s) Positivo(s);
GP	Processo Gaussiano (<i>Gaussian Process</i>);
ID3	<i>Iterative Dichotomiser 3</i> ;
k-fold	Validação Cruzada por k-partições
L-BFGS-B	<i>Limited-memory Broyden-Fletcher-Goldfarb-Shanno Bound-constrained</i> ;
LOOCV	Validação Cruzada por Unidade;
MAP	Estimação de Máxima a Posteriori (<i>Maximum a Posteriori Estimation</i>);
MLE	Estimação de Máxima Verossimilhança (<i>Maximum Likelihood Estimation</i>);
MLP	Rede Neural de Múltiplas Camadas (<i>Multilayer Perceptron</i>);
OOB	Estimativa Fora-do-saco (<i>Out-of-bag estimation</i>);
PI	Função de aquisição Probabilidade de Melhoria (<i>Probability of Improvement</i>);
ROC	Característica de Operação do Receptor (<i>Receiver Operating Characteristic</i>);

SBMO	Otimização Baseada em Modelo Sequencial (<i>Sequential Model-Based Optimization</i>);
SE	Função de covariância Exponencial Quadrático (<i>Squared Exponential</i>);
Sens	Sensibilidade;
SGD	Gradiente Descendente Estocástico (<i>Stochastic Gradient Descent</i>);
SMAC	Configuração de Algoritmo Baseada em Modelo Sequencial (<i>Sequential Model-based Algorithm Configuration</i>);
SVM	Máquina de Vetores de Suporte (<i>Support Vector Machine</i>);
TPE	Estimação de Parzen com Estrutura em Árvore (<i>Tree Parzen Estimation</i>);
UCB	Função de aquisição Limite de Confiança Superior (<i>Upper Confidence Bound</i>);
VN	Verdadeiro(s) Negativo(s);
VP	Verdadeiro(s) Positivo(s);

Introdução

1.1 Motivação

Nos últimos anos, a utilização de Modelos de Aprendizado de Máquina vem transformando radicalmente áreas como: automotiva (veículos autônomos), medicina (diagnóstico de exames de imagens), dispositivos móveis (reconhecimento de fala e facial), militar (visão computacional), mídias sociais (análise de cenários e opiniões), financeira (detecção de padrões de compra), agricultura (detecção de pragas e agricultura de precisão), dentre outras (Pyle e Jose, 2015).

A teoria do aprendizado estatístico foi introduzida no final da década de 1960 (Vapnik, 1995), sedimentando as bases do aprendizado de máquina. E desde então, este último tem sido aplicado cada vez mais, apoiado no crescimento do volume de dados disponíveis (*big data*), no desenvolvimento de novos recursos com maior poder computacional, utilizados na implementação e treinamento de modelos de aprendizado, e no aperfeiçoamento destes últimos, por meio de novos algoritmos ou pela otimização dos já existentes.

Modelos de Aprendizado são compostos por parâmetros que fazem parte de sua formulação matemática, e cujo valores são estimados a partir de um conjunto de dados de treinamento (Murphy, 2012). Nestes modelos, existem ainda parâmetros denominados Hiperparâmetros, que se diferenciam dos primeiros por não serem estimados do mesmo modo e por necessitarem de uma definição de valores definitiva, antes mesmo que o treinamento se inicie. Os hiperparâmetros definem propriedades como a complexidade do modelo e o quão rápido os parâmetros serão aprendidos (Bishop, 2006). Ou seja, os hiperparâmetros estão diretamente ligados ao desempenho do modelo treinado e ao número de operações computacionais necessárias no aprendizado, assim como seu respectivo tempo de duração.

Tradicionalmente, a otimização de hiperparâmetros tem sido uma tarefa humana, pela sua grande eficiência quando apenas poucos hiperparâmetros (1 ou 2) estão presentes no problema de aprendizado (Bergstra et al., 2011). Contudo, a exploração manual do espaço hiperparamétrico, na maior parte dos casos, é uma tarefa tediosa e inclinada a resultar em um desempenho insatisfatório do modelo (Hutter et al., 2010; Swersky et al., 2014), visto o caso de modelos com grande número de hiperparâmetros, por exemplo. Outro ponto pertinente é que para modelos com arquitetura profunda (*deep learning*), como algumas redes neurais, o

1.2 Definição do Problema

treinamento a partir de um vasto conjunto de dados pode levar desde horas a dias (Krizhevsky et al., 2012), o que torna a escolha por um dos métodos de otimização hiperparamétrica, uma questão bastante relevante para minimizar o tempo total gasto na obtenção de um modelo com desempenho satisfatório.

Assim, há atualmente, um grande apelo por parte da comunidade científica, por métodos automáticos para realizar a otimização hiperparamétrica de modelos. Várias abordagens diferentes são encontradas na literatura para lidar com este problema, como: a Busca em Grade (Hsu et al., 2003), a Busca Aleatória (Bergstra e Bengio, 2012), a Otimização baseada no Gradiente (Bengio, 2000; Maclaurin et al., 2015), métodos de Computação Evolucionária (Friedrichs e Igel, 2005) e a Otimização Bayesiana, em suas 3 vertentes, sendo estas: Configuração de Algoritmo Baseada em Modelo Sequencial (SMAC) (Hutter et al., 2010), a técnica do Estimador de Parzen com Estrutura em Árvore (TPE) (Bergstra et al., 2011), e a que utiliza o Processo Gaussiano (GP) (Snoek et al., 2012).

1.2 Definição do Problema

O problema de otimização hiperparamétrica pode ser visto a partir das seguintes considerações: dado um modelo de aprendizado A , contendo os hiperparâmetros $\lambda_1, \lambda_2, \dots, \lambda_H$, pertencentes aos respectivos domínios $\Lambda_1, \Lambda_2, \dots, \Lambda_H$. Então, para cada configuração hiperparamétrica $\lambda \in \Lambda$, representa-se como A_λ o modelo configurado pelos hiperparâmetros λ . Considere ainda uma função de perda (*loss-function* - \mathcal{L}), um conjunto de dados $\mathcal{D} = (\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$ dividido em conjunto de treinamento \mathcal{D}_{train} e validação \mathcal{D}_{valid} , onde \mathbf{x}_i , \mathbf{y}_i e n são: vetor com características, vetor com variáveis de saída e número total de amostras, respectivamente. Então, o problema de otimização hiperparamétrica pode ser escrito como a busca pelo,

$$\lambda^* \in \arg \min_{\lambda \in \Lambda} \mathcal{L}(A_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}), \quad (1.1)$$

Sendo aqui assumido a utilização de um conjunto de validação. Dependendo do tipo de função de perda que se emprega, a equação 1.1 é tomada como o $\lambda^* \in \arg \max$.

Um ponto importante em relação à equação 1.1 é que a função $f : \lambda \mapsto \mathbf{f}$, que mapeia os hiperparâmetros ao valor da avaliação pela função de perda, é desconhecida a princípio.

1.3 Objetivos

A partir do contexto anteriormente apresentado, o presente trabalho possui como objetivo o estudo e a comparação dos métodos de otimização hiperparamétrica: busca em grade, busca aleatória e otimização Bayesiana utilizando o processo Gaussiano (Snoek et al., 2012). Estes 3 métodos de otimização hiperparamétrica são aplicados na sintonia de hiperparâmetros presentes nos modelos de aprendizado: Árvore de Decisão (Breiman et al., 1984) e Floresta Aleatória (Breiman, 2001). A comparação visa medir o desempenho de predição destes dois modelos,

otimizados pelos 3 métodos, quando são empregados na resolução do problema de classificação binária¹, utilizando para isto diferentes conjunto de dados. Este trabalho optou pela abordagem de Aprendizado Supervisionado, no qual o modelo procura aprender as relações presentes entre os atributos de entrada e as variáveis de saída, presentes nos dados de treinamento.

Assim, esta dissertação possui como objetivos:

- Apresentar o problema de otimização hiperparamétrica e descrever, especificamente, os métodos de busca em grade, busca aleatória, otimização Bayesiana (GP);
- Discorrer sobre os modelos de aprendizado: árvores de decisão e floresta aleatória, discutindo sobre suas estruturas e principais hiperparâmetros;
- Relatar os experimentos realizados com a comparação dos 3 métodos de otimização (busca em grade e aleatória, e otimização Bayesiana utilizando o processo Gaussiano) na sintonia dos hiperparâmetros dos modelos de árvore de decisão e floresta aleatória quando estes são utilizados na classificação de conjuntos de dados binários. Os testes comparativos visam constatar se algum dos 3 métodos investigados conduz a um desempenho de classificação, estatisticamente superior, quando um mesmo número de treinamento, de um modelo de aprendizado, é realizado na aplicação de cada uma das técnicas de otimização.

1.4 Estrutura do Texto

A presente dissertação está dividida em 5 capítulos. O Capítulo atual apresenta a motivação para o desenvolvimento do estudo, define a formulação do problema de otimização hiperparamétrica para um algoritmo de aprendizado e relata os objetivos deste trabalho.

O Capítulo 2 introduz o conceito de aprendizado supervisionado, abordando para isto: a definição de risco esperado, a minimização do risco empírico e o dilema entre o viés e a variância. Logo em seguida a avaliação e seleção de modelos é discutida, apresentando métricas de avaliação e a técnica de validação cruzada. Então, dois modelos de aprendizado, árvore de decisão e floresta aleatória, são estudados. Para isto, a definição estrutural destes modelos são abordadas, assim como seus hiperparâmetros e respectivas possibilidades de incremento no desempenho de predição.

No Capítulo 3 é caracterizada uma variável hiperparamétrica e abordada a sua importância quanto ao desempenho de um modelo de aprendizado. Uma revisão bibliográfica sobre métodos de otimização hiperparamétrica é exposta. São descritos alguns aspectos que estão por trás do desafio da busca ótima dos hiperparâmetros, no que se refere as causas do problema de otimização hiperparamétrica. Em seguida, são apresentados os métodos de busca em grade, busca aleatória, o conceito da Otimização Baseada em Modelo Sequencial (SBMO), e

¹Um problema de Classificação Binária é composto por d atributos representados por um vetor \mathbf{x} , pertencentes a algum espaço \mathcal{X}^d , e variáveis de saída que assumem somente dois valores simbólicos, $y \in Y = \{0,1\}$, referentes às duas classes existentes (positiva, $y = 1$, e negativa, $y = 0$). Então, sua resolução pode ser dada por um modelo de aprendizado que utiliza um conjunto finito de amostras para aprender uma função $f : \mathcal{X}^d \mapsto \{0,1\}$.

abordada a técnica de otimização Bayesiana. São brevemente descritos a Estimação de Máxima Verossimilhança e o algoritmo L-BFGS-B como ferramentas da otimização Bayesiana.

No Capítulo 4 são relatados os experimentos comparativos entre os três métodos de otimização hiperparamétrica. São expostos os conjuntos de dados utilizados nos testes, a metodologia adotada quanto a configuração dos algoritmos de aprendizado e métodos de otimização hiperparamétrica. Os resultados dos experimentos são apresentados, bem como a conclusão do teste estatístico e uma análise de convergência sobre a otimização Bayesiana na sintonia de hiperparâmetros de floresta aleatória. E por fim, estes resultados são discutidos, e as configurações dos recursos computacionais e ambiente de programação são exibidos.

E finalmente no Capítulo 5, é traçada uma conclusão sobre o estudo e feita a indicação de sugestões para trabalhos futuros.

Modelos de Aprendizado Supervisionado

2.1 Introdução

O Aprendizado de Máquina é usualmente dividido em duas vertentes principais, identificadas como: aprendizado com um supervisor e aprendizado sem um supervisor (Murphy, 2012). Há ainda outras formas de aprendizado que se diferenciam desta duas primeiras, sendo: aprendizado semi-supervisionado, aprendizado por reforço, e outros.

No aprendizado com um supervisor, ou Aprendizado supervisionado, o modelo tem como objetivo a estimação de uma função, a partir de um conjunto de dados finitos (Vapnik, 1995). Dentro do aprendizado supervisionado há diferentes tarefas que aprendem à partir de dados, sendo a classificação de padrões uma destas.

A formulação de um problema de aprendizado supervisionado pode ser vista como a estimação de uma dependência funcional desconhecida, com entrada e saída, em um cenário envolvendo três componentes (Cherkassky e Mulier, 2007), relacionados a seguir:

- Gerador de Dados - responsável por produzir vetores aleatórios \mathbf{x} em algum espaço \mathcal{X}^d , que são amostrados independentemente por meio uma densidade de probabilidade $p(\mathbf{x})$ que é desconhecida. Ou seja, o gerador de dados é responsável por obter amostras \mathbf{x} contendo os atributos de um determinado problema.
- Supervisor - encarregado de produzir um valor de saída $y \in Y$ para cada vetor \mathbf{x} , de acordo com a densidade condicional $p(y|\mathbf{x})$ que também é desconhecida. Em outras palavras, o supervisor possui a tarefa de determinar o rótulo de y cada vetor de atributos \mathbf{x} .
- Máquina de Aprendizado - responsável pela implementação de um conjunto de funções, em que cada uma pode ser representada por $f(\mathbf{x}, \Theta)$, $\Theta \in \Omega$, em que Θ são parâmetros da máquina de aprendizado e Ω algum espaço de busca. A máquina de aprendizado retorna um valor aproximado \hat{y} para uma determinada entrada \mathbf{x} .

A Figura 2.1 mostra um esquema com o cenário para o aprendizado supervisionado e seus três componentes.

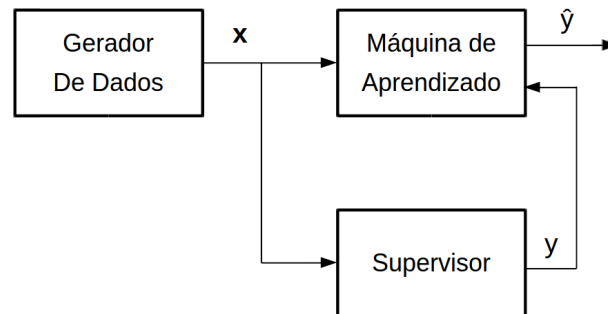


Figura 2.1: Esquema com cenário do aprendizado supervisionado e seus três componentes: gerador de dados, supervisor e máquina de aprendizado

No problema de aprendizado supervisionado, busca-se selecionar a melhor função, dentre as fornecidas pela máquina de aprendizado, que possui a resposta mais próxima à fornecida pelo supervisor. Para isto, utiliza-se de uma função de perda (\mathcal{L}), que quantifica, de alguma forma, a diferença entre a resposta da máquina de aprendizado (\hat{y}) e a resposta do supervisor (y).

2.2 Risco Esperado

Dado uma função de aproximação $f(\mathbf{x}, \Theta)$, em que Θ foi de alguma forma selecionado, e considerando as condições ideais em que a função de densidade de probabilidade conjunta $p(\mathbf{x}, y)$ é conhecida, como também todas as relações de entrada e saída (\mathbf{x}, y) , então, o valor esperado da perda é dado pelo risco esperado,

$$R(\Theta) = \int \mathcal{L}(y, f(\mathbf{x}, \Theta)) dp(\mathbf{x}, y) = E[\mathcal{L}(y, f(\mathbf{x}, \Theta))], \quad (2.1)$$

onde, $\mathcal{L}(y, f(\mathbf{x}, \Theta))$ é a função de perda, e $E[\cdot]$ o operador de Esperança.

Então, o aprendizado pode ser definido como o processo de busca pela função de aproximação ótima $f(\mathbf{x}, \Theta^*)$, em que Θ^* é o vetor que minimiza $R(\Theta)$ (Vapnik, 1995).

2.3 Risco Empírico

A incerteza da distribuição $p(\mathbf{x}, y)$ e a pequena amostragem das relações entrada-saída (\mathbf{x}, y) impedem o cálculo da integral da equação 2.1. Contudo, o aprendizado supervisionado pode aproximar $R(\Theta)$ por meio do Risco Empírico (R_{emp}), utilizando para isto um conjunto de dados

2.4 Dilema entre o Viés e a Variância

$\mathcal{D} = (\mathbf{x}_{1:n}, y_{1:n})$, em que n é o número total de amostras. O risco empírico converge para o risco esperada a medida de n tende ao infinito.

2.4 Dilema entre o Viés e a Variância

O dilema entre o viés e a variância (Geman et al., 1992) trata da decomposição da esperança do erro de generalização $E[(f(\mathbf{x}, \Theta) - E[y|\mathbf{x}, \Theta])^2]$ (em que Θ são os parâmetros do modelo de aprendizado, originalmente para um problema de regressão) em três parcelas: o viés, a variância e o erro irreduzível. Para compreender o impacto de cada parcela é necessário definir a Capacidade de um modelo. Esta última está relacionado ao grau de flexibilidade do modelo, ditando o nível de complexidade da função de aproximação que foi aprendida. A figura 2.2 ilustra o dilema entre o viés e a variância, no que tange o erro de generalização (mensurado através do conjunto de validação) quando se altera a capacidade do modelo.

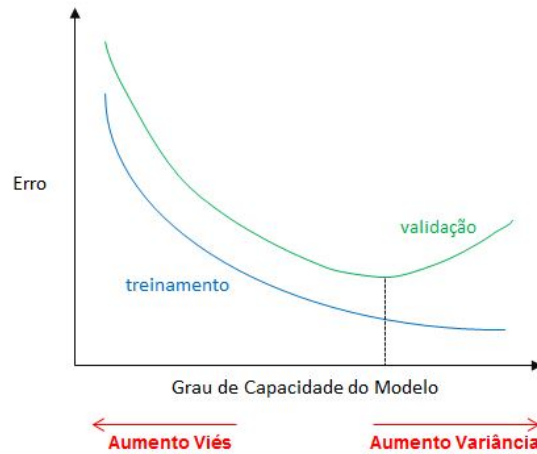


Figura 2.2: Esquema para o dilema viés-variância.

Para tratar do dilema entre o viés e a variância, é considerado aqui, o erro quadrático como função de perda $((y - f(\mathbf{x}, \Theta))^2)$. Assim, o erro empírico é expresso como,

$$R_{emp}(\Theta) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y, f(\mathbf{x}, \Theta)) = E[(y - f(\mathbf{x}, \Theta))^2]. \quad (2.2)$$

A equação 2.2 pode ser descomposta em dois termos como mostrado a seguir,

$$E[(y - f(\mathbf{x}, \Theta))^2] = E[(y - E[y|\mathbf{x}, \Theta])^2] + E[(f(\mathbf{x}, \Theta) - E[y|\mathbf{x}, \Theta])^2]. \quad (2.3)$$

A equação 2.3 mostra que o erro pode ser dividido em duas porções, uma parte passível de redução (segundo termo) e outra não redutível (primeiro termo). A parte do erro sujeita à

redução pode ser decomposta em duas, como mostra a equação à seguir,

$$E[(f(\mathbf{x}, \Theta) - E[y|\mathbf{x}, \Theta])^2] = (E[(f(\mathbf{x}, \Theta))] - E[y|\mathbf{x}, \Theta])^2 + E[(f(\mathbf{x}, \Theta) - E[f(\mathbf{x}, \Theta)])^2], \quad (2.4)$$

em que o primeiro e segundo termos do lado direito representam o viés e a variância, respectivamente.

A Capacidade de um Modelo depende da quantidade de amostras contidas no conjunto de dados de treinamento e da complexidade das funções implementadas pela máquina de aprendizado (Vapnik, 1998). Quando se utiliza uma capacidade maior que a requerida para um determinado problema, pode-se obter modelos com baixo viés e grande variância, acarretando em um sobreajuste do modelo. Já o uso de uma capacidade inferior ao requisitado pelo problema, pode-se resultar em um modelo com alto viés e pequena variância, levando ao efeito de subajuste do modelo.

2.5 Avaliação e Seleção de Modelos

A estimação do erro de generalização de um algoritmo de aprendizado é conhecido como Avaliação de Modelo. Esta avaliação trata da estimação do erro de predição sobre dados não vistos pelo modelo, durante o seu treinamento. Já a tarefa de estimar o desempenho de diferentes algoritmos de aprendizado, para a escolha do melhor entre estes, é conhecida como Seleção de Modelos (Hastie et al., 2009). Para realizar estas operações, existe na literatura uma gama de métricas de avaliações, dentre as quais algumas são abordadas nesta seção.

A tarefa de avaliação de modelos está sujeita à diferentes fatores relacionados ao tipo de modelo utilizado e ao problema que se quer solucionar. Isto envolve questões como: se o modelo é de regressão ou classificação, se o mesmo é linear ou não linear, se o custo de se classificar erroneamente uma classe é o mesmo que para as demais, se os dados estão ou não balanceados, e outros. Para lidar com estes variados cenários, existem diferentes métricas que possuem, cada uma, uma forma particular de mensurar o desempenho de um modelo. A correta escolha por uma métrica é fundamental para obter um modelo com melhor resultado de predição (Japkowicz e Shah, 2011).

A Seleção de Modelos é uma técnica usada para estimar o desempenho de diversos modelos para determinar qual destes possui o melhor desempenho para um dado problema. Sua utilização é pertinente, em razão de alguns modelos de aprendizado assumirem premissas sobre a estrutura dos dados, relacionados ao problema, que podem não ser válidas, resultando em um menor desempenho (Thornton et al., 2013). Alguns trabalhos abordam a escolha do modelo de aprendizado para um dado problema específico como uma escolha hiperparamétrica (Thornton et al., 2013). Este caso, é abordado da seguinte maneira: dado um conjunto de algoritmos de aprendizado \mathcal{A} e um conjunto de dados $\mathcal{D} = (\mathbf{x}_{1:n}, y_{1:n})$ em que n é o número total de amostras, dividido em conjunto de treinamento \mathcal{D}_{train} e validação \mathcal{D}_{valid} , o objetivo deste problema é selecionar o modelo $A^* \in \mathcal{A}$ com o máximo desempenho de generalização, utilizando a função

custo \mathcal{L} , como expresso em,

$$A^* \in \arg \min_{A \in \mathcal{A}} \mathcal{L}(A, \mathcal{D}_{train}, \mathcal{D}_{valid}), \quad (2.5)$$

dependendo de \mathcal{L} , o problema é tomado pela maximização.

2.5.1 Métricas de Avaliação

2.5.1.1 Erro Médio de Classificação

Em um problema de classificação, o objetivo de uma métrica de avaliação é comparar a resposta categórica ($\hat{y} = \hat{f}(\mathbf{x})$) estimada pelo modelo, para uma dada entrada \mathbf{x} , em relação ao rótulo verdadeiro ($y = f(\mathbf{x})$), que representa o valor referente a uma das K classes existentes. Uma forma simples de se fazer isto é por meio do Erro Médio de Classificação, dado por,

$$\mathcal{L}(y, \hat{f}(\mathbf{x})) = \frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i), \quad (2.6)$$

onde, n é o número total de amostras, e $I()$ é uma função indicadora que retorna 0 (se $y_i = \hat{y}_i$) ou 1 (se $y_i \neq \hat{y}_i$).

