

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Davi Alves Leão

**Novos métodos de poda iterativos de máquinas de vetores-suporte de
mínimos quadrados**

Fortaleza – CE

27 de fevereiro de 2025

Davi Alves Leão

Novos métodos de poda iterativos de máquinas de vetores-suporte de mínimos quadrados

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Orientador: Prof. Dr. Ajalmar Rêgo da Rocha Neto

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

27 de fevereiro de 2025

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI

Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

L433n Leão, Davi Alves.
Novos métodos de poda iterativos de máquinas de vetores-suporte de mínimos quadrados / Davi Alves
Leão. - 2024.
70 f. : il.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus
Fortaleza, 2024.

Orientação: Prof. Dr. Ajalmar Rêgo da Rocha Neto.

1. LSSVM. 2. Levenberg Marquardt. 3. Poda. 4. Esparsidade. 5. Otimização. I. Título.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Davi Alves Leão

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

**Orientador: Prof. Dr. Ajalmar Rêgo da Rocha
Neto**

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Prof. Dr. Auzuir Ripardo de Alexandria

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Profa. Dra. Ana Luiza Bessa de Paula Barros

Universidade Estadual do Ceará (UECE)

Fortaleza – CE
27 de fevereiro de 2025

*Aos meus pais Valter e Evanea, pelo apoio constante e ensinamentos, a minha irmã, e à
minha esposa Letícia, pela paciência e incentivo ao longo desta jornada.*

Agradecimentos

A Deus, por me conceder força, sabedoria e coragem para superar os desafios ao longo desta jornada.

Aos meus pais, pelo apoio incondicional e pelos valores que me transmitiram, fundamentais para a realização desta etapa.

À minha esposa, pela paciência, compreensão e incentivo, sendo minha força e inspiração ao longo do percurso.

Aos meus colegas e amigos, que contribuíram com apoio e trocas de ideias valiosas ao longo dessa caminhada.

Ao meu orientador, pela orientação e dedicação que foram essenciais para o desenvolvimento deste trabalho.

Resumo

As máquinas de vetores-suporte de mínimos quadrados (*Least Square Support Vector Machines* - LSSVM) são resolvidas a partir de um sistema linear, ao contrário das máquinas vetores-suporte (*Support Vector Machines* - SVM), que é necessário a aplicação de programação quadrática. Com isso, é preferível, em alguns problemas, a utilização do LSSVM, uma vez que a complexidade de resolução será menor. Entretanto, ao contrário do SVM, o LSSVM gera soluções não esparsas, e essa é uma desvantagem significativa, dado que todos os padrões de treinamento serão vetores-suporte do LSSVM. Para superar tais soluções não esparsas, propõe-se neste trabalho soluções iterativas com poda, baseadas no método de Levenberg-Marquardt, sem causar perda de eficiência. Os métodos propostos, são comparados com outros da literatura. Por fim, a avaliação da solução proposta consistiu na comparação das médias das taxas de acerto.

Palavras-chave: LSSVM, Levenberg–Marquardt, Poda, Esparsidade, Otimização.

Abstract

Least-Square Support Vector Machines (LSSVM) are solved from a linear system, unlike support vector machines (SVM), which require the application of quadratic programming. So it is preferable in some problems to use LSSVM because the complexity of the resolution will be lower. However, unlike SVM, LSSVM generates non-sparse solutions. A significant disadvantage, given that all training patterns will be support vectors of LSSVM. To overcome such non-sparse solutions, this work proposes iterative solutions with pruning, based on the Levenberg-Marquardt method, without causing loss of efficiency. The proposed methods are compared with others in the literature. Finally, the evaluation of the proposed solution consisted of comparing the average of the hit rates.

Keywords: LSSVM, Levenberg–Marquardt, Pruning, Sparseness, Optimization.

Lista de ilustrações

Figura 1 – (a) Exemplo Artificial com 3 classes. (b) Exemplo Artificial com 2 classes	29
Figura 2 – Exemplo codificação 1-de-C	29
Figura 3 – Problema de classificação, linearmente separável, com duas classes, classe 1 (azul) e classe -1 (vermelha).	30
Figura 4 – Representação da margem das máquinas de vetores-suporte	31
Figura 5 – Janela de poda (Proposta 1)	49
Figura 6 – Remoção dos vetores-suporte(Proposta 2)	52
Figura 7 – Gráfico tamanho matriz $\mathbf{A}$ pelo número de padrões de treino.	53
Figura 8 – Exemplo com $\tau = 5$ (Proposta 2).	54
Figura 9 – Vetores-suporte no processo iterativo do RFSLM-LSSVM.	57
Figura 10 – Fluxo das simulações computacionais.	60
Figura 11 – Acurácia <i>versus</i> tamanho da poda para as patologias da coluna vertebral. (a) Metodologia 80/20. (b) Metodologia 60/20/20	62
Figura 12 – Acurácia <i>versus</i> tamanho da poda para o conjunto do câncer de mama. (a) Metodologia 80/20. (b) Metodologia 60/20/20	64
Figura 13 – Acurácia <i>versus</i> tamanho da poda para o conjunto das diabetes. (a) Metodologia 80/20. (b) Metodologia 60/20/20	66
Figura 14 – Acurácia <i>versus</i> tamanho da poda para o conjunto de dados de Haberman e Ripley. (a) Haberman com metodologia 80/20. (b) Haberman com metodologia 60/20/20. (c) Ripley com metodologia 80/20. (d) Ripley com metodologia 60/20/20.	68
Figura 15 – Visualização 2D dos vetores-suporte em quatro bases de dados diferentes aplicando o modelo FSLM-LSSVM(Proposta 1) com 80 % de poda, aplicando Kernel RBF. (a) Conjunto de dados Ripley. (b) Conjunto de dados artificial TWC. (c) Conjunto Artificial TWM. (d) Conjunto Artificial TWS.	69
Figura 16 – Visualização 2D dos vetores-suporte em quatro bases de dados diferentes aplicando o modelo RFSLM-LSSVM(Proposta 2) com 80 % de poda, aplicando Kernel RBF. (a) Conjunto de dados Ripley. (b) Conjunto de dados artificial TWC. (c) Conjunto Artificial TWM. (d) Conjunto Artificial TWS.	70

Figura 17 – Analise de todas as bases de dados de acordo com a porcentagem de corte. (a) Acurácia x porcentagem de corte (Direita) e Diagrama de caixa de cada base (Esquerda), aplicando modelo FSLM-LSSVM com kernel Linear. (b) Acurácia x porcentagem de corte (Direita) e Diagrama de caixa de cada base (Esquerda), aplicando modelo RFSLM-LSSVM com kernel Linear. 72

Lista de tabelas

Tabela 1	–	Lista de bases de dados usadas nesse trabalho.	59
Tabela 2	–	Funções de Kernel.	60
Tabela 3	–	Lista dos hiperparâmetros aplicados nas simulações computacionais. . .	61
Tabela 4	–	Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema da coluna vertebral. . .	61
Tabela 5	–	Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema da coluna vertebral.	62
Tabela 6	–	Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema do câncer de mama. . .	63
Tabela 7	–	Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema do câncer de mama.	63
Tabela 8	–	Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema da diabete.	65
Tabela 9	–	Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema da diabete.	65
Tabela 10	–	Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema de haberman.	66
Tabela 11	–	Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema de haberman. . . .	67
Tabela 12	–	Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema ripley.	67
Tabela 13	–	Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema ripley.	68

Lista de abreviaturas e siglas

SVN	<i>Support Vector Networks</i> ou redes de vetores-suporte
SVM	<i>Support Vector Machines</i> ou Máquinas de vetores-suporte
SVC	<i>Support Vector Classifier</i>
SVR	<i>Support Vector Regression</i>
LSSVM	<i>Least Squares Support Vector Machines</i> ou Máquinas de vetores-suporte de mínimos quadrados.
LM	Levenberg-Marquardt
FSGAS-LSSVM	<i>Fixed-Size Genetic Algorithm for Sparse LSSVM</i>
FSLM-LSSVM	<i>Fixed size Levenberg-Marquardt LSSVM</i>
RFSLM-LSSVM	<i>Reduced Fixed set Levenberg-Marquardt LSSVM</i>
P-LSSVM	<i>prunning LSSVM</i>
IP-LSSVM	<i>IP LSSVM</i>

Lista de símbolos

$\mathbf{X}$	Matriz que contem os padrões
$\mathbf{x}$	Vetor de atributos
$\hat{y}_i$	i-ésima saída do classificador
$\mathbf{x}_i$	i-ésimo padrão de D
d_i	rótulo associado ao i-ésimo padrão.
$\mathbf{x}^{(s)}$	Vetor-suporte
R_i	i-ésima região dos dados
C_j	j-ésima classe
C_{+1}	Classe positiva
C_{-1}	Classe negativa
R_{+1}	Região dos dados da classe positiva
R_{-1}	Região dos dados da classe negativa
E_{in}	Erro interno, associado aos padrões de treino
E_{out}	Erro associado aos padrões de teste
$\mathcal{D}$	Conjunto de padrões ou amostras.
$\mathcal{D}_{in}$	Padrões ou amostras de treino
$\mathcal{D}_{out}$	Padrões ou amostras de teste
$\mathcal{D}_{vs}^{\mathcal{F}\%}$	Conjunto reduzido de vetores-suporte para treinamento do RFSLM-LSSVM
$\mathcal{D}_{in}^{1-\mathcal{F}\%}$	Conjunto dos dados restante, para o treinamento do RFSLM-LSSVM
$(\mathbf{x}_i, d_i)$	i-ésima instância do conjunto de dados
$\mathbf{w}$	Vetor de pesos.
b	Viés.
$\mathbf{w}_*$	Vetor de pesos ótimo.

b_*	Viés ótimo.
ρ	Margem de separação
$\mathcal{T}$	Limiar para SVM
$r^{(s)}$	Distância do vetor-suporte ao hiperplano ótimo
$\mathbf{x}_i^{(s)}$	i-ésimo vetor-suporte
α	Vetor de multiplicadores de Lagrange
α_i	i-ésimo multiplicador de Lagrange
A	Matriz padrão para resolução do sistema Linear do LSSVM
$\mathcal{A}$	Matriz reduzida para resolução do sistema Linear para o RFSLM-LSSVM.
$\mathbf{z}_i$	i-ésimo vetor solução.
$\mathbf{z}_i^{\mathcal{F}\%}$	i-ésimo vetor solução, com uma porcentagem fixa de vetores-suporte.
$\mathbf{z}^*$	vetor solução ótimo.
sign(.)	Função sinal
max	Função Máximo
min	Função Mínimo
mse(.)	Função do erro quadrático médio
k(.)	Função de kernel
$O(.)$	Função de Complexidade
γ	Parâmetro de regularização do classificador LSSVM
ξ_i	i-ésima variável de folga
μ	coeficiente de aprendizado de Levenberg-Marquardt
J	Matriz jacobiana
τ	número de iterações para o corte de vetores-suporte no RFSLM-LSSVM.
$\mathcal{P}_{\%}$	Porcentagem de troca dos vetores-suporte, a cada τ iterações, para o algoritmo RFSLM-LSSVM.
κ	Faixa de remoção para FSLM-LSSVM

$\mathcal{F}_{\%}$	Porcentagem fixa do tamanho do conjunto de vetores-suporte para RFSLM-LSSVM.
$\mathcal{R}_{\%}$	Porcentagem de poda ou redução de vetores-suporte
N_{atr}	Número de atributos de um padrão
N_{vs}	Número de vetores-suporte
$N_{vs}^{\mathcal{F}_{\%}}$	Número fixo de vetores-suporte para computar a solução do modelo RFSLM-LSSVM
$N_{vs}^{\mathcal{P}_{\%}}$	Número de vetores-suporte a serem trocados no processo do RFSLM-LSSVM.
N_{in}	Número de padrões no conjunto de treinamento
N_t	Número total de iterações

Sumário

1	INTRODUÇÃO	25
1.1	Motivação para a Dissertação	25
1.2	Objetivos Geral e Específicos	26
1.3	Produção científica	26
1.4	Estrutura da Dissertação	26
2	FUNDAMENTAÇÃO TEÓRICA	27
2.1	Introdução	27
2.2	O que é um Padrão	27
2.3	Aprendizado	27
2.3.1	Aprendizado supervisionado	28
2.3.1.1	Classificação de padrões	28
2.4	Máquinas de Vetores-suporte	29
2.4.1	Definições e conceitos	30
2.4.2	Princípios do classificador SVM	33
2.4.2.1	Máquinas de vetores-suporte de margem rígida	33
2.4.2.2	Máquinas de vetores-suporte de margem flexível	35
2.5	Máquinas de Vetores-suporte de mínimos quadrados	38
2.5.1	Definições e conceitos	38
2.5.2	Obtenção dos Parâmetros Ótimos para o classificador LSSVM	39
2.5.2.1	Solução baseada na Matriz inversa	39
2.5.2.2	Solução baseada na Pseudo Inversa	40
2.5.2.3	Solução baseada no método de Levenberg-Marquardt	40
2.6	Conclusão	42
3	TÉCNICAS PARA A OBTENÇÃO DE MODELOS REDUZIDOS EM LSSVM	45
3.1	Introdução	45
3.2	P-LSSVM	45
3.3	IP-LSSVM	46
3.4	Fixed-Size Genetic Algorithm for Sparse LSSVM	47
3.5	Modelos propostos	48
3.5.1	Definições e conceitos	48
3.5.2	Fixed Size Levenberg Marquardt LSSVM (Proposta 1)	49
3.5.2.1	Introdução	49
3.5.2.2	Janela de poda	49

3.5.2.3	Solução Esparsa	49
3.5.2.4	Algoritmo	50
3.5.3	Reduced Fixed set Levenberg Marquardt LSSVM (Proposta 2)	51
3.5.3.1	Introdução	51
3.5.3.2	Reduced fixed sets	51
3.5.3.3	Análise Matricial	52
3.5.3.4	Algoritmo	53
3.5.3.5	Exemplos	55
3.6	Conclusão	56
4	SIMULAÇÕES COMPUTACIONAIS	59
4.1	Metodologia	59
4.2	Análise dos dados com bases reais	61
4.3	Análise dos vetores-suporte	69
4.4	Análise geral de desempenho dos modelos propostos	71
5	CONCLUSÕES	73
	REFERÊNCIAS	75

1 Introdução

1.1 Motivação para a Dissertação

Existem inúmeros algoritmos de aprendizagem de máquina na literatura, alguns resolvem, apenas, problemas de classificação, outros, apenas, problemas relacionados à aproximação de funções e existem aqueles que resolvem ambos. Um desses algoritmos que possui a capacidade de resolver ambos os tipos de problema, são as máquinas de vetores-suporte (*Support Vector Machines - SVM*) (CORTES; VAPNIK, 1995), algoritmo com uma sólida base formal e capacidade de generalização sobre uma gama de problemas reais. Contudo, como será explanado mais a frente, as máquinas de vetores-suporte é um modelo de aprendizagem que necessita de um tempo elevado para treinar o modelo, tornando-se inviável dependendo da aplicação.

Uma alternativa ao uso das SVM's, que necessitam da resolução de um problema de programação quadrática, é utilizar as máquinas de vetores-suporte de mínimos quadrados (*Least Squares Support Vector Machines - LSSVM*) (SUYKENS; VANDEWALLE, 1999), um algoritmo capaz de resolver problemas de classificação e regressão. Contudo, diferentemente do seu antecessor, tem um tempo de treinamento menor, agilizando o processo de aprendizagem em problemas com uma grande quantidade de dados, como problemas de *big data*. Entretanto, ao deixar o treinamento mais eficiente acaba tornando o processo de classificação final mais lento, dependendo do tamanho da base de dados, uma vez que ao final do treino todos exemplos de entrada tornam-se vetores-suporte, perdendo, assim, a característica esparsa do seu antecessor.

A fim de construir soluções esparsas com o LSSVM, foram propostos vários métodos para diminuição de complexidade, podendo citar os métodos de poda. Os métodos de poda ou corte, são baseados na retirada de informações que impactam pouco ou quase nada nas decisões finais dos modelos, esses compreendem um conjunto de técnicas que visam simplificar a estrutura interna dos modelos, sem que esses percam a sua eficiência. Uma ideia mais detalhada sobre o assunto pode ser estudada a partir do trabalho (BURGES, 1996). Para superar a desvantagem do LSSVM, outros métodos foram desenvolvidos a fim de aumentar a esparsidade no seu projeto (SUYKENS; LUKAS; VANDEWALLE, 2000; CARVALHO; BRAGA, 2009; BURBIDGE; BUXTON, 2001).

O LSSVM gera soluções não esparsas, e por isso precisa armazenar todos os vetores-suporte, amostras de treino, para que consiga realizar a etapa de predição. Esta característica, é bastante prejudicial em problemas com grandes quantidades de dados, que necessitem ao mesmo tempo de um treinamento em tempo linear, gastem pouca memória

para armazenar o modelo final e tenham que ter uma predição rápida. Os algoritmos da literatura, resolvem o problema da esparsidade, como será visto no capítulo 3. Este trabalho busca resolver o problema da perda de esparsidade do LSSVM de forma iterativa.

1.2 Objetivos Geral e Específicos

Em seu sentido mais amplo, o principal objetivo dessa dissertação é construir soluções esparsas com o LSSVM.

Em seu sentido mais particular, esse trabalho tem os seguintes objetivos específicos:

- Modelar um algoritmo de classificação, que remova vetores-suporte iterativamente.
- Diminuir a complexidade dos cálculos matriciais, realizados iterativamente, em busca do ponto ótimo.
- Construir soluções esparsas, com um número fixo de vetores-suporte.

1.3 Produção científica

Durante este projeto de mestrado, os seguintes trabalhos científicos foram aceitos e publicados, a saber:

- **Davi A. Leão** Ajalmar R. R. Neto (2022), "Um novo método de poda iterativo de máquinas de vetores-suporte de mínimos quadrados", submetido e aceito no XXIV Congresso Brasileiro de Automática(CBA'2022), Fortaleza-CE.

1.4 Estrutura da Dissertação

Este trabalho é formado por quatro capítulos no total. Organizados da seguinte maneira: Introdução, Fundamentação Teórica, Técnicas para a obtenção de modelos reduzidos em LSSVM, Simulações computacionais e Conclusões.

O capítulo 2, apresenta uma abordagem introdutória e geral sobre aprendizagem de máquina, detalhando os principais conceitos a respeito das máquinas de vetores-suporte e das máquinas de vetores-suporte de mínimos quadrados. Seguindo, o capítulo 3 apresenta três algoritmos da literatura são eles: P-LSSVM, IP-LSSVM e FSGAS-LSSVM. Além disso, são detalhadas duas propostas para solucionar o problema da perda de esparsidade do LSSVM. O próximo, capítulo 4, exhibe os resultados das simulações computacionais seguidas de algumas análises. Por fim, no último capítulo são feitas considerações finais a respeito do trabalho.

