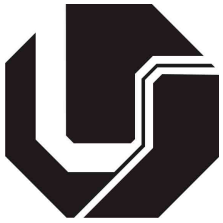
Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Pós-Graduação em Engenharia Elétrica



Algoritmo de Aprendizado Supervisionado - Baseado em Máquinas de Vetores de Suporte - Uma Contribuição Para o Reconhecimento de Dados Desbalanceados

Orientador: Antônio Cláudio Paschoarelli Veiga, Dr
Orientado: Hugo Leonardo Pereira Rufino

Setembro
2011

Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Pós-Graduação em Engenharia Elétrica

Algoritmo de Aprendizado Supervisionado - Baseado em Máquinas de Vetores de Suporte - Uma Contribuição Para o Reconhecimento de Dados Desbalanceados

Tese apresentada por Hugo Leonardo Pereira Rufino à Universidade Federal de Uberlândia como requisito parcial para obtenção do título de Doutor em Ciências, avaliado em 26/09/2011 pela Banca Examinadora:

Antônio Cláudio Paschoarelli Veiga, Dr (UFU) - Orientador
Celso Gonçalves Camilo Junior, Dr (UFG)
Gilberto Arantes Carrijo, PhD (UFU)
Keiji Yamanaka, PhD (UFU)
Marley Maria Bernardes Rebuzzi Vellasco, PhD (PUC-RJ)

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG - Brasil

R926a Rufino, Hugo Leonardo Pereira, 1978-
Algoritmo de aprendizado supervisionado - baseado em máquinas de vetores de suporte : uma contribuição para o reconhecimento de dados balanceados / Hugo Leonardo Pereira Rufino. - 2011.

107 f. : il.

Orientador: Antônio Cláudio Paschoarelli Veiga.

Tese (doutorado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.
Inclui bibliografia.

1. Engenharia elétrica - Teses. 2. Aprendizado do computador - Teses. 3. Inteligência artificial - Teses. 4. Algoritmos de computador - Teses. I. Veiga, Antônio Cláudio Paschoarelli, 1963- II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

Algoritmo de Aprendizado Supervisionado - Baseado em Máquinas de Vetores de Suporte - Uma Contribuição Para o Reconhecimento de Dados Desbalanceados

Hugo Leonardo Pereira Rufino

Tese apresentada à Universidade Federal de Uberlândia como requisito parcial para obtenção do título de Doutor em Ciências.

Antônio Cláudio Paschoarelli Veiga, Dr	Alexandre Cardoso, Dr
Orientador	Coordenador do Curso de Pós-Graduação

Aos meus pais José da Cruz e Luciene, pelos exemplos de respeito, honestidade e simplicidade. E à Paula, minha esposa, pelo carinho, dedicação e contribuição no desenvolvimento deste trabalho.

Agradecimentos

Ao meu orientador Professor Antônio Cláudio Paschoarelli Veiga pela amizade, incentivo e confiança no meu trabalho.

Aos Professores Gilberto Arantes Carrijo e Keiji Yamanaka. Os conhecimentos transmitidos em suas aulas foram importantes nessa jornada.

À Professora Marley Maria Bernardes Rebuzzi Vellasco e ao Professor Celso Gonçalves Camilo Júnior pela aceitação em participar da banca e pelas observações em relação ao meu trabalho, que muito contribuíram para a melhoria do mesmo.

À Cinara Fagundes Paranhos Mattos, pela simpatia e presteza na secretaria da Pós-Graduação.

Aos demais, mestres, amigos ou simples conhecidos, que foram, em níveis diferentes, fundamentais para a minha formação e que me prestaram auxílio em muitas ocasiões.

Resumo

RUFINO, Hugo L. P. *Algoritmo de Aprendizado Supervisionado - Baseado em Máquinas de Vetores de Suporte - Uma Contribuição Para o Reconhecimento de Dados Desbalanceados*, Uberlândia, Faculdade de Engenharia Elétrica - UFU, 2011.

O aprendizado de máquina em conjuntos de dados que possuam classes desbalanceadas tem recebido considerável atenção na comunidade científica, pois os algoritmos de classificação tradicionais não fornecem um desempenho satisfatório. Este baixo desempenho pode ser justificado pelo fato das técnicas tradicionais de aprendizado de máquina considerarem que cada classe presente em um conjunto de dados possui um número aproximadamente igual de instâncias. Entretanto, a maioria dos conjuntos de dados reais possuem classes com uma distribuição desbalanceada, onde uma classe de dados está super representada em comparação com outras classes. Isto faz com que surjam classificadores com uma alta precisão para a predição da classe majoritária e com baixa precisão para prever a classe minoritária. Logo, a classe minoritária é ignorada pelo classificador. Esta predisposição do classificador em relação à classe majoritária ocorre em função dos classificadores serem projetados para maximizar a precisão em relação ao conjunto de dados que está sendo utilizado para o treinamento. No treinamento do classificador é assumido que quando for fazer a predição de dados ainda não vistos, estes terão a mesma distribuição dos dados que foram utilizados no treinamento. Isto limita sua habilidade em reconhecer exemplos da classe minoritária. Várias melhorias nos algoritmos tradicionais de classificação têm sido propostas na literatura, onde foram feitas considerações a nível de dados e a nível de algoritmos. O primeiro utiliza diversas formas de reamostragem, tal como super-amostragem de exemplos da classe minoritária, sub-amostragem de exemplos da classe majoritária ou a combinação de ambos. Os últimos tentam adaptar (inserindo custos diferenciados em exemplos da classe minoritária e majoritária, alterando *kernels*, e outras técnicas) os algoritmos de classificação já existentes para melhorar o desempenho da classe minoritária. Vários algoritmos na forma de um comitê de máquinas também são reportados como meta-técnicas para trabalhar com classes desbalanceadas. Esta tese estuda os principais algoritmos que lidam com classes desbalanceadas, destacando suas principais características como: a geração de novos exemplos "sintéticos" ao invés da replicação de dados de forma aleatória, no processo de super-amostragem; o uso de penalidades diferentes para erros de classificação da classe minoritária e majoritária; e a utilização de comitês de máquinas para que os classificadores gerados possuam uma capacidade de generalização maior. Após o levantamento das contribuições que cada algoritmo fornece, foi feito um estudo se poderia obter algo mais das características de cada um. Foi feita uma modificação no algoritmo que gera novos exemplos "sintéticos" de forma que reduzisse a possibilidade de geração de novos elementos na região incorreta. Como em conjuntos de dados altamente desbalanceados, a geração de elementos "sintéticos" não é suficiente para equilibrar o conjunto, houve a necessidade da criação de um novo algoritmo para efetuar uma sub-amostragem de exemplos da classe majoritária. E, para melhorar a capacidade de generalização do classificador gerado, também foi feita uma modificação em um algoritmo de comitês de máquinas. Utilizando estas três etapas, obteve-se um algoritmo composto que possui uma taxa de acerto na classificação de dados melhor que os algoritmos nos quais se baseou.

Palavras-chave: Aprendizado Supervisionado, Máquinas de Vetores de Suporte e Conjuntos de Dados Desbalanceados.

Abstract

RUFINO, Hugo L. P. *Supervised learning Algorithm - Based on Support Vector Machines - A Contribution to the Recognition of Unbalanced Data*, Uberlândia, Faculty of Electric Engineering - UFU, 2011.

The machine learning in datasets that have unbalanced classes, has received considerable attention in the scientific community, because the traditional classification algorithms don't provide a satisfactory performance. This low performance can be explained by the fact that the traditional techniques of machine learning consider that each class present in the database has an approximately equal number of instances. However, most real datasets, have classes with an unbalanced distribution, where one class is over represented in comparison with the others. This gives rise to classifiers with high accuracy to predict the majority class and low accuracy for predicting the minority class. Therefore, the minority class is ignored by the classifier. This predisposition of the classifier for the majority class occurs, because the classifiers are designed to maximize accuracy in relation to the database being used for training. In training the classifier, it is assumed that when making the prediction of data not yet seen, they have the same distribution of the data that were used in training. This limits its ability to recognize examples of the minority class. Several improvements in the traditional classification algorithms have been proposed in the literature, where considerations were made at the level of data and algorithms. The former uses various ways of resampling, such as oversampling of examples from the minority class, undersampling the majority class or a combination of both. The latter attempt to adapt (by inserting different costs in the minority class examples and majority, changing kernels and other techniques) the existing classification algorithms to improve the performance of minority class. Several algorithms in the form of a ensemble machine, are also reported as meta-techniques for working with unbalanced classes. This thesis studies the main algorithms that deal with unbalanced class, highlighting its main features as: the generation of new "synthetic" examples instead of replicating data at random, in the process of oversampling; the use of different penalties to misclassification of the minority and majority class; and the use of ensembles for that the generated classifiers have a greater ability to generalize. After assessing the contributions that each algorithm provides, a study was done if one could get something more of the characteristics of each one. It was made a modification in the algorithm that generates new "synthetic" examples of way that reduces the possibility of generating new elements in the incorrect region. As with highly unbalanced datasets, the generation of "synthetic" elements is not enough to balance the whole, there was a need to develop a new algorithm to perform an undersampling the majority class examples. And to enhance the generalization ability of the generated classifier, was also made a change to an ensemble algorithm. Using these three steps, we obtained an compound algorithm that has a hit rate of data classification better than the algorithms on which it was relied.

Keywords: Supervised Learning, Support Vector Machines, Unbalanced Datasets.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

KKT Karuch-Kuhn-Tucker

IA Inteligência Artificial

AM Aprendizado de Máquina

TAE Teoria do Aprendizado Estatístico

VC Vapnik-Chervonenkis

MRE Minimização do Risco Empírico

iid independente e identicamente distribuídos

fdp função de distribuição de probabilidade

SVM Máquinas de Vetores de Suporte

MREs Minimização do Risco Estrutural

IEEE Institute of Electrical and Electronics Engineers

ACM Association for Computing Machinery

ROC Receiver Operating Characteristics

AUC Area Under the Receiver Operating Characteristics (ROC) Curve

BED Boundary Elimination and Domination

Lista de Símbolos

m indica a quantidade de padrões

n indica a quantidade de dimensão(ões)

x um número escalar

$\mathbf{x}$ um vetor

$P(\mathbf{x}, y)$ função de distribuição de probabilidade

$\mathbf{R}^n$ o valor em n indica a dimensão de um vetor $\mathbf{x}$, ou seja, quantos atributos cada vetor $\mathbf{x}$ possui

$f(\mathbf{x})$ função (ideal) gerada pela máquina de aprendizado

$\mathfrak{F}$ conjunto de todas as funções que uma máquina de aprendizado pode gerar

$L(f(\mathbf{x}), y)$ medida de discrepância, função de custo ou função de perda

$R(f)$ risco esperado

$R_{emp}(f)$ risco empírico

α conjunto de parâmetros ajustáveis da função a ser gerada pela máquina de aprendizado

$\hat{f}$ função gerada pela máquina de aprendizado, baseando-se nos pares $(\mathbf{x}, y)$ utilizados no treinamento

$\hat{f}(\mathbf{x}, \alpha)$ forma mais detalhada de representar a função gerada pela máquina de aprendizado

$p(y \mid \mathbf{x})$ densidade condicional

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Motivação

O aprendizado de máquina é o campo da Inteligência Artificial (IA) responsável pelo desenvolvimento de modelos (hipóteses) gerados a partir de dados, e que automaticamente aperfeiçoam-se com a experiência. Existem diversas aplicações para aprendizado de máquina, tais como processamento de linguagem natural, detecção de fraudes, análise de imagens ou reconhecimento de padrões (Mitchell, 1997).

Por volta da metade da década de 90, uma técnica de aprendizado de máquina que ganhou destaque foram as Máquinas de Vetores de Suporte (SVM), por seu alto desempenho de classificação em vários domínios, incluindo reconhecimento de padrões, mineração de dados e bioinformática. As SVM possuem um forte embasamento teórico e uma ótima capacidade de generalização (Vapnik, 1998). Também na década de 90, comitês de máquinas foram criados e popularizados através de *bagging* e *boosting*. O sucesso dos comitês de máquinas se deve ao fato de serem mais precisos que qualquer um dos classificadores que o compõem (Breiman, 1996; Freund & Schapire, 1996; Schapire, 1990). *Bagging* e *boosting* foram criados para trabalhar com classificadores fracos[1]. Mas logo pesquisadores perceberam que também é possível

[1] Um classificador fraco possui uma probabilidade de acerto um pouco maior que $0,5$ (ou seja, um pouco melhor que uma classificação de forma randômica), enquanto um forte possui uma probabilidade de acerto $1 - \epsilon$, para ϵ tão pequeno quanto se queira (Kearns & Valiant, 1994; Hulley & Marwala, 2007).

construir comitês de máquinas utilizando um classificador forte (Kim, Pang, Je, Kim, & Bang, 2003; Lima, Neto, & Melo, 2009). Com isso, passaram a incluir nos experimentos as SVM, onde obtiveram sucesso em suas aplicações. Desde então têm surgido novas metodologias para o aprimoramento das técnicas de classificação. Uma meta-técnica que vem se destacando no processo de melhoria da precisão de um classificador é o algoritmo *AdaBoost* (Freund & Schapire, 1997). Continua se destacando porque várias outras estratégias de aprimoramento do processo de classificação como: algoritmos para lidar com conjuntos de dados que possuam classes desbalanceadas (Ditzler, Muhlbaier, & Polikar, 2010; Sun, Kamel, Wong, & Wang, 2007) ou algoritmos que conseguem lidar com treinamento de dados que são inseridos em lotes (algoritmos de aprendizado incremental)(Polikar, Udpa, Honavar, & Udpa, 2000; Kidera, Ozawa, & Abe, 2006), têm como base o *AdaBoost*. Este trabalho dará ênfase na primeira estratégia de aprimoramento.

O problema de classes desbalanceadas é relativamente novo. Surgiu quando o aprendizado de máquina desenvolveu-se de uma ciência embrionária para tecnologia aplicada, amplamente utilizada no mundo dos negócios, indústria e pesquisa científica. Embora os profissionais já tenham ouvido falar deste problema anteriormente, este se destacou na área de aprendizado de máquina há cerca de uma década (em 2000 ocorreu o primeiro workshop em "Aprendizado a partir de Conjuntos de Dados Desbalanceados"[2]). Este problema geralmente acontece quando, em um problema de classificação, existem muito mais instâncias de algumas classes que em outras (T.-Y. Liu, 2009).

Existem diversos trabalhos que propõem melhorias tanto a nível de dados quanto a nível de algoritmos: SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) propõe a geração de novos elementos "sintéticos" para a classe minoritária; o uso diferentes penalidades para a classe positiva e negativa, em caso de classificação incorreta (Veropoulos, Campbell, & Cristianini, 1999); a união das duas técnicas citadas anteriormente (Akbani, Kwek, & Japkowicz, 2004); a inserção de diferentes penalidades na função de distribuição de pesos que é utilizada no algoritmo *AdaBoost* (Freund & Schapire, 1997) deu origem aos algoritmos AdaC1, AdaC2

[2]no original é referenciado como AAAI-2000 Workshop on "Learning from Imbalanced Data Sets".

e AdaC3 (Sun et al., 2007); o uso de um comitê de máquinas para melhorar a taxa de reconhecimento em conjuntos de dados desbalanceados (X.-Y. Liu, Wu, & Zhou, 2009) e o uso de heurísticas para efetuar sub e super-amostragem de dados (algoritmo *BED*), fazendo com que os conjuntos de dados fiquem menos desbalanceados (Castro, Carvalho, & Braga, 2009).

Depois de feito um estudo nos algoritmos citados anteriormente, descobriu-se que existem algumas situações em que o uso da solução proposta pode levar a uma redução da taxa de reconhecimento de dados no processo de classificação. No caso do algoritmo SMOTE, não é feito um controle da posição onde o novo elemento é criado. Em instâncias da classe minoritária que estão fora de sua região de classificação (ruídos), ao gerar novo elementos "sintéticos", podem aparecer mais ruídos e prejudicar o processo de classificação. A técnica proposta por Veropoulos et al. (1999) melhora a representatividade da classe minoritária. Mas sozinha, não possui um desempenho tão bom quanto utilizada com uma técnica de sub-amostragem de dados. Na solução proposta por Akbani et al. (2004) é desprezada a técnica de sub-amostragem, com a afirmação de que isto leva à perda de dados importantes. Mas nesta afirmação o autor considera apenas a sub-amostragem de forma randômica. Caso seja utilizada alguma heurística, esta possibilidade de perda pode ser superada (Gu, Cai, Zhu, & Huang, 2008; He & Garcia, 2009). Os algoritmos criados por Sun et al. (2007) possuem o mesmo princípio de funcionamento da técnica apresentada por Veropoulos et al. (1999), mas funciona exclusivamente para o algoritmo *AdaBoost*. O autor X.-Y. Liu et al. (2009) também não utilizou a técnica de sub-amostragem de dados, utilizando o mesmo argumento de Akbani et al. (2004). No algoritmo *BED*, caso um elemento pertencente à classe minoritária for considerado um ruído, ele é eliminado do conjunto. Dependendo da quantidade de ruídos, esta técnica pode levar à eliminação de muitos elementos, prejudicando o processo de super-amostragem.

Diante do que foi exposto, verificou-se a necessidade de uma continuar a pesquisa na área de conjuntos de dados desbalanceados, com o objetivo de melhorar algoritmos já existentes ou encontrar uma combinação entre algoritmos de forma a extrair o máximo que cada técnica pode fornecer.

1.2 Objetivos

O destaque que as SVM e as técnicas utilizadas para classificar conjuntos de dados desbalanceados vêm recebendo na comunidade científica motivaram o estudo das SVM e dos algoritmos que lidam com classes desbalanceadas, com o objetivo da criação de um novo algoritmo que tenha algumas das melhores características desses algoritmos e também um desempenho superior aos quais ele foi baseado. Para isto foram selecionados alguns algoritmos que tratam o problema de classificação de conjuntos de dados desbalanceados. Dentre eles: SMOTE (Chawla et al., 2002), que é um algoritmo para efetuar super-amostragem em conjuntos de dados da classe minoritária; o uso de diferentes penalidades para classificações incorretas de diferentes classes (Veropoulos et al., 1999); e um algoritmo de comitê de máquinas denominado *EasyEnsemble* (X.-Y. Liu et al., 2009). Foi feita uma análise em cada um para verificar o que poderia ser melhorado e chegou-se a uma modificação no algoritmo SMOTE de forma que reduzisse a possibilidade de geração de novos elementos na região incorreta. Como em conjuntos de dados desbalanceados, a geração de elementos "sintéticos" nem sempre é suficiente para equilibrar o conjunto, criou-se um novo algoritmo para efetuar uma sub-amostragem de exemplos da classe majoritária. E, para melhorar a capacidade de generalização do classificador gerado, também foi feita uma modificação no algoritmo de comitês de máquinas *EasyEnsemble*. Unindo estas três etapas, obteve-se um algoritmo composto que possui uma taxa de acerto na classificação de dados melhor que os algoritmos nos quais se baseou.

1.3 Estrutura da Tese

Esta tese está organizada da seguinte forma. No capítulo 2 é apresentada uma breve introdução sobre a teoria de aprendizagem estatística, ressaltando conceitos importantes como risco empírico, risco esperado e dimensão VC. Também é apresentada a técnica de classificação de dados SVM, enfatizando as classes que são linearmente separáveis, as que não são linearmente separáveis e o mapeamento dos dados em alta dimensão. No capítulo 3 são apresentados

os problemas gerados por conjuntos de dados em que existam classes desbalanceadas, e também alguns algoritmos propostos para resolver este tipo de problema. No capítulo 4 são apresentadas três etapas que auxiliam no tratamento de conjuntos de dados desbalanceados para que se obtenha um desempenho de classificação de dados satisfatório. Essas três etapas compõem o algoritmo criado como resultado desta tese de doutorado. No capítulo 5 são apresentados os resultados obtidos da aplicação do algoritmo criado nesta tese em 20 conjuntos de dados, mostrando que possui um melhor desempenho, quando comparado com os algoritmos nos quais ele foi baseado. Finalmente no capítulo 6 é feita a conclusão desta tese.

Capítulo 2

Máquinas de Vetores de Suporte

Dado um problema de classificação, a tarefa de uma máquina de aprendizado é de induzir um classificador que seja capaz de separar cada um dos padrões de acordo com a classe a que estes pertencem. Este classificador é uma função modelada de forma indutiva, através da informação contida em cada um dos padrões utilizados em seu treinamento. Seu principal objetivo é classificar corretamente os padrões utilizados no treinamento e também os que ainda não foram vistos. Ou seja, a função deverá ter uma boa capacidade de generalização. Considerando um espaço de classificação dividido por um de vários possíveis hiperplanos, deve-se escolher um que maximiza a generalização, mas com a precaução de não chegar a um super-ajuste[1] (o modelo gerado se torna muito especializado no conjunto de treinamento, obtendo baixo desempenho quando confrontado com novos padrões, ou seja, o algoritmo "memoriza" os dados do conjunto de treinamento). Mas nem sempre é possível obter um hiperplano que separe perfeitamente todos os dados presentes em um conjunto de dados. Então, o objetivo passa a ser um que minimiza a probabilidade de erros de classificação. Existem casos em que é praticamente impossível separar a maior parte dos dados através de um hiperplano. Nestes casos se faz necessário o mapeamento dos dados para uma alta dimensão, onde pode ser feita a separação dos dados de forma linear. A Teoria do Aprendizado Estatístico (TAE), desenvolvida por Vapnik (Vapnik, 1999), apresenta uma formulação matemática para o controle da capacidade

[1] do inglês *overfitting*.

de generalização de uma máquina de aprendizado genérica (Haykin, 1999), conceitos estes nos quais as Máquinas de Vetores de Suporte (SVM) se baseiam. A TAE formaliza a relação entre a capacidade de generalização do classificador, a complexidade das classes de funções que o algoritmo de aprendizado pode gerar e a minimização dos erros no treinamento.

Dados uma tarefa de aprendizado e um número finito de exemplos de treinamento, uma máquina de aprendizado muito complexa irá memorizar todos os dados de treinamento, sem ser capaz de realizar as classificações corretas nos exemplos ainda não vistos, enquanto uma máquina muito simples não será capaz nem mesmo de aprender o próprio conjunto de dados utilizado para seu treinamento. Um compromisso entre a capacidade da máquina de aprendizado e o desempenho no conjunto de treinamento deve ser obtido para que se obtenha uma boa generalização (Vapnik, 1999). Este é o papel da TAE.

Neste capítulo será apresentada a estrutura de uma máquina de aprendizado, a capacidade de generalização que ela deve ter para obter boas previsões, o que é risco esperado e dimensão VC. Também será apresentada a inequação que se deve obedecer para minimizar o risco estrutural. Além disso, serão apresentadas as características das SVM, como é encontrado o hiperplano de separação para os casos lineares e não lineares, e também como é feito o mapeamento dos dados em alta dimensão, para que seja possível efetuar a separação dos dados.

2.1 Máquina de Aprendizado

A figura 2.1 que segue representa a técnica de aprendizado por exemplos (Chekassky & Mulier, 2007).

A função do gerador de exemplos é produzir vetores randômicos $\mathbf{x}$, independente e identicamente distribuídos (iid) de acordo com uma função de distribuição de probabilidade (fdp) $P(\mathbf{x}, y)$. Estes vetores seguem para o sistema, onde uma saída y é produzida para todo vetor $\mathbf{x}$ de entrada de acordo com uma densidade condicional $p(y \mid \mathbf{x})$. Então sempre são gerados pares $(\mathbf{x}, y)$, onde $\mathbf{x}$ é um vetor $\in \mathbf{R}^n$ e y é um número $\in \mathbf{R}$. A máquina de aprendizado baseia-se em m pares $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$ para construir uma função $\hat{f} \in \mathfrak{F}$, onde $\mathfrak{F}$ é o

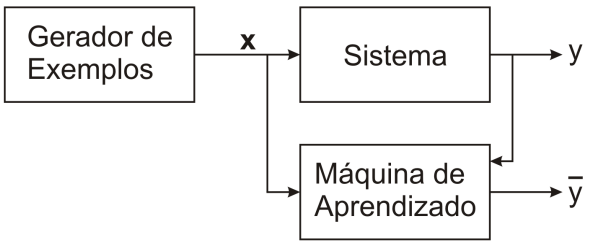


Figura 2.1: Um modelo de máquina de aprendizado através de exemplos
Fonte: (Vapnik, 1998)

conjunto de todas as funções (classificadores) que um determinado algoritmo de Aprendizado de Máquina (AM) pode gerar. Esta função possui a forma $\bar{y} = \hat{f}(\mathbf{x}, \alpha)$, onde o vetor $\mathbf{x}$ não pertence ao conjunto inicial de treinamento, mas é originado da mesma distribuição de probabilidade, e $\bar{y}$ é o valor associado a ele. Por fim, α é um conjunto de parâmetros ajustáveis da função. Logo após o aprendizado, o conjunto de parâmetros do estimador mantém-se constante, razão pela qual é comum substituir a notação $f(\mathbf{x}, \alpha)$ por $f(\mathbf{x})$. Então, a função da máquina de aprendizado é selecionar uma função (de um conjunto de funções que ela suporta) que melhor aproxima a resposta do sistema. Vapnik (Vapnik, 1998) considera que "O processo de aprendizado é um processo de escolher uma função apropriada de um dado conjunto de funções".