O erro médio de classificação, também conhecido como Acurácia, fornece o acerto médio global entre todas as classes de um classificador. Para o caso de um problema de classificação binária (classes positiva e negativa), a acurácia é expressa como,

$$Acc = \frac{VP + VN}{N}, \quad (2.7)$$

onde, VP representa as amostras positivas classificadas corretamente (verdadeiros positivos), VN retrata as amostras negativas classificadas corretamente (verdadeiros negativos), e N representa o número total de amostras avaliadas pelo classificador.

2.5.1.2 Entropia Cruzada

Uma outra forma de avaliar um classificador é através do uso das probabilidades $p_k(\mathbf{x}) = P(G = k|\mathbf{x})$, resultantes da apresentação de uma amostra \mathbf{x} ao modelo. Um conceito muito importante neste sentido é a divergência de Kullback-Leibler que mede o quanto uma distribuição de probabilidade $Q(x)$ se diverge de uma segunda distribuição de probabilidade $P(x)$ (MacKay, 2003), sendo expressa como,

$$KL(Q||P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)}, \quad (2.8)$$

em que $KL(Q||P) \geq 0$, e a igualdade ocorre apenas se $P = Q$. Então, ao se admitir a resposta verdadeira como y , e a predição do modelo como $\hat{y} = \hat{f}(\mathbf{x})$, obtêm-se a partir da equação 2.8, a

função de custo Entropia Cruzada, dada por,

$$\mathcal{L}(y, \hat{f}(\mathbf{x})) = -\frac{1}{n} \sum_{i=1}^n [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \quad (\text{Entropia Cruzada, caso binário}), \quad (2.9)$$

$$\mathcal{L}(y, \hat{f}(\mathbf{x})) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^k y_{ij} \log \hat{y}_{ij} \quad (\text{Entropia Cruzada, caso multiclasse}), \quad (2.10)$$

onde, n é o número total de amostras, e K é o número total de classes.

A entropia cruzada toma valores próximos de 0, à medida em que \hat{y}_i e y_i convergem.

2.5.1.3 Discriminando Erros e Acertos: Sensibilidade, Especificidade e AUC

Em se tratando de avaliação de desempenho para um modelo de aprendizado aplicado a um problema de classificação, o uso da acurácia (equação 2.6) negligencia aspectos como o diferente custo em se classificar erroneamente cada classe existente ou o fato de haver um desbalanceamento no conjunto de dados que representa o problema (de Castro, 2011). Um exemplo deste último caso é a situação de um modelo de classificação, cujo o problema à que se aplica possui duas classes com uma razão de desbalanceamento igual à 0,01/0,99. Neste cenário, uma acurácia da ordem de 0,99 possivelmente pode estar classificando erroneamente todas as amostras da classe minoritária (que aparece em menor número).

Neste caso, é mais interessante distinguir os erros e acertos que são cometidos por um classificador, para cada classe existente. Para um problema de classificação binária (classes positiva e negativa), estas informações estão discriminadas nas seguintes medidas:

- Verdadeiros Positivos (VP)
- Verdadeiros Negativos (VN)
- Falsos Positivos (FP)
- Falsos Negativos (FN)

Estas quatro medidas podem ser combinadas para obter várias métricas que avaliam diferentes aspectos de um classificador. Uma destas é a Sensibilidade que mede a porcentagem de amostras verdadeiramente positivas que foram classificadas como positivas, dada por,

$$Sens = \frac{VP}{VP + FN} \quad (2.11)$$

Uma outra métrica é a Especificidade, que mede a porcentagem de amostras verdadeiramente negativas que foram classificadas como tal. A especificidade é expressa como,

$$Espec = \frac{VN}{VN + FP} \quad (2.12)$$

A Área Abaixo da Curva ROC (AUC) (Hanley e McNeil, 1982) é uma métrica robusta que avalia o desempenho geral de um classificador sem considerar um limiar de decisão (t_d) específico. Esta métrica pode ser expressa como,

$$AUC = \int_{-\infty}^{+\infty} Sens(t_d)(1 - Espec(t_d))dt_d \quad (2.13)$$

A área expressa pela equação 2.13 representa a área presente no gráfico ROC (Fawcett, 2006) de duas dimensões, em que a métrica Taxa de Verdadeiros Positivos ($VP_{taxa} = VP/(VP + FN)$) é plotada no eixo vertical e a Taxa de Falsos Positivos ($FP_{taxa} = FP/(FP + VN)$) é plotada em seu eixo horizontal, a partir da variação do limiar de decisão. Assim, a métrica AUC resume estas duas medidas em um único valor.

2.5.2 Validação Cruzada

Para um problema em que está disponível uma grande quantidade (relativa¹) de dados, a melhor forma para lidar com as tarefas de seleção e avaliação de modelos, é por meio da divisão aleatória do conjunto de dados em três partes: treinamento, validação e teste, podendo corresponder a: 50%, 25% e 25%, respectivamente. Nesta abordagem, o conjunto de teste deve ser utilizado somente no final do processo, para mensurar o erro de generalização do modelo tido como o escolhido. Porém não é sempre que se dispõem de um vasto conjunto de dados para proceder desta maneira. Para casos em que os dados estão em quantidade limitada, pode-se utilizar o método de Validação Cruzada (CV).

A validação cruzada é uma técnica popular, e usualmente utilizada na seleção e avaliação de modelos de aprendizado (Arlot et al., 2010). A ideia por trás do método CV é a divisão do conjunto de dados disponíveis em duas ou mais partes, afim de estimar o desempenho esperado do(s) modelo(s) treinado(s). Neste particionamento, uma parcela dos dados é utilizada efetivamente no treinamento do modelo, com a estimação de seus parâmetros, enquanto que o restante dos dados são empregados na estimação de desempenho do algoritmo de aprendizado. O primeiro conjunto recebe o nome de Treinamento e o segundo de Validação. Na execução do método, o particionamento dos dados acontece um determinado número de vezes (predeterminado), sendo que para cada treinamento, utiliza-se diferentes conjuntos de treinamento e validação.

O método de CV apresenta diferentes formas de divisão do conjunto de dados. Alguns exemplos são: Validação Cruzada por k-partições (k-fold) e Validação Cruzada por Unidade (LOOCV) e *Hold-out*. Cada uma destas configurações possui diferentes aspectos quanto ao custo computacional, grau de viés e variância para a métrica de avaliação, além do próprio modo de divisão do conjunto de dados. A Figura 2.3 mostra a partição do conjunto de dados

¹Não existe uma regra para se definir a quantidade de dados necessária ao treinamento de um modelo (Hastie et al., 2009). É necessário que exista uma quantidade mínima para estimar os parâmetros do modelo, e que esta quantidade seja representativa na descrição do problema

para os tipos k-fold e LOOCV.

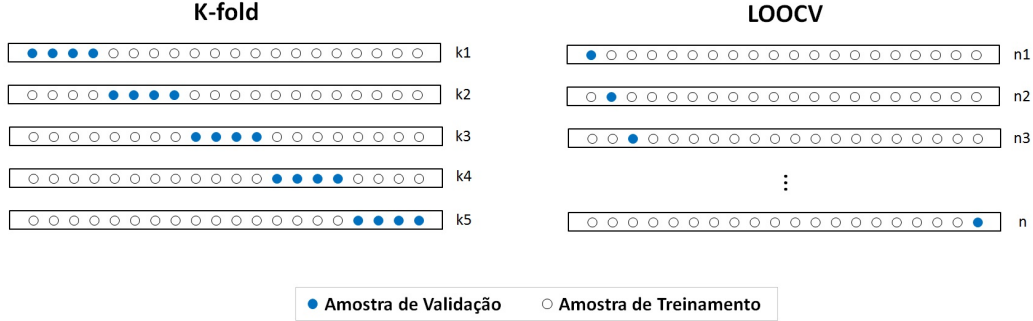


Figura 2.3: Esquema com os tipos de validação cruzada: k-fold e LOOCV.

A validação cruzada do tipo LOOCV, utiliza como conjunto de treinamento $n - 1$ observações, de um total de n instâncias, separando apenas uma (1) amostra para a validação. Em cada treinamento realizado, a amostra de validação é trocada de modo que todas as amostras existentes sejam utilizada para esta finalidade. Ou seja, para um conjunto de n amostras, haverá n treinamentos. Esta forma de realizar a CV possui um alto custo computacional, uma vez que um grande número de treinamentos pode ocorrer, dependendo do conjunto utilizado, chegando a ser inviável em alguns casos. A validação cruzada no modo LOOCV é um estimador não enviesado para o erro de teste, mas que exibe alta variância por se basear em apenas uma observação (Hastie et al., 2009). O resultado da avaliação é dado pela média simples da avaliação de cada rodada,

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n \mathcal{L}_i \quad (2.14)$$

Já a validação cruzada por k partições define um valor de k_{cv} que usualmente recebe o valor de 5 ou 10, para delimitar em quantas partições de iguais tamanhos, o conjunto de dados será dividido. Então, para cada treinamento na CV, uma partição é usada como conjunto de validação e as demais como conjunto de treinamento. Esta abordagem possui um custo computacional baixo (comparado ao modo LOOCV), com um moderado grau de viés para a avaliação estimada. O resultado é obtido por meio da média simples da avaliação (\mathcal{L}) de cada treinado realizado,

$$CV_{(k_{cv})} = \frac{1}{k_{cv}} \sum_{i=1}^{k_{cv}} \mathcal{L}_i \quad (2.15)$$

O problema de otimização hiperparamétrica, visto através da validação cruzada (k-fold), é definido pela minimização (ou dependendo da função de perda, pela maximização) da seguinte equação,

$$f(\lambda) = \frac{1}{k_{cv}} \sum_{i=1}^{k_{cv}} \mathcal{L}(A_{\lambda}, \mathcal{D}_{train}^{(i)}, \mathcal{D}_{valid}^{(i)}), \quad (2.16)$$

onde, $\mathcal{D}_{train}^{(i)}$ e $\mathcal{D}_{valid}^{(i)}$ são, respectivamente, os conjuntos de treinamento e validação para a partição k_{cv} no método CV, e λ é um vetor contendo os hiperparâmetros do algoritmo de aprendizado (A).

2.5.2.1 Amostragem Aleatória Estratificada

A Amostragem Aleatória Estratificada é um método que amostra instâncias, que leva em consideração a existência de grupos disjuntos em uma população, para produzir amostras cujas proporções dos grupos são mantidas. Quando a técnica de validação cruzada é empregada na estimação do erro em um problema de classificação, cuja as classes apresentam desbalanceamento (diferentes proporções), o método de amostragem aleatória estratificada garante que cada partição da CV possua a mesma relação do número de instâncias de cada classe, tal qual como a existente no conjunto total de dados. Isto é muito importante para manter a representatividade dos dados em cada partição que é usada na estimação do desempenho do classificador.

2.6 Árvores de Decisão

2.6.1 Introdução

Uma Árvore de Decisão é um modelo não-paramétrico que modela relações complexas entre as entradas e saídas de um problema de classificação ou regressão, sem a necessidade de assumir hipóteses *a priori*.

Árvores de decisão são modelos de aprendizado que possuem a capacidade de tratar de atributos do tipo numérico, categórico ou ambos. Uma árvore de decisão implementa, intrinsecamente, a seleção de características, o que proporciona a este algoritmo uma robustez na tratativa de casos em que haja variáveis irrelevantes ou que apresentem ruído. Além disto, árvores de decisão possuem fácil interpretabilidade quanto à suas regras de predição (Louppe, 2014). Estas particularidades fazem deste modelo, um algoritmo de aprendizado popular e muito difundido (Criminisi e Shotton, 2013; Wu et al., 2008).

Existem diferentes algoritmos que implementam uma árvore de decisão. Alguns exemplos destes métodos são: Árvores de Classificação e Regressão (CART) (Breiman et al., 1984), ID3 (Quinlan, 1986) e C4.5 (Quinlan, 1993).

O método ID3 induz o crescimento de uma árvore de decisão a partir de um conjunto de dados com atributos e variável de saída. O algoritmo processa recursivamente, a seleção de um dos atributos que ainda não foi usado, e que apresenta a menor entropia entre os avaliados para ser usado no procedimento de partição. O método C4.5 é uma extensão do algoritmo ID3, que lida com dados numéricos e categóricos, produzindo regras de divisão do espaço de atributos que podem resultar em duas ou mais sub-áreas (para atributos do tipo categórico). Este algoritmo utiliza como métrica de pureza: a entropia e a razão de ganho. Seu procedimento para crescer uma árvore possui como critério de interrupção, a ocorrência de pureza em um nó,

ou quando o número de pontos de uma região está abaixo de um limite predeterminado para a divisão do nó. Após o crescimento da árvore, é realizado a remoção de nós de sua estrutura, por meio da técnica Poda Pessimista que estima o erro de teste baseado em amostras mal classificadas do conjunto de treinamento. Esta avaliação acontece recursivamente, estimando-se a razão de erro associada a um nó da árvore baseada na razão de erros de suas ramificações para decidir qual a melhor estrutura para o modelo (Quinlan, 2014).

O algoritmo CART² não foi o primeiro modelo de árvore de decisão introduzido no aprendizado estatístico, contudo, foi o primeiro a ser descrito com rigor analítico e apoiado por uma teoria de probabilidade e estatística sofisticada (Wu et al., 2008). Assim como o método C4.5, o algoritmo CART consegue lidar com dados numéricos e categóricos. Este algoritmo executa primeiramente o crescimento máximo da árvore, por meio de um particionamento binário recursivo, produzindo sempre 2 sub-regiões, sendo executado sem um critério de paradas até que haja apenas um ponto em cada sub-região. Então a árvore passa por um processo conhecido como Poda de Custo-Complexidade, que ocorre no sentido reverso do crescimento de suas ramificações. Este mecanismo resulta em uma sequência de sub-árvores candidatas à modelo ótimo. Cada uma destas é obtida através da simples remoção de um nó interno, e respectivas folhas, de uma sub-árvore anterior. Deste modo, a última árvore da série apresenta apenas uma única folha (Hastie et al., 2009). Este procedimento visa selecionar o melhor modelo entre as sub-árvores, podendo para isto, utilizar a técnica CV.

2.6.2 Definição de uma Árvore de Decisão para Classificação

Dado um problema de classificação, representado por conjunto de dados $(x_{1:n}, y_{1:n})$ em que x_i são os atributos e y_i o rótulo, uma árvore de decisão particiona o espaço dos atributos de modo recursivo durante o crescimento da árvore, dividindo (*split*) uma região R_j em sub-regiões e atribuindo um valor de saída (γ_j) à cada uma destas. Para o caso de um problema de classificação, o valor de saída (γ_j) pode ser definido a partir da moda ou das probabilidades (para cada classe), tomadas com base nos pontos que habitam a respectiva R_j . Durante o crescimento da árvore, o particionamento ocorre até que cada ponto do conjunto de treinamento esteja sozinho em uma região R_j , ou que esta possua um grau máximo de pureza, ou então, até que um critério de parada ocorra. Ao final do crescimento da árvore, haverá um número de $j = 1, 2, \dots, J$ regiões disjuntas. Então a regra de predição é dada por,

$$T(\mathbf{x}; \Theta) = \sum_{j=1}^J \gamma_j I(\mathbf{x}_i \in R_j), \quad (2.17)$$

onde \mathbf{x} é o vetor com os atributos, $\Theta = \{R_j, \gamma_j\}_1^J$, γ_j é a saída atribuída à região R_j , $I()$ é uma função de indicação que retorna 1 se $\mathbf{x}_i \in R_j$, e 0 caso contrário.

A Figura 2.4 mostra um exemplo com o resultado do particionamento feito por uma árvore

²Este trabalho, utilizou em seus experimentos, uma versão otimizada do modelo CART implementada na biblioteca de aprendizado de máquina *Scikit-learn* para a linguagem de programação Python.

2.6 Árvores de Decisão

de decisão, para um conjunto de dados sintéticos que representa um problema de classificação binária com dois atributos. Os dados são divididos em treinamento (60%) e teste (40%). O gráfico mostra a métrica AUC alcançada pelo classificador.

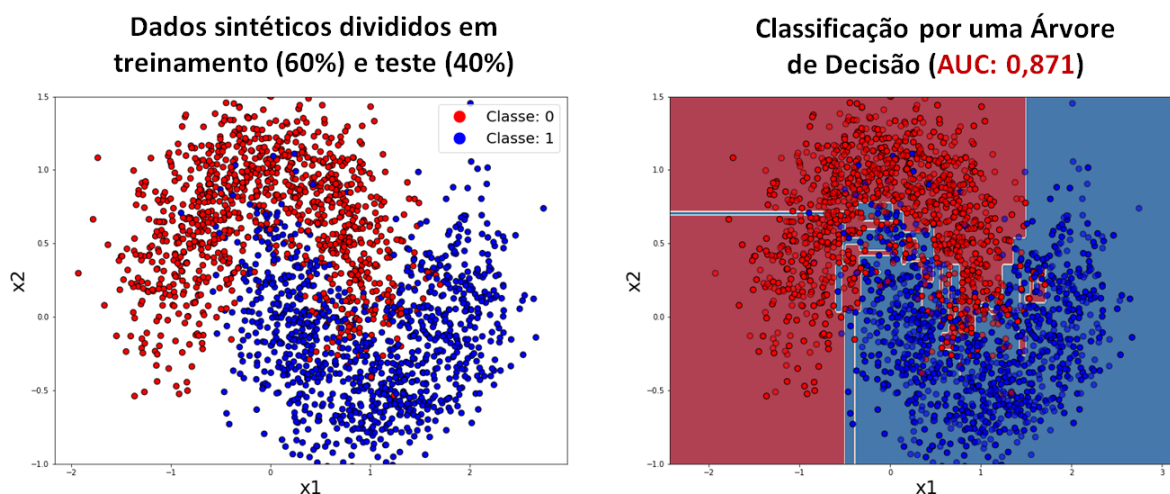


Figura 2.4: Gráficos com conjunto de dados sintéticos representando um problema de classificação binária com 2 atributos (à esquerda), e particionamento feito por uma árvore de decisão, com o respectivo valor de AUC atingindo ao se usar 60% dos dados para treinamento e 40% para teste (à direita). A classificação é dada pela cor que pinta cada partição, sendo a tonalidade uma escala degradê que reflete a estratificação das classes.

2.6.3 Estrutura de uma Árvore de Decisão

Uma árvore de decisão possui uma estrutura composta por nós (j) que são arranjados como em uma ramificação. Estes nós se apresentam em dois tipos distintos: o *interno* e o *terminal*. Um nó interno representa um particionamento no qual se determinou o atributo x_i e um valor de corte (s_j) que melhor estratifica (ou segmenta) a respectiva região R_j . Na maior parte das vezes, este *split* particiona o espaço do atributo em apenas duas partes (Árvore Binária). O uso de apenas duas sub-regiões evita a brusca fragmentação dos dados e a posterior existência de poucos pontos para os próximos *splits* (Hastie et al., 2009). Assim, um nó interno é composto por uma regra do tipo *Se-Então*, como $x_i \leq s_j$, para valores numéricos³ ou $x_i = \text{categorico}_j$ (onde categorico_j é uma das categorias possíveis) para os do tipo categórico. Então, como saída há apenas duas possibilidades (*verdadeiro* ou *falso*). Já um nó terminal (folha), como o próprio nome indica, está posicionado em uma das extremidades da estrutura do modelo. Este tipo de nó define o valor de saída em uma predição, caso a amostra apresentada ao modelo atinja esta terminação da árvore. A Figura 2.5 mostra a Árvore de Decisão CART treinada com 60% dos

³É possível formas mais elaboradas como a combinação linear $a_j x_i \leq s_j$, em que a_j é uma constante, que possibilita a melhoria da predição, ao custo de se reduzir o interpretabilidade do modelo.

dados sintéticos da Figura 2.4.

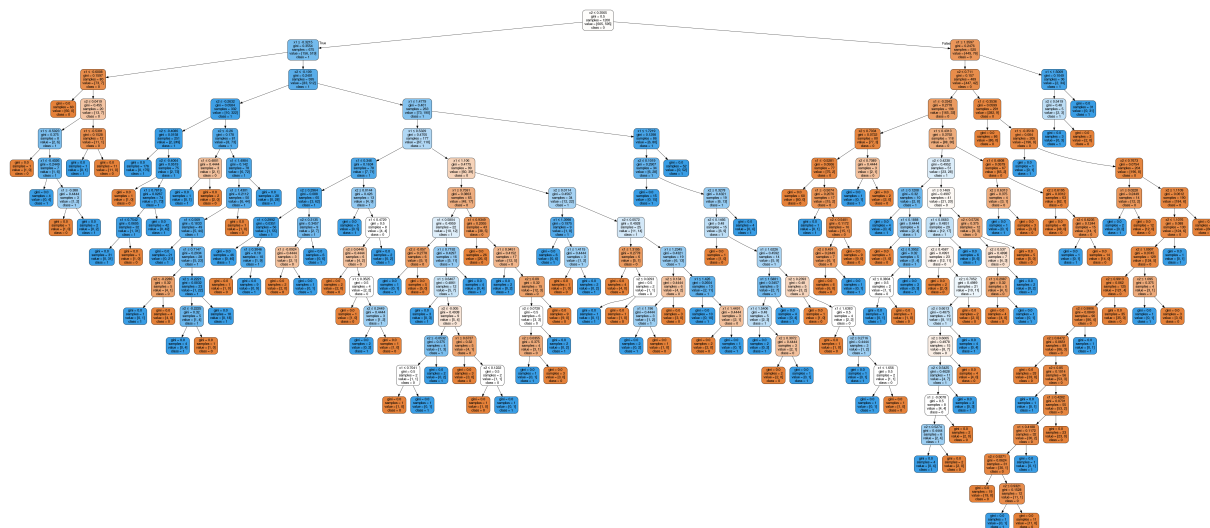


Figura 2.5: Árvore de Decisão CART construída com o conjunto de dados sintéticos da Figura 2.4. Na imagem, cada nó possui: o atributo considerado no *split* e seu valor de corte, o valor de pureza (gini), o número de amostras, a estratificação para as classes e a classe dominante. O caminho à esquerda de um nó representa valores verdadeiros (em relação ao critério de *split*), e à direita os falsos.

