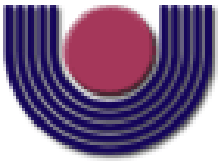

UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Um Estudo sobre o Uso de Máquinas de Vetores de Suporte em Problemas de Classificação

Adriano Douglas Girardello

CASCAVEL

2010

Adriano Douglas Girardello

Um Estudo sobre o Uso de Máquinas de Vetores de Suporte em Problemas de Classificação

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Clodis Boscarioli

CASCAVEL

2010

Adriano Douglas Girardello

Um Estudo sobre o Uso de Máquinas de Vetores de Suporte em Problemas de Classificação

Monografia apresentada como requisito parcial para obtenção do Título de *Bacharel em Ciência da Computação*, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Clodis Boscarioli (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Reginaldo Aparecido Zara
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Rosangela Villwock
Colegiado de Economia Doméstica, Centro de Ciências Sociais Aplicadas, UNIOESTE – Campus Francisco Beltrão

Cascavel, 06 de dezembro de 2010.

DEDICATÓRIA

Dedico este trabalho a todos que me acompanharam estes anos na graduação, em especial minha família e minha namorada.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem ele nada seria possível.

Aos meus pais, pela força, incentivo, patrocínio e acima de tudo, valores que vou carregar por toda a vida.

Ao meu irmão, pela companhia, paciência e pela oportunidade de transmitir meu conhecimento e aprender com ele.

A minha mais que especial namorada Patrícia, pelo seu amor, carinho, muita paciência, compreensão e companheirismo durante os últimos quase 3 anos que passou ao meu lado.

Aos professores do curso de Ciência da Computação da Unioeste, que me ajudaram a trilhar o caminho para conhecimento. Em especial ao orientador Clodis pela oportunidade de realizar esta monografia e sua fundamental ajuda na realização da mesma. Ao professor Clézio do Curso de Matemática, pela ajuda com as fórmulas do método estudado.

A todos os colegas de curso, que ao longo desses 5 anos, ajudaram de alguma forma a passar por todas as dificuldades e também a curtir momentos bons.

Em especial, agradeço ao Evaristo que considero um irmão mais velho, por tudo que fez por mim. Agradeço a Ana e o Tiago, que foram meu grupo de trabalho durante o ano mais difícil do Curso. Agradeço ao Jhonata, Marcelo e Alexandre (Pardal), que além de ter participado de trabalhos em grupo, foram com quem passei mais tempo na faculdade, especialmente os dias que antecederam o fechamento deste trabalho. Foram dias de muito estudo, mas também de muitas risadas.

Agradeço também aos colegas William, Allan, Hudson, Lucas, Marcos e Leandro pela parceria de sempre e pelas festas. Sem esquecer o futebol com toda a galera. Obrigado a todos.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AM	Aprendizado de Máquina
DM	*Data Mining*
KDD	*Knowledge Discovery in Databases*
SVM	*Support Vector Machine*
ERM	*Empirical Risk Minimization*
SRM	*Structural Risk Minimization*
NFR	*Non Functional Requirement*
YADMT	*Yet Another Data Mining Tool*
IHC	Interação Humano-Computador
SGBD	Sistema Gerenciador de Banco de Dados
SIG	*Softgoal Interdependency Graph*
JDBC	*Java DataBase Connectivity*
CT-RC	Classificação de Textos por *Ranking* Categoria
SMO	*Sequential Minimal Optimization*
RBF	*Radial Basis Function*
OAO	*One Against One*
OAA	*One Against All*
RAM	*Random Access Memory*
RBF	*Radial Basis Function*
SPC	Serviço de Proteção ao Crédito
KKT	*Karush Kuhn Tucker*
Ms	Milissegundo
UCI	*University of California Irvine*

Lista de Símbolos

b	*Bias*
P	Distância do hiperplano ótimo ao dado de entrada mais próximo
H_1	Hiperplano de suporte 1 para delimitar uma classe
H_2	Hiperplano de suporte 2 para delimitar uma classe
D	Distância de separação de classes, definida pelos hiperplanos H_1 e H_2
X	Conjunto de itens base para Classificação
Y	Conjunto de itens alvo
X	Vetor de entrada
W	Vetor de peso ajustável
ω	Função de mapeamento do espaço de entrada para um espaço de característica
p	Parâmetro do Kernel Polinomial
σ	Sigma quadrado, parâmetro do Kernel Gaussiano (RBF)
β_1	Parâmetro 1 do Kernel Perceptron Multi Camadas
β_2	Parâmetro 2 do Kernel Perceptron Multi Camadas
N	Número de tuplas de um conjunto de entrada
a1	Multiplicador de Lagrange novo para a primeira tupla analisada pelo SMO
a2	Multiplicador de Lagrange novo para a segunda tupla analisada pelo SMO
alph1	Multiplicador de Lagrange velho para a primeira tupla analisada pelo SMO
alph2	Multiplicador de Lagrange velho para a segunda tupla analisada pelo SMO
E_1	Distância do ponto y_1 até o hiperplano atualmente calculado pelo SMO
y_1	Classe da primeira tupla analisada pelo SMO
y_2	Classe da segunda tupla analisada pelo SMO

Sumário

Resumo

Visto a grande quantidade de dados gerada nos dias atuais, métodos automatizados para análise se fazem necessários. Este trabalho está focado na etapa de *Data Mining* (Mineração de Dados), especificamente na tarefa de Classificação. Para realização dessa tarefa de classificação foi escolhida a técnica SVM – *Support Vector Machine*, uma técnica baseada na teoria do aprendizado estatístico. O objetivo geral deste trabalho foi realizar um estudo teórico-prático sobre SVM, com intuito de avaliar/validar essa técnica a partir de bases de dados de acesso público.

Palavras-chave: Máquinas de Vetores de Suporte, Mineração de Dados, Classificação.

Capítulo 1

Introdução

Uma grande quantidade de dados é gerada, coletada e armazenada todos os dias no mundo. Os fins são variados, como científicos, sociais, econômicos, comerciais, segurança, etc, o que vem aumentando em muito o volume das bases de dados, o que impõe sérias dificuldades para analisá-las manualmente. Com esta realidade, métodos automatizados para a descoberta de informações se fazem necessários.

Neste contexto, surge uma nova linha de pesquisa denominada KDD – *Knowledge Discovery in Databases* (Descoberta de Conhecimento em Bases de Dados) definida por FAYYAD *et al.* (1996) como um processo não trivial, interativo e iterativo, que visa melhorar os processos de análise de grandes massas de dados para a geração de conhecimento, e é composto das etapas de Pré-Processamento, Mineração de Dados (DM, do inglês *Data Mining*) e Pós-Processamento.

O Pré-Processamento consiste no tratamento e na preparação dos dados a serem usados pelos algoritmos na fase de mineração, pois a qualidade dos mesmos é um fator determinante para a eficiência dos algoritmos de mineração. Nesta etapa, são identificados e retirados valores inválidos e inconsistentes, recuperados dados incompletos e avaliados possíveis dados discrepantes ao conjunto chamados de *outliers* (dados com valores extremos ou atípicos). No caso de falta de dados (*missing values*) em uma tupla[1], pode-se tratar a inconsistência utilizando diferentes técnicas, como a imputação ou substituir o valor faltante pela média aritmética da variável (NEVEZ, 2005). Nota-se que a presença de um especialista se faz necessário nesta etapa, pois é ele quem determinará o que fazer com um determinado dado inconsistente.

Após obter uma base de dados limpa, padronizada e bem estruturada, inicia-se a Mineração de Dados, onde se dá a descoberta do conhecimento, o que consiste na busca por padrões

[1] Seqüência ordenada de n elementos. Sinônimo de registro, entrada, exemplo, objeto ou amostra de dados.

através da aplicação de algoritmos e técnicas computacionais específicas, segundo uma tarefa definida (NEVEZ, 2005).

O Pós-Processamento envolve a interpretação do conhecimento descoberto ou algum processamento do mesmo. O conhecimento extraído pode ser simplificado; avaliado por meio de critérios como precisão e compreensibilidade entre outros; visualizado, ou simplesmente documentado para o usuário final (DOMINGUES; REZENDE, 2005).

Há muita possibilidade de emprego da tecnologia de Mineração de Dados de acordo com os objetivos. Para cada tipo de conhecimento requerido, existe um algoritmo para seu descobrimento. Neste contexto, as tarefas de Classificação, Regras de Associação e Análise de Agrupamento (*Clustering*) são tipicamente as mais utilizadas (SOUSA, 1998).

O princípio da tarefa de Classificação é descobrir algum tipo de relacionamento entre as tuplas de entrada com a determinada classe que elas pertencem, de modo a descobrir um conhecimento através de um modelo (função) que possa ser útil para prever a classe de uma tupla desconhecida (AURÉLIO *et al.*, 1999). Já as Regras de Associação consistem em encontrar relacionamentos entre atributos de uma tupla. Os padrões de informação são do tipo $X \rightarrow Y$, onde X e Y são conjuntos de itens. Assim, são pesquisadas todas as possíveis regras $X \rightarrow Y$, onde as transações da base de dados que contêm X tendem a conter também Y. Já a Análise de agrupamentos busca similaridade entre os dados para poder agrupá-los em classes. A principal diferença em relação à Classificação, é que não se tem conhecimento prévio sobre as classes.

Dentre essas tarefas, a Classificação é a mais utilizada. Para a realização da Classificação, diferentes técnicas são empregadas, a exemplo de Árvores de Decisão e Redes Neurais Artificiais. Um método que vem sendo empregado na classificação de dados obtendo altas acuidades é Máquinas de Vetores de Suporte[2] (SVM, do inglês *Support Vector Machines*) (HAN; KAMBER, 2001), que será o objeto principal dessa pesquisa.

A técnica SVM é embasada pela teoria do aprendizado estatístico, desenvolvida por VAPNIK (2000). Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa generalização, definida como sua capacidade de prever corretamente a classe de novos dados do mesmo domínio em que o aprendizado ocorreu. A SVM procura minimizar o risco estrutural, isto é, a probabilidade de classificar de forma

[2] Por conveniência, Máquinas de Vetores de Suporte serão tratadas no feminino, referindo-se a "a técnica SVM".

errada padrões ainda não vistos por uma distribuição de probabilidade dos dados fixa e desconhecida (LORENA; CARVALHO, 2007). Essas características motivaram a escolha da SVM para esse estudo.

2.3 Justificativa

Uma das principais dificuldades em aplicar técnicas de Mineração de Dados em grandes bases de dados é a falta de integração entre as ferramentas e o SGBD (Sistema Gerenciador de Banco de Dados) (SILVA; XEXÉO, 1998). A solução para esse problema são as ferramentas de Mineração de Dados acopladas aos SGBDs, a exemplo de SAS Enterprise Miner (SAS, 2010), IBM Intellingent Miner (IBM, 2010) e Oracle Data Mining (ORACLE, 2010), mas o alto custo dessas ferramentas torna o processo inviável. Além disso, não é possível utilizar essas ferramentas em outros formatos de dados.

Uma alternativa às ferramentas proprietárias são as ferramentas independentes de SGBDs como Weka (WEKA, 2010), Tanagra (TANAGRA, 2010) e Orange (ORANGE, 2010), mas ainda assim, há necessidade da transformação dos dados para o formato de entrada destas ferramentas (OLIVEIRA; GARCIA, 2004). Fica evidenciada a necessidade de ferramentas que possam ser utilizadas em diferentes formatos de dados e em diferentes domínios de aplicação, como ressalta VOLTOLINI (2008).

Muitos dos classificadores existentes são baseados em técnicas de Árvores de Decisão e Redes Neurais Artificiais. Porém, na literatura (Ver (COELHO, 2005), (KINTO, 2005), (LIMA, 2002), (LORENA; CARVALHO, 2007)), averigua-se que os resultados obtidos com a técnica de Máquinas de Vetores de Suporte são comparáveis e muitas vezes superiores a outros algoritmos de aprendizado, o que justifica estudar e implementar esse método, e analisar os resultados obtidos, para não apenas confirmar a assertividade desse algoritmo, mas criar um ambiente de comparação com outras técnicas de classificação.

O módulo de SVM aqui desenvolvido fará parte da ferramenta YADMT (*Yet Another Mining Tool*) em desenvolvimento, compondo juntamente com técnicas de Redes Neurais Artificiais e Algoritmos Estatísticos um módulo de classificação de dados.

1.2 Objetivos

O objetivo geral desse trabalho é o estudo teórico-prático de SVM aplicada a classificação de dados.

Para chegar a esse objetivo, as seguintes metas foram traçadas:

1. Levantamento de trabalhos correlatos que utilizam SVM em tarefas de classificação diversas.
2. Implementação de um módulo de Classificação com SVM.
3. Realização de experimentos com bases de dados para testes e validação da implementação.

1.3 Organização do Documento

O documento está estruturado da seguinte maneira: O Capítulo 2 faz a contextualização sobre classificação, e explica o funcionamento das Máquinas de Vetores de Suporte e métodos de avaliação de classificadores. Ao final do Capítulo são também descritos alguns trabalhos correlatos.

No Capítulo 3 é apresentado o experimento realizado, mostrando como os conceitos foram aplicados e como o trabalho foi desenvolvido. Os resultados obtidos também são demonstrados e discutidos.

Por fim, no Capítulo 4 é feita uma conclusão de todo o trabalho realizado e são feitas sugestões para trabalhos futuros.

Capítulo 2

Classificação e Máquinas de Vetor de Suporte

Classificação é uma tarefa que o ser humano executa frequentemente sem maiores dificuldades. De forma ágil, ele pode receber e interpretar dados do mundo exterior por meio dos sentidos e pode reconhecer em algum determinado contexto, a que classe pertence. Através de um processo de aprendizagem o conhecimento é adquirido para executar a classificação, sem muito esforço. Porém, quando grandes quantidades de dados são investigadas, o processo se torna inviável, haja vista o tempo gasto no processo, além da dificuldade de diferenciar os dados. Sendo assim a automação do processo de classificação se faz necessário. Um processo análogo ao do ser humano ocorre computacionalmente, onde se busca gerar conhecimento em bases de dados através de algoritmos.

Este Capítulo contextualiza as Máquinas de Vetores de Suporte à classificação de dados. Na Seção 2.1 são apresentados conceitos sobre classificação. A seguir na Seção 2.2 é apresentada a técnica SVM em suas duas formas, lineares e não lineares, e como fazer a classificação em bases Multiclasses. A Seção 2.3 trata de métodos de avaliação de Classificadores, e por fim, na Seção 2.4, são apresentados alguns trabalhos correlatos.

2.1 Classificação

Classificação é o processo que encontra propriedades comuns entre um conjunto de instâncias num banco de dados e classifica-os em diferentes classes (pré-definidas), de acordo com um modelo de classificação (VASCONCELOS, 2002). O modelo é derivado a partir da análise de um conjunto de dados de treinamento, ou seja, dados cujas classes são conhecidas (HAN; KAMBER, 2001).

HAN e KAMBER (2001) propõem a classificação em duas etapas. Na primeira etapa um algoritmo se baseia em um conjunto de treinamento composto por tuplas e suas respectivas

classes. Na segunda etapa, o modelo gerado é utilizado para classificar novos dados, a fim de testá-lo. A precisão de um classificador em um dado conjunto de testes é baseada na quantidade de dados corretamente classificados. Portanto, a classificação necessita ter um conhecimento prévio dos dados para extrair o conhecimento, denominado aprendizado supervisionado, um conceito de Aprendizagem de Máquina (AM).

No Aprendizado Supervisionado os conjuntos de exemplos são pares do tipo entrada e saída. O algoritmo de AM extrai a representação do conhecimento a partir desses exemplos. O objetivo é que o modelo gerado seja capaz de classificar corretamente novas entradas desconhecidas (SEMOLINI, 2002).

Já no Aprendizado Não-Supervisionado os exemplos de entrada não são rotulados, ou seja, não tem a saída correspondente. O algoritmo de AM aprende a representar (ou agrupar) as entradas submetidas segundo uma medida de qualidade, classificando as novas entradas em alguma categoria. Caso a entrada não se enquadre em nenhuma classe existente, uma nova classe é gerada (LORENA; CARVALHO, 2007).

No presente trabalho é empregado o aprendizado supervisionado, onde as entradas são do tipo (X_i, Y_i), onde X_i representa a "i-ésima" tupla de entrada, e Y_i sua respectiva classe. A partir disso, deve-se produzir um classificador capaz de predizer a classe de novas entradas. Este processo de indução do classificador a partir de uma amostra de dados é denominado treinamento.