2 Fundamentação Teórica

2.1 Introdução

Na era da tecnologia, diariamente, geramos grandes quantidades de dados que são analisados para fornecer informações valiosas. Esses dados podem ser aplicados para solucionar problemas específicos, como, por exemplo, o reconhecimento de caracteres, dígitos ou letras. Neste processo, um sistema computacional coleta informações sobre o problema e aplica algoritmos específicos para encontrar a solução. Este exemplo é uma aplicação do campo do Reconhecimento de Padrões (RP). Esta é uma disciplina científica que oferece uma ampla gama de métodos aplicáveis a uma vasta variedade de aplicações, tais como Visão Computacional, Reconhecimento de Caracteres, Auxílio a Diagnósticos, entre outras. Esta disciplina visa identificar e descrever regularidades e relações presentes em dados complexos, para ajudar a compreender e tomar decisões informadas sobre o assunto em questão.

2.2 O que é um Padrão

Um padrão é definido como uma coleção de características distintas que descrevem de forma única objetos reais ou abstratos presentes em um conjunto de dados. De maneira formal, um padrão ou vetor pode ser definido como $\mathbf{x}_i = [x_1, x_2, \dots, x_d]^\top$, um conjunto de características, na qual pode ser visto como um ponto no espaço multidimensional. Exemplificando, um padrão poderia ser as características faciais de uma pessoa ou até mesmo imagens médicas, uma vez que a junção desses pequenos dados geram uma informação única.

$$\mathbf{x}_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = [x_1, x_2, \dots, x_d]^\top \quad (2.1)$$

2.3 Aprendizado

Os métodos de aprendizagem de máquina, podem ser classificados quanto ao modo de aprendizado, quatro no total: supervisionado, não supervisionado, semi-supervisionado e por reforço.

No contexto supervisionado, os algoritmos são treinados com dados rotulados, onde as entradas estão associadas a rótulos ou classes conhecidas. Por outro lado, algoritmos não supervisionados trabalham com dados não rotulados, buscando padrões, estruturas ou agrupamentos nos dados. Já os modelos semi-supervisionados, combinam elementos dos supervisionados e não supervisionados. Eles utilizam um conjunto de dados com rótulos e outro sem rótulos para melhorar o aprendizado e a generalização do modelo. Por fim, algoritmos de aprendizado por reforço são usados para treinar agentes a tomar ações sequenciais em um ambiente, visando maximizar recompensas ao longo do tempo. Os agentes aprendem por tentativa e erro, tomando decisões para otimizar seu desempenho com base nas recompensas ou penalidades recebidas.

Neste capítulo abordaremos apenas o aprendizado supervisionado, uma vez que este trabalho irá focar mais em algoritmos supervisionados.

2.3.1 Aprendizado supervisionado

Os métodos supervisionados podem ser de dois tipos; classificação e Regressão. Estes são treinados mediante a utilização de dados que foram previamente rotulados d_i . Na regressão, os modelos são capacitados para fazer previsões de valores numéricos contínuos, $d_i \in \mathbb{R}$, a partir de características de entrada. Por exemplo, um modelo de regressão pode ser empregado para estimar o preço de uma habitação com base em atributos como tamanho e localização.

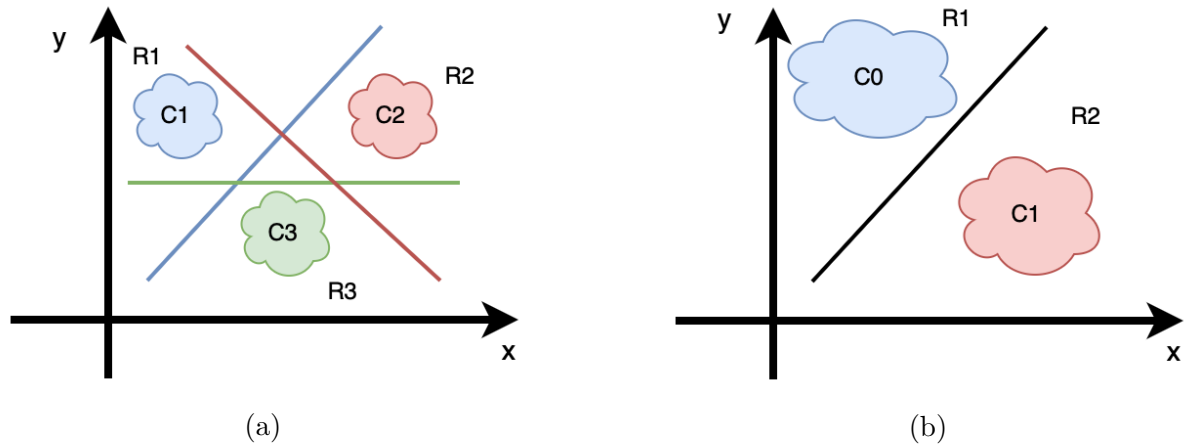
Por outro Lado, na classificação esses modelos aprendem a mapear entradas para categorias ou classes predefinidas, um rótulo d_i discreto. Como exemplo, um modelo de classificação pode ser instruído a discernir entre e-mails de spam e não spam com base em amostras previamente categorizadas.

2.3.1.1 Classificação de padrões

Dado um conjunto $\mathcal{D} = \{(\mathbf{x}_i, d_i)\}_i^N$, em que $\mathbf{x}_i$ é o i -ésimo padrão (2.1) e d_i o rótulo associado. Um problema de classificação, consiste em descobrir a classe na qual um padrão $\mathbf{x}_i$ pertence. O objetivo é dividir o espaço de dados em K regiões $R_1, \dots, R_k$, de tal maneira que cada uma contenha os padrões da classe ou categoria C_k . Quando $K > 2$ o problema é dito multiclasse, já se $K = 2$ é binário, figura 1b. Para esses problemas, o rótulo d_i é discreto, quando existem apenas duas classes, é conveniente $d_i \in \{0, 1\}$, em que $d_i = 1$ representa a classe C_1 e $d_i = 0$ a classe C_0 . Em alguns algoritmos, como os de máquinas de vetores-suporte, que será visto no capítulo 2.4, tem-se que $d_i \in \{-1, 1\}$.

Em um ambiente multiclasse, $d_i \in \{0, 1, 2, \dots, K\}$, figura 1a, da mesma forma que na classificação binária, é mais adequado representar os rótulos em termos binários. Para isso aplica-se um tipo de transformação, chamada codificação 1-de-C, figura 2, nessa o

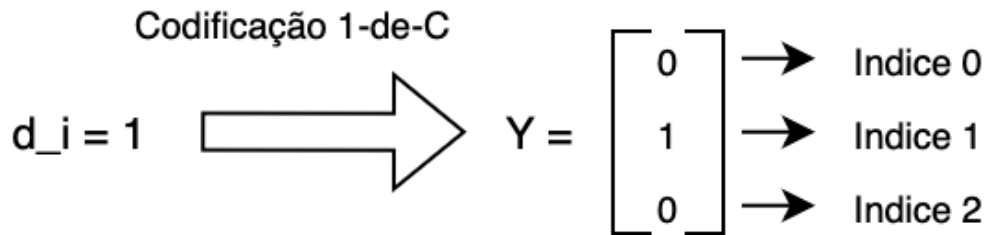
Figura 1 – (a) Exemplo Artificial com 3 classes. (b) Exemplo Artificial com 2 classes



Fonte: Autor.

rótulo é um vetor de tamanho K de tal forma que na classe C_j todos os elementos de d_i são zero, com exceção do termo presente no índice C_j , que é 1. Para exemplificar, suponha $K=3$, se d_i é o rótulo da classe C_1 , então esse pode ser descrito como um vetor $d_i = [0, 1, 0]$.

Figura 2 – Exemplo codificação 1-de-C



Fonte: Autor.

O espaço dos dados, caso as K regiões sejam mutuamente exclusivas $R_i \cap R_j = 0$, poderá ser dividido por superfícies lineares, da forma $\mathbf{w}^T \mathbf{x} + b$, por outro lado caso as regiões sejam sobrepostas, as superfícies geradas no espaço de entrada serão não lineares. Para problemas binários, quando as regiões são disjuntas, a tarefa de classificação é dita linearmente separável. Caso contrario o problema é dito não linearmente separável.

2.4 Máquinas de Vetores-suporte

As redes de vetores-suporte (*Support Vector Networks* - SVN) (CORTES; VAPNIK, 1995), ou na nomenclatura mais difundida, máquinas de vetores-suporte (*Support Vector Machines* - SVM) (BURBIDGE; BUXTON, 2001), é um algoritmo de aprendizado supervisionado que pode ser usado tanto para a classificação de padrões, (*Support Vector Classifier* - SVC), quanto para aproximação de funções, (*Support Vector Regression* - SVR). Fundamentalmente, a SVM é uma máquina linear que contém propriedades interessantes,

como a geração de soluções esparsas (CRISTIANINI; SHAW-ETAYLOR, 2000). Neste capítulo, descreveremos esse modelo no viés da classificação.

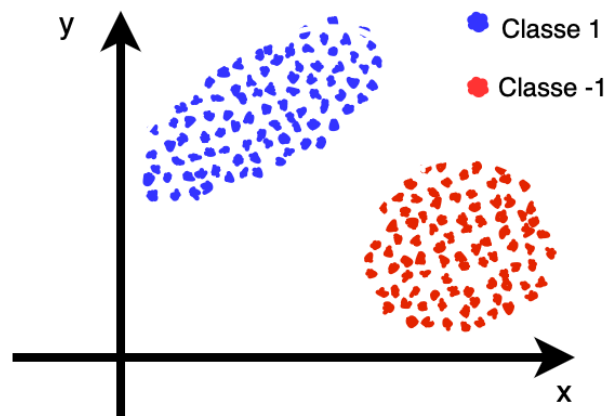
No contexto da classificação de padrões, o objetivo das máquinas de vetores-suporte é encontrar um hiperplano de separação em que a distância entre os padrões da classe positiva C_{+1} e da classe negativa C_{-1} seja máxima. Para isso, o processo de aprendizagem dos SVC's objetiva não só a minimização do risco empírico (*Empirical Risk Minimization*), como também a minimização do risco estrutural (*Structural Risk Minimization*). (NETO, 2011)

A minimização do risco empírico está associada ao erro interno ou de treinamento E_{in} , ele nos informa quantos padrões de treinamento o modelo não conseguiu classificar de forma correta. Por outro lado, a minimização do risco estrutural está associada ao erro de generalização, externo ou de teste (instâncias do conjunto de dados não vistas no processo de aprendizagem) E_{out} . Dessa forma, as máquinas de vetores-suporte aumentam a capacidade de generalização.

2.4.1 Definições e conceitos

Dado um conjunto de dados, formado por $\mathcal{D} = \{(\mathbf{x}_i, d_i)\}_1^N$, o objetivo de um SVC é estimar a função $\mathcal{U} : \mathbb{R}^N \rightarrow \{-1, +1\}$, na qual $\mathbf{x}_i$ é o padrão de entrada para o i -ésimo exemplo e d_i é a saída desejada. A função $\mathcal{U}$ deve classificar as instâncias $(\mathbf{x}_i, d_i)$, em outros termos, $\hat{y}_i = \mathcal{U}(\mathbf{x}_i) = d_i$. No processo de aprendizado, o conjunto dos dados é dividido em dois conjuntos, $\mathcal{D} = (\mathcal{D}_{in} \cup \mathcal{D}_{out})$, um conjunto $\mathcal{D}_{in}$ para estimar a função e outro $\mathcal{D}_{out}$ para guardar os exemplos não aplicados, $\mathcal{D}_{in}$ é chamado de conjunto de treino, já o $\mathcal{D}_{out}$ de conjunto de teste.

Figura 3 – Problema de classificação, linearmente separável, com duas classes, classe 1 (azul) e classe -1 (vermelha).



Fonte: Autor.

Considerando que as máquinas de vetores-suporte, aprendem de forma supervisionada. Suponha um problema de classificação binária linearmente separável, no qual $d_i \in \{-1, +1\}$ e $\mathbf{x}_i \in \mathbb{R}^2$, representado na figura 3. É sabido que, em um espaço d -dimensional, existem infinitas soluções para o problema, na qual podem ser representadas na forma da Equação de um hiperplano.

$$\mathbf{w}^\top \mathbf{x} + b = 0 \quad (2.2)$$

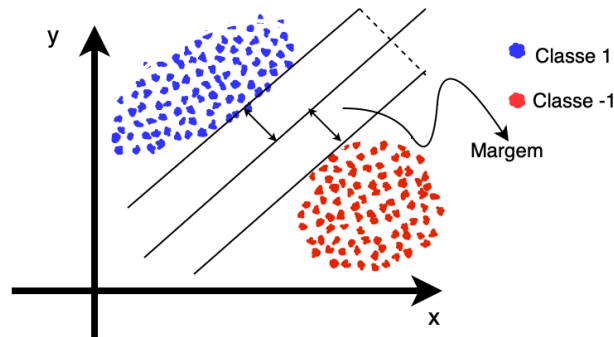
contanto que o vetor $\mathbf{w} \in \mathbb{R}^2$ e o intercepto (viés) $b \in \mathbb{R}$, sejam tais que o espaço de dados consiga ser separado em dois subespaços, regiões R_{+1} e R_{-1} , de modo que todos os padrões da classe C_{+1} estejam na região R_{+1} , e o resto na R_{-1} . O conjunto de hiperplanos que resolvem essa tarefa, devem seguir as seguintes restrições lineares

$$d_i = \begin{cases} +1, & \text{se } \mathbf{w}^\top \mathbf{x}_i + b \geq \mathcal{T}, \\ -1, & \text{se } \mathbf{w}^\top \mathbf{x}_i + b \leq -\mathcal{T}. \end{cases} \quad (2.3)$$

sabendo que as máquinas de vetores-suporte minimizam tanto o erro estrutural quanto o erro empírico, o conjunto solução é reduzido para apenas um hiperplano, aquele que possui uma distância máxima dele até as classes C_{+1} e C_{-1} . Em outras palavras, existe uma margem de tamanho ρ , que representa a menor distância entre o hiperplano ótimo e o padrão de treinamento mais próximo, maximizando a margem as SVM's conseguem encontrar o hiperplano ótimo, descrito por

$$\mathbf{w}_*^\top \mathbf{x} + b_* = 0 \quad (2.4)$$

Figura 4 – Representação da margem das máquinas de vetores-suporte



Fonte: Autor.

Dado a Equação 2.2 e as restrições, a função discriminante do classificador SVM pode ser descrita como

$$g(\mathbf{x}) = \mathbf{w}_*^T \mathbf{x} + b_* \quad (2.5)$$

em que $\mathbf{w}_*$ é o vetor de pesos ótimo e b_* o viés ótimo. Em problemas de classificação de padrões, é aplicada a função sinal, logo

$$g(\mathbf{x}) = \text{sign}(\mathbf{w}_*^T \mathbf{x} + b_*) \quad (2.6)$$

de forma que

$$g(\mathbf{x}) = \begin{cases} +1, & \text{se } \mathbf{w}_*^T \mathbf{x}_i + b_* \geq 0, \\ -1, & \text{se } \mathbf{w}_*^T \mathbf{x}_i + b_* < 0. \end{cases} \quad (2.7)$$

para encontrar uma solução ótima, o conjunto $D_{in} = \{(\mathbf{x}_i, d_i)\}_1^N$ deve ser utilizado de tal forma que

$$d_i = \begin{cases} +1, & \text{se } \mathbf{w}_*^T \mathbf{x}_i + b_* \geq +1, \\ -1, & \text{se } \mathbf{w}_*^T \mathbf{x}_i + b_* \leq -1. \end{cases} \quad (2.8)$$

Os padrões do conjunto de treinamento que satisfazem a Equação 2.8 com o sinal de igualdade são nomeados como vetores-suporte. Suponha $\mathbf{x}_p$ como sendo a projeção de $\mathbf{x}$ sobre o hiperplano ótimo, e r a distância do ponto até o hiperplano. Portanto, um padrão $\mathbf{x}$ pode ser representado como:

$$\mathbf{x} = \mathbf{x}_p + r \frac{\mathbf{w}_*}{\|\mathbf{w}_*\|} \quad (2.9)$$

Dado as Equações 2.6, 2.9 e sabendo que no ponto $\mathbf{x}_p$, $g(\mathbf{x}_p) = 0$, a distância r é descrita como

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{w}_*^T \mathbf{x} + b_* \\ g(\mathbf{x}) &= \mathbf{w}_*^T \left[\mathbf{x}_p + r \frac{\mathbf{w}_*}{\|\mathbf{w}_*\|} \right] + b_* \\ g(\mathbf{x}) &= \mathbf{w}_*^T \mathbf{x}_p + b_* + \mathbf{w}_*^T r \frac{\mathbf{w}_*}{\|\mathbf{w}_*\|} \\ g(\mathbf{x}) &= 0 + r \frac{\mathbf{w}_*^T \mathbf{w}_*}{\|\mathbf{w}_*\|} \\ g(\mathbf{x}) &= r \|\mathbf{w}_*\| \\ r &= \frac{g(\mathbf{x})}{\|\mathbf{w}_*\|} \end{aligned} \quad (2.10)$$

sabendo que $r^{(s)}$ é a distância dos vetores-suporte para o hiperplano ótimo. Logo, a partir das Equação 2.10, temos que

$$\begin{aligned} r^{(s)} &= \frac{g(\mathbf{x}^s)}{\|\mathbf{w}_*\|} \\ r^{(s)} &= \frac{1}{\|\mathbf{w}_*\|} \end{aligned} \quad (2.11)$$

Dado um vetor-suporte $\mathbf{x}_i^{(s)}$ e sabendo que $r^{(s)}$ indica a distância. Nesse contexto, a margem de separação ótima $\rho = 2r^{(s)}$ pode ser obtida a partir da Equação 2.11, tal que

$$\rho = 2r^{(s)} = \frac{2}{\|\mathbf{w}_*\|} \quad (2.12)$$

Dando seguimento, a formulação do problema de obtenção do hiperplano ótimo pode ser descrita a partir da Equação 2.12, com a maximização da margem tem-se que

$$\max \rho = \max \frac{2}{\|\mathbf{w}_*\|} \quad (2.13)$$

reformulando a Equação 2.13, tem-se que

$$\max \frac{2}{\|\mathbf{w}_*\|} \iff \min \frac{1}{2} \|\mathbf{w}_*\| \iff \min \frac{1}{2} \mathbf{w}^\top \mathbf{w} \quad (2.14)$$

Logo a função de custo do SVM é

$$v(\mathbf{w}) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} \quad (2.15)$$

2.4.2 Princípios do classificador SVM

Dado a Equação 2.14 e combinando as duas linhas da Equação 2.8, pode-se apresentar o problema otimização clássico de obtenção dos parâmetros ótimos do classificador SVM, como

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \mathbf{w}^\top \mathbf{w} \\ \text{s.a} \quad & d_i[\mathbf{w}^\top \mathbf{x}_i + b] \geq 1, i = 1, \dots, N \end{aligned} \quad (2.16)$$

2.4.2.1 Máquinas de vetores-suporte de margem rígida

O problema primal das máquinas de vetores-suporte, Equação (2.16), é uma tarefa de otimização não linear, no qual a função custo é estritamente convexa. Sabendo disso, pode-se aplicar as condições de otimalidade de KKT, a partir da função lagrangiana definida na Equação (2.17).