2.6.4 Métricas de Impureza

No treinamento de uma árvore de decisão é utilizado alguma função para medir o grau de impureza (ou pureza) $i(j)$ induzido pela divisão de um nó j em sub-regiões (Breiman et al., 1984). Quanto mais puro é um nó, melhor é a sua predição. Assumindo que quanto menor o valor de $i(j)$ mais puro é o nó j , o decrescimento da pureza do mesmo, quando este se divide resultando em: saída falsa (j_F) e verdadeira (j_T) (árvore binária), é expresso como,

$$\Delta i(s_j, j) = i(j) - p_T i(j_T) - p_F i(j_F), \quad (2.18)$$

onde p_T e p_F são as respectivas proporções $\frac{N_{j_T}}{N_j}$ e $\frac{N_{j_F}}{N_j}$, em que N_j representa a quantidade de amostras de treinamento que passam pelo nó j , N_{j_T} e N_{j_F} são número de vezes, entre os pontos de N_j , que uma amostra tomou o caminho Verdadeiro e Falso, respectivamente.

Em problemas de classificação, usualmente são utilizadas as seguintes métricas para medir o grau de pureza dos nós de uma árvore: índice de Gini e entropia cruzada (Hastie et al., 2009),

que aqui se apresentam da seguinte maneira,

$$\text{Índice de Gini} = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}), \quad (2.19)$$

$$\text{Entropia Cruzada} = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}, \quad (2.20)$$

onde, \hat{p}_{mk} representa a proporção de pontos para uma determinada classe k (de um total de K existentes), que habita a região R_j . Para o caso de um problema de classificação binária, estas métricas são dadas por,

$$\text{Índice de Gini} = 2p(1 - p), \quad (2.21)$$

$$\text{Entropia Cruzada} = -p \log p - (1 - p) \log(1 - p), \quad (2.22)$$

onde p é a proporção de uma das duas classes existentes.

O índice de gini é uma medida da variância total entre as K classes existentes e que estão presentes em uma região específica que se avalia. Esta medida mostra um baixo valor quando todos os \hat{p}_{mk} estão próximos de 0 ou 1, o que descreve um cenário de relativa pureza com a predominância de apenas uma única classe nesta região. Entropia cruzada e índice de gini são métricas mais sensíveis às mudanças causadas pela divisão de um nó quando comparadas ao Erro de Má Classificação, equação 2.6. Por isto, uma destas medidas deve ser utilizada quando se constrói uma árvore de decisão (Hastie et al., 2009).

2.6.5 Indução de um Árvore de Decisão

A indução (ou crescimento) de uma árvore de decisão é um problema de Otimização Combinatória⁴ que envolve a escolha do melhor par (x_i, s_j) para cada nó j do modelo. Para isto, os parâmetros da árvore, $\Theta = \{R_j, \gamma_j\}_1^J$, são encontrados através da minimização do risco empírico, dado por,

$$\hat{\Theta} = \arg \min_{\Theta} \sum_{j=1}^J \sum_{x_i \in R_j} \mathcal{L}(y_i, \gamma_j). \quad (2.23)$$

A resolução deste problema de otimização é dividida em duas partes (Hastie et al., 2009):

1. Dado uma região R_j , estima-se o valor para γ_j utilizando alguma medida de impureza. Esta parte é usualmente trivial de se obter;
2. Já a determinação de R_j é uma tarefa difícil, em que aproximações são realizadas. Uma estratégia usual neste caso é o crescimento recursivo da árvore, a partir de

⁴Em um problema de otimização combinatória, procura-se pela solução ótima entre um conjunto finito de possibilidades.

sua raiz, atuando em um nó de cada vez, sem um planejamento antecipado.

2.6.6 Poda e Hiperparâmetros

Analisando o modo como se dá o treinamento de árvores de decisão, é intuitivamente fácil perceber que se o procedimento que cria novos nós for realizado indefinidamente, até que seja atingido folhas com apenas uma única amostra, haverá como resultado nós terminais exclusivamente com pureza máxima. Isto pode fazer da árvore um modelo complexo com muitas regras e uma relativa chance de um sobreajuste, reduzindo assim o desempenho de predição. Por outro lado, uma árvore muito rasa (com poucos *splits*) pode não conseguir capturar toda a estrutura presente no conjunto de dados. Então, é preciso achar um balanceamento no crescimento de uma árvore, afim de reduzir o erro de generalização.

O tamanho de uma árvore de decisão é governado por algum hiperparâmetro que deve ser estimado para um melhor ajuste do modelo aos dados do problema. O algoritmo CART utiliza o método de poda de custo-complexidade, o qual cresce uma árvore até o seu tamanho máximo T_0 e obtêm, a partir desta, sub-árvores $T \in T_0$ que eliminam um certo número das ramificações de T_0 . O critério de custo-complexidade é dado por,

$$C_{\alpha_{cut}}(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha_{cut}|T|, \quad (2.24)$$

onde, N_m é o número de pontos presentes em uma folha m , do total de $|T|$ existentes, $Q_m(T)$ é a métrica de pureza utilizada, e α_{cut} é o hiperparâmetro que controla o tamanho da árvore.

Um valor grande para α_{cut} resulta em uma árvore rasa, e vice-versa. De fato o uso de $\alpha_{cut} = 0$ acarreta a obtenção da própria árvore original T_0 . Para estimar o valor ótimo de α_{cut} é utilizado o método de poda do galho mais fraco *weakest link pruning* que sucessivamente remove o nó interno, e respectivas folhas, que produzem o menor aumento em $\sum_{m=1} N_m Q_m(T)$, até que reste apenas o nó da raiz. A técnica de validação cruzada é usada ao longo desta sequência para estimar o erro de cada sub-árvore candidata ao modelo ótimo.

A poda de custo-complexidade é um método que atua após o crescimento da árvore. Uma outra forma de tratar da questão da profundidade, e que pode ser realizada durante o crescimento da mesma, é através de hiperparâmetros que levam em consideração o quão bom está o valor da métrica de pureza em um nó, ou se alguma configuração satisfatória foi atingida (Louppe, 2014). Ações que podem ser adotados neste caso, agem através dos seguintes hiperparâmetros:

- Determinação do número mínimo de pontos, abaixo do qual um nó se torna terminal (folha), não ocorrendo assim o *split* (hiperparâmetro N_{min}).
- Controle do tamanho da árvore, determinando a profundidade máxima que um nó pode atingir (hiperparâmetro d_{max}).

- Limiar mínimo que define se um nó será dividido, comparando-o com seu grau de pureza pós divisão (hiperparâmetro β).
- Determinação do número mínimo de pontos que deve haver em cada folha da árvore (hiperparâmetro N_{folha}).

A Figura 2.6 mostra a estrutura de árvores de decisão (CART) que utilizaram estes quatro hiperparâmetros (N_{min} , d_{max} , β e N_{folha}) no treinamento com os dados da Figura 2.4. E a Figura 2.7 mostra gráficos com a classificação realizada por cada uma das quatro árvores da Figura 2.6. Um ponto importante é o valor da avaliação de classificação dada pelo métrica AUC. O uso de um hiperparâmetro que controla o crescimento de uma árvore propicia galgar uma redução no do erro de generalização do modelo (vide comparação com o resultado da Figura 2.5). Nestas demonstrações, os quatro hiperparâmetros foram utilizados isoladamente, em cada árvore, porém os mesmos podem ser usados em conjunto para tentar obter melhores resultados do ajuste entre a estrutura da árvore e os dados de treinamento.

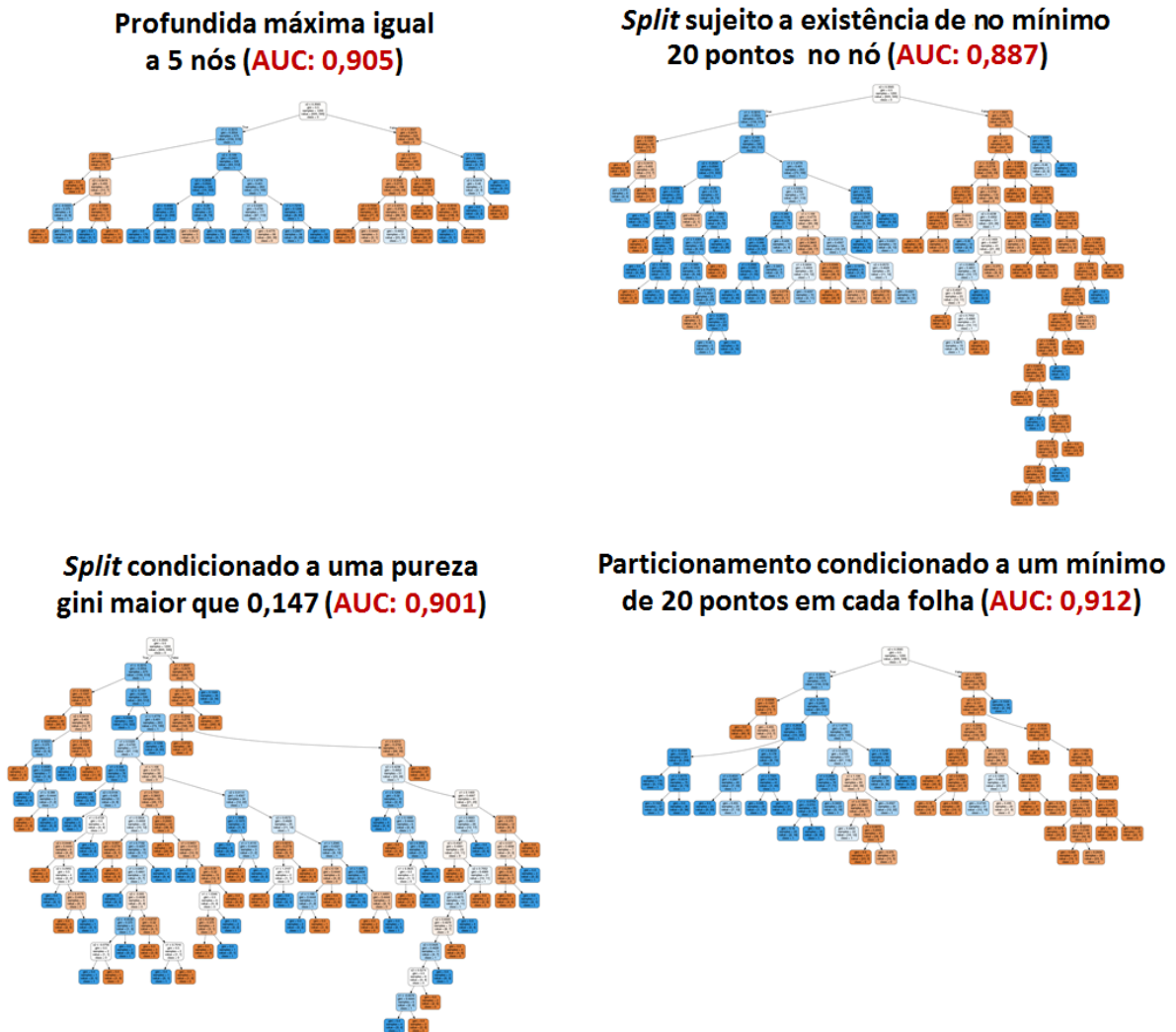


Figura 2.6: Árvores de decisão (CART) construídas com o conjunto de dados da Figura 2.4. Partindo do quadrante superior direito (no sentido horário) cada modelo fez uso do seguinte hiperparâmetro: N_{min} , N_{folha} , β e d_{max} , respectivamente.

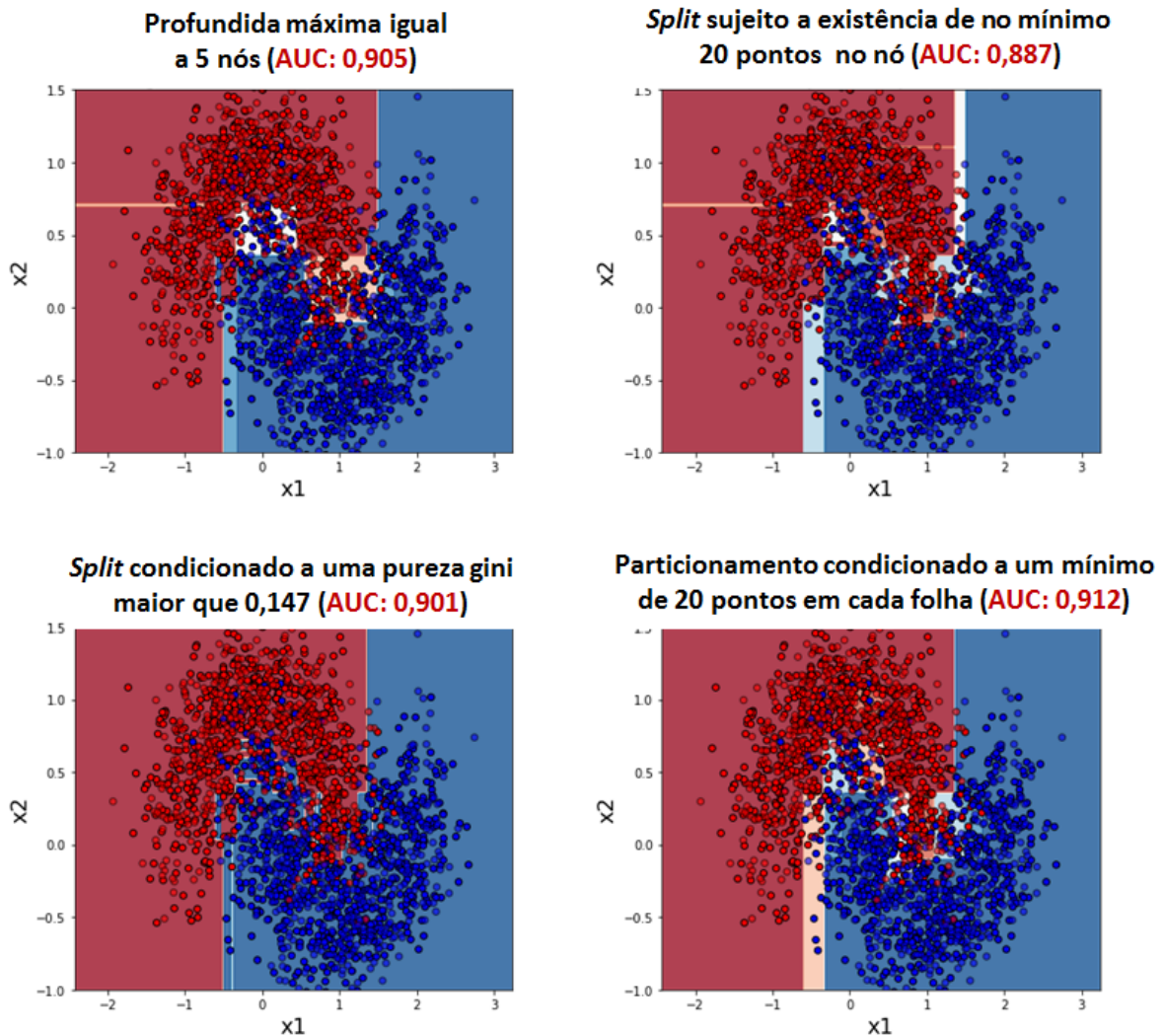


Figura 2.7: Os gráficos mostram a classificação do conjunto de dados da Figura 2.4, feito por cada árvore de decisão que está ilustrada na Figura 2.6.

Apesar do modelo de árvore de decisão possuir a capacidade de modelar as relações complexas de um problema, este algoritmo não é usualmente competitivo quando comparado às melhores abordagens de aprendizado supervisionado (James et al., 2013). Isto porque árvores de decisão são modelos que possuem alta variância, devido à chance de ocorrer uma série de diferentes *splits*, caso uma pequena mudança no conjunto de dados ocorra (Hastie et al., 2009). Porém, ao se considerar como modelo preditivo, não apenas uma (1) árvore de decisão, mas várias destas em conjunto (como o modelo de Floresta Aleatória o faz), o problema relacionado a variância é tratado. Esta questão é vista na subseção a seguir.

2.7 Floresta Aleatória

2.7.1 Introdução

Floresta Aleatória é um modelo baseado em árvores de decisão, que lida bem com conjunto de dados de alta dimensão, e com presença de multicolinearidade (Hastie et al., 2009; Belgiu e Drăguț, 2016). Este tipo de modelo é usualmente utilizado não apenas para classificação, mas também para regressão, estudo de importância e seleção de variáveis, e detecção de *outliers* (Verikas et al., 2011).

O modelo de Floresta Aleatória é usado em diversas aplicações que requerem o aprendizado a partir de dados, como as áreas de diagnóstico médico por imagem (Criminisi e Shotton, 2013; Criminisi et al., 2013), predição rápida de movimentos (Shotton et al., 2013), sensoriamento remoto (Belgiu e Drăguț, 2016), análise de *big data* (Genuer et al., 2017; Del Río et al., 2014), detecção de falhas (Yang et al., 2008), dentre outras.

Para compreender o funcionamento de uma floresta aleatória, e os detalhes envolvidos em seu processo de treinamento e predição, é oportuno introduzir as técnicas Bootstrap e Bagging, expostas a seguir.

2.7.2 Bootstrap

A técnica Bootstrap (Efron, 1979) é um método de reamostragem que pode ser utilizado com o propósito de medir o desempenho de um modelo de aprendizado, ou até mesmo empregada na melhoria da predição deste último. Sua forma geral pode ser definida como: dado um conjunto de dados $\mathcal{D} = (\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$, a técnica bootstrap amostra instâncias aleatoriamente e com reposição, a partir de \mathcal{D} , para formar novos conjuntos de mesmo tamanho que o original, sendo estes representados como $\mathbf{Z} = (\mathbf{Z}^1, \mathbf{Z}^2, \dots, \mathbf{Z}^B)$, onde B é o número total de novos conjuntos criados. A Figura 2.8 ilustra a criação do conjunto \mathbf{Z} a partir de uma simples amostra contendo 3 instâncias.

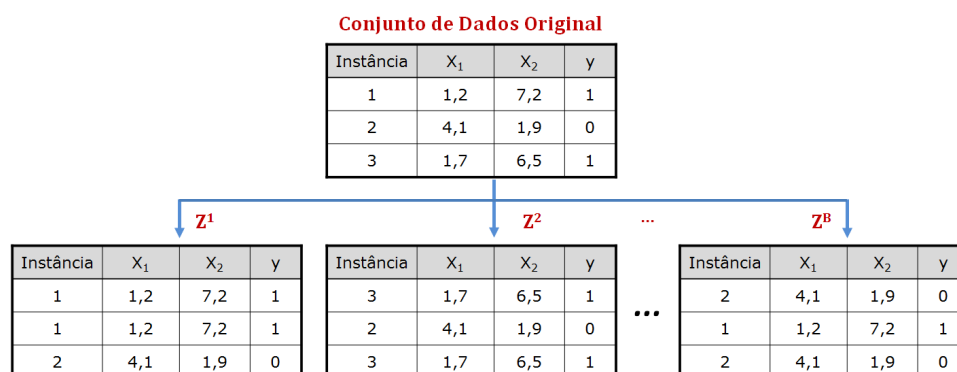


Figura 2.8: Ilustração da técnica bootstrap, na criação de B conjuntos reamostrados, com reposição, a partir do conjunto original contendo 3 instâncias.

2.7.3 Bagging

O método Bagging, também conhecido por Bootstrap Aggregating (Breiman, 1996a), é uma técnica que constrói múltiplas versões de um preditor para usá-las em conjunto, de forma agregada, de modo que a saída para uma determinada entrada apresentada ao grupo é tomada através de uma votação majoritária utilizando as respostas obtidas dos preditores. Para a construção dos B preditores do modelo, o método bagging utiliza a técnica bootstrap para gerar conjuntos de dados diferentes para cada um destes, a partir de $\mathcal{D} = (\mathbf{x}_{1:n}, y_{1:n})$. Ou seja, para cada amostra bootstrap $\mathbf{Z}^b, b = 1, 2, \dots, B$ um preditor $\hat{f}^{*b}(\mathbf{x})$ é ajustado. Então a resposta de estimação é dada por,

$$\hat{f}_{bag}(\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(\mathbf{x}_i). \quad (2.25)$$

A ideia essencial do método bagging é tirar a média das respostas de vários preditores que apresentam certa variância, mas que são modelos aproximadamente não enviesados. A média de B variáveis aleatórias i.i.d. (independente e identicamente distribuída), cada uma com variância σ^2 , resulta em uma média com variância igual a $\frac{\sigma^2}{B}$ (Hastie et al., 2009). Se estas variáveis são apenas i.d. (identicamente distribuídas) com uma correlação positiva par-a-par ρ , a variância da média se torna,

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2. \quad (2.26)$$

Com o crescimento de B , o segundo termo tende a desaparecer, permanecendo o primeiro com a correlação par-a-par ρ .

2.7.4 Definição de uma Floresta Aleatória

Florestas Aleatórias são modelos do tipo *ensemble methods*, que combinam a predição de um conjunto de árvores de decisão, para obter uma única resposta como saída, que tende a apresentar melhor desempenho que as obtidas com cada árvore do modelo em separado, devido a redução de variância. A equação a seguir descreve a saída para este modelo,

$$\hat{f}_{fa}^B(\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^B T(\mathbf{x}_i, \Theta_b), \quad (2.27)$$

onde, B é o número total de árvores, $T()$ representa a resposta de uma árvore b para um vetor de entrada \mathbf{x}_i , e Θ_b representa os parâmetros desta árvore.

A construção de uma Floresta Aleatória envolve o uso da técnica de bootstrap para criar subconjuntos de dados utilizados no crescimento das árvores do modelo. Porém, diferente do método bagging, o modelo de floresta aleatória possui a característica peculiar de utilizar somente um certo número (m) do total de d atributos existentes no conjunto de dados para

realizar os *splits* dos nós. Mais especificamente, o que ocorre é que cada vez que um nó de uma árvore é avaliado para a divisão, somente m atributos, que são escolhidos aleatoriamente, são considerados para o particionamento.

Na construção de preditores através do método bagging, em média dois terços das amostras do conjunto de dados são utilizadas no treinamento de cada preditor. O restante das amostras recebe o nome *Out-of-bag* (OOB) (Breiman, 1996b). Uma característica importante de uma floresta aleatória é a possibilidade do uso destas amostras para estimar o desempenho do modelo, e também como critério de parada no treinamento deste último, por meio da interrupção do acréscimo de novas árvores à floresta aleatória, com base no valor da estimativa OOB calculada. Esta estimativa é feita tomando a média do desempenho da classificação para cada amostra apresentada a floresta aleatória, com a peculiaridade de considerar somente as árvores de decisão que não utilizaram a amostra em questão em seus treinamentos.