A Figura 2.1 representa o processo de geração de um classificador representado de forma simplificada, onde se tem um conjunto de entrada com n dados. Cada tupla x_i possui m atributos. As variáveis y_i representam as classes. A partir dos exemplos e suas respectivas classes é aplicada uma técnica de AM extraindo um classificador.

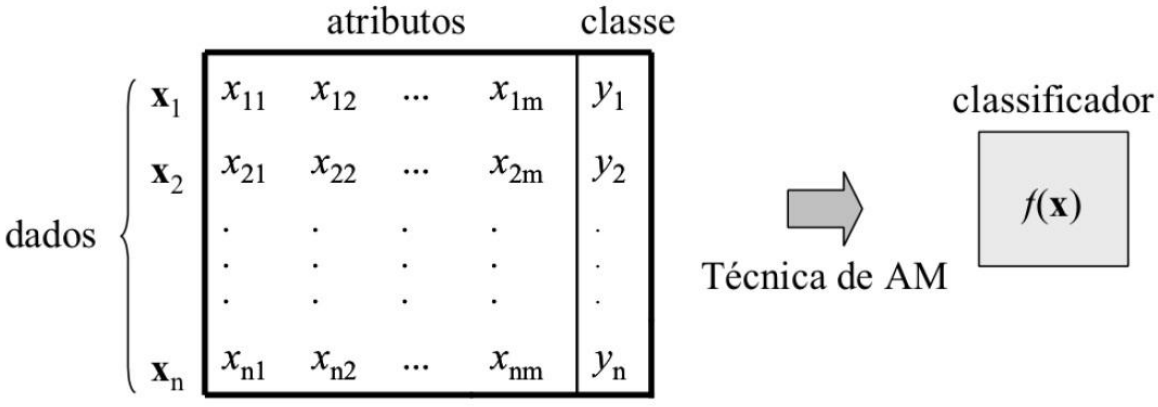

Figura 2.1: Indução de um classificador em aprendizado supervisionado (LORENA; CARVALHO, 2007)

Em AM um conceito muito utilizado é a capacidade de generalização de um classificador, definida como sua capacidade de prever corretamente novos dados. No caso em que o modelo se especializa no conjunto de exemplo utilizado no seu treinamento, apresentando uma baixa taxa de acerto quando confrontado com novos dados, tem-se a ocorrência de um super-ajustamento (*overfitting*). Já quando o modelo apresenta uma baixa taxa de acerto no conjunto de treinamento, configura-se uma condição de sub-ajustamento (*underfitting*). Isso pode ocorrer quando os exemplos de treinamento são pouco representativos ou quando o modelo gerado é muito simples (LORENA; CARVALHO, 2007).

Tendo como base a Figura 2.2, analisa-se a capacidade de generalização de um classificador sobre duas classes diferentes: "círculo" e "triângulo". O modelo de classificação é representado por meio de uma linha, separando as classes. A Figura 2.2 (a) representa a situação onde o modelo classifica corretamente todos os dados de exemplo, inclusive os ruídos. Este caso representa a situação de super-ajustamento, pois devido ao modelo ser bastante específico para o conjunto de treinamento, acaba por ser suscetível a erros quando confrontado com dados desconhecidos. Na Figura 2.2 (c) o classificador desconsidera pontos pertencentes a classes opostas que estejam muito próximas entre si. Esta hipótese comete muitos erros até mesmo em casos simples. Neste caso há um sub-ajustamento. Um meio termo entre os dois casos citados anteriormente é a representação da Figura 2.2 (b), onde o classificador tem complexidade intermediária e classifica corretamente grande parte dos dados sem se prender a exemplos de entrada.

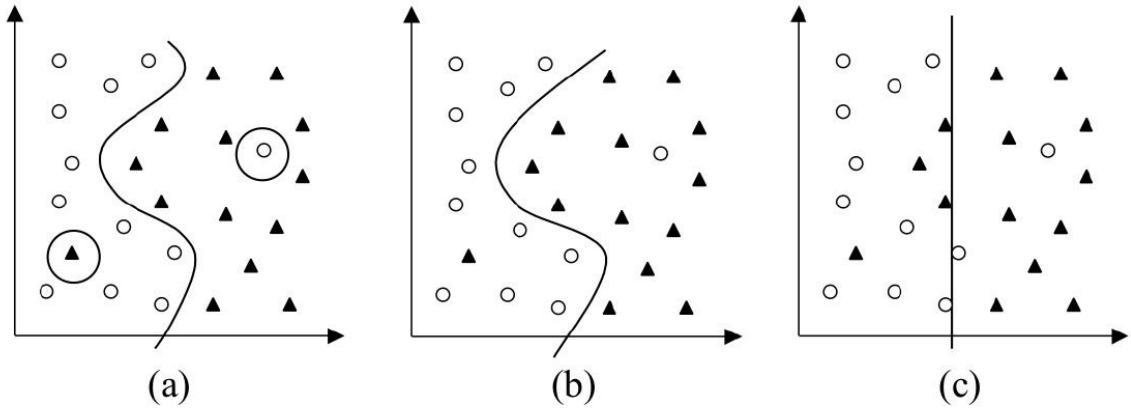

Figura 2.2: Capacidade de Generalização de um Classificador (LORENA; CARVALHO, 2007)

2.2 Máquinas de Vetores de Suporte

Máquinas de Vetor de Suporte (SVM, do inglês *Support Vector Machines*), proposta por Vapnik em 1992 (VAPNIK, 2000), embora, segundo HAN e KAMBER (2001) as bases para SVM já fossem consolidadas na década de 60, é uma técnica de aprendizado de máquina (AM) fundamentada nos princípios indutivos da Minimização do Risco Estrutural (SRM, do inglês *Structural Risk Minimization*). Estes princípios são provenientes da Teoria do Aprendizado Estatístico. A idéia básica de uma SVM é construir um hiperplano ótimo como superfície de decisão, de tal forma que a margem de separação entre as classes seja máxima.

A SVM se baseia na minimização do risco estrutural ao contrário de outras técnicas que tem como base a minimização do risco empírico (ERM, do inglês *Empirical Risk Minimization*) (VAPNIK, 2006). Assim, o SRM minimiza o erro de generalização, diferentemente do ERM que visa minimizar o erro associado às amostras de treinamento, sendo um diferencial em relação a outras técnicas de classificação. SVM pode ser utilizada tanto para classificação quanto para regressão (HAN; KAMBER, 2001).

2.2.1 Máquinas de Vetores de Suporte Lineares

Tratando-se de classificação binária, o problema é encontrar uma função paramétrica, linear ou não, que defina um hiperplano de separação das classes (SILVA; SCARPEL, 2007). Seja T um conjunto de entrada do tipo (X_i, Y_i) onde X_i representa a "i-ésima" tupla de entrada e Y_i sua respectiva classe sendo $Y = \{-1, +1\}$, T é linearmente separável se é possível separar os dados das classes -1 e +1 por um hiperplano (LORENA; CARVALHO, 2007). A superfície de decisão na forma de um hiperplano que realiza essa separação é obtida pela Equação 2.1.

$$\mathbf{w}^T.\mathbf{x} + b = 0 \tag{2.1}$$

Onde $\mathbf{x}$ é um vetor de entrada, $\mathbf{w}$ é um vetor de peso ajustável e b é um *bias*. Assim, a Equação 2.1 é reescrita com base nas classes $Y = +1$ e $Y = -1$, como mostra a Equação 2.2 e a Equação 2.3.

$$\mathbf{w}^T.\mathbf{x} + b \geq 0 \quad \text{para Y} = +1 \tag{2.2}$$

$$\mathbf{w}^T.\mathbf{x} + b < 0 \quad \text{para Y} = -1 \tag{2.3}$$

Para um dado vetor **w** e um bias *b*, a separação entre o hiperplano definido na Equação 2.1 e o ponto de dado mais próximo é denominada a *margem de separação*, representada por ρ (HAYKIN, 2001). Sempre que for possível obter um $\rho > 0$, existirão infinitos hiperplanos entre as duas classes, como ilustrado na Figura 2.3.

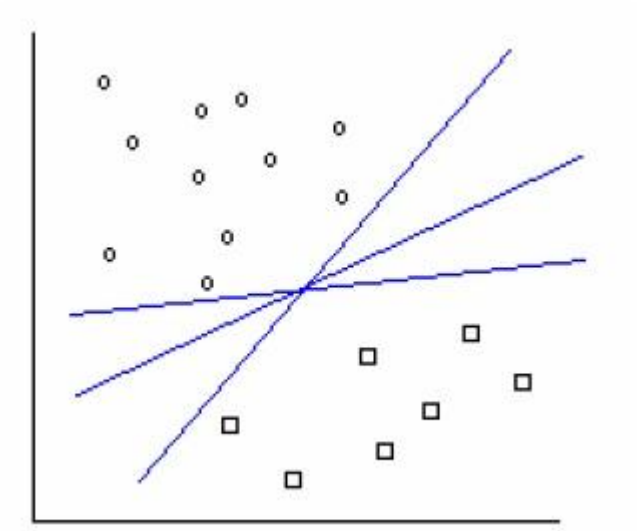

Figura 2.3: Possíveis Hiperplanos de separação entre duas classes (CRUZ, 2007)

O objetivo da SVM é encontrar o hiperplano particular para o qual a margem de separação ρ é máxima, como é o caso da Figura 2.4.

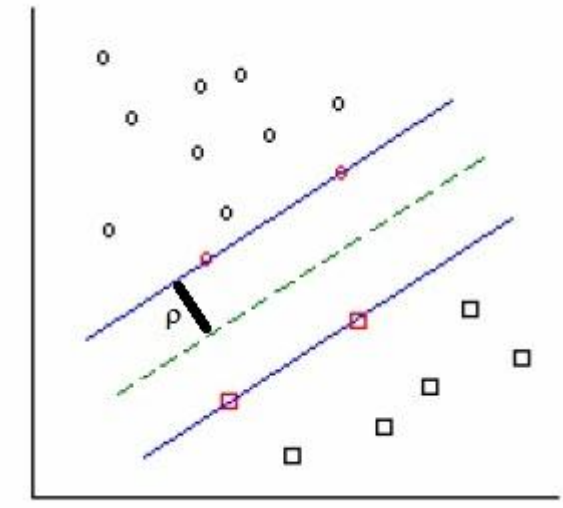

Figura 2.4: Hiperplano ótimo para separação de duas classes, adaptado de (CRUZ, 2007)

O hiperplano ótimo, em verde (linha tracejada), é encontrado com auxílio de dois vetores de suporte, em azul (linhas contínuas), por isso o nome “Máquinas de Vetores de Suporte”. Estes vetores desempenham um papel importante na operação desta máquina de

aprendizagem. Em termos conceituais, os vetores de suporte são aqueles pontos de dados que se encontram mais próximos da superfície de decisão e são, portanto, os mais difíceis de classificar (HAYKIN, 2001).

Considerando que $\mathbf{w_o}$ e b_o representem os valores ótimos do vetor peso e do *bias*, respectivamente, o hiperplano ótimo, representando uma superfície de decisão linear multidimensional no espaço de entrada é definido pela Equação 2.4 (HAYKIN, 2001).

$$\mathbf{w_o}^T.\mathbf{x} + b_o = 0 \tag{2.4}$$

Seja x_1 um ponto no hiperplano H_1: $\mathbf{w}^T.\mathbf{x} + b = +1$ e x_2 um ponto no hiperplano H_2: $\mathbf{w}^T.\mathbf{x} + b = -1$, conforme ilustrado na Figura 2.5.

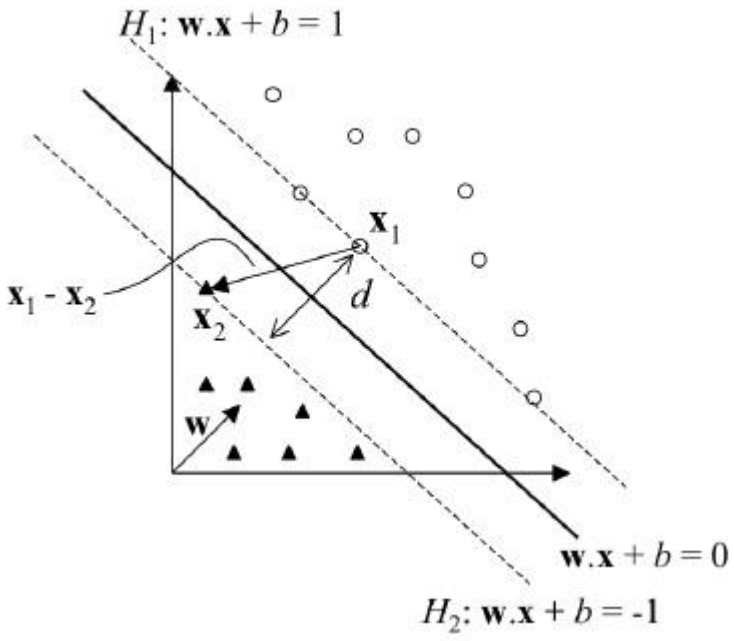

Figura 2.5: Cálculo da distância d entre os hiperplanos H1 e H2 (LORENA; CARVALHO, 2007)

Projetando $x_1 - x_2$ na direção de **w**, perpendicular ao hiperplano separador $\mathbf{w}^T.\mathbf{x} + b = 0$, é possível obter a distância entre os Hiperplanos H_1 e H_2 (LORENA; CARVALHO, 2007), projeção apresentada na Equação 2.5.

$$d = (x_1 - x_2)\left(\frac{w}{||w||}.\frac{(x_1 - x_2)}{||x_1 - x_2||}\right) \tag{2.5}$$

Tem-se que $\mathbf{w}^T.\mathbf{x_1} + b = +1$ e $\mathbf{w}^T.\mathbf{x_2} + b = -1$. A diferença entre essas equações fornece $\mathbf{w}^T.(\mathbf{x_1} - \mathbf{x_2}) = 2$. Substituindo esse resultado na Equação 2.5, obtêm-se a Equação 2.6:

$$d = \frac{2(x_1 - x_2)}{||w||\ ||x_1 - x_2||} \tag{2.6}$$

Como se deseja obter o comprimento do vetor projetado toma-se a norma da Equação 2.6, obtendo a Equação 2.7.

$$d = \frac{2}{||w||} \tag{2.7}$$

Essa é a distância *d* ilustrada na Figura 2.5, entre os hiperplanos H_1 *e* H_2, paralelos ao hiperplano separador. Como $\boldsymbol{w}$ e *b* foram escalados de forma a não haver exemplos entre H_1 e H_2, $1/||\boldsymbol{w}||$ é a distância mínima entre o hiperplano separador e os dados de treinamento.

Nota-se que a maximização da margem de separação dos dados em relação ao hiperplano ótimo pode ser obtida pela minimização de $||\boldsymbol{w}||$. Dessa forma, segundo HAYKIN (2001), o problema pode ser descrito como:

Dada uma amostra de treinamento $\{(x_i, y_i)\}$ *onde* x_i *representa a "i-ésima" tupla de entrada e* y_i *sua respectiva classe, encontre os valores do vetor peso* $\boldsymbol{w}$ *e bias b de modo que satisfaçam as restrições:*

$$Y_i(\mathbf{w}^T.\mathbf{x_i} + b) \geq 1 \text{ para } i = 1,2,...,n \tag{2.8}$$

e o vetor peso $\boldsymbol{w}$ *minimize a função de custo:*

$$f(w) = \frac{1}{2} w^T w \tag{2.9}$$

O fator de escala ½ é incluído aqui por conveniência de apresentação. O problema acima é chamado de *formulação primal* (HAYKIN, 2001).

Devido à natureza das restrições do problema primal, pode-se ter dificuldades na obtenção da solução do problema. Segundo ALES *et al.* (2009), problemas desse tipo podem ser solucionados com a introdução de uma função Lagrangiana que engloba as restrições a função

objetivo associadas a parâmetros denominados Multiplicadores de Lagrange (LEITHOLD, 1994), como mostra a Equação 2.10.

$$L(w, b, \alpha) = \frac{1}{2} \ ||w||^2 \ - \sum_{i=1}^{n} \alpha i \ (yi(w^T.x_i + b) - \ 1) \tag{2.10}$$

onde as variáveis auxiliares não negativas α_i são chamadas de Multiplicadores de Lagrange. A função Lagrangiana deve ser minimizada, isso significa que α_1 deve ser maximizado e $\boldsymbol{w}$ e b minimizado. Tem-se então um ponto de sela[3], no qual:

$$\frac{\omega L}{\omega w} = 0 \ e \ \frac{\omega L}{\omega b} = 0 \tag{2.11}$$

A resolução dessas equações leva aos resultados apresentados a seguir nas Equações 2.12 e 2.13:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.12}$$

$$w = \sum_{i=1}^{n} \alpha_i y_i \mathrm{x_i} \tag{2.13}$$

Substituindo as Equações 2.12 e 2.13 na Equação 2.10, obtêm-se o seguinte problema de otimização:

$$maximizar \ \alpha \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i \mathrm{y_j} \ (x_i.x_j) \tag{2.14}$$

com as restrições: $\alpha_i \geq 0, \forall i = 1, \ldots, n$

[3] Ponto de sela é um ponto no domínio de função de duas variáveis, que é um ponto estacionário, mas não um extremo local.