A equação 2.1 mostra um estimador que descreve a resposta de um sistema a um determinado conjunto de valores de entrada. X representa o domínio e Y a imagem (resposta) do sistema.

$$f : X \rightarrow Y \tag{2.1}$$

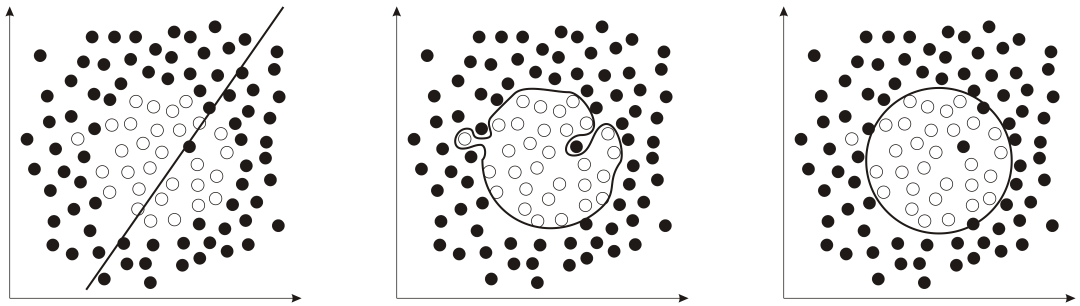
2.1.1 Capacidade de Generalização

Depois que a máquina de aprendizado é treinada, além de ter uma boa capacidade de identificar a classe a que pertence cada instância, também deve ser capaz de classificar corretamente outros conjuntos de objetos não vistos anteriormente. Esta é a definição de capacidade de generalização.

Conforme mostrado na seção 2.1, o processo de aprendizagem supervisionada consiste

em encontrar uma função, baseada nas instâncias utilizadas para o treinamento, que busque a maximização da capacidade de generalização do modelo gerado.

Caso existam ruídos no conjunto de treinamento, também chamados de *outliers* (exemplos muito distintos dos demais presentes no conjunto de dados), a geração do modelo pode ser prejudicada, levando ao aparecimento de um super-ajuste. O oposto de super-ajuste é sub-ajuste[2] (o classificador não é capaz de se ajustar até mesmo às instâncias utilizadas para treinamento). Veja uma ilustração destas situações nas figuras 2.2(b) e 2.2(a). Um modelo que representa um compromisso entre as duas funções citadas, utilizando um modelo de complexidade intermediária, com a qual seleciona a maioria dos pontos de forma correta, está ilustrado na figura 2.2(c).



(a) Sub-ajuste (b) Super-ajuste (c) Ajuste correto

Figura 2.2: Ilustração de situações onde ocorre sub-ajuste e super-ajuste

Logo, deve-se evitar a geração de um modelo muito específico para o conjunto de treinamento, pois os *outliers* podem ser levados em conta e, com isso, gerar o problema de super-ajuste. Para restringir a geração de um modelo (função) muito específico, deve-se restringir a complexidade da classe de funções $\mathfrak{F}$ da qual é escolhida a função $\hat{f}$ (Müller, Mika, Rätsch, Tsuda, & Schölkopf, 2001). Para controlar esta complexidade usa-se a teoria VC e o princípio de Minimização do Risco Estrutural (MREs) (Vapnik, 1998), que serão vistos adiante (seção 2.3 e 2.4, respectivamente).

[2]do inglês *underfitting*.

2.2 Risco Esperado

A melhor função f para efetuar a classificação correta dos dados é aquela que obtém o menor risco esperado (também chamado de erro esperado). O risco esperado é a discrepância entre a saída produzida pelo sistema e a saída esperada da máquina de aprendizado, para um dado valor de entrada $\mathbf{x}$. Para minimizar este erro, utiliza-se a seguinte função (Müller et al., 2001):

$$R(f) = \int L(f(\mathbf{x}), y) dP(\mathbf{x}, y) \tag{2.2}$$

L representa o resultado da escolha da função de custo. Um tipo de função de custo usada em classificação de dados é definida como

$$L(f(\mathbf{x}), y) = \begin{cases} 0 & \text{se } y = f(\mathbf{x}) \\ \\ 1 & \text{se } y \neq f(\mathbf{x}) \end{cases} \tag{2.3}$$

Esta função é também chamada de perda 0/1. Sempre que a função de custo assume apenas dois valores, ela recebe o nome de função indicadora (Vapnik, 1998). Na equação 2.2, a distribuição de probabilidade $P(\mathbf{x}, y)$ é desconhecida. Portanto, não é possível minimizar o risco diretamente. Logo, deve-se encontrar uma função que chegue o mais próximo da que trata do risco esperado, utilizando apenas as informações que estão disponíveis. No caso, são os exemplos de treinamento e as propriedades das classes de funções $\mathfrak{F}$ da qual $\hat{f}$ foi escolhida. Para isso é necessário utilizar o princípio indutivo (Müller et al., 2001). Um princípio simples consiste em aproximar o mínimo da equação 2.2 com o mínimo do risco empírico (equação 2.4)

$$R_{emp}(f) = \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}_i), y_i) \tag{2.4}$$

onde m é a quantidade de padrões (instâncias). Assintoticamente, baseado na Lei dos Grandes Números, com ($m \to \infty$), é possível estabelecer condições que fazem com que a máquina de aprendizado tenha condições de que o risco empírico convirja para o risco esperado (Schölkopf

& Smola, 2002). Mas este método pode não ser eficiente por si só. Considere o caso em que o sistema memoriza todos os dados de treinamento e gera saídas randômicas para os outros dados. Neste caso o risco empírico é nulo e o risco esperado é $0,5$ (Smola, Bartlett, Schölkopf, & Schuurmans, 2000).

Outra situação é quando o conjunto de dados de treinamento é pequeno, não sendo possível obter um pequeno erro de generalização na equação 2.4, gerando um super-ajuste.

A solução é restringir a complexidade da classe de funções $\mathfrak{F}$, da qual $\hat{f}$ é escolhida. A idéia é que, de acordo com a Navalha de Occam, uma função "simples" que pode definir a maioria dos dados é preferível a uma complexa (Müller et al., 2001). A forma de controlar a complexidade da classe da função é dada pela Dimensão VC e pela Minimização do Risco Estrutural, a serem tratados nas seções 2.3 e 2.4, respectivamente.

2.3 Dimensão VC

A dimensão VC[3] é definida como a máxima quantidade de exemplos de treinamento (m) que podem ser classificados, sem erros, por um conjunto de funções (Burges, 1998). Caso m não exista, a dimensão VC é definida como infinita (∞) (Schölkopf & Smola, 2002). A dimensão VC é utilizada para medir a capacidade do conjunto de funções $\mathfrak{F}$ do qual $\hat{f}$ é escolhida.

Considere um problema de classificação com duas classes e uma função de custo dada pela equação 2.3. Segundo Vapnik (1998), a dimensão VC de um conjunto de funções $\mathfrak{F}$, é igual ao maior número h de vetores $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ que podem ser separados em duas classes diferentes em todas as 2^h possíveis formas de combinações.

A dimensão VC mede a capacidade de classificação de um conjunto de funções (Haykin, 1999). Veja que esta definição não assume nada a respeito da distribuição de probabilidade $P(\mathbf{x}, y)$ e considera somente uma amostragem de tamanho m e não uma média sobre todas as possíveis amostragens de tamanho m.

[3] VC em homenagem a Vapnik e Chervonenkis.

Imagine todas as possíveis dicotomias[4] em um conjunto de m pontos que podem ser induzidas por um conjunto de funções $\mathfrak{F}$. A maior quantidade de dicotomia é chamada de função de crescimento (Haykin, 1999).

Para ilustrar este conceito, veja a figura 2.3. Nela são mostrados três pontos organizados de oito (2^3) formas diferentes. Estes três pontos são classificados em duas classes (bola cheia e bola vazia). Uma linha (função de custo) pode separar as classes em todos os oito casos. Portanto, a dimensão VC da figura 2.3 é três. O conjunto de funções de custo lineares em um espaço n dimensional, possui uma dimensão VC igual a n+1.

Outra forma de explicar: No exemplo, existem oito (2^3) formas de atribuir três pontos a duas classes. Para os pontos apresentados em $\mathbf{R}^2$, todas as oito situações podem ser separadas com o hiperplano. Em outras palavras, a função pode separar três pontos. Este mesmo raciocínio não funcionará com quatro pontos, não importando a localização deles. Portanto, a dimensão VC da classe dos hiperplanos de separação ($\mathfrak{F}$) em $\mathbf{R}^2$ é três (Schölkopf & Smola, 2002).

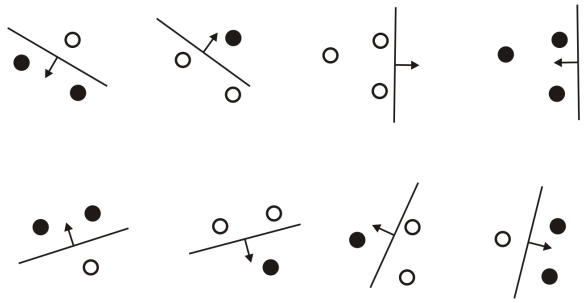


Figura 2.3: Separações de três pontos em $\mathbf{R}^2$, por meio de retas orientadas
Fonte: (Burges, 1998)

2.4 Minimização do Risco Estrutural (MREs)

O princípio de minimização do risco estrutural é baseado no fato de que a taxa de erro de uma máquina de aprendizado nos dados de teste (taxa de erro de generalização) está limitada

[4]Dicotomia se refere a uma função de classificação binária.

pela soma da taxa dos erros de treinamento e um termo que depende da dimensão VC, definida a seguir (inequação 2.5).

Um importante limite, que vem da TAE, é que para um η, tal que $0 \leq \eta \leq 1$, com probabilidade $1 - \eta$, o seguinte limite é válido

$$R(f) \leq R_{emp}(f) + \sqrt{\frac{h(ln(\frac{2m}{h}) + 1) - ln(\frac{\eta}{4})}{m}} \tag{2.5}$$

onde h é a dimensão VC. O lado direito da desigualdade 2.5 é chamado de "*risk bound*". A parcela de raiz na soma é chamada de "*VC confidence*" (Burges, 1998). O termo "*VC confidence*" também é chamado de termo de complexidade (Müller et al., 2001).

O objetivo é minimizar o risco esperado (também chamado de erro de generalização), que pode ser feito obtendo um pequeno erro de treinamento (risco empírico) enquanto mantém a classe de funções $\mathfrak{F}$ a menor possível. Na análise da desigualdade 2.5, surgem dois extremos: 1) uma estrutura pequena (S_1), leva a extinção do termo da raiz quadrada ("*VC Confidence*"), mas a um alto R_{emp}, enquanto 2) uma alta estrutura (S_k) leva a extinção de R_{emp}, mas com um alto *"VC Confidence"*. A melhor escolha geralmente é um meio termo entre R_{emp} e o termo da raiz quadrada, levando a uma função que descreve os dados de uma forma interessante e um pequeno risco em obter tal função (Müller et al., 2001).

Para pequenos conjuntos de dados de treinamento[5], o processo de minimização do risco esperado não pode ser obtido simplesmente minimizando o risco empírico (Müller et al., 2001). Deve-se minimizar ambos os termos (o risco empírico e "*VC confidence*"). Esse princípio é chamado de Minimização do Risco Estrutural (MREs).

Na desigualdade 2.5, o termo denominado "*VC confidence*" se refere à classe de funções $\mathfrak{F}$ e o risco empírico se refere a um classificador particular f (Müller et al., 2001). Para minimizar ambas as parcelas, divide-se $\mathfrak{F}$ em estruturas $S_m = \{L(f(\mathbf{x}), y)\}$, de tal forma que $S_1 \subset S_2 \subset \ldots \subset S_m \ldots$, ou seja, as estruturas estão organizadas de forma aninhada (figura 2.4). A

[5]Um conjunto de dados é considerado pequeno se a relação m/h (proporção entre o número de padrões de treinamento e a dimensão VC das funções de uma máquina de aprendizado) for menor que 20, ou seja, $m/h < 20$ (Vapnik, 1999).

dimensão VC h de cada conjunto S_k é finita. Portanto $h_1 \leq h_2 \ldots \leq h_m \ldots$. A MREs escolhe uma classe de função $\mathfrak{F}$ de tal forma que o limite superior do erro de generalização é minimizado. Isto é calculado com base na desigualdade 2.5.

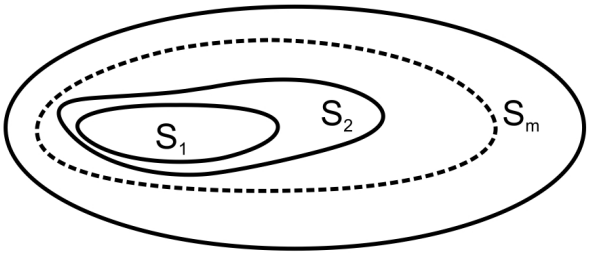


Figura 2.4: Estruturas aninhadas para diferentes modelos de complexidades
Fonte: (Vapnik, 1999)

O princípio MREs escolhe a função $L(f(\mathbf{x}), y)$ no subconjunto S_k para o qual o lado direito da desigualdade 2.5 seja mínimo. Isto pode ser visualizado na figura 2.5.

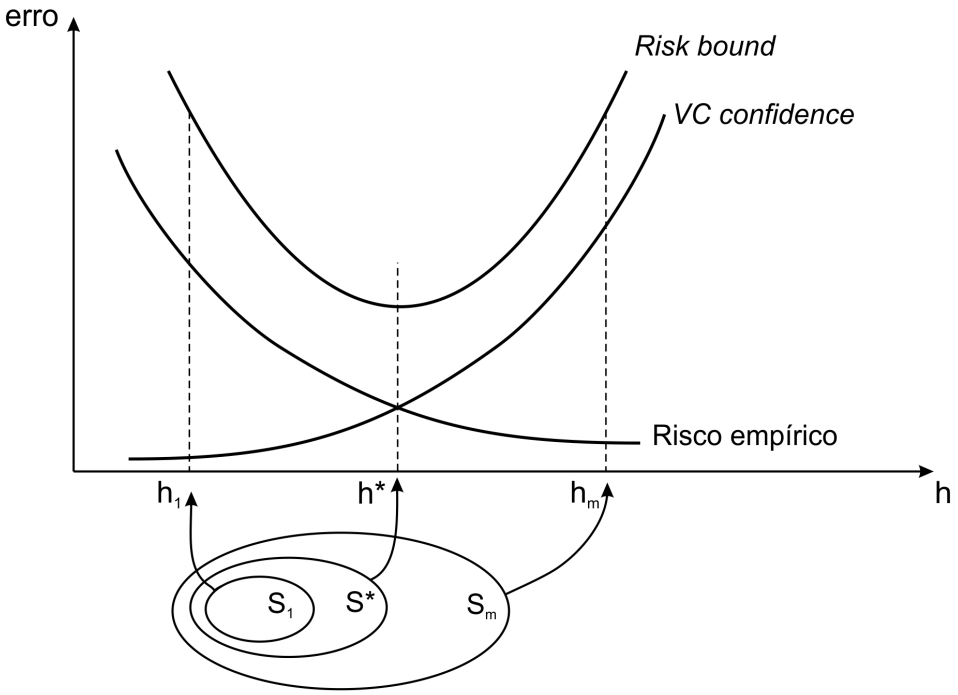


Figura 2.5: Princípio de minimização do risco estrutural
Fonte: (Vapnik, 1999)

Portanto, com uma função de custo L e uma função f, o processo de construção de uma máquina de aprendizado é iniciado com a seleção da estrutura. Por exemplo, caso a função

f seja uma função polinomial, o seu grau pode ser usado como um parâmetro para selecionar estruturas diferentes.

2.5 Características das SVM

Dentre os algoritmos de aprendizado de máquina, um que tem se destacado são as SVM, baseado na TAE (Vapnik, 1999). As SVM foram originalmente projetadas para problemas bi-classe. Para a solução de problemas multiclasses (Hsu & Lin, 2002), três estratégias utilizadas são os métodos "um-contra-todos" (Bottou et al., 1994), "um-contra-um" (J. H. Friedman, 1996) e DAGSVM (Platt, Cristianini, & Shawe-Taylor, 1999).

A formulação mais simples de SVM é aquela que lida com problemas linearmente separáveis (Boser, Guyon, & Vapnik, 1992). As SVM trabalham também com problemas não lineares mapeando o conjunto de treinamento de seu espaço original para um novo espaço de maior dimensão, chamado de espaço de características (Hearst, 1998). Esta operação pode ser justificada invocando-se o célebre Teorema de Cover, que afirma que padrões não-linearmente separáveis pertencentes a um dado espaço de características são, com alta probabilidade, linearmente separáveis num espaço de características transformado, desde que a transformação seja não-linear e a dimensão do espaço transformado seja alta o suficiente (Haykin, 1999).

A tarefa de classificação usualmente envolve dados de treinamento e teste que consistem de algumas instâncias de dados. Cada instância no conjunto de treinamento contém um valor objetivo (rótulo da classe) e vários atributos (características). O objetivo da SVM é produzir um modelo que prediz o valor objetivo das instâncias no conjunto de teste, que possuem somente os atributos.

Algumas características que motivam o uso de SVM são:

- Possui uma boa capacidade de generalização, ou seja, uma vez que as SVM são apresentadas ao conjunto de treinamento, são capazes de aprender a regra que pode classificar corretamente outro conjunto de objetos (Smola et al., 2000);
- O uso das chamadas funções *kernel* torna possível a construção de um hiperplano

de separação ótimo no espaço de características (também chamado de espaço de alta dimensão), sem ter que considerar o próprio espaço de características de forma explícita (Haykin, 1999). Isto torna o algoritmo eficiente do ponto de vista computacional (Burges, 1998);

- É eficiente em termos de velocidade e complexidade. É equivalente a procurar por um mínimo de uma função convexa, isto é, sem mínimos locais. Isto elimina muitos dos problemas que ocorrem quando são utilizadas outras metodologias como redes neurais e árvores de decisão (Bennett & Campbell, 2000);
- Possui uma base teórica bem estabelecida na matemática e estatística. Isso faz com que sejam robustas quando lidam com objetos de grandes dimensões, como imagens (Smola et al., 2000);
- O método é relativamente simples de utilizar. Não há necessidade de um especialista para aplicar um software de SVM a novos problemas;
- As SVM foram aplicadas com sucesso em várias áreas (Fan, Zhang, & Xia, 2008; Li, Fu, & Yang, 2008; Osuna, Freund, & Girosi, 1997b; Tao, Tang, & Li, 2008; Zien et al., 2000). Isto prova que são robustas a ruídos e têm um bom desempenho em várias tarefas.

2.6 Hiperplano de Separação para Classes Linearmente Separáveis

O problema de classificação pode ser restrito à análise de um problema binário sem perda de generalidade (Gunn, 1998). Neste problema o objetivo é separar duas classes por uma função que é induzida dos exemplos disponíveis (instâncias de treinamento). Esta função deve ter um bom desempenho na separação de exemplos não vistos, isto é, ela deve ter uma boa generalização. Considere a figura 2.6 onde existem vários hiperplanos que podem separar os dados. Dentre estes hiperplanos, existe somente um que maximiza a margem (distância

entre o hiperplano e o ponto mais próximo de cada classe). Ele recebe o nome de hiperplano de separação ótimo. Intuitivamente, é esperado que este hiperplano tenha uma generalização melhor que os outros possíveis.

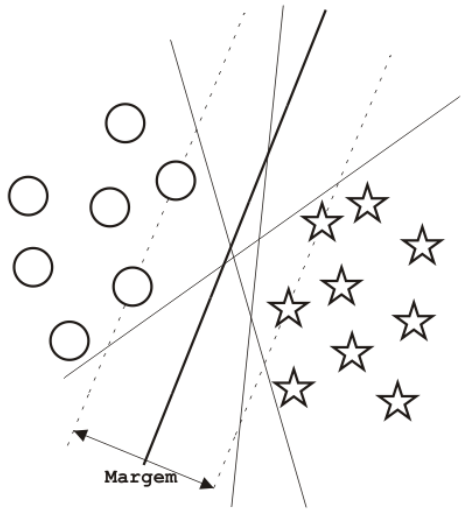


Figura 2.6: Hiperplano de separação ótimo
Fonte: Adaptado de (Gunn, 1998)

Um hiperplano de separação é uma função linear capaz de separar os dados de treinamento sem erros[6]. Suponha que os dados de treinamento compostos de m exemplos $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, $\mathbf{x} \in \mathbf{R}^n$, $y \in \{+1, -1\}$, podem ser separados pelo hiperplano

$$D(\mathbf{x}) = (\mathbf{w}.\mathbf{x}) + b \tag{2.6}$$

onde $\mathbf{x}$ é o vetor de entrada, $\mathbf{w}$ é um vetor de pesos ajustáveis normal ao hiperplano, $|b|/\|\mathbf{w}\|$ é a distância perpendicular do hiperplano à origem (figura 2.7), e $\|\mathbf{w}\|$ é a norma euclidiana de $\mathbf{w}$ (Burges, 1998; Haykin, 1999).

A menor distância entre o hiperplano de separação para os dados mais próximos é denotada por $\triangle$. Um hiperplano de separação com margem $2\triangle$ satisfaz as seguintes restrições (inequação 2.7), resumidas na inequação 2.8

[6]Aqui será considerado que os dados são linearmente separáveis, para facilitar a abordagem utilizando SVM. Adiante, esta restrição será desconsiderada.

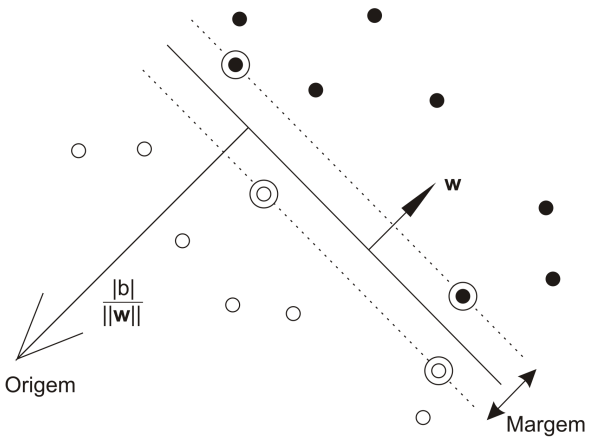


Figura 2.7: Hiperplano de separação
Fonte: Adaptado de (Burges, 1998)

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +\triangle \quad \text{se} \quad y_i = +1,$$
$$i = 1, \ldots, m, \tag{2.7}$$
$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -\triangle \quad \text{se} \quad y_i = -1,$$

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq \triangle, \qquad i = 1, \ldots, m. \tag{2.8}$$

A seguir descreve-se um hiperplano ótimo formalmente e também como determiná-lo a partir dos dados de treinamento. A distância entre o hiperplano $(\mathbf{w} \cdot \mathbf{x}) + b = 0$ e um exemplo $\mathbf{x}'$ é $|(\mathbf{w} \cdot \mathbf{x}') + b|/\|\mathbf{w}\|$. No caso particular de $\mathbf{x} = 0$, o hiperplano ótimo é dado por $b/\|\mathbf{w}\|$. Se $b > 0$, o hiperplano está no lado positivo em relação à origem. Para $b < 0$, o hiperplano está no lado negativo. Caso $b = 0$, o hiperplano ótimo passa pela origem (Haykin, 1999). Para uma margem de $2\triangle$, todos os padrões de treinamento estão a uma distância de no mínimo $\triangle$ da região de decisão e portanto obedecem à desigualdade

$$\frac{y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b]}{\|\mathbf{w}\|} \geq \triangle, \qquad i = 1, \ldots, m, \tag{2.9}$$

onde $y_i \in \{-1, 1\}$.

Analisando a inequação 2.9, pode-se concluir que maximizando a margem $\triangle$ é equi-

valente a minimizar $\|\mathbf{w}\|$ (Chekassky & Mulier, 2007). Redimensionando os parâmetros $\mathbf{w}$ e b através da fixação da escala $\triangle\|\mathbf{w}\| = 1$ leva à representação canônica do hiperplano de separação.