2.7.5 Características e Hiperparâmetros de uma Floresta Aleatória

O número de árvores (B) é um hiperparâmetro do modelo de floresta aleatória que, em geral, quanto maior é seu valor, melhor a acurácia de predição. Porém, o aumento deste número é benéfico até certo ponto, uma vez que a partir de certa quantidade de árvores, a melhoria na resposta combinada cessa, além do fato de um maior número de árvores consumirem uma maior quantidade de recursos computacionais. A Figura 2.9 mostra a relação entre o número de árvores e o erro de classificação (AUC), de uma floresta aleatória que utilizou os dados da Figura 2.4.

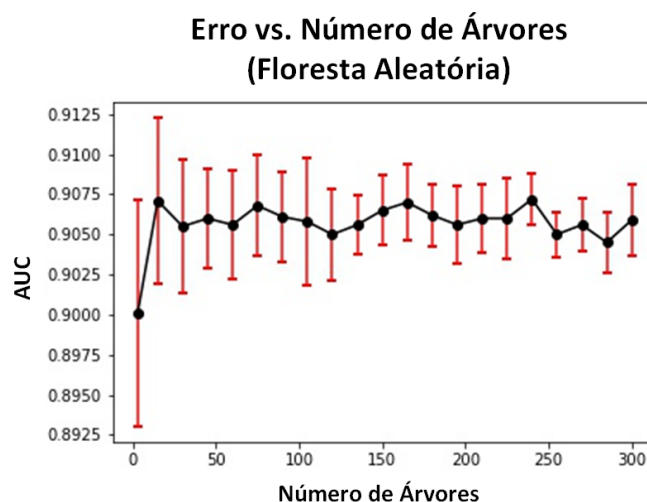


Figura 2.9: Relação entre número de árvores e erro de predição (utilizando a métrica AUC), de uma floresta aleatória empregada na classificação dos dados da Figura 2.4, cujo hiperparâmetro $d_{max} = 5$ foi utilizado. Os dados apresentados no gráfico são média e desvio padrão de 10 execuções.

O hiperparâmetro m controla o grau de decorrelação entre as árvores de uma floresta aleatória. Este modelo promove uma melhoria na redução da variância proporcionada pelo método de bagging 2.26, através do ajuste da correlação existente entre as árvores (m). Esta ação propicia uma melhora da generalização do modelo e de sua robustez (Criminisi e Shotton, 2013). Quando o número de atributos no conjunto de dados é grande, porém com poucos que exibam relevância para a predição, a floresta aleatória tende a apresentar um pobre desempenho caso se utilize um pequeno valor para m (Hastie et al., 2009). Isto ocorre devido a menor chance de num *split*, um bom (e "raro") atributo ser amostrado. Valores usualmente utilizados para m são: \sqrt{p} para classificação, e $p/3$ para regressão. Porém, este hiperparâmetro depende do problema abordado, requerendo assim uma sintonia.

Em uma floresta aleatória, há ainda os hiperparâmetros que estão relacionados a suas árvores, como: critério de pureza e hiperparâmetros que controlam a profundidade do modelo. Em Segal (2004), o autor faz experimentos com o modelo de floresta aleatória e conjunto de dados do repositório UCI, ajustando hiperparâmetros relacionados ao crescimento de uma árvore.

2.7.6 Construção de uma Floresta Aleatória

O Pseudocódigo 1 mostra um procedimento para construir uma Floresta Aleatória, para a qual o hiperparâmetro d_{\max} é usado como critério de ajuste do modelo aos dados do problema. Para isto, são utilizados: um conjunto de dados \mathcal{D} (com p atributos), uma métrica de pureza Q_m e os hiperparâmetros: número de árvores (B), quantidade de atributos para *split* (m) e profundidade máxima de um nó (d_{\max}). Para a predição de uma amostra x_i , a floresta aleatória apresenta esta informação a cada árvore do modelo, iniciando de suas raízes até atingir as correspondentes folhas. Então a equação 2.27 é utilizada para obter a saída.

Pseudocódigo 1 Construção de uma Floresta Aleatória

Entrada: $Q_m, B, d_{\max}, \mathcal{D}$

Saída: modelo de Floresta Aleatória \hat{f}_{fa}^B

para $b=1$ **até** B **faça**

- a) Amostre os dados de treinamento utilizando o método bootstrap à partir de \mathcal{D} .
- b) Cresça a árvore T_b utilizando o conjunto amostrado, repetindo-se os próximos passos para cada nó terminal da árvore, até que o último nó (d_{\max}) seja alcançado:
 - i. selecione aleatoriamente m atributos do total de p existentes.
 - ii. estime o par (x_i, s_j) que maximiza o *split* do nó.
 - iii. divida o nó em dois nós filhos.

fim para

retorna \hat{f}_{fa}^B

Otimização Hiperparamétrica

3.1 Introdução

Em Aprendizado de Máquina, corriqueiramente se lança mão de um certo número de hiperparâmetros que se relacionam: ao pré-processamento de dados, à regularização, ao treinamento e à definição estrutural de um modelo, para obter um melhor desempenho de predição. O uso de hiperparâmetros com valores ótimos, possibilita lograr um modelo com melhor desempenho, evitando assim, problemas como: sobre-ajuste (*overfitting*¹), modelos simplistas com baixa Capacidade², dentre outros. Ou seja, hiperparâmetros são usados para configurar vários aspectos de um algoritmo de aprendizado, possuindo então um amplo espectro de influência sobre o resultado final do modelo.

A questão em torno dos Hiperparâmetros tem se destacado muito na última década. A importância dos hiperparâmetros está tão em evidência, que a pergunta "quão bom é este modelo para o conjunto de dados?", tornou-se mal formulada, abrindo espaço para o questionamento sobre a qualidade da melhor configuração de um modelo, que pode vir a ser descoberta em um certo intervalo de tempo, para uma dada tarefa (Bergstra et al., 2013).

Modelos de Aprendizado de Máquina raramente não possuem hiperparâmetros em sua estrutura (Snoek et al., 2012). Geralmente tais modelos apresentam no mínimo um (1) hiperparâmetro relacionado à capacidade do modelo em se ajustar com maior perfeição aos dados de treinamento, e dependendo do método utilizado no aprendizado, um número mínimo de hiperparâmetros ligados ao algoritmo de treinamento para definir características relacionadas à convergência, na busca pelos parâmetros do modelo. Na Tabela 3.1 é listado alguns modelos de aprendizado, distinguindo seus parâmetros dos hiperparâmetros³.

Na próxima seção é apresentada uma revisão bibliográfica sobre técnicas de otimização hiperparamétrica, seguida por uma seção que aborda dificuldades ligadas à determinação do valor ótimo de um hiperparâmetro. Nas seções que se seguem, são apresentados os métodos de otimização hiperparamétrica: busca em grade, busca aleatória e a otimização Bayesiana pela

¹O termo *overfitting* é empregado a um modelo com baixo erro de treinamento, porém com alto erro de teste.

²Para um melhor desempenho, um modelo deve apresentar uma Capacidade adequada à complexidade do problema apresentado.

³Somente os hiperparâmetros que estão relacionados ao modelo. Não incluindo assim os hiperparâmetros dos métodos de otimização utilizados no treinamento.

Tabela 3.1: Hiperparâmetros e parâmetros ordinários para alguns modelos de aprendizado.

Modelos	Parâmetros Ordinários	Hiperparâmetros
Árvore de Decisão	- atributo utilizado em cada nó - valor de corte em cada nó	- profundidade da árvore - critério de pureza
Floresta Aleatória	- atributo utilizado em cada nó (<i>de cada árvore</i>) - valor de corte em cada nó (<i>de cada árvore</i>)	- número de árvores - profundidade das árvores - quantidade de atributos no <i>split</i> - critério de pureza
SVM	- vetores de suporte	- tipo de <i>kernel</i> - constante regularização - constante de tolerância
MLP	- pesos e <i>bias</i>	- número de camadas e neurônios - funções de ativação
CNN	- pesos e <i>bias</i>	- número de camadas e seu arranjo - filtros (quantidade, dimensão e deslocamento) - tipo de <i>pooling</i> e dimensão da janela - funções de ativação

abordagem do processo gaussiano.

3.2 Revisão Bibliográfica

Em [Hsu et al. \(2003\)](#) os autores experimentam o método de busca em grade, utilizando a técnica de validação cruzada, para sintonizar os hiperparâmetros do modelo de Máquina de Vetores de Suporte (SVM). Os testes envolveram a sintonia do hiperparâmetro do *kernel*, função de base radial, e da constante de regularização do modelo. Já o trabalho apresentado em [Bergstra e Bengio \(2012\)](#), mostra que o uso do método de Busca Aleatória Pura, na sintonia de hiperparâmetros de modelos de redes neurais, é capaz de obter resultados próximos, e até melhores, que a técnica de Busca em Grade utilizando apenas uma fração do tempo deste último. Quando o experimento envolve o mesmo número de treinamentos para os métodos, a Busca Aleatória mostrou melhores resultados por possibilitar uma exploração mais efetiva de cada hiperparâmetro, segundo os autores.

Em [Hutter et al. \(2010\)](#), os autores apresentam o método SMAC que utiliza o modelo de Floresta Aleatória para modelar uma função de probabilidade *a posteriori* como uma distribuição gaussiana cuja média e variância são a média e a variância empírica dos preditores da floresta aleatória. O método utiliza a função de aquisição Esperança de Melhoria (EI), e para a sua maximização é usada uma busca local *multi-start* que considera, entre todos os resultados, os 10 maiores (melhores) pontos para reiniciar a busca nestes. O algoritmo SMAC é capaz de lidar com hiperparâmetros contínuos, categóricos e condicionais. Uma particularidade deste trabalho é o uso de problemas de sintonia de parâmetros e configuração de algoritmos, como forma de testar o desempenho do algoritmo. Não havendo assim, experimentos diretamente com modelos de aprendizado.

O método TPE é proposto em [Bergstra et al. \(2011\)](#) como um algoritmo de otimização Bayesiana, não convencional, que se difere das técnicas SMAC e GP no modo como o método

modela o problema. Na abordagem introduzida neste trabalho, duas funções de densidades são estimadas, com base em parte do conjunto de pares formados pelo vetor com os hiperparâmetros candidatos e a avaliação da função de perda. Para isto, é usado uma quantia fixa que divide esses pares entre as duas densidades que se estima. Os autores mostram a formulação da função de esperança de melhoria para esta abordagem, e apontam que a maximização da mesma pode ser feita empregando a geração de pontos aleatórios, em uma das densidades, e tomando o maior valor para a relação entre essas duas densidades estimadas. O método apresentado possui a capacidade de lidar com hiperparâmetros contínuos, categóricos e condicionais.

O problema de otimização hiperparamétrica, abordado pela otimização Bayesiana com o GP, é introduzido em [Snoek et al. \(2012\)](#). Neste trabalho são utilizados modelos de aprendizado: alocação latente de Dirichlet (LDA), SVM e redes neurais convolucionais (CNN). A técnica utiliza a função de aquisição EI. Os resultados obtidos pelo método Bayesiano conseguem atingir, e até superar, o desempenho alcançado pela otimização realizada por um humano *expert* na configuração hiperparamétrica. Os autores propõem a substituição da escolha padrão (*kernel* exponencial quadrático), como a função de covariância, pelo *kernel* Matérn 5/2, justificando que o primeiro possui um grau de suavidade não condizente com os problemas práticos de otimização. Em dois dos testes realizados, com um pequeno número de hiperparâmetros (2-4), o método GP é comparado com a técnica TPE, e exibe melhores resultados. O método apresentado consegue lidar com hiperparâmetros contínuos e inteiros, necessitando este último de arredondamento, pois o método o trata como contínuo.

Em [Thornton et al. \(2013\)](#), os autores tratam não somente da otimização hiperparamétrica, mas também da seleção do melhor modelo de aprendizado para um dado problema. Ou seja, a escolha do algoritmo de aprendizado é vista como um hiperparâmetro, também a ser otimizado, no problema CASH (*Combined Algorithm Selection and Hyperparameter Optimization Problem*). Este problema é tratado como uma combinação hierárquica, condicionando hiperparâmetros à ativação de um certo modelo avaliado. Os autores propõem uma ferramenta (Auto-WEKA) que faz uso dos métodos TPE e SMAC, e a testa no problema CASH, envolvendo 21 conjuntos de dados distintos e 39 modelos de classificação. Os resultados demonstram que o uso do método SMAC na ferramenta apresentada atinge melhores resultados.

O trabalho de Swersky et al. ([Swersky et al., 2014](#)) lida com a otimização Bayesiana, utilizando o GP, para a sintonia de hiperparâmetros em três problemas distintos. A novidade proposta está no método passível de interromper e retomar cada treinamento do modelo, tomando como base de decisão, a informação parcial da minimização do erro durante o processo de treinamento, que é comparada a uma distribuição *a priori* que caracteriza curvas de treinamento, de modo a prever se o corrente processo de aprendizado é pertinente ou não.

Os três métodos de otimização Bayesiana (SMAC, TPE e GP) são comparados em [Eggenberger et al. \(2013\)](#), por meio de diversos problemas envolvendo casos com um número pequeno (2-6), médio (14-38) e grande (786) de hiperparâmetros. Os testes são realizados utilizando um *cluster* com placas GPUs NVIDIA Tesla M2070s. Os resultados mostram um melhor resultado

do método GP⁴ para casos que envolvem um pequeno número de hiperparâmetros. Quando o número de hiperparâmetros é grande, os métodos SMAC e TPE se saem melhor. Nos testes os autores fazem uso de 10 repetições para cada método, em cada problema.

Em [Friedrichs e Igel \(2005\)](#) os autores propõem o uso da Estratégia Evolucionária de Adaptação da Matriz de Covariância (CMA-ES) para estimar os hiperparâmetros do *kernel* e da constante de regularização de máquinas de vetores de suporte.

O método de Gradiente, como forma de otimização hiperparamétrica, foi proposto por [Bengio \(2000\)](#). Em seus experimentos, o gradiente é utilizado para estimar um valor ótimo de hiperparâmetros contínuos em diversos casos simples de regressão linear com decaimento de pesos e em casos de predição de séries temporais, com hiperparâmetros que controlam a ponderação da importância dos pontos passados na predição futura. Neste último, é utilizado um conjunto de dados com 473 atributos e cerca de 120 amostras. Uma grande restrição do método de Gradiente, na otimização hiperparamétrica, é o tamanho da memória computacional que se faz necessário, presente nos complexos problemas de aprendizado de máquina atual. Em [\(Maclaurin et al., 2015\)](#) é proposto o cálculo do gradiente dos hiperparâmetros (contínuos), através do Gradiente Descendente Estocástico (SGD) reverso com *momentum* e da validação cruzada. Esta metodologia, por si só, pode vir a consumir uma quantidade inefável de memória computacional, dependendo da configuração do modelo treinado. Os autores apresentaram uma forma de lidar com essa restrição, utilizando uma regra para determinar um número ideal de bits para historiar a informação perdida no decorrer do cálculo, devido ao termo de decaimento de *momentum*. Em [\(Luketina et al., 2015\)](#), os autores mostram uma proposta do uso do SGD reverso para ajustar localmente hiperparâmetros durante o treinamento, tratando-os como parâmetros usuais. Nesta abordagem, enquanto que os verdadeiros parâmetros usuais são estimados com o conjunto de treinamento e uma função de custo com regularização, os hiperparâmetros são computados com o conjunto de validação e uma função de custo não regularizada. Este método não possui garantia de convergência.

3.3 Desafios da Busca Hiperparamétrica

No treinamento de um algoritmo de aprendizado, os hiperparâmetros são variáveis que, de alguma forma, governam o espaço do modelo ou o procedimento de ajuste do mesmo, visando assim a redução do seu erro de generalização. Se por um lado, esta característica peculiar dos hiperparâmetros possibilita a obtenção de modelos com melhores desempenho de predição, por outro, paga-se um preço pelo esforço inerente à estimação dos valores ótimos. A estimação hiperparamétrica ótima possui desafios que estão ligados ao tipo de algoritmo de aprendizado utilizado, à função de custo empregada, aos conjuntos de dados de treinamento e teste, dentre outros. Na maior parte dos casos, a otimização hiperparamétrica é tipicamente abordada como um problema de derivada inexistente, mono-objetivo e de domínio restrito ([Claesen e De Moor, 2015](#)).

⁴Implementado com a biblioteca *Spearmint*.

3.3 Desafios da Busca Hiperparamétrica

Uma questão fundamental da busca hiper-paramétrica é o custo de se avaliar a função objetivo. Cada avaliação requer o cálculo do desempenho do modelo treinado com um(s) dado(s) valor(es) para o(s) hiperparâmetro(s). Dependendo dos recursos computacionais disponíveis, da natureza do algoritmo de aprendizagem e do tamanho dos conjuntos de dados, cada avaliação pode levar um tempo que varia desde minutos a vários dias (Krizhevsky et al., 2012). A otimização hiperparamétrica dos modelos de redes neurais com arquitetura profunda é um notável exemplo desta situação, frequentemente necessitando de um vasto conjunto de dados de treinamento.

Um outro ponto é o fato de que hiperparâmetros podem ter uma influência óbvia quanto ao tempo de treinamento, como no caso da arquitetura de uma rede neural (Bishop, 2006). Já para outros, a influência pode ser sutil, porém de grande alteração no desempenho do modelo, dados os casos do uso da regularização e de *kernel* (Murphy, 2012).

Um outro fator importante na busca hiperparamétrica é a frequente existência de um componente estocástico na função objetivo (Claesen e De Moor, 2015), induzido por fatores ligados ao próprio modelo, como por exemplo: valores iniciais dos pesos de uma rede neural, reamostragem de dados empregados no treinamento (como ocorre na construção de uma floresta aleatória), dentre outros. Esse comportamento estocástico, implica então que o conjunto de hiperparâmetros ótimos encontrados empiricamente após algumas avaliações, possa não ser o verdadeiro valor ótimo.

Uma outra questão está relacionada a quantidade de hiperparâmetros presentes no modelo. Esse número de hiperparâmetros é usualmente pequeno (menor que 5), mas este valor pode chegar a casa de centenas para algoritmos de aprendizado complexos, como as redes neurais convolucionais com muitas camadas (Bergstra et al., 2013), ou quando etapas de pré-processamento estão envolvidas na otimização. Contudo, em alguns casos, apenas poucos hiperparâmetros do algoritmo de aprendizado impactam significativamente no desempenho do modelo (Bergstra e Bengio, 2012), apesar da dificuldade de se identificar previamente quais são estes.

Algumas vezes, a existência de certos hiperparâmetros estão condicionados aos valores de outros, como no caso da estrutura de uma rede neural, a qual possui o hiperparâmetro que governa o número de camadas, e a posterior definição do número de neurônios em cada camada, sendo dada por hiperparâmetros distintos. Um outro aspecto é a existência simultânea, em um mesmo modelo, como no caso de uma rede neural de múltiplas camadas (MLP) com hiperparâmetros contínuos (ex.: taxa de aprendizado e constante de regularização), discretos (ex.: número de neurônios e de camadas de uma rede neural) e categóricos (ex.: tipo de função de ativação). A Figura 3.1 ilustra este cenário.

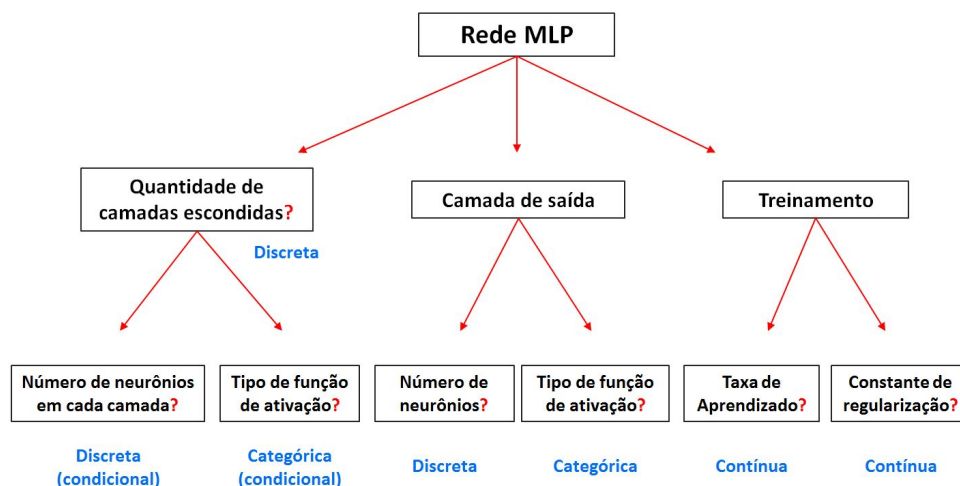


Figura 3.1: Esquema com o espaço de configuração hiperparamétrico de uma rede MLP.

Um ponto também pertinente, é que dentre os diversos modelos de aprendizado, não há nenhum específico que exiba sempre o melhor desempenho na resolução de todos os problemas existentes, sendo estes lineares ou não lineares. O que na verdade ocorre, é que cada tipo de modelo possui uma característica que se sobressai na tratativa de determinadas questões, devendo a escolha do algoritmo ser feita de acordo com a estrutura presente nos dados que serão preditos. Ou seja, a própria escolha de um modelo, para um determinado problema, pode ser vista como uma escolha parametrizada por um hiperparâmetro (Thornton et al., 2013; Brazdil et al., 2008).

3.4 Métodos de Busca Hiperparamétrica

3.4.1 Otimização via Busca em Grade

A estratégia da Busca em Grade é um método tradicionalmente utilizado na otimização hiperparamétrica de modelos com um número reduzido de hiperparâmetros. Esta técnica apresenta uma forma de resolução simples e direta, possuindo como característica: fácil implementação e paralelização.

No método de busca em grade, cada hiperparâmetro é delimitado em torno de um intervalo particular de busca, no qual acredita-se que seja um potencial local para a varredura. Para cada hiperparâmetro, é estabelecido uma resolução de grade que determina a quantidade de pontos candidatos à serem considerados para cada um destes (aqui também, levando em conta uma quantidade que se julga, a princípio, ser a mais adequada). A execução deste procedimento gera no espaço hiperparamétrico, uma estrutura em grade, formada pelo conjunto de \mathbb{T} vetores que compõem todos os arranjos candidatos, $\lambda_{1:\mathbb{T}}$, sendo estes avaliados nos treinamentos do modelo de aprendizado, por meio uma função de perda (\mathcal{L}), resultando na avaliação f . A

finalidade de todo este procedimento é achar o melhor ponto de configuração (λ^*), entre todas as configurações ($\lambda_{1:T}$) experimentadas.

A Figura 3.2 mostra exemplos de grade para o espaço hiperparamétrico.