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.15}$$

Essa formulação é denominada forma dual, enquanto o problema original é referenciado como forma primal. Também é importante notar que no ponto de sela, para cada Multiplicador de Lagrange α_1, o produto daquele multiplicador pela restrição correspondente desaparece, como mostrado pela Equação 2.16:

$$\alpha_1 [y_i (w^T x_i + b) - 1] = 0 \; para \; i = 1, 2, \ldots N \tag{2.16}$$

Dessa forma, apenas aqueles multiplicadores que satisfazerem exatamente a Equação 2.16 podem assumir valores não-nulos. Esta propriedade resulta das condições de *Karush Kuhn Tucker* (KKT), que são condições necessárias para que uma solução em problemas de programação não-linear seja ótima, dado que ela satisfaça determinadas condições de regularidade (GERVERT, 2009).

2.2.2 Máquinas de Vetores de Suporte com Margens Suaves

Na Classificação não linearmente separável, tem-se um conjunto de tuplas onde não é possível separar as classes por meio de um hiperplano sem se deparar com erros de classificação. Apesar disso, deseja-se encontrar um hiperplano ótimo que minimize a probabilidade de erro de classificação (HAYKIN, 2001). Para isso, permite-se que alguns dados possam violar a restrição estabelecida na Equação 2.8. Isso é feito com a introdução de variáveis de folga ε_i para todo i = 1, 2,...,n. Estas variáveis relaxam as restrições impostas ao problema de otimização primal (Equação 2.8), como mostra a Equação 2.17.

$$y_i(\mathbf{w}^{\mathrm{T}} \mathrm{x_i} + \mathrm{b}) \geq 1 - \varepsilon_i, \; \varepsilon_i \geq 0, \quad \text{para i} = 1,2,\ldots,n \tag{2.17}$$

A aplicação desse procedimento suaviza as margens do classificador linear, permitindo que alguns dados estejam entre os vetores de suporte e também a ocorrência de alguns erros de classificação (LORENA; CARVALHO, 2007).

As violações podem ocorrer de três maneiras, como pode ser visto na Figura 2.6. Os casos representam dados não linearmente separáveis. Na Figura 2.6 (a) o ponto x_i encontra-se dentro

da região de separação e no lado correto. Na Figura 2.6 (b) o ponto x_i encontra-se dentro da região de separação, porém no lado incorreto da superfície de decisão. Já na Figura 2.6 (c) o ponto x_i encontra-se fora da região de separação e no lado incorreto da superfície de decisão (SEMOLINI, 2002).

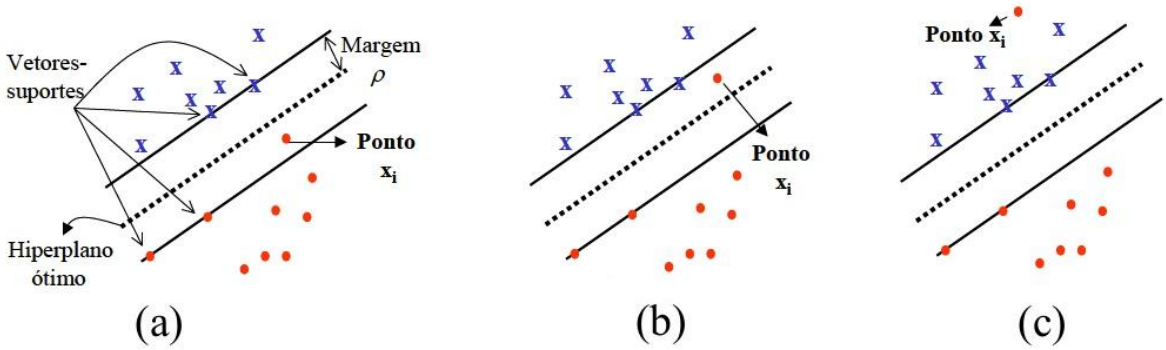

Figura 2.6: Hiperplanos ótimos para padrões não linearmente separáveis (SEMOLINI, 2002)

2.2.3 Maquinas de Vetores de Suporte Não Lineares

Muitas aplicações utilizam tuplas não linearmente separáveis. A SVM tem a capacidade de aprender em espaços não lineares pelo mapeamento dos dados para um espaço de características onde eles podem ser efetivamente separados, como mostra a Figura 2.7.

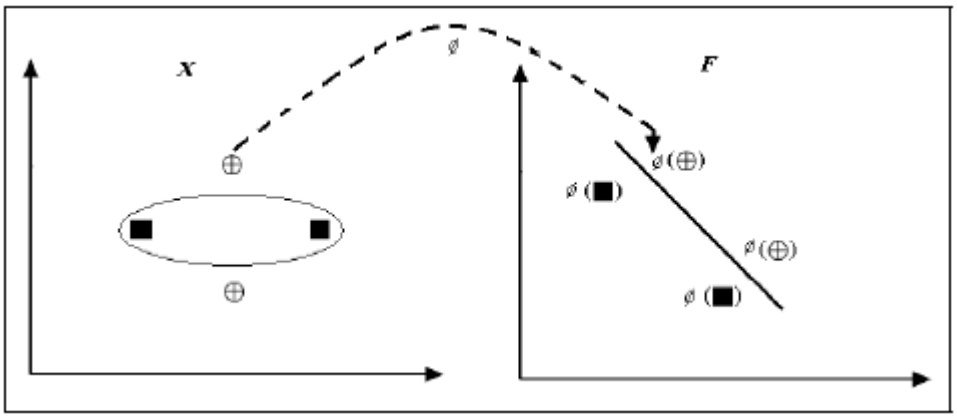

Figura 2.7: Representação de mudança de um espaço de entrada bidimensional, para um espaço de característica (SEMOLINI, 2002)

Para realizar esta transposição de espaços é utilizada uma função chamada *Kernel*. Os dados originais estão no espaço de entrada. Quando se faz uma função $\emptyset$: $R^n \rightarrow R^p$, com $p > n$, os dados são mapeados para um espaço de característica. Esta mudança é representada na

Equação 2.18:

$$x = (x_1, \dots, x_n) \rightarrow \emptyset(x) = (\emptyset(x_1), \dots, \emptyset(x_n)) \quad (2.18)$$

O *Kernel* é uma função $K(x^i, x^j) = K(x^j, x^i) = \emptyset(x^i).\ \emptyset(x^j)$ para x^i e x^j pertencente ao espaço de entrada, onde $\emptyset$ é um mapeamento do espaço de entrada para um espaço de característica de maior dimensão que o original. As funções de *Kernel* mais conhecidas e utilizadas são:

- Polinomial → $K(x,y) = ((x_i, y_j) + 1)^p$
- Linear → $K(x,y) = x.y$
- Perceptron Multi Camadas → $\tanh(\beta_0 x.y + \beta_1)$
- Gaussiana (*RBF*) → $e^{\frac{||x_i - x_j||^2}{2\sigma^2}}$

A função Gaussiana também é conhecida como Função de Base Radial (RBF, do inglês *Radial Basis Function*).

Os parâmetros p, σ, β_0 e β_1 são especificados *a priori* pelo usuário.

Esta transposição baseia-se no Teorema de Cover sobre a separabilidade de padrões, onde o mesmo afirma que um problema complexo de classificação de padrões dispostos não linearmente em um espaço de alta dimensionalidade tem maior probabilidade de ser linearmente separável do que em um espaço de baixa dimensionalidade (HAYKIN, 2001).

2.2.4 Estratégias de Separação Multiclasses

PLATT *et al.* (2000) dizem que o problema de classificação Multiclasse, especialmente para a SVM não apresenta uma solução fácil e a construção de SVMs Multiclasses é ainda um problema de pesquisa não resolvido.

Para problemas que possuem mais de duas classes, o conjunto de dados de treinamento deve ser combinado para formar problemas de duas classes (BISOGNIN, 2007). A seguir são descritos os dois principais métodos: Um Contra Um (em inglês, *One Against One*, OAO) e Um Contra Todos (em inglês, *One Against All*, OAA).

2.2.4.1 Um Contra Um

É um método simples e eficiente para a resolução de problemas Multiclasses. Supondo um problema com *n* classes, para cada par dessas *n* classes é construído um classificador binário. Cada classificador é construído utilizando elementos das duas classes envolvidas, obtendo um

total de *n* (*n* - 1) / 2 classificadores, como ilustra a Figura 2.8. Portanto os conjuntos de treinamento devem ser rotulados novamente para cada par de entradas.

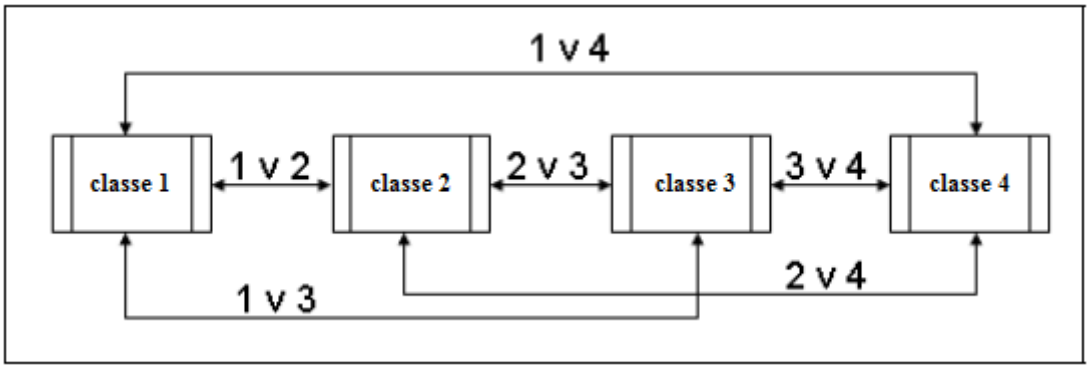

Figura 2.8: Representação do método Um Contra Um (BISOGNIN, 2007)

2.2.4.2 Um Contra Todos

Supondo que um problema tem *n* classes, este método consiste em particionar estas *n* classes em dois grupos. Um grupo é formado por uma classe e o outro é formado pelas classes restantes. Um classificador binário é treinado para esses dois grupos e este procedimento é repetido para cada uma das *n* classes.

Uma vantagem desse método é o número reduzido de classificadores, comparado ao método Um Contra Um, o que torna a classificação mais rápida em casos de poucas classes. Uma desvantagem é que cada classificador utiliza todas as classes, sendo assim o desempenho depende do número de classes.

2.3 Métodos de Avaliação de Classificação

Segundo HAN e KAMBER (2001), classificadores podem ser validados/comparados através dos seguintes critérios:

- Acuidade: refere-se à habilidade de um classificador predizer corretamente o rótulo de classe de um novo dado.
- Velocidade: refere-se ao custo computacional envolvido na geração e uso de um dado classificador.
- Robustez: habilidade de um classificador de fazer predições corretas de dados, mesmo com ruídos ou valores faltando.

- Escalabilidade: refere-se à habilidade de construir um classificador eficiente dado uma grande quantidade de dados.
- Interpretabilidade: refere-se ao nível de entendimento e compreensão que é fornecido por um classificador. É muito subjetivo e, portanto, mais difícil de ser analisado.

Dentre estes critérios, a acuidade é um dos mais utilizados, pois representa em termos matemáticos a porcentagem de tuplas do conjunto de testes que são classificadas corretamente.

Uma ferramenta útil que auxilia na avaliação da acuidade é a matriz de confusão. Uma matriz de confusão de *m* classes é uma tabela, de tamanho *m* por *m*, onde uma entrada M_{ij} indica o número de tuplas pertencentes à classe *i* que foram classificadas como classe *j*. Para um classificador ter boa acuidade, a maioria das tuplas deve ser representada ao longo da diagonal principal da matriz, com o resto das entradas sendo igual a zero.

Tabela 2.1: Exemplo de matriz de confusão

	Classe Correta			
Classe predita		**A**	**B**	**C**
	A	3	1	0
	B	1	2	0
	C	0	0	4

A Tabela 2.1 traz um exemplo de matriz de confusão para um conjunto de 11 tuplas, onde os resultados mostrados podem ser interpretados da seguinte forma:

- 3 tuplas da classe A foram corretamente classificadas;
- 1 tupla da classe A foi erroneamente classificada como classe B;
- 1 tupla da classe B foi erroneamente classificada como classe A;
- 2 tuplas da classe B foram corretamente classificadas;
- 4 tuplas da classe C foram corretamente classificadas.

Um tipo especial de matriz de confusão ocorre quando há apenas duas classes à classificar, classificação binária. Neste tipo de matriz pode-se falar em termos de tupla positiva *versus* tupla negativa (Tabela 2.2) e nas seguintes definições:

- Verdadeiro positivo: refere-se à tupla que foi corretamente classificada pelo classificador;
- Verdadeiro negativo: refere-se à tupla negativa que foi corretamente classificada pelo classificador;

- Falso positivo: refere-se à tupla negativa que foi incorretamente classificada como positiva;
- Falso negativo: refere-se à tupla positiva que foi incorretamente classificada como negativa.

Tabela 2.2: Matriz de confusão para tuplas positivas e negativas

		Classe classificada	
		C1	**C2**
Classe Atual	**C1**	verdadeiro positivo	falso negativo
	C2	falso positivo	verdadeiro negativo

Por meio dos valores contidos nesta matriz de confusão é possível definir métricas que podem ser utilizadas como avaliadores de tal classificação, a exemplo de sensibilidade e especificidade. Sensibilidade é referida como taxa de reconhecimento de verdadeiros positivos, que corresponde a proporção de tuplas positivas que foram corretamente classificadas. Já especificidade é a taxa de verdadeiros negativos, que corresponde à proporção de tuplas negativas que foram corretamente identificadas. Estas medidas estão definidas nas Equações 2.19 e 2.20.

$$\text{sensibilidade} = \frac{v\_pos}{pos} \tag{2.19}$$

$$\text{especificidade} = \frac{v\_neg}{neg} \tag{2.20}$$

onde:

v_pos: número de tuplas classificadas corretamente como positivo (verdadeiro positivo);

pos: número de tuplas classificadas como positivo (verdadeiro positivo + falso positivo);

v_neg: número de tuplas classificadas corretamente como negativo (verdadeiro negativo);

neg: número de tuplas classificadas como negativo (verdadeiro negativo + falso negativo).

Para garantir que o valor da acuidade de um classificador seja uma estimativa confiável utilizam-se algumas técnicas de avaliação, dentre as quais citam-se: *Holdout*, *Random Subsampling* e *k-fold-cross-validation* (HAN e KAMBER, 2001).

No presente trabalho, fez-se o uso do método *Holdout* estratificado, no qual os dados fornecidos para o algoritmo de classificação são aleatoriamente particionados em dois conjuntos independentes: conjunto de treinamento e conjunto de testes. Tipicamente, dois

terços dos dados são alocados para o conjunto de treinamento e o restante é alocado para o conjunto de testes. O conjunto de treinamento é usado para gerar o modelo, o qual tem a acuidade estimada com o conjunto de teste (Figura 2.9).