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i [d_i (\mathbf{x}_i^T \mathbf{w} + b) - 1] \quad (2.17)$$

em que os multiplicadores de Lagrange $\{\alpha_i\}_{i=1}^T$ são grandezas não negativas (i.e. $\{\alpha_i \geq 0\}_{i=1}^T$). Expandindo os termos da Equação Lagrangiana, temos que

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \quad (2.18)$$

Para encontrar a solução, a função deve ser minimizada em relação a $\mathbf{w}$ e b e maximizada em relação a α . Assim, calculando as derivadas parciais, obtém-se as seguintes condições de otimização:

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 &\Rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i d_i \mathbf{x}_i \\ \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 &\Rightarrow \sum_{i=1}^n \alpha_i d_i = 0 \end{aligned} \quad (2.19)$$

Vale ressaltar que, o terceiro termo da Equação $(-b \sum_{i=1}^N \alpha_i d_i)$ da forma expandida é zerado devido as condições de otimalidade resultantes, Equação (2.19). Desta forma, pode-se reescrever a Equação (2.18) da seguinte forma:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i + \sum_{i=1}^N \alpha_i \quad (2.20)$$

Prosseguindo, pode-se notar que o termo $(\sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i)$, quando aplicado juntamente com a condição de otimalidade $\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0$, resulta em $-\mathbf{w}^T \mathbf{w}$. Reescrevendo a Equação (2.20), tem-se que

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} - \mathbf{w}^T \mathbf{w} + \sum_{i=1}^N \alpha_i \\ L(\mathbf{w}, b, \alpha) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^N \alpha_i \end{aligned} \quad (2.21)$$

aplicando a transformação de $\mathbf{w}$, obtida pelas condições de KKT, Equação (2.21), obtém-se

$$L(\alpha) = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^N \alpha_i \quad (2.22)$$

ou

$$L(\alpha) = \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \quad (2.23)$$

A partir das Equação (2.23) e das condições de otimalidade, pode-se formular o problema dual do SVM como

$$\begin{aligned} \max_{\alpha} L(\alpha) &= \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.a. } \sum_{i=1}^n \alpha_i d_i &= 0, \\ \text{s.a. } \alpha_i &\geq 0 \text{ para } i = 1, \dots, n \end{aligned} \quad (2.24)$$

No processo de resolução do problema dual, encontram-se os multiplicadores de Lagrange ótimos $\{\alpha_i^*\}_{i=1}^N$. Posteriormente, o vetor $\mathbf{w}$ e o bias b podem ser calculados por

$$\mathbf{w}_* = \sum_{i=1}^N \alpha_i^* d_i \mathbf{x}_i \quad (2.25)$$

$$b_* = 1 - \mathbf{w}_*^T \mathbf{x}^{(s)} \quad (2.26)$$

A função discriminante, Equação (2.6), do problema primal das máquinas de vetores-suporte pode ser reescrita com base no problema dual, Equação (2.24), com isso

$$f(\mathbf{x}) = \sum_{i=1}^N \alpha_i^* d_i \mathbf{x}_i^T \mathbf{x} + b_* \quad (2.27)$$

2.4.2.2 Máquinas de vetores-suporte de margem flexível

A grande parte dos problemas reais tendem a ser não linearmente separáveis, possuindo uma sobreposição entre os dados que compõe as classes. Dessa forma, encontrar o hiperplano ótimo torna-se uma tarefa mais árdua. Dessa forma, é essencial propor uma formulação para o problema do classificador SVM que leve em consideração essa dificuldade, permitindo que alguns padrões sejam incorretamente classificados.

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, n. \quad (2.28)$$

para isso, são adicionadas ao problema primal, variáveis de folga (*slack variables*) $\xi_i \geq 0, i = 1, \dots, n$. As variáveis de folga, são obtidas durante o processo de otimização do classificador SVM (CORTES; VAPNIK, 1995). Quando $\xi_i > 1$, os padrões estão ao lado oposta ao da sua classe e classificados incorretamente. Por outro lado, se $0 < \xi_i \leq 1$

os vetores ultrapassam a margem de separação, porém estão corretamente posicionados em relação ao hiperplano.

A partir disso, o problema de otimização (primal) para classificadores SVM com margem flexível é formulado como

$$\begin{aligned} \min v(\mathbf{w}, \xi) = \min & \left\{ \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^N \xi_i \right\} \\ \text{s.a. } d_i[(\mathbf{w}^\top \mathbf{x}_i) + b] & \geq 1 - \xi_i, \quad i = 1, \dots, n \\ \text{s.a. } \xi_i & \geq 0, \quad i = 1, \dots, n. \end{aligned} \quad (2.29)$$

em que a constante C é um parâmetro de regularização. Nota-se que, a dimensão VC é minimizada pela maximização da margem, além disso as SVM's buscam da mesma forma minimizar os valores assumidos pelas variáveis de folga (NETO, 2011). Assim, de forma análoga ao processo feito no classificador com margem rígida. Neste caso, a função Lagrangiana, Equação (2.30), deve ser minimizada em relação a $\mathbf{w}$, b e ξ e maximizada em relação a α e β .

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [d_i(\mathbf{x}_i^\top \mathbf{w} + b) - 1 + \xi_i] - \sum_{i=1}^N \beta_i \xi_i \quad (2.30)$$

no qual todos os elementos dos conjuntos $\alpha = \{\alpha_i\}_{i=1}^N$, $\beta = \{\beta_i\}_{i=1}^N$ e $\xi = \{\xi_i\}_{i=1}^N$ apresentam valores não-negativos. Expandindo a Equação (2.30), tem-se que

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^\top \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \alpha_i \xi_i - \sum_{i=1}^N \beta_i \xi_i \quad (2.31)$$

A partir da equação (2.31) pode-se calcular as condições de otimalidade, definidas a partir das derivadas parciais igualadas a zero.

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \alpha, \xi)}{\partial \mathbf{w}} = 0 & \Rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i \\ \frac{\partial L(\mathbf{w}, b, \alpha, \xi)}{\partial b} = 0 & \Rightarrow \sum_{i=1}^N \alpha_i d_i = 0 \\ \frac{\partial L(\mathbf{w}, b, \alpha, \xi)}{\partial \xi} = 0 & \Rightarrow C = \alpha_i + \beta_i \end{aligned} \quad (2.32)$$

Aplicando as condições de otimalidade na Equação (2.31), pode-se reformular a função de Lagrange

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + \sum_{i=1}^N \xi_i (\alpha_i + \beta_i) - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^\top \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \xi_i (\alpha_i + \beta_i) \quad (2.33)$$

equivale a

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^\top \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \quad (2.34)$$

O termo $(-b \sum_{i=1}^N \alpha_i d_i)$ da forma expandida da função Lagrangiana, Equação (2.34), pode ser zerado devido as condições de otimalidade, Equação (2.32). Logo, a Equação (2.34) pode ser reescrita da seguinte forma

$$L(\mathbf{w}, b, \xi, \alpha) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^\top \mathbf{x}_i + \sum_{i=1}^N \alpha_i \quad (2.35)$$

Dando seguimento, ao aplicar o resultado obtido na Equação (2.32) ao termo $(-\sum_{i=1}^N \alpha_i d_i \mathbf{w}^\top \mathbf{x}_i)$ da Equação (2.35), obtém-se $-\mathbf{w}^\top \mathbf{w}$. Logo, tem-se que:

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2} \mathbf{w}^\top \mathbf{w} - \mathbf{w}^\top \mathbf{w} + \sum_{i=1}^N \alpha_i \\ L(\mathbf{w}, b, \alpha) &= -\frac{1}{2} \mathbf{w}^\top \mathbf{w} + \sum_{i=1}^N \alpha_i \end{aligned} \quad (2.36)$$

aplicando a transformação de $\mathbf{w}$, obtida pelas condições de KKT, Equação (2.32), obtém-se

$$L(\alpha) = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_{i=1}^N \alpha_i \quad (2.37)$$

ou

$$L(\alpha) = \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^\top \mathbf{x}_j \quad (2.38)$$

A partir das Equação (2.38) e das condições de otimalidade, pode-se formular o problema dual do SVM de margem flexível como

$$\begin{aligned} \max L(\alpha) &= \max \left\{ \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^\top \mathbf{x}_j \right\} \\ \text{s.a.} \quad &\sum_{i=1}^N \alpha_i d_i = 0 \\ &\text{s.a. } 0 \leq \alpha_i \leq C \text{ para } i = 1, \dots, n \end{aligned} \quad (2.39)$$

2.5 Máquinas de Vetores-suporte de mínimos quadrados

As Máquinas Vetores-suporte de Mínimos Quadrados (*Least Squares Support vectors machines* - LSSVM) (SUYKENS; VANDEWALLE, 1999) representam uma técnica de aprendizado supervisionado derivada das *Support Vector Machines* (SVM), na qual é capaz de solucionar problemas de classificação de padrões e aproximação de funções. O LSSVM difere do SVM em sua abordagem de otimização, a qual consiste em resolver um problema de otimização de mínimos quadrados

As mudanças na formulação do problema primal tornam o processo de otimização do problema mais simples, uma vez que, agora, a partir de um sistema linear, mais precisamente um sistema Karush-Khun-Tucker (KKT) (FLETCHER, 1987), é possível encontrar a solução ótima. Do exposto, o problema de otimização para o classificador LSSVM, apresenta-se na seguinte forma:

$$\begin{aligned} \min_{\mathbf{w}, \xi_i, b} \left\{ \frac{1}{2} \mathbf{w}^\top \mathbf{w} + \gamma \frac{1}{2} \sum_{i=1}^N \xi_i^2 \right\} \\ \text{s.a } d_i[\mathbf{w}^\top \mathbf{x}_i + b] = 1 - \xi_i, i = 1, \dots, N \end{aligned} \quad (2.40)$$

2.5.1 Definições e conceitos

Dado que os classificadores LSSVM podem ser resolvidos a partir de um sistema KKT, o processo de treinamento desse método torna-se mais simples e mais rápido do que o do SVM, que é solucionado, por exemplo, por programação quadrática. Todavia, existem limitações que surgem desta reformulação. Uma delas relaciona-se a perda da natureza esparsa na solução do problema. De outro modo, os multiplicadores de Lagrange $\{\alpha_i\}_{i=1}^n$ obtidos no processo de aprendizagem do classificador LSSVM são não-nulos.

Portanto, logo após a etapa de aprendizagem, é preciso guardar todos os padrões de treinamento, assim como os multiplicadores de Lagrange associados para computar a função discriminante. Logo, em problemas com grandes quantidades de dados, aplicar o LSSVM original torna-se inviável.

A função Lagrangiana $L(\mathbf{w}, b, \xi, \alpha)$ para o problema de otimização primal apresenta-se então da seguinte maneira:

$$L(\mathbf{W}, b, \xi, \alpha) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + \gamma \frac{1}{2} \sum_{i=1}^N \xi_i^2 - \sum_{i=1}^N \alpha_i (d_i(\mathbf{x}_i \mathbf{w} + b) - 1 + \xi_i) \quad (2.41)$$

no qual $\{\alpha_i\}_{i=1}^N$ são os multiplicadores associados à restrição de igualdade do problema primal. As condições de otimalidade para o problema descrito, são:

$$\begin{aligned}
\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \mathbf{w}} = 0 &\Rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i d_i \mathbf{x}_i \\
\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial b} = 0 &\Rightarrow \sum_{i=1}^n \alpha_i d_i = 0 \\
\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \alpha_i} = 0 &\Rightarrow d_i (\mathbf{x}_i^\top \mathbf{w} + b) - 1 + \xi_i \\
\frac{\partial L(\mathbf{w}, b, \xi, \alpha)}{\partial \xi_i} = 0 &\Rightarrow \alpha_i = \gamma \xi_i
\end{aligned} \tag{2.42}$$

Após obter as condições de otimalidade, descritas acima, pode-se formular um sistema de equações lineares, $\mathbf{Ax} = \mathbf{b}$, tal qual

$$\left[\begin{array}{c|c} 0 & -d^\top \\ \hline d & \Omega + \gamma^{-1} \mathbf{I} \end{array} \right] \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{2.43}$$

em que $\Omega \in \mathbb{R}^{N \times N}$ é uma matriz quadrada, que pode ser descrita como $\Omega_{i,j} = d_i d_j \mathbf{x}_i^\top \mathbf{x}_j$, para $i, j = 1, \dots, N$. Ademais, d é um vetor linha que contém os rótulos de uma base e $\mathbf{1}$ é um vetor de dimensão N . Depois de encontrar os multiplicadores de Lagrange e o bias, a saída pode ser calculada de acordo com a função

$$f(\mathbf{x}) = \text{sign} \left(\sum_i^N \alpha_i d_i \mathbf{x}^\top \mathbf{x}_i + b \right) \tag{2.44}$$

O produto $\mathbf{x}^\top \mathbf{x}_i$ pode ser substituído por uma função de kernel da forma $k(\mathbf{x}, \mathbf{x}_i)$.

2.5.2 Obtenção dos Parâmetros Ótimos para o classificador LSSVM

2.5.2.1 Solução baseada na Matriz inversa

A solução do método, que consiste na obtenção de b e α contida no vetor $\mathbf{x}^\top = [b \ \alpha^\top]$, pode ser descrita da seguinte maneira

$$\begin{aligned}
\mathbf{Ax} &= \mathbf{B} \\
\mathbf{A}^{-1} \mathbf{Ax} &= \mathbf{A}^{-1} \mathbf{B} \\
\mathbf{x} &= \mathbf{A}^{-1} \mathbf{B}
\end{aligned} \tag{2.45}$$

ou seja,

$$\mathbf{x} = \left[\begin{array}{c|c} 0 & -d^\top \\ \hline d & \Omega + \gamma^{-1} \mathbf{I} \end{array} \right]^{-1} \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{2.46}$$

bastando apenas que a matriz A seja não-singular.

2.5.2.2 Solução baseada na Pseudo Inversa

Na seção anterior, foi apresentada a solução mais simples para o problema de otimização do LSSVM, que se baseia no cálculo da inversa da matriz $\mathbf{A}$. Entretanto, quando se trata de uma matriz não-quadrada, faz-se necessário recorrer ao método da Pseudo Inversa para a obtenção da solução do sistema (MOORE, 1920; PENROSE, 1955).

$$\mathbf{x} = \mathbf{A}^\dagger \mathbf{B} \quad (2.47)$$

$$\mathbf{A}^\dagger = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \quad (2.48)$$

2.5.2.3 Solução baseada no método de Levenberg-Marquardt

Esta subseção irá atentar-se a discutir outra forma de resolução do problema de otimização do LSSVM, apresentada inicialmente em (NETO, 2011). O método de Levenberg-Marquardt (LM) (RANGANATHAN, 2004), é um algoritmo de otimização iterativo que aplica treinamento em lote, no qual consiste em aperfeiçoamento do método Gauss-Newton, que por sua vez é uma variante do método de Newton.

$$L(\mathbf{z}) = \frac{1}{2} \sum_{i=1}^N e_i^2(\mathbf{z}) = \frac{1}{2} \mathbf{e}(\mathbf{z})^\top \mathbf{e}(\mathbf{z}) \quad (2.49)$$

A ideia básica do método de Levenberg-Marquardt é combinar o método do gradiente descendente $\nabla L(\mathbf{z})$ com o método da matriz de Hessiana $\nabla^2 L(\mathbf{z})$. A matriz de Hessiana é uma matriz que contém todas as derivadas parciais de segunda ordem de uma função. O método de Levenberg-Marquardt ajusta uma função não linear aos dados observados, minimizando a soma dos quadrados das diferenças entre os valores previstos pela função e os valores observados.