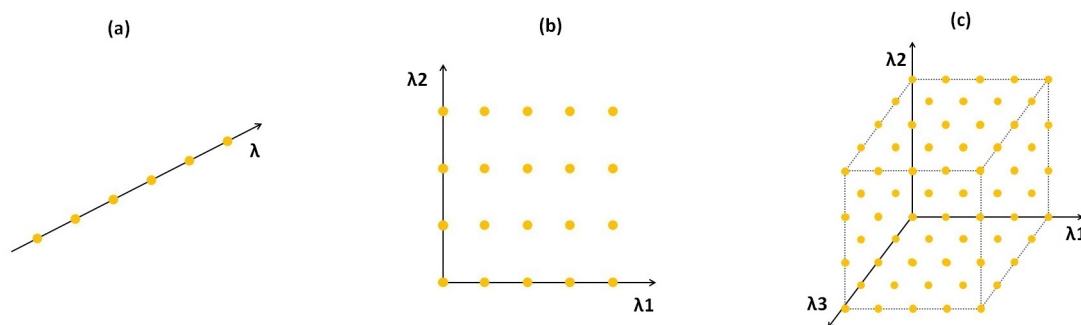


Figura 3.2: Exemplos de grades hiperparamétricas para (a) uma, (b) duas e (c) três dimensões.

Uma vez que se tenha delimitado um intervalo do espaço hiperparamétrico, e este captura a região onde habita o valor ótimo de configuração, e que, além disso, tenha-se escolhido uma resolução fina o suficiente para cobrir o subespaço de cada hiperparâmetro, o método de busca em grade possui grande chance de atingir um valor ótimo, ou muito próximo do ótimo.

O problema da utilização do método de busca em grade reside no crescimento exponencial de treinamentos necessários, à medida que o número de hiperparâmetros a serem estimados aumenta. Por exemplo: para o caso da otimização hiperparamétrica de um modelo com 3 hiperparâmetros, cada um com 4 possíveis valores candidatos, requer um total de 64 treinamentos completos, o que dependendo da situação possa ser até aceitável. Entretanto, para um outro modelo com 6 hiperparâmetros, e o mesmo valor de 4 candidatos para cada, resulta no montante de 4.096 treinamentos. O que é sem dúvida, demasiadamente oneroso, principalmente para casos como os modelos com arquitetura profunda, que exibem um número considerável de hiperparâmetros, e que, além disto, geralmente fazem uso de um vasto conjunto de dados no treinamento. Resultando assim em um tempo relativamente longo de resolução.

3.4.2 Otimização via Busca Aleatória

3.4.2.1 Introdução

Os métodos de busca aleatória são poderosas técnicas de otimização. A valiosa habilidade que essas técnicas possuem em encontrar o extremo global de uma função é indispensável para muitas áreas da ciência e da engenharia. Os métodos de busca aleatória incluem a Busca Aleatória Pura, a Busca Aleatória Adaptativa, *Simulated Annealing*, dentre outros. As técnicas de busca aleatória foram introduzidas por Anderson (1953), Rastrigin (1963) e Karnopp (1963). Desde então, diversos algoritmos modificados surgiram a partir desses.

Os algoritmos de busca aleatória podem ser utilizados na resolução de problemas em que

a diferenciabilidade da função não é assumida, sendo particularmente competitivos para casos em que: a função objetivo é custosa de se avaliar, a quantidade de memória computacional disponível é limitada, a função a ser minimizada possui múltiplos pontos de mínimo local e para situações em que a função possua superfície irregular (Baba, 1981; Solis e Wets, 1981).

3.4.2.2 Busca Aleatória Pura

Diferente da maior parte dos métodos de busca aleatória que utilizam um conjunto de pontos históricos e suas avaliações para influenciar, de alguma forma, a escolha do próximo ponto para avaliação, a Busca Aleatória Pura ignora os pontos já avaliados, de modo que cada novo ponto candidato que é sugerido, não sofre qualquer influência de pontos que foram anteriormente indicados no processo de resolução. Assim, o método de busca aleatória pura pode ser definido como a amostragem de valores aleatórios, x , repetidas vezes, a partir de um conjunto definido ou de um subespaço \mathbb{R}^x ou de uma função de densidades ζ .

No problema de otimização hiperparamétrica, o método de busca aleatória amostra valores candidatos para cada hiperparâmetro (λ_i) do algoritmo de aprendizado, por meio da definição *a priori* de uma função de densidades (ζ), para hiperparâmetro do tipo contínuo/discreto, ou de um conjunto de probabilidades, para hiperparâmetros do tipo categórico. O método consiste então em uma amostragem seguida pelo treinamento e avaliação do modelo utilizando o vetor amostrado (λ_i). Estas duas tarefas acontecem em ciclo, até que uma condição de parada ocorra (número máximo de treinamentos ou tempo de processamento).

Em Bergstra e Bengio (2012), os autores observaram que para certas classes de problemas de aprendizado de máquina, a maior parte das dimensões hiperparamétricas, envolvidas nos modelos propostos, podem não alterar significativamente a função objetivo avaliada. Para aproveitar esta característica, os autores propuseram o uso de busca aleatória pura como método de otimização hiperparamétrica. O racional por de trás deste método é que pontos uniformemente amostrados de modo aleatório, em cada dimensão, podem cobrir densamente cada subespaço de baixa dimensão. Como resultado, a procura aleatória pode explorar efetivamente uma baixa dimensionalidade sem o conhecimento de quais dimensões são importantes, aumentando assim as chances de encontrar uma configuração de melhor desempenho. A Figura 3.3 mostra essa ideia por meio de uma comparação entre busca em grade e busca aleatória. Nos gráficos mostrados, pode-se ver como os pontos com as configurações de cada hiperparâmetro estão distribuídos no espaço de busca, assim como a curva da função de custo para cada subespaço de baixa dimensão.

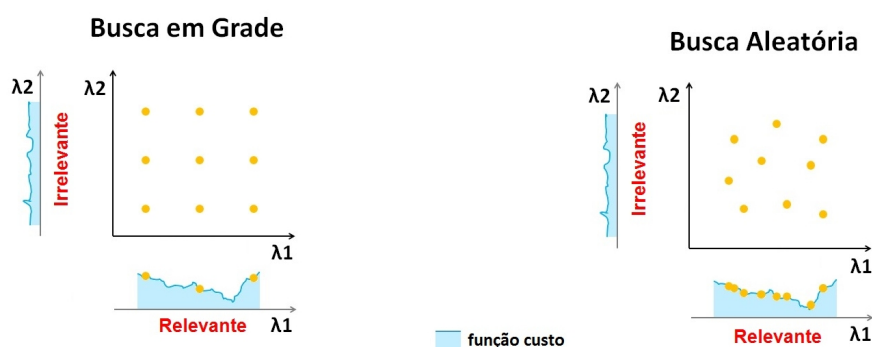


Figura 3.3: Otimização hiperparamétrica com 9 pontos de configuração, utilizando Busca em Grade e Busca Aleatória. Imagem inspirada em (Bergstra e Bengio, 2012)

3.4.3 Otimização Bayesiana

3.4.3.1 Otimização Baseada em Modelo Sequencial

A Otimização Baseada em Modelo Sequencial é uma abordagem que constrói um modelo de regressão, conhecido por Modelo de Superfície de Resposta (M_p), gerado a partir de uma sequência de experimentos com o objetivo de mapear, através de uma aproximação, a relação entre as variáveis preditoras (\mathbf{x}) e uma ou mais variáveis de saída (\mathbf{y}) de uma dada função desconhecida. A obtenção desta aproximação viabiliza a busca por um ponto ótimo em M_p . A iteração do método SBMO acontece em ciclos, alternando entre a obtenção de novos pares (\mathbf{x}, \mathbf{y}) , e a processo de otimização do modelo. Tipicamente, algoritmos SBMO se diferenciam quanto ao critério de concepção do modelo M_p e ao método de otimização (Opt) empregado na busca pelo próximo ponto candidato.

No problema de otimização hiperparamétrica, o método SBMO busca o λ^* da equação 1.1, que a princípio, não pode ser obtido devido ao desconhecimento da função $f : \lambda \mapsto \mathbf{f}$. Para isto, a técnica SBMO constrói um modelo que representa o desempenho de predição de um certo algoritmo de aprendizado A , por meio da informação contida na relação entre cada λ de A e seu respectivo resultado de avaliação \mathbf{f} ($\mathbf{D} = (\lambda, \mathbf{f})$). Este modelo de superfície de resposta é refinado à medida em que A passa pelo processo de aprendizado, fornecendo então novos pares (λ, \mathbf{f}) . Após cada atualização do modelo M_p , algum método de otimização realiza uma avaliação na superfície aproximada, em busca do melhor λ^* a ser utilizado como próximo λ no treinamento, retroalimentando então o algoritmo, até que uma condição de parada seja atingida.

Um detalhe importante do algoritmo é a coleta inicial de pares (λ, \mathbf{f}) , realizada por meio de treinamentos com valores de λ amostrados a partir de ζ , como ocorre no método de busca aleatória. Este procedimento inicial objetiva obter um mínimo (relativo) de informação para

construir a superfície de resposta antes de se iniciar a busca por um ponto ótimo. Assim, o valor de treinamentos iniciais (T') é um hiperparâmetro deste algoritmo.

O método SBMO se difere da técnica de busca aleatória por empregar algum mecanismo de otimização na seleção do próximo vetor candidato λ_{t+1} , que se acredita possuir um maior potencial a ponto ótimo ou de uma informação que melhor elucidará o espaço do modelo de superfície estimado.

3.4.3.2 Otimização Bayesiana: Introdução

A Otimização Bayesiana é uma estratégia para encontrar o extremo de uma função que possa não ter uma expressão de forma fechada, mas que seja possível obter observações (entradas/saídas) na forma de amostras. Este método é particularmente útil quando o custo de avaliação da função é alto, quando não se tem acesso à suas derivadas e/ou quando o problema não é convexo (Brochu et al., 2010). A otimização Bayesiana é uma abordagem baseada em modelo sequencial (SBMO), que trata de problemas de maximização (ou minimização) global de uma função objetivo, dado por,

$$x^* = \arg \max f(x), \quad \text{para } x \in \mathcal{X}, \quad (3.1)$$

onde \mathcal{X} é algum espaço de interesse, geralmente um subespaço de \mathbb{R}^d .

Para tratar da otimização Bayesiana é necessário introduzir alguns conceitos de Estatística Bayesiana. Ao contrário da estatística clássica (frequentista), que associa probabilidades somente à variáveis aleatórias, a estatística Bayesiana relaciona probabilidade a qualquer grau de crença ou incerteza em relação a um evento, hipótese ou valor aleatório. Esta abordagem permite a determinação de probabilidades *a priori*, que resulta de informações passadas de uma entidade, sem o conhecimento de qualquer outro evento, e probabilidades *a posteriori*, que é a probabilidade condicionada a algum outro evento sabido. Uma regra muito importante neste contexto é o teorema de Bayes, dado por,

$$p(A|B) = \frac{p(B|A)}{p(B)} p(A), \quad (3.2)$$

$$\text{posteriori} = \frac{\text{verossimilhança}}{\text{evidência}} \text{priori},$$

em que A e B são eventos, sendo $p(B) \neq 0$.

A regra de Bayes descreve uma distribuição *a posteriori* $p(A|B)$, que é a probabilidade condicional de se observar o evento A dado que B seja verdadeiro, a partir do conhecimento da evidência $p(B)$, da probabilidade *a priori* $p(A)$ e do modelo de verossimilhança $p(B|A)$.

Na resolução do problema de otimização hiperparamétrica pelo método Bayesiano, o par (λ, f) é representado como $(\mathbf{x}, f(\mathbf{x}))$. Assim, a função de probabilidade *a posteriori*, que busca

aproximar a função desconhecida $f : \lambda \mapsto \mathbf{f}$, é expressa como,

$$p(f|\mathbf{D}) \propto p(\mathbf{D}|f)p(f), \quad (3.3)$$

onde $\mathbf{D} = (\lambda, \mathbf{f})$, $p(f)$ representa a crença sobre o espaço de possibilidades para a função desconhecida, $p(\mathbf{D}|f)$ é a função de verossimilhança e $p(f|\mathbf{D})$ é a distribuição *a posteriori*.

Para lidar com a equação 3.3 a vertente da otimização Bayesiana, que utiliza o processo gaussiano, modela diretamente a distribuição *a posteriori* $p(f|\mathbf{D})$. Esta é vista a seguir.

3.4.3.3 Processo Gaussiano

O Processo Gaussiano é uma generalização da distribuição de probabilidade Gaussiana multivariada para o caso de variáveis aleatórias de um processo estocástico⁵, que descrevem o comportamento de um sistema (MacKay, 1998). A técnica GP possui uma abordagem prática e probabilística para construir modelos de regressão e classificação, utilizando para isto uma função média e uma função de covariância, para o caso de regressão, e uma função logística, para modelos de classificação (binária) (Rasmussen e Williams, 2006).

O processo gaussiano, para o caso da regressão, é descrito pela seguinte equação,

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')), \quad (3.4)$$

em que $m : \mathcal{X} \mapsto \mathbb{R}$ é uma função média, e $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ é uma função de covariância, também conhecida como *kernel*. Um exemplo de definição para estas duas funções é,

$$m(x) = m, \quad (3.5)$$

$$k_{SE}(x, x') = \sigma_0^2 \exp\left(-\frac{r^2}{2}\right), \quad \text{Exponencial Quadrático} \quad (3.6)$$

onde, $r^2 = (x - x')^T \Psi (x - x')$, e Ψ é uma matriz diagonal com os hiperparâmetros l_i para cada x_i , m e σ_0^2 são também hiperparâmetros do modelo do Processo Gaussiano.

A função média (m) pode ser definida como igual a uma constante (geralmente 0) (Brochu et al., 2010), apesar da existência de métodos para a sua estimação à partir dos dados, como a Estimação Pontual (Shahriari et al., 2016). O hiperparâmetro l está relacionado ao quanto é necessário se mover em um determinado eixo no espaço de entrada para que dois valores da função se tornem não correlacionados. E o hiperparâmetro σ_0^2 pode controlar a quantidade de variância global da função.

A função de covariância codifica o que se assume sobre o grau de suavidade da superfície da função desconhecida (Rasmussen e Williams, 2006). Ou seja, se esta varia rapidamente sua resposta para uma pequena mudança em x (possuindo assim escala de comprimento característica muito curta) ou não. Para isto, o *kernel* especifica a similaridade entre dois pontos (x, x') ,

⁵Processo Estocástico é uma ou mais variáveis aleatórias representando a evolução de um sistema ao longo do tempo

3.4 Métodos de Busca Hiperparamétrica

por meio de uma medida que retorna valores em um intervalo entre 1, para pontos vizinhos, e 0 quando distantes.

O *kernel* exponencial quadrático (equação 3.6) é uma função infinitamente diferenciável, o que se traduz em uma forma suave para o modelo gerado. Outro tipo de função de covariância que é capaz de assumir um perfil menos suave que o SE, são os *kernels* de Matérn (Shahriari et al., 2016).

A família dos *kernels* de Matérn é definida pela equação,

$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}r}{l} \right), \quad (3.7)$$

em que $\Gamma(\nu) = (\nu - 1)!$ é a função Gama, ν e l são hiperparâmetros positivos, e K_ν é uma função de Bessel modificada. Para o caso em que $\nu \rightarrow \infty$, a equação 3.7 resulta no *kernel* Exponencial Quadrático. A Figura 3.4 mostra alguns dos *kernels* matérn e o *kernel* SE.

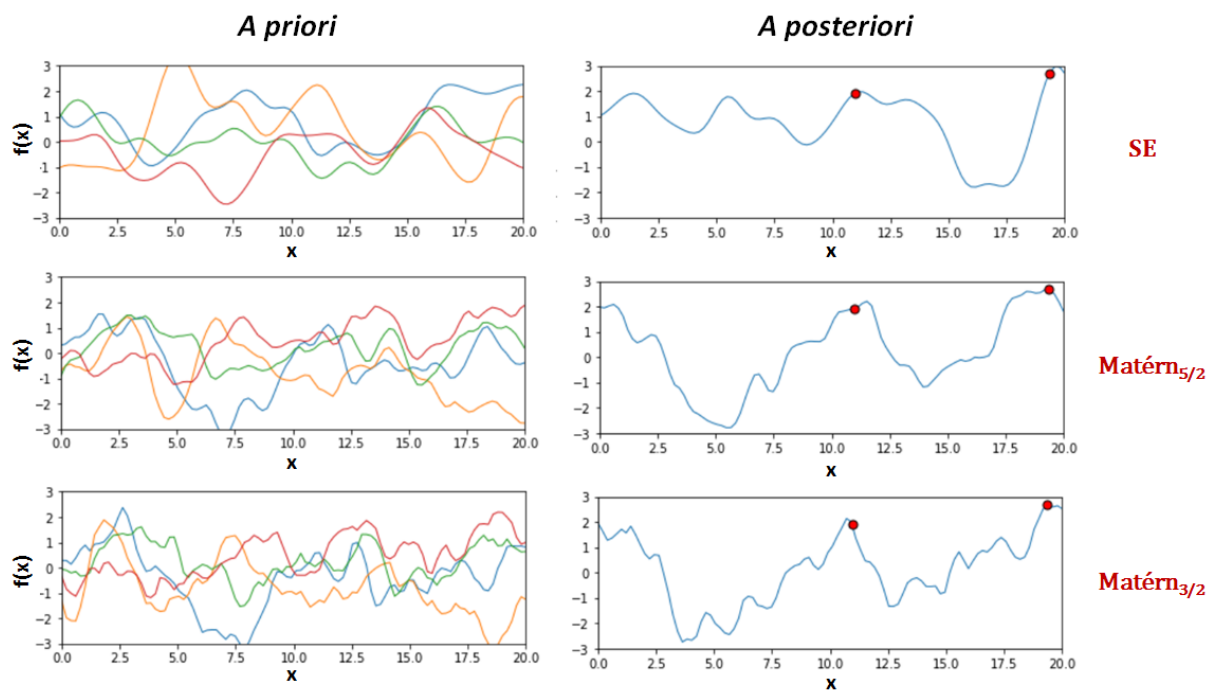


Figura 3.4: Exemplos de *kernels* de família matérn, para $\nu = 3/2$ e $\nu = 5/2$, e o caso particular que resulta na Exponencial Quadrático (SE) quando o $\nu \rightarrow \infty$.

Segundo Rasmussen e Williams (2006), dois casos particulares da família matérn que se

3.4 Métodos de Busca Hiperparamétrica

destacam quanto à sua importância para o aprendizado de máquina, são:

$$k_{v=3/2}(r) = \sigma_0^2 \left(1 + \frac{\sqrt{3}r}{l} \right) \exp\left(-\frac{\sqrt{3}r}{l}\right), \quad (3.8)$$

$$k_{v=5/2}(r) = \sigma_0^2 \left(1 + \sqrt{5}r + \frac{5r^2}{3} \right) \exp(-\sqrt{5}r). \quad (3.9)$$

onde, $r^2 = (x - x')^T \Psi (x - x')$, e Ψ é uma matriz diagonal com os hiperparâmetros l_i para cada x_i , e σ_0^2 também é um hiperparâmetro.

Existe uma variedade de tipos de funções de covariância, como por exemplo: exponencial quadrática, matérn, quadrática racional, linear, polinomial, exponencial, e até mesmo a possibilidade de se criar novas funções a partir das já existentes, por meio de operações de adição, multiplicação, dentre outras. A escolha adequada do *kernel* é crucial para a correta captura da função desconhecida pelo modelo gerado. Além disto, funções de covariância possuem em sua definição hiperparâmetro(s) responsáveis pela generalização das mesmas, cujos valores ótimos são de grande importância para um modelamento efetivo. A Figura 3.5 mostra variados graus de suavidade do modelo *a posteriori*, em função de diferentes *kernels* e valores para seus hiperparâmetros.

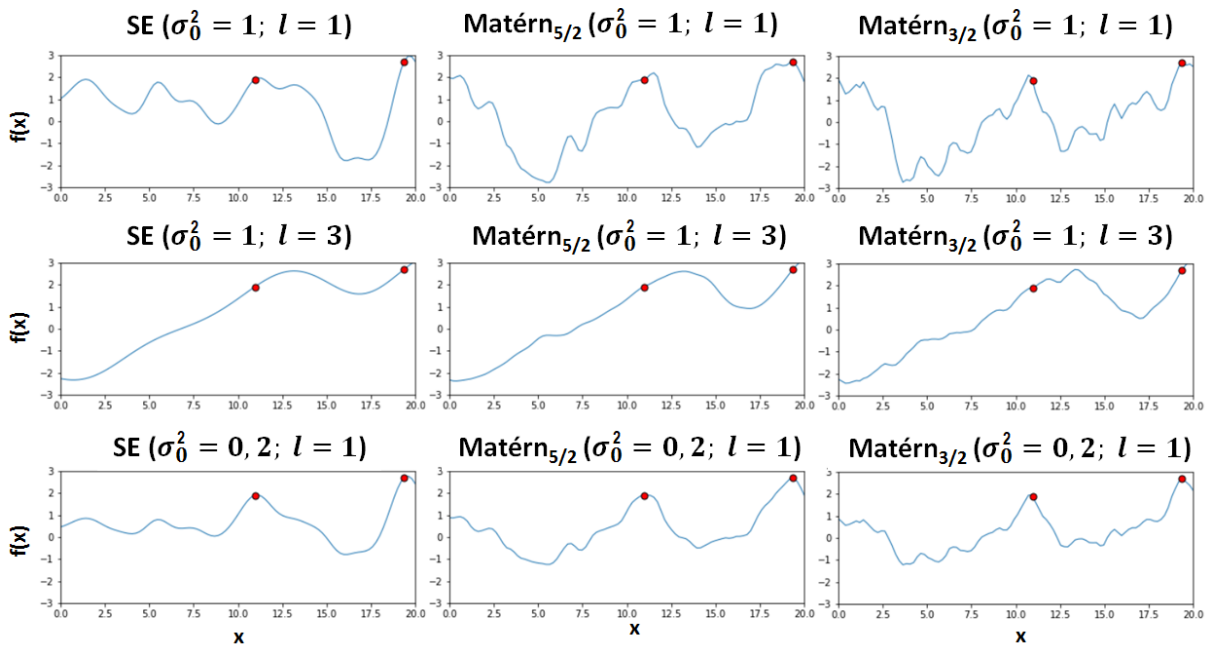


Figura 3.5: Gráficos mostram comportamentos variados para o modelo *a posteriori*, em função do tipo de *kernel* (matérn $v = 3/2$ e $v = 5/2$ e exponencial quadrático) e diferentes valores para seus hiperparâmetros.