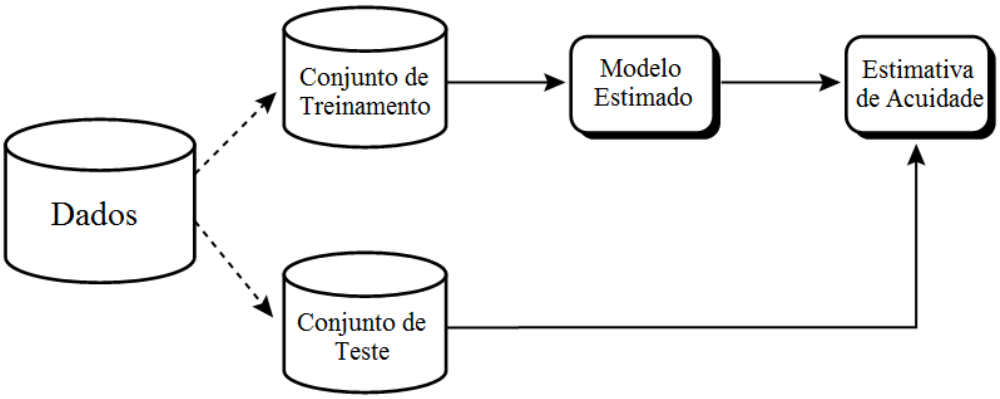

Figura 2.9: Estimativa de acuidade pelo método *Holdout*. Adaptado de (HAN;KAMBER, 2001)

2.4 Trabalhos Correlatos

A seguir são citados alguns trabalhos envolvendo SVM empregada na tarefa de Classificação de dados, é descrito a idéia proposta, passos realizados, problemas encontrados e resultados. Estes trabalhos mostram como a técnica SVM está sendo empregada e que avaliação obteve nestas circunstâncias.

2.4.1 Máquinas de Vetores de Suporte Aplicadas a Classificação de Textos Reduzidos

KINTO (2005) aplica SVM na Classificação de textos reduzidos (textos que apresentam poucas palavras). A Classificação de textos reduzidos é o processo de identificar um documento por um nome (*label*) pré-definido e é justificada pela necessidade de organizar e classificar grandes quantidades de informações acessadas na atualidade, proporcionados pelo avanço da tecnologia. Essa Classificação de textos reduzidos teve aplicação no domínio de *e-commerce* brasileiro.

O sistema de Classificação automática de texto empregado consistiu nas seguintes etapas:

1) Coleta de Informação;
2) Seleção dos termos: nesta fase os termos mais relevantes para fins de classificação foram identificados;
3) Atribuição de peso ao termo: um peso foi atribuído para cada termo selecionado na etapa 2;

4) Aprendizado do Classificador: usaram-se as entradas com seus respectivos pesos atribuídos na etapa 3 para gerar o modelo classificador. Esse processo envolveu o aprendizado por SVM;
5) Classificação propriamente dita.

KINTO (2005) mostrou que um documento deve ser de alguma forma, convertido em números para que seja possível trabalhar usando métodos computacionais. Isso foi feito criando um vetor de palavras (elemento principal para a Classificação de textos) que contém as ocorrências dos termos no domínio de estudo, de forma que cada um dos termos teve um número associado. A seleção e atribuição de peso referiram-se ao Pré-Processamento.

O AM ocorreu nas etapas 2 (seleção do termo) e 4 (aprendizado do modelo classificador). Os pesos dos atributos foram números que indicavam a importância de uma palavra para fins de classificação ou recuperação da informação.

Uma categorização de textos totalmente automática pode ser muito custosa devido à necessidade de se prever todas as possíveis situações que o sistema irá presenciar, dessa forma, o trabalho utilizou uma abordagem visando uma classificação semi-automática. A abordagem foi a *Ranking* de categoria (CT-RC), onde, dado um texto, listam-se as *n* mais prováveis categorias.

Um vetor de índice muito extenso poderia inviabilizar a Classificação de textos, com isso uma redução da dimensão foi feita por meio da eliminação de informações redundantes, remoção de conectivos (conjunções coordenativas, conjunções subordinadas, pronomes, advérbios, artigos e preposições).

O classificador foi avaliado pelas medidas *precision*, *recall*, *accuracy* e suas variações.

O algoritmo *Sequential Minimal Optimization* (SMO) desenvolvido por John C. Platt (PLATT, 1998) para resolução de um problema de programação quadrática foi utilizado.

O Classificador de textos reduzidos foi composto em três partes principais:

- Módulo de captura: responsável pela captura de informações (páginas da *web e-commerces*), armazenando seu conteúdo em arquivos e banco de dados.
- Módulo de Pré-Processamento: onde foram realizadas filtragem de palavras irrelevantes, normalização das mesmas, e atribuição de pesos de atributos para as palavras.
- Módulo de Classificação: realizado durante o processo de aprendizado do SVM, bem como na fase de classificação para auxílio à tomada de decisão humana.

A linguagem de programação utilizada foi PERL (PERL, 2010), devido à facilidade no tratamento de *strings*[4], por ser gratuita, por ser muito utilizada por pesquisadores da área, nativa em Linux e expansível para Windows e por ter um bom tempo de desenvolvimento de novos módulos, muito inferior a outras linguagens.

A obtenção dos textos foi feita por meio de uma ferramenta chamada *"Spider"* desenvolvida pela empresa "BuscaPé" (BUSCAPE, 2010). Sendo assim, não foi necessário a implementação da etapa inicial. Outra etapa eliminada graças ao uso da base de dados com textos previamente capturados foi a Classificação dos textos utilizados nos treinos/testes. A fase de Pré-Processamento foi realizada com auxilio de um SGBD, como MySQL (2010), facilitando a manipulação das informações.

Baseado em testes preliminares e a literatura consultada, Kinto optou pelo uso da abordagem Um Contra Um. Foram testados os *Kernels* Linear, RBF (função de base radial) e polinomial, chegando à conclusão de que o RBF seria o mais indicado para classificação de textos.

A taxa de acerto do um multi-classificador Um Contra Um baseado em SVM foi em média 80%. Um valor considerado bom.

O classificador de texto mostrou-se fundamental para aumentar a produtividade no processo de Classificação de textos reduzidos. Comparações entre o tempo gasto do classificador automatizado *vs.* trabalho humano, mostraram que o ganho de tempo final ficou em torno de 13 dias.

2.4.2 Detecção de fraude na distribuição de energia elétrica utilizando SVM

SILVA e SCARPEL (2007) aplicam SVM para detectar fraudes na distribuição de energia elétrica. Sua justificativa se deu pela alta necessidade de segurança nos mais diversos setores de serviços, propensos a atividades ilícitas. As fraudes representam perdas significativas de receita para as empresas. São representadas pelo consumo "gratuito" dos produtos/serviços oferecidos. Segundo a Associação Brasileira de Distribuidores de Energia Elétrica, em amostra com 18 principais companhias de distribuição de energia, detentoras de 81% da energia elétrica distribuída nacionalmente, o índice de perda comercial através de fraudes esteve em 5,0% em 2004, nível considerado alto, já que o nível mundial foi de 1%.

[4]Cadeia de caracteres.

O desempenho do modelo desenvolvido foi comparado ao desempenho da técnica de Análise Discriminante Linear que, historicamente, é o modelo quantitativo mais utilizado na criação de modelos de classificação.

Os dados trabalhados pelos autores foram de uma empresa distribuidora de energia elétrica. Considerou-se as informações relacionadas a questões geográficas, perfil do cliente e padrões de consumo. A base de dados continha dados de 596 clientes de um determinado período, onde 298 representavam clientes fraudulentos e 298 representavam clientes honestos.

As informações disponíveis para cada cliente e que serviram como parâmetros de avaliação foram:

- Código do cliente;
- Tempo de residência;
- Valor da ultima conta;
- Valor da conta média;
- Região (Norte, Sul, Leste, Oeste);
- Forma de pagamento;
- Variação de valor da última conta;
- Fraude (representando se o cliente é fraudulento ou não).

As validações do modelo foram realizadas utilizando matrizes de confusão. Na comparação de eficiência e desempenho do SVM, utilizaram-se duas funções: Linear e Polinomial de grau 2. Em ambas as análises foram adicionadas um custo *C* (erro), para que o modelo fosse suscetível a possíveis ruídos. Os valores para a constante de penalização foram C_1: 0,01 e C_2: 10. Os resultados obtidos estão sintetizados na Tabela 2.3.

Tabela 2.3: Resultados da Classificação utilizando SVM Linear e Polinômial, Configurados com C1 = 0,01 e C1=10, para classificação de clientes fraudulentos e honestos em uma empresa de Energia Elétrica (SILVA, SCARPEL, 2007)

	$C_1 = 0,01$		$C_1 = 10$	
Modelo	**Fraudes**	**Honestos**	**Fraudes**	**Honestos**
Função Linear	79,9%	61,7%	74,5%	73,8%
Polinômio de grau 2	74,5%	66,4%	73,2%	77,9%

Utilizando Análise de Discriminante, os resultados foram de 73,3% de acerto tanto para Fraudes quanto Honestos.

A partir dos resultados, observou-se que ao utilizar uma constante de penalização de baixo valor, os modelos mostraram maior eficiência para clientes fraudulentos. Sendo neste contexto a melhor solução, pois o principal objetivo de uma distribuidora de energia elétrica é identificar clientes que cometem fraudes.

Os resultados mostraram que a técnica é uma alternativa viável para a tarefa de Classificação.

2.4.3 Máquinas de Vetores de Suporte na Classificação de Impressões Digitais

LIMA (2002) apresentou um estudo de SVM empregado na classificação de impressões digitais. Justificou-se por motivos de segurança, mais especificamente para a identificação de pessoas.

No trabalho, a Biometria é definida como a(s) característica(s) que identifica(m) de forma única um indivíduo. Qualquer característica fisiológica ou comportamental pode ser usada para construir uma métrica que satisfaça total ou parcialmente os seguintes requisitos:

1. Universalidade: todas as pessoas devem ter a(s) característica(s);
2. Unicidade: duas pessoas não podem ter a(s) mesma(s) característica(s);
3. Permanência: a(s) característica(s) deve(m) ser invariante(s) com o tempo;
4. Coletabilidade: a(s) característica(s) pode(m) ser medida(s) quantitativamente.

Exemplos clássicos de sinais biométricos são as impressões digitais, geometria da mão, íris, retina, face, voz e DNA. Analisando os 4 requisitos necessários para ser utilizado como métrica, o sinal biométrico que melhor os atendeu é a impressão digital, do qual tem-se a aplicação em controle de acesso, pagamentos de benefícios, carteira de motorista entre outros.

Os métodos mais comuns para aquisição de digitais são em tinta e *scanner*. As características analisadas nas impressões digitais são as minúcias como terminações e bifurcações, poros e campo direcional. Os dois tipos de características relacionadas ao mapa de direção são os pontos *delta*[5] e *core*[6] que são denominados de pontos de singularidade.

Os principais propósitos de classificar impressões digitais apresentados foram facilitar o gerenciamento de grandes bancos de dados de impressões digitais e acelerar o processo de identificação.

[5] O ponto *core* é definido como ponto mais ao topo sobre a linha curva mais interna.
[6] O ponto *delta* é o centro de uma região triangular onde três diferentes direções se encontram.

As impressões digitais foram classificadas em cinco grupos: *"Plain Arch" (A)*, *"Tented Arch" (T)*, *"Right Loop" (R)*, *"Left Loop" (L)* e *"Whorl" (W)*. "Essas classes são determinadas pelo fluxo da linha sobre a área core e o número e as localizações relativas dos pontos *core* e *delta"* (LIMA, 2002). Quando uma impressão digital não se enquadra em nenhuma dessas classes elas são associadas a uma classe chamada *acidental*.

O critério de desempenho do sistema automático de verificação foi baseado na taxa de falsas rejeições (sistema não autentica corretamente um usuário autorizado) e na taxa de falsas aceitações (autentica um usuário não autorizado).

Os testes foram realizados sobre um conjunto de 4.000 imagens de impressão digitais, igualmente distribuídas nas cinco classes. Os resultados obtidos por Lima para a estratégia Um Contra Um estão na Tabela 2.4.

Tabela 2.4: Resultado da estratégia Um Contra Um, aplicado a 5 tipos de impressões digitais (A, L, R, T e W) (LIMA, 2002)

Classe	Precisão
AL	92,38%
AR	91,37%
AT	73,18%
AW	95,99%
LR	98,84%
LT	87,74%
LW	93,66%
RT	87,96%
RW	94,16%
TW	96,49%

O método Um Contra Um obteve média de 91,18%. Também foram realizados testes utilizando a estratégia Um Contra Todos. O método Um Contra Todos obteve uma média de acerto de 90,43%. Os resultados obtidos estão listados na Tabela 2.5.

Tabela 2.5: Resultado da estratégia Um Contra Todos, aplicado a 5 tipos de impressões digitais (A, L, R, T e W) (LIMA, 2002)

Classe	Precisão
(A)LRTW	87,83%
(T) ALRW	83,95%
(L) ARTW	93,52%
(R) ALTW	91,94%
(w) ALRT	94,96%

2.4.4 Análise de Crédito Bancário com o Uso de Modelos de Regressão Logística, Redes Neurais e *Support Vector Machine*

A proposta de GEVERT (2009) é a análise de crédito bancário em instituições financeiras. Segundo o mesmo, a concessão de crédito é o estudo da situação financeira do indivíduo que deseja o crédito, verificando se a pessoa tem condições de pagar certa dívida a contrair, ou no mínimo garantir com outro bem o custo nas condições contratadas. Nesta análise, são levadas em conta informações obtidas através de várias fontes externas como Serasa (2010) e SPC (2010) (Serviço de Proteção ao Crédito). Neste contexto, precisam distinguir, entre clientes que pagarão a dívida contraída, e quais não honrarão o compromisso assumido.

Também segundo o autor, embora os analistas consigam apontar os fatores que influenciam nas decisões, muitas vezes não conseguem descrever o processo de tomada de decisão. Pois estes ambientes são dinâmicos, com constantes alterações, onde as decisões devem ser tomadas rapidamente, surgindo assim a necessidade de ter ferramentas que auxiliem o analisa financeiro na tomada de decisão.

O objetivo do trabalho de Gevert foi comparar o uso das técnicas de Regressão Logística, Redes Neurais Artificiais e SVM.

Foram utilizados dados reais de uma agência bancária na cidade de Wenceslau Braz, do estado do Paraná, com unidades em todo o Brasil. Das amostras utilizadas de 199 empresas, 64 são inadimplentes e 135 são adimplentes, das quais foram extraídas 21 informações, escolhidas pela necessidade do preenchimento do cadastro requerido pela agência. Entre as informações estiveram: existência de restrição em nome da empresa, tempo de conta em banco, setor de atividade, tempo de atividade, número de funcionários, faturamento, etc.

Usou-se *cross validation*, optando por separar os dados na proporção 80-20 para o conjunto de treinamento/teste respectivamente.

Na implementação da técnica de Regressão Logística, utilizou-se o *software Statistica*. Os resultados atingiram 100% de classificação correta na fase de treinamento e de 96,99% na fase de teste, logo a média de acerto do modelo é de 98,5%.

Para a implementação da técnica de Redes Neurais Artificiais, foi utilizado o *software* MATLAB – *Neural Networks Toolbox* (MATLAB, 2010). Os resultados atingiram acerto de 93,39% e 90,16% nas fases de treinamento e testes, respectivamente, sendo a média de acerto do modelo de 91,78%.

Para a implementação do SVM foi utilizado um programa desenvolvido por ALES *et. al.* (2009) em linguagem *Visual Basic* (VISUAL BASIC, 2010) com o uso do algoritmo SMO.

Foram testados os *Kernels* Gaussiano, Polinomial e Sigmoidal, sendo o melhor resultado obtido pelo *Kernel* Polinomial homogêneo, alcançando 99,01% de acerto.

A Tabela 2.6, mostra os resultados obtidos pelos três métodos. Todos eles apresentaram bons resultados de classificação.

Tabela 2.6: Comparação da classificação dos métodos de Regressão Logística, Redes Neurais Artificiais e SVM, sobre os dados de empresas para adquirir crédito bancário (GEVERT, 2009)

	Média de Acertos		
	Treinamento	**Testes**	**Geral**
Regressão Logística	100,00%	96,99%	98,5%
Redes Neurais Artificiais	93,39%	90,16%	91,78%
SVM	99,68%	99,05%	99,37%

Gevert concluiu que a técnica mais indicada para a predição de empresas adimplentes e inadimplentes é a técnica SVM, que proporcionou a média de resultados de 99,37%.

2.4.5 Considerações Finais

Pode-se encontrar também o uso de SVM em reconhecimento de fala, reconhecimento facial, detecção de spam, análise de imagens, classificação de defeitos em couro bovino, bioinformática, entre outras aplicações. Uma lista com mais de 40 possíveis aplicações de SVM pode ser encontrada em GUYON (2010).

A SVM apresentou resultados superiores a outras técnicas de classificação, a exemplo de Redes Neurais Artificiais, sendo considerada uma boa técnica de classificação.

Este Capítulo abordou trabalhos relacionados à utilização de SVM para a tarefa de classificação, evidenciando que SVM é uma técnica interessante para análise de dados.