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1, \qquad i = 1, \ldots, m. \tag{2.10}$$

Os pontos que ficam sobre a borda da margem, ou, de forma equivalente, os pontos para os quais a inequação 2.10 é uma igualdade, são chamados de vetores de suporte (figura 2.8). Como os vetores de suporte são pontos próximos à superfície de decisão, conceitualmente eles são as amostras mais difíceis de classificar e portanto definem a localização da região de decisão (Haykin, 1999; Duda, Hart, & Stork, 2001). A região de decisão de um hiperplano ótimo é definida pelos pontos $\mathbf{x}$ para os quais $D(\mathbf{x}) = 0$. A distância entre o hiperplano e qualquer exemplo $\mathbf{x}'$ é:

$$d(\mathbf{w}, b; \mathbf{x}') = \frac{|D(\mathbf{x}')|}{\|\mathbf{w}\|} \tag{2.11}$$

A distância entre o vetor de suporte (o qual define a margem) e o hiperplano ótimo é $1/\|\mathbf{w}\|$. Portanto, a margem entre duas classes distintas, medida perpendicularmente ao hiperplano ótimo é $2/\|\mathbf{w}\|$ (Schölkopf, 1997). O hiperplano ótimo é obtido maximizando essa margem.

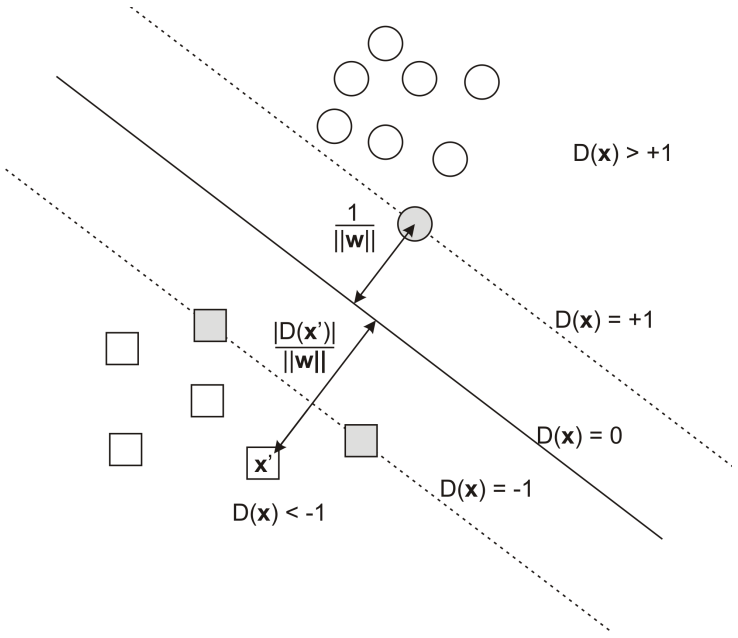


Figura 2.8: Região de decisão de um hiperplano ótimo
Fonte: (Chekassky & Mulier, 2007)

Então, o hiperplano que separa os dados da melhor forma é aquele que minimiza:

$$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.12}$$

Ele é independente de b desde que a equação 2.10 seja satisfeita. Caso o valor de b seja alterado, o hiperplano irá mover-se na sua direção normal. Conseqüentemente, a margem continuará inalterada, mas o hiperplano já não será mais ótimo, pois estará mais próximo de uma classe que de outra. Minimizar a equação 2.12 é equivalente a implementar o princípio MREs(Gunn, 1998). Isso será ilustrado, estabelecendo um limite A para a norma de $\mathbf{w}$

$$\|\mathbf{w}\| < A \tag{2.13}$$

Então, das equações 2.10 e 2.11, resulta em:

$$d(\mathbf{w}, b; \mathbf{x}') \geq \frac{1}{A} \tag{2.14}$$

Conseqüentemente, os hiperplanos não podem estar mais próximos que $\frac{1}{A}$ para qualquer amostra de treinamento e, intuitivamente pode ser visto na figura 2.9 como esta restrição reduz a quantidade de possíveis hiperplanos, e portanto elimina aqueles que tendem a generalizar pior.

De acordo com Vapnik (Vapnik, 1999), a dimensãoVC (h) no conjunto de hiperplanos canônicos em um espaço de entrada de dimensão n é limitada por,

$$h \leq min[R^2A^2, n] + 1, \tag{2.15}$$

onde R é o raio da menor circunferência que contém todos os dados de entrada. Assim, minimizar a equação 2.12 é equivalente a minimizar o limite superior da dimensão VC. A solução para o problema de otimização da equação 2.12, considerando as restrições da equação 2.10, gera o

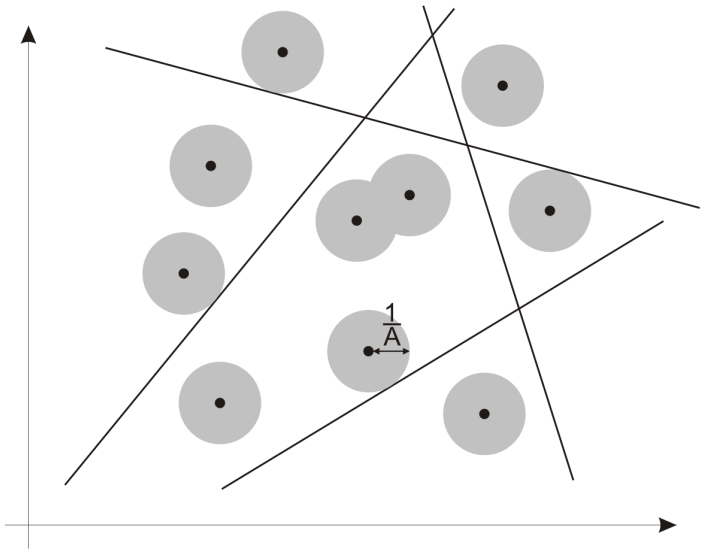


Figura 2.9: Limitando os hiperplanos canônicos a uma distância mínima de $\frac{1}{A}$
Fonte: (Gunn, 1998)

seguinte problema de otimização quadrático (Cristianini & Shawe-Taylor, 2000)

$$\begin{aligned} &\underset{w,b}{\text{minimizar}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{sujeito a} \qquad y_i\big[(\mathbf{w}\cdot\mathbf{x}_i)+b\big] \geq 1, \qquad i = 1,\ldots,m. \end{aligned} \tag{2.16}$$

o fator $1/2$ em 2.16 é incluído por conveniência (Bishop, 2006). A restrição é para assegurar que não haja dados de treinamento entre as margem de separação das classes.

Problemas com uma função objetivo e uma restrição de desigualdade são chamados de problemas de restrição de otimização e são solucionados utilizando os multiplicadores de Lagrange λ_i (Schölkopf & Smola, 2002).

$$\begin{aligned} L_P(\mathbf{w},b,\mathbf{\Lambda}) = f(\mathbf{x}) + \sum_{i=0}^{m}\lambda_i g_i(\mathbf{x}) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\lambda_i(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1) \\ &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\lambda_i y_i(\mathbf{w}\cdot\mathbf{x}_i+b) + \sum_{i=1}^{m}\lambda_i y_i \\ &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\lambda_i y_i \mathbf{w}\cdot\mathbf{x}_i - \sum_{i=1}^{m}\lambda_i y_i b + \sum_{i=1}^{m}\lambda_i \end{aligned} \tag{2.17}$$

Onde $\mathbf{\Lambda} = (\lambda_1,\lambda_2,\ldots,\lambda_m)$ é o conjunto dos multiplicadores de Lagrange dos padrões de trei-

namento, com $\lambda_i \geq 0$, e a letra P em L_P indica a formulação primal do problema. A função L_P deve ser minimizada em relação a $\mathbf{w}$ e b, e maximizada em relação a λ_i, sujeita a restrição $\lambda_i \geq 0$ (Ivanciuc, 2007). O sinal negativo em frente ao multiplicador de Lagrange é devido ao fato da função estar sendo minimizada em relação às variáveis $\mathbf{w}$ e b e maximizadas em relação a $\mathbf{\Lambda}$ (Bishop, 2006).

De acordo com o teorema de Kuhn-Tucker (Bertsekas, 1999), as soluções ótimas $\mathbf{w}$, b e $\mathbf{\Lambda}$ devem satisfazer as seguintes condições:

$$\begin{aligned} \frac{\partial}{\partial b} L_P(\mathbf{w}, b, \mathbf{\Lambda}) = 0, \\ \frac{\partial}{\partial w} L_P(\mathbf{w}, b, \mathbf{\Lambda}) = 0, \end{aligned} \tag{2.18}$$

Resolvendo as derivadas parciais, tem-se como resultado as seguintes propriedades do hiperplano ótimo (Chekassky & Mulier, 2007):

$$\sum_{i=1}^{m} \lambda_i y_i = 0, \qquad \lambda_i \geq 0, \qquad i = 1, \ldots, m. \tag{2.19}$$

$$\mathbf{w} = \sum_{i=1}^{m} \lambda_i y_i \mathbf{x}_i, \qquad \lambda_i \geq 0, \qquad i = 1, \ldots, m. \tag{2.20}$$

Além disso, de acordo com o teorema de Kuhn-Tucker, qualquer parâmetro λ_i é diferente de zero somente se os exemplos $(\mathbf{x}_i, y_i)$ correspondentes satisfazem a restrição 2.10 com igualdade. Isto é descrito pela condição

$$\lambda_i [y_i ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1] = 0, \qquad i = 1, \ldots, m. \tag{2.21}$$

Os valores de $\mathbf{x}_i$ para os quais $\lambda_i > 0$ são chamados de vetores de suporte (Schölkopf, 1997).

Substituindo as equações 2.19 e 2.20 em 2.17, obtém-se o seguinte problema de oti-

mização, chamado de problema dual:

$$\begin{aligned} L_D(\mathbf{\Lambda}) = \quad & \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{sujeito a} \quad & \lambda_i \geq 0, \qquad i = 1, \ldots, m \\ \text{e} \quad & \sum_{i=1}^{m} \lambda_i y_i = 0 \end{aligned} \tag{2.22}$$

Ambas as formas primal L_P e dual L_D são derivadas da mesma função objetivo, mas com diferentes restrições, e a solução é encontrada pela minimização em L_P e maximização em L_D. Esta última forma é moldada em termos dos dados de treinamento. Portanto, para ser maximizada depende somente dos padrões de entrada, na forma do produto escalar $\{\mathbf{x}_i \cdot \mathbf{x}_j\}_{i,j}^{m}$ (Haykin, 1999).

Depois de determinados os multiplicadores de Lagrange $(\mathbf{\Lambda})$, pode-se calcular o vetor $\mathbf{w}$. Para isto, basta substituir a equação 2.20 em 2.6. Esta operação leva ao hiperplano

$$D(\mathbf{x}) = \sum_{i=1}^{m} \lambda_i y_i (\mathbf{x} \cdot \mathbf{x}_i) + b \tag{2.23}$$

Para calcular o parâmetro b, usa-se a condição para um padrão de entrada ser um vetor de suporte. Dado qualquer vetor de suporte $(\mathbf{x}_s, y_s)$, este satisfaz (Chekassky & Mulier, 2007)

$$y_s[(\mathbf{w} \cdot \mathbf{x}_s) + b] = 1 \tag{2.24}$$

Substituindo a expressão 2.20 em 2.24 tem-se:

$$b = y_s - \sum_{i=1}^{m} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}_s) \tag{2.25}$$

Após o processo de treinamento de um classificador de dados linearmente separáveis de uma SVM, o classificador deve ser capaz de predizer a classe para novos padrões de entrada, diferente dos padrões utilizados no treinamento. A classe do novo padrão $\mathbf{x}_s$ é determinada pela equação 2.26.

$$classe(\mathbf{x}_s) = \begin{cases} +1 & \text{se} \quad \mathbf{w} \cdot \mathbf{x_s} + b > 0 \\ -1 & \text{se} \quad \mathbf{w} \cdot \mathbf{x}_s + b < 0 \end{cases} \tag{2.26}$$

Então, a classificação de novos padrões depende somente do sinal da expressão $\mathbf{w} \cdot \mathbf{x} + b$. A equação 2.20 permite classificar um novo padrão sem utilizar o vetor **w** explicitamente. Serão aproveitados somente os padrões utilizados para treinamento e seus correspondentes valores λ_i

$$classe(\mathbf{x}_s) = sgn(\sum_{i=1}^{m} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{x}_s) + b) \tag{2.27}$$

O uso da equação 2.27 possui uma importante vantagem em relação a 2.26: para classificar um novo padrão $\mathbf{x}_s$, é necessário somente calcular o produto escalar entre $\mathbf{x}_s$ e todo vetor de suporte. Já na expressão 2.26, é necessário calcular o valor do vetor **w** explicitamente. Isto resulta em uma economia de tempo computacional, pois a quantidade de vetores de suporte é menor quando comparada ao total de padrões utilizados no treinamento (Ivanciuc, 2007).

Uma função linear é muito limitada na tarefa de separação de dados, uma vez que, normalmente, os dados não são linearmente separáveis. Com isso, não será possível conseguir a separação dos dados de treinamento, levando a um sub-ajuste. Felizmente, existe uma maneira de ter modelos lineares e um ótimo conjunto de funções não lineares, utilizando *kernels*[7] (Müller et al., 2001).

2.7 Hiperplano de Separação para Classes Não Linearmente Separáveis

Nem sempre é possível construir um hiperplano de separação sem deparar-se com erros de classificação. No entanto, é possível encontrar um hiperplano ótimo que minimiza a probabilidade de erros de classificação.

A margem de separação entre classes é dita suave se um dado $(\mathbf{x}_i, y_i)$ viola a condição

[7] será dado mais detalhes sobre o funcionamento de *kernels* na seção 2.8.

(desigualdade 2.10):

$$y_i\big[(\mathbf{w}\cdot\mathbf{x}_i)+b\big]\geq 1, \qquad i=1,\ldots,m.$$

Esta violação pode ocorrer de duas formas (Haykin, 1999):

- O dado $(\mathbf{x}_i, y_i)$ se encontra dentro da região de separação (margem), mas no lado correto da superfície de decisão, conforme ilustrado na figura 2.10(a);
- O dado $(\mathbf{x}_i, y_i)$ se encontra no lado incorreto da superfície de decisão (figura 2.10(b)).

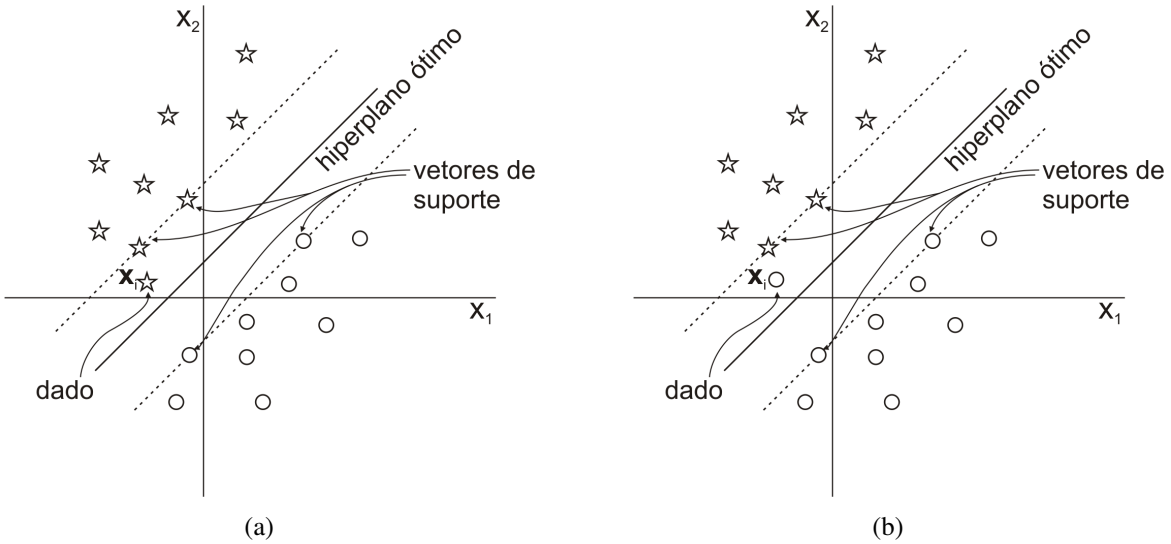


Figura 2.10: (a) Dado $\mathbf{x}_i$ (pertencente a classe ☆) dentro da região de separação, mas no lado correto da superfície de decisão. (b) Dado $\mathbf{x}_i$ (pertencente a classe ○) dentro da região de separação, mas no lado correto incorreto da superfície de decisão.
Fonte: Adaptado de (Haykin, 1999)

Para tratar estes dados não separáveis linearmente, um novo conjunto de valores escalares não negativos $\{\xi_i\}_{i=1}^{m}$ é introduzido na definição do hiperplano de separação (Cortes & Vapnik, 1995).

$$y_i\big[(\mathbf{w}\cdot\mathbf{x}_i)+b\big]\geq 1-\xi_i, \quad \xi_i\geq 0, \quad i=1,\ldots,m. \tag{2.28}$$

A variável ξ_i é chamada de variável de relaxamento. Ela mede o desvio de um dado da borda da margem de separação. Para $0\leq\xi_i\leq 1$, o dado se encontra na margem de separação, mas no lado correto do hiperplano. Para $\xi_i > 1$, ele se encontra no lado incorreto do hiperplano

de separação. Na figura 2.11 três dados $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ não podem ser separados, pois estão dentro da margem de separação. Os dados $\mathbf{x}_2, \mathbf{x}_3$ são classificados incorretamente, pois eles estão no lado incorreto do hiperplano de classificação, enquanto que $\mathbf{x}_1$ ilustra o caso em que não é separável, mas classificado corretamente.

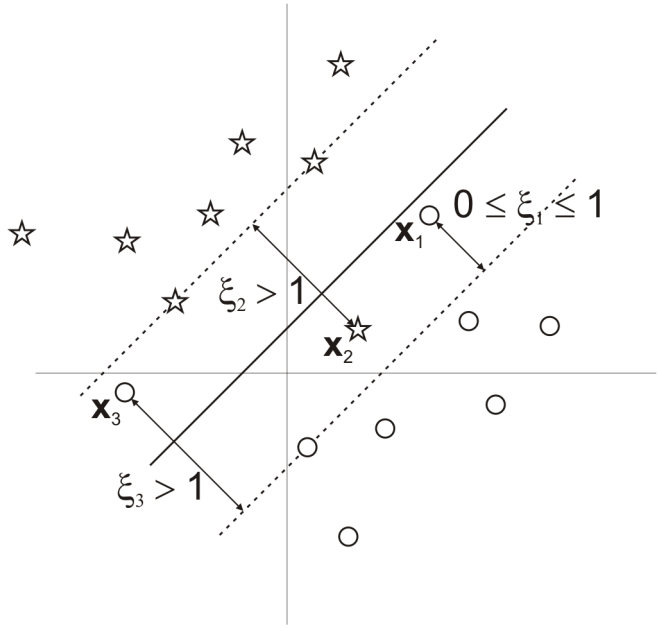


Figura 2.11: Variáveis de relaxamento para casos em que os dados não são totalmente separáveis

A tarefa de encontrar um hiperplano de separação ótimo é mais difícil quando as variáveis de relaxamento são utilizadas. Isto porque o hiperplano ótimo deve ser um compromisso entre duas condições opostas. Por um lado, uma boa SVM corresponde a um hiperplano $(\mathbf{w}, b)$ com a maior margem possível para que seja garantida uma boa predição, que se traduz em minimizar $\frac{1}{2}\|\mathbf{w}\|^2$. Por outro lado, o hiperplano ótimo deve minimizar a quantidade de erros de classificação e também minimizar o erro das classes classificadas incorretamente, que se traduz em minimizar o número de variáveis de relaxamento positivas e ao mesmo tempo minimizar o valor de cada variável de relaxamento. Esta última condição tende a reduzir a margem do hiperplano de separação, o que contradiz com a primeira condição. Uma forma de combinar estas duas condições e atribuir uma penalidade para os erros de classificação é alterar o problema de

otimização 2.16 para (Ivanciuc, 2007).

$$\begin{aligned} &\underset{\xi,w,b}{\text{minimizar}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ &\text{sujeito a} \quad y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1-\xi_i, \qquad i = 1,\ldots,m \\ &\text{e} \quad \xi_i \geq 0, \qquad i = 1,\ldots,m \end{aligned} \tag{2.29}$$

onde C é um parâmetro a ser ajustado pelo usuário, e pode tanto aumentar quanto reduzir a penalidade para os erros de classificação. Um alto valor para C atribui uma penalidade maior para uma classificação incorreta, reduzindo a quantidade de erros. Um baixo valor para C maximiza a margem de modo que o hiperplano é menos sensível a erros no conjunto de aprendizado.

Para resolver o problema 2.29, será utilizada novamente a abordagem que utiliza os multiplicadores de Lagrange. Considere os multiplicadores de Lagrange $\mathbf{\Lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ para cada restrição $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1-\xi_i$ e os multiplicadores $\mathbf{M} = (\mu_1, \mu_2, \ldots, \mu_m)$ para cada restrição $\xi_i \geq 0$. Com estas notações, agora é possível definir a forma primal do problema

$$L_P(\mathbf{w}, b, \mathbf{\Lambda}, \mathbf{M}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\lambda_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{m}\mu_i\xi_i \tag{2.30}$$

As condições de Karuch-Kuhn-Tucker (KKT) (Arora, 2004) para o problema 2.30 são:

$$\frac{\partial L_P(\mathbf{w}, b, \mathbf{\Lambda}, \mathbf{M})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m}\lambda_i y_i \mathbf{x}_i = 0 \tag{2.31}$$

$$\frac{\partial L_P(\mathbf{w}, b, \mathbf{\Lambda}, \mathbf{M})}{\partial b} = \sum_{i=1}^{m}\lambda_i y_i = 0 \tag{2.32}$$

$$\frac{\partial L_P(\mathbf{w}, b, \mathbf{\Lambda}, \mathbf{M})}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \tag{2.33}$$

$$\lambda_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] = 0, \qquad i = 1,\ldots,m. \tag{2.34}$$

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] \geq 0, \qquad i = 1,\ldots,m. \tag{2.35}$$

$$\begin{aligned}
&\xi_i \geq 0, && i = 1, \ldots, m \\
&\lambda_i \geq 0, && i = 1, \ldots, m \\
&\mu_i \geq 0, && i = 1, \ldots, m \\
&\mu_i \xi_i = 0, && i = 1, \ldots, m
\end{aligned} \tag{2.36}$$

Substituindo as equações 2.31 e 2.32 em 2.30, obtém-se o problema na forma dual

$$\begin{aligned}
&L_D(\boldsymbol{\Lambda}) = \quad \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\
&\text{sujeito a} \quad 0 \leq \lambda_i \leq C, \qquad i = 1, \ldots, m \\
&\text{e} \qquad \sum_{i=1}^{m} \lambda_i y_i = 0
\end{aligned} \tag{2.37}$$

A solução para o vetor **w** é obtida pela equação 2.31, e o valor de b pode ser calculado pela equação 2.34 juntamente com a que segue

$$(C - \lambda_i)\xi_i = 0 \tag{2.38}$$

A equação 2.38 é o resultado da substituição de 2.33 em 2.36. Destas duas últimas equações, tem-se que $\xi_i = 0$ se $\lambda_i < C$. Portanto, b só pode ser calculado para aqueles dados que tenham $0 \leq \lambda_i < C$ (Haykin, 1999).

Na figura 2.12 é ilustrada a relação entre a posição de um dado $\mathbf{x}_i$ e os valores de λ_i, ξ_i e C.

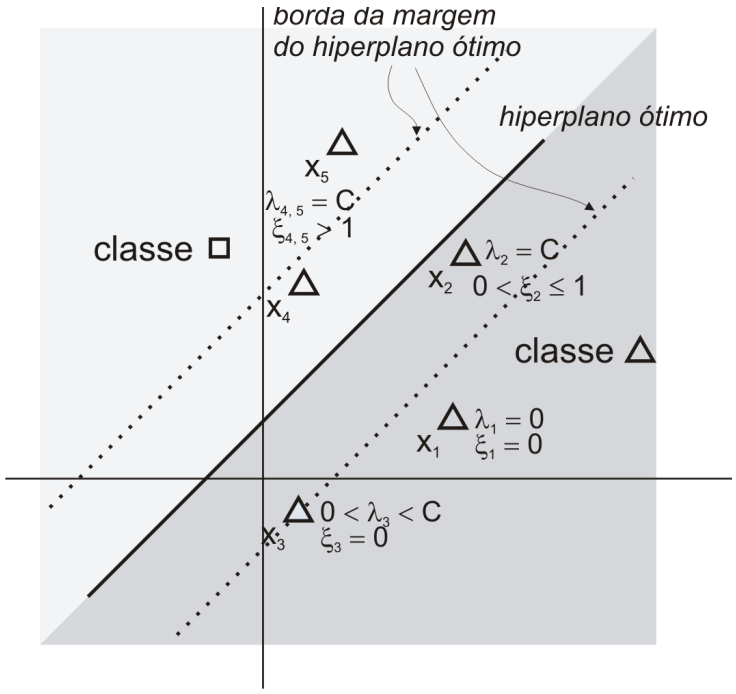


Figura 2.12: Influência dos valores de λ_i, ξ_i e C na posição de um dado $\mathbf{x}_i$

Considerando que $y_i = +1$ é representado pela classe $\triangle$ e $y_i = -1$ pela classe $\square$, a figura 2.12 pode ser interpretada da seguinte forma:

$(\lambda_1 = 0; \xi_1 = 0)$ O dado $\mathbf{x}_1$ está na região $\triangle$ $(\mathbf{w} \cdot \mathbf{x}_i + b > 1)$, ou seja, é classificado corretamente e sua distância até o hiperplano de separação é maior que $1/\|\mathbf{w}\|$. Como este dado não faz parte do conjunto de vetores de suporte, pode ser deletado do conjunto de aprendizado sem que haja prejuízo para a geração do hiperplano ótimo.