A estimação do valor ótimo para os hiperparâmetros de uma função de covariância pode

3.4 Métodos de Busca Hiperparamétrica

ser feita por meio de estimação pontual⁶ ou por marginalização aproximada⁷ (Shahriari et al., 2016). A estimativa dos hiperparâmetros no Processo Gaussiano, quando utilizado poucas avaliações da função, pode resultar em baixo desempenho devido aos métodos citados anteriormente estarem sujeitos a ficarem presos em "armadilhas" (*traps*) durante o processo de busca, principalmente quando se utiliza as técnicas de Estimação de Máxima Verossimilhança (MLE) e Estimação de Máxima a Posteriori (MAP) (Shahriari et al., 2016).

Após a definição do modelo *a priori* do processo gaussiano, e à medida que os treinamentos do modelo de aprendizado são realizados, produzindo assim os pares $(\mathbf{x}, f(\mathbf{x}))$, que representam os hiperparâmetros e as avaliações, o modelo *a priori* é combinado com estes pares de dados para obter a função de probabilidade *a posteriori*, a qual possibilita encontrar um próximo ponto candidato (\mathbf{x}_{t+1}) . Escrevendo $f(\mathbf{x}_{1:T})$ como $\mathbf{f}_{1:T}$, e representando o valor da função no próximo ponto como f_{t+1} , então, pelas propriedades do Processo Gaussiano, tem-se que $\mathbf{f}_{1:T}$ e f_{t+1} são distribuições Gaussianas conjuntas, dadas por,

$$\begin{bmatrix} \mathbf{f}_{1:T} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right), \quad (3.10)$$

onde,

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \dots & k(x_t, x_t) \end{bmatrix},$$

é a matriz de covariância, em que cada elemento ij representa a covariância entre os elementos i e j de um vetor \mathbf{x} .

$$\mathbf{k} = [k(x_{t+1}, x_1) \quad k(x_{t+1}, x_2) \quad \dots \quad k(x_{t+1}, x_t)],$$

e \mathbf{k} é um vetor com a covariância entre x_{t+1} e cada elemento do vetor \mathbf{x} .

Então, a distribuição a posteriori $p(f|\mathbf{D})$ é dada pela seguinte distribuição Normal,

$$p(f_{t+1}|\mathbf{D}_{1:t}, x_{t+1}) = \mathcal{N}(\mu_t(x_{t+1}), \sigma_t^2(x_{t+1})), \quad (3.11)$$

onde,

$$\mu_t(x_{t+1}) = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_{1:T}, \quad \text{média a posteriori}, \quad (3.12)$$

$$\sigma_t^2(x_{t+1}) = k(x_{t+1}, x_{t+1}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}, \quad \text{incerteza a posteriori}. \quad (3.13)$$

Um exemplo de distribuição a posteriori, para uma função de dimensão 1, pode ser visto na Figura 3.6.

⁶Estimadores de Máxima Verossimilhança (ML) e Máxima a Posteriori (MAP) são métodos de estimação pontual utilizados na determinação dos hiperparâmetros *kernels*

⁷Métodos como Monte Carlo via Cadeia de Markov (MCMC) e Monte Carlo Sequencial (SMC)

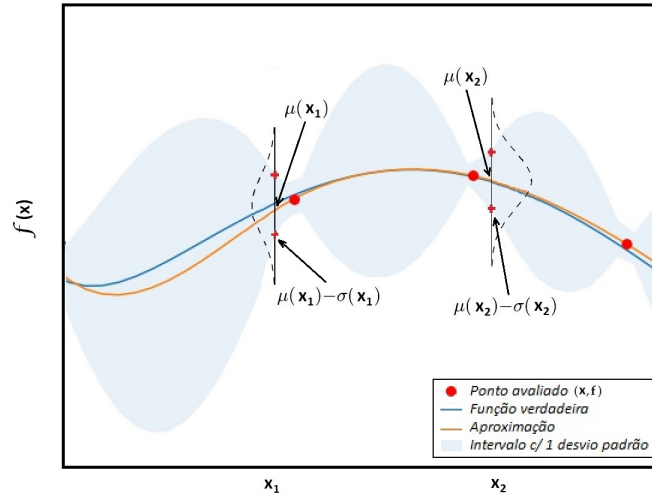


Figura 3.6: Exemplo de distribuição *a posteriori* de um processo Gaussiano de dimensão 1, com 3 pares (x, f) . A linha azul é a função desconhecida, a linha laranja mostra a média *a posteriori* (aproximação), e a área sombreada a variância.

3.4.3.4 Função de Aquisição

Todo o fundamento, apresentado anteriormente, permite a obtenção da distribuição *a posteriori*. Porém, é preciso ainda determinar qual será o próximo ponto x_{t+1} . Isto é feito por meio da otimização de uma Função de Aquisição, que é utilizada para avaliar a perda esperada associada a um ponto específico em $f(x)$. A função de aquisição guia a busca pelo ponto ótimo, balanceando o Aproveitamento da escolha, que visa valores de melhores desempenhos, e a Exploração, que alveja locais com maior incerteza (variância) da distribuição *a posteriori* (Brochu et al., 2010). A definição do tipo de função de aquisição é feita previamente à busca pelo ponto ótimo, sendo que as escolhas tradicionalmente preferidas na abordagem da otimização Bayesiana são: Probabilidade de Melhoria (PI), Esperança de Melhoria (EI) e Limite de Confiança Superior (UCB). Estas duas últimas mostraram ser eficientes em se tratando do número de avaliações necessárias para a otimização global de funções caixa preta (Srinivas et al., 2009). Estas três são descritas brevemente a seguir.

A Probabilidade de Melhoria utiliza o ponto atual de melhor desempenho x^+ para buscar o próximo ponto x_{t+1} . A função PI é dada pela seguinte equação,

$$\begin{aligned}
 PI(x_{t+1}) &= p(f(x_{t+1}) \geq f(x^+) + \xi), \\
 &= \Phi\left(\frac{\mu_t(x_{t+1}) - f(x^+) - \xi}{\sigma_t(x_{t+1})}\right),
 \end{aligned} \tag{3.14}$$

onde Φ é uma função de distribuição acumulada normal, μ_t e σ_t são os termos das equações 3.12 e 3.13, respectivamente, x^+ representa o atual ponto (x, f) com o melhor desempenho para f ,

3.4 Métodos de Busca Hiperparamétrica

e $\xi \geq 0$ é um hiperparâmetro que controla o compromisso entre *Aproveitamento* e *Exploração* no processo de busca. Para $\xi = 0$, a PI exibe puramente um perfil de Aproveitamento, amostrando pontos com alta probabilidade de serem infinitesimalmente maiores que x^+ , restringindo assim a busca localmente. Para valores de ξ muito grande, áreas com grande incerteza são privilegiadas na busca, delongando por outro lado, a conversão para pontos promissores.

Diferente da função de aquisição PI, que leva em consideração somente a probabilidade de melhoria, a Esperança de Melhoria avalia também a magnitude de melhoria para um dado ponto. A EI é dada pelas seguintes equações,

$$EI(x_{t+1}) = \begin{cases} (\mu_t(x_{t+1}) - f(x^+) - \xi)\Phi(Z) + \sigma_t(x_{t+1})\phi(Z), & \text{se } \sigma_t(x_{t+1}) > 0 \\ 0, & \text{se } \sigma_t(x_{t+1}) = 0 \end{cases} \quad (3.15)$$

$$Z = \frac{\mu_t(x_{t+1}) - f(x^+) - \xi}{\sigma_t(x_{t+1})} \quad (3.16)$$

onde ϕ é uma função de distribuição de probabilidade normal, e Φ uma função de distribuição acumulada normal. O termo ξ possui a mesma função apresentada na equação da PI (3.14).

O Limite de Confiança Superior é definido como,

$$UCB(x_{t+1}) = \mu_t(x_{t+1}) + \kappa_t \sigma_t(x_{t+1}) \quad (3.17)$$

em que $\kappa_t > 0$ é um hiperparâmetro que controla o percentual de captura de x .

Uma vez escolhida a função de aquisição, é necessário ainda a definição de um método de otimização para maximizá-la. Funções de aquisição são frequentemente multimodais, sendo sua maximização uma tarefa não trivial (Shahriari et al., 2016). Vários métodos têm sido aplicados para esta finalidade, como por exemplo: Estratégia Evolucionária de Adaptação da Matriz de Covariância (CMA-ES), Busca em Grade, Amostragem por Hipercubo Latino (*Latin Hypercube Search*), método de Busca Local *Multi-start*, método DIRECT (*Dividing RECTangles*), abordagem *hill-climbing* para métodos Quasi-Newton, como o algoritmo L-BFGS-B, dentre outros.

A Figura 3.7, mostra a influência das funções de aquisição PI, UCB e EI, com seus hiperparâmetros favorecendo o aproveitamento (PI com $\xi = 0$, UCB com $\kappa = 1$, e EI com $\xi = 0$) ou a exploração (PI com $\xi = 10$, UCB com $\kappa = 10$, e EI com $\xi = 1$), em um exemplo em que o algoritmo L-BFGS-B é utilizado na otimização Bayesiana para buscar o ponto ótimo (maximização) da função $f(x) = (e^{-(x-2)^2} + e^{-(x-6)^2/10} + 1/(x^2 + 0.5))/2$ no intervalo $0 < x < 100$, por meio da avaliação de 13 pontos e com o uso do *kernel* matérn ($\nu = 5/2$).

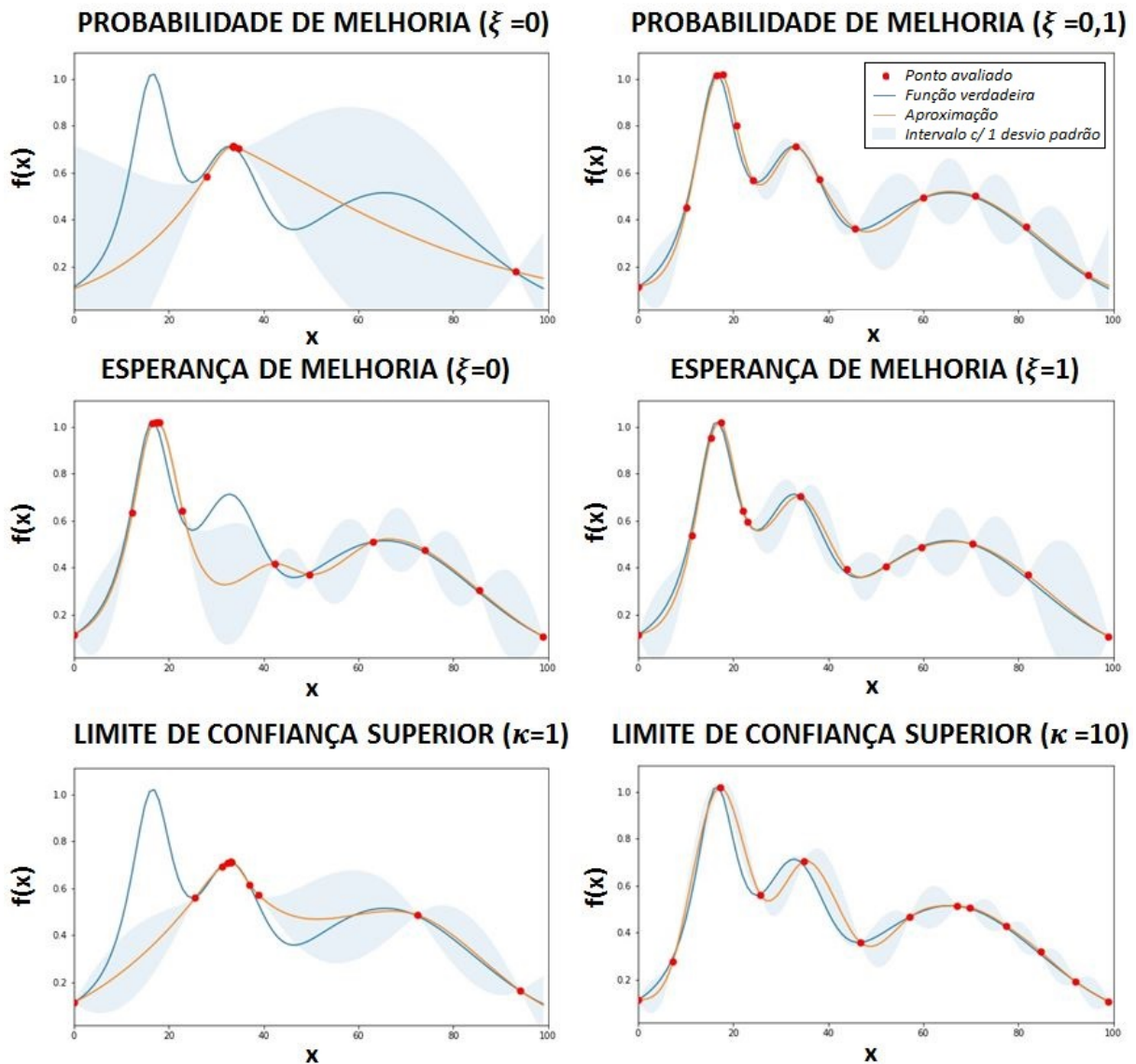


Figura 3.7: Influência das funções de aquisição PI, UCB e EI, para diferentes valores de seus hiperparâmetros. Em azul, a função verdadeira $f(x) = (e^{-(x-2)^2} + e^{-(x-6)^{2/10}} + 1/(x^2 + 0.5))/2$. Este exemplo utiliza o algoritmo L-BFGS-B, como método de otimização (com 13 avaliações), e o *kernel* matérn ($\nu = 5/2$).

3.4.3.5 Estimação de Máxima Verossimilhança

Considere um vetor com variáveis aleatória $\mathbf{X} = (X_1, \dots, X_d)$, independentes e identicamente distribuídas (i.i.d.), em que cada uma possua distribuição de densidade (probabilidade) $p(x_i|\theta_i)$, tal que $\theta \in \Upsilon \subseteq \mathbb{R}^d$. Se $\mathbf{x} = (x_1, \dots, x_d)$ são os valores observados para o vetor \mathbf{X} , então, a função

3.4 Métodos de Busca Hiperparamétrica

de verossimilhança de θ é expressa como,

$$L(\theta, \mathbf{x}) = \prod_{i=1}^n p(\mathbf{x}|\theta), \quad (\text{Função de Verossimilhança}), \quad (3.18)$$

onde, n é o número total de amostras de \mathbf{x} . Uma modificação que usualmente é feita na equação 3.18, é a aplicação do operador logaritmo,

$$l(\theta, \mathbf{x}) = \prod_{i=1}^n \ln p(\mathbf{x}|\theta), \quad (\text{Função log-Verossimilhança}), \quad (3.19)$$

que permite simplificações algébricas ao se derivar a equação 3.19, ao mesmo tempo que não causa alterações na maximização da função. (MacKay, 2003).

A Estimação da Máxima Verossimilhança (Fisher, 1922) é um método de busca pelo valor de θ que maximiza $L(\theta, \mathbf{x})$, expressa por,

$$\hat{\theta} = \arg \max_{\theta \in \Theta} L(\theta, \mathbf{x}), \quad (3.20)$$

em que $\hat{\theta}$ é o ponto de Υ que maximiza a densidade conjunta avaliada com os valores de \mathbf{x} .

A MLE para o vetor $\theta = (\theta_1, \dots, \theta_g)$ é feita pela diferenciação da equação 3.19, fazendo a derivada de primeira ordem igual a zero, para obter as g Equações de Verossimilhança,

$$\frac{\partial}{\partial \theta_i} l(\theta, \mathbf{x}) = 0, \quad \text{para } i = 1, \dots, g, \quad (3.21)$$

que compõem um sistema de equações a ser resolvido para obter $\hat{\theta}$.

Para verificar se a solução encontrada por meio da equação 3.21 corresponde realmente a um ponto de máximo, e não a um ponto de mínimo ou de sela, é preciso investigar se a matriz de derivadas segundas (matriz Hessiana), expressa por,

$$\mathbf{H} = \frac{\partial^2 l(\theta, \mathbf{x})}{\partial \theta \partial \theta^T} \Big|_{\theta = \hat{\theta}}, \quad (3.22)$$

é negativa definida. Ou seja, $\mathbf{z}^T \mathbf{H} \mathbf{z} < 0$, $\forall \mathbf{z} \in \mathbb{R}^g$.

A técnica MLE não possui garantia de convergência, por ser um processo heurístico que busca a melhoria a partir de um conjunto de parâmetros iniciais (θ_0) (Myung, 2003). Assim, o algoritmo está sujeito a uma variação de seu resultado em função de θ_0 . Um resultado sub-ótimo ocorre quando um ponto de máximo local é atingido, e este possui valor para a função relativamente distante do encontrado no ponto de máximo global. Para contornar este problema, o algoritmo pode ser reinicializado u vezes a partir de pontos diferentes.

3.4.3.6 Algoritmo L-BFGS-B

O método L-BFGS-B (Byrd et al., 1995) é um algoritmo da família quasi-Newton que trata de problemas de otimização não linear com restrição simples ($a \leq x \leq b$), mas que também pode ser usado em problemas irrestritos. O método assume que o cálculo da matriz Hessiana é impraticável ou muito custoso, e utiliza uma aproximação através da memória limitada BFGS (Broyden-Fletcher-Goldfarb-Shanno) que atualiza a matriz aproximada, a cada iteração, utilizando uma determinada quantidade de informação de $m_{l-bfgs-b}$ iterações passadas. Valores usualmente empregados para o hiperparâmetro $m_{l-bfgs-b}$ estão entre 3 à 20 (Zhu et al., 1994).

A matriz de memória aproximada é usada na definição de um modelo quadrático da função objetivo, para que uma busca em direção seja realizada. Primeiramente o método de gradiente projetado identifica o conjunto com os limites (das variáveis de entrada) que estão ativos, e então o modelo quadrático é aproximadamente minimizado, em relação as variáveis livres. Então, o próximo ponto de avaliação é calculado através da busca em linha, utilizando um determinado tamanho de passo $\alpha_{l-bfgs-b}$ (hiperparâmetro).

Experimentos e Resultados

4.1 Introdução

O presente capítulo descreve os experimentos realizados na comparação do desempenho das técnicas de otimização hiperparamétrica: busca em grade, busca aleatória e otimização Bayesiana utilizando o processo gaussiano. Estes métodos foram utilizados na sintonia de hiperparâmetros dos modelos de árvore de decisão e floresta aleatória. Estes dois algoritmos de aprendizado foram empregados na resolução de problemas de classificação binária, utilizando diferentes conjuntos de dados.

Os testes foram realizados com o objetivo final de responder a pergunta: algum dos três métodos de otimização hiperparamétrica proporciona um resultado de predição, para um modelo de aprendizado, estatisticamente superior?

O capítulo está dividido da seguinte forma: na próxima seção são descritos os conjuntos de dados utilizados. Em seguida, são expostos as configurações dos recursos computacionais utilizados nos experimentos, o ambiente de programação e bibliotecas utilizadas nos experimentos. A seção seguinte descreve a metodologia utilizada nos testes de comparação das técnicas de otimização hiperparamétrica. Em seguida, são expostos os resultados obtidos com o modelo de árvore de decisão e floresta aleatória. E por fim, os resultados dos testes são discutidos na última seção do Capítulo atual.

4.2 Conjunto de Dados

Os conjunto de dados utilizados nos testes comparativos foram coletados do repositório UCI-Machine Learning¹ (Asuncion e Newman, 2007). A descrição de cada um dos 14 conjuntos de dados utilizados nos experimentos é feita na Tabela 4.1, a qual especifica o número de instâncias e atributos, e a razão de balanceamento (classe minoritária : classe majoritária) para cada um destes. Originalmente, o conjunto Cardiotocography possui 3 classes. Porém, este conjunto teve a classe de menor número de amostras (com 176, representando 8,3% dos dados) retirada para adequá-lo aos experimentos de classificação binária. Apesar destes conjuntos

¹<http://archive.ics.uci.edu/ml/>

4.3 Recursos Computacionais e Ambiente de Programação

serem bastante conhecidos nos trabalhos de aprendizado de máquina, estes possuem como características um baixo volume de dados e pequena dimensão.

Tabela 4.1: Conjunto de dados utilizados nos experimentos. A tabela mostra nome, número de instâncias e atributos, e a razão de balanceamento entre as duas classes, para cada conjunto.

<i>Conjunto de Dados</i>	<i>Instâncias</i>	<i>Atributos</i>	<i>Razão Balanceamento</i>
Breast Cancer	569	30	0,37 : 0,63
Vertebral Column	310	6	0,32 : 0,68
Cardiotocography ¹	2.126 (-176)	23	0,15 : 0,85
Diabetes	1.151	20	0,47 : 0,53
German Credit	1.000	20	0,30 : 0,70
Ionosphere	351	34	0,36 : 0,64
Liver	345	6	0,42 : 0,58
Musk-1	476	168	0,43 : 0,57
Musk-2	6.598	168	0,15 : 0,85
Parkinsons	197	23	0,25 : 0,75
Pima Indians	768	8	0,35 : 0,65
Qsar	1.055	41	0,34 : 0,66
Sonar	208	60	0,47 : 0,53
Spambase	4.601	57	0,39 : 0,61

¹ Utilizou-se apenas as duas classes com maior número de instâncias.

Em todos os testes realizados, não houve o pré-processamento dos dados destes conjuntos, para extração de características ou normalização dos valores.

4.3 Recursos Computacionais e Ambiente de Programação

Para realizar os experimentos foram utilizados dois computadores iMac com configurações de *hardware* idênticas, sendo estas: processador 2.8 GHz Intel i7 (8 *cores*) e 12 GB de memória. Em cada um destes foi configurado uma máquina virtual com 4 *cores* para processamento, 6 GB de memória e sistema operacional Ubuntu 16.4.

Os experimentos foram implementados utilizando a linguagem de programação Python, com o uso das bibliotecas: Scikit-learn² para os métodos de busca em grade e aleatória, e bayes_opt³ para o método de otimização Bayesiana. Os modelos de aprendizado, árvore de decisão e floresta aleatória, foram experimentados através da biblioteca Scikit-learn.

4.4 Metodologia

Nos experimentos para a comparação das técnicas de otimização hiperparamétrica, foram adotadas as premissas de que cada método poderia realizar um mesmo número máximo de

²<http://scikit-learn.org>

³<https://github.com/fmfrn/BayesianOptimization>

4.4 Metodologia

treinamentos e que os hiperparâmetros que não estivessem sendo variados (como o número de árvores para o modelo de floresta aleatória) iriam permanecer sempre fixos com os mesmos valores (descritos mais adiante). Estas condições visam proporcionar um cenário de igualdade para a comparação das técnicas.