Capítulo 3

Experimento Realizado e Resultados

O objetivo deste Capítulo é apresentar de forma experimental, como os conceitos introduzidos no Capítulo anterior foram aplicados no trabalho, além dos testes realizados com seus respectivos resultados.

O foco do trabalho está no projeto e desenvolvimento da ferramenta YADMT – *Yet Another Data Mining Tool* contribuindo com a implementação do algoritmo de SVM para o módulo de Classificação, o qual é tema deste trabalho. Informações técnicas a respeito da ferramenta podem ser encontradas em BENFATTI *et al.* (2010). A ferramenta é baseada na arquitetura de *plugins*, onde cada módulo é acoplado à ferramenta por meio de interfaces. A Seção 3.1 apresenta como essa comunicação é realizada.

A Linguagem para desenvolvimento foi JAVA (2010), a mesma da YADMT. Foi utilizado o algoritmo *Sequential Minimal Optimization* (SMO), desenvolvido por John C. Platt (PLATT, 1998). A descrição do algoritmo encontra-se na Seção 3.2. Para compreensão do mapeamento das fórmulas em código fonte, o Apêndice A traz os pseudocódigos do algoritmo SMO.

Pelo fato do algoritmo SMO ser um classificador binário, para classificar bases de dados Multiclasses fez-se necessário o arranjo do conjunto de dados de treinamento, foram testados os métodos Um Contra Um e Um Contra Todos. A descrição destes testes encontra-se na Seção 3.3. Na Seção 3.4 é descrito como os testes foram realizados e seus resultados.

As bases de dados para avaliação e validação do módulo proposto foram escolhidas em conjunto para título de comparação com outras técnicas que fazem parte da ferramenta YADMT. A descrição das bases de dados escolhidas encontra-se no Apêndice B.

3.1 Comunicação da técnica SVM com a ferramenta YADMT

O primeiro módulo a ser implementado na ferramenta YADMT é o módulo de

Classificação, o qual terá além da SVM outros algoritmos, quais descritos em BENFATTI (2010) e BONIFACIO (2010). Esta Seção descreve detalhes de comunicação entre as técnicas implementadas e a ferramenta YADMT.

A comunicação entre os módulos e a ferramenta é feita através de *Interfaces* e *Annotations*[7], sendo assim, a primeira coisa a se fazer para desenvolver os módulos é a importação da biblioteca *InterfacesYADMT*, qual possui estruturas básicas para comunicação.

A Figura 3.1 demonstra as assinaturas dos métodos da Interface *ClassifierModuleInterface* que os módulos devem implementar. Os dois métodos principais desta Interface são os métodos *train()* e *test()* que executam as principais funcionalidades de um módulo de Classificação. O método *train()* é responsável por executar o treinamento do algoritmo de classificação, para isto ele recebe dois parâmetros, *Object input[][]* e *Object output[]*, o primeiro é uma matriz que contém os valores de cada tupla de entrada do módulo e o segundo é um vetor de valores que corresponde à saída desejada correspondente a cada tupla de entrada (rótulo de classes). O método *test()* recebe como parâmetro um vetor de valores, que representa um registro que vai ser classificado pelo módulo e retorna a classificação do respectivo registro.

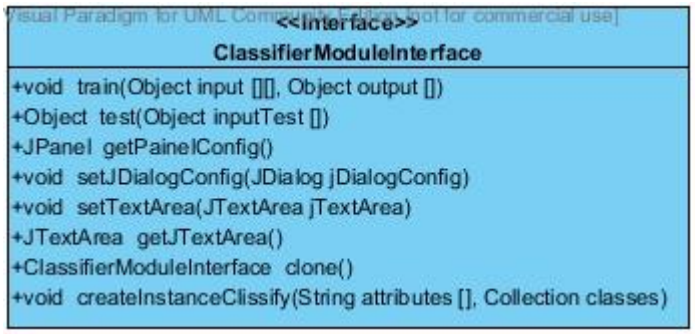

Figura 3.1: Diagrama de Classes da Interface ClassifierModuleInterface

Os demais métodos da Interface que um módulo de Classificação tem que implementar ocupam funções secundárias no funcionamento entre o módulo e a ferramenta. São eles:

- getPainelConfig(): retorna um painel contendo as opções de configuração de criação e treinamento de um algoritmo de Classificação. No caso da SVM, são os parâmetros *C*, *eps*, *tolerance*, *sigma quadrado* e se os dados são esparsos, tipo de *Kernel* a se utilizar na classificação e se é classificação binária;

[7] Um tipo de metadados que podem ser posteriormente interpretadas por um compilador.

- setJDialogConfig(): recebe como parâmetro o container que contém getPainelConfig;
- setTextArea(): recebe um objeto do tipo JTextArea que serve para o módulo de Classificação imprimir alguns dados durante a sua execução;
- createInstanceClassify(): responsável pela instanciação do algoritmo de classificação, para isto recebe como parâmetros o nome dos atributos de entrada e o número de classes que a base de dados possui.

3.2 O Algoritmo *Sequential Minimal Optimization* - SMO

Nesse trabalho, o algoritmo SVM utilizado é o *Sequential Minimal Optimization* (SMO) proposto por John C. Platt (PLATT, 1998). É um algoritmo de programação quadrática para resolver problemas de otimização, amplamente utilizado para o treinamento de SVM. O SMO quebra grandes problemas em séries de pequenos possíveis problemas, que são resolvidos analiticamente.

O problema de otimização utiliza apenas duas variáveis a cada iteração e admite uma solução analítica, eliminando a necessidade de resolver um problema quadrático da SVM.

Isto se dá trabalhando com dois Multiplicadores de Lagrange (LEITHOLD, 1994) de cada vez e mantendo os demais fixos. Considerando os dois multiplicadores a serem atualizados como *alph1*[8] e *alph2* e suas respectivas classes y_1 e y_2, a condição inicial para esse algoritmo é descrita na Equação 3.1.

$$\sum_{i=0}^{N} alph1_i * y_i = 0 \tag{3.1}$$

onde *N* é o número de tuplas. Quando um multiplicador for atualizado, essa condição obriga a ajustar o outro, para que a condição continue verdadeira.

A escolha dos dois pontos é determinada por uma heurística, enquanto a otimização dos dois multiplicadores é realizada analiticamente. Para não violar a condição da Equação 3.1, os novos Multiplicadores de Lagrange devem respeitar a seguinte condição:

$$a1.y_1 + a2.y_2 = constante = alph1.y_1 + alph2.y_2 \tag{3.2}$$

[8] Os nomes das variáveis utilizadas na descrição do SMO são as mesmas utilizadas no código-fonte para título de compreensão.

O algoritmo encontra primeiramente o valor para *a2* e sucessivamente utiliza-o para encontrar o *a1*.

Em virtude disso, os Multiplicadores de Lagrange além de possuírem os valores limitados entre 0 e *C*, restrição do problema do SVM, estão limitados por duas constantes *L* e *H*, para manterem a condição inicial verdadeira, ou seja, restrito pela condição:

$$H < a2 < L \tag{3.3}$$

onde:

Se y_1 for igual a y_2:

$$L = \text{Max}\ (0,\ alph1 + alph2 - C)$$
$$H = \text{Min}\ (C,\ alph1 + alph2) \tag{3.4}$$

Se y_1 for diferente de y_2:

$$L = \text{Max}\ (0,\ alph2 - alph1)$$
$$H = \text{Min}\ (C,\ C + alph2 - alhp1) \tag{3.5}$$

onde a_i é o novo valor do Multiplicador de Lagrange do ponto x_i e $alph_i$ é o valor anterior.

O parâmetro *C* controla o compromisso entre a complexidade da máquina e o número de pontos não-separáveis; por isso, pode ser visto como uma forma de parâmetro de "regularização". O parâmetro *C* deve ser selecionado pelo usuário.

O valor da função em x_i que denota a função atual determinada pelos valores dos Multiplicadores de Lagrange e por *b* no estágio atual da aprendizagem é dado por:

$$f(x_i) = \sum_{j=1}^{N} a_j y_j\ k\left(x^j, x^i\right) + b \tag{3.6}$$

O valor de E_1 determina a diferença de $f(x^i)$ e o padrão y_i a que pertence o ponto x^i, ou seja, é a distância do ponto ao hiperplano atual dado pela atualização dos Multiplicadores de Lagrange, é dado por:

$$E_1 = f(x^i) - y_i \tag{3.7}$$

Esta quantidade adicional E_1 exigida é a segunda derivada da função objetivo ao longo da linha diagonal, que pode ser expressa por *eta*, Definido pela Equação 3.8:

$$eta = K(x^1,x^1) + K(x^2,x^2) - 2K(x^1,x^2) = ||\phi(x^1) - \phi(x^2)||^2 \tag{3.8}$$

Onde x_1 e x_2 são os pontos associados a a_1 e a_2 respectivamente, e ϕ é o mapeamento no espaço de características.

O máximo valor da função objetivo será obtido com o valor:

$$a2 = \begin{cases} H, se\ a2 > H \\ a2, se\ L \leq a2 \leq H \\ L, se\ a2 < L \end{cases} \tag{3.9}$$

sendo *a2* um valor truncado, ou seja, limitado por $L \leq a2 \leq H$, tem-se que:

$$a2 = alph2 + \frac{y_2(E_1 - E_2)}{eta} \tag{3.10}$$

onde E_1 é dado na Equação 3.7, *eta* é dado na equação 3.8, *H* e *L* são dados pelas Equações 3.4 e 3.5. O valor de *a1* é obtido de *a2*, como segue (Equação 3.11):

$$a1 = alph1 + y_1 y_2(alph2 - a2) \tag{3.11}$$

Analisando as equações, têm-se duas heurísticas, uma para escolha de *a1*, e outra para escolha de *a2*. Na primeira heurística, o ponto x^2 é escolhido entre os pontos que violam as condições de KKT. O algoritmo percorre todo o conjunto de dados de treinamento que violam as condições de KKT e seleciona um para atualizar, isso é feito através de um dos testes: $E_2 y_2 < -tol$ e $alph2 < C$ ou $E_2 y_2 > tol$ e $alph2 > 0$. Quando tal ponto é encontrado, utiliza-se a segunda heurística para selecionar o ponto x^1, que deve ser escolhido de tal maneira que seja atualizado com x^2, causando um grande acréscimo na função objetivo dual.

Para encontrar um bom ponto sem muitos cálculos, uma heurística rápida escolhe x^1, maximizando o valor dado por $|E_1 - E_2|$, se E_2 é positivo, o SMO escolhe o x^1 com o menor E_1, e se E_2 é negativo, então o SMO escolhe x^1 com o maior E_1.

Se esta escolha falhar em obter um acréscimo significante na função objetivo dual, o SMO experimenta cada ponto x^1 que tenha valores de α diferente dos limites, ou seja, $0 < \alpha < C$, aleatoriamente. Se ainda não houver progresso significativo, o SMO procura por todo o conjunto de dados de treinamento para encontrar um ponto x^1 adequado.

Se ocorrer alteração de valores, a heurística retorna para escolher outros pontos x^2 e x^1, até que todos os pontos estejam obedecendo às condições de KKT. Caso contrário termina o processo.

A lista dos erros de todos os pontos de treinamento é mantida na memória para reduzir contas adicionais.

A solução que satisfaz as condições de complementaridade de KKT (Equação 2.16) é equivalente a:

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i f(x^i) \geq 1 \\ 0 \leq \alpha_i \leq C &\Leftrightarrow y_i f(x^i) = 1 \\ \alpha_i = C &\Leftrightarrow y_i f(x^i) \leq 1 \end{aligned} \tag{3.12}$$

Após cada iteração, em que as condições de KKT são satisfeitas para ambos os pontos x^1 e x^2. Os valores podem ser atualizados analisando sempre o valor atual com o anterior da função $f(x^i)$.

O valor de b para um ponto x^1 definido por b_1, qual deve forçar a saída do SMV para y_1 quando a entrada for o ponto x^i, é dado por:

$$b_1 = -[E_1 + y_1(a1 - alph1)\, K(x^1,x^1) + y_2\,(a2 = alph2)\, K(x^1,x^2) + b \tag{3.13}$$

O valor de b para o ponto x^2 é definido por b_2, que deve forçar a saída do SVM para y_2 quando a entrada for o ponto x^2, é dado por:

$$b_2 = -[E_2 + y_1(a1 - alph1)\, K(x^1,x^2) + y_2\,(a2 = alph2)\, K(x^2,x^2) + b \tag{3.14}$$

Se os valores de b_1 e b_2 forem iguais, este será o novo valor de b, ou seja, $b^{new} = b_1 = b_2$.

Caso contrário, o intervalo entre b_1 e b_2 são todos os *thresholds*[9] que são consistentes com as restrições de KKT, portanto o SMO escolhe o *threshold* que está no meio do intervalo, ou seja:

$$b^{new} = \frac{b_1 + b_2}{2} \tag{3.15}$$

Quando os dados são separados linearmente, pode-se atualizar o valor do vetor ***w*** por:

$$w^{new} = w^{old} + y_1(a1 - alph1)x^1 + y_2(a2 - alph2)x^2 \tag{3.16}$$

Os erros E_1 são atualizados a cada iteração por:

$$E_i^{new} = E_i^{old} + y_i(a1 - alph1)\,K(x^1, x^i) + y_2\,(a2 = alph2)\,K(x^2, x^i)\,b^{new} - b^{old} \tag{3.17}$$

O objetivo da heurística SMO é obter os valores dos Multiplicadores de Lagrange para que estes tenham os erros tendendo à zero. É incorreto afirmar que $E1 = 0$ e $E2 = 0$, pois nas primeiras iterações, os valores dos Multiplicadores de Lagrange não estão ótimos, eles ainda poderão ser atualizados, portanto podem existir erros ainda. A atualização da função objetivo pode ser feita pelo *gap*, que é a diferença entre a função objetivo atual e anterior:

$$\begin{aligned} gap = (a1 - alph1) + (a2 - alph2) - \left[\sum_{j=1}^{2}\sum_{i=1}^{N} y_j y_i (a_j - alph_j) K(x^j, x^i)\right]_{\substack{i \neq 1 \\ i \neq 2}} \\ + -\frac{1}{2}y_1^2(a1 - alph1)\,K(x^1, x^1) - \frac{1}{2}y_2^2\,(a2 - alph2)K(x^2, x^2) \\ + -y_1 y_2[a1\ a2 - alph1\ alph2]\,K(x^1, x^2) \end{aligned} \tag{3.18}$$

Para melhor entender o processo, a seguir é apresentado o fluxograma da heurística dividido em três partes principais. O primeiro fluxograma detalha a escolha do primeiro ponto, analisando se esse ponto está ou não violando as condições de KKT. No segundo fluxograma, tem-se a escolha do segundo ponto, que será otimizado juntamente com o

[9] Um *threshold* ou limiar é um valor mínimo de alguma quantidade.

primeiro ponto escolhido, e no terceiro fluxograma, a otimização dos pontos.