$(\lambda_2 = C; 0 < \xi_2 \leq 1)$ O dado $\mathbf{x}_2$, que é classificado corretamente, é chamado de vetor de suporte de fronteira, e sua distância até o hiperplano ótimo é menor que $1/\|\mathbf{w}\|$. Um dado $\mathbf{x}_i$ que pertença à classe $\triangle$ estará entre o hiperplano de separação ótimo $(\mathbf{w} \cdot \mathbf{x}_i + b = 0)$ e a borda da margem do hiperplano ótimo $(\mathbf{w} \cdot \mathbf{x}_i + b = +1)$. Caso pertença à classe $\square$ estará entre o hiperplano de separação ótimo $(\mathbf{w} \cdot \mathbf{x}_i + b = 0)$ e a borda da margem do hiperplano ótimo $(\mathbf{w} \cdot \mathbf{x}_i + b = -1)$.

$(0 < \lambda_3 < C; \xi_3 = 0)$ Esta situação corresponde à correta classificação do dado $\mathbf{x}_3$ que está situado na borda da margem do hiperplano ótimo. Sempre que isto ocorrer, um dado $\mathbf{x}_i$ estará no hiperplano $(\mathbf{w} \cdot \mathbf{x}_i + b = +1)$, caso pertença à classe $\triangle$ ou no

hiperplano $(\mathbf{w} \cdot \mathbf{x}_i + b = -1)$, caso pertença à classe □. A distância entre estes dados e o hiperplano ótimo é $1/\|\mathbf{w}\|$ e são chamados de vetores de suporte de margem.

$(\lambda_{4,5} = C; \xi_{4,5} > 1)$ Os dados $\mathbf{x}_4$ e $\mathbf{x}_5$ são classificados incorretamente. Esta situação corresponde a um dado $\mathbf{x}_i$, pertencente à classe △ que está situado na região da classe □, definida pelo hiperplano de separação $(\mathbf{w} \cdot \mathbf{x}_i + b < 0)$, enquanto que um dado que pertence à classe □ está situado na região da classe △, definida pelo hiperplano de separação $(\mathbf{w} \cdot \mathbf{x}_i + b < 0)$.

Para a classificação de novos dados, será utilizada a mesma função 2.27.

2.8 Mapeamento em Alta Dimensão e Produto Interno Kernel

Até o momento foram consideradas funções lineares, capazes de separar grande parte dos dados em problemas de classificação. Para problemas não lineares é feito um mapeamento do conjunto de treinamento, referenciado como espaço de entradas, para um novo espaço de maior dimensão, chamado de espaço de características. Esse mapeamento será representado pela função $\Phi : \mathbf{R}^n \rightarrow \mathcal{F}$, em que $\mathbf{R}^n$ é o espaço de entradas e $\mathcal{F}$ o espaço de características. Após esta transformação, poderá ser feita a separação dos dados por um classificador linear (Smola et al., 2000). Por exemplo, na figura 2.13, um classificador linear, mesmo com variáveis de relaxamento, não é apropriado para efetuar a classificação. O mapeamento dos dados originais para um espaço de alta dimensão, via função $\Phi(\mathbf{x})$, permite a separação por um classificador linear.

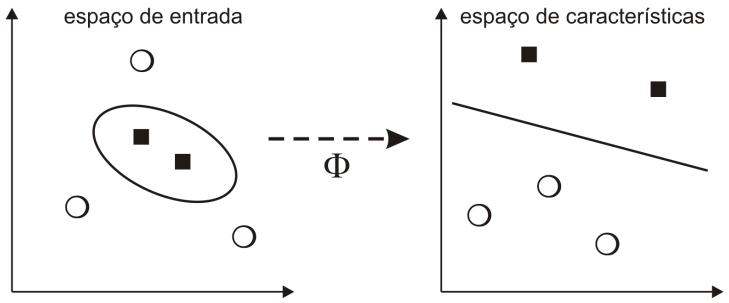


Figura 2.13: Separação linear dos dados no espaço de características
Fonte: Adaptado de (Smola et al., 2000)

No problema de treinamento (equação 2.37), a única maneira que os dados aparecem é na forma de produto escalar, $\mathbf{x}_i \cdot \mathbf{x}_j$. Portanto, o algoritmo de treinamento somente dependerá dos dados em $\mathcal{F}$ também na forma de produto escalar, ou seja, $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ (Burges, 1998). Para resolver o problema de classificação quando é feito o mapeamento para uma alta dimensão, é considerada praticamente a mesma estrutura de hiperplano de separação para classes não linearmente separáveis (seção 2.7), com a exceção que as variáveis $\mathbf{x}$ são substituídas pelo vetor de características $\Phi(\mathbf{x})$. Desta forma, a classe de um novo dado $\mathbf{x}_k$ é determinada por meio da equação 2.39 (Ivanciuc, 2007).

$$classe(\mathbf{x}_k) = sgn(\sum_{i=1}^{m} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_k) + b) \tag{2.39}$$

Segue um exemplo de classes linearmente não separáveis que podem ser separadas de forma linear no espaço de características. Considere o conjunto de dados bi-dimensional, representado na tabela 2.1. Suas dimensões x_1 e x_2 consistem de três dados pertencentes à classe $+1$ e seis que pertencem à classe -1.

Tabela 2.1: Dados no espaço de entrada versus espaço de características
Fonte: Adaptado de (Ivanciuc, 2007)

Dado	x_1	x_2	x_1^2	Classe
1	−1	−1	+1	−1
2	−1	0	+1	−1
3	−1	+1	+1	−1
4	0	−1	0	+1
5	0	0	0	+1
6	0	+1	0	+1
7	+1	−1	+1	−1
8	+1	0	+1	−1
9	+1	+1	+1	−1

Pela figura 2.14, pode-se perceber que não é possível separar as duas classes por uma função linear.

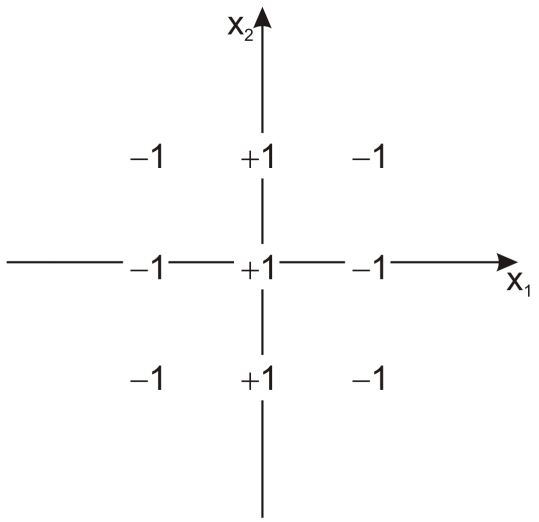


Figura 2.14: Dados bi-dimensionais, não separáveis linearmente
Fonte: Adaptado de (Ivanciuc, 2007)

Adicionar x_1^2 aos dados de entrada (quarta coluna na tabela 2.1) é considerado como a adição de uma nova dimensão. Após esta transformação, o conjunto de dados está representado em uma espaço de características de três dimensões.

A superfície $f(x_1, x_2) = x_1^2$ está representada na figura 2.15, juntamente com a projeção dos pontos (x_1, x_2, x_1^2), que representam os dados de entrada no espaço de características.

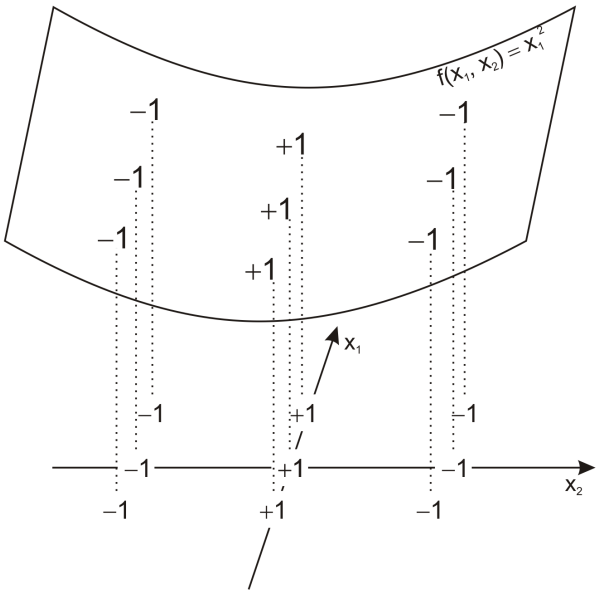


Figura 2.15: Pontos (x_1, x_2, x_1^2) no espaço de características
Fonte: Adaptado de (Ivanciuc, 2007)

Com esta transformação, agora é possível efetuar a separação das classes por um hiperplano, conforme apresentado na figura 2.16.

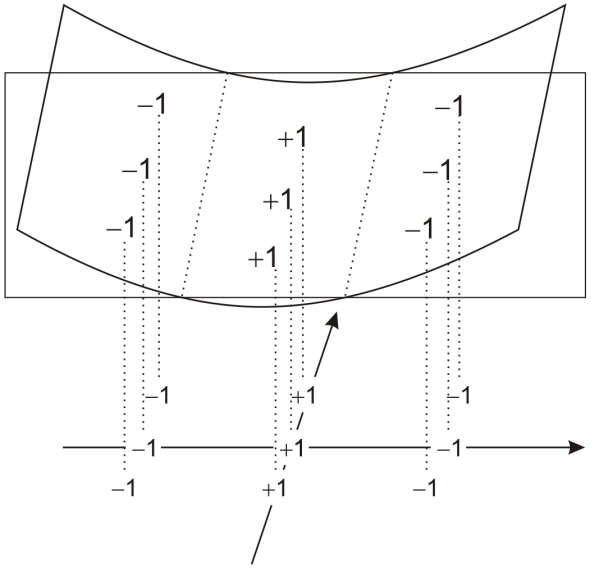


Figura 2.16: Um hiperplano que separa os dados +1 dos −1
Fonte: Adaptado de (Ivanciuc, 2007)

A intersecção entre o espaço de características e o hiperplano define a região de decisão, que é então projetada de volta no espaço original (figura 2.17).

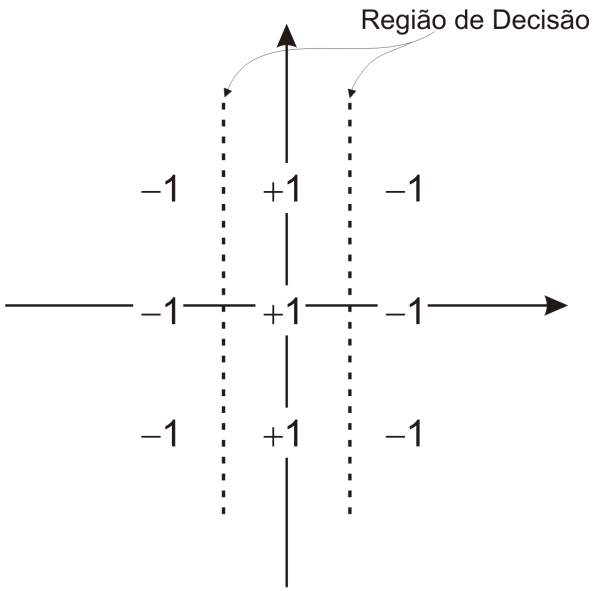


Figura 2.17: Projeção do hiperplano de separação
Fonte: Adaptado de (Ivanciuc, 2007)

Infelizmente, para um dado conjunto de dados, não é possível prever quais funções no espaço de características farão com que os dados sejam linearmente separáveis. Encontrar uma boa função é, portanto, um processo de tentativa e erro.

Na equação 2.39, uma dificuldade em obter uma boa região de separação é identificar a função Φ que melhor se adapta a determinado conjunto de entrada (Ivanciuc, 2007). Uma forma de contornar este problema é utilizar uma classe de funções K, chamadas *kernels*. Um *kernel* K é uma função que recebe dois dados $\mathbf{x}_i$ e $\mathbf{x}_k$ do espaço de entradas e computa o produto escalar desses dados no espaço de características $\left(K(\mathbf{x}_i, \mathbf{x}_k) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_k)\right)$ (Herbrich, 2001).

Um exemplo de função *Kernel* é:

$$K(\mathbf{x}_i, \mathbf{x}_k) = e^{-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma^2} \tag{2.40}$$

onde σ é fornecido pelo usuário. O cálculo do produto escalar dos dados no espaço de características é feito de forma implícita, fazendo com que seja desnecessário o conhecimento de qual função Φ está sendo utilizada (Burges, 1998).

Segundo Herbrich (2001), as condições necessárias e suficientes para uma função ser

kernel são dadas pelo teorema de Mercer (Mercer, 1909). De forma resumida, para um *kernel* satisfazer as condições de Mercer, ele deve dar origem a matrizes positivas semi-definidas **K**, em que cada elemento K_{ij} é definido por $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

Os primeiros kernels investigados e mais utilizados são:

Tabela 2.2: Kernels mais utilizados
Fonte: Adaptado de (Burges, 1998)

Kernel	$K(\mathbf{x}_i, \mathbf{x}_k)$
Polinomial	$(\mathbf{x}_i \cdot \mathbf{x}_k + 1)^p$
Radial Basis Function (RBF)	$e^{-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma^2}$
Sigmoidal	$\tanh(v\mathbf{x}_i \cdot \mathbf{x}_k - \delta)$

Com o uso da função *kernel*, a classe para o novo dado $\mathbf{x}_k$ (equação 2.39), pode ser calculada da seguinte forma (equação 2.41):

$$classe(\mathbf{x}_k) = sgn(\sum_{i=1}^{m} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}_k) + b) \tag{2.41}$$

2.9 Considerações Finais

Neste capítulo foram apresentados conceitos básicos da teoria de aprendizagem estatística, em especial os princípios de minimização do risco estrutural e empírico. São estes princípios que geram uma base sólida para a construção de máquinas de aprendizagem eficientes. Também foi apresentada a técnica de aprendizagem de máquina denominada Máquinas de Vetores de Suporte (SVM), que emprega estes princípios em sua formulação. Nos conceitos relacionadas à SVM, foi mostrada sua formulação básica com classes linearmente separáveis e não linearmente separáveis, tanto nas versões primais e duais. Apresentou-se também o mapeamento dos dados em alta dimensão, para facilitar a tarefa de separação dos dados. No próximo capítulo será apresentado um tipo de problema muito comum em conjuntos de dados reais - o problema com classes desbalanceadas.

Capítulo 3

Classes Desbalanceadas

A maioria dos algoritmos de classificação assume que a distribuição dos dados seja balanceada ou com custos iguais quando classificados incorretamente. Entretanto, quando apresentados a conjuntos de dados desbalanceados, estes algoritmos falham e favorecem apenas a classe dominante (também chamada de classe majoritária ou classe negativa). A classe dominada também pode ser identificada como minoritária ou positiva. No restante deste trabalho serão utilizadas as denominações negativa e positiva.

A dificuldade em se trabalhar com problemas de classificação onde existem classes desbalanceadas e sua ocorrência em aplicações práticas de aprendizado de máquina têm atraído considerável interesse na área científica (Chawla, Japkowicz, & Kotcz, 2004; Japkowicz, 2000; F. Provost, 2000). Outra considerável demonstração de interesse na área de classificação de dados com classes desbalanceadas pode ser vista na Figura 3.1 (He & Garcia, 2009), onde é mostrada uma estimativa do número de publicações em problemas de aprendizado de dados desbalanceados no período de 1997 a 2007, baseado no Institute of Electrical and Electronics Engineers (IEEE) e Association for Computing Machinery (ACM).

Neste capítulo serão apresentados os conceitos que envolvem classes desbalanceadas e algumas soluções mais conhecidas para resolver este problema.

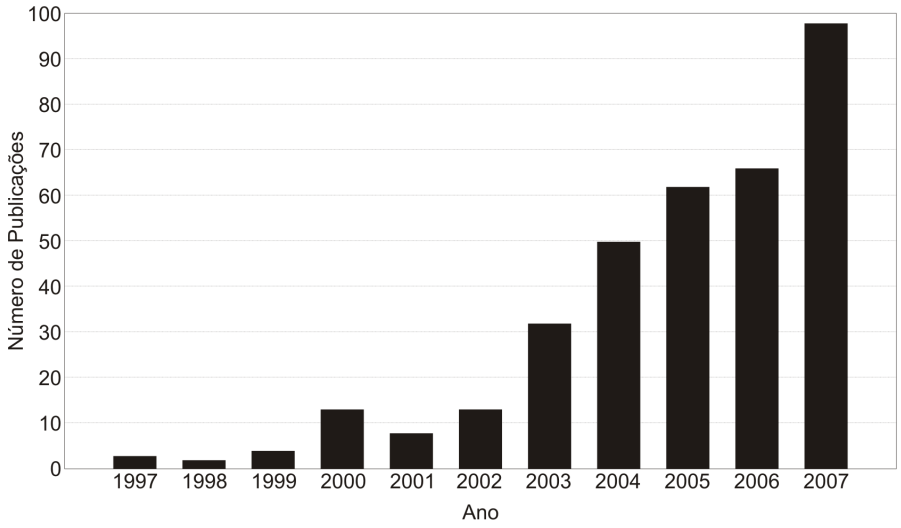


Figura 3.1: Número de publicações relacionadas a dados desbalanceados
Fonte: (He & Garcia, 2009)

3.1 A dificuldade na classificação de dados desbalanceados

O problema que envolve classes desbalanceadas é caracterizado quando existem mais instâncias de uma certa classe que em outras (pode ser na ordem de 1:100, 1:1000 ou mais) (Chawla et al., 2004). Em proporções menores (1:35, 1:10 ou até menos) também pode haver problema na geração de um bom modelo de classificação (Joshi, 2002). Na maioria das aplicações, a classificação correta de um dado pertencente à classe positiva freqüentemente tem mais valor que no caso contrário. Por exemplo, em problemas de diagnóstico de doenças, onde a quantidade de casos de doenças é bem menor quando comparada com a população total, o objetivo do reconhecimento é detectar pessoas doentes. Então, um bom modelo de classificação é aquele que fornece uma alta taxa de identificação para instâncias que pertencem a esta categoria. Para ilustrar melhor esta situação, considere um conjunto de dados de imagens de mamografias. Analisando as imagens, pode-se obter classes "positivas" e "negativas", onde representam pacientes com câncer e saudáveis, respectivamente. Existem 11.183 imagens, das quais 10.923 pertencem à classe negativa (classe majoritária) e 260 à classe positiva (classe minoritária). Um classificador ideal seria aquele que tivesse uma taxa de acerto de 100% para ambas as classes. Mas na prática, existem classificadores que podem gerar uma taxa de acerto próxima de 100%

para a classe negativa e de 0 a 10% para a classe positiva (He & Garcia, 2009). Desta taxa de acerto, pode-se concluir que 234 imagens são incorretamente classificadas como pessoas saudáveis (pertencentes à classe negativa). Na área médica, isto gera conseqüências mais sérias que classificar um paciente saudável como portador de câncer. Outras áreas onde são utilizadas classes desbalanceadas são em detecção de fraude, detecção de intrusos em redes e derramamento de óleo (Kubat, Holte, & Matwin, 1998; Rao, Krishnan, & Niculescu, 2006; Chan, Fan, Prodromidis, & Stolfo, 1999).

Logo, é importante diferenciar os dois tipos de erros que podem aparecer: um alarme falso (classificar uma classe negativa como positiva) não é tão grave quanto a ausência de um alarme correto (classificar uma classe positiva como negativa). Por exemplo, no uso de scanners de ressonância magnética para detecção de tumores, é importante evitar falsos resultados negativos, mas um pequeno número de falso positivo pode ser tolerado se o exame for posteriormente conferido pelo médico. Da mesma forma, em condições de monitoramento de máquinas, falsos alarmes ocasionais são menos catastróficos que deixar de emitir um alarme correto de uma falha em uma máquina (Veropoulos et al., 1999).

Geralmente, o desempenho dos classificadores em conjuntos de dados desbalanceados é fraco, pois eles são projetados para generalizar a partir dos dados de treinamento e geram o classificador (hipótese) que melhor classifica estes dados, baseados no princípio da Navalha de Occam. Em dados desbalanceados, a hipótese mais simples, muitas vezes, é aquela que classifica todas as instâncias como negativa (Akbani et al., 2004).

3.2 Meios para avaliação

Os meios tradicionais para avaliação, tal como taxa de acerto global ou taxa de erro, não fornecem informações adequadas nos casos de aprendizado de classes desbalanceadas. Por exemplo, na seção 3.1, a classe positiva é representada por aproximadamente $2,3\%$ dos dados de treinamentos. Uma estratégia simples de classificação pode classificar todas as classes do conjunto de treinamento como sendo pertencente à classe negativa. Com isto, terá uma taxa de

acerto de $97,5\%$. Portanto são necessárias outras métricas para avaliação, tal como *F-measure* (Lewis & Gale, 1994), *G-mean* (Kubat et al., 1998) e área abaixo da curva ROC (do inglês *Area Under the ROC Curve (AUC)*) (Bradley, 1997).

Para formular critérios de desempenho dos sistemas de reconhecimento de padrões, estatísticos trabalham com a matriz de confusão 3.1, cujos campos mostram o comportamento de um dado sistema (Kubat & Matwin, 1997).

Tabela 3.1: Matriz de confusão: as colunas representam as classes atribuídas pelo classificador; as linhas representam as verdadeiras classes.

	Classe predita positiva	Classe predita negativa
Classe positiva	VP (Verdadeiro Positivo)	FN (Falso Negativo)
Classe negativa	FP (Falso Positivo)	VN (Verdadeiro Negativo)

Da tabela 3.1, várias medidas podem ser derivadas:

- Taxa de Verdadeiro Positivo: $VP_{taxa} = Recall = \dfrac{VP}{VP + FN}$
- Taxa de Verdadeiro Negativo: $VN_{taxa} = \dfrac{VN}{VN + FP}$
- Taxa de Falso Positivo: $FP_{taxa} = \dfrac{FP}{VN + FP}$
- Taxa de Falso Negativo: $FN_{taxa} = \dfrac{FN}{VP + FN}$
- Valor Predito Positivo: PP_{valor} = Precisão = $\dfrac{VP}{VP + FP}$
- Valor Predito Negativo: $PN_{valor} = \dfrac{VN}{VN + FN}$
- $F\text{-}measure = \dfrac{(1 + \beta^2) \cdot Recall \cdot \text{Precisão}}{\beta^2 \cdot Recall + \text{Precisão}}$

onde β corresponde à importância relativa de precisão × *recall* e é usualmente escolhido com o valor 1. O principal objetivo dos algoritmos de aprendizado é aprimorar o *recall* sem sacrificar a precisão (Chawla, Lazarevic, Hall, & Bowyer, 2003). Conforme mostrado, *F-measure* engloba tanto a precisão quanto o *recall*. Isso permite medir a "aceitação" do algoritmo de aprendizado para a classe minoritária.

Quando o desempenho de ambas as classes é importante, tanto a Taxa de Verdadeiro Positivo VP_{taxa} quanto a Taxa de Verdadeiro Negativo VN_{taxa} são esperados ter um alto valor. Kubat et al. (1998) sugeriu a métrica:

$$G\text{-}mean = \sqrt{VP_{taxa} \cdot VN_{taxa}} \tag{3.1}$$

Portanto, G-mean só terá um bom resultado caso VP_{taxa} e VN_{taxa} tenham um alto valor. Se VP_{taxa} tiver um alto valor e VN_{taxa} um baixo, ou vice-versa, o resultado de *G-Mean* será baixo.