Dado uma função de custo $L(\mathbf{z})$ com base nos erros quadráticos, Equação (2.49), em que $e_i(\mathbf{z})$ caracteriza o erro da i -ésima amostra e $\mathbf{e}(\mathbf{z})$ é um vetor de erros. Aplicando uma expansão de Taylor em torno de um ponto $\mathbf{z}_0$ em $\nabla L(\mathbf{z})$ tem-se

$$\nabla L(\mathbf{z}) = \nabla L(\mathbf{z}_0) + (\mathbf{z} - \mathbf{z}_0)^\top \nabla^2 L(\mathbf{z}_0) + \dots \quad (2.50)$$

Sabendo que $L(\mathbf{z})$ é convexa e duas vezes diferenciável, uma condição necessária e suficiente para $\mathbf{z}$ ser um ponto ótimo é

$$\nabla L(\mathbf{z}) = 0 \quad (2.51)$$

Considerando apenas os termos de segunda ordem e desprezando o resto, na expansão de Taylor, Equação (2.50), pode-se aproximar $\Delta \mathbf{z} = (\mathbf{z} - \mathbf{z}_0)$ de forma

$$\Delta \mathbf{z} = -[\nabla^2 L(\mathbf{z}_0)]^{-1} \nabla L(\mathbf{z}_0) \quad (2.52)$$

O gradiente $\nabla L(\mathbf{z})$ e a hessiana $\nabla^2 L(\mathbf{z})$ podem ser escritos com base na matriz de derivadas parciais da função-custo $L(\mathbf{z})$, chamada de matriz jacobiana $\mathbf{J}(\mathbf{z})$. Assim, tem-se que

$$\nabla L(\mathbf{z}) = \mathbf{J}(\mathbf{z})\mathbf{e}(\mathbf{z}) \quad (2.53)$$

$$\nabla^2 L(\mathbf{z}) = \mathbf{J}^\top(\mathbf{z})\mathbf{J}(\mathbf{z}) + \sum_{i=1}^N e_i(\mathbf{z})\nabla^2 e_i(\mathbf{z}) \quad (2.54)$$

Por conta da complexidade de calcular-se a matriz hessiana, foram propostos métodos que aplicam aproximações, como é o caso do método de Levenberg-Marquardt. Levando em conta que o método Levenberg-Marquardt é um método Quase-Newton, assim como método de Gauss-Newton, assume-se que:

$$\sum_{i=1}^N e_i(\mathbf{z})\nabla^2 e_i(\mathbf{z}) \approx 0 \quad (2.55)$$

Com isso o termo adicional em (2.55), pode ser retirado do cálculo da hessiana aproximada. A regra de atualização agora é dada por

$$\Delta \mathbf{z} = -[\mathbf{J}^\top(\mathbf{z})\mathbf{J}(\mathbf{z})]^{-1} \mathbf{J}(\mathbf{z})\mathbf{e}(\mathbf{z}) \quad (2.56)$$

Considerando que a matriz Hessiana aproximada $[\mathbf{J}^\top(\mathbf{z})\mathbf{J}(\mathbf{z})]$ pode não possuir inversa, (LEVENBERG, 1944) propôs adicionar um termo μI ao cálculo dessa matriz, onde μ é um escalar e I é a matriz identidade. Essa alteração leva à seguinte regra de atualização:

$$\Delta \mathbf{z} = -[\mathbf{J}^\top(\mathbf{z})\mathbf{J}(\mathbf{z}) + \mu \mathbf{I}]^{-1} \mathbf{J}(\mathbf{z})\mathbf{e}(\mathbf{z}) \quad (2.57)$$

Após o trabalho de (LEVENBERG, 1944), em (MARQUARDT, 1963) foi proposto substituir a matriz identidade pela matriz diagonal da matriz hessiana aproximada, para contornar outro problema, o crescimento acelerado do μ , uma vez que quando isso acontece a informação dada pela matriz hessiana aproximada não é útil no cálculo da atualização. Esta alteração resulta na seguinte regra de atualização:

$$\Delta \mathbf{z} = -[\mathbf{J}^\top(\mathbf{z})\mathbf{J}(\mathbf{z}) + \mu \text{diag}(\mathbf{J}^\top(\mathbf{z})\mathbf{J}(\mathbf{z}))]^{-1}\mathbf{J}(\mathbf{z})\mathbf{e}(\mathbf{z}) \quad (2.58)$$

Dado a Equação 2.58, e sabendo que o LSSVM é um problema de minimização, $\mathbf{A}\mathbf{z} = \mathbf{b}$, pode-se calcular a matriz jacobiana com base nas derivadas parciais da função de erro, logo

$$\mathbf{J}(z) = \frac{\partial \mathbf{e}(z)}{\partial \mathbf{z}} = -\mathbf{A} \quad (2.59)$$

no qual

$$\mathbf{e}(\mathbf{z}) = \mathbf{B} - \mathbf{A}\mathbf{z} \quad (2.60)$$

Por fim, a atualização do vetor $\mathbf{z}$ que contém os multiplicadores de Lagrange e o viés pode ser calculada de acordo com a seguinte expressão:

$$\mathbf{z}_{i+1} = \mathbf{z}_i + [\mathbf{A}^\top \mathbf{A} + \mu \text{diag}(\mathbf{A}^\top \mathbf{A})]^{-1} \mathbf{A}^\top \mathbf{e}(\mathbf{z}) \quad (2.61)$$

no qual i corresponde a i -ésima iteração, e o componente $\mu \text{diag}(\mathbf{A}^\top \mathbf{A})$ representa um termo de regularização. O método Levenberg-Marquardt pode ser resumido no seguinte pseudocódigo (NETO, 2011):

- PASSO 1 - Atribuir valores de forma aleatória para o vetor $\mathbf{z}_i$, tal que $i = 0$;
- PASSO 2 - Calcular o novo vetor $\mathbf{z}_{i+1}$, com base da Equação (2.61);
- PASSO 3 - Avaliar o erro quadrático médio;
- PASSO 3.1 - Se o erro quadrático médio aumentar então desfazer a atualização do vetor $\mathbf{z}_i$, bem como reduzir o valor de μ e retornar ao [PASSO 2];
- PASSO 3.2 - Caso contrário, atualizar o vetor $\mathbf{z}_i$ e retornar ao [PASSO 2];
- PASSO 4 - Avaliar a convergência;
- PASSO 4.1 - Se o algoritmo convergir então finalizar e retornar o vetor $\mathbf{z}_i$;
- PASSO 4.2 - Caso contrário, retornar ao [PASSO 2].

2.6 Conclusão

Neste capítulo, foi explorado alguns conceitos fundamentais no campo do aprendizado de máquina, começando por entender o que é um padrão. Os padrões são representações de dados que podem ser usadas para fazer previsões e tomar decisões com base em informações passadas. Além disso, foram discutidos os diferentes tipos de aprendizado, que variam desde o supervisionado até o não supervisionado e o aprendizado por reforço.

Nesse cenário, foi discutida a técnica das máquinas de vetores-suporte (SVM), um poderoso algoritmo, que pode ser aplicado tanto em tarefas de classificação quanto para aproximação de funções. As SVM's trabalham identificando um hiperplano de separação que maximiza a margem entre classes distintas. Isso permite uma classificação precisa e robusta dos dados. Contudo, para treinar as máquinas de vetores-suporte, uma das maneiras é aplicando programação quadrática, o que pode gerar um tempo e complexidade maior na aprendizagem de bases de dados com maiores quantidades de dados.

Por fim, foi discutido um outro algoritmo. As máquinas de vetores-suporte de mínimos quadrados (LSSVM), uma variante das SVM's que oferece vantagens em termos de eficiência computacional e facilidade de ajuste de parâmetros. Entretanto, assim com as SVM's, o algoritmo LSSVM apresenta uma limitação, uma vez que produz soluções não esparsas, necessitando salvar todos os vetores-suporte ao final do processo de treino.

3 Técnicas para a obtenção de modelos reduzidos em LSSVM

3.1 Introdução

Atualmente, uma ampla variedade de aplicações pode se beneficiar das informações valiosas geradas por estudos e análises de algoritmos de aprendizado de máquina. Como exemplo, o modelo conhecido como Máquinas de vetores-suporte (*Support Vector Machines* - SVM), devido à sua sólida base matemática, tem sido aplicado em uma ampla gama de problemas, que vão desde classificação binária até tarefas de regressão.

O método SVM apresenta propriedades vantajosas quando comparado a outros modelos de aprendizado de máquina. Uma delas é a geração do hiperplano de decisão com base em um número pequeno de exemplos de entrada, denominados vetores-suporte (*Support Vectors* - SV). Em outras palavras, são geradas soluções esparsas. Porém, no pior caso o algoritmo tem um custo $O(N_{atr} + N_{in} \cdot N_{vs} + N_{atr} \cdot N_{vs} \cdot N_{in})$, no qual N_{in} representa o número de padrões de treinamento, N_{vs} denota o número total de vetores-suporte e N_{atr} representa o número de atributos de $\mathbf{x}_i$ (NETO, 2011).

Em contrapartida, existe o método LSSVM, solucionável a partir de um sistema linear, em que o custo assintótico de treinamento é menor do que o SVM, uma vez que a solução de um sistema linear possui custo assintótico menor do que a de um problema de programação quadrática. Todavia, as soluções geradas pelo classificador LSSVM são não esparsas, diferentemente do SVM, com isso ao final da etapa de treino é necessário armazenar todos os vetores-suporte e todos os multiplicadores de Lagrange associados, para o cálculo da função discriminante.

A fim de construir soluções esparsas com o LSSVM, foram propostos vários métodos para diminuição de complexidade baseados na redução dos vetores-suporte, podendo citar os métodos de poda. Os métodos de poda compreendem um conjunto de técnicas que visam simplificar a estrutura interna dos classificadores LSSVM, sem que o mesmo perca eficiência. A ideia principal sobre os métodos de poda é detalhada por (BURGES, 1996). Neste capítulo serão abordados três métodos da literatura.

3.2 P-LSSVM

O método Pruning LSSVM, proposto por (SUYKENS; LUKAS; VANDEWALLE, 2000), é uma alternativa ao LSSVM tradicional, uma vez que permite o LSSVM eleger um subconjunto de vetores-suporte, pertencentes ao conjunto de treino, para ser o conjunto

de vetores-suporte final. Para isso, o P-LSSVM, aplica os conceitos de poda, no qual padrões do conjunto de treino são selecionado de acordo com o valor dos multiplicadores de Lagrange, com base no valor absoluto (magnitude) $|\alpha_i|$.

A seleção dos vetores-suporte é feita de forma recursiva. A cada iteração, uma pequena porção desses é eliminada, com base nos multiplicadores de Lagrange associados, do conjunto de treino. Seu critério de parada é o desempenho do classificador, no qual é informado pelo usuário, logo caso o desempenho venha a ser menor já na segunda iteração, não haverá poda. Para isso, utiliza-se de um conjunto de validação, subconjunto de treinamento, e a cada iteração verifica-se o desempenho continua o mesmo com base nesse conjunto de validação. Para as simulações deste trabalho, aplicou-se como critério de parada a porcentagem de redução. De forma resumida, de acordo com (NETO, 2011), o método Pruning LSSVM tem quatro passos principais:

- PASSO 1 - Treinar um classificador LSSVM com todo o conjunto de treinamento.
- PASSO 2 - Remover uma pequena quantidade dos vetores pertencentes ao conjunto de treinamento, cujos valores de $|\alpha_i|$ sejam os menores dentre os existentes.
- PASSO 3 - Retreinar o classificador LSSVM usando o método da pseudo inversa com o conjunto de treinamento reduzido.
- PASSO 4 - Retornar ao PASSO 2, desde que a performance no conjunto de validação não diminua. Caso contrário, finalizar o processo com o conjunto de treinamento da iteração anterior.

3.3 IP-LSSVM

De forma semelhante, (CARVALHO; BRAGA, 2009) propõem outra forma de solucionar o problema da perda de esparsidade vista em classificadores LSSVM. Contudo, desta vez a análise dos multiplicadores de Lagrange é feita a partir do seu valor direto. Esse método, obtém o conjunto de vetores-suporte em dois passos principais. No primeiro, aplica-se a inversa para encontrar a solução do sistema linear. Posteriormente, na segunda, é empregado o método da pseudo inversa para resolver o novo sistema linear, modificado no processo de poda, no qual colunas são removidas com base nos multiplicadores de Lagrange obtidos na etapa anterior. De acordo com (NETO, 2011), o processo de treinamento do classificador IP-LSSVM, pode ser descrito como:

- PASSO 1- Treinar um classificador LSSVM com todo o conjunto de treinamento, usando a matriz inversa pois $\mathbf{A}$ é uma matriz quadrada.
- PASSO 2 - Especificar um parâmetro τ , que define o percentual dos vetores de treinamento que serão vetores-suporte.
- PASSO 3 - Ordenar os vetores de treinamento $(\mathbf{x}_i, d_i)_{i=1}^N$ por seus valores de α_i .

- PASSO 4 - Selecionar um percentual dos dados de treinamento $(1 - \tau)$ que corresponde aos menores valores de α_i .
- PASSO 5 - Construir uma matriz não-quadrada $\mathbf{A}_{rs}$ pela remoção das colunas de $\mathbf{A}$ associadas aos vetores com menores valores de α_i .
- PASSO 6 - Solucionar o sistema linear $\mathbf{x}_{rs} = \mathbf{A}^*b$, por $\mathbf{A}^* = (\mathbf{A}_{rs}^\top \mathbf{A}_{rs})^{-1} \mathbf{A}_{rs}^\top$.
- PASSO 7 - Os vetores-suporte são os vetores do conjunto de treinamento associados às colunas contidas na matriz $\mathbf{A}_{rs}$.
- PASSO 8 - Os valores de α_i e b para os vetores-suporte podem ser obtidos de $\mathbf{x}_{rs}$, enquanto o valor de α_i para os outros vetores é igual a zero.

3.4 Fixed-Size Genetic Algorithm for Sparse LSSVM

Um método mais recente que aborda a redução do número de vetores de suporte para LSSVM é o classificador chamado *Fixed-Size Genetic Algorithm for Sparse LSSVM* (FSGAS-LSSVM) (NETO; SILVA, 2015). Este método utiliza algoritmos genéticos para realizar a poda dos padrões que não são vetores-suporte. Em sua simulação, cada indivíduo ou cromossomo é representado por um vetor binário composto por zeros e uns, em que “1” indica que o padrão será considerado como vetor de suporte, enquanto “0” indica o contrário. A função de “fitness” desse algoritmo é baseada na acurácia do classificador, sendo que busca-se maximizar esse valor para obter uma solução ótima.

De acordo com a referência (NETO; SILVA, 2015), o algoritmo de treinamento do FSGAS-LSSVM pode ser descrito da seguinte forma:

- PASSO 1 - Iniciar $t=0$, no qual t indica a geração atual.
- PASSO 2 - Gerar uma população inicial $P(t)$ de indivíduos, i.e, setar os genes para uma quantidade fixa de “uns” e construir suas relativas matrizes $\{\mathbf{D}_i\}_{i=1}^S$, no qual s é o numero de indivíduos na geração t , randomicamente;
- PASSO 3 - Para cada individuo em $P(t)$
 - PASSO 3.1 - Resolver o LSSVM linear $(\mathbf{z}_i = (\mathbf{D}_i^\top \mathbf{D}_i)^{-1} \mathbf{D}_i^\top \mathbf{l}_i)$;
 - PASSO 3.2 - Avaliar a função fitness, i.e, do conjunto treino (ou validação).
- PASSO 4 - Enquanto $t < t_{max}$, onde t_{max} é o numero máximo de gerações
 - PASSO 4.1 - Selecione os i -ésimos indivíduos e suas matrizes $\mathbf{D}_i$ usando o método de seleção da roleta.
 - PASSO 4.2 - Realizar a operação de cruzamento por ordem nos indivíduos previamente selecionados para gerar novos indivíduos.
 - PASSO 4.3 - Realizar a operação de mutação nos indivíduos selecionados.
 - PASSO 4.4 - Computar $t = t + 1$
 - PASSO 4.5 - Construir a matriz $\mathbf{D}_i$ para cada individuo em $P(t + 1)$.
 - PASSO 4.6 - Avaliar as acurácias do conjunto de treinamento (ou validação)

para cada $\mathbf{D}_i$

- PASSO 5 - Selecionar o melhor individuo ou a solução.

3.5 Modelos propostos

Como visto nas seções anteriores, os modelos da literatura utilizam-se tanto de técnicas iterativas quanto não iterativas, para podar vetores-suporte do modelo LSSVM. A ideia de retirar-se vetores-suporte a cada iteração ou a cada um número de iterações, é vantajosa, pois o modelo conseguira reaprender melhor os dados apos a poda. Seguindo essa ideia, foi proposto dois modelos de redução de vetores-suporte para LSSVM iterativos.

3.5.1 Definições e conceitos

O primeiro e mais importante conceito dos algoritmos a seguir, é a forma de como os algoritmos removem os vetores-suporte, e o porque alguns vetores-suporte estão mais condicionados a serem escolhidos. Para responder estas perguntas, será comentado alguns conceitos de vetores-suporte.

Pela definição do SVM, sabe-se que os multiplicadores de Lagrange α_i refletem a relevância dos vetores-suporte, padrões de treino $\mathbf{x}_i$, para a modelagem da superfície de decisão. Já para o LSSVM, a condição $\alpha_i = \gamma\xi_i$, implica que o valor dos multiplicares de Lagrange é diretamente proporcionais à variável de folga associada ao padrão de treinamento. Isso pode implicar que, assim como no SVM, o valor do multiplicador de Lagrange pode determinar a importância de um vetor-suporte na solução. (NETO, 2011)

Levando isso em consideração, as duas propostas seguem a mesma ideia, analisar os multiplicadores de Lagrange, para podar vetores-suporte. Com isso a análise é feita a partir do seu valor absoluto (magnitude) do multiplicador de Lagrange, similarmente ao que ocorre com o P-LSSVM. A ideia geral, é ordenar o vetor dos multiplicadores com base em seu valor absoluto $|\alpha_i|$, preservando as relações com seus determinados padrões, e remover os que possuírem os menores valores. Valendo ressaltar ainda que zerar um multiplicador de Lagrange implica na remoção do seu vetor-suporte associado, no calculo da função (NETO, 2011).

Para aplicar essa ideia, é necessário um algoritmo de otimização iterativo para resolver o LSSVM. Com esse proposito, foi escolhido o método Levenberg-Marquardt, pois além de ser iterativo, é um algoritmo de otimização de segunda-ordem (seção 2.5.2.3) alcançando o ponto ótimo em menos iterações.

3.5.2 Fixed Size Levenberg Marquardt LSSVM (Proposta 1)

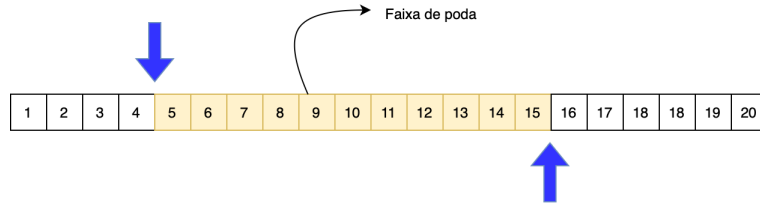
3.5.2.1 Introdução

O classificador *Fixed Size Levenberg Marquardt - LSSVM* (FSLM-LSSVM), é um método derivado do LSSVM tradicional que visa a encontrar o ponto ótimo $\mathbf{z}^*$ do sistema linear KKT de forma iterativa, ao mesmo tempo em que é removido uma quantidade fixa de vetores-suporte. Com esse propósito, foi adaptado o método de Levenberg-Marquardt para fins de remoção de vetores-suporte com base nos valores dos multiplicadores de Lagrange obtidos durante o processo iterativo para alcançar a solução ótima.

3.5.2.2 Janela de poda

Levando em conta a proposta, uma poda iterativa, deve definir-se quando o algoritmo deve podar os vetores-suporte. Tendo isso em vista, foi determinado uma Janela de poda, na qual indica as iterações em que os vetores-suporte serão removidos durante a execução deste método. Logo, dado N iterações, os padrões seriam podados na faixa $\kappa \leq t \leq N_t - \kappa$. Por exemplo, figura 5, definiu-se $N_t=20$ (20 iterações) e $\kappa = 5$, portanto a faixa de poda será $5 \leq t \leq 15$. Vale destacar que nas simulações computacionais realizadas neste trabalho o valor de $\kappa = 3$.

Figura 5 – Janela de poda (Proposta 1)



Fonte: Autor.