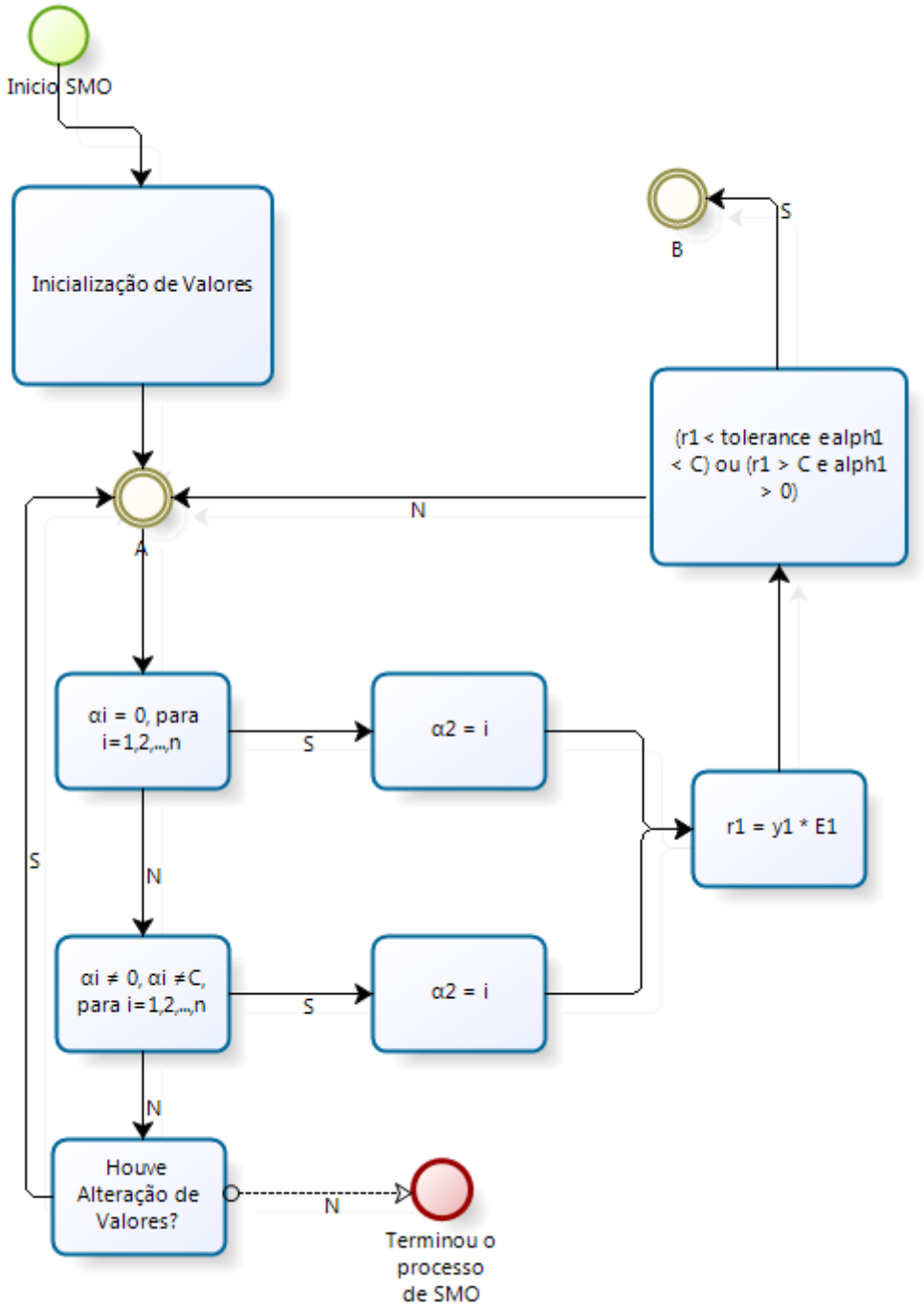

Figura 3.2: Fluxograma da escolha do primeiro ponto pelo SMO. Adaptado de (GEVERT, 2009)

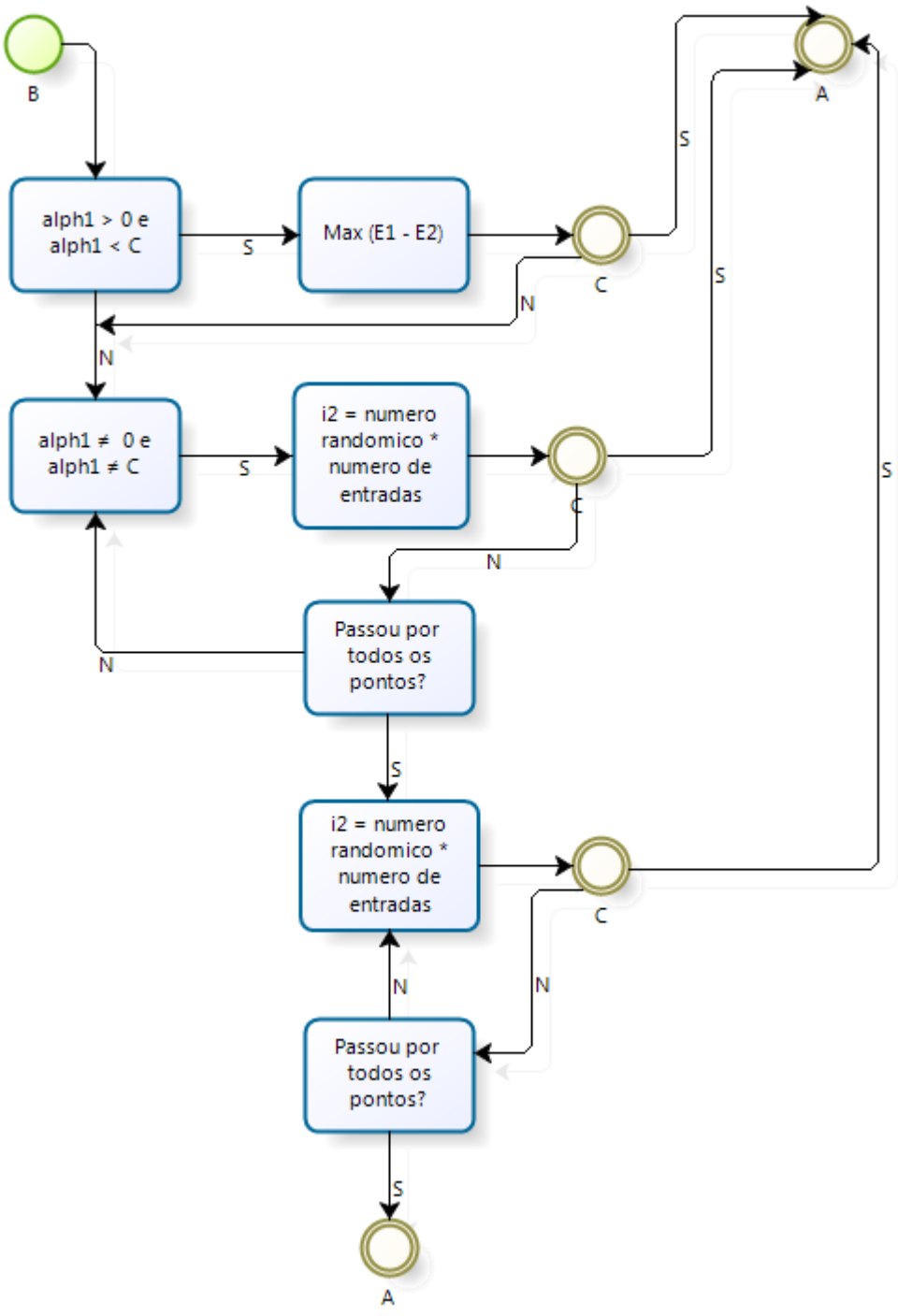

Figura 3.3: Fluxograma da escolha do segundo ponto pelo SMO. Adaptado de (GEVERT, 2009)

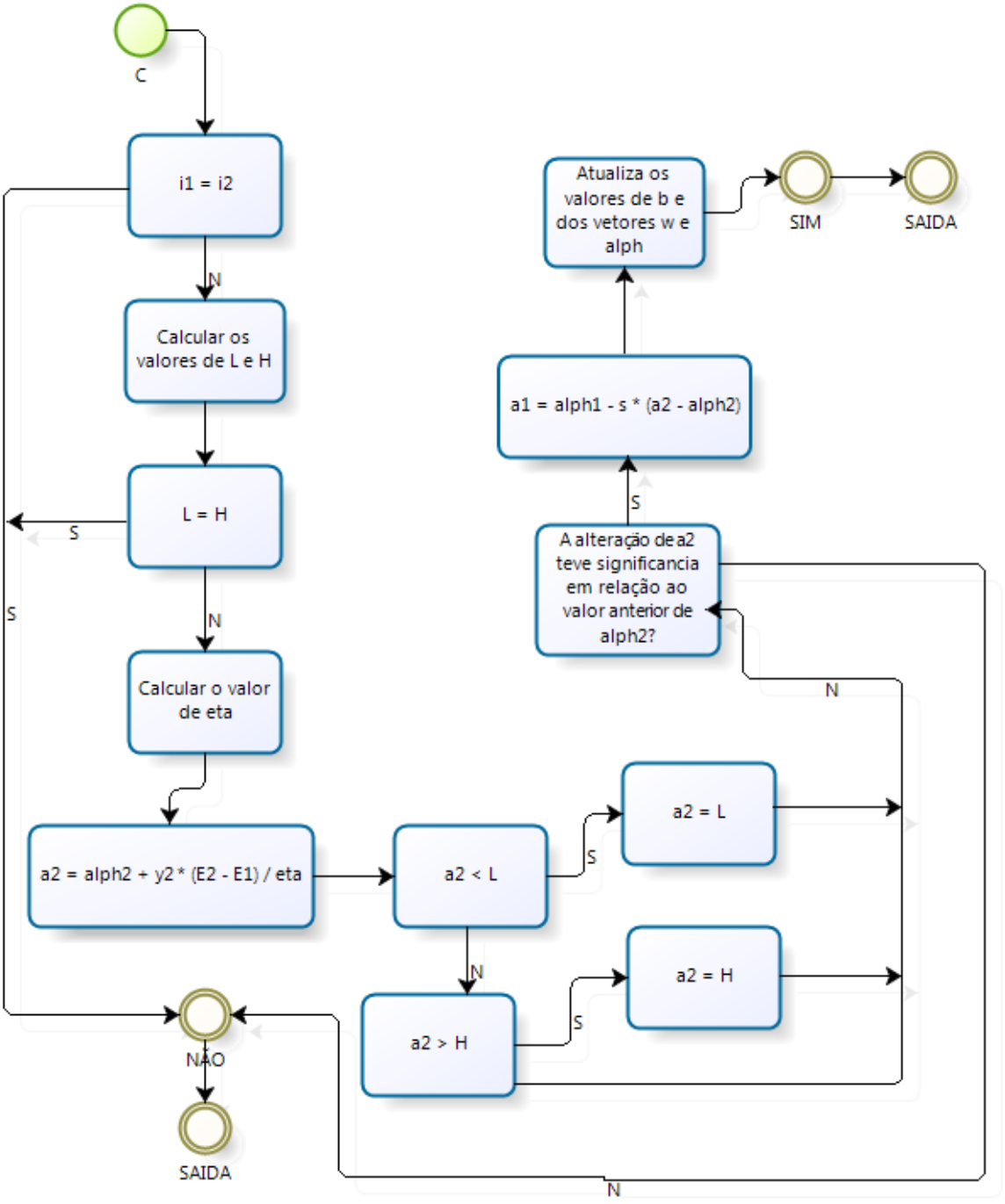

Figura 3.4: Fluxograma da otimização dos pontos escolhidos pelo SMO. Adaptado de (GEVERT, 2009)

3.3 Classificação Multiclasse

Segundo PLATT *et. al.* (2000) a classificação binária utilizando SVM é uma tarefa bem desenvolvida, entretanto, problemas reais podem apresentar diversas classes e estes são solucionados utilizando-se de alguns métodos como o Um Contra Um e Um Contra Todos, descritos na Seção 2.2.4.

A título de avaliar os dois métodos de separação Multiclasse, foi escolhida uma base de dados baseada em alguns requisitos. São eles:

- Número de classes superior a 3, para haver um número diferentes de classificadores gerados;
- Diferentes números de amostras por classe, para avaliar os métodos quando uma classe tem uma menor representatividade que a outra.

A base de dados que se enquadrou nos requisitos foi a *Ecoli*, que possui 8 classes com uma distribuição desigual entre as classes. Maiores informações sobre a base *Ecoli*, assim como as demais, encontram-se no Apêndice B.

Para o método Um Contra Um, a partir do conjunto de dados de entrada, foram combinados em outros 28 conjuntos de dados, como prega o método.

Os resultados de cada par de entradas, tempo gasto e a média obtida pelo método Um Contra Um estão dispostos na Tabela 3.1:

Tabela 3.1: Treinamentos e testes do método Um Contra Um para a base *Ecoli*

Classes	Acuidade Treino	Acuidade Teste	Tempo Treino (ms)	Tempo Teste (ms)
1x2	0,6712	0,6756	401	320
1x3	0,9895	0,9795	285	225
1x4	0,9895	0,9795	296	229
1x5	0,805	0,7999	335	244
1x6	0,8796	0,8727	287	261
1x7	0,9693	0,96	283	231
1x8	0,7307	0,7384	403	281
2x3	0,9807	0,9629	235	222
2x4	0,9807	0,9629	227	221
2x5	0,6891	0,6842	301	233
2x6	0,7968	0,7878	262	225
2x7	0,9444	0,9285	275	220
2x8	0,593	0,6046	356	236
3x4	1	0,5	193	188

3x5	0,9583	0,923	219	373
3x6	0,9285	0,875	205	212
3x7	0,75	0,6666	192	217
3x8	0,9722	0,9444	235	202
4x5	0,9583	0,923	233	214
4x6	0,9285	0,875	207	192
4x7	0,75	0,6666	213	179
4x8	0,9722	0,9444	225	197
5x6	0,6388	0,6315	240	199
5x7	0,8846	0,8571	229	207
5x8	0,6034	0,5862	256	211
6x7	0,8125	0,7777	216	192
6x8	0,7291	0,7083	295	211
7x8	0,921	0,8947	231	185
Média	**0,8509**	**0,811**	**261**	**225**

Analisando os dados da Tabela 3.1, constata-se que o classificador obteve uma média de 85,09% de acerto no treino e de 81,1% de acerto nos testes. A média de tempo gasto foi pequena, em torno de 261 ms (milissegundos) para treino e de 225 ms para teste.

A Tabela 3.2 que apresenta os testes referentes ao método Um Contra Todos mostra que o método obteve um desempenho de 92,21% de acerto no treinamento e de 90,44% de acerto nos testes. Embora o tempo fora maior que o método anterior, ainda assim é considerado baixo em termos computacionais, 522 ms para treino e 306 ms para teste.

Tabela 3.2: Treinamentos e testes do método Um Contra Todos para a base *Ecoli*

Classes	Desempenho Treino	Desempenho Teste	Tempo Treino	Tempo Teste
1xTodos	0,9375	0,8214	637	275
2xTodos	0,7713	0,7699	526	287
3xTodos	0,9955	0,9911	428	317
4xTodos	0,9955	0,9911	431	316
5xTodos	0,8968	0,8938	534	284
6xTodos	0,9417	0,938	518	331
7xTodos	0,9865	0,9823	437	290
8xTodos	0,8437	0,8482	665	351
Média	**0,921**	**0,9044**	**522**	**306**

Nota-se que o método Um Contra Todos obteve um desempenho superior ao método Um Contra Um. Com base nestas informações escolheu-se o método Um Contra Todos para empregar nas demais bases de dados.

Ambos os testes foram realizados com as seguintes configurações do SMO:

- *C*: 0,05;
- *eps*: 0,001;
- *Sigma quadrado*: 2;
- *tolerance*: 0,001.

Configurações do computador:

- Intel Core 2 Duo, 1.83 GHz;
- 2 GB de memória RAM.

3.4 Testes Realizados

A fim de testar o módulo, algumas bases de dados foram utilizadas para teste. Para obter parâmetros válidos a fim de testar a eficiência da técnica, foram testadas bases de dados clássicas na área de DM obtidas do repositório público UCI (*University of California Irvine*) (UCI, 2010). A Tabela 3.3 traz as bases de dados utilizadas nos testes. O Apêndice B apresenta uma descrição detalhada de cada base de dados utilizada.

Tabela 3.3: Bases de dados escolhidas para teste

Nome da Base	Tamanho	Atributos	Classes	Atributos Categóricos
Abalone	4177	8	28	1
Acute Inflammations	120	6	2	5
Ecoli	336	7	8	0
Iris	150	4	3	0
Libras Movement	360	90	15	0
Statlog (Image Segmentation)	2310	19	7	0
Teaching Assistant Evaluation	151	5	3	3
Wine Quality	4898	11	7	0

As bases de dados para testes foram escolhidas a partir de parâmetros que permitam avaliar o classificador SVM em diversas situações. Alguns parâmetros avaliados são:

- Dados linearmente separáveis;
- Dados não linearmente separáveis;
- Variação do tamanho das bases de dados;
- Quantidade de classes em relação aos dados;

- proporcionalidade dos dados em cada classe.

Neste trabalho a classificação adotada para as bases de dados é a mesma proposta por ZHENG (2006), da seguinte forma:

- Quanto ao número de registros:
    - Pequena: menos de 210 registros (Iris, Teaching Assistant Evaluation, Acute Inflammations);
    - Média: entre 210 e 3170 registros (Ecoli, Statlog (Image Segmentation), Libras Movement);
    - Grande: mais de 3170 registros (Abalone, Wine Quality).
- Quanto ao número de atributos
    - Pequena: menos de 10 atributos (Abalone, Ecoli, Iris, Teaching Assistant Evaluation, Acute Inflammations);
    - Média: entre 10 e 30 atributos (Wine Quality, Statlog (Image Segmentation));
    - Grande: mais 30 atributos (Libras Movement).
- Quanto ao número de classes:
    - Pequena (binário): duas classes (Acute Inflammations);
    - Média: entre 3 e 10 classes (Ecoli, Iris, Teaching Assistant Evaluation, Statlog (Image Segmentation));
    - Grande: mais de 10 classes (Abalone, Wine Quality, Libras Movement).

