UNIVERSIDADE FEDERAL DE JUIZ DE FORA
PROGRAMA DE PÓS GRADUAÇÃO EM MODELAGEM COMPUTACIONAL

Renan Motta Goulart

Uma abordagem geométrica para o estudo do problema de classificação binária

Juiz de Fora
2024

Renan Motta Goulart

Uma abordagem geométrica para o estudo do problema de classificação binária

Tese apresentada ao Programa de Pós Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Doutor em Modelagem Computacional.

Orientador: Prof. Dr. Raul Fonseca Neto

Coorientador: Prof. Dr. Carlos Cristiano Hasenclever Borges

Coorientador: Prof. Dr. Saulo Moraes Villela

Juiz de Fora

2024

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Goulart, Renan Motta.
Uma abordagem geométrica para o estudo do problema de classificação binária / Renan Motta Goulart. -- 2024.
130 f. : il.

Orientador: Raul Fonseca Neto
Coorientadores: Carlos Cristiano Hasenclever Borges, Saulo Moraes Villela
Tese (doutorado) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Modelagem Computacional, 2024.

1. Aprendizado de Máquina. 2. Classificação Binária de Larga Margem. 3. Análise de Separabilidade. 4. Análise de Relevância. 5. Computação Evolucionista. I. Fonseca Neto, Raul, orient. II. Borges, Carlos Cristiano Hasenclever , coorient. III. Villela, Saulo Moraes, coorient. IV. Título.

Renan Motta Goulart

Uma abordagem geométrica para o estudo do problema de classificação binária

Tese apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Doutor em Modelagem Computacional. Área de concentração: Modelagem Computacional.

Aprovada em 05 de setembro de 2024.

BANCA EXAMINADORA

Prof. Dr. Raul Fonseca Neto - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Carlos Cristiano Hasenclever Borges - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Saulo Moraes Villela - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Wilhelm Passarella Freire

PROPP 01.5: Termo de aprovação COORD-PPG-MODELAGEM-COMPUTACIO 1941225 SEI 23071.927083/2024-23 / pg. 1

Universidade Federal de Juiz de Fora

Prof. Dr. Leonardo Goliatt da Fonseca
Universidade Federal de Juiz de Fora

Prof. Dr. Antônio de Pádua Braga
Universidade Federal de Minas Gerais

Prof. Dr. Luiz Carlos Bambirra Torres
Universidade Federal de Ouro Preto

Juiz de Fora, 22/08/2024.

Documento assinado eletronicamente por **Raul Fonseca Neto**, **Professor(a)**, em 06/09/2024, às 10:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Carlos Cristiano Hasenclever Borges**, **Professor(a)**, em 06/09/2024, às 10:07, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Carlos Bambirra Torres**, **Usuário Externo**, em 06/09/2024, às 13:29, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Antônio de Pádua Braga**, **Usuário Externo**, em 08/09/2024, às 07:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leonardo Goliatt da Fonseca**, **Professor(a)**, em 09/09/2024, às 11:27, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Wilhelm Passarella Freire**, **Professor(a)**, em 01/10/2024, às 11:28, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Saulo Moraes Villela**, **Professor(a)**, em 01/10/2024, às 17:22, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1941225** e o código CRC **9D670175**.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

"Complex theories do not work, simple algorithms do."
Vladimir Vapnik.

"Geometry is illuminating, probability theory is powerful."
Pál Ruján.

RESUMO

Classificação binária é um problema essencial na área de Aprendizado de Máquina, na qual encontrar classificadores com um alto nível de acurácia e poder de generalização na predição de novos dados é um de seus principais objetivos. Esta pesquisa estudou o problema de classificação binária segundo uma perspectiva geométrica, produzindo novos resultados segundo duas abordagens. A primeira abordagem está relacionada ao uso da Computação Evolucionista e de um sistema esférico de representação de coordenadas resultando no desenvolvimento de dois algoritmos evolutivos: o primeiro sendo um classificador de larga margem; e o segundo um classificador que aproxima o cálculo do centro analítico. A segunda abordagem envolveu conceitos de Geometria Computacional produzindo um método eficiente, e que termina em tempo finito, para determinar quais instâncias de um problema são relevantes. Para tanto, foi feita uma redução deste problema ao problema de determinar se um ponto é extremo em uma nuvem de pontos. Como consequência deste estudo desenvolveu-se outra técnica eficiente, e que também termina em tempo finito, para determinar se um problema de classificação binária apresenta separabilidade linear no espaço de entrada ou no espaço de características para uma função kernel escolhida. É importante destacar que todos os estudos foram realizados tanto no espaço primal quanto no espaço dual possibilitando o uso de um mapeamento implícito. Os resultados obtidos foram bastante satisfatórios e muitas vezes superiores a outros métodos que representam o estado da arte do problema de classificação binária.