3.5.2.3 Solução Esparsa

Como foi comentado na seção 3.5.1, uma solução esparsa será alcançada a partir da remoção dos vetores-suporte com os menores valores de multiplicadores de Lagrange. A remoção de colunas da matriz A equivale a zerar os multiplicadores de Lagrange associados (CARVALHO; BRAGA, 2009). O algoritmo FSLM-LSSVM durante a janela de poda remove uma quantidade específica de colunas de A e linhas $\mathbf{z}$, as linhas de $\mathbf{z}$ removidas, são aquelas com o multiplicador de Lagrange igual a zero, ou seja as linhas que multiplicam as colunas retiradas da matriz A .

$$\Upsilon = \frac{N_{in} - N_{in}\mathcal{R}_{\%}}{N_t - 2\kappa} \quad (3.1)$$

A equação 3.1, indica o número de padrões podados, colunas removidas de A , em cada iteração da janela. As variáveis são as seguintes; N_{in} número total de exemplos

do conjunto de treino D_{in} , $\mathcal{R}_{\%}$ percentual de redução, N_t numero total de iterações e κ variável que define a janela de poda. Caso aconteça de Υ não ser um numero inteiro, na ultima iteração da janela são podados $\Upsilon + \psi$, no qual ψ , equação 3.2, é apenas o resto da divisão da equação 3.1

$$\psi = \{N_{in} - N_{in}\mathcal{R}_{\%}\} \bmod \{N_t - 2\kappa\} \quad (3.2)$$

3.5.2.4 Algoritmo

O passo a passo a seguir descreve o fluxo de treinamento do FSLM-LSSVM, na qual os parâmetros e hiperparâmetros de entrada são: coeficiente μ , percentual de redução $\mathcal{R}_{\%}$, número de iterações N_t e faixa de remoção κ . Inicialmente, o algoritmo calcula, a partir do conjunto de treino, a matriz $\mathbf{A}$, vetor $\mathbf{b}$ e inicializa ,aleatoriamente ,o vetor de pesos $\mathbf{z}_i$. Dando seguimento, os próximos passos do algoritmo são: o cálculo de erro da i-ésima iteração, o cálculo da variação $\Delta\mathbf{z}$ e cálculo do erro quadrático médio (*Mean Squared Error* - MSE). Após isso, é feita uma análise do valor do MSE, com e sem ajuste no vetor de pesos $\mathbf{z}_i$, caso aumente o coeficiente μ é diminuído, caso contrário atualiza-se o vetor $\mathbf{z}_i$. Após esse passo serão ordenados, apenas, os multiplicadores de Lagrange e, a partir desses, é verificado quais colunas de $\mathbf{A}$ e quais linhas de $\mathbf{z}$ são removidas.

- PASSO 1 - Construir a matriz $\mathbf{A}$ e o vetor $\mathbf{b}$, com base em $\mathcal{D}_{in}$, conjunto de treino, e inicializar, de maneria aleatória, o vetor $\mathbf{z}_i$.
- PASSO 2 - Calcular o erro, $e_i(\mathbf{z}_i) = \mathbf{b} - \mathbf{A}^T\mathbf{z}_i$, da i-ésima iteração.
- PASSO 3 - Calcular a variação, $\Delta\mathbf{z} = [\mathbf{A}^T\mathbf{A} + \mu \text{diag}(\mathbf{A}^T\mathbf{A})]^{-1}\mathbf{A}^T\mathbf{e}_i(\mathbf{z}_i)$
- PASSO 4 - Calcular os erros quadráticos médios, e verificar se $\mathbf{MSE}(\mathbf{z}_i + \Delta\mathbf{z}) < \mathbf{MSE}(\mathbf{z}_i)$, se sim atualiza-se o vetor $\mathbf{z}_{i+1} = \mathbf{z}_i + \Delta\mathbf{z}$, caso contrário $\mu = \frac{1}{10}\mu$.
- PASSO 5 - Se $\kappa < i < N - \kappa$, então remover Υ colunas de $\mathbf{A}$ e linhas de $\mathbf{z}$ cujos valores de $|\alpha_i|$ sejam os menores dentre os existentes.
- PASSO 6 - Se $i = N - \kappa$, então remover $\Upsilon + \psi$ colunas de $\mathbf{A}$ e linhas de $\mathbf{z}$ cujos valores de $|\alpha_i|$ sejam os menores dentre os existentes.
- PASSO 7 - Voltar ao passo 2, até o número de iterações acabar.

3.5.3 Reduced Fixed set Levenberg Marquardt LSSVM (Proposta 2)

3.5.3.1 Introdução

O classificador *Reduced Fixed set Levenberg Marquardt - LSSVM* (RFSLM-LSSVM), a segunda proposta deste trabalho, tem como objetivo achar o ponto ótimo $\mathbf{z}^*$ iterativamente. Assim como seu antecessor, FSLM-LSSVM, para RFSLM-LSSVM foi adaptado o método de Levenberg-Marquardt para fins de remoção de vetores-suporte, na qual a análise dos multiplicadores de Lagrange é feita a partir do seu valor absoluto (magnitude). A ideia principal do algoritmo proposto, esta na realização de operações com subconjuntos, de tal forma que seja necessário realizar cálculos matriciais, apenas, com uma porção dos dados em todas as iterações.

3.5.3.2 Reduced fixed sets

A característica fundamental dessa segunda proposta, concentra-se na diminuição da complexidade das operações matriciais realizadas pelo método de Levenberg-Marquardt, quando aplicado para resolver o LSSVM, em busca do ponto ótimo. O método de Levenberg-Marquardt tem diversas operações matriciais para iterativamente encontrar o ponto ótimo, semelhantemente a outros algoritmos da literatura, porem todas essas operações possuem um gasto computacional, implicando que alguns algoritmos não consigam processar grandes bases de dados. Neste trabalho, a maior base de dados testada foi a Ripley com 1250 dados em seu conjunto.

Pode-se comprovar esse gasto computacional no passo 2 do algoritmo (2.5.2.3). No qual o método precisa calcular a atualização do do vetor $\mathbf{z}_t$. Para isso, é necessário realizar operações de multiplicação, soma e inversão de matrizes. Todas esses cálculos tem um consumo da capacidade de processamento, que é diretamente proporcional ao tamanho da base de dados de treino. Logo, se a base de dados for muito grande, o tempo e o processamento seriam, igualmente, grandes.

Dessa forma, foi criado um novo classificador RFSLM-LSSVM. A ideia desse algoritmo gira em torno de operações com subconjuntos dos dados de treino, porções fixas dos dados, em que diminua-se a complexidade dos cálculos matriciais. Em resumo, todas as iterações o algoritmo trabalha com um subconjunto fixo dos dados. No qual representa os vetores-suporte finais. Por exemplo, caso deseje-se podar 80 % dos dados, o subconjunto conteria apenas 20% dos dados de treino, e todas as operações matriciais seriam feitas apenas com essa porção dos dados.

Detalhando, dado um conjunto de treino $\mathcal{D}_{in}$ contendo N_{in} amostras e uma porcentagem de poda $R_{\%}$. O subconjunto dos vetores-suporte sera definido como $\mathcal{D}_{vs}^{\mathcal{F}_{\%}}$, com $\mathcal{F}_{\%} = 1 - R_{\%}$. O restante dos dados ficaram em $\mathcal{D}_{in}^{1-\mathcal{F}_{\%}}$, no qual $\mathcal{D}_{in} = \mathcal{D}_{vs}^{\mathcal{F}_{\%}} \cup \mathcal{D}_{in}^{1-\mathcal{F}_{\%}}$. A matriz $\mathcal{A}$ de vera ser

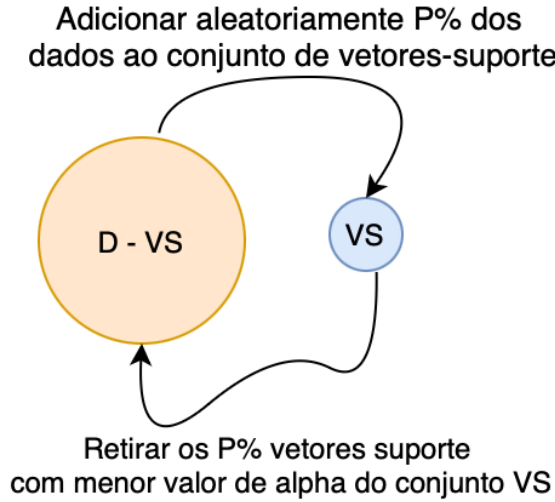
$$\mathcal{A} = \mathbf{A}^{\mathcal{F}\%} = \left[\begin{array}{c|c} 0 & -d^\top \\ \hline d & \Omega + \gamma^{-1}\mathbf{I} \end{array} \right] \quad (3.3)$$

em que $\Omega \in \mathbb{R}^{N_{vs}^{\mathcal{F}\%} \times N_{vs}^{\mathcal{F}\%}}$ é uma matriz quadrada, formada apenas com $\mathcal{F}\%$ dos dados de treino. Sabendo disso, o novo calculo de atualização de pesos será

$$\mathbf{z}_{i+1}^{\mathcal{F}\%} = \mathbf{z}_i^{\mathcal{F}\%} + [\mathcal{A}^\top \mathcal{A} + \mu \text{diag}(\mathcal{A}^\top \mathcal{A})]^{-1} \mathcal{A}^\top \mathbf{e}(\mathbf{z}_i^{\mathcal{F}\%}) \quad (3.4)$$

Após as iterações de treino, a cada τ iterações, será realizada uma troca, $\mathcal{P}\%$ dos vetores-suporte seriam removidos, de acordo com a análise dos multiplicadores, seção 3.5.1, do conjunto $\mathcal{D}_{vs}^{\mathcal{F}\%}$ e seriam escolhidos, aleatoriamente, $\mathcal{P}\%$ dos padrões em $\mathcal{D}_{in}^{1-\mathcal{F}\%}$ e adicionados em $\mathcal{D}_{vs}^{\mathcal{F}\%}$, figura 6. Com isso, o conjunto de vetores-suporte continuara com o mesmo tamanho, porem, os vetores-suporte com menos importância são podados.

Figura 6 – Remoção dos vetores-suporte(Proposta 2)



Fonte: Autor.

Tendo em mente que, agora o conjunto $\mathcal{D}_{vs}^{\mathcal{F}\%}$, contem “novos” dados. È necessário inicializar, novamente, apenas os multiplicadores de Lagrange associados aos padrões retirados do conjunto de vetores-suporte. Assim,

$$\mathbf{z}_{i+1}^{\mathcal{F}\%} = \mathbf{z}_{i+1}^{\mathcal{F}\% - \mathcal{P}\%} \cup \mathbf{z}^{\mathcal{P}\%} \quad (3.5)$$

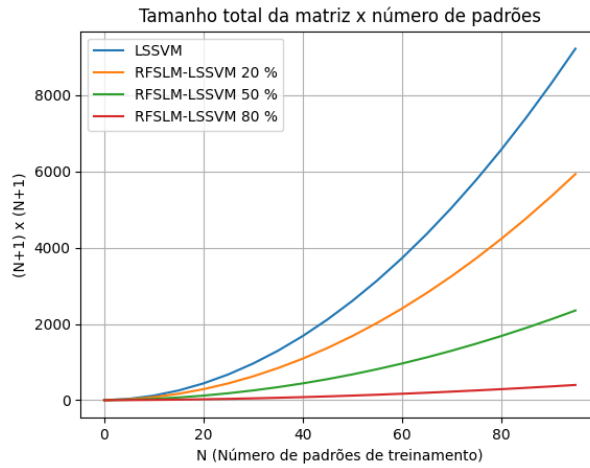
No qual $\mathbf{z}^{\mathcal{P}\%}$ é um vetor de pesos inicializado aleatoriamente.

3.5.3.3 Análise Matricial

Conforme discutido na subseção anterior, é importante notar que o custo computacional, para solucionar o LSSVM, aumenta significativamente à medida que a base de

dados cresce. Esse aumento ocorre devido à dependência direta da matriz $\mathbf{A}$ com o número de padrões no conjunto de treinamento, uma vez que $\Omega \in \mathbb{R}^{N \times N}$. À medida que o conjunto de treinamento se expande, a matriz Ω se torna cada vez maior, o que requer mais recursos computacionais para armazenar, manipular e otimizar. Na figura 7, tem-se um gráfico do tamanho total da matriz $\mathbf{A}$ pelo número de padrões de entrada. Neste é perceptível um aumento exponencial do tamanho da matriz, mesmo com poucos padrões para no conjunto de treinamento. No RFSLM-LSSVM, percebe-se o mesmo comportamento, entretanto como neste trabalha-se com um subconjunto dos vetores-suporte, o aumento não é tão drástico quanto no LSSVM tradicional.

Figura 7 – Gráfico tamanho matriz $\mathbf{A}$ pelo número de padrões de treino.



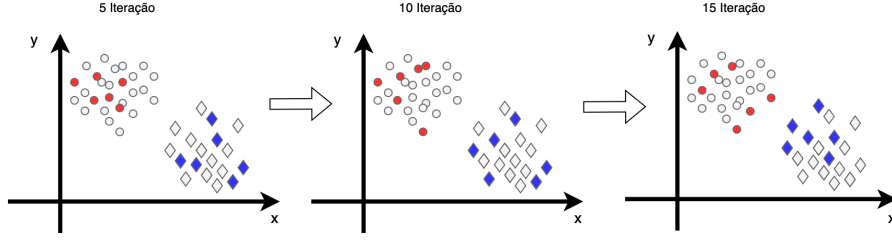
Fonte: Autor.

3.5.3.4 Algoritmo

O processo de treino do classificador RFSLM-LSSVM, requer alguns parâmetros e hiperparâmetros de entrada, estes são: coeficiente μ , percentual de redução $R\%$, número de iterações N_t , iterações de corte τ e porcentagem de redução por iteração de corte $\mathcal{P}\%$.

Na primeira etapa, o conjunto de treino é dividido em dois, um subconjunto para os vetores-suporte $\mathcal{D}_{vs}^{\mathcal{F}\%}$, na qual o seu tamanho é $\mathcal{F}\% = 1 - R\%$. Exemplificando, caso $R = 20\%$, $\mathcal{D}_{vs}^{\mathcal{F}\%}$ terá 80% dos dados de treino. O segundo conjunto, $\mathcal{D}_{in}^{1-\mathcal{F}\%}$ compreende o restante dos dados. Dando continuidade, agora, a partir do conjunto $\mathcal{D}_{vs}^{\mathcal{F}\%}$ será definida a matriz $\mathcal{A}$, vetor $\mathbf{b}$ e logo após é inicializado, aleatoriamente, o vetor de pesos $\mathbf{z}_t$, onde $t=0$.

Os próximos passos do algoritmo, compreendem: o cálculo de erro da i -ésima iteração, o cálculo da variação $\Delta \mathbf{z}$ e cálculo do erro quadrático médio (*Mean Squared Error* - MSE). A partir do MSE, é verificado se o vetor de pesos será atualizado ou a constante μ diminuída.

Figura 8 – Exemplo com $\tau = 5$ (Proposta 2).

Fonte: Autor.

Após essas etapas, a cada τ iterações o algoritmo realizará os cortes em $\mathcal{D}_{vs}^{\mathcal{F}\%}$, assim como foi explicado na seção anterior. O processo se resume em, cortar $\mathcal{P}\%$ dos vetores-suporte com os menores multiplicadores Lagrange ordenados por $|\alpha_i|$, ao mesmo tempo que escolhe, aleatoriamente, $\mathcal{P}\%$ dos exemplos em $\mathcal{D}_{in}^{1-\mathcal{F}\%}$ para adicionar em $\mathcal{D}_{vs}^{\mathcal{F}\%}$. Exemplificando, caso $\tau = 5$ e $\mathcal{P}\% = 5\%$, a cada 5 iterações do algoritmo será retirado 5% dos dados em $\mathcal{D}_{vs}^{\mathcal{F}\%}$ e será adicionado, na mesma proporção, novos exemplos a partir de $\mathcal{D}_{in}^{1-\mathcal{F}\%}$, figura 8.

Dado que, os vetores-suporte relacionados aos multiplicadores de Lagrange são removidos. O algoritmo deve retirar do vetor $\mathbf{z}_{i+1}^{\mathcal{F}\%}$, os multiplicadores associados aos exemplos removidos, gerando $\mathbf{z}_{i+1}^{\mathcal{F}\%-\mathcal{P}\%}$, seção 3.5.3.2. Após isso, para manter a mesma proporção, é inicializado um novo vetor de tamanho $\mathcal{P}\%$, $\mathbf{z}_{\mathcal{P}\%}$, e em seguida é feito a união dos conjuntos, equação 3.5.

O processo de treino irá finalizar após o número de iterações total N for alcançado. Vale destacar que nas simulações computacionais realizadas neste trabalho o valor de $\tau = 5$ e $\mathcal{P}\% = 5\%$. O treinamento do classificador RFSLM-LSSVM pode ser descrito como

- PASSO 1 - Inicializar $\mathcal{D}_{vs}^{\mathcal{F}\%}$, sabendo que $\mathcal{F}\% = 1 - R\%$ restrito a $\mathcal{D}_{vs}^{\mathcal{F}\%} \cap \mathcal{D}_{in}^{1-\mathcal{F}\%} = 0$.
- PASSO 2 - Inicializar aleatoriamente $\mathbf{z}_t^{\mathcal{F}\%}$, onde $t=0$
- PASSO 3 - Criar a matriz $\mathcal{A}$, vetor $\mathbf{b}^{\mathcal{F}\%}$.
- PASSO 4 - Calcular o erro, $\mathbf{e}_t(\mathbf{z}_t^{\mathcal{F}\%}) = \mathbf{b} - \mathcal{A}^T \mathbf{z}_t^{\mathcal{F}\%}$, da t-ésima iteração.
- PASSO 5 - Calcular a variação, $\Delta \mathbf{z} = [\mathcal{A}^T \mathcal{A} + \mu \text{diag}(\mathcal{A}^T \mathcal{A})]^{-1} \mathcal{A}^T \mathbf{e}_t(\mathbf{z}_t^{\mathcal{F}\%})$
- PASSO 6 - Calcular os erros quadráticos médios. Se $\text{mse}(\mathbf{z}_t^{\mathcal{F}\%} + \Delta \mathbf{z}) < \text{mse}(\mathbf{z}_t^{\mathcal{F}\%})$, atualiza-se o vetor $\mathbf{z}_{t+1} = \mathbf{z}_t + \Delta \mathbf{z}$, se não $\mu = \frac{1}{10} \mu$.
- PASSO 7 - A cada τ iterações faça:
 - PASSO 7.1 - Remover $\mathcal{P}\%$ dos vetores-suporte, cujo os valores $|\alpha_i|$ sejam os menores dentre os existentes.
 - PASSO 7.2 - Remover os multiplicadores de Lagrange associados aos vetores-suporte, $\mathbf{z}_t^{\mathcal{F}\%-\mathcal{P}\%}$
 - PASSO 7.3 - Escolher, aleatoriamente, $\mathcal{P}\%$ dos dados em $\mathcal{D}_{in}^{1-\mathcal{F}\%}$, e adicionar em $\mathcal{D}_{vs}^{\mathcal{F}\%}$, respeitando $\mathcal{D}_{vs}^{\mathcal{F}\%} \cap \mathcal{D}_{in}^{1-\mathcal{F}\%} = 0$.
 - PASSO 7.4 - Inicializar aleatoriamente $\mathbf{z}_t^{\mathcal{P}\%}$, onde $t=0$.