Os critérios de avaliação utilizados foram acuidade, velocidade e escalabilidade.

De acordo com ZHENG (2006), um classificador tem alta taxa de acuidade quando a porcentagem da taxa de acertos for maior que 75%, média quando estiver entre 40 e 75% e baixa quando for menor que 40%. Estes valores serão considerados na análise dos resultados.

Quando necessário, bases de dados que contenham atributos ou classes em caracteres, tiveram seus parâmetros mapeados para uma entrada aceita pela SVM. Um exemplo é a base de dados *Iris*, onde as classes "*Iris-Setosa*", "*Iris-Versiolour*" e "*Iris-Virginica*", foram mapeados para as classes 1, 2 e 3, respectivamente.

A fim de avaliar a SVM, foi adotado o método de particionamento do conjunto de dados de entrada *Holdout*, que consiste em dividir uma parte da base de dados para treinamento e outra para teste. A proporção utilizada foi de 2/3 das amostras para treino, e 1/3 para testes. O particionamento das tuplas se deu de forma manual, a fim de obter uma representatividade igualitária das classes para treino e teste.

As configurações do SMO para os testes foram as mesmas utilizadas para escolha do método Multiclasse, descrito na Seção anterior.

As configurações dos computadores utilizados para testes foram:

- AMD Athlon Dual Core, 2.60 GHz;
- 2 GB de memória RAM.

Na seção seguinte, os resultados obtidos são detalhados e analisados.

3.5 Resultados

Os experimentos foram gerados a partir do módulo de SVM implementado para a ferramenta YADMT. Os resultados para as bases de dados selecionadas em relação à acuidade estão dispostos na Tabela 3.4. Os dados estão ordenados da maior acuidade nos testes para a menor.

Tabela 3.4: Acuidade da SVM nas bases de dados escolhidas no treino e teste

Base de Dados	Acuidade Treinamento	Acuidade Teste
Abalone	0,9655	0,9655
Statlog (Image Segmentation)	0,9635	0,9581
Ecoli	0,9210	0,9044
Wine Quality	0,8571	0,8571
Libras Movement	0,9597	0,7444
Iris	0,9199	0,6999
Teaching Assistant Evaluation	0,6655	0,6688
Acute Inflammations	1,000	0,525

O melhor resultado foi obtido pela base de dados *Abalone*, atingindo acuidade de 96,55% no treino/teste. Analogamente, a base de dados *Acute Inflammations* obteve a pior acuidade nos testes, que foi de 52.5%, apesar de classificar 100% das amostras corretamente no treinamento, possivelmente ocasionado por um super ajustamento das tuplas de treinamento ou conjuntos distintos de tuplas para treino e teste. Em geral, seguindo o método de avaliação de ZHENG (2006) o resultado foi satisfatório, obtendo alta taxa de acuidade (maior que 75%) nas bases de dados *Abalone*, *Statlog* (*Image Segmentation*), *Ecoli*, *Wine Quality*, média taxa de acuidade (entre 40% e 75%) nas bases de dados *Libras Movement*, *Iris*, *Teaching Assistant Evaluation* e *Acute Inflammations*, e nenhuma base com baixa taxa de acuidade (menor que 40%).

Quanto à velocidade, os resultados dos testes da aplicação da SVM sobre as bases de dados escolhidas estão listados na Tabela 3.5. Os dados estão ordenados do menor tempo de treinamento para o maior.

Tabela 3.5: Tempo gasto para treinamento e teste da SVM nas bases de dados escolhidas

Base de Dados	Número de Amostras da base	Número de Atributos da base	Tempo Treinamento (ms)	Tempo Teste (ms)
Iris	150	4	358	338
Ecoli	336	7	408	350
Acute Inflammations	120	6	681	379
Teaching Assistant Evaluation	151	5	1055	316
Libras Movement	360	90	2402	1141
Abalone	4177	8	2800	994
Wine Quality	4898	11	2446692	1148
Statlog (Image Segmentation)	2310	19	2655373	1316

Analisando a Tabela 3.5, observa-se que o tempo de treinamento é diretamente proporcional ao número de amostras e ao número de atributos da base de dados, pois o algoritmo trabalha sobre todos os atributos de todas as amostras, conseqüentemente aumentando o tempo de processamento.

Comparando o tempo de custo com a acuidade das bases de dados, observa-se que os parâmetros não têm relação de proporcionalidade, pois tanto a base *Statlog (Image Segmentation)* que teve um tempo elevado de treinamento e a base *Ecoli* que teve um baixo tempo de treinamento, obtiveram acuidades altas, superiores a 90%.

Quanto à escalabilidade, a SVM se portou de maneira satisfatória, pois conseguiu obter acuidades altas tanto em bases de dados com um pequeno número de registros, como em bases de dados com um grande número de registros. Neste quesito nota-se algo interessante, onde bases de dados com maiores números de registros tiveram acuidades maiores que bases de dados com pequeno número de registros.

Capítulo 4

Conclusões

Este trabalho propôs o estudo teórico-prático da aplicação de SVM na classificação de dados, objetivando a avaliação da técnica empregada a diferentes bases de dados.

A técnica se mostrou bastante útil, cumprindo seus propósitos. Confirmando resultados obtidos em outros trabalhos correlatos, como mostra COELHO (2005), KINTO (2005), LIMA (2002) e LORENA e CARVALHO (2007).

Para a classificação Multiclase foram testados os métodos Um Contra Um e Um Contra Todos em uma dada base de dados escolhida a partir de alguns requisitos pré-determinados, onde baseado nos resultados optou-se pelo segundo método, qual foi aplicado às demais bases de dados escolhidas.

Quanto à ferramenta YADMT, possui comunicação com os módulos por meio de *Interfaces*, facilitando o desenvolvimento das técnicas. A implementação mostrou-se fácil e rápida, confirmando o principal objetivo da proposta desta nova ferramenta, que é ser uma ferramenta livre, passível de modificação e evolução, proporcionando arcabouço suficiente para o acoplamento de novos módulos sem o considerável impacto do desenvolvimento de uma nova ferramenta.

Os resultados obtidos nos testes aplicados as 8 bases de dados foram considerados satisfatórios, pois 4 bases de dados obtiveram altas acuidades e as outras 4 obtiveram médias acuidades, mostrando ser uma técnica eficiente na classificação de dados.

4.1 Trabalhos Futuros

Observando as limitações deste trabalho em relação ao algoritmo SMO e analisando a ampla área de estudos relacionada à Classificação de Dados, podem-se citar trabalhos futuros a esta pesquisa como:

- Implementação de outros métodos de *Kernel* para transposição dos dados para um novo espaço de característica, a exemplo do *Kernel* Sigmoidal, Linear e Polinomial. No presente trabalho foi utilizado o *Kernel* Gaussiano, também chamado de *Radial Basis Function* (RBF). Com mais opções de *Kernels* a se escolher, é interessante também fazer estimativas de escolha do melhor para uma determinada base de dados.
- Testes para todas as bases de dados para o método Multiclasse Um Contra Um.
- Estudo de caso com SVM, a fim de obter o uso da técnica em um novo domínio de aplicação.
- Técnicas de Pré-Processamento, que abrangeriam todas as técnicas de Classificação da ferramenta YADMT.
- Representação gráfica da saída, dispondo visualmente os dados separados.

Apêndice A

Pseudocódigo do algoritmo SMO

Este apêndice tem como objetivo auxiliar no entendimento de como as formulações matemáticas foram mapeadas no código-fonte. A seguir é descrito alguns Pseudocódigos da implementação do SMO.

Kernel Gaussiano

Kernel Gaussiano, também chamado de *Radial Basis Function* (RBF), faz a transposição dos dados do do R^2 para um espaço de característica de maior dimensão:

$$e^{\frac{||x_i - x_j||^2}{2\sigma^2}} \quad \text{(A.1)}$$

O Algoritmo A.1 traz sua implementação:

Algoritmo A.1: Kernel Gaussiano

```
real funcao rbf_kernel(inteiro i, inteiro j) {
        real s = produto_escalar_pre_computado[i][j];
        s = -2 * s;
        s = s + produto_escalar_pre_computado_mesma_entrada[i] +
        produto_escalar_pre_computado_mesma_entrada[i];
        retorna exp(s / sigma_quadrado);
}
```

Descrição das variáveis:

- i: representa a entrada x_i;
- j: representa a entrada x_j;
- s: variável temporária de auxilio do cálculo;
- produto_escalar_pre_computado[i][j]: vetor contendo o produto escalar entre as

entradas i e j;

- produto_escalar_pre_computado_mesma_entrada[i]: vetor contendo o produto escalar entre a entrada i e i;

A função “exp” retorna a constante de Neper (*e*) elevada aos parâmetros da função.

O produto escalar é apresentado no Algoritmo A.2:

Algoritmo A.2: Produto Escalar.

```
funcao produto_escalar(i1, i2){
        p1 = 0; p2 = 0; escalar = 0;
        num1 = numero de atributos da entrada i1;
        num2 = numero de atributos da entrada i2;
        enquanto (P1 < num1 e p2 < num2){
                a1 = indice do elemento p1;
                a2 = índice do elemento p2;
                se(a1 == a2){
                        val1 = valor do elemento p1;
                        val2 = valor do elemento p2;
                        escalar += val1 * val2;
                        p1++;
                        p2++;
                }
                se(a1 > a2){
                        p2++;
                }senão{
                        p1++;
                }
        }
        retorna escalar;
}
```

O Algoritmo A.3 traz a função *TakeStep* onde são encontrados e atualizados os valores dos Multiplicadores de Lagrange e do vetor de peso ***w***.

Algoritmo A.3: TakeStep.

```
rotulo = valor da classe desejado;
funcao takeStep(i1,i2){
        se (i1 == i2)
                retorna 0;
        alph1 = Multiplicador de Lagrange para i1
        y1 = rotulo[i1]
        E1 = Saida_SVM(i1) - y1
        s = y1*y2;
        Calcula L, H
        if (L == H)
                return 0;
        k11 = kernel(i1, i1)
        k12 = kernel(i1, i2)
        k22 = kernel(i2, i2)
        eta = 2 * k12 - k11 - k22;
        se (eta < 0){
                a2 = alph2 - y2 * (E1 - E2) / eta;
                se (a2 < L)
                        a2 = L;
                senao
                se (a2 > H)
                        a2 = H;
        }
        senao{
                Lobj = funcao objetivo em a2 = L;
                Hobj = funcao objetivo em a2 = H;
        se (Lobj > Hobj + eps)
                a2 = L;
        senao
        se (Lobj < Hobj-eps)
                a2 = H;
        senao
                a2 = alph2;
        }
        se (|a2 - alph2| < eps * (a2 + alph2 + eps))
                return 0;
        a1 = alph1 + s * (alph2 - a2);
        Atualização dos Multiplicadores de Lagrange;
        Atualização do vetor w;
        Atualização da cachê de erro;
        retorna 1;
}
```

O Algoritmo A.4 traz a função *ExamineExample*.

Algoritmo A.4: ExamineExample.

```
funcao examineExample(i1)
        y1 = rotulo[i1];
        alph1 = Multiplicador de Lagrange para i1;
        E1 = Saida_SVM(i1) – y1;
        R1 = E1 * y1;
        if ((r1 < -tol e alph1 < C) || (r1 > tol e alph1 > 0)){
                loop de i = 0 até o número de amostras{
                        se (alph(i) > 0 e alph(i) < C)
                                i2 = resultado da segunda escolha heurística;
                }
                se (takeStep(i1,i2) == 1)
                        retorna 1;

                loop sobre todos os alph !=0 e alph != C, a partir de um ponto aleatório{
                        i2 = id do atual alph;
                        se (takeStep(i1,i2) == 1)
                                retorna 1;
                }
                loop sobre todos i2 possiveis, iniciando de um ponto aleatório{
                        i2 = variavel do loop;
                        se (takeStep(i1,i2) == 1)
                                retorna 1;

                }
        }
        retorna 0;
}
```

O Algoritmo A.5 traz a função principal do algoritmo SMO.

Algoritmo A.5: Função Principal.

```
funcao principal{
        initializa o vetor alpha (Multiplicadores de Lagrange) com zero;
        inicializa o limiar b com zero;
        numChanged = 0;
        examineAll = 1;
        enquanto (numChanged > 0 | examineAll){
                numChanged = 0;
                se (examineAll > 0)
                        loop de i=0 até todos os exemplos de treinamento{
                                numChanged += examineExample(i);
                        }
                senao
                loop de i=0 até todos os exemplos onde alph !=0 e alph != C{
                        numChanged += examineExample(i);
                }
                se (examineAll == 1)
                        examineAll = 0;
                senao
                se (numChanged == 0)
                        examineAll = 1;
        }
}
```

Apêndice B

Descrição das Bases de Dados utilizadas nos Testes

Este apêndice traz a descrição detalhada das bases de dados utilizadas para testes. Informações como criador, local, ano, o que os dados representam, quantidade de tuplas, número de atributos, entre outras informações.

Base de Dados *Abalone*

A base de dados *Abalone* foi criada por Sam Waugh do Departamento de Ciência da Computação da Universidade da Tasmânia, em 1995. Consiste de informações relacionadas a um molusco chamado holiote, que serve de alimento principalmente nos países asiáticos. Os dados foram inicialmente propostos no estudo NASH (1994), que analisava os moluscos a partir das características físicas e determinava qual seria a idade do mesmo para o consumo, e posteriormente foram retiradas as tuplas que continham valores ausentes. O conjunto consiste de:

- 1 de atributo categórico (sexo),
- 6 atributos numéricos (comprimento, diâmetro, altura, peso bruto, peso do molusco descascado, peso das vísceras, peso da concha),
- 1 atributo classificador (anéis), utilizado para determinar a idade do haliote.

A base de dados consiste de 4177 amostras e a distribuição dos dados nas classes segue na Tabela B.1.

Tabela B.1: Distribuição de classes da Base de Dados *Abalone*

Classe	Número de Exemplos
1	1
2	1
3	15
4	57
5	115
6	259
7	391
8	568
9	689
10	634
11	487
12	267
13	203
14	126
15	103
16	67
17	58
18	42
19	32
20	26
21	14
22	6
23	9
24	2
25	1
26	1
27	2
28	1

Base de dados *Acute Inflammations*

A base *Acute Inflammations* foi criada por Jacek Czerniak, do Instituto de Pesquisa de Sistemas da Acadêmia Polonesa de Ciências, e está disponível na UCI desde 2009. Apresenta dados criados para testar um sistema especialista, o qual deveria realizar o diagnóstico presumível de doenças do sistema urinário. Originalmente esta base possuía 8 atributos sendo:

- 1 atributo numérico, indicando a temperatura do paciente,
- 7 categóricos (ocorrência de náuseas, dor lombar, necessidade constante de urinar, dores de micção, dores de uretra, ocorrência de inflamação urinária na bexiga e nefrite de origem pelve renal), que podem assumir valores "yes" e "no", destes, os dois últimos atributos representam atributos de decisão de classe baseado nos seis primeiros atributos.

Para este trabalho eliminou-se o último atributo classe (nefrite de origem pelve renal), ficando a base com um total de 7 atributos, incluso o atributo que define a classe "inflamação urinária na bexiga".

Esta base de dados contém um total de 120 amostras com a distribuição das classes de acordo com a Tabela B.2.

Tabela B.2: Distribuição de classes da Base de Dados *Acute Inflammations*

Classe	N° de Exemplos
yes	59
no	61

Base de Dados *Ecoli*

A base de dados *Ecoli* foi criada por Kenta Nakai ciêntista do Instituto de Biologia Celular e Molecular da Universidade de Osaka em 1996. Contêm dados sobre a localização de proteínas em células. Originalmente consistia de:

- 1 atributo categórico (*Sequence Name*, em inglês), o qual foi retirado pois serve apenas para identificação do registro dentro do banco de dados;
- 7 atributos numéricos de medições de características de células;
- Um último atributo que define a classe do objeto, neste caso a localização da célula que tem a maior concentração de proteína.

Esta base contém 336 amostras e as classes estão distribuídas de acordo com a Tabela B.3.