A área abaixo da curva ROC fornece uma medida do desempenho do classificador para avaliar qual é melhor, na média (Fawcett, 2004). Para obter a área abaixo da curva ROC, é necessário obter a curva ROC. Ela é obtida usando os pares (FP_{taxa}, VP_{taxa}). A figura 3.2 ilustra uma curva ROC. A parte sombreada representa a área abaixo da curva ROC. O algoritmo para criar a curva e a área abaixo da curva podem ser encontrados em (Fawcett, 2004). Para comparar vários classificadores pela curva ROC, é difícil eleger qual é o vencedor, a menos que uma curva claramente domine a outra por todo o espaço (F. J. Provost & Fawcett, 1997).

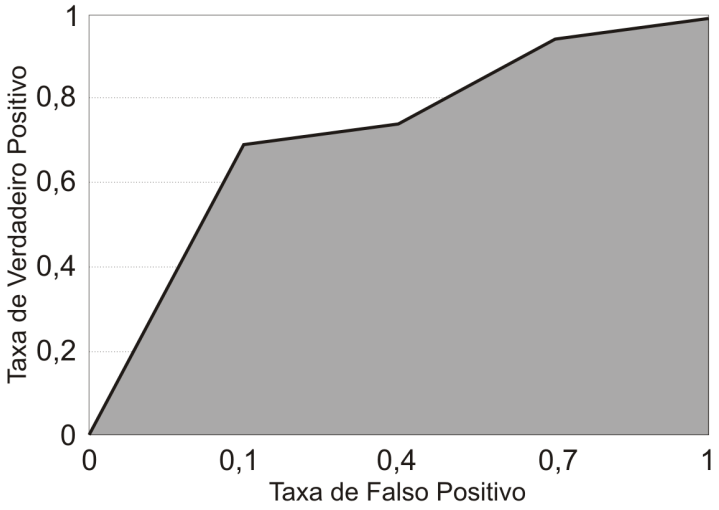


Figura 3.2: Representação ROC
Fonte: (X.-Y. Liu et al., 2009)

3.3 Soluções para o problema das classes desbalanceadas

Várias soluções para problemas que lidam com classes desbalanceadas foram propostos, tanto a nível de dados quanto a nível de algoritmo (Chawla et al., 2004).

Na abordagem a nível de dados, o objetivo é balancear a distribuição das classes, reamostrando de diferentes formas os dados no espaço de entrada. As soluções a nível de dados utilizam diversas formas de reamostragem, tal como super-amostragem[1] de exemplos da classe minoritária e sub-amostragem[2] de exemplos da classe majoritária. Super-amostragem é a duplicação de exemplos já existentes ou geração de novos dados de forma sintética, a partir dos exemplos já existentes (Chawla et al., 2002). A escolha dos elementos a serem replicados pode ser feita de forma randômica ou não. Sub-amostragem é a eliminação de elementos da classe majoritária. Os elementos a serem eliminados podem ser selecionados randomicamente ou de acordo com determinados critérios. Também existem técnicas que combinam super-amostragem com sub-amostragem (Chawla et al., 2002; Castro et al., 2009).

A nível de algoritmo, as soluções tentam adaptar (inserindo custos diferenciados em exemplos da classe minoritária e majoritária, alterando *kernels*, e outras técnicas) os algoritmos de classificação já existentes, para melhorar o desempenho da classe minoritária (Wu & Chang, 2003; Akbani et al., 2004; Lin, Lee, & Wahba, 2002; Veropoulos et al., 1999). Vários algoritmos na forma de um comitê de máquinas também são reportados como meta-técnicas para trabalhar com classes desbalanceadas. Essa abordagem será discutida na próxima seção (seção 3.3.1).

3.3.1 Comitês de Máquinas

Um comitê de máquinas (comitê de classificadores) representa a agregação de mais de uma máquina de aprendizado na produção de uma solução computacional para um determinado problema (Haykin, 1999). Além de estarem motivados pela busca da maximização da capacidade de generalização, os comitês de máquinas são motivados por pelo menos outras duas

[1] do inglês *oversampling*.
[2] do inglês *undersampling*.

razões (Moraes Lima, 2004):

- Grande disponibilidade de recursos computacionais para possibilitar a síntese de múltiplas propostas de soluções para todo ou parte do problema sendo tratado;
- A demonstração, através do teorema "no free lunch" (Wolpert & Macready, 1997), de que não existem modelos genéricos de aprendizado de máquina que, em média, apresentem melhor desempenho que qualquer outro modelo para uma classe de problemas quaisquer.

Uma condição necessária e suficiente para um comitê de máquinas ser mais preciso que qualquer um de seus membros (classificador individual) é que cada um possua uma taxa de erro melhor que uma previsão randômica, e que cada membro tenha erros diferentes para novos dados (Hansen & Salamon, 1990). Na figura 3.3 é apresentada uma estrutura de um comitê de máquinas.

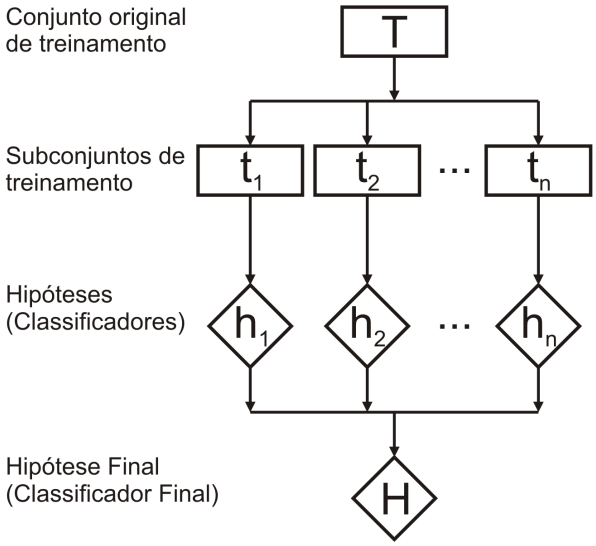


Figura 3.3: Estrutura de um comitê de máquinas

Dois métodos populares que são utilizados para a produção de comitês de máquinas, *boosting* e *bagging*, geram múltiplos classificadores através da amostragem do conjunto de treinamento (espaço de entrada): *boosting* trabalha inserindo pesos nas amostras e *bagging* reamostrando subconjuntos das instâncias de treinamento. A melhoria no desempenho que aparece ao utilizar comitês de máquinas geralmente é o resultado da redução da variância. A variância

mede o quanto as suposições do algoritmo de aprendizado variam para diferentes conjuntos de treinamento. Para um único classificador, uma alta variância não é bom, pois significa que a hipótese gerada se especializou no conjunto de treinamento. Com isso, ao ser apresentada a novos dados para o reconhecimento, não terá um bom desempenho. Isto é conseqüência de super-ajuste[3]. Mas para comitês de máquinas, uma alta variância é um requisito, para que haja diversidade para a geração de um classificação que tenha um bom desempenho, tanto nos dados utilizados para o treinamento quanto para os que ainda não foram vistos (Cunningham, 2000). *Boosting* e *Bagging* são capazes de reduzir a variância, logo são imunes ao problema de super-ajuste. Esses métodos serão mais detalhados nas seções seguintes.

3.3.1.1 Bagging

Este método foi proposto por Breiman (1996) e seu princípio de funcionamento é: dado um classificador fraco[4] e um conjunto de treinamento $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, a cada iteração selecione randomicamente m' exemplos de treinamento (tal que $m' < m$) do conjunto de treinamento, e coloque-os de volta após o treinamento. Após t iterações, terá uma lista de classificadores $h_1, h_2, \ldots, h_t$ que são então combinados por votação pela maioria de suas decisões. Para uma dada instância, a classe escolhida pela maioria dos classificadores é a decisão final (Polikar, 2006).

3.3.1.2 Boosting

Criado em 1990 por Schapire (1990), o algoritmo *boosting*, da mesma forma que *bagging*, cria um comitê de máquinas selecionando conjuntos de dados, que mais tarde são unidos por votação pela maioria (Polikar, 2006). O que diferencia o *boosting* é que a seleção do conjunto de dados é feita estrategicamente para prover dados de treinamento mais informativos

[3] detalhes na seção 2.1.1.

[4] Uma hipótese fraca (classificador fraco) é um classificador que classifica dados com uma precisão de 50% (Hulley & Marwala, 2007). Este termo "classificador fraco" é muito utilizado quando se fala em comitê de máquinas, apesar de que, na prática pode-se construir comitês de máquinas utilizando um classificador forte (Freund & Schapire, 1996). Em Lima et al. (2009); Kim et al. (2003) podem ser encontrados comitês de máquinas utilizando SVM.

para cada classificador que é gerado.

O algoritmo cria três classificadores (classificador A, B e C): o primeiro classificador (*A*) é treinado com um subconjunto selecionado randomicamente dos dados de treinamento. Para o segundo classificador (*B*), o subconjunto de treinamento é escolhido com o auxílio de *A*. *B* é treinado utilizando um conjunto de treinamento que é a metade do que foi classificado corretamente por *A*, e a outra metade que foi classificado incorretamente. O trecho deste algoritmo está representado na figura 3.4. O terceiro classificador (*C*) é treinado com instâncias que *A* e *B* discordaram na classificação. Na classificação final, são utilizados os classificadores *A* e *B*. Se eles preverem a mesma classe, a classificação é aceita. Caso contrário, a classificação será a que foi predita por *C* (Schapire, 1990).

```
Jogue uma moeda.

Se der cara, selecione instâncias do con-
junto de treinamento e forneça-as ao clas-
sificador A até que se obtenha a primeira
classificação incorreta. Adicione esta
instância ao subconjunto que será utili-
zado para treinar o segundo classificador B.

Caso dê coroa, selecione instâncias do
conjunto de treinamento e forneça-as a A
até obter uma classificação correta. Adi-
cione esta instância ao subconjunto que
será utilizado para treinar o segundo clas-
sificador B.

Continue jogando a moeda até que selecione
a quantidade pré-estabelecida de elementos
para o subconjunto de treinamento de B.
```

Figura 3.4: Trecho para seleção do subconjunto de treinamento para o segundo classificador

3.3.1.2.1 *AdaBoost*

Da mesma forma que *bagging* e *boosting*, *AdaBoost* (Freund & Schapire, 1997) - que é a implementação mais conhecida de *boosting* - manipula dados de treinamento utilizando pesos e gera um conjunto de hipóteses. A cada iteração t, o algoritmo de aprendizado é chamado para minimizar o erro no conjunto de treinamento, e retornar uma hipótese h_t. O erro de h_t é

calculado e usado para atualizar os pesos dos exemplos de treinamento. O efeito na alteração dos pesos é que valores mais altos são colocados para os exemplos de treinamento que foram incorretamente classificados por h_t e valores menores em exemplos que foram corretamente classificados. Com isso, nas iterações subseqüentes, são construídos problemas de aprendizado mais difíceis.

O classificador final é construído por votação com pesos dos classificadores individuais. Cada classificador tem o peso calculado de acordo com a precisão obtida em seu conjunto de treinamento.

AdaBoost possui versões para problemas binários (*AdaBoost*) e multi-classe (*AdaBoost*.M1). A diferença entre eles está na forma como é calculada a hipótese final. O algoritmo 1 é a representação do *AdaBoost*.M1.

Entrada:

seqüência de m exemplos $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\mathsf{m}, y_\mathsf{m})$ com rótulos $y_\mathsf{i} \in Y = \{1, \ldots, \mathsf{k}\}$;
algoritmo de aprendizado fraco `AAF`;
inteiro T que especifica o número de iterações;

Saída:

hipótese final h_{fin};

Inicialize $\mathsf{D}_1(\mathsf{i}) = 1/\mathsf{m}$ para todo i;
para $\mathsf{t} \leftarrow 1$ **até** T **faça**
  Chame `AAF`, passando como argumento a distribuição D_t;
  Receba como retorno a hipótese $\mathsf{h}_\mathsf{t} : X \rightarrow Y$;
  Calcule o erro de $\mathsf{h}_\mathsf{t} : \epsilon_\mathsf{t} = \sum_{\mathsf{i}:\mathsf{h}_\mathsf{t}(\mathbf{x}_\mathsf{i}) \neq y_\mathsf{i}} \mathsf{D}_\mathsf{t}(\mathsf{i})$;
  se $\epsilon_\mathsf{t} > 1/2$ **então** aborte;
  $\beta_\mathsf{t} = \frac{1}{2} \ln \frac{(1 - \epsilon_\mathsf{t})}{\epsilon_\mathsf{t}}$;
  `// Atualize a distribuição` D_t.
  $\mathsf{D}_{\mathsf{t}+1}(\mathsf{i}) = \frac{\mathsf{D}_\mathsf{t}(\mathsf{i}) \exp(-\beta_\mathsf{t} \mathsf{h}_\mathsf{t}(\mathbf{x}_\mathsf{i}) y_\mathsf{i})}{\mathsf{Z}_\mathsf{t}}$;
  `/* onde` Z_t `é o fator de normalização, de forma que` $\mathsf{D}_{\mathsf{t}+1}$ `seja uma distribuição. */`
fim
`// A hipótese final será:`
$\mathsf{h}_{fin}(\mathbf{x}) = \arg\max_{y \in Y} \sum_{\mathsf{t}=1}^{T} \beta_\mathsf{t} [\mathsf{h}_\mathsf{t}(\mathbf{x}_\mathsf{i}) = y_\mathsf{i}]$;
`// onde` $[v] = 1$ `se` v `for verdadeiro e` $[v] = 0$ `caso contrário.`

Algoritmo 1: Algoritmo *AdaBoost*.M1

3.4 Trabalhos Relacionados

3.4.1 SMOTE (Synthetic Minority Over-sampling TEchnique)

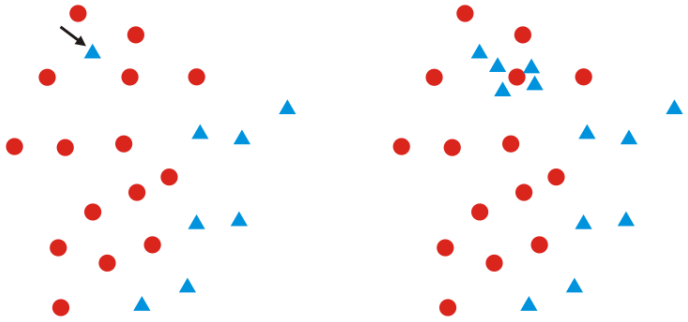
Desenvolvido por Chawla et al. (2002), o método denominado SMOTE efetua super-amostragem da classe positiva, criando novas instâncias "sintéticas", ao invés da seleção de amostras. Sendo mais específico, tem-se que $|S| = |S_{pos}| + |S_{neg}|$, onde S representa um dado conjunto de treinamento com m instâncias, ou seja, ($|S| = m$), S_{pos} e S_{neg} representam os conjuntos das classes positivas e negativas, respectivamente. Para o subconjunto S_{pos}, considere os k-vizinhos mais próximos[5] para cada instância $\mathbf{x}_i \in S_{pos}$, para um dado valor de k. Os k-vizinhos mais próximos são definidos como os k elementos de S_{pos} cuja distância euclidiana entre si e a instância $\mathbf{x}_i$ possuem o menor valor. Para criar uma nova instância "sintética", randomicamente selecione um dos k-vizinhos mais próximos, subtraia a instância $\mathbf{x}_i$ de seu vizinho mais próximo, multiplique esta diferença por um número randômico entre 0 e 1 e, adicione-a ao valor da instância ($\mathbf{x}_i$). Veja um exemplo prático na tabela 3.2.

Tabela 3.2: Exemplo de geração de instâncias "sintéticas"
Fonte: Adaptado de (Chawla et al., 2002)

Considere a instância $(6,4)$ e seja $(4,3)$ um de seus k-vizinhos mais próximos.
Seja:
$f_{1,1} = 6$ $f_{2,1} = 4$ $f_{2,1} - f_{1,1} = -2$
$f_{1,2} = 4$ $f_{2,2} = 3$ $f_{2,2} - f_{1,2} = -1$
A nova instância será gerada como
$(f_1', f_2') = (6,4) + \text{rand}(0-1) \times (-2,-1)$
$\text{rand}(0-1)$ gera um número randômico entre 0 e 1

Apesar deste algoritmo ser uma evolução quando comparado com super-amostragem de forma randômica, não é feito um controle da posição onde o novo elemento é criado. Em instâncias da classe positiva que estão fora de sua região de classificação (ruídos), ao gerar novo elementos "sintéticos", podem aparecer mais ruídos e prejudicar o processo de classificação. Conforme ilustrado na figura 3.5.

[5] detalhes do funcionamento deste algoritmo pode ser encontrado em Duda et al. (2001)



(a) Ruído na classe negativa

(b) Geração de dados "sintéticos" na região incorreta

Figura 3.5: Caso em que a geração de novos dados positivos aumenta a quantidade de ruídos

3.4.2 Veropoulos

Veropoulos et al. (1999) propôs diferentes penalidades para a classe positiva e negativa, fazendo com que os erros nas instâncias positivas sejam mais penalizados que os das instâncias negativas. Especificamente, Veropoulos et al. sugeriu alterar a forma primal do problema (veja equação 2.30) para

$$L_P(\mathbf{w}, b, \mathbf{\Lambda}, \mathbf{M}) = \frac{1}{2}\|\mathbf{w}\|^2 + C^+ \sum_{\{i|yi=+1\}}^{m_+} \xi_i + C^- \sum_{\{j|yj=-1\}}^{m_-} \xi_j - \sum_{i=1}^{m} \lambda_i [y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{m} \mu_i \xi_i \tag{3.2}$$

onde $\lambda_i \geq 0$ e $\mu_i \geq 0$. A forma dual é a mesma da equação 2.37, mas com a restrição de λ_i alterada para:

$$0 \leq \lambda_i \leq C^+ \text{ se } y_i = +1 \quad e \quad 0 \leq \lambda_i \leq C^- \text{ se } y_i = -1 \tag{3.3}$$

Esta proposta altera o comportamento do algoritmo de tal forma que a margem ficará assimétrica[6]. Isto fará com que o hiperplano de decisão fique mais afastado da classe positiva que da classe negativa. A vantagem é que reduzirá o risco de uma classificação incorreta da classe positiva.

Esta técnica para melhorar a representatividade da classe positiva é eficiente e utilizada

[6]na proposta original de SVM, a margem é simétrica, ou seja, sua distância até os vetores de suporte da classe positiva é a mesma para os vetores de suporte da classe negativa.

até hoje. Prova disso é que ainda é utilizada nas implementações de SVM mais populares: LIBSVM (Chang & Lin, 2001) e SVM^{light} (Joachims, 1999). Mas sozinha, não possui um desempenho tão bom quanto utilizada com uma técnica de sub-amostragem.

3.4.3 SDC (SMOTE *with Different Costs*)

SDC foi proposto por Akbani et al. (2004). A idéia principal de seu funcionamento é unir a técnica SMOTE, apresentada na seção 3.4.1, com custos diferentes para a classe negativa e positiva (3.4.2). Ele utilizou esta idéia porque afirma que uma abordagem que é aceita para lidar com conjuntos de dados desbalanceados é induzir o classificador, de forma que preste mais atenção às instâncias da classe positiva. E isto pode ser feito com a introdução de diferentes penalidades para as diferentes classes. A outra abordagem é efetuar o pré-processamento dos dados através da super-amostragem da classe positiva ou sub-amostragem da classe negativa, a fim de criar um conjunto mais balanceado. Para isto, utiliza as seguintes estratégias:

- não efetuar sub-amostragem da classe negativa, pois isto leva a perda de informações;
- utilizar custos diferenciados para erros de classificação de classes diferentes para afastar o hiperplano de separação da classe positiva;
- utilizar SMOTE para tornar as instâncias positivas distribuídas de uma forma mais densa, para que o hiperplano seja melhor definido.

Em seu trabalho, Akbani não efetua sub-amostragem da classe negativa alegando que pode levar à perda de informações importantes. Mas nos testes que constam em seu artigo, as sub-amostragens realizadas foram realizadas de forma randômicas. Efetuar a sub-amostragem de dados utilizando algum tipo de heurística, a possibilidade de acontecer perda de informações potencialmente importantes é menor. Isso foi provado nesta tese.

3.4.4 AdaC1, AdaC2 e AdaC3

Sun et al. (2007) utilizou o algoritmo *AdaBoost* (seção 3.3.1.2.1) e inseriu um fator de custo na função de distribuição D_t (algoritmo 1) que faz a atualização dos pesos associados a cada instância utilizada no treinamento do algoritmo *AdaBoost*. Este fator de custo foi inserido de três formas - dentro da exponencial, fora da exponencial e em ambos: fora e dentro da exponencial - gerando três algoritmos denominados AdaC1, AdaC2 e AdaC3. Com estas alterações, a função de distribuição será, respectivamente:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\beta_t C_i h_t(\mathbf{x}_i) y_i)}{Z_t} \tag{3.4}$$

$$D_{t+1}(i) = \frac{C_i D_t(i)\exp(-\beta_t h_t(\mathbf{x}_i) y_i)}{Z_t} \tag{3.5}$$

$$D_{t+1}(i) = \frac{C_i D_t(i)\exp(-\beta_t C_i h_t(\mathbf{x}_i) y_i)}{Z_t} \tag{3.6}$$

O fator de custo C_i é associado a cada instância $\mathbf{x}_i$ e os valores mais altos de C_i correspondem àquelas instâncias com maior custo, quando classificadas incorretamente. Estes algoritmos aumentam a probabilidade de seleção das instâncias mais difíceis de serem classificadas a cada iteração.

Esta modificação possui o mesmo princípio de funcionamento da técnica apresentada na seção 3.4.2, mas funciona exclusivamente para o algoritmo *AdaBoost*.

3.4.5 *EasyEnsemble*

Como o uso de sub-amostragem da classe negativa pode gerar perda de informações, X.-Y. Liu et al. (2009) desenvolveram o método denominado *EasyEnsemble* (algoritmo 2), que é um sistema baseado em um comitê de máquinas que cria vários subconjuntos independentes da classe negativa e desenvolve múltiplos classificadores com a combinação destes subconjuntos com a classe positiva.

Apesar do autor não ter utilizado a técnica de sub-amostragem, afirmando que ocor-

Entrada:
conjunto dos exemplos da classe minoritária P;
conjunto dos exemplos da classe majoritária N;
// |P| < |N|
o número de subconjuntos T a serem amostrados de N;
o número de iterações s_i para treinar o comitê de máquinas (AdaBoost) H_i;

i = 0;
repita
i =i +1;
Randomicamente selecione um subconjunto N_i de N, de forma que $|N_i| = |P|$;
Obtenha H_i utilizando P e N_i. Onde H_i é o comitê de máquinas *Adaboost* com s_i classificadores $h_{i,j}$ e os correspondentes pesos $\alpha_{i,j}$. Θ_i representa o limiar do comitê de máquinas

$$H_i(x) = sgn(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \Theta_i);$$

até i =T;

Saída:
o comitê de máquinas

$$H(x) = sgn(\sum_{i=1}^{T} \sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \sum_{i=1}^{T} \Theta_i);$$

Algoritmo 2: Algoritmo *EasyEnsemble*

rerá perda de dados importantes, ao efetuar sub-amostragem de dados utilizando algum tipo de heurística, esse problema pode ser superado (Gu et al., 2008; He & Garcia, 2009).

3.4.6 *Boundary Elimination and Domination (BED)*

A idéia principal do algoritmo *BED* (Castro et al., 2009) é melhorar a representatividade da classe positiva através do pré-processamento dos dados no espaço de entrada. Este pré-processamento é feito através da sub-amostragem das instâncias negativas consideradas ruidosas e da super-amostragem da classe positiva. Estes processos são guiados por uma "pontuação de credibilidade" dada a cada amostra do conjunto de treinamento. Esta pontuação é calculada através dos *k*-vizinhos mais próximos de cada amostra, através da equação

$$cs(x_i^c) = \frac{nC^c}{nC^+ + nC^-} \tag{3.7}$$

onde:

$cs(x_i^c)$ - i-ésima instância de treinamento da classe C^c. O símbolo $c = \{+, -\}$ indica se o exemplo pertence à classe positiva ou negativa, respectivamente;

nC^+ - quantidade de instâncias que pertencem à classe positiva que estão presentes nos k-vizinhos mais próximos;

nC^- - quantidade de instâncias que pertencem à classe negativa que estão presentes nos k-vizinhos mais próximos.

Se uma instância pertencente à classe negativa tiver uma "pontuação de credibilidade" menor que *max_tol* (valor a ser fornecido, como parâmetro), ele é eliminado do conjunto de treinamento. Se a instância pertencer à classe positiva e sua pontuação for maior ou igual que *min_tol* (valor a ser fornecido, como parâmetro), um novo elemento é criado, segundo os critérios:

- para cada atributo contínuo m, seu novo valor é calculado baseado na média dos nC^+ vizinhos mais próximos e a amostra x_i^+;
- para os atributos nominais, será assumido o valor que tiver a maior frequência nos nC^+ vizinhos mais próximos e a amostra x_i^+.