- PASSO 7.5 - Fazer com que $\mathbf{z}_{t+1}^{\mathcal{F}\%} = \mathbf{z}_{t+1}^{\mathcal{F}\% - \mathcal{P}\%} \cup \mathbf{z}^{\mathcal{P}\%}$, respeitando as devidas relações com os vetores-suporte.
- PASSO 8 - Volte até o passo 3, até acabar o número de iterações

3.5.3.5 Exemplos

Para uma melhor compreensão de como o RFSLM-LSSVM calcula a solução, vamos utilizar um exemplo artificial. Considere um problema de classificação linearmente separável, onde temos um conjunto de dados de treino $\mathcal{D}_{in} = \{(\mathbf{x}_i, d_i)\}_1^N$, com $N_{in} = 4$ amostras. No qual cada $\mathbf{x}_i \in \mathbb{R}^2$ e $d_i \in \{+1, -1\}$. Vamos aplicar uma poda de 50%, ou seja, $R\% = 50\%$, o que significa que $\mathcal{F}\% = 50\%$.

De acordo com PASSO 1, o conjunto de vetores-suporte será $\mathcal{D}_{vs}^{\mathcal{F}\%}$, logo o número de vetores-suporte será $N_{vs} = N_{in} \times \mathcal{F}\% = 2$. Ainda no PASSO 1, o restante dos dados ficara contido no conjunto $\mathcal{D}_{vs}^{1-\mathcal{F}\%}$. Adiante nos PASSOS 2 e 3, será inicializado $\mathbf{z}_i^{\mathcal{F}\%}$, $\mathcal{A}$ e $b^{\mathcal{F}\%}$, com isso tem-se

$$\mathbf{z}_i^{\mathcal{F}\%} = \begin{bmatrix} b \\ \alpha_0 \\ \alpha_1 \end{bmatrix}_{3 \times 1} \quad b^{\mathcal{F}\%} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}_{3 \times 1} \quad \mathcal{A} = \begin{bmatrix} 0 & -d_0 & -d_1 \\ d_0 & d_0 d_0 \mathbf{x}_0^T \mathbf{x}_0 + \gamma^{-1} & d_0 d_1 \mathbf{x}_0^T \mathbf{x}_1 + \gamma^{-1} \\ d_1 & d_1 d_0 \mathbf{x}_1^T \mathbf{x}_0 + \gamma^{-1} & d_1 d_1 \mathbf{x}_1^T \mathbf{x}_1 + \gamma^{-1} \end{bmatrix}_{3 \times 3} \quad (3.6)$$

Posteriormente, deverá ser calculado o i-ésimo erro e, dependendo deste, o vetor $\mathbf{z}_i^{\mathcal{F}\%}$ é atualizado. Prosseguindo, mais à frente, o PASSO 7 indica que a cada um número de iterações, especificado pelo usuário, será feito o processo de troca, conforme descrito na subseção 3.5.3.2. Para ilustrar esse processo, suponha $\mathcal{P}\% = 50\%$, com isso será removido $N_{vs}^{\mathcal{P}\%} = N_{vs} \times \mathcal{P}\% = 1$. Logo será removido um vetor-suporte do conjunto solução, de acordo com a regra descrita no PASSO 7.1 e no PASSO 7.2 seu multiplicador correspondente. Considerando que o vetor-suporte removido foi $\mathbf{x}_1$, logo

$$\mathbf{z}_i^{\mathcal{F}\% - \mathcal{P}\%} = \begin{bmatrix} b \\ \alpha_0 \end{bmatrix}_{2 \times 1} \quad b^{\mathcal{F}\% - \mathcal{P}\%} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}_{2 \times 1} \quad (3.7)$$

Continuando, no PASSO 7.3 são escolhidos, aleatoriamente, do conjunto $\mathcal{D}_{in}^{1-\mathcal{F}\%}$, vetores-suporte para compor a solução. Suponha que, o vetor-suporte escolhido, aleatoriamente, foi $\mathbf{x}_2$, com isso

$$\mathbf{z}_i^{\mathcal{F}\%} = \begin{bmatrix} b \\ \alpha_0 \\ \alpha_2 \end{bmatrix}_{3 \times 1} \quad b^{\mathcal{F}\%} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}_{3 \times 1} \quad \mathcal{A} = \begin{bmatrix} 0 & -d_0 & -d_2 \\ d_0 & d_0 d_0 \mathbf{x}_0^T \mathbf{x}_0 + \gamma^{-1} & d_0 d_2 \mathbf{x}_0^T \mathbf{x}_2 + \gamma^{-1} \\ d_2 & d_2 d_0 \mathbf{x}_2^T \mathbf{x}_0 + \gamma^{-1} & d_2 d_2 \mathbf{x}_2^T \mathbf{x}_2 + \gamma^{-1} \end{bmatrix}_{3 \times 3} \quad (3.8)$$

Após isso, o algoritmo volta ao fluxo de atualização, PASSO 4 ao 6. O processo acaba quando o número total de iterações é alcançado.

A figura 9, ilustra um segundo exemplo, agora com $N_{in} = 400$, 400 amostras de treino. Sabendo que, $\tau = 5$, foi plotado o gráfico de vetores-suporte a cada cinco iterações, para mostrar como os vetores-suporte se distribuem ao longo do processo iterativo. Vale ressaltar que a “iteração 0” é a primeira distribuição dos vetores-suporte, em outras palavras o primeiro conjunto, inicializado no PASSO 1.

3.6 Conclusão

No presente capítulo, foram analisados diversos modelos encontrados na literatura que abordam o problema da perda de esparsidade do LSSVM. Os dois primeiros modelos, denominados P-LSSVM e IP-LSSVM, resolvem essa problemática ao examinar os multiplicadores de Lagrange. Eles classificam esses multiplicadores com base em seus valores absolutos e, posteriormente, eliminam aqueles com os valores mais baixos. O terceiro modelo mencionado, o FSGAS-LSSVM, fundamenta-se em algoritmos genéticos. Todos os três modelos adotam uma abordagem de poda fixa

Além disso, foram propostos dois novos modelos: *Fixed Size Levenberg Marquardt - LSSVM* (FSLM-LSSVM) e *Reduced Fixed set Levenberg Marquardt - LSSVM* (RFSLM-LSSVM). Os dois são classificadores que combinam a abordagem de poda fixa com um processo iterativo, tornando-os particularmente úteis em situações em que é necessário manter um número fixo de vetores-suporte.

Figura 9 – Vetores-suporte no processo iterativo do RFSLM-LSSVM.



Fonte: Autor.

4 Simulações computacionais

4.1 Metodologia

O principal objetivo dos experimentos é avaliar o desempenho de duas propostas apresentadas no capítulo 3 e, simultaneamente, compará-las com outros métodos encontrados na literatura. Para alcançar esse objetivo, foram empregadas 11 bases de dados reais, conforme detalhado na tabela 1. É importante destacar que, neste trabalho, foi utilizada a versão binária da base de dados relacionada a patologias da coluna. Além das bases de dados reais, também foram criadas três bases de dados artificiais.

Uma dentre essas bases de dados artificiais é linearmente separável, enquanto as outras duas não apresentam essa característica. A criação desses dados artificiais tem como objetivo analisar o comportamento dos vetores-suporte nos modelos FSLM-LSSVM e RFSLM-LSSVM, conforme detalhado na seção 4.3.

Tabela 1 – Lista de bases de dados usadas nesse trabalho.

Base de dados	Abreviação	# Padrões	# Atributos
Artificial 1 (Two Squares)	TWS	1000	2
Artificial 2 (Two Moon)	TWM	800	2
Artificial 3 (Two Circle)	TWC	800	2
Ripley	RIP	1250	2
Haberman	HAB	306	3
Pima Indians Diabets	PID	768	8
Breast Cancer Winconsin	BCW	570	10
Vertebral Column Pathologies	VCP	310	6
Tic-Tac-Toe Endgame	TTT	958	9
Bupa liver disorders	BLD	345	5
Statlog heart disease	HEA	270	13
ionosphere	ION	351	34
Statlog German credit	GCR	1000	20
Statlog Australian credit	AUS	690	14

Fonte: Autor.

Para as simulações, todos os conjuntos de dados seguiram os seguinte fluxo, figura 10. Inicialmente as características da base são normalizadas, de tal forma que a média e variância sejam, respectivamente, zero e um. Logo após, as simulações foram divididas em dois grupos, na qual cada uma segue uma metodologia diferente. Na primeira, a base é dividida em duas partes, um conjunto de treino contendo 80% dos dados e um conjunto de teste com o restante, nomeada segmentação 80/20, metodologia 80/20 ou grupo um.

Na segunda metodologia, a base é repartida em três partes, conjunto de treino com 60%, conjunto de validação com 20 % e um conjunto de teste com o restante, denominada segmentação 60/20/20, metodologia 60/20/20 ou grupo dois. O processo de segmentação dos conjuntos de treino e teste é feito de forma aleatória.

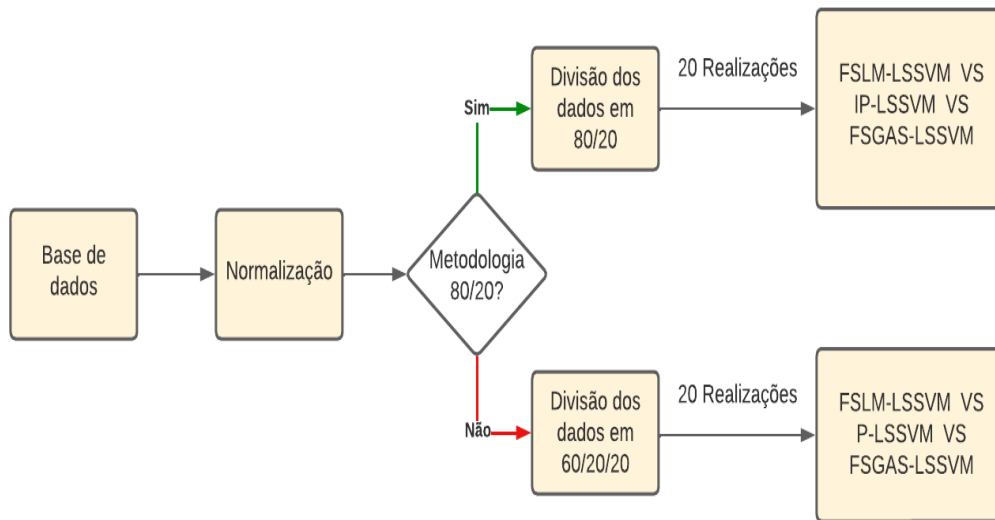
Tabela 2 – Funções de Kernel.

Kernel	função	hiperparâmetros
Linear	$\mathbf{x}_i^T \mathbf{x}_j$	-
RBF	$\exp(-\lambda \ \mathbf{x}_i - \mathbf{x}_j\ ^2)$	λ

Fonte: Autor.

O processo de simular os métodos propostos de acordo com duas metodologias faz-se necessário, visto que o algoritmo P-LSSVM, em sua proposição, utiliza um conjunto de validação para verificar o ponto de término do algoritmo. Logo, no grupo dois, compara-se as propostas com o P-LSSVM e no grupo de simulações um compara-se com o IP-LSSVM.

Figura 10 – Fluxo das simulações computacionais.



Fonte: Autor.

A normalização da base e divisão dos dados é repetida 20 vezes(realizações) para avaliar a variabilidade dos resultados. Objetivando uma comparação mais assertiva, em cada realização, foram aplicados os mesmos hiperparâmetros, tabela 2. Por ultimo, foram aplicados dois tipos de kernel nesse trabalho, Linear e RBF. O kernel RBF foi utilizado para o aprendizado dos problemas artificiais não linearmente separáveis. Os hiperparâmetros foram escolhidos a partir de testes com as bases da tabela 1.

As tabelas de resultados, contêm as colunas TS, Red.(%) e VS na qual cada uma significa, respectivamente número de vetores-suporte total, porcentagem de redução e número de vetores-suporte após a redução.

Tabela 3 – Lista dos hiperparâmetros aplicados nas simulações computacionais.

Hiperparâmetro	Valor	Algoritmo
γ	0.05	ambos
μ	0.001	ambos
κ	3	(FSLM-LSSVM)
τ	5	(RFSLM-LSSVM)
$\mathcal{P}\%$	5 %	(RFSLM-LSSVM)

Fonte: Autor.

4.2 Análise dos dados com bases reais

Em uma primeira análise, foram selecionadas cinco das onze bases de dados reais. Nesta, os algoritmos propostos foram comparados com outros três previamente discutidos no capítulo 3. Os resultados completos, incluindo acurácias acompanhadas de desvio padrão, estão registrados nas tabelas 4 até 13. Nas tabelas 4 e 5, estão os resultados alcançados pelos classificadores FSLM-LSSVM e RFSLM-LSSVM, quando aplicados ao problema da coluna vertebral com dois rótulos. Na tabela 4, estão os resultados de acordo com a metodologia 80/20, com essa nota-se que as duas propostas conseguiram ficar acima dos resultados encontrados no IP-LSSVM. Porém, apenas a segunda proposta conseguiu ter resultados parecidos com o algoritmo FSGAS-LSSVM, com 20% e 50% de redução. Nota-se também que o desvio padrão relacionado foi semelhante.

Tabela 4 – Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema da coluna vertebral.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
VCP	SVM	$85 \pm 3,4$	247	132	–
VCP	LSSVM	$83,63 \pm 4,5$	247	247	–
VCP	IP-LSSVM	$72,98 \pm 5,1$	247	198	20
		$72,66 \pm 6,1$		124	50
		$72,10 \pm 6,0$		50	80
VCP	FSGAS-LSSVM	$83,7 \pm 3,7$	248	198	20
		$83,6 \pm 4,1$		124	50
		$81,2 \pm 3,9$		50	80
VCP	FSLM-LSSVM	$82,42 \pm 5,2$	247	198	20
		$72,66 \pm 6,1$		124	50
		$75,16 \pm 4,7$		50	80
VCP	RFSLM-LSSVM	$82,31 \pm 4,66$	247	198	20
		$81,61 \pm 7,0$		124	50
		$74,62 \pm 6,4$		50	80

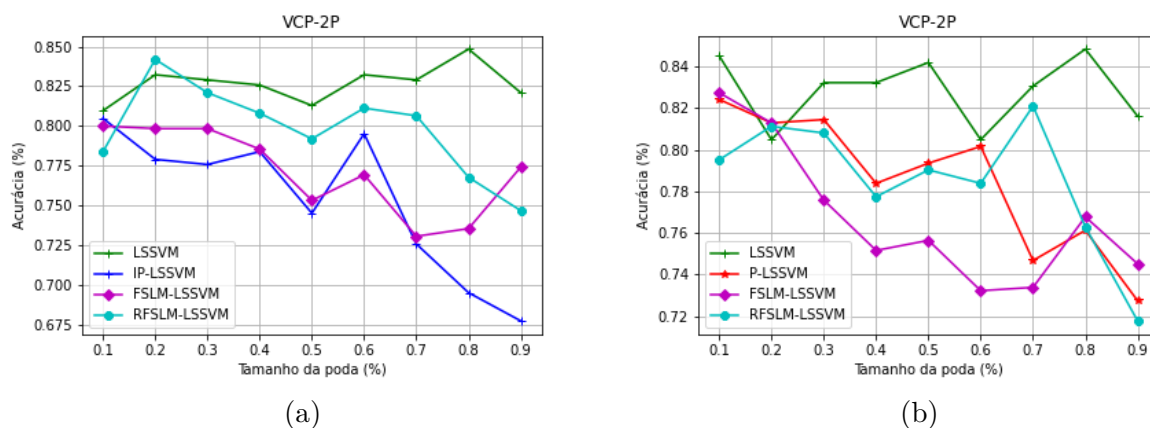
Fonte: Autor.

Tabela 5 – Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema da coluna vertebral.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
VCP	SVM	$85,0 \pm 3,4$	186	132	–
VCP	LSSVM	$83,63 \pm 4,5$	186	186	–
VCP	P-LSSVM	$79,76 \pm 5,7$	186	149	20
		$79,35 \pm 4,8$		93	50
		$75,73 \pm 8,0$		37	80
VCP	FSGAS-LSSVM	$81,7 \pm 4,4$	186	149	20
		$81,1 \pm 4,8$		93	50
		$81,1 \pm 4,4$		37	80
VCP	FSLM-LSSVM	$81,51 \pm 4,7$	186	149	20
		$76,34 \pm 6,0$		93	50
		$75,65 \pm 7,2$		37	80
VCP	RFSLM-LSSVM	$82,25 \pm 2,6$	186	149	20
		$79,03 \pm 6,1$		93	50
		$77,58 \pm 4,4$		37	80

Fonte: Autor.

Os resultados para o segundo grupo de simulações, metodologia 60/20/20, estão na Tabela 5. Nessa pode-se notar uma similaridade com os resultados anteriores, em que os dois algoritmos propostos tiveram resultados satisfatórios, semelhantes com os da literatura, contudo apenas o RFSLM-LSSVM conseguiu taxas próximas do FSGAS-LSSVM. Na figura 11, está uma análise comparativa da acurácia de acordo com o tamanho do corte. Em ambos os gráficos, notou-se que o RFSLM-LSSVM ficou mais próximo da taxa padrão do LSSVM do que o FSLM-LSSVM. Além disso, o desempenho das propostas ficaram bem próximas ao da literatura existente.

Figura 11 – Acurácia *versus* tamanho da poda para as patologias da coluna vertebral. (a) Metodologia 80/20. (b) Metodologia 60/20/20

Fonte: Autor.

Nas tabelas 6 e 7, estão os resultados para o conjunto do câncer de mama. Na tabela 6, nota-se que as acurácias de todos os classificadores foram semelhantes, onde

Tabela 6 – Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema do câncer de mama.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
BCW	SVM	$97,59 \pm 0,7$	546	60	–
BCW	LSSVM	$96,24 \pm 1,1$	546	546	–
BCW	IP-LSSVM	$97,45 \pm 1,2$	546	437	20
		$97,12 \pm 1,2$		273	50
		$96,64 \pm 1,4$		109	80
BCW	FSGAS-LSSVM	$95,9 \pm 1,6$	546	437	20
		$95,9 \pm 1,8$		273	50
		$95,9 \pm 1,6$		109	80
BCW	FSLM-LSSVM	$96,40 \pm 1,3$	546	437	20
		$96,74 \pm 1,4$		273	50
		$96,62 \pm 1,6$		109	80
BCW	RFSLM-LSSVM	$97,79 \pm 1,66$	546	437	20
		$97,06 \pm 1,58$		273	50
		$97,13 \pm 1,74$		109	80

Fonte: Autor.

o FSLM-LSSVM e o RFSLM-LSSVM ficaram com taxas próximas ao LSSVM padrão mesmo com cortes de até 80%. Comparando com os métodos da literatura, notou-se um desempenho satisfatório, no qual as propostas obtiveram resultados similares. Assim como nas patologias da coluna, o desvio padrão das acurácias foi baixo. Dando continuidade, na tabela 7 estão as acurácias aplicando a metodologia 60/20/20. Os resultados foram similares aos anteriores, mantendo-se acurácias próximas aos modelos da literatura, com um desvio padrão baixo.