Tabela B.3: Distribuição de classes da Base de Dados *Ecoli*

Classe	N° de Exemplos
cp (citoplasma)	143
im (membrana interna sem sinal de sequência)	77
pp (periplasma)	52
imU (membrana interna, com sinal de sequência não clivável)	35
om (membrana externa)	20
omL (membrana externa de lipoproteína)	5
imL (membrana interna de lipoproteína)	2
imS (membrana interna, com sinal de sequência clivável)	2

Base de Dados *Iris*

A base de dados *Iris*, bastante conhecida, foi criada por Ronald Aylmer Fisher em 1988. Contém dados de três tipos de flores conhecidas por Iris: Setosa, Vesicolour e Virginica. Além do atributo de classificação possui outros 4 atributos que trazem informações de tamanho e espessura da sépala e tamanho e espessura de pétala, com todas as dimensões dadas em centímetros.

Sabe-se que classe Setosa é linearmente independente das outras, porém estas não o são entre si. A base de dados é composta por 150 amostras e a distribuição igualitária dos dados é dada pela Tabela B.4.

Tabela B.4: Distribuição de classes da Base de Dados *Iris*

Classes	N° de Exemplos
Iris Setosa	50
Iris Versicolour	50
Iris Virginica	50

Base de Dados LIBRAS *Movement*

A base de dados LIBRAS *Movement* foi criada por Daniel Baptista Dias, Sarajane Marques Peres e Helton Hideraldo Bíscaro, da Universidade de São Paulo - USP. Esta base de dados

contém informações de movimentos da mão que representam o tipo de sinal LIBRAS (Língua Brasileira de Sinais). É composta de 90 atributos numéricos que representam a posição da mão em cada instante de tempo em um vídeo de 45 frames, além de um atributo categórico que define a classe do objeto, neste caso um dos 15 tipos de movimento LIBRAS.

Esta base de dados é composta por 360 amostras com a igualitária distribuição das classes de acordo com a Tabela B.5.

Tabela B.5: Distribuição de classes da Base de Dados LIBRAS *Movement*

Classe	N° Exemplos
curved swing	24
horizontal swing	24
vertical swing	24
anti-clockwise arc	24
clockwise arc	24
circle	24
horizontal straight-line	24
vertical straight-line	24
horizontal zigzag	24
vertical zigzag	24
horizontal wavy	24
vertical wavy	24
face-up curve	24
face-down curve	24
tremble	24

Base de Dados *Teaching Assistant Evaluation* (TAE)

A base de dados *Teaching Assistant Evaluation* foi criada por Wei-Yin Loh do Departamento de Estatística da Universidade de Wisconsin - Madison em 1997. Consiste de dados da avaliação de desempenho de assistentes de ensino (TA - *Teaching Assistent*) designados no Departamento de Estatística da Universidade de Wisconsin-Madison. É composta por 6 atributos numéricos, o primeiro atributo indica se o TA tem ou não o idioma Inglês como seu

idioma nativo, o segundo atributo indica qual o instrutor do curso, o terceiro atributo indica o curso, o quarto atributo indica se é um semestre regular ou um semestre de verão, o quinto atributo indica o tamanho da turma e, por fim, o sexto atributo indica a pontuação do assistente como pequena, média ou grande.

Esta base de dados contém 151 amostras e as classes estão distribuídas nas três classes de acordo com a Tabela B.6.

Tabela B.6 - Distribuição de classes da Base de Dados *Teaching Assistant Evaluation*

Classe	**N° de Exemplos**
1 (pequena)	49
2 (média)	50
3 (grande)	52

Base de Dados *Statlog* (*Image Segmentation*)

A base de dados Statlog (*Image Segmentation*) foi criada pelo Vision Group da Universidade de Massachusetts em 1990. Contém 19 atributos numéricos contínuos (Tabela B.7) que representam várias informações sobre segmentos de imagem de tamanho 3x3 pixels, mais um atributo numérico que define a classe do objeto, cuja informação representa um tipo de imagem (tijolo, céu, folhagem, cimento, janela, caminho e grama).

Tabela B.7 - Atributos da Base de Dados Statlog (*Image Segmentation*)

1	***region-centroid-col**:* coluna da região central do pixel.
2	***region-centroid-row**:* linha da região central do pixel.
3	***region-pixel-count**:* número de pixels na região (sempre 9).
4	***short-line-density-5**:* resultado de um algoritmo extrator de linhas, que conta quantas linhas de comprimento 5 (em qualquer orientação) com baixo contraste, inferior ou igual a 5, percorrendo a região.
5	***short-line-density-2**:* mesmo que short-line-density-5, mas conta linhas de alto contraste, superiores ou iguais a 5.
6	***vedge-mean***: medida média de contraste de pixels adjacentes horizontais.

	Este atributo é usado como um detector vertical de arestas.
7	***vegde-sd***: desvio padrão do atributo 6
8	***hedge-mean***: medida média de contraste de pixels adjacentes verticais. Usado como detector de linhas horizontais.
9	***hedge-sd***: desvio padrão do atributo 8
10	***intensity-mean****:* média de intensidade sobre a região *(R + G + B)/3*
11	***rawred-mean****:* média de intesidade da região no canal R.
12	***rawblue-mean****:* média de intesidade da região no canal B.
13	***rawgreen-mean****:* média de intesidade da região no canal G.
14	***exred-mean****:* medida de excesso de vermelho (2R - (G + B))
15	***exblue-mean****:* medida de excesso de azul (2B - (G + R))
16	***exgreen-mean****:* medida de excesso de verde (2G - (R + B))
17	***value-mean****:* valor da transformação 3-d não-linear.
18	***saturatoin-mean****:* saturação da transformação 3-d não-linear.
19	***hue-mean****:* matiz da transformação 3-d não-linear.

Esta base de dados contém 2310 amostras distribuídas de acordo com a Tabela B.8.

Tabela B.8 - Distribuição de classes da Base de Dados Statlog (*Image Segmentation*)

Classe	**N° de Exemplos**
1	330
2	330
3	330
4	330
5	330
6	330
7	330

Base de Dados *Wine Quality*

A base de dados *Wine Quality* foi criada por Paulo Cortez, Antonio Cerdeira, Fernando Almeida, Telmo Matos e Jose Reis da Universidade de MInho, Guimarães - Portugal em 2009. Esta base de dados contém informações físico-químicas da variação do vinho Português de cor vermelha. É composta por 11 atributos numéricos contínuos que representam as caracteristicas físico-químicas do vinho (*fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol*) e um atributo que quantitativo que representa a qualidade do vinho.

Esta base apresenta 4898 amostras cuja distribuição das classes está definida na Tabela B.9.

Tabela B.9 - Distribuição de classes da Base de Dados *Wine Quality*

Classe	N° de Exemplos
3	20
4	163
5	1457
6	2198
7	880
8	175
9	5

Referências Bibliográficas

ALES, T. V; GEVERT, G.V; CARNIERI, C; SILVA, L. C. A. **Análise de Crédito Bancário Utilizando O Algoritimo Sequential Minimal Optimization.** XLI Simpósio Brasileiro de Pesquisa Operacional 2009 – Pesquisa Operacional na Gestão do Conhecimento. p. 2242-2253. 2009.

AURÉLIO, M.; VELLASCO, M.; LOPES H. C. **Descoberta de Conhecimento e Mineração de Dados.** Apostila. ICA – Inteligência Computacional Aplicada, DEE, PUC – Rio (Versão 26/08/1999).

BENFATTI, E. W. **Um estudo sobre a aplicação de KNN, c4.5 e Naive Bayes na Classificação de dados.** Universidade Estadual do Oeste do Paraná – Centro de Ciências Exatas e Tecnológicas - CCET. Colegiado de Ciência da Computação. Curso de Ciência da Computação. Novembro, 2010. Monografia de Graduação.

BENFATTI, E. W.; BOSCARIOLI, C.; GIRARDELLO A. D.; BONIFÁCIO F. N. **YADMT - Yet Another Miner Tool.** Universidade Estadual do Estado do Paraná - UNIOESTE, Centro de Ciências Exatas e Tecnológicas – CCET, Curso de Ciência da Computação, Novembro, 2010. Relatório Técnico número 1.

BISOGNIN, G. **Utilização de Máquinas de Vetores de Suporte Para Predição de Estruturas Terciárias de Proteínas.** São Leopoldo, Universidade do Vale do Rio dos Sinos, Ciências Exatas e Tecnológicas, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, 2007. Dissertação de Mestrado.

Buscapé, Disponível em: http://www.buscape.com.br, Acessado em: 03/11/2010.

BONIFACIO, F. N. **Comparação entre as Redes Neurais Artificiais MLP, RBF e LVQ na Classificação de Dados.** Universidade Estadual do Oeste do Paraná – Centro de Ciências

Exatas e Tecnológicas - CCET, Curso de Ciência da Computação, Novembro, 2010. Monografia de Graduação.

COELHO, S. T. **Reconhecimento de Fala usando Máquinas de Vetor de Suporte (SVMs).** Inatel – Instituto Nacional de Telecomunicações, Novembro, 2005. Dissertação de Mestrado.

CRUZ, A. J. R. **Data Mining via Redes Neuronais Artificiais e Máquinas de Vectores de Suporte.** Universidade do Minho. Escola de Engenharia. Departamento de Sistemas de Informação, 2007. Dissertação de Mestrado.

DOMINGUES, M. A.; REZENDE, S. O. **Pós-Processamento de Regras de Associação usando Taxonomias.** Artigo. Journal of Computer Science. v. 4, p. 26-37. 2005.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; UTHURUSAMY, R. **Advances in Knowledge Discovery and Data Mining**. AAAI/MIT Press, 1996.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. **The kdd process for extracting useful knowledge from volumes of data.** Communications of the ACM, New York, NY, USA, v.39, n.11, p.27–34, 1996.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. **Knowledge Discovery and Data Mining Towards a Unifying Framework.** Artigo. KDD-96 Conference Proceeding, ed. E. 1996.

GEVERT, V. G. **Análise de Crédito Bancário com o Uso de Modelos de Regressão Logística, Redes Neurais e *Support Vector Machine*.** Programa de Pós Graduação em Métodos Numéricos em Engenharia, na Área de Concentração em Programação Matemática, Setor de Tecnologia, Departamento de Construção Civil e Setor de Ciências Exatas, Departamento de Matemática da Universidade Federal do Paraná. Curitiba, 2009. Dissertação de Mestrado.

GUYON I. **SVM Application List. Disponível em:** http://www.clopinet.com/isabelle/Projects/SVM/applist.html, Consultado na Internet em:

22/07/2010.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques.** 1. ed., San Francisco California, USA, Morgan Kaufmann Publishers, 2001.

HAYKIN, S. **Redes Neurais Princípios e Prática.** 2º Edição. Editora Bookman, 2001.

IBM Inteligent Miner. Disponível em: http://www-947.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_Intelligent_Miner_Modeling,Consultado na Internet em: 30/03/2010.

Java Sun Microsystem. Disponível em: http://br.sun.com/, Consultado na Internet em: 30/03/2010.

KINTO, E. **Máquinas de Vetores-Suporte Aplicadas à Classificação de Textos Reduzidos.** Escola Politécnica da Universidade de São Paulo. São Paulo, 2005. Dissertação de Mestrado.

LEITHOLD, L. **O Cálculo com Geometria Analítica – dois.** Editora HARBRA ltda. 3º edição. 1994.

LIMA, R. G. **Máquinas de Vetores de Suporte na Classificação de Impressões Digitais.** Universidade Federal do Ceará, Departamento de Computação, Fortaleza - Ceará. 2002. Dissertação de Mestrado.

LORENA, A. C.; CARVALHO, A. C. P. L. F. **Uma introdução às Support Vector Machines (*In Portuguese*)**. Revista de Informática Teórica e Aplicada, v. 14, p. 43-, 2007.

MATLAB. Disponível em: http://www.mathworks.com/products/matlab, Acessado em 03/11/2010.

MYSQL. Disponível em: http://www.mysql.com/, Acessado na Internet em: 16/08/2010.

NASH, W. J., SELLERS, T. L., TALBOT, S. R., CAWTHORN, A. J., FORD, W. B., ***The Population Biology of Abalone (_Haliotis_species) in Tasmania***. *I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait*, Sea Fisheries Division, Technical Report No. 48, 1994.

NEVEZ, R. C. D. **Pré-Processamento no Processo de Descoberta de Conhecimento em Banco de Dados.** Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Computação, 2005. Dissertação de Mestrado.

OLIVEIRA, A. G.; GARCIA, D. F. **Mineração da Base de Dados de um Processo Seletivo Universitário.** UNIFOR/MG – Centro Universitário de Formiga. Artigo. INFOCOMP Revista de Ciência da Computação. 2004. v. 3. nº 2. p. 38-43.

Oracle Data Mining. Disponível em: http://www.oracle.com/technology/documentation/darwin.html, Consultado na Internet em: 30/03/2010.

Orange. Disponível em: http://www.ailab.si/orange/, Consultado na Internet em: 30/03/2010.

PERL, Disponível em: http://www.perl.org/get.html, Consultado na Internet em: 03/11/2010.

PLATT, J. C. **Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machine.** Technical Report MSR-TR, Microsoft Research, 1998.

PLATT, J. C. **Fast Training os Support Vector Machines using Sequential Minimal Optimization.** Microsoft Research, 2000.

PLATT, J. C.; CRISTIANINI, N.; SHAWE-TAYLOR, J; **Large Margin DAGs for Multiclass Classification.** S.A Solla, T.K. Leen and K.-R. Muller (eds.), 547 – 553, MIT Press. 2000.

SAS Enterprise Miner. Disponível em: http://www.sas.com/technologies/analytics/datamining/miner/, Consultado na Internet em:

30/03/2010.

SEMOLINI, R. **Support Vector Machines, Inferência Transdutiva e o Problema de Classificação.** Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Departamento de Engenharia de Computação e Automação Industrial, 2002. Dissertação de Mestrado.

SERASA. Disponível em: http://www.serasaexperian.com.br, Consultado na Internet em: 03/11/2010.

SILVA, E. B.; XEXÉO, G. B. Construction data mining funcionalities in dbms. In: INTERNATIONAL CONFERENCE ON DATA MINING, 1998. Proceedings... COPPE - Universidade Federal do Rio de Janeiro, RJ, BR: WIT PRESS, 1998.

SILVA, V. D.; SCARPEL, R. A. **Detecção de Fraudes na distribuição de energia elétrica utilizando support vector machine.** 2007, Artigo. Investigação Operacional, Vol. 27, p. 139 – 150.

SOUSA, M. S. R. **Mineração de Dados: Uma implementação Fortemente Acoplada a um Sistema Gerenciador de Banco de Dados Paralelo.** Universidade Federal do Rio de Janeiro, Programa: Engenharia de Sistemas e Computação, 1998. Dissertação de Mestrado.

SPC – Serviço de Proteção ao Crédito. Disponível em: http://www.spcbrasil.org.br, Consultado na Internet em: 03/11/2010.

Tanagra. Disponível em: http://eric.univ-lyon2.fr/~ricco/tanagra/, Consultado na Internet em: 30/03/2010.

UCI Machine Learning Repository. Disponível em: http://archive.ics.uci.edu/ml/index.html, Consultado na Internet em: 30/03/2010.

VAPNIK, V. **Estimation of Dependences Based on Empirical Data.** 2º Edição. Editora Springer. 2006.

VAPNIK, V. N. **The Nature of Statistical Learning Theory.** 2º Edição. Editora Springer. 2000.

VASCONCELOS, B. S. **Mineração de Regras de Classificação com Sistemas de Banco de Dados Objeto-Relacional.** Universidade Federal de Campina Grande, Centro de Ciência e Tecnologia, Coordenação de Pós-Graduação em Informática, Dezembro, 2002. Dissertação de Mestrado.

VISUAL BASIC. Disponível em: http://www.microsoft.com/visualstudio/pt-br/visual-studio-2010-launch, Acessado em: 03/11/2010.

VOLTOLINI, R. **Projeto de um ambiente para Classificação de Dados com Árvore de Decisão e Rede Neural Artificial Multicamada.** Universidade Estadual do Oeste do Paraná – Centro de Ciências Exatas e Tecnológicas. Colegiado de Informática. Curso de Bacharelado em Informática. 2008. Monografia de Graduação.

ZHENG, Z. **A benchmark for classifier learning** (Technical Report TR474). Basser Department of Computer Science, N.S.W Australia, 2006.

Weka. Disponível em: http://www.cs.waikato.ac.nz/~ml/weka/, Consultado na Internet em: 30/03/2010.