Palavras-chave: Algoritmo Evolucionista. Análise de Relevância. Análise de Separabilidade. Centro Analítico. Classificação Binária. Coordenadas Esféricas. Larga Margem. Ponto Extremo.

ABSTRACT

Binary Classification is an essential task in the field of Machine Learning, where finding a classifier with high accuracy and generalization performance is one of its central problems. This research studies the binary classification problem from a geometric perspective, producing new insights through two main approaches. The first approach is related to the development of evolutionary algorithms and the use of the spherical coordinate system. This part of the research resulted in two evolutionary algorithms: the first, a large-margin classifier, and the second, a classifier that approximates the analytical center. The second approach, based on Computational Geometry, produced an efficient method, which requires finite time, to analyze the relevance of instances in a binary classification problem. This is achieved by reducing the problem to determining if a point is an extreme point of a convex cloud. As a consequence of this reduction, an efficient technique, also capable of being executed in finite time, was developed to determine if a binary classification problem is linearly separable in the input or feature space for a chosen kernel function. It is important to highlight that both the primal and dual representation spaces were considered during the development of these methods, enabling the use of an implicit mapping. The obtained results are competitive with other methods that represent the current state of the art for binary classification.

Keywords: Analytic Center. Binary Classification. Evolutionary Algorithm. Extreme Point. Large Margin. Relevance Analysis. Separability Analysis. Spherical Coordinates.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um dos principais pilares da área de Aprendizado de Máquinas é o problema de classificação binária, onde a partir de um conjunto de pares, chamados de amostras, cada uma composta por um vetor de características, denominado instância, e um rótulo associado, tem-se como objetivo inferir ou mapear corretamente o rótulo de um novo dado, conhecendo apenas o seu vetor de características. Esta formulação é utilizada para modelar diversos problemas em várias áreas do conhecimento como exemplo: na medicina para detecção e prevenção de doenças [84, 83, 75], na economia para triagem de créditos [79], na política para prever votações do congresso [81] e na sociologia para previsão de divórcios [76]. Portanto, encontrar classificadores eficientes é um tópico de alta relevância não somente para a desenvolvimento teórico da área de Aprendizado de Máquinas como também para o desenvolvimento e utilidade de suas aplicações. Apesar de existirem inúmeros algoritmos voltados para geração de classificadores, explorando diferentes técnicas e paradigmas, observa-se que ainda existe espaço para o desenvolvimento de métodos mais eficientes tanto em relação ao esforço computacional quanto à qualidade da predição e que explorem aspectos geométricos relacionados ao paradigma de maximização da margem e da aproximação do ponto de Bayes.

Um dos problemas críticos enfrentados pelos métodos mais utilizados é possuírem um custo computacional de tempo quadrático em relação a quantidade de amostras do conjunto de dados. Neste sentido, desenvolver métodos que tenham um custo de tempo linear e gerem resultados eficientes é um problema de interesse. Portanto, a motivação inicial para esta pesquisa foi encontrar meios de realizar o processo de classificação de modo mais eficiente. Esta tese tem como objetivo principal apresentar os estudos realizados e os resultados obtidos, dentre eles: o aprofundamento de conceitos teóricos referentes ao problema de classificação binária; o desenvolvimento de dois classificadores que utilizam computação evolucionista e um sistema esférico de representação; a análise de quais instâncias são relevantes segundo a geometria do espaço de versões; a verificação eficiente se um determinado ponto é extremo, e por fim a análise de separabilidade linear. Esses estudos proporcionaram a obtenção de conhecimento avançado relacionado ao tema, bem como a geração de conhecimento científico relevante, sendo ele composto de novas ideias e métodos para solucionar este problema de classificação. Os resultados obtidos provaram-se competitivos em relação as outras técnicas utilizadas, assim demonstrando a relevância do conhecimento produzido.

A pesquisa pode ser dividida em duas partes. A primeira parte limitou-se ao caso mais simples de classificação binária relacionado à solução de problemas com conjuntos de dados linearmente separáveis. Consequentemente, para a solução deste problema foram desenvolvidos dois algoritmos com implementação primal. Após uma análise dos métodos de referência foi constatado que houveram poucas tentativas no uso de

computação evolucionista para o desenvolvimento de algoritmos de classificação, o que abriu possibilidades para explorar a criação de novos métodos para a solução do problema de classificação binária. A partir disto, dois algoritmos evolucionistas foram criados, o *Algoritmo Evolucionista de Larga Margem* (AELM) que será apresentado no capítulo 4, e o *Algoritmo Evolucionista de Centtro Analítico* (AECA) que será introduzido no capítulo 5. Entretanto, a implementação dual destes algoritmos, necessária para a solução de casos não linearmente separáveis, não