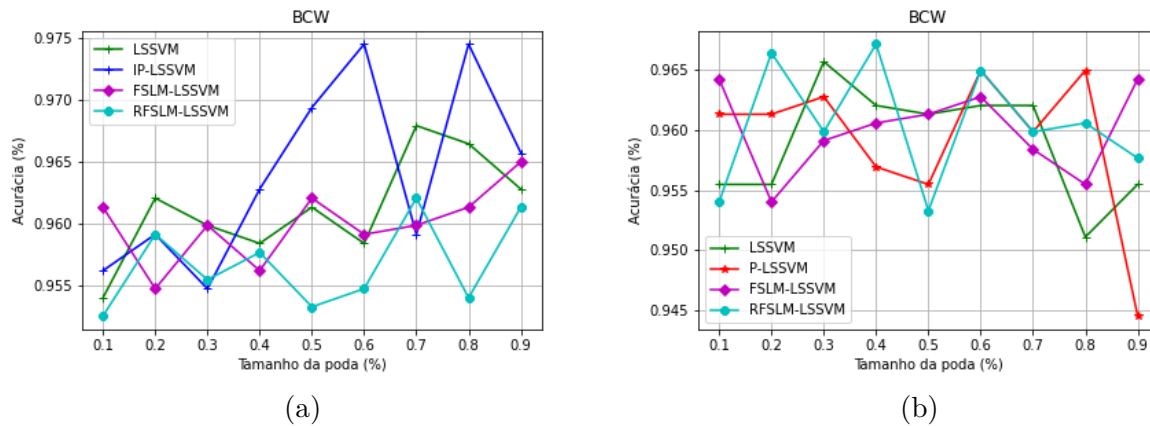
Tabela 7 – Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema do câncer de mama.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
BCW	SVM	$97,54 \pm 1,0$	410	49	–
BCW	LSSVM	$96,11 \pm 1,6$	410	410	–
BCW	P-LSSVM	$96,40 \pm 1,3$	410	328	20
		$96,18 \pm 1,6$		205	50
		$95,62 \pm 1,6$		82	80
BCW	FSGAS-LSSVM	$96,4 \pm 1,6$	410	328	20
		$96,1 \pm 1,8$		205	50
		$96,1 \pm 1,0$		82	80
BCW	FSLM-LSSVM	$97,40 \pm 1,3$	410	328	20
		$96,74 \pm 1,6$		205	50
		$97,01 \pm 1,0$		82	80
BCW	RFSLM-LSSVM	$97,91 \pm 1,79$	410	328	20
		$95,83 \pm 1,60$		205	50
		$96,27 \pm 1,61$		82	80

Fonte: Autor.

Na figura 12, nota-se, primeiramente que o eixo y (Acurácias) tem uma variação bem pequena, variando de -2% a $+2\%$. Assim como nas tabelas, o desempenho de todos ficou similar, conseguindo uma boa taxa de acerto com até 90% de corte.

Figura 12 – Acurácia *versus* tamanho da poda para o conjunto do câncer de mama. (a) Metodologia 80/20. (b) Metodologia 60/20/20



Fonte: Autor.

A próxima base de dados analisada foi a diabetes. Nas tabelas 8 e 9, estão as respectivas acurácias. Na tabela 8, encontram-se os resultados do grupo um, nestes assim como nos anteriores, foi obtido resultados semelhantes com o da literatura, no qual as propostas mesmo com 80% de remoção, ficou com uma acurácia bem próxima ao LSSVM padrão e com um baixo desvio padrão. O segundo grupo, tabela 9, pode-se afirmar que os resultados foram semelhantes, igualmente visto na metodologia 80/20.

A figura 13, percebe-se que o modelo FSLM-LSSVM sofre um decaimento na taxa de acerto em ambas as metodologias. Ao contrário do FSLM-LSSVM, o RFSLM-LSSVM obteve um resultado satisfatório em quase todas as faixas de corte, apenas com uma poda de 70 %, que houve perca no seu desempenho.

Finalizando, tem-se as tabelas 10 até 13 que mostram os resultados das bases de Haberman e Ripley. Todos os resultados obtidos com essas duas bases, foram satisfatórios e semelhantes com os da literatura. Valendo ressaltar que as acurácias dos modelos propostos, ficaram próximas a padrão, acurácia do LSSVM. Na figura 14, observa-se os gráficos da acurácias para ambas as bases, nestes os modelos obtiveram, taxas próximas e até maiores do que o IP-LSSVM e o P-LSSVM, em alguns pontos.

Tabela 8 – Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema da diabetes.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
PID	SVM	$77,35 \pm 3,0$	614	332	–
PID	LSSVM	$76,55 \pm 2,5$	614	614	–
PID	IP-LSSVM	$70,97 \pm 3,9$	614	491	20
		$72,79 \pm 3,2$		307	50
		$68,02 \pm 3,0$		123	80
PID	FSGAS-LSSVM	$77,5 \pm 3,7$	614	491	20
		$76,4 \pm 3,3$		307	50
		$76,6 \pm 3,1$		123	80
PID	FSLM-LSSVM	$75,45 \pm 3,2$	614	491	20
		$75,88 \pm 2,8$		307	50
		$74,06 \pm 4,2$		123	80
PID	RFSLM-LSSVM	$76,56 \pm 3,10$	614	491	20
		$75,47 \pm 3,33$		307	50
		$75,83 \pm 3,15$		123	80

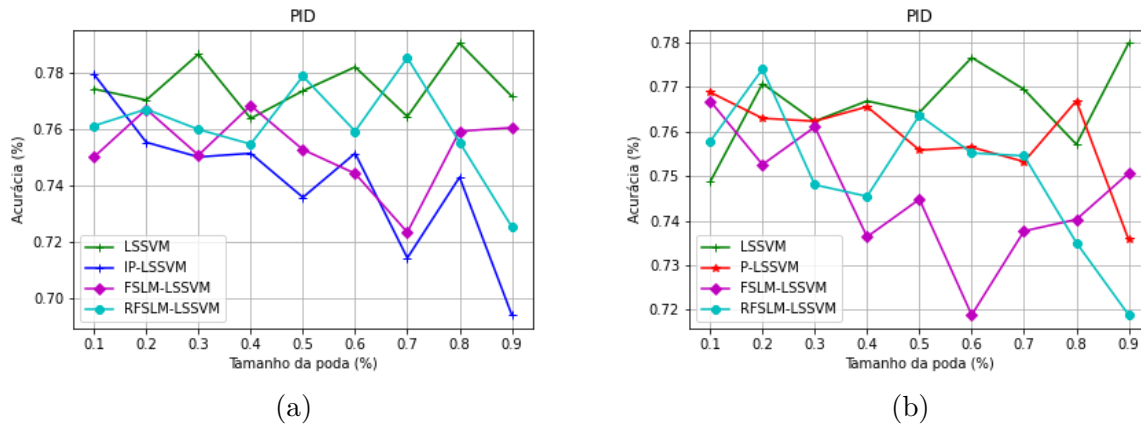
Fonte: Autor.

Tabela 9 – Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema da diabetes.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
PID	SVM	$76,95 \pm 2,3$	461	312	–
PID	LSSVM	$77,45 \pm 2,0$	461	461	–
PID	P-LSSVM	$76,41 \pm 3,6$	461	369	20
		$75,88 \pm 2,8$		230	50
		$75,74 \pm 2,4$		92	80
PID	FSGAS-LSSVM	$76,8 \pm 2,4$	461	369	20
		$77,1 \pm 3,0$		230	50
		$76,9 \pm 3,0$		92	80
PID	FSLM-LSSVM	$76,69 \pm 3,2$	461	369	20
		$75,48 \pm 3,5$		230	50
		$74,20 \pm 3,5$		92	80
PID	RFSLM-LSSVM	$76,88 \pm 3,81$	461	369	20
		$75,77 \pm 1,94$		230	50
		$74,54 \pm 3,68$		92	80

Fonte: Autor.

Figura 13 – Acurácia *versus* tamanho da poda para o conjunto das diabetes. (a) Metodologia 80/20. (b) Metodologia 60/20/20



Fonte: Autor.

Tabela 10 – Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema de haberman.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
HAB	SVM	73,79 ± 4,1	245	60	—
HAB	LSSVM	73,63 ± 3,5	244	245	—
HAB	IP-LSSVM	72,03 ± 7,8	245	196	20
		70,79 ± 5,8		122	50
		64,23 ± 9,4		49	80
HAB	FSGAS-LSSVM	74,3 ± 4,5	245	196	20
		75,4 ± 6,2		122	50
		74,4 ± 4,8		49	80
HAB	FSLM-LSSVM	73,87 ± 5,6	245	196	20
		75,24 ± 3,8		122	50
		73,63 ± 4,4		49	80
HAB	RFSLM-LSSVM	75,27 ± 2,09	245	196	20
		71,88 ± 5,02		122	50
		72,19 ± 6,58		49	80

Fonte: Autor.

Tabela 11 – Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema de haberman.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
HAB	SVM	$74,73 \pm 1,7$	184	128	–
HAB	LSSVM	$75,05 \pm 3,4$	184	184	–
HAB	P-LSSVM	$74,57 \pm 3,6$	184	147	20
		$73,71 \pm 5,9$		92	50
		$71,94 \pm 4,6$		37	80
HAB	FSGAS-LSSVM	$72,5 \pm 5,8$	184	147	20
		$75,0 \pm 7,4$		92	50
		$73,7 \pm 5,0$		37	80
HAB	FSLM-LSSVM	$74,73 \pm 6,1$	184	147	20
		$73,71 \pm 5,5$		92	50
		$74,20 \pm 3,5$		37	80
HAB	RFSLM-LSSVM	$73,87 \pm 2,81$	184	147	20
		$75,00 \pm 2,72$		92	50
		$73,38 \pm 2,07$		37	80

Fonte: Autor.

Tabela 12 – Resultados para os classificadores treinados de acordo com a metodologia 80/20 utilizando kernel linear, para o problema ripley.

Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
RIP	SVM	$88,0 \pm 1,9$	1000	333	–
RIP	LSSVM	$87,52 \pm 1,1$	1000	1000	–
RIP	IP-LSSVM	$87,76 \pm 2,5$	1000	800	20
		$87,84 \pm 2,1$		500	50
		$88,20 \pm 1,5$		200	80
RIP	FSGAS-LSSVM	$87,8 \pm 2,2$	1000	800	20
		$87,7 \pm 1,9$		500	50
		$87,9 \pm 2,0$		200	80
RIP	FSLM-LSSVM	$88,84 \pm 1,5$	1000	800	20
		$88,90 \pm 1,4$		500	50
		$88,10 \pm 1,6$		200	80
RIP	RFSLM-LSSVM	$88,99 \pm 1,73$	1000	800	20
		$87,53 \pm 2,00$		500	50
		$88,35 \pm 1,85$		200	80

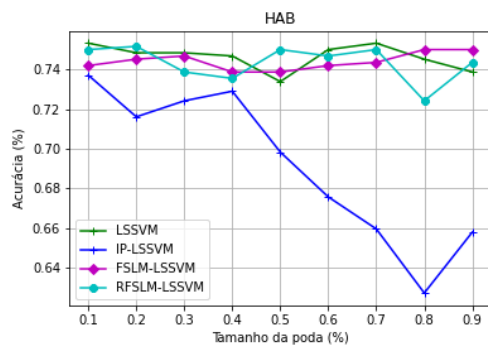
Fonte: Autor.

Tabela 13 – Resultados para os classificadores treinados de acordo com a metodologia 60/20/20 utilizando kernel linear, para o problema ripley.

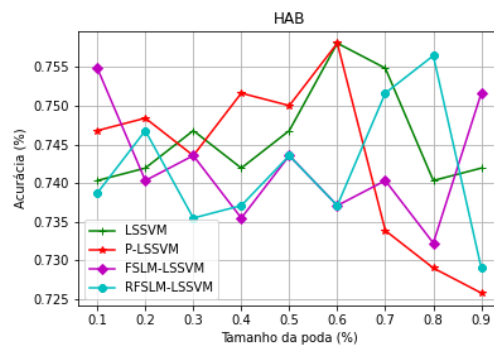
Base	Classificador	Acurácia (%)	#TS	#VS	Red.(%)
RIP	SVM	$88,21 \pm 1,5$	750	315	–
RIP	LSSVM	$88,33 \pm 1,4$	750	750	–
RIP	P-LSSVM	$88,01 \pm 2,0$	750	600	20
		$88,35 \pm 1,7$		375	50
		$87,84 \pm 1,8$		150	80
RIP	FSGAS-LSSVM	$88,2 \pm 1,5$	750	600	20
		$87,9 \pm 1,9$		375	50
		$87,7 \pm 1,7$		150	80
RIP	FSLM-LSSVM	$88,02 \pm 1,6$	750	600	20
		$88,32 \pm 1,8$		375	50
		$87,85 \pm 1,6$		150	80
RIP	RFSLM-LSSVM	$87,44 \pm 1,69$	750	600	20
		$87,75 \pm 1,61$		375	50
		$87,18 \pm 1,37$		150	80

Fonte: Autor.

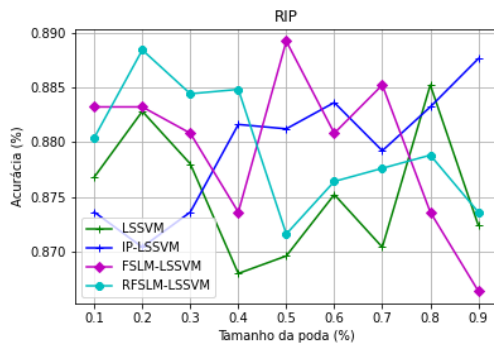
Figura 14 – Acurácia *versus* tamanho da poda para o conjunto de dados de Haberman e Ripley. (a) Haberman com metodologia 80/20. (b) Haberman com metodologia 60/20/20. (c) Ripley com metodologia 80/20. (d) Ripley com metodologia 60/20/20.



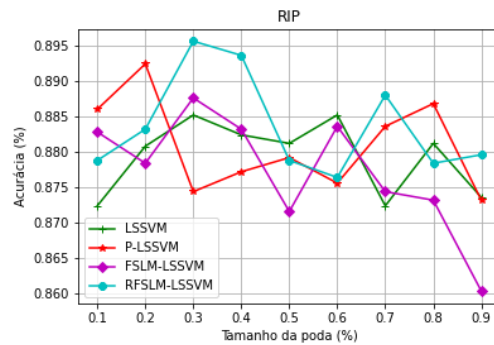
(a)



(b)



(c)



(d)

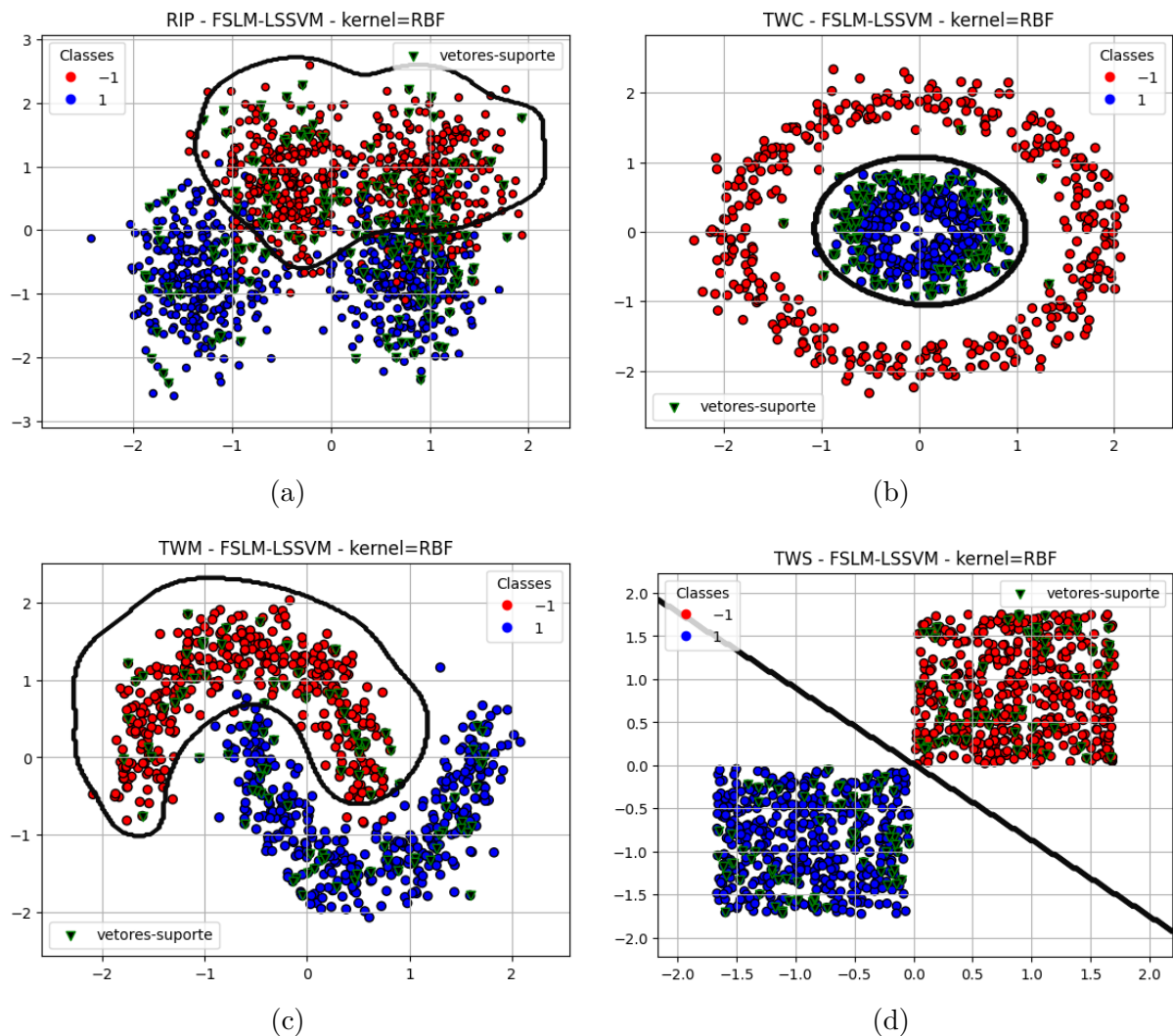
Fonte: Autor.

4.3 Análise dos vetores-suporte

Para visualizar e analisar como os vetores-suporte estão dispersos nas soluções das duas propostas apresentadas, foram aplicadas quatro bases de dados, cada uma com duas dimensões, onde três bases são artificiais e uma base tem dados reais (Ripley). Aqui segue-se com o mesmo fluxo metodológico, figura 10, para a visualização, neste caso foi aplicada a 60/20/20, pois foi necessário realizar uma validação cruzada (*k-fold cross validation*), para encontrar os hiperparâmetros do kernel RBF.