Os testes com o modelo de árvore de decisão envolveram o uso de todos os 14 conjuntos de dados listados na tabela 4.1. Cada conjunto foi tratado por cada um dos três métodos de otimização hiperparamétrica para sintonizar os hiperparâmetros do modelo que definem: a quantidade mínima de pontos nas folhas (N_{folha}), e se um nó será dividido com base no valor da sua pureza (β). A Tabela 4.2 mostra os limites inferior e superior de busca que foram utilizados pela busca aleatória e otimização Bayesiana, para estes dois hiperparâmetros. A construção da grade do método de busca em grade usou os valores destes limites (inferior e superior) mais os descritos pelos valores intermediários.

Tabela 4.2: Intervalo de busca para os hiperparâmetros do modelo de árvore de decisão, e respectivos valores intermediários (utilizados apenas na busca em grade).

Hiperparâmetro	Limite Inferior	Valores Intermediários*	Limite Superior
N_{folha} (%)**	0,1	10, 20, 30 e 40	50
β	0	$3 \times (10^{-7}, 10^{-5} \text{ e } 10^{-3})$	0.3

*Utilizado apenas para a busca em grade.

**A porcentagem se refere a operação:

$$N_{folha} = \lfloor n_{train} \times (N_{folha} \%) \rfloor, \text{ em que } n_{train} \text{ é o número de amostras no treinamento.}$$

Nos testes, o demais hiperparâmetros da árvore de decisão foram definidos como: número mínimo de pontos abaixo do qual um nó se torna terminal ($N_{min} = 2$) mantido fixo, profundidade máxima que um nó pode atingir (d_{max}) foi colocado como não atuante, e como critério de pureza foi definido o Índice de Gini (equação 2.19).

Testes preliminares mostraram que este arranjo para os hiperparâmetros (descrito na Tabela 4.2) proporciona uma relativa variação do desempenho do classificador, à medida que N_{folha} e β sofrem variações. Outro ponto é que a escolha da faixa de variação para N_{folha} e β proporciona a chance que conseguir bons desempenhos de predição. Estes são comportamentos pertinentes para a comparação entre os métodos de otimização hiperparamétrica.

Para se avaliar a qualidade de predição das árvores de decisão foi utilizado a métrica AUC e o uso da técnica de validação cruzada ($k_{cv} = 5$), com amostragem aleatória estratificada. A opção por $k_{cv} = 5$ se deve ao fato dos recursos computacionais utilizados nos experimentos serem limitados quanto ao poder de processamento. O experimento com cada conjunto de dados, e para cada um dos três métodos, foi repetido 30 vezes para fins de significância estatística.

Em relação aos métodos de otimização, a busca aleatória utilizou a função de probabilidade Uniforme para amostrar aleatoriamente os valores de cada hiperparâmetro, respeitando o limite informado na Tabela 4.2.

O método de otimização Bayesiana foi configurado para utilizar a função de aquisição Esperança de Melhoria (equação 3.15) com o seu hiperparâmetro definido como $\xi = 0$. A

4.4 Metodologia

função média *a priori* foi definida igual a zero. E para *kernel*, foi escolhida a função Matérn, com $\nu = 2,5$, como em [Eggensperger et al. \(2013\)](#), e para os hiperparâmetros do *kernel* foram definidos os seguintes valores iniciais. $\sigma_0^2 = 1.0$ e $l = 1.0$. A otimização Bayesiana utilizou o algoritmo L-BFGS-B com os hiperparâmetros $m_{l-bfgs-b} = 10$ e $\alpha_{l-bfgs-b} = 1.0 \times 10^{-8}$ para maximizar o modelo *a posteriori*. A cada iteração, o método MLE foi executado $u = 25$ para evitar que o processo de otimização ficasse preso em um ponto de máximo local.

A busca em grade teve sua grade construída com os valores da Tabela 4.2, totalizando $6 \times 5 = 30$ opções de configuração para os hiperparâmetros. Para que a premissa referente à igualdade do número de avaliações fosse mantida, os demais métodos foram configurados com este mesmo número de treinamentos, sendo que o método de otimização Bayesiana utilizou metade deste número (15) para avaliar inicialmente valores aleatórios. Após cada rodada com estes 30 treinamentos, separou-se o modelo com melhor avaliação pela métrica AUC.

Para embasar a comparação entre as técnicas de otimização hiperparamétrica nos experimentos com o modelo de árvore de decisão, foi realizado um teste estatístico utilizando a média dos 30 resultados da métrica AUC, para determinar se há diferença estatística significativa entre os resultados obtidos por cada uma das três técnicas.

Foi escolhido o teste de Quade ([Quade, 1979](#)), que é um teste não paramétrico e uma alternativa ao teste de variância ANOVA quando as premissas de normalidade e homocedasticidade dos dados não estão asseguradas. Segundo [Demšar \(2006\)](#) testes não paramétricos devem ser preferidos aos testes paramétricos, por serem mais seguros ao exibir maior tendência de rejeitar a hipótese nula quando não se assume premissas de distribuição dos dados. Em se tratando de Poder de Teste⁴, o teste de Quade exibe maior valor quando comparado ao teste não paramétrico de Friedman, para casos em que o número de grupos avaliados for menor que 5 ([Conover, 1980](#)), como o presente.

O teste de Quade cria um *rank*, a partir dos dados (na presente comparação, estes são as médias obtidas com as 30 repetições), entre os grupos (neste caso, entre os métodos) para cada bloco (que aqui representa um dos conjuntos). Este *rank* é feito com a designação do número 1 para o menor valor (dentro de um bloco), seguido por um 2 para o segundo menor, e assim por diante. Após a construção desta matriz, em que os elementos são representados por $R(X_{ij})$, procede-se com o cálculo do intervalo para cada bloco Q_i (AUC máxima menos AUC mínima de cada conjunto). Então, uma segunda matriz ponderada, é definida por meio de,

$$S_{ij} = Q_i \left[R(X_{ij}) - \frac{c+1}{2} \right], \quad (4.1)$$

onde c é o número de grupos, e S_{ij} é um elemento da matriz.

A hipótese nula do teste é que os grupos são equivalentes. Assim, a hipótese alternativa afirma que ao menos um dos grupos avaliados é diferente de algum dos demais. Para que a hipótese nula seja rejeitada, a estatística do teste F_q deve ser maior que o valor obtido da tabela

⁴O Poder do Teste está relacionado ao erro do tipo II, ou seja, quando a hipótese nula é falsa e a mesma não é rejeitada.

4.4 Metodologia

de distribuição F, para um determinado nível de significância α , e com os graus de liberdade $gl_1 = c - 1$ e $gl_2 = (a - 1)(c - 1)$, em que a é o número de blocos. O valor de F_q é dado por,

$$F_q = \frac{(a - 1)B_1}{(A_1 - B_1)}, \quad (4.2)$$

em que $B_1 = (\sum_{j=1}^k S_j^2)$, $A_1 = (\sum_{i=1}^a \sum_{j=1}^c S_{ij}^2)$, $S_j = (\sum_{i=1}^a S_{ij})$, para $j = 1, 2, \dots, c$.

Caso a hipótese nula seja rejeitada, prossegue-se então com um teste *post hoc* de comparações múltiplas, baseado na distribuição *t* de *student*, em que dois grupos possuem diferença significativa, somente se a diferença entre as suas médias (da matriz ponderada) for,

$$|S_j - S_i| > t_{1-\alpha/2^*, (a-1)(c-1)} \sqrt{\frac{2a(A_1 - B_1)}{(a - 1)(c - 1)}}, \quad (4.3)$$

para um determinado $t_{1-\alpha/2^*, (a-1)(c-1)}$, de acordo com o nível de significância α e graus de liberdade $gl = (a - 1)(c - 1)$.

Os experimentos comparativos entre as técnicas de busca em grade e aleatória, e da otimização Bayesiana, realizados com o modelo de floresta aleatória, seguiram os mesmos procedimentos que os testes feitos com o modelo de árvore de decisão, com exceção do teste estatístico que não foi realizado devido ao menor número de repetições dos experimentos (10). Isto aconteceu devido ao custo computacional requerido nos experimentos e ao limitado recurso computacional de que se fez uso. Porém, além desta diferença e da modificação do tipo de algoritmo de aprendizado, algumas alterações foram feitas, e são descritas a seguir.

Os hiperparâmetros do modelo de floresta aleatória que foram considerados nos experimentos envolveram: a quantidade mínima de pontos nas folhas (N_{folha}), critério de divisão de um nó com base no valor da sua pureza (β), e o número de atributos m considerados no *split* das árvores durante o crescimento da floresta aleatória. A Tabela 4.3 mostra os limites inferior e superior de busca que foram utilizados pela busca aleatória e otimização Bayesiana, para estes três hiperparâmetros. O método de busca em grade usou os valores destes dois limites e os descritos pelos valores intermediários para construção de sua grade. Em relação ao número total de treinamentos realizados por cada método, os testes com o modelo de floresta aleatória utilizaram um total de $7 \times 8 \times 7 = 392$ avaliações. Este número corresponde ao total de combinações contidas na grade da busca em grade. Assim, a busca aleatória e a otimização Bayesiana executaram este mesmo número de treinamentos, sendo que esta última teve suas 30 primeiras avaliações executadas com valores aleatórios para os hiperparâmetros, como foi feito em (Bergstra et al., 2011).

4.5 Resultados

Tabela 4.3: Hiperparâmetros sintonizados com os métodos de busca em grade, aleatória e otimização Bayesiana, nos experimentos com o modelo de Floresta Aleatória. A tabela mostra os limites inferior e superior, e valores intermediários. Estes últimos são utilizados apenas na busca em grade.

<i>Hiperparâmetro</i>	<i>Limite Inferior</i>	<i>Valores Intermediário*</i>	<i>Limite Superior</i>
$N_{folha} (\%)^{**}$	0,1	1, 10, 20, 30 e 40	50
β	0	$3 \times (10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3} \text{ e } 10^{-2})$	0,3
$m (\%)^{**}$	1	15, 30, 45, 60 e 75	99

*Fez-se uso do valor Intermediário apenas no método de Busca em Grade.

**A porcentagem se refere às operações:

$N_{folha} = \lfloor n_{train} \times (N_{folha} \%) \rfloor$, em que n_{train} é o número de amostras no treinamento.

$m = \lfloor p \times (m \%) \rfloor$, em que p é o número total de atributos.

Demais hiperparâmetros, também relacionados ao modelo de floresta aleatória, foram definidos como: número de árvores ($B = 1.000$) permaneceu fixo, número mínimo de pontos abaixo do qual um nó se torna terminal ($N_{min} = 2$) mantido fixo, profundidade máxima que um nó pode atingir (d_{max}) foi colocado como não atuante, e como critério de pureza foi definido o índice de gini (equação 2.19). Estas configurações e intervalos de buscas mostraram-se adequados às necessidades dos testes de comparação dos métodos (relativa à variação na métrica AUC, à medida que os hiperparâmetros eram variados, e à possibilidade de alcançar bons resultados).

Em termos das configurações dos métodos de otimização hiperparamétrica, e seus próprios hiperparâmetros, foram usadas as mesmas definições utilizados nos testes com o modelo de árvore de decisão. E em relação ao número de repetições, foi definido um total de 10 amostragens para cada método, e cada conjunto de dados.

4.5 Resultados

4.5.1 Comparações Utilizando Árvores de Decisão

O resultado com os melhores modelos nas 30 repetições, executadas para cada conjunto e cada método, é mostrado nas Figuras 4.1, 4.2 e 4.3, por meio de gráficos de diagrama de caixa.

4.5 Resultados

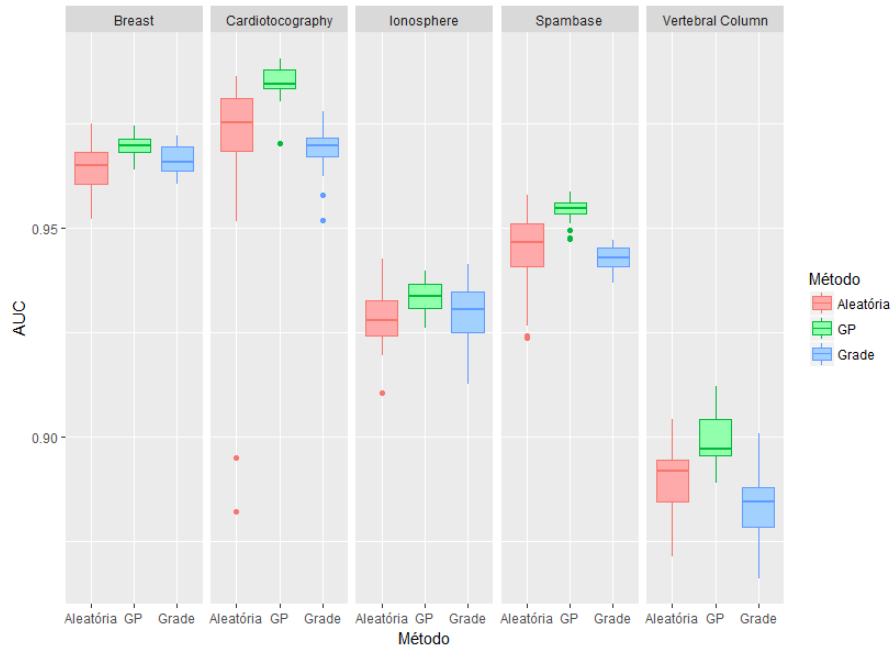


Figura 4.1: Resultados dos conjuntos Breast Cancer, Cardiotocography, Ionosphere, Spambase e Vertebral Column, para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de árvore de decisão.

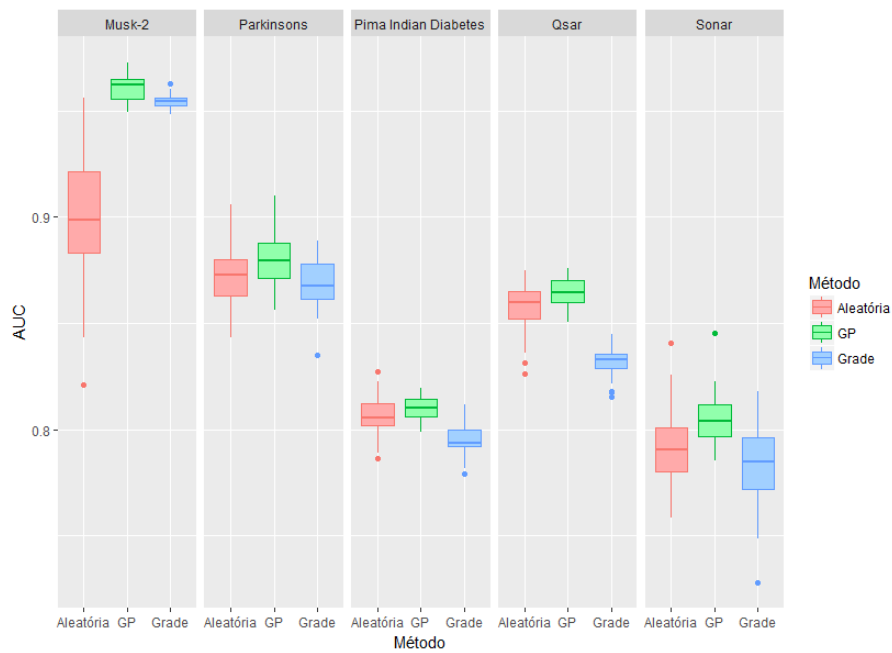


Figura 4.2: Resultados dos conjuntos Musk-2, Parkinsons, Pima Indian Diabetes, Qsar e Sonar, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de árvore de decisão.

4.5 Resultados

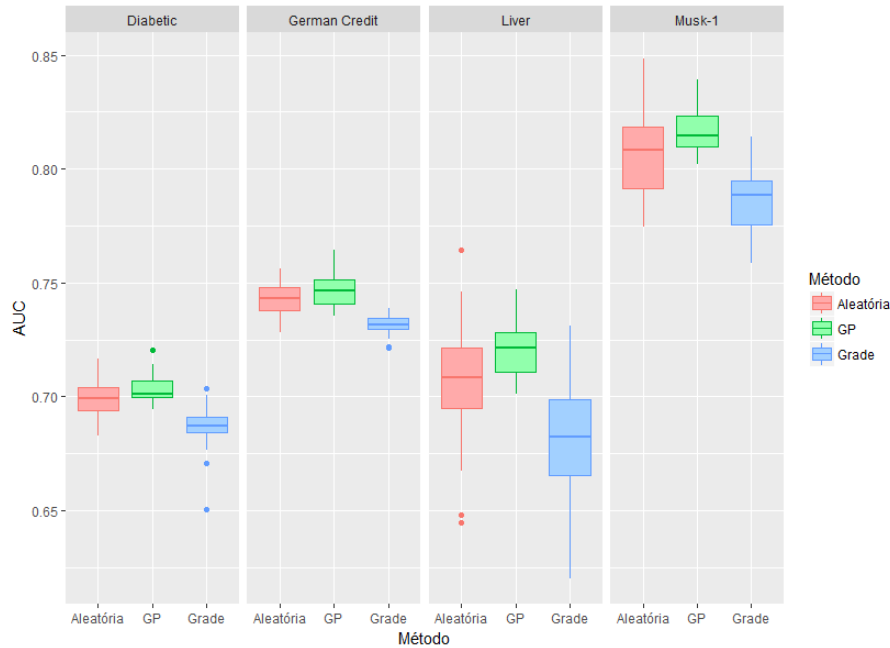


Figura 4.3: Resultados dos conjuntos Diabetic, German Credit, Liver e Musk-1, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de árvore de decisão.

Os resultados mostrados nos gráficos anteriores são resumidos, por meio da média e desvio padrão, e listados na Tabela 4.4.

Tabela 4.4: Resultados para os métodos de busca em grade, busca aleatória e otimização Bayesiana, na sintonia de hiperparâmetros de árvores de decisão. Valores apresentados são média±desvio padrão da métrica AUC, para as 30 repetições dos experimentos, para cada conjunto de dados.

Conjunto de Dados	Métodos		
	Grade	Aleatória	GP
Breast	0,966 ± 0,003	0,965 ± 0,006	0,970 ± 0,003
Cardiotocography	0,969 ± 0,005	0,969 ± 0,023	0,985 ± 0,004
Diabetic	0,687 ± 0,009	0,700 ± 0,009	0,704 ± 0,006
German	0,731 ± 0,004	0,743 ± 0,007	0,747 ± 0,007
Ionosphere	0,929 ± 0,007	0,928 ± 0,007	0,933 ± 0,004
Liver	0,682 ± 0,025	0,707 ± 0,025	0,720 ± 0,012
Musk-1	0,786 ± 0,014	0,807 ± 0,017	0,817 ± 0,009
Musk-2	0,955 ± 0,003	0,896 ± 0,031	0,961 ± 0,006
Parkinsons	0,869 ± 0,012	0,872 ± 0,015	0,880 ± 0,012
Qsar	0,831 ± 0,007	0,857 ± 0,012	0,865 ± 0,007
Sonar	0,784 ± 0,021	0,791 ± 0,020	0,805 ± 0,013
Spambase	0,943 ± 0,003	0,944 ± 0,010	0,954 ± 0,003
Vertebral	0,884 ± 0,009	0,890 ± 0,008	0,899 ± 0,006

4.5.1.1 Testes Estatísticos

Na realização do teste de Quade, foi adotado um nível de significância (α) igual a 5%. Para este valor de α , e com os graus de liberdade igual a $gl_1 = 2$ e $gl_2 = 26$ (calculados a partir do número de grupos e blocos), a tabela com os dados da distribuição F retorna a estatística com valor de 3,37. Já a estatística do teste de Quade resultou em um $F_q = 26,464$, o que causa a rejeição da hipótese nula de igualdade entre todos os métodos.

De acordo com os resultados do teste *post hoc* de comparação múltipla, a otimização Bayesiana difere significativamente dos demais métodos ($\alpha < 5\%$). Os métodos de busca em grade e busca aleatória diferem significativamente um do outro ($\alpha < 5\%$).

4.5.2 Comparações Utilizando Florestas Aleatórias

O resultado com os melhores modelos de floresta aleatória, nas 10 repetições executadas, é mostrado nas Figuras 4.4, 4.5 e 4.6 que conta com gráficos de diagrama de caixa com a métrica AUC discriminadas entre a busca em grade, busca aleatória e otimização Bayesiana.

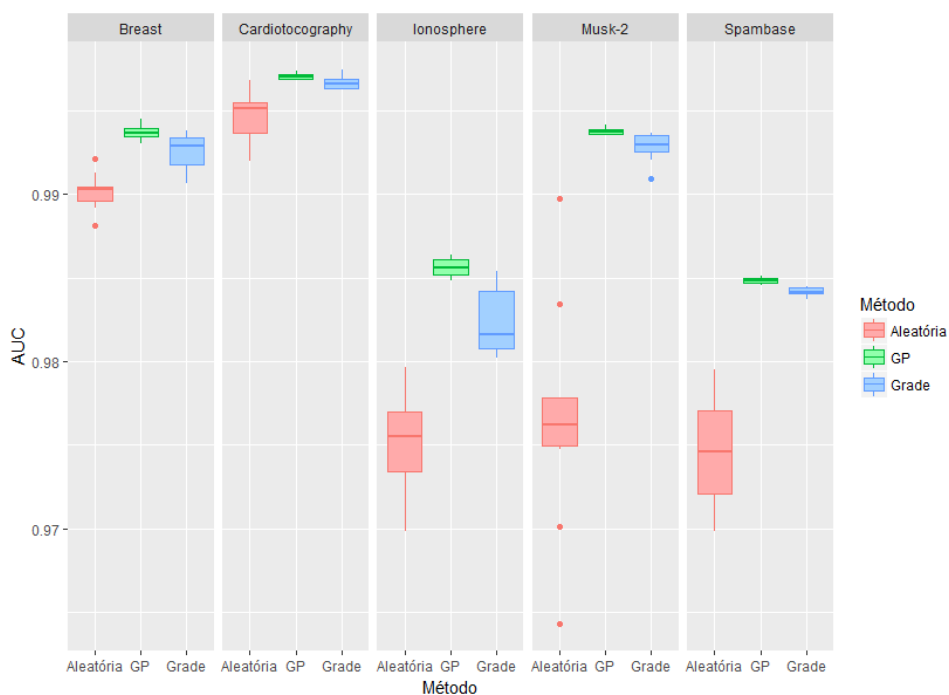


Figura 4.4: Resultados dos conjuntos Breast Cancer, Cardiotocography, Ionosphere, Musk-2 e Spambase, com a métrica AUC para as 10 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de floresta aleatória.