Caso a instância pertencente à classe positiva não satisfaça a "pontuação de credibilidade", ela não é utilizada para a geração no novo elemento. Mas continua no conjunto de dados, e pode fazer com que a geração do classificador seja prejudicada.

3.5 Considerações Finais

Este capítulo apresentou as principais características de um problema de classificação. Também mostrou que as técnicas tradicionais de avaliação de desempenho de algoritmos não são eficientes para conjuntos de dados desbalanceados. Por fim apresentou algumas propostas conhecidas que propõem melhorar o desempenho de classificação para dados desbalanceados. No próximo capítulo será apresentado um novo algoritmo que tem como base o SMOTE e *EasyEnsemble*, apresentados neste capítulo.

Capítulo 4

Abordagem Proposta

Conforme apresentado na seção 3.3, para tratar o problema gerado por classes desbalanceadas, existem propostas que trabalham a nível de dados e a nível de algoritmo. Na abordagem a nível de dados, o objetivo é balancear a distribuição das classes, reamostrando de diferentes formas os dados no espaço de entrada, tal como super-amostragem de exemplos da classe positiva e sub-amostragem de exemplos da classe negativa. Existem técnicas que combinam super-amostragem com sub-amostragem (Chawla et al., 2002; Castro et al., 2009). A nível de algoritmo, uma solução muito empregada é a inserção de custos diferenciados para classes positivas e negativas (Veropoulos et al., 1999; Sun et al., 2007; Morik, Brockhausen, & Joachims, 1999; Domingos, 1999; Osuna, Freund, & Girosi, 1997a). Essa técnica de inserção de custos diferenciados já é um recurso implementado nas ferramentas de classificação de dados mais populares como: LIBSVM (Chang & Lin, 2001) e SVM^{light} (Joachims, 1999). Outra técnica que também é utilizada para melhorar o desempenho de classificação de dados é o uso de comitês de máquinas (apresentada na seção 3.3.1). Existem trabalhos que propõem o uso de mais de uma técnica citada anteriormente, para que se possa chegar a uma melhor taxa de classificação. No artigo Akbani et al. (2004), é proposto a combinação da técnica SMOTE (Chawla et al., 2002) com a inserção de custos diferenciados para classes positivas e negativas (Veropoulos et al., 1999). No artigo Sun et al. (2007) utiliza-se o algoritmo de comitê de máquina *AdaBoost* e a técnica de tratamento de dados a nível de algoritmo, onde são inseridos

custos diferenciados, tanto para a classe positiva quanto para a negativa.

Após o levantamento das contribuições que cada algoritmo fornece, foi feito um estudo se poderia obter algo mais das características de cada um. Chegou-se a uma modificação no algoritmo SMOTE (que gera novos exemplos "sintéticos"), de forma que reduzisse a possibilidade de geração de novos elementos na região incorreta. Como em conjuntos de dados altamente desbalanceados, a geração de elementos "sintéticos" não é suficiente para equilibrar o conjunto, houve a necessidade da criação de um novo algoritmo para efetuar uma sub-amostragem de exemplos da classe majoritária. E, para melhorar a capacidade de generalização do classificador gerado, também foi feita uma modificação no algoritmo de comitês de máquinas (*EasyEnsemble*). Utilizando estas três etapas, obteve-se um algoritmo composto que possui uma taxa de acerto na classificação de dados melhor que os algoritmos nos quais se baseou. Essas etapas do algoritmo são detalhadas a seguir.

4.1 Redução da quantidade de elementos da classe negativa

Inicialmente é feita uma sub-amostragem de exemplos da classe negativa. Esta etapa é importante pois elimina grande parte dos exemplos que estão na região incorreta de classificação. Estes exemplos também são chamados de *outliers*. O algoritmo 3 ilustra todo o processo de sub-amostragem de exemplos da classe negativa.

Entrada:
conjunto de m dados $C = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ com $y_i \in Y = \{1, 2\}$;
conjunto de dados $C' = \{\ \}$;

Saída:
conjunto de m' dados $C' = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m'}, y_{m'})\}$ com $y_i \in Y = \{1, 2\}$;

```
/* contando a quantidade de elementos pertencentes à classe
   positiva e à classe negativa                                */
```
tampos = quantidade de elementos da classe positiva;

Conforme apresentado no algoritmo 3, o processo consiste em encontrar o centróide de cada classe, que é calculado pela média aritmética de cada atributo. Logo em seguida é cal-

tamneg = quantidade de elementos da classe negativa;
// dim é a quantidade de atributos que o dado **x** possui
dim = dimensão de **x**;
/* separar o conjunto dos dados que pertence à classe positiva
do conjunto dos dados pertencentes à classe negativa */
conjpos = conjunto de dados de C que pertencem à classe positiva;
conjneg = conjunto de dados de C que pertencem à classe negativa;

/* calculando o centróide da classe positiva e da classe
negativa */
para t ← 1 **até** dim **faça**

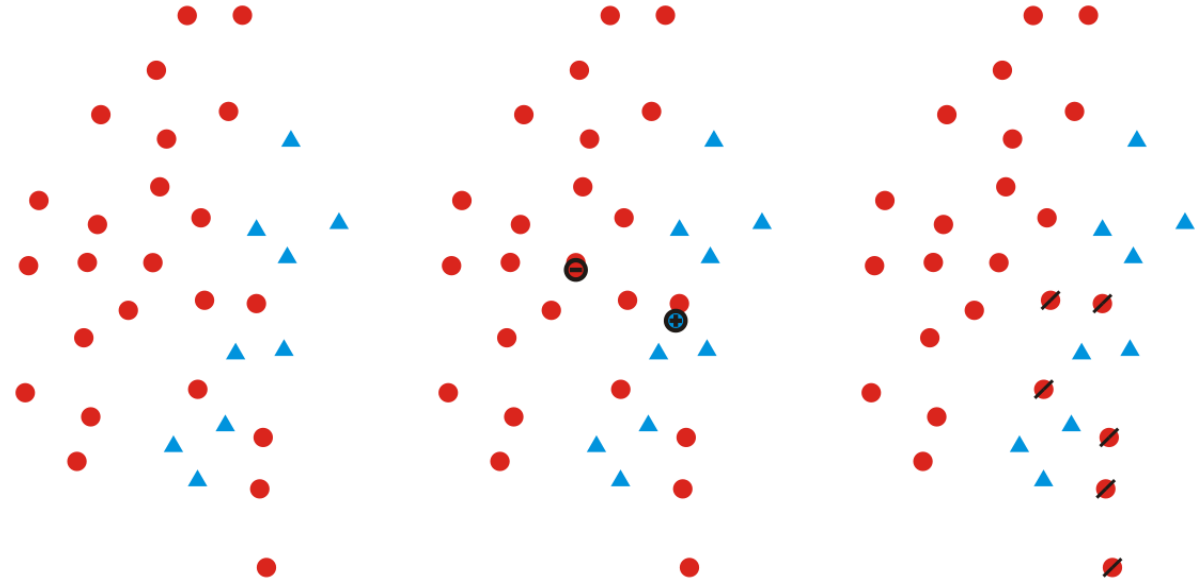
$$\text{centropos}(1,t) = \frac{\text{soma}(\text{conjpos}(1,t) \text{ até } \text{conjpos}(\text{tampos},t))}{\text{tampos}};$$

$$\text{centroneg}(1,t) = \frac{\text{soma}(\text{conjneg}(1,t) \text{ até } \text{conjneg}(\text{tamneg},t))}{\text{tamneg}};$$

fim
/* verificando se existe a presença de *outliers* no conjunto de
dados correspondentes à classe negativa. Para isso será
calculada a distância entre cada dado da classe negativa e os
centróides que representam a classe positiva e a classe
negativa */
para t ← 1 **até** tamneg **faça**
se (distEuclidiana(conjneg(t), centroneg) <
distEuclidiana(conjneg(t), centropos)) **então**
incluir conjneg (t) no novo conjunto de dados C' que formará a classe
negativa;
senão
/* conjneg (t) é considerado como *outlier* e não fará parte
do novo conjunto C' da classe negativa */
fim
fim

Algoritmo 3: Algoritmo para efetuar a sub-amostragem dos dados pertencentes à classe negativa.

culada a distância euclidiana entre cada elemento pertencente à classe negativa e os centróides, tanto positivo quanto negativo. Se a distância euclidiana em relação ao centróide negativo for menor, significa que o elemento da classe negativa está em sua posição correta. Caso a distância euclidiana em relação ao centróide positivo for menor, o elemento da classe negativa é considerado como um ruído e é removido do conjunto. O funcionamento do processo de redução é ilustrado na figura 4.1.



(a) Classes positivas e negativas

(b) Localização dos centróides

(c) Eliminação dos dados negativos que estão mais próximos do centróide positivo

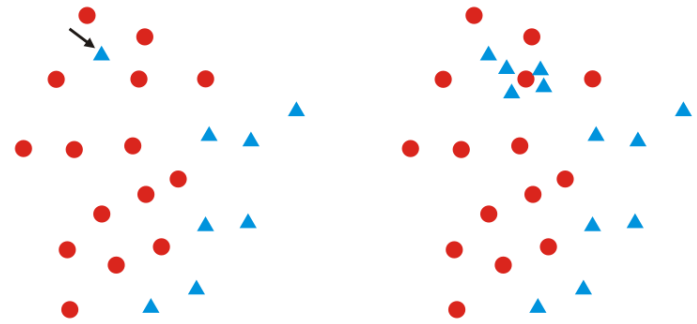
Figura 4.1: Ilustração do procedimento de localização dos centróides positivos (classe ▲) e negativos (classe ●), e eliminação dos elementos da classe negativa que são considerados como ruídos

O uso da técnica de sub-amostragem - quando feita de forma randômica - pode levar à perda de informações importantes (Chawla et al., 2004; García, Sánchez, & Mollineda, 2010). Mas quando é realizada utilizando algum tipo de heurística, esse problema pode ser superado (Gu et al., 2008; He & Garcia, 2009). O uso desta etapa é importante, pois grande parte dos ruídos é removida, conforme ilustrado na figura 4.1(c).

4.2 Super-amostragem de elementos da classe positiva

A aplicação deste passo deve ser feita após a redução da quantidade de elementos da classe negativa, técnica que foi mostrada na seção anterior (seção 4.1). Com isso é reduzida a possibilidade de criação de novos elementos "sintéticos" em regiões incorretas.

Uma das técnicas mais conhecidas para efetuar super-amostragem é a SMOTE, descrita na seção 3.4.1. Porém, caso exista algum ruído na classe negativa, ao gerar os novos elementos positivos "sintéticos", poderá aumentar a quantidade de ruídos. A figura 4.2 ilustra um caso onde foi aplicado o algoritmo SMOTE (com $k = 4$) no elemento marcado com uma seta. E uma possível configuração de novos exemplos "sintéticos" da classe positiva está representada na figura 4.2(b). Veja que neste caso particular, a quantidade de ruídos foi aumentada.



(a) Ruído na classe negativa

(b) Geração de dados "sintéticos" na região incorreta

Figura 4.2: Caso em que a geração de novos dados positivos aumenta a quantidade de ruídos

Para reduzir a possibilidade de introdução de novos ruídos, o método criado para efetuar a super-amostragem dos elementos pertencentes à classe positiva foi baseado na técnica empregada no algoritmo SMOTE, com o acréscimo de uma condição que identifica um ruído (algoritmo 4). A partir de agora, o algoritmo SMOTE com as alterações incluídas, passará a ser identificado como SMOTE-Modificado.

A alteração foi feita da seguinte forma: Dois novos parâmetros foram inseridos - *klimite* e *tam*. O primeiro indica a quantidade de vizinhos ao dado $\mathbf{x}_i$. O segundo indica a quantidade mínima de dados pertencentes à classe positiva. Se este critério for satisfeito, o

Entrada:
conjunto de m dados C = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ com $y_i \in Y = \{1, 2\}$;
/* esse k representa a porcentagem aproximada que o conjunto de dados que representa a classe positiva será replicada. Se k = 2, a quantidade de elementos que pertencem à classe positiva aumentará em aproximadamente 200% */
número de vizinhos mais próximos k;
/* klimite indica a quantidade de vizinhos ao dado $\mathbf{x}_i$ que serão avaliados */
quantidade de vizinhos klimite;
/* tam indica a quantidade mínima de vizinhos pertencentes à classe positiva */
quantidade mínima de vizinhos tam;

Saída:
/* Foi acrescido $|C'| > |C|$ porque após a aplicação de SMOTE, o conjunto da classe positiva ficará maior que seu tamanho original */
conjunto de m' dados C′ = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m'}, y_{m'})\}$ com $y_i \in Y = \{1, 2\}$ e $|C'| > |C|$;

tampos = quantidade de elementos da classe positiva;
// dim é a quantidade de atributos que o dado **x** possui
dim = dimensão de **x**;
conjpos = conjunto de dados de C que pertencem à classe positiva;
conjneg = conjunto de dados de C que pertencem à classe negativa;
// C′ é o conjunto de dados de saída que inicialmente é vazio
C′ = { };

para i ← 1 **até** tampos **faça**
  encontrar os klimite elementos mais próximos do elemento $\mathbf{x}_i$ e armazenar seus vizinhos em klimiteDados (i);
  tm = quantidade de elementos positivos presentes em klimiteDados (i);
  /* se tm < k for verdade, então significa que para o elemento $\mathbf{x}_i$ não existem os k vizinhos próximos para utilizar. Portanto, deve-se utilizar a quantidade máxima permitida que é tm */
  se (tm < k) **então**
    kaux = tm
  fim
  /* se tm ≥ tam significa que o elemento $\mathbf{x}_i$ participará do processo de geração do novo elemento sintético e ficará no conjunto final (C′) que será gerado */

```
    se (tm ≥ tam) então
        para j ← 1 até kaux faça
            para atributo ← 1 até dim faça
                dif = klimiteDados(i, atributo) − conjpos(i, atributo);
                // rand (0-1) significa que será gerado um número
                   randômico entre 0 e 1
                novoElem(i, atributo) = conjpos(i, atributo) + rand(0 − 1) × dif;
            fim
            adicionar novoElem(i) a C′;
            adicionar conjpos(i) a C′;
        fim
    senão
        /* se não satisfazer a condição (tm ≥ tam) então o elemento
           x_i participará do processo de geração do novo elemento
           sintético, mas não ficará no conjunto final (C′) que
           será gerado                                              */
        para j ← 1 até kaux faça
            para atributo ← 1 até dim faça
                dif = klimiteDados(i, atributo) − conjpos(i, atributo);
                // rand (0.7-1) significa que será gerado um número
                   randômico entre 0.7 e 1
                novoElem(i, atributo) = conjpos(i, atributo) + rand(0.7 − 1) × dif;
            fim
            adicionar novoElem(i) a C′;
        fim
    fim
fim
adicionar conjneg a C′;
```

Algoritmo 4: Algoritmo para efetuar a super-amostragem dos dados pertencentes à classe positiva.

dado participará do processo de criação do novo elemento exatamente da mesma forma descrita na seção 3.4.1. Caso contrário, ele é considerado um ruído e participará da criação do elemento "sintético" de uma forma especial. No processo de geração do novo exemplo, ao invés de multiplicar a diferença da subtração da instância $\mathbf{x}_i$ de seu vizinho mais próximo por um número randômico entre 0 e 1, será multiplicado por um número randômico entre 0.7 e 1. Isso faz com que o novo exemplo seja criado mais próximo da região onde é representada a classe positiva. E, logo em seguida, o elemento que foi considerado como um ruído será excluído da classe positiva. Isto faz com que a distribuição dos exemplos positivos seja melhor distribuída e evita o aparecimento de novos ruídos. Um exemplo pode ser visto na figura 4.3, onde o SMOTE-Modificado foi aplicado no conjunto de dados representado na figura 4.2(a). Para *klimite* = 5 e *tam* = 3, o elemento identificado pela seta (figura 4.2(a)) foi considerado um ruído. Portanto, participa da geração dos novos elementos "sintéticos", mas de forma controlada. Logo em seguida é excluído.

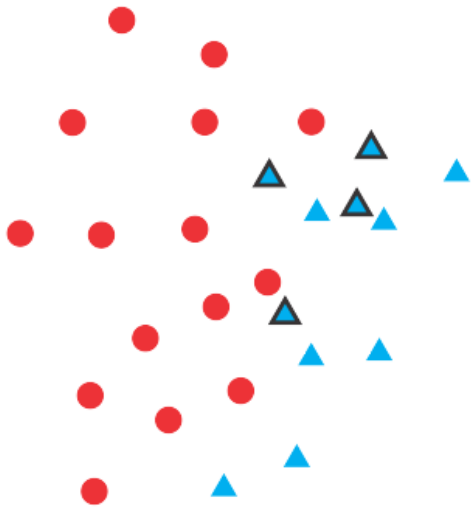


Figura 4.3: Um possível resultado da aplicação do algoritmo SMOTE no conjunto de dados positivos representados na figura 4.2(a). Note que o elemento que estava representado com uma seta foi excluído.

Este processo de geração de novos elementos "sintéticos" juntamente com a etapa anterior (seção 4.1) faz com que a proporção entre elementos positivos e negativos seja reduzida, facilitando o processo de classificação.

4.3 Aplicação do Comitê de Máquinas

Depois de feito o pré-processamento dos dados através da sub e super-amostragem, deve ser aplicado o comitê de máquinas através do algoritmo *EasyEnsemble*.

No algoritmo *EasyEnsemble* original, é sempre selecionado um subconjunto da classe negativa para unir-se com os elementos da classe positiva (segunda linha do laço repita no algoritmo 2), e logo em seguida aplicar o algoritmo *AdaBoost*. Mas em conjuntos de dados onde o desbalanceamento é pequeno, pode acontecer de após a aplicação dos passos descritos nas seções 4.1 e 4.2, a classe positiva ficar com alguns elementos a mais que a negativa. Geralmente menos de dez. Para tratar este tipo de situação, a segunda linha do laço repita no algoritmo 2 foi alterada para

se $|P| < |N|$ **então**
  Randomicamente selecione um subconjunto N_i de N, de forma que $|N_i| = |P|$;
senão
  Randomicamente selecione um subconjunto P_i de P, de forma que $|P_i| = |N|$;
fim

4.4 Considerações Finais

Neste capítulo foi apresentado o algoritmo gerado como produto desta tese. Ele utiliza como base a técnica de super-amostragem denominada SMOTE, que recebeu melhorias no processo de geração no novo elemento "sintético" e passou a ser denominado como SMOTE-Modificado e também o algoritmo de comitê de máquina *EasyEnsemble*, que também foi alterado para que tratasse casos especiais em que a proporção do número de elementos presentes na classe positiva e negativa fosse reduzida. Além disso, também foi criada uma nova proposta de sub-amostragem de dados, cuja função é reduzir o conjunto de dados que representa a classe negativa, através da eliminação de *outliers*. No próximo capítulo, este algoritmo proposto será aplicado em diversos conjuntos de dados para que sua capacidade de classificação possa ser avaliada.

Capítulo 5

Resultados Obtidos

Este capítulo aplica o algoritmo criado nesta tese em diversos conjuntos de dados e faz uma comparação com os algoritmos nos quais ele foi baseado, para provar que obtém uma taxa de acerto melhor que os algoritmos nos quais se baseou.

5.1 Forma de Avaliação

Conforme explicado anteriormente, existe uma forma especial de avaliar dados desbalanceados. Serão utilizadas as três métricas (*G-Mean*, *F-Measure* e *AUC*) apresentadas na seção 3.2. Os conjuntos de dados utilizados no experimento foram extraídos do Repositório UCI (Frank & Asuncion, 2010), com exceção do conjunto denominado *Phoneme*, que foi extraído do projeto ELENA (Aviles-Cruz, Guérin-Dugué, Voz, & Cappel, 1995)[1]. São 20 conjuntos de dados com diferentes níveis de desbalanceamento (apresentados na tabela 5.1). Para cada conjunto de dados, foram feitas 10 repetições de validação cruzada estratificada com fator 5. Logo em seguida foram calculadas as médias de *G-Mean*, *F-Measure* e *AUC*. Os algoritmos avaliados neste trabalho utilizaram como classificador base a forma de classificação denominada Máquinas de Vetores de Suporte (SVM), implementada pela ferramenta LIBSVM (Chang & Lin, 2001), por ser amplamente utilizada em diversos artigos (Manevitz & Yousef, 2002;

[1]Mais detalhes sobre os conjuntos de dados utilizados podem ser encontrados no Apêndice A

Zhang & Ren, 2008; Tang, Zhang, Chawla, & Krasser, 2009). Foram utilizados os *kernels* RBF ($e^{-\|\mathbf{x}_i-\mathbf{x}_k\|^2/2\sigma^2}$), Polinomial $\left((\mathbf{x}_i \cdot \mathbf{x}_k + 1)^p\right)$ e Linear $\left(\mathbf{x}_i \cdot \mathbf{x}_k\right)$ para todos os conjuntos de dados. Os parâmetros utilizados para o treinamento são apresentados na tabela 5.2. Na penúltima coluna (σ/p) desta tabela, caso o *kernel* utilizado for o RBF, o valor corresponderá ao parâmetro σ. Se for polinomial, o valor corresponderá ao parâmetro p. Na última coluna, o parâmetro C diz respeito à penalidade associada a variável de relaxamento ξ (equação 2.29).

Tabela 5.1: Conjuntos de dados utilizados no experimento. Para conjuntos de dados com mais de duas classes, os valores das classes entre parênteses indicam a classe positiva escolhida. As outras foram unidas e transformadas em classes negativas.

Conjunto de Dados	quant. +	quant. −
Abalone (19)	32	4145
Balance (B)	49	576
Breast	81	196
Car (3)	69	1659
CMC (2)	333	1140
Diabetes	268	500
German (2)	300	700
Glass (7)	29	185
Haberman (2)	81	225
Heart	55	212
Hepatitis (die)	32	123
Housing [20-23]	106	400
Ionosphere (bad)	126	225
Phoneme (1)	1586	3818
Sat-Image (4)	626	5809
Vehicle (1)	212	634
WDBC (M)	212	357
Wine (3)	48	130
WPBC (R)	47	151
Yeast (5)	51	1433

Tabela 5.2: Parâmetros utilizados no treinamento.

Conjunto de Dados	*kernel*	σ/p	C
Abalone (19)	RBF	12	15
Balance (B)	Polinomial	2	2
Breast	Polinomial	2	2
Car (3)	Polinomial	2	2
CMC (2)	RBF	12	15
Diabetes	Polinomial	2	2
German (2)	Polinomial	2	2
Glass (7)	Polinomial	2	2
Haberman (2)	RBF	12	15
Heart	Polinomial	2	2
Hepatitis (die)	Polinomial	2	2
Housing [20-23]	Polinomial	2	2
Ionosphere (bad)	Polinomial	2	2
Phoneme (1)	RBF	12	15
Sat-Image (4)	Polinomial	2	2
Vehicle (1)	Polinomial	2	2
WDBC (M)	Polinomial	2	2
Wine (3)	Linear	—	110
WPBC (R)	Linear	—	110
Yeast (5)	RBF	12	15

Na tabela 5.3 são mostrados os conjuntos de dados após a aplicação do algoritmo SMOTE e também o valor do parâmetro *k* utilizado na geração dos elementos "sintéticos". Na escolha do parâmetro *k* ideal, considerou-se que a classe positiva deveria ficar com a quantidade de elementos mais próxima possível da classe negativa, obedecendo o limite de que o valor de *k* deve ser no máximo 10. Pois no processo de geração de novos elementos "sintéticos", um alto valor de *k* faz com que os novos elementos criados fiquem muito próximos uns dos outros. Isso faz com que esse processo de geração de novos elementos "sintéticos" seja muito parecido com uma simples super-amostragem de forma randômica.

Tabela 5.3: Conjuntos de dados utilizados no experimento após a aplicação do algoritmo SMOTE.

Conjunto de Dados	quant. +	quant. −	k
Abalone (19)	352	4145	10
Balance (B)	539	576	10
Breast	162	196	1
Car (3)	759	1659	10
CMC (2)	999	1140	2
Diabetes	536	500	1
German (2)	600	700	1
Glass (7)	174	185	5
Haberman (2)	243	225	2
Heart	220	212	3
Hepatitis (die)	128	123	3
Housing [20-23]	424	400	3
Ionosphere (bad)	252	225	1
Phoneme (1)	3172	3818	1
Sat-Image (4)	5634	5809	8
Vehicle (1)	636	634	2
WDBC (M)	242	357	1
Wine (3)	144	130	2
WPBC (R)	141	151	2
Yeast (5)	561	1433	10

Na tabela 5.4 são apresentados os conjuntos de dados após a aplicação do algoritmo SMOTE-Modificado, juntamente com os valores de seus parâmetros k, *klimite* e *tam*. O valor de k foi exatamente o mesmo escolhido para o algoritmo SMOTE, para que a comparação entre os algoritmos fosse a mais justa possível. Para a escolha dos parâmetros *klimite* e *tam* procurou-se encontrar uma combinação que replicasse a maior quantidade de elementos possível, mas sempre levando em consideração os elementos que foram considerados ruídos, os quais são replicados de uma forma "especial" (seção 4.2). Em alguns conjuntos de dados como *Abalone*, *Balance*, *Yeast* dentre outros, a restrição imposta por *tam* foi bem branda, ou seja, o valor de *tam* foi o mais baixo possível. Fazendo com que somente os ruídos que estavam muito isolados fossem identificados. Nestes casos, a restrição não pode ser maior pois a classe positiva aumentaria muito pouco de tamanho.