Na figura 15, estão os resultados para as quatro bases de dados. Cada gráfico, contém os vetores-suporte encontrados pelo modelo FSLM-LSSVM com 80 % de corte. Em todos os resultados, constatou-se que as superfícies de decisão geradas ficaram bem projetadas, ou seja de acordo com os devidos vetores-suporte.

Figura 15 – Visualização 2D dos vetores-suporte em quatro bases de dados diferentes aplicando o modelo FSLM-LSSVM(Proposta 1) com 80 % de poda, aplicando Kernel RBF. (a) Conjunto de dados Ripley. (b) Conjunto de dados artificial TWC. (c) Conjunto Artificial TWM. (d) Conjunto Artificial TWS.

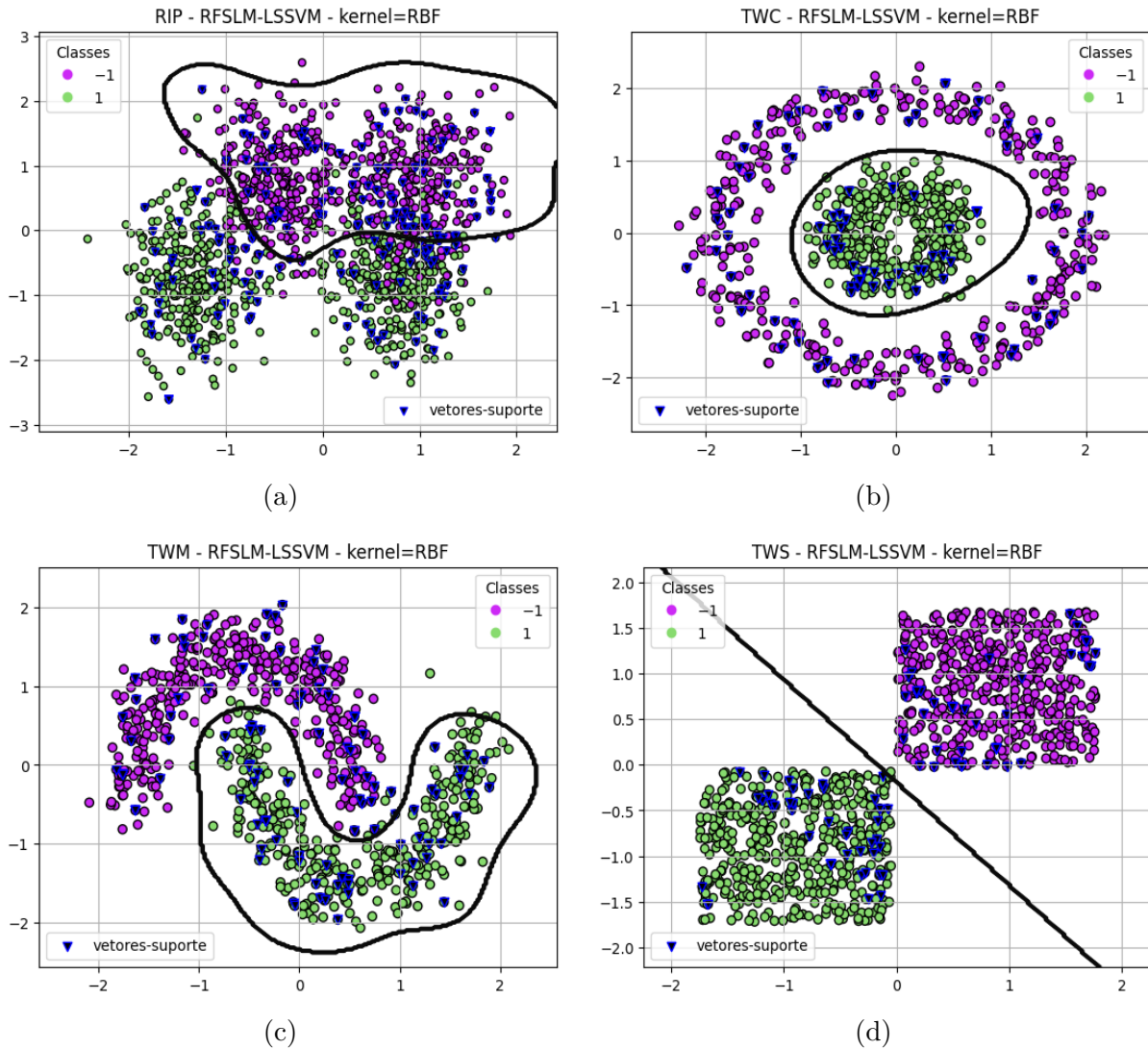


Fonte: Autor.

O gráfico em que a superfície de decisão é mais perceptível, é o da figura 15b. Nessa, os vetores-suporte (pontos pretos-verdes) ficaram mais próximos da borda do círculo, na classe +1 (pontos azuis). Seguindo, temos o problema linearmente separável, figura 15d, nesse assim como no *TWC*, os vetores-suporte ficaram próximos a superfície de decisão. Entretanto, alguns ficaram um pouco mais longe da reta.

Finalizando a análise com FSLM-LSSVM, temos um conjunto de dados real e um artificial, ambos não linearmente separáveis. Os vetores-suporte encontrados para o problema Ripley e conjunto artificial, figura 15a e 15c, mostraram que os vetores-suporte ficaram mais dispersos.

Figura 16 – Visualização 2D dos vetores-suporte em quatro bases de dados diferentes aplicando o modelo RFSLM-LSSVM(Proposta 2) com 80 % de poda, aplicando Kernel RBF. (a) Conjunto de dados Ripley. (b) Conjunto de dados artificial TWC. (c) Conjunto Artificial TWM. (d) Conjunto Artificial TWS.



Fonte: Autor.

A segunda fase desta avaliação envolveu a segunda proposta, o algoritmo RFSLM-

LSSVM. Assim como no FSLM-LSSVM, a metodologia de divisão de dados 60/20/20 foi aplicada. Inicialmente, em uma rápida análise dos gráficos, representados na Figura 16, revela-se que as distribuições dos vetores-suporte se assemelham às observadas no FSLM-LSSVM. Contudo, existe algumas particularidades notáveis. Na figura 16b, os vetores-suporte (pontos preto-azuis) ficaram concentrados na borda da classe +1 (pontos verdes), mais especificamente concentrados em uma parte da borda. Ainda na figura 16b notou-se que uma quantidade razoável de vetores-suporte ficaram distribuídos na classe -1 (pontos roxos), ao contrario do resultado visto no FSLM-LSSVM.

Prosseguindo, na figura 16d, esta o problema linearmente separável. Neste, os vetores-suporte ficaram mais próximos da superfície de decisão. Já os últimos dois conjuntos, figuras 16a e 16c, apresentaram soluções muito parecidas com as do FSLM-LSSVM. Concluindo, observou-se que as soluções encontradas pelos modelos propostos, vetores-suporte, ficaram bem distribuídos, com superfícies de decisão bem projetadas.

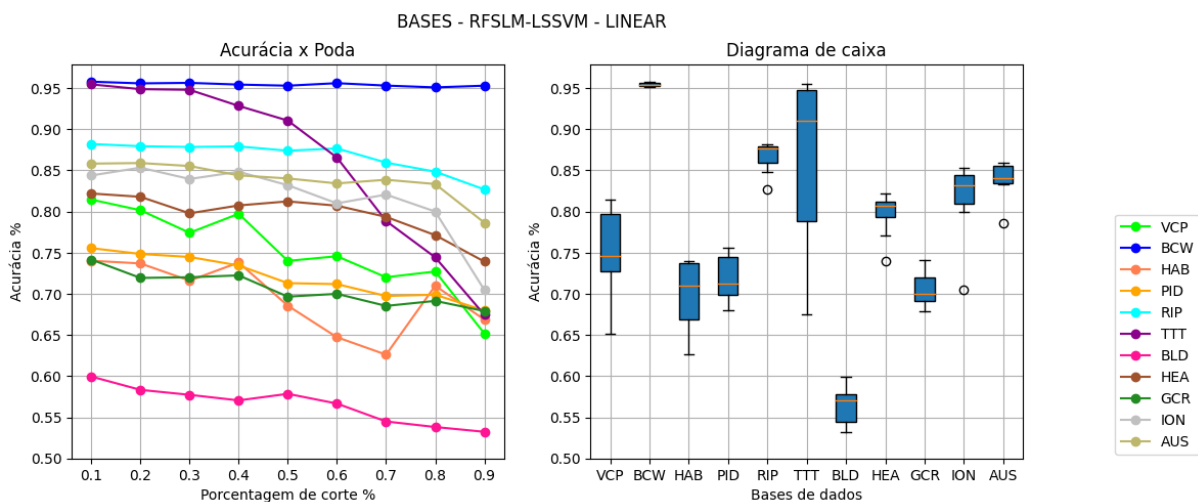
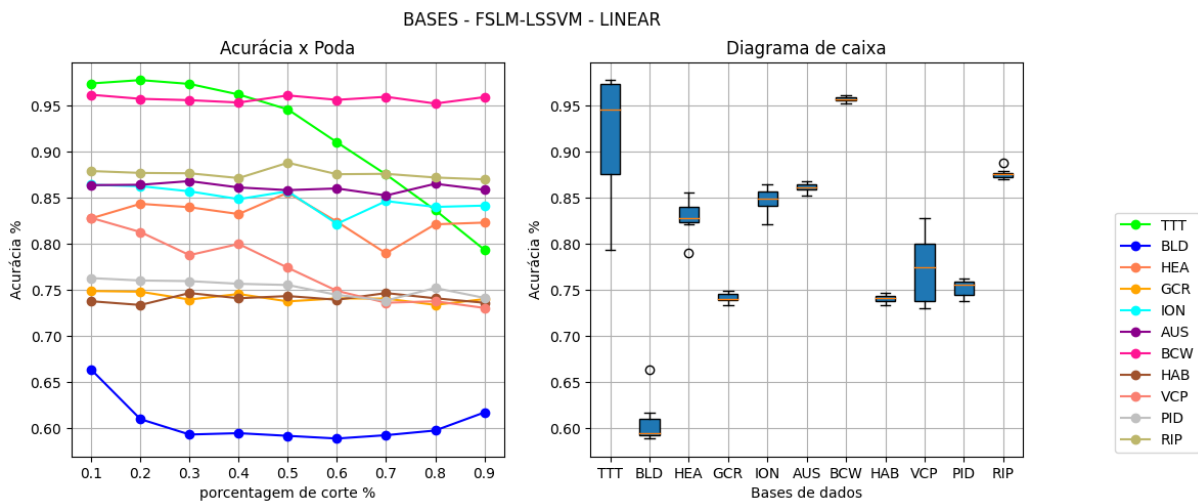
4.4 Análise geral de desempenho dos modelos propostos

Para finalizar as simulações computacionais, foram utilizadas todas as 11 bases de dados reais, tabela 1. Seguiu-se a mesma metodologia já discutida, com uma divisão dos dados em 80/20. O kernel aplicado nesta foi o Linear. Em cada conjunto, foi criado um gráfico da acurácia pela porcentagem de corte, no qual podem ser visualizados na figura 17. Ao lado destes, foi criado um diagrama de caixa, para a análise da variabilidade de cada conjunto ao decorrer das porcentagens de corte.

Na primeira, figura 17a, foi utilizado o algoritmo FSLM-LSSVM. À direita, visualiza-se o gráfico das acurácias pelas porcentagens de corte. Neste, a maioria das bases de dados se manteve com acurácias similares. No qual, com os diagramas de caixa, figura 17a, fica mais perceptível. Entretanto, em algumas bases, como a TTT, houve uma queda na taxa de acerto. Por fim, uma grande parte das acurácias ficaram no intervalo de 75 % a 95 %.

Prosseguindo, a análise da segunda proposta, figura 17b. Nesta houve uma queda maior das taxas de acerto ao decorrer dos cortes, contudo a maioria das acurácias ficou no mesmo intervalo anterior de 75 % a 95 %. Concluindo, os dois modelos apresentaram resultados satisfatórios em todas as bases de dados, mostrando-se uma boa alternativa aos modelos da literatura já existentes.

Figura 17 – Análise de todas as bases de dados de acordo com a porcentagem de corte. (a) Acurácia x porcentagem de corte (Direita) e Diagrama de caixa de cada base (Esquerda), aplicando modelo FSLM-LSSVM com kernel Linear. (b) Acurácia x porcentagem de corte (Direita) e Diagrama de caixa de cada base (Esquerda), aplicando modelo RFSLM-LSSVM com kernel Linear.



Fonte: Autor.

5 Conclusões

Neste trabalho, foram investigadas técnicas da literatura com o objetivo principal de construir soluções esparsas para o modelo LSSVM. No capítulo 2.5, observou-se que as soluções obtidas por meio desse algoritmo não são esparsas, o que compromete a etapa de classificação de novos padrões. Na literatura, capítulo 3, os modelos IP-LSSVM e P-LSSVM utilizam as propriedades intrínsecas do próprio algoritmo, analisando os multiplicadores de Lagrange para alcançar a esparsidade desejada. Por outro lado, o modelo FSGAS-LSSVM emprega algoritmos genéticos, modelando assim uma solução iterativa.

Com a finalidade de construir soluções esparsas para o LSSVM, foi utilizada uma abordagem semelhante à análise dos vetores-suporte com base nos multiplicadores de Lagrange. A partir disso, foram propostos dois algoritmos para geração de soluções esparsas com o LSSVM, conforme detalhado no capítulo 3.5. Eles foram nomeados de *Fixed Size Levenberg Marquardt - LSSVM* (FSLM-LSSVM) e *Reduced Fixed set Levenberg Marquardt - LSSVM* (RFSLM-LSSVM).

Os dois modelos são de tamanho fixo, o que significa que eles removem uma quantidade predefinida de vetores-suporte. Essa quantidade é um parâmetro que o usuário pode especificar. A característica de soluções de tamanho fixo é vantajosa em algumas abordagens devido à sua capacidade de simplificar o desenvolvimento. Isso ocorre porque os usuários podem projetar uma solução que requer, por exemplo, o mínimo possível de espaço em disco para armazenar o modelo final.

Adicionalmente, ambas as propostas operam de maneira iterativa, eliminando vetores-suporte durante o processo de treinamento. Essa característica também é observada no classificador FSGAS-LSSVM. Ao abordar a remoção de vetores-suporte dessa maneira, elimina-se a necessidade de um estágio de pós-treinamento ao final do processo.

Na primeira proposta, o FSLM-LSSVM, opera com uma matriz $\mathbf{A}$ completa durante os cálculos matriciais. Em outras palavras, trata-se de uma matriz de dimensões $N \times N$, onde N representa o número de padrões. Utilizar uma matriz completa implica em cálculos matriciais mais complexos e, frequentemente, mais demorados. Com o intuito de mitigar essa complexidade durante o treinamento, foi desenvolvido o segundo classificador, o RFSLM-LSSVM. A segunda proposta trabalha com uma matriz $\mathbf{A}$ em escala reduzida, reduzindo assim a complexidade computacional.

Finalmente, nas simulações computacionais, foi avaliada a eficácia das propostas, utilizando a métrica de acurácia. Resumindo, os modelos demonstraram resultados satisfatórios, aproximando-se dos resultados encontrados na literatura existente. Isso os posiciona como alternativas eficazes para solucionar o problema da perda de esparsidade no LSSVM.

Referências

- BURBIDGE, R.; BUXTON, B. An introduction to support vector machines for data mining. 2001. Disponível em: <<https://api.semanticscholar.org/CorpusID:8133449>>. Citado 2 vezes nas páginas 25 e 29.
- BURGES, C. Simplified support vector decision rules. In: . [S.l.: s.n.], 1996. p. 71–77. Citado 2 vezes nas páginas 25 e 45.
- CARVALHO, B.; BRAGA, A. IP-LSSVM: A two-step sparse classifier. *Pattern Recognition Letters*, v. 30, n. 16, p. 1507–1515, 2009. ISSN 0167-8655. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0167865509002086>>. Citado 3 vezes nas páginas 25, 46 e 49.
- CORTES, C.; VAPNIK, V. Support vector networks. *Machine Learning*, v. 20, p. 273–297, 1995. Citado 3 vezes nas páginas 25, 29 e 35.
- CRISTIANINI, N.; SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 1. ed. Cambridge University Press, 2000. ISBN 0521780195. Disponível em: <http://www.amazon.com/Introduction-Support-Machines-Kernel-based-Learning/dp/0521780195/ref=sr_1_1?ie=UTF8&s=books&qid=1280243230&sr=8-1>. Citado na página 30.
- FLETCHER, R. *Practical methods of optimization*. Place of publication not identified: Wiley, 1987. ISBN 1-118-72320-1. Citado na página 38.
- LEVENBERG, K. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, Brown University, v. 2, n. 2, p. 164–168, 1944. ISSN 0033569X, 15524485. Disponível em: <<http://www.jstor.org/stable/43633451>>. Citado na página 41.
- MARQUARDT, D. W. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, Society for Industrial and Applied Mathematics, v. 11, n. 2, p. 431–441, 1963. ISSN 03684245. Disponível em: <<http://www.jstor.org/stable/2098941>>. Citado na página 41.
- MOORE, E. H. On the reciprocal of the general algebraic matrix. p. 394–395, 1920. Citado na página 40.
- NETO, A. R.; SILVA, D. A. A genetic algorithms-based lssvm classifier for fixed-size set of support vectors. In: . [S.l.: s.n.], 2015. v. 9095. ISBN 978-3-319-19221-5. Citado na página 47.
- NETO, A. R. R. *SINPATCO II: NOVAS ESTRATÉGIAS DE APRENDIZADO DE MÁQUINA PARA CLASSIFICAÇÃO DE PATOLOGIAS DA COLUNA VERTEBRAL*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO CEARÁ, 2011. Citado 7 vezes nas páginas 30, 36, 40, 42, 45, 46 e 48.
- PENROSE, R. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, Cambridge University Press (CUP), v. 51, n. 3, p.

406–413, jul. 1955. Disponível em: <<https://doi.org/10.1017/s0305004100030401>>. Citado na página 40.

RANGANATHAN, A. The levenberg-marquardt algorithm. *Academia.edu*, 07 2004. Citado na página 40.

SUYKENS, J.; LUKAS, L.; VANDEWALLE, J. Sparse least squares support vector machine classifiers. In: . [S.l.: s.n.], 2000. v. 1, p. 37–42. Citado 2 vezes nas páginas 25 e 45.

SUYKENS, J.; VANDEWALLE, J. Least squares support vector machine classifiers. *Neural Processing Letters*, Springer Science and Business Media LLC, v. 9, n. 3, p. 293–300, 1999. Disponível em: <<https://doi.org/10.1023/a:1018628609742>>. Citado 2 vezes nas páginas 25 e 38.