4.5 Resultados

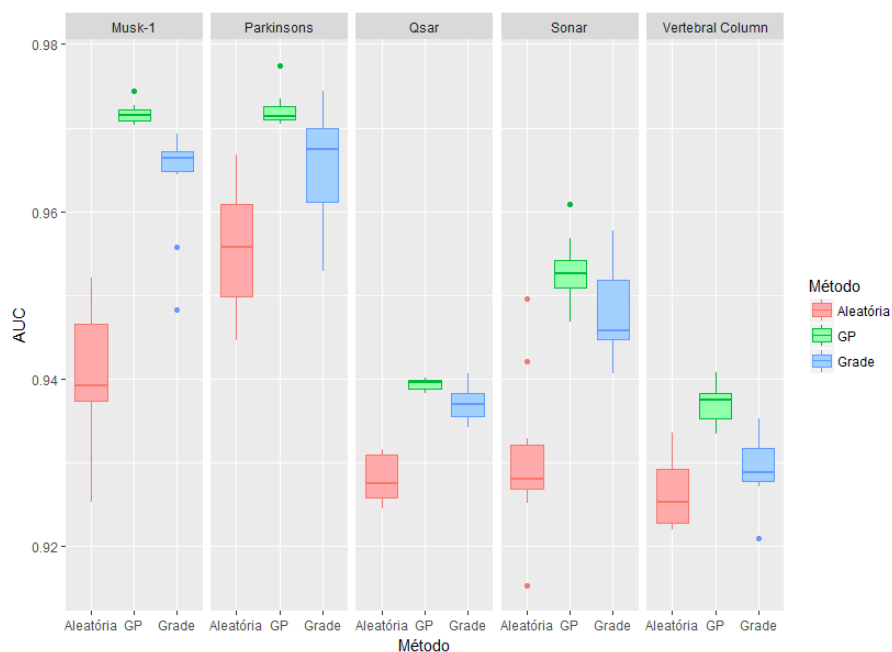


Figura 4.5: Resultados dos conjuntos Musk-1, Parkinsons, Qsar, Sonar e Vertebral Column, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de floresta aleatória.

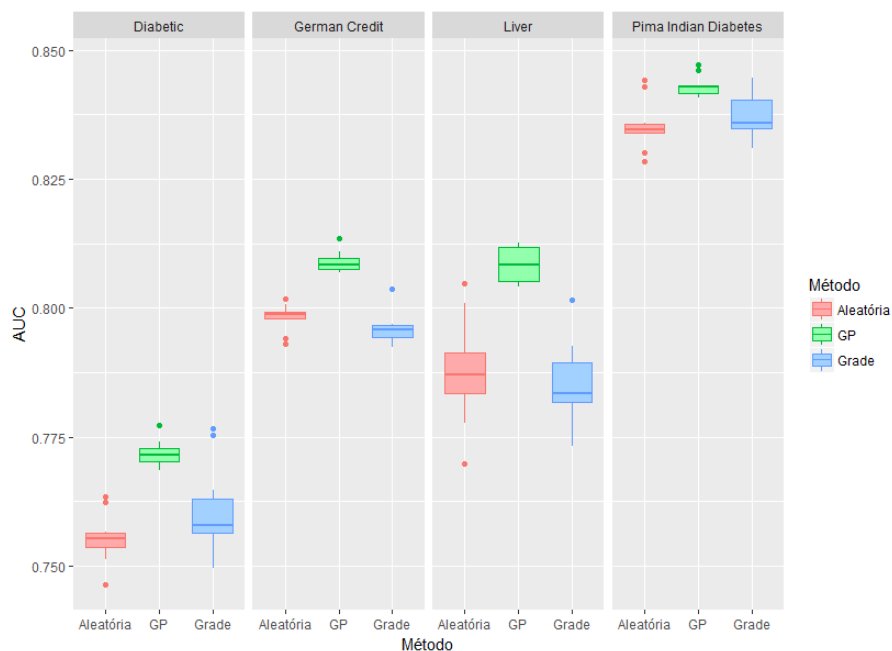


Figura 4.6: Resultados dos conjuntos Diabetic, German Credit, Liver e Pima Indian Diabetes, com a métrica AUC para as 30 repetições da busca em grade e aleatória, e a otimização Bayesiana, com o uso do modelo de floresta aleatória.

4.5 Resultados

Os resultados mostrados nos gráficos anteriores são resumidos, por meio da média e desvio padrão, e listados na Tabela 4.5.

Tabela 4.5: Resultados para os métodos de busca em grade, busca aleatória e otimização Bayesiana, na sintonia de hiperparâmetros de árvores de decisão. Valores apresentados são média \pm desvio padrão da métrica AUC, para as 30 repetições dos experimentos, para cada conjunto de dados.

Conjunto de Dados	Métodos		
	Grade	Aleatória	GP
Breast	0,993 \pm 0,001	0,990 \pm 0,001	0,994 \pm 0,001
Cardiotocography	0,997 \pm 0,001	0,995 \pm 0,002	0,998 \pm 0,000
Diabetic	0,761 \pm 0,009	0,756 \pm 0,005	0,772 \pm 0,003
German	0,796 \pm 0,003	0,798 \pm 0,003	0,809 \pm 0,002
Ionosphere	0,982 \pm 0,002	0,975 \pm 0,003	0,986 \pm 0,001
Liver	0,785 \pm 0,008	0,787 \pm 0,001	0,808 \pm 0,004
Musk-1	0,964 \pm 0,007	0,940 \pm 0,008	0,972 \pm 0,001
Musk-2	0,993 \pm 0,001	0,977 \pm 0,007	0,994 \pm 0,001
Parkinsons	0,966 \pm 0,007	0,955 \pm 0,008	0,972 \pm 0,002
PID	0,837 \pm 0,004	0,835 \pm 0,005	0,843 \pm 0,002
Qsar	0,937 \pm 0,002	0,928 \pm 0,003	0,939 \pm 0,001
Sonar	0,948 \pm 0,006	0,930 \pm 0,009	0,953 \pm 0,004
Spambase	0,984 \pm 0,001	0,975 \pm 0,003	0,985 \pm 0,001
Vertebral	0,929 \pm 0,004	0,926 \pm 0,004	0,937 \pm 0,002

4.5.2.1 Análise de Convergência da Otimização Bayesiana

Com o objetivo de avaliar a velocidade de convergência da técnica de otimização Bayesiana aplicada ao problema de busca ótima dos hiperparâmetros N_{folha} , β e m , do modelo de floresta aleatória, foram conduzidos testes com os conjunto de dados: Ionosphere, Liver e Musk-1, utilizando os mesmos intervalos de busca, mostrados na Tabela 4.3, com a definição de 500 treinamentos e 5 repetições dos testes para cada conjunto. O resultado é mostrado na Figura 4.7 que conta com os gráficos dos três conjuntos, nos quais a curva representa o maior valor da métrica AUC, até o ponto em questão, à medida que o modelo de floresta aleatória passa pelo ciclo envolvendo o treinamento, a avaliação, e o processo da otimização Bayesiana. Um detalhe importante nos gráficos é o uso de 30 avaliações iniciais com valores aleatórios.

4.6 Discussão

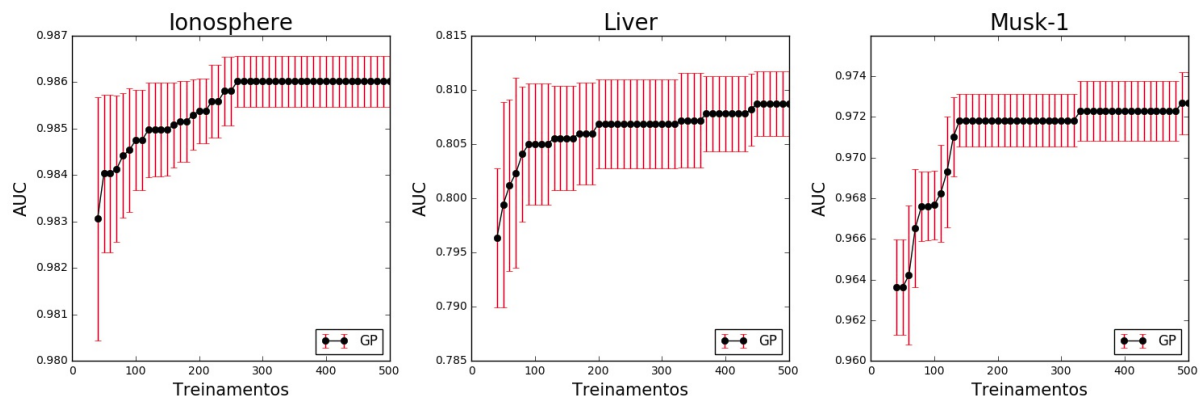


Figura 4.7: Resultado dos testes com o método de otimização Bayesiana na estimação dos hiperparâmetros N_{folha} , β e m de um modelo de floresta aleatória, para os conjuntos: Ionosphere, Liver e Musk-1. Em preto o maior valor da métrica AUC, até o ponto em questão. Desvio padrão em vermelho para 5 repetições.

Para o caso específico do modelo de floresta aleatória (com os hiperparâmetros N_{folha} , β e m) e os três conjunto de dados da Figura 4.7 o processo de otimização Bayesiana apresenta uma convergência mais rápida nos primeiros 150-200 treinamentos, experimentando uma relativa estagnação a partir de 300-400 treinamentos.

4.6 Discussão

Ao se analisar os diagramas de caixa, presentes nas Figuras 4.1, 4.2 e 4.3, e os resultados numéricos da Tabela 4.4, pode-se concluir que as médias da métrica AUC, obtidas pela técnica de otimização Bayesiana tiveram melhores valores que as médias dos outros dois métodos, em todos os conjuntos de dados testados. Ao se analisar o resultado da busca em grade e aleatória, enquanto que esta última obteve um número maior como segundo colocado (9), a busca em grade exibiu um menor desvio padrão dentro de cada conjunto de dados experimentados.

No teste estatístico de Quade, foi observado evidências para rejeitar a igualdade entre os 3 métodos de otimização hiperparamétrica, com um nível de confiança de 95%. Já o teste *post hoc* realizado com um nível de confiança de 95%, atestou uma diferença entre as técnicas de busca em grade e busca aleatória, assim como uma diferença entre a otimização Bayesiana e estes dois métodos anteriores. Porém, para um nível de confiança de 99%, apenas a diferença da otimização Bayesiana em relação aos demais métodos se mantêm.

Os resultados dos experimentos de comparação utilizando o modelo de floresta aleatória, Figuras 4.4, 4.5 e 4.6, mostram melhores resultados obtidos com a otimização Bayesiana para todos os conjuntos testados, ao se analisar a média da métrica AUC (Tabela 4.5).

Já os testes de convergência com o método de otimização Bayesiana, e os conjuntos Ionosphere, Liver e Musk-1, mostraram que para o ajuste dos hiperparâmetros N_{folha} , β e m , de

4.6 Discussão

uma floresta aleatória, é nos primeiros treinamentos (150-200) do modelo que ocorrem as mais rápidas razões de convergência, não havendo grandes melhorias após estas iterações.

Conclusões

5.1 Conclusões

Este trabalho apresentou uma análise comparativa entre os métodos de otimização hiperparamétrica: busca em grade, busca aleatória e otimização Bayesiana utilizando o processo gaussiano, para a sintonia de hiperparâmetros de árvores de decisão e florestas aleatórias, empregadas em problemas de classificação binária, utilizando conjuntos de dados do repositório UCI-Machine Learning.

O propósito deste estudo foi investigar se algum dos métodos de otimização hiperparamétrica analisados neste trabalho apresenta um resultado significativamente melhor que os demais.

Os modelos de aprendizado escolhidos para os experimentos, árvore de decisão e floresta aleatória, são algoritmos não paramétricos, possuidores de hiperparâmetros para o controle de ajuste do modelo aos dados do problema e cujo o uso é bastante difundido. Os conjuntos de dados utilizados nos experimentos são problemas usualmente tratados pela comunidade científica de aprendizado de máquina.

Em relação às técnicas comparadas, o método de busca em grade possui como única tarefa a definição de sua grade composta pelas combinações de hiperparâmetros a serem testados. O método de busca aleatória exige como requisitos a definição de limites, máximo e mínimo, para a busca de cada hiperparâmetro, bem como de função(ões) de distribuição(ões). Por outro lado, a otimização Bayesiana exige a definição de um modelo *a priori* para o processo gaussiano, contando com as funções média e de covariância (e seus hiperparâmetros), uma função de aquisição (e seu hiperparâmetro) e um método de otimização (e seus hiperparâmetros).

Apesar do esforço extra de configuração que a otimização Bayesiana impõe (comparado as buscas em grade e aleatória), seu uso como ferramenta de otimização hiperparamétrica é muito válido, como pode ser visto nos seus resultados que mostraram um desempenho superior em ambas as configurações dos experimentos: testes com árvores de decisão e número de treinamentos igual a 30 (sendo 15 iniciais aleatórios p/ a otimização Bayesiana) e testes com floresta aleatória e número de treinamentos igual a 392 (sendo 30 iniciais aleatórios p/ a otimização Bayesiana).

Um fator que deve ser levado em consideração, e que este trabalho não foi objeto de estudo, seria o tempo de execução dos métodos de otimização hiperparamétrica quando um

mesmo número de treinamentos são realizados. Desconsiderando os tempos gastos com os treinamentos do modelo de aprendizado, as técnicas de busca em grade e aleatória possuem tempos que qualitativamente poderiam ser considerados como desprezíveis. O que não é o caso da otimização Bayesiana, que requer um custo computacional no refinamento do modelo *a posteriori* e na otimização do mesmo.

Uma dificuldade encontrada neste estudo foi a condução dos experimentos com a limitada quantidade de recursos computacionais. De fato, alguns trabalhos que foram revisados para a presente pesquisa tiveram a sua disposição, meios mais robustos para a simulação dos scripts, que envolveram o uso de *clusters* com placas gráficas GPUs, o que fornece um grande poder de processamento para os cálculos implementados.

5.2 Trabalhos Futuros

A seguir são listadas sugestões para trabalhos futuros relacionados ao tema deste estudo:

1. A análise de comparação feita neste trabalho incluiu os métodos de otimização hiperparamétrica: busca em grade e aleatória, e a otimização Bayesiana. Existe ainda outras vertentes da otimização Bayesiana, como as técnicas SMAC e TPE. Outras opções de métodos que também podem ser comparados são técnicas de algoritmos evolucionários e a otimização via gradiente, sendo a implementação desta última mais trabalhosa.
2. O presente estudo se restringiu aos conjuntos de dados do repositório UCI-Machine Learning. A análise de outros conjuntos, talvez de problemas reais, possa trazer uma compreensão mais abrangente do comportamento dos métodos de otimização hiperparamétrica, aqui estudados, empregados na sintonia dos hiperparâmetros dos modelos árvore de decisão e floresta aleatória.
3. Outro ponto de restrição, é que os testes do presente trabalho foram concentrados em hiperparâmetros de árvores de decisão e florestas aleatórias. Outros tipos de modelos de aprendizado podem ser explorados, como por exemplo: SVM, MLP, Redes Neurais de Base Radial, CNN, outros modelos baseados em árvore de decisão, como por exemplo: Gradient Boosting Machine, e outros. Diferentes espaços hiperparamétricos podem gerar funções que mapeiam os hiperparâmetros nas avaliações de uma função custo com diferentes complexidades. E talvez os métodos de otimização possam exibir distintas habilidades de encontrar o ponto ótimo nesta variedade de funções.
4. Em se tratando do número de hiperparâmetros sintonizados, neste estudo foi feita a otimização hora de 2, hora de 3. Uma sugestão de investigação seria a comparação utilizando um número maior destes, de modo a confirmar se a técnica de otimização Bayesiana possui um melhor resultado tanto no caso

5.2 Trabalhos Futuros

com poucos hiperparâmetros, quanto em uma situação que apresenta muitos hiperparâmetros.

Referências Bibliográficas

- Arlot, S., Celisse, A., et al. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79. [11](#)
- Asuncion, A. e Newman, D. (2007). Uci machine learning repository. [45](#)
- Baba, N. (1981). Convergence of a random optimization method for constrained optimization problems. *Journal of Optimization Theory and Applications*, 33(4):451–461. [33](#)
- Belgiu, M. e Drăguț, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31. [22](#)
- Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900. [2](#), [29](#)
- Bergstra, J. e Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305. [v](#), [2](#), [27](#), [30](#), [33](#), [34](#)
- Bergstra, J., Yamins, D., e Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, páginas 115–123. [26](#), [30](#)
- Bergstra, J. S., Bardenet, R., Bengio, Y., e Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, páginas 2546–2554. [1](#), [2](#), [27](#), [49](#)
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, 1th edição. [1](#), [30](#)
- Brazdil, P., Carrier, C. G., Soares, C., e Vilalta, R. (2008). *Metalearning: Applications to data mining*. Springer Science & Business Media. [31](#)
- Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2):123–140. [23](#)
- Breiman, L. (1996b). Out-of-bag estimation. [24](#)
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. [2](#)

REFERÊNCIAS BIBLIOGRÁFICAS

- Breiman, L., Friedman, J., Stone, C. J., e Olshen, R. A. (1984). *Classification and regression trees*. CRC press. [2](#), [13](#), [16](#)
- Brochu, E., Cora, V. M., e De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*. [35](#), [36](#), [40](#)
- Byrd, R. H., Lu, P., Nocedal, J., e Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208. [44](#)
- Cherkassky, V. e Mulier, F. M. (2007). *Learning from data: concepts, theory, and methods*. John Wiley & Sons. [5](#)
- Claesen, M. e De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*. [29](#), [30](#)
- Conover, W. J. (1980). *Practical nonparametric statistics*. Wiley New York. [48](#)
- Criminisi, A., Robertson, D., Konukoglu, E., Shotton, J., Pathak, S., White, S., e Siddiqui, K. (2013). Regression forests for efficient anatomy detection and localization in computed tomography scans. *Medical image analysis*, 17(8):1293–1303. [22](#)
- Criminisi, A. e Shotton, J. (2013). *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media. [13](#), [22](#), [25](#)
- de Castro, C. L. (2011). *Novos critérios para seleção de modelos neurais em problemas de classificação com dados desbalanceados*. Tese de Doutorado, Universidade Federal de Minas Gerais. [10](#)
- Del Río, S., López, V., Benítez, J. M., e Herrera, F. (2014). On the use of mapreduce for imbalanced big data using random forest. *Information Sciences*, 285:112–137. [22](#)
- Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30. [48](#)
- Efron, B. (1979). Computers and the theory of statistics: thinking the unthinkable. *SIAM review*, 21(4):460–480. [22](#)
- Eggenesperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., e Leyton-Brown, K. (2013). Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10. [28](#), [48](#)
- Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874. [11](#)
- Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368. [43](#)

REFERÊNCIAS BIBLIOGRÁFICAS

- Friedrichs, F. e Igel, C. (2005). Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117. [2](#), [29](#)
- Geman, S., Bienenstock, E., e Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58. [7](#)
- Genuer, R., Poggi, J.-M., Tuleau-Malot, C., e Villa-Vialaneix, N. (2017). Random forests for big data. *Big Data Research*. [22](#)
- Hanley, J. A. e McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36. [11](#)
- Hastie, T., Tibshirani, R., e Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag. [8](#), [11](#), [12](#), [14](#), [15](#), [16](#), [17](#), [21](#), [22](#), [23](#), [25](#)
- Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification. [2](#), [27](#)
- Hutter, F., Hoos, H. H., e Leyton-Brown, K. (2010). Sequential model-based optimization for general algorithm configuration (extended version). *Technical Report TR-2010–10, University of British Columbia, Computer Science, Tech. Rep.* [1](#), [2](#), [27](#)
- James, G., Witten, D., Hastie, T., e Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer. [21](#)
- Japkowicz, N. e Shah, M. (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press. [8](#)
- Krizhevsky, A., Sutskever, I., e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, páginas 1097–1105. [2](#), [30](#)
- Louppe, G. (2014). *Understanding random forests: From theory to practice*. Tese de Doutorado, Universidade de Lieja. [13](#), [18](#)
- Luketina, J., Berglund, M., e Raiko, T. (2015). Scalable gradient-based tuning of continuous regularization hyperparameters. *arXiv preprint arXiv:1511.06727*. [29](#)
- MacKay, D. J. (1998). Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166. [36](#)
- MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press. [9](#), [43](#)

REFERÊNCIAS BIBLIOGRÁFICAS

- Maclaurin, D., Duvenaud, D., e Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, páginas 2113–2122. 2, 29
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. 1, 5, 30
- Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100. 43
- Pyle, D. e Jose, C. S. (2015). An executive guide to machine learning. Disponível em <http://www.mckinsey.com/industries/high-tech/our-insights/an-executives-guide-to-machine-learning>. Acessado em 20 Fevereiro 2016. 1
- Quade, D. (1979). Using weighted rankings in the analysis of complete blocks with additive block effects. *Journal of the American Statistical Association*, 74(367):680–683. 48
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106. 13
- Quinlan, J. R. (1993). C4. 5: Programming for machine learning. *Morgan Kauffmann*, 38. x, 13
- Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier. 14
- Rasmussen, C. E. e Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press, primeira edição. 36, 37
- Segal, M. R. (2004). Machine learning benchmarks and random forest regression. *Center for Bioinformatics & Molecular Biostatistics*. 25
- Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., e de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175. 36, 37, 39, 41
- Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., e Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124. 22
- Snoek, J., Larochelle, H., e Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, páginas 2951–2959. 2, 26, 28
- Solis, F. J. e Wets, R. J.-B. (1981). Minimization by random search techniques. *Mathematics of operations research*, 6(1):19–30. 33
- Srinivas, N., Krause, A., Kakade, S. M., e Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*. 40
- Swersky, K., Snoek, J., e Adams, R. P. (2014). Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*. 1, 28

REFERÊNCIAS BIBLIOGRÁFICAS

- Thornton, C., Hutter, F., Hoos, H. H., e Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13*, páginas 847–855, New York, NY, USA. ACM. [8](#), [28](#), [31](#)
- Vapnik, V. (1995). *The nature of statistical learning theory*. Springer. [1](#), [5](#), [6](#)
- Vapnik, V. N. (1998). *Statistical learning theory*. Wiley New York. [8](#)
- Verikas, A., Gelzinis, A., e Bacauskiene, M. (2011). Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 44(2):330–349. [22](#)
- Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37. [13](#), [14](#)
- Yang, B.-S., Di, X., e Han, T. (2008). Random forests classifier for machine fault diagnosis. *Journal of mechanical science and technology*, 22(9):1716–1725. [22](#)
- Zhu, C., Byrd, R. H., Lu, P., e Nocedal, J. (1994). Lbfgs-b: Fortran subroutines for large-scale bound constrained optimization. *Report NAM-11, EECS Department, Northwestern University*. [44](#)