Tabela 5.4: Conjuntos de dados utilizados no experimento depois de aplicadas as etapas de sub e super-amostragem do algoritmo criado nesta tese.

Conjunto de Dados	quant. +	quant. −	k	*klimite*	*tam*
Abalone (19)	209	2330	10	25	1
Balance (B)	253	408	10	20	1
Breast	106	104	1	10	3
Car (3)	649	1601	10	5	3
CMC (2)	514	604	2	15	3
Diabetes	346	332	1	5	3
German (2)	406	417	1	10	4
Glass (7)	144	163	5	5	2
Haberman (2)	132	121	2	8	2
Heart	144	148	3	5	3
Hepatitis (die)	92	84	3	6	2
Housing [20-23]	222	243	3	5	2
Ionosphere (bad)	160	158	1	5	2
Phoneme (1)	2770	2726	1	9	4
Sat-Image (4)	2945	2808	8	9	4
Vehicle (1)	312	352	2	11	2
WDBC (M)	370	323	1	7	4
Wine (3)	96	117	2	7	4
WPBC (R)	74	100	2	9	3
Yeast (5)	396	1142	10	20	1

Para a comparação estatística entre os classificadores, foram utilizadas as técnicas propostas por Demšar (2006). Primeiro deve-se realizar o Teste de Friedman (M. Friedman, 1937, 1940) e sua extensão, proposta em Iman and Davenport (1980). No Teste de Friedman os algoritmos são classificados para cada conjunto de dados separadamente, sendo que o algoritmo que obtiver o melhor desempenho terá classificação 1, o segundo melhor, classificação 2, e assim por diante. Em caso de empate, é feita uma média entre as classificações.

Considere que r_i^j seja a classificação do j-ésimo de k algoritmos no i-ésimo de N conjuntos de dados. O Teste de Friedman compara a média das classificações dos algoritmos, $R_j = \sum_i r_i^j$. Sob a hipótese nula, que considera que todos os algoritmos são equivalentes e portanto suas classificações R_j deveriam ser iguais, a estatística de Friedman (equação 5.1) é

distribuída de acordo com $\mathcal{X}_F^2$ com $k-1$ graus de liberdade.

$$\mathcal{X}_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{5.1}$$

Iman and Davenport (1980) mostraram que a estatística de Friedman ($\mathcal{X}_F^2$) é muito conservadora e propuseram uma nova estatística (equação 5.2) que é distribuída de acordo com a distribuição F com $k-1$ e $(k-1)(N-1)$ graus de liberdade.

$$F_F = \frac{(N-1)\mathcal{X}_F^2}{N(k-1)-\mathcal{X}_F^2} \tag{5.2}$$

Se a hipótese nula for rejeitada, deve-se utilizar o procedimento Post-Hoc chamado de Método de Holm (Holm, 1979). O teste estatístico para comparar o i-ésimo com o j-ésimo classificador utilizando este método utiliza a equação 5.3

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6N}}} \tag{5.3}$$

onde o valor z é utilizado para encontrar a probabilidade (p-valor) correspondente da tabela de distribuição normal, que é então comparada com o valor apropriado de α (nível de significância).

O p-valor é ordenado por $p_1, p_2, \ldots$, de tal forma que $p_1 \leq p_2 \leq \ldots \leq p_{(k-1)}$. Então cada p_i é comparado com $\frac{\alpha}{(k-i)}$ (α ajustado). Se p_1 é menor que $\frac{\alpha}{(k-1)}$, a hipótese correspondente é rejeitada e deve-se continuar e comparar p_2 com $\frac{\alpha}{(k-2)}$. Se a segunda hipótese for rejeitada, o teste continua com a terceira e assim por diante. Se a hipótese nula não for rejeitada, todas as hipóteses seguintes também não serão rejeitadas.

5.2 Resultados Obtidos

Nas figuras a seguir (figuras 5.1, 5.2 e 5.3) é feita uma comparação utilizando as métricas *G-Mean*, *F-Measure* e *AUC*, nos algoritmos SVM, SMOTE e no Algoritmo criado nesta tese (aqui será denominado de Algoritmo Composto - AlgComp). Veja que o algoritmo Alg-

Comp obteve grande parte dos melhores resultados em todos os conjuntos de dados. As tabelas com os valores que foram utilizados para a geração dos gráficos se encontram no Apêndice B.

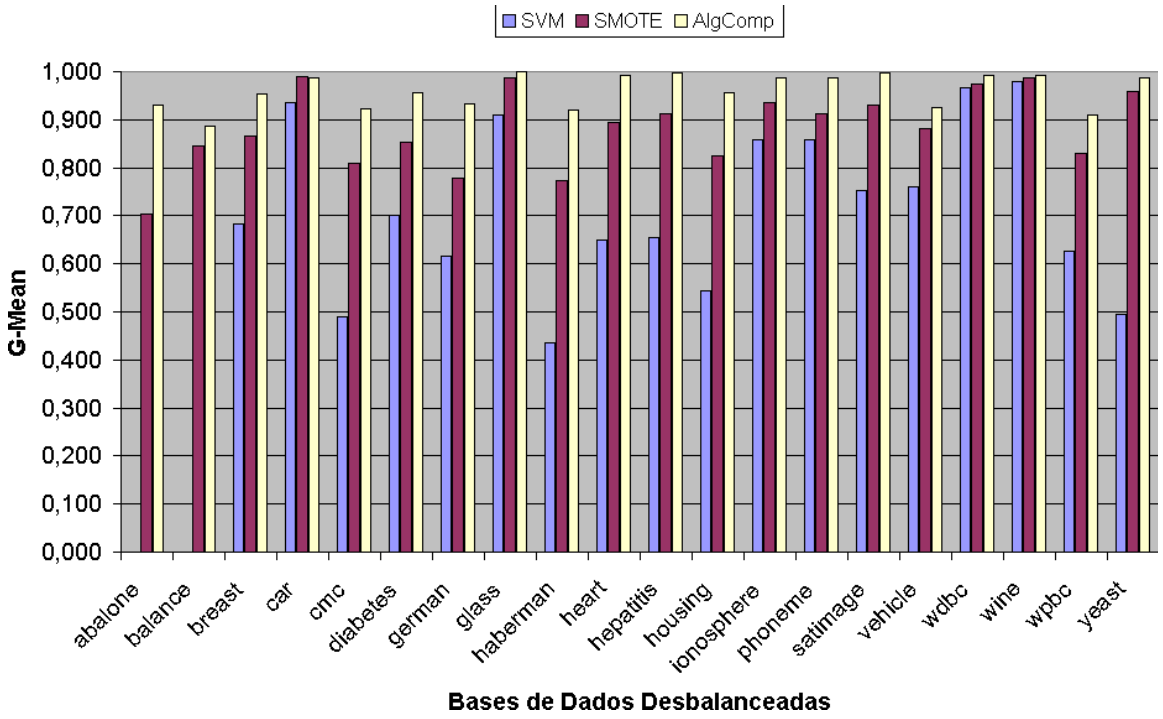


Figura 5.1: Gráfico *G-Mean*

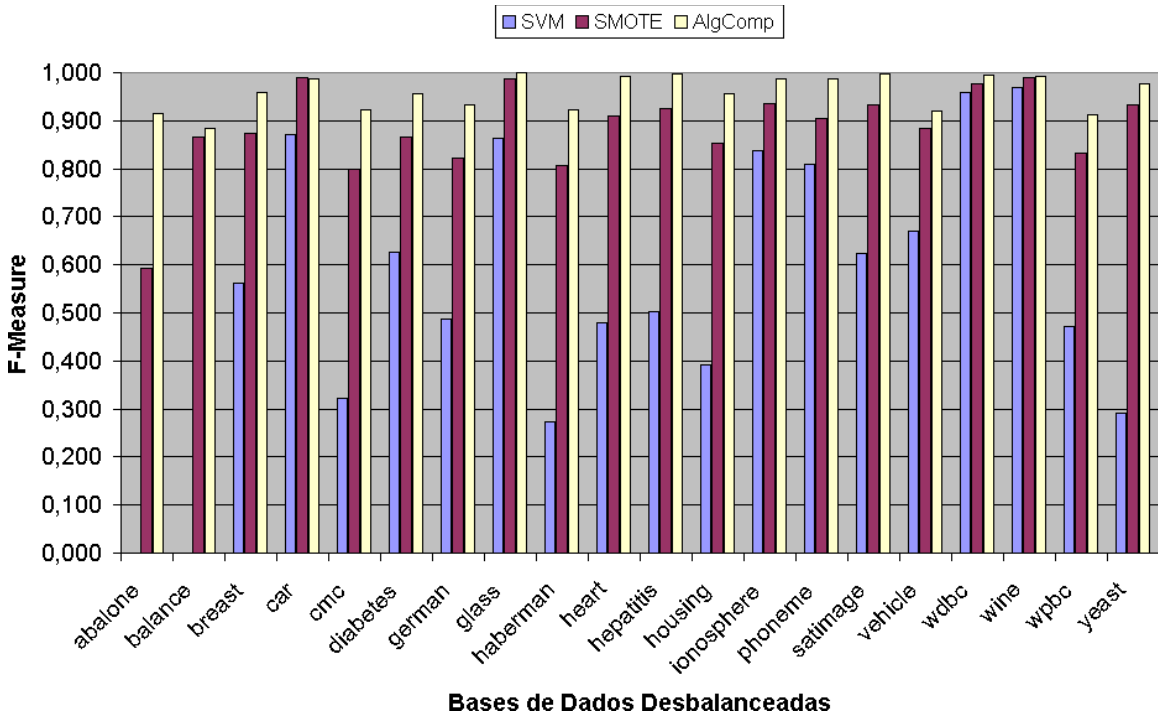


Figura 5.2: Gráfico *F-Measure*

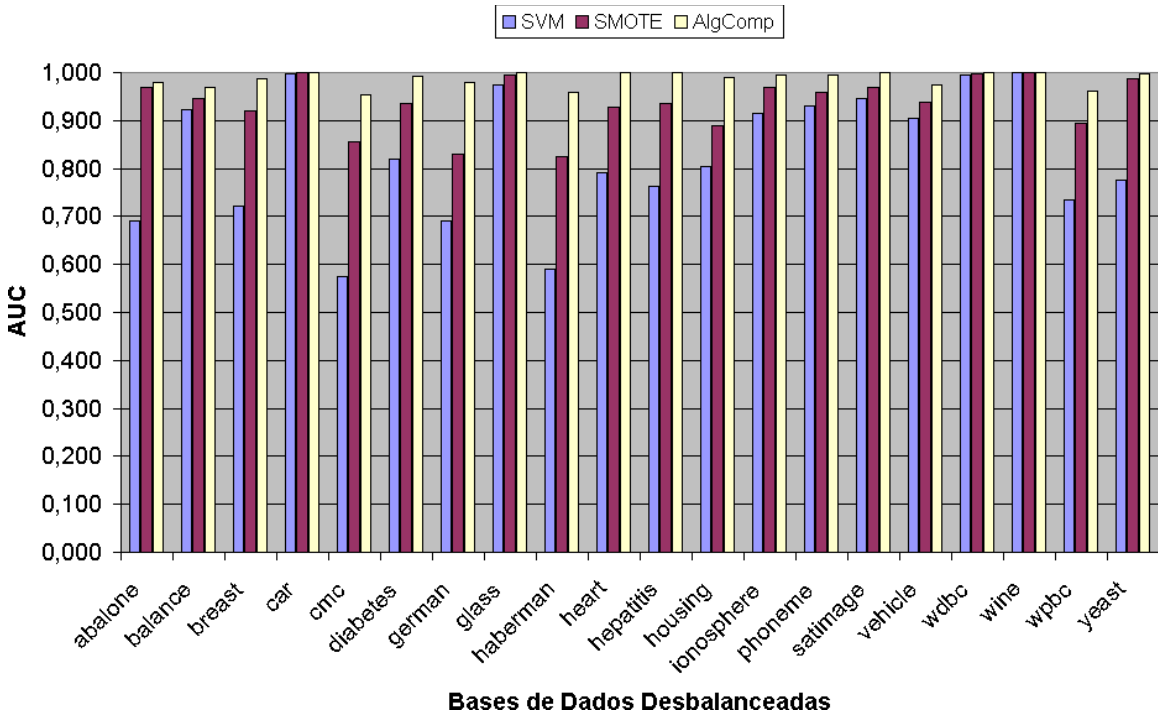


Figura 5.3: Gráfico *AUC*

Lembrando que cada métrica utilizada possui uma característica em especial. O resultado do uso da métrica *G-Mean* foca no desempenho tanto da classe positiva quanto negativa. Um bom resultado nesta métrica significa que o algoritmo não privilegia uma classe em função de outra. Analisando o resultado apresentado no gráfico da figura 5.1 pode-se perceber que o AlgComp obteve um melhor desempenho em quase todos os conjuntos de dados. Perdeu apenas para o conjunto de dados *Car*. Outra característica do gráfico é que em seis conjuntos de dados (*Car*, *Glass*, *Ionosphere*, *Phoneme*, *WDBC* e *Wine*), os três algoritmos comparados obtiveram resultados acima de 80%. Isto induz a conclusão que apesar de desbalanceados, os exemplos dos conjuntos de dados possuem apenas uma pequena área de dados em comum, mesmo no caso de um alto desbalanceamento (*Car*). Isto facilita o processo de geração do hiperplano, fazendo com que ele consiga efetuar a separação da maior parte dos dados de forma correta. A métrica *F-Measure* foca somente o desempenho da classe positiva. Então, quanto mais alto seu valor, melhor o desempenho na identificação de exemplos pertencentes à classe positiva. No gráfico da figura 5.2 pode-se perceber que o AlgComp possui um desempenho na taxa de reconhecimento melhor que os outros dois algoritmos. O objetivo da métrica *AUC* é mostrar,

na média, qual classificador de dados possui o melhor desempenho. No gráfico da figura 5.3 pode-se perceber que o desempenho médio dos classificadores nos conjuntos de dados *Car*, *Glass*, *WDBC* e *Wine* é muito próximo. Isto indica que as classes positivas e negativas estão quase 100% separadas, pois o classificador SVM, sem nenhum tratamento para lidar com classes desbalanceadas, obteve um desempenho muito próximo dos outros dois.

Na avaliação estatística, o ranking médio dos algoritmos SVM, SMOTE e AlgComp foram, respectivamente, 3, 1.95 e 1.05. O nível de significância (α) utilizado foi 0.05. No Teste de Friedman e de Iman/Daverport obteve-se, respectivamente os valores 38.1 e 381. Os valores críticos de Friedman e Iman/Daverport são 7.815 e 3.24. Como estes valores críticos são menores que os valores obtidos, a hipótese nula[2] deve ser rejeitada. Ao aplicar o procedimento post-hoc denominado método de Holm (tabela 5.5), obteve-se os resultados:

Tabela 5.5: Método de Holm

Método de Holm	z	p-valor	α ajustado	i
SVM	-6.166	≈ 0	0.025	1
SMOTE	-2.846	0.0044	0.05	2

Como os valores correspondentes de p dos algoritmos SVM e SMOTE são menores que α ajustado, estas hipóteses são rejeitadas. O que leva à conclusão que o algoritmo AlgComp é significantemente melhor que o SVM e também que o SMOTE.

5.3 Considerações Finais

Este capítulo mostrou o algoritmo criado aplicado em 20 conjuntos de dados desbalanceados. Aplicou-se três tipos de métricas e provou-se que ele possui um bom desempenho. No próximo capítulo serão feitas as conclusões finais desta tese.

[2]Lembrando que a hipótese nula considera que o desempenho dos algoritmos são semelhantes.

Capítulo 6

Conclusão

Neste trabalho foi mostrado que a classificação em conjuntos de dados desbalanceados é muito comum, e que se a ferramenta para executar esta tarefa não for bem elaborada, poderá não conseguir bons resultados. Ou conseguirá falsos resultados, através da classificação de todos os dados como pertencentes somente a uma classe. Também apresentou algumas propostas para lidar com esse problema de desbalanceamento, citando suas deficiências.

Em vista disso, esta tese propôs a criação de um algoritmo que apresentasse algumas características das propostas já existentes, mas com melhorias. Disto surgiu um algoritmo que trabalha em três fases de processamento:

1. Efetuar sub-amostragem dos elementos pertencentes à classe negativa;
2. Aplicar o algoritmo SMOTE-Modificado na classe positiva, para melhorar sua representatividade;
3. Efetuar o treinamento do conjunto de dados utilizando um comitê de máquinas aplicado através do algoritmo *EasyEnsemble*, com o classificador base SVM.

Depois de aplicado em 20 conjuntos de dados, mostrou-se eficaz, obtendo resultados melhores que os alcançados pelos algoritmos originais, nos quais o algoritmo proposto foi baseado.

Como contribuições pode-se citar um algoritmo capaz de efetuar sub-amostragem de dados com uma heurística que evita que dados importantes sejam excluídos, o algoritmo

SMOTE-Modificado que cria novos elementos “sintéticos” em regiões corretas, eliminando possíveis ruídos da classe positiva, e por fim, mostrou-se que é possível combinar algoritmos existentes para a criação outro com o desempenho melhor que os originais.

Uma das dificuldades deste trabalho foi na definição dos parâmetros ideais para a aplicação do algoritmo de sub-amostragem e super-amostragem.

Apesar da atualidade do assunto de tratamento de conjuntos de dados desbalanceados, para que o processo de classificação seja mais eficiente, detectou-se a pouca disponibilidade de conjuntos de dados onde já existe o desbalanceamento. Por isso houve a necessidade de fazer com que alguns conjuntos de dados ficassem desbalanceados, através da escolha de uma classe para representar o conjunto de dados positivos e o agrupamento das restantes para representar a classe negativa. Vale lembrar que isto não foi feito de maneira aleatória. Procurou-se seguir o padrão (escolha da classe para representar a classe positiva) adotado em outros artigos que abordam o tema.

Como trabalho futuro, sugere-se: desenvolver uma forma automatizada para encontrar os parâmetros ótimos nos processos de sub e super-amostragem; criar conjuntos de dados desbalanceados artificiais, para que posteriores propostas tenham um conjunto de teste maior; comparar o AlgComp com outros algoritmos de aprendizagem supervisionada, como Redes Neurais Artificiais, com relação aos seguintes aspectos: tempo de processamento, taxa de acerto, e avaliar se o desempenho de um algoritmo se mantém para diferentes conjuntos de dados.

Referências Bibliográficas

Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *Machine learning: ECML 2004, 15th european conference on machine learning, pisa, italy, september 20-24, 2004, proceedings* (pp. 39–50). Springer.

Arora, J. S. (2004). *Introduction to optimum design* (2nd ed.). Elsevier Academic Press.

Aviles-Cruz, C., Guérin-Dugué, A., Voz, J. L., & Cappel, D. V. (1995). *Deliverable r3-b1-p, task b1: Databases* (ESPRIT Basic Research Project No. 6891). The ELENA Project.

Bennett, K. P., & Campbell, C. (2000). Support vector machines: Hype or hallelujah? *SIGKDD Explorations*.

Bertsekas, D. P. (1999). *Nonlinear programming* (2nd ed.). Athena Scientific.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT: Proceedings of the Workshop on Computational Learning Theory.* Morgan Kaufmann.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., et al. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *International conference on pattern recognition* (pp. 77–82).

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159.

Breiman, L. (1996, Ago.). Bagging predictors. *Machine Learning*, *24*, 123–140.

Burges, C. J. C. (1998, Jun.). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Castro, C. L., Carvalho, M. A., & Braga, A. P. (2009). An Improved Algorithm for SVMs Classification of Imbalanced Data Sets. In D. Palmer-Brown, C. Draganova, E. Pimenidis, & H. Mouratidis (Eds.), *Engineering applications of neural networks* (Vol. 43, p. 108-118). Springer Berlin Heidelberg.

Chan, P., Fan, W., Prodromidis, A., & Stolfo, S. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, *14*, 67–74.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: a library for support vector machines [Computer software manual]. (Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`)

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, *16*, 321–357.

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations*, *6*(1), 1–6.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In *Principles and practice of knowledge discovery in databases* (pp. 107–119). Springer.

Chekassky, V., & Mulier, F. (2007). *Learning from data - concepts, theory, and methods* (2nd ed.). Wiley.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, U.K.: Cambridge University Press.

Cunningham, P. (2000, February). *Overfitting and diversity in classification ensembles based on feature selection* (Tech. Rep.).

Demšar, J. (2006, December). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Ditzler, G., Muhlbaier, M., & Polikar, R. (2010). Incremental Learning of New Classes in Unbalanced Datasets: Learn++.UDNC. In N. Gayar, J. Kittler, & F. Roli (Eds.), *Multiple classifier systems* (Vol. 5997, pp. 33–42). Springer Berlin Heidelberg.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Kdd* (pp. 155–164).

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). New York: Wiley.

Fan, X., Zhang, G., & Xia, X. (2008). Performance Evaluation of SVM in Image Segmentation. In *IWSCA '08: Proceedings of the 2008 IEEE International Workshop on Semantic Computing and Applications.*

Fawcett, T. (2004). *ROC Graphs: Notes and Practical Considerations for Researchers* (Tech. Rep. No. HPL-2003-4). Available from `http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf`

Frank, A., & Asuncion, A. (2010). *UCI machine learning repository.* Available from `http://archive.ics.uci.edu/ml`

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Icml* (pp. 148–156).

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Friedman, J. H. (1996). *Another approach to polychotomous classification* (Tech. Rep.). Stanford, CA: Department of Statistics, Stanford University. Available from `http://www-stat.stanford.edu/~jhf/ftp/poly.ps.Z`

Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, *32*(200), 675–701.

Friedman, M. (1940). A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, *11*(1), 86–92.

García, V., Sánchez, J. S., & Mollineda, R. A. (2010). Exploring the performance of resampling strategies for the class imbalance problem. In *Proceedings of the 23rd international conference on industrial engineering and other applications of applied intelligent systems - volume part i* (pp. 541–549). Berlin, Heidelberg: Springer-Verlag.

Gu, Q., Cai, Z., Zhu, L., & Huang, B. (2008). Data mining on imbalanced data sets. In *Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering* (pp. 1020–1024). Washington, DC, USA: IEEE Computer Society.

Gunn, S. R. (1998). *Support vector machines for classification and regression* (Technical Report, School of Electronics and Computer Science). Southampton, U.K.: University of Southampton. Available from `http://www.ecs.soton.ac.uk/~srg/publications/pdf/SVM.pdf`

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

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Prentice Hall.

He, H., & Garcia, E. (2009, Set.). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, *21*(9), 1263–1284.

Hearst, M. A. (1998). Trends & controversies: Support vector machines. *IEEE Intelligent Systems*, *13*(4), 18–28.

Herbrich, R. (2001). *Learning kernel classifiers: Theory and algorithms*. The MIT Press.

Holm, S. (1979). A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, *6*, 65–70.

Hsu, C.-W., & Lin, C.-J. (2002, Mar.). A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, *13*(2), 415–425.

Hulley, G., & Marwala, T. (2007). Evolving classifiers: Methods for incremental learning. *Computing Research Repository*, *abs/0709.3965*.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the friedman statistic. *Communication in Statistics - Theory and Methods*, *9*(6), 571–595.

Ivanciuc, O. (2007). Applications of support vector machines in chemistry. In K.B.Lipkowitz & T.R.Cundari (Eds.), (Vol. 23, pp. 291–400). Wiley-VCH.

Japkowicz, N. (2000). Learning from imbalanced data sets: A comparison of various strategies. In *Proceedings of Learning from Imbalanced Data Sets, Papers from the AAAI workshop, Technical Report ws-00-05* (pp. 10–15). AAAI Press.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In (pp. 169–184). Cambridge, MA, USA: MIT Press.

Joshi, M. V. (2002). *Learning classifier models for predicting rare phenomena*. Unpublished doctoral dissertation, University of Minnesota.

Kearns, M., & Valiant, L. (1994, Jan.). Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, *41*, 67–95.

Kidera, T., Ozawa, S., & Abe, S. (2006). An incremental learning algorithm of ensemble classifier systems. In *Neural networks, 2006. ijcnn '06. international joint conference on* (pp. 3421–3427).

Kim, H.-C., Pang, S., Je, H.-M., Kim, D., & Bang, S.-Y. (2003). Constructing support vector machine ensemble. *Pattern Recognition*, *36*, 2757–2767.

Kubat, M., Holte, R. C., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, *30*, 195–215.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Proc. 14th international conference on machine learning* (pp. 179–186). Morgan Kaufmann.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 3–12). Springer-Verlag New York, Inc.

Li, S., Fu, X., & Yang, B. (2008). Nonsubsampled contourlet transform for texture classifications using support vector machines. In *ICNSC '08: IEEE International Conference on Networking, Sensing and Control.*

Lima, N. H. C., Neto, A. D. D., & Melo, J. D. de. (2009). Creating an ensemble of diverse support vector machines using adaboost. In *Proceedings of international joint conference on neural networks.*

Lin, Y., Lee, Y., & Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning*, *46*, 191–202.

Liu, T.-Y. (2009). Easyensemble and feature selection for imbalance data sets. In *Ijcbs '09: Proceedings of the 2009 international joint conference on bioinformatics, systems biology and intelligent computing* (pp. 517–520). IEEE Computer Society.

Liu, X.-Y., Wu, J., & Zhou, Z.-H. (2009, Abr.). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Manevitz, L. M., & Yousef, M. (2002, March). One-Class SVMs for Document Classification. *Journal of Machine Learning Research*, *2*, 139–154.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, *209*, 415–446.

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., & Schölkopf, B. (2001, Mar.). An introduction to kernel-based learning algorithms. *IEEE Transaction on Neural Networks*, *12*(2).

Moraes Lima, C. A. de. (2004). *Comitê de máquinas: Uma abordagem unificada empregando máquinas de vetores-suporte*. Tese de doutorado, Universidade Federal de Campinas.

Morik, K., Brockhausen, P., & Joachims, T. (1999). Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of the sixteenth international conference on machine learning* (pp. 268–277). Morgan Kaufmann Publishers Inc.

Osuna, E., Freund, R., & Girosi, F. (1997a). *Support vector machines: Training and applications* (Tech. Rep.). Cambridge, MA, USA.

Osuna, E., Freund, R., & Girosi, F. (1997b). Training support vector machines: An application to face detection. In *Ieee conference on computer vision and pattern recognition.*

Platt, J. C., Cristianini, N., & Shawe-Taylor, J. (1999). Large margin DAGs for multiclass classification. In *Neural information processing systems* (Vol. 12, pp. 547–553). The MIT Press.

Polikar, R. (2006). Ensemble Based Systems in Decision Making. *IEEE Circuits and Systems*

Magazine, *6*(3), 21–45.

Polikar, R., Udpa, L., Honavar, V., & Udpa, S. (2000). Learn++: An incremental learning algorithm for multilayer perceptron networks. *IEEE Transactions on System, Man and Cybernetics*, *31*, 497–508.

Provost, F. (2000). *Machine learning from imbalanced data sets 101.* (Invited paper for the AAAI'2000 Workshop on Imbalanced Data Sets)

Provost, F. J., & Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Knowledge discovery and data mining* (pp. 43–48).

Rao, R. B., Krishnan, S., & Niculescu, R. S. (2006). Data mining for improved cardiac care. *SIGKDD Explorations*, *8*(1), 3–10.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels - suppor vector machines, regularization, optimization, and beyond*. The MIT Press.

Schölkopf, B. (1997). *Support vector learning*. Tese de doutorado, Technische Universität Berlin.

Smola, A. J., Bartlett, P. L., Schölkopf, B., & Schuurmans, D. (2000). Introduction to large margin classifiers. In A. J. Smola, P. L. Bartlett, B. Schölkopf, & D. Schuurmans (Eds.), *Advances in large margin classifiers* (pp. 1–29). The MIT Press.

Sun, Y. M., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007, Dez.). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, *40*(12), 3358–3378.

Tang, Y., Zhang, Y.-Q., Chawla, N. V., & Krasser, S. (2009, Fev.). SVMs Modeling for Highly Imbalanced Classification. *Transactions on Systems, Man and Cybernetics - Part B*, *39*, 281–288.

Tao, D., Tang, X., & Li, X. (2008, Jan.). Which Components are Important for Interactive Image Searching? *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(1), 3–11.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley & Sons, Inc.

Vapnik, V. N. (1999). *The nature of statistical learning theory* (2nd ed.). Springer-Verlag New York.

Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the Sensitivity of Support Vector Machines. In *Proceedings of the International Joint Conference on AI* (pp. 55–60).

Wolpert, D., & Macready, W. (1997, Abr.). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Wu, G., & Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *In icml 2003 workshop on learning from imbalanced data sets* (pp. 49–56).

Zhang, X., & Ren, F. (2008). Improving SVM Learning Accuracy with Adaboost. In *Proceedings of the Fourth International Conference on Natural Computation* (Vol. 3, pp. 221–225). Washington, DC, USA: IEEE Computer Society.

Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., & Muller, K. R. (2000). Engineering support vector machine kernels that recognize translation initiation sites. *BIOINF: Bioinformatics*, *16*(9), 799-807.

Apêndice A

Descrição dos Conjuntos de Dados Utilizados

Neste apêndice são apresentados mais detalhes sobre os conjuntos de dados utilizados no capítulo 5 (Resultados Obtidos).

A.1 *Abalone*

O objetivo deste conjunto de dados é determinar a idade de um tipo de molusco comestível, que varia de 1 a 29 anos. Os atributos para determinar a idade incluem o sexo do molusco e medições como seu diâmetro, a altura, o peso. Ao todo são 8 atributos. A classe que representa a idade 19 foi escolhida para representar a classe positiva (32 instâncias). As outras classes foram agrupadas para formar a classe negativa, gerando um total de 4145 instâncias.

A.2 *Balance*

Este conjunto de dados foi gerado para modelar os resultados de experimentos psicológicos. A criança é convidada a prever o resultado da inserção de uma certa quantidade de pesos em diferentes distâncias no lado esquerdo ou direito de uma gangorra. Logo, as possíveis

posições (classes) são: “inclinada para a direita” (*right*), “inclinada para a esquerda” (*left*) ou “balanceada” (*balanced*). Existem 4 atributos que são: quantidade de pesos na esquerda, distância na esquerda, quantidade de pesos na direita e distância na direita. A classe balanceada foi selecionada para representar a classe positiva com 49 instâncias. O restante das instâncias foram agrupadas para formar a classe negativa, gerando um total de 576 instâncias.

A.3 *Breast*

Este conjunto de dados mostra casos recorrentes de cancer de mama em mulheres. O conjunto é composto de 277 casos (instâncias). Destas 81 são casos “recorrentes” (*recurrence-events*) e 196 casos “não-recorrentes” (*no-recurrence-events*). Existem 9 atributos que contém informação a respeito das características das pacientes como sexo, idade de aparecimento da menopausa e também a respeito do nódulo encontrado como tamanho e localização do tumor.

A.4 *Car*

Este conjunto apresenta dados para a avaliação de carros. Os atributos constituem de características dos carros, como o seu preço de compra e os seus níveis de conforto e de segurança. Ao todo são 6 atributos a partir dos quais um índice de aceitação é atribuído, que pode ser: “não aceitável” (*unacceptable*), “aceitável” (*acceptable*), “bom” (*good*) ou “muito bom” (*very good*). Dessas quatro, a classe “bom” que possui 69 instâncias foi escolhida para representar a classe positiva e as outras classes, que juntas possuem 1659 instâncias, foram agrupadas para formar a classe negativa.

A.5 CMC (*Contraceptive Method Choice*)

O objetivo deste conjunto de dados é prever a escolha do atual método contraceptivo em mulheres casadas que não estavam grávidas ou não sabiam que estavam grávidas na época

da entrevista. As possíveis previsões são: “não usam” (*no use*), “métodos a longo prazo” (*long-term methods*), “métodos a curto prazo” (*short-term methods*), que foram levantadas a partir de 9 atributos baseados nas características sócio-econômicas e demográficas da mulher. “Métodos a longo prazo” foi selecionado para representar a classe positiva com 333 instâncias e as outras duas possíveis previsões foram unidas para representar a classe negativa com 1140 instâncias.

A.6 *Diabetes*

O objetivo deste conjunto de dados é diagnosticar se a paciente possui ou não diabetes. Logo os possíveis resultados (classes) deste conjunto são: 0 (teste negativo para diabetes) e 1 (teste positivo para diabetes). São necessários 8 atributos para obter o diagnóstico. Dentre eles, idade da mulher, número de vezes que engravidou, pressão sanguínea e outros. A classe 1 é selecionada para representar a classe positiva com 268 instâncias, enquanto que a classe negativa possui 500 instâncias.

A.7 *German*

Este conjunto de dados classifica o tipo de risco de concessão de crédito para as pessoas, através de um conjunto de 24 atributos (idade, tipo de emprego, sexo, motivo da solicitação de crédito, dentre outros) como sendo uma “concessão de crédito de risco” (representada pelo valor 2) ou “não” (representada pelo valor 1). As instâncias onde as características geram uma “concessão de crédito de risco” representam a classe positiva com 300 instâncias e as instâncias onde “não representam uma concessão de crédito de risco” representam a classe negativa com 700 instâncias.

A.8 *Glass*

O objetivo deste conjunto de dados é classificar vidros entre seis possíveis tipos: "vidro de edifício processado" (*building window float processed*), "vidro de edifício não processado" (*building window non float processed*), "vidro de carro processado" (*vehicle window float processed*), "recipiente" (*container*), "louça" (*tableware*) e "farol" (*headlamps*). Os atributos são características físico-químicas dos vidros, como o seu índice de refração e a porcentagem de elementos químicos contidos nos mesmos. Ao todo são 10 atributos. O vidro tipo "farol", que possui 29 instâncias representa a classe positiva e os outros tipos de vidros foram unidos para representar a classe negativa, que terá 185 instâncias.

A.9 *Haberman*

Conjunto de dados que contém casos de estudo realizado sobre a sobrevida dos pacientes submetidos à cirurgia de câncer de mama. Para isso foram necessários 3 atributos: idade do paciente na época da operação, Ano do paciente da operação (obtido através do cálculo *Ano* − 1900) e número de nódulos encontrados. A classe 1 representa os pacientes que sobreviveram 5 anos ou mais. A classe 2 representa os pacientes que faleceram dentro de 5 anos. Esta classe representa as classes positivas com 81 instâncias e a classe 1 representa as negativas com 225 instâncias.

A.10 *Heart*

Conjunto de dados que fornece o diagnóstico através de imagens obtidas por tomografia computada por emissão de fóton único (do inglês SPECT). O paciente é classificado em uma de duas possíveis classes: 0 (normal), que representará a classe positiva com 55 instâncias e 1 (anormal), que representará a classe negativa com 212 instâncias. Foram extraídas 44 características dos estudos de SPECT cardíaco. Cada característica é um número que mede uma quantidade radioativa que representa a perfusão no músculo do ventrículo esquerdo em

específicas regiões de interesse.

A.11 *Hepatitis*

(die) Este conjunto de dados diz respeito a diversos sintomas - idade, sexo, presença da anorexia, ocorrência de mal-estar, e outros - (ao todo são 19 atributos) de pessoas com hepatite que "faleceu" (*die*) ou "não faleceu" (*live*). A classe positiva *die* possui 32 instâncias enquanto que a classe negativa *live* possui 123 instâncias.

A.12 *Housing*

Conjunto de dados onde é avaliado o valor médio (em 1000 dólares) das residências dos moradores dos subúrbios de Boston. Um total de 13 atributos (proporção entre professor e aluno por região, taxa de crime per capita por região, distância aos 5 centros de empregos de Boston, entre outros) são utilizados para esta avaliação. As residências com valor médio no intervalo $[20, 23]$ representam a classe positiva com 106 instâncias. O restante compõem a classe negativa com 400 instâncias.

A.13 *Ionosphere*

(bad) Conjunto de dados que classifica os dados de um radar da ionosfera, cujo objetivo é analisar os elétrons livres. Foram utilizados 34 atributos. As possíveis análises era "bom" (*good*) ou "ruim" (*bad*). A primeira significa que houve evidência de algum tipo de estrutura na ionosfera, enquanto que a última significa que não houve evidência. *Good* representa a classe negativa com 225 instâncias e *bad* representa a classe positiva com 126 instâncias.

A.14 *Phoneme*

Conjunto de dados do projeto ELENA (Aviles-Cruz et al., 1995) de um problema cujo objetivo é o reconhecimento de fonemas, consistindo da distinção entre um fonema "nasal" e "oral". Os fonemas orais representam a classe positiva com 1586 instâncias, enquanto que os nasais representam a classe negativa com 3818 instâncias. Cada um possui 5 atributos (a amplitude normalizada das 5 primeiras harmônicas).

A.15 *Sat-Image*

Este conjunto de dados contém fragmentos de 3x3 pixels de parte de uma imagem de satélite com 82x100 pixels. Deve-se classificar o pixel central de cada um desses fragmentos em uma de seis categorias, as quais descrevem o solo ou a vegetação presente na área. São elas: "solo vermelho" (*red soil*), "colheita de algodão" (*cotton crop*), "solo cinza" (*grey soil*), "solo cinza umedecido" (*damp grey soil*), "solo com resíduos de vegetação" (*soil with vegetation stubble*) e "solo cinza bastante umedecido"(*very damp grey soil*). Para cada pixel, tem-se quatro valores de bandas espectrais. Cada fragmento é então constituído de $9 \times 4 = 36$ atributos numéricos. A classe positiva "solo cinza umedecido" possui 626 instâncias. As outras classes foram agrupadas para formar a classe negativa com 5809 instâncias.

A.16 *Vehicle*

O objetivo deste conjunto de dados é identificar a silhueta entre 4 tipos de veículos: um "ônibus de dois andares" (*double decker bus*), uma "Van Chevrolet" (*Chevrolet van*), um "Saab 9000" e um "Opel Manta 400". Os atributos são um conjunto de características extraídas da silhueta, tais como: comprimento máximo da relação entre largura e altura, taxa de dispersão, dentre outros. Ao todo são 18 atributos. O veículo "Opel" representa a classe positiva com 212 instâncias, enquanto que os outros tipos de veículos foram agrupados para formar a classe negativa com 634 instâncias.

A.17 WDBC (*Wisconsin Diagnostic Breast Cancer*)

Este conjunto de dados diagnostica um tumor "maligno" (M) ou "benigno" (B) através de 30 características (atributos) que são obtidos a partir de uma imagem digitalizada de uma aspiração com agulha fina (*Fine-Needle Aspiration*) de uma pequena massa da mama. Estas características descrevem o núcleo da célula presente na imagem. O tumor "maligno" representa a classe positiva com 212 instâncias, enquanto que o "benigno" representa a classe negativa com 357 instâncias.

A.18 *Wine*

Conjunto de dados que são resultados de análises químicas de vinhos produzidos na mesma região da Itália, mas com matéria prima derivada de 3 diferentes plantações (classe 1, classe 2 e classe 3). A análise determina a quantidade de 13 elementos (13 atributos) encontrados em cada um dos 3 tipos de vinhos. A classe 3 representa a classe positiva com 48 instâncias, enquanto que as outras duas são unidas para formar a classe negativa com 130 instâncias.

A.19 WPBC (*Wisconsin Prognostic Breast Cancer*)

Conjunto de dados onde cada instância representa o acompanhamento de um caso de câncer de mama, informando se ocorreu "recorrência" (R) ou "não recorrência" (N) do tumor. Ao todo são 33 características (33 atributos) coletadas, sendo que 30 são as mesma do conjunto de dados WDBC (seção A.17). As outras 3 são: tempo em que está livre do tumor caso a classe seja "não recorrência" ou tempo em que o tumor voltou a aparecer caso a classe seja "recorrência", tamanho do tumor retirado e número de nódulos observados na época da cirurgia. Os casos onde ocorreu "recorrência" representam a classe positiva com 47 instâncias e os que ocorreram "não recorrência" representam a classe negativa com 151 instâncias.

A.20 *Yeast*

Conjunto de dados onde proteínas de leveduras são classificadas em 10 classes: "CYT" (*cytosolic or cytoskeletal*), "NUC" (*nuclear*), "MIT" (*mitochondrial*), "ME3" (*membrane protein, no N-terminal signal*), "ME2" (*membrane protein, uncleaved signal*), "ME1" (*membrane protein, cleaved signal*), "EXC" (*extracellular*), "VAC" (*vacuolar*), "POX" (*peroxisomal*), "ERL" (*endoplasmic reticulum lumen*). Para cada tipo de proteína foram necessárias 8 características (8 atributos) obtidas através de seqüências de aminoácido. A proteína "ME2" foi selecionada para representar a classe positiva com 51 instâncias. O restante foi agrupado para representar a classe negativa com 1433 instâncias.

Apêndice B

Tabelas com os Resultados da Execução do Algoritmo Composto

Neste apêndice são apresentadas as tabelas com os resultados obtidos da execução do algoritmo composto em 20 conjuntos de dados com diferentes níveis de desbalanceamento, apresentados no Apêndice A.

B.1 *G-Mean*

Tabela com o resultado da métrica *G-Mean*.

Tabela B.1: Comparação dos valores do *G-Mean* dos conjuntos de dados avaliados

G-Mean	SVM	SMOTE	AlgComp
Abalone (19)	$0,000 \pm 0,000$	$0,703 \pm 0,007$	$0,930 \pm 0,005$
Balance (B)	$0,000 \pm 0,000$	$0,846 \pm 0,005$	$0,886 \pm 0,004$
Breast	$0,683 \pm 0,039$	$0,866 \pm 0,011$	$0,954 \pm 0,003$
Car (3)	$0,935 \pm 0,006$	$0,989 \pm 0,001$	$0,988 \pm 0,001$
CMC (2)	$0,490 \pm 0,013$	$0,809 \pm 0,005$	$0,924 \pm 0,004$
Diabetes	$0,700 \pm 0,009$	$0,853 \pm 0,007$	$0,957 \pm 0,004$
German (2)	$0,616 \pm 0,018$	$0,778 \pm 0,009$	$0,932 \pm 0,007$
Glass (7)	$0,910 \pm 0,014$	$0,987 \pm 0,004$	$1,000 \pm 0,000$
Haberman (2)	$0,437 \pm 0,024$	$0,772 \pm 0,014$	$0,921 \pm 0,008$
Heart	$0,649 \pm 0,049$	$0,894 \pm 0,008$	$0,992 \pm 0,002$
Hepatitis (die)	$0,656 \pm 0,030$	$0,912 \pm 0,010$	$0,998 \pm 0,005$
Housing [20-23]	$0,545 \pm 0,030$	$0,825 \pm 0,008$	$0,955 \pm 0,008$
Ionosphere (bad)	$0,858 \pm 0,009$	$0,934 \pm 0,013$	$0,987 \pm 0,004$
Phoneme (1)	$0,860 \pm 0,003$	$0,912 \pm 0,001$	$0,986 \pm 0,001$
Sat-Image (4)	$0,753 \pm 0,003$	$0,930 \pm 0,001$	$0,997 \pm 0,000$
Vehicle (1)	$0,760 \pm 0,009$	$0,881 \pm 0,003$	$0,924 \pm 0,005$
WDBC (M)	$0,967 \pm 0,003$	$0,974 \pm 0,004$	$0,993 \pm 0,003$
Wine (3)	$0,978 \pm 0,006$	$0,987 \pm 0,004$	$0,992 \pm 0,002$
WPBC (R)	$0,627 \pm 0,044$	$0,831 \pm 0,008$	$0,910 \pm 0,010$
Yeast (5)	$0,495 \pm 0,046$	$0,959 \pm 0,004$	$0,987 \pm 0,002$

B.2 *F-Measure*

Tabela com o resultado da métrica *F-Measure*.

Tabela B.2: Comparação dos valores do *F-Measure* dos conjuntos de dados avaliados

F-Measure	SVM	SMOTE	AlgComp
Abalone (19)	$0,000 \pm 0,000$	$0,594 \pm 0,008$	$0,916 \pm 0,006$
Balance (B)	$0,000 \pm 0,000$	$0,866 \pm 0,004$	$0,884 \pm 0,004$
Breast	$0,562 \pm 0,034$	$0,873 \pm 0,012$	$0,959 \pm 0,006$
Car (3)	$0,887 \pm 0,009$	$0,989 \pm 0,001$	$0,988 \pm 0,001$
CMC (2)	$0,323 \pm 0,015$	$0,799 \pm 0,006$	$0,921 \pm 0,005$
Diabetes	$0,626 \pm 0,012$	$0,866 \pm 0,007$	$0,957 \pm 0,004$
German (2)	$0,487 \pm 0,022$	$0,822 \pm 0,006$	$0,933 \pm 0,007$
Glass (7)	$0,863 \pm 0,024$	$0,987 \pm 0,004$	$1,000 \pm 0,000$
Haberman (2)	$0,274 \pm 0,025$	$0,807 \pm 0,010$	$0,923 \pm 0,008$
Heart	$0,479 \pm 0,056$	$0,910 \pm 0,006$	$0,992 \pm 0,003$
Hepatitis (die)	$0,502 \pm 0,033$	$0,925 \pm 0,008$	$0,998 \pm 0,005$
Housing [20-23]	$0,392 \pm 0,035$	$0,852 \pm 0,006$	$0,956 \pm 0,008$
Ionosphere (bad)	$0,837 \pm 0,011$	$0,935 \pm 0,014$	$0,988 \pm 0,003$
Phoneme (1)	$0,810 \pm 0,004$	$0,904 \pm 0,001$	$0,987 \pm 0,001$
Sat-Image (4)	$0,625 \pm 0,005$	$0,932 \pm 0,001$	$0,997 \pm 0,000$
Vehicle (1)	$0,669 \pm 0,011$	$0,884 \pm 0,003$	$0,920 \pm 0,005$
WDBC (M)	$0,960 \pm 0,005$	$0,976 \pm 0,003$	$0,994 \pm 0,002$
Wine (3)	$0,968 \pm 0,010$	$0,989 \pm 0,003$	$0,994 \pm 0,002$
WPBC (R)	$0,472 \pm 0,052$	$0,833 \pm 0,010$	$0,913 \pm 0,011$
Yeast (5)	$0,291 \pm 0,050$	$0,932 \pm 0,006$	$0,978 \pm 0,003$

B.3 *AUC*

Tabela com o resultado da métrica *AUC*.

Tabela B.3: Comparação dos valores do *AUC* dos conjuntos de dados avaliados

AUC	SVM	SMOTE	AlgComp
Abalone (19)	$0,690 \pm 0,020$	$0,969 \pm 0,002$	$0,979 \pm 0,003$
Balance (B)	$0,921 \pm 0,010$	$0,945 \pm 0,008$	$0,969 \pm 0,003$
Breast	$0,721 \pm 0,022$	$0,921 \pm 0,012$	$0,987 \pm 0,003$
Car (3)	$0,996 \pm 0,001$	$1,000 \pm 0,000$	$1,000 \pm 0,000$
CMC (2)	$0,575 \pm 0,014$	$0,856 \pm 0,006$	$0,955 \pm 0,003$
Diabetes	$0,821 \pm 0,007$	$0,936 \pm 0,005$	$0,993 \pm 0,001$
German (2)	$0,690 \pm 0,013$	$0,831 \pm 0,005$	$0,980 \pm 0,002$
Glass (7)	$0,975 \pm 0,009$	$0,995 \pm 0,001$	$1,000 \pm 0,000$
Haberman (2)	$0,590 \pm 0,032$	$0,825 \pm 0,009$	$0,960 \pm 0,006$
Heart	$0,792 \pm 0,014$	$0,928 \pm 0,009$	$1,000 \pm 0,000$
Hepatitis (die)	$0,764 \pm 0,034$	$0,935 \pm 0,009$	$1,000 \pm 0,000$
Housing [20-23]	$0,805 \pm 0,007$	$0,890 \pm 0,007$	$0,988 \pm 0,005$
Ionosphere (bad)	$0,914 \pm 0,012$	$0,970 \pm 0,005$	$0,995 \pm 0,003$
Phoneme (1)	$0,930 \pm 0,002$	$0,960 \pm 0,001$	$0,994 \pm 0,001$
Sat-Image (4)	$0,945 \pm 0,001$	$0,968 \pm 0,000$	$1,000 \pm 0,000$
Vehicle (1)	$0,905 \pm 0,004$	$0,938 \pm 0,001$	$0,973 \pm 0,001$
WDBC (M)	$0,994 \pm 0,001$	$0,998 \pm 0,000$	$1,000 \pm 0,000$
Wine (3)	$0,999 \pm 0,001$	$1,000 \pm 0,000$	$1,000 \pm 0,000$
WPBC (R)	$0,734 \pm 0,022$	$0,895 \pm 0,012$	$0,961 \pm 0,006$
Yeast (5)	$0,776 \pm 0,039$	$0,987 \pm 0,001$	$0,998 \pm 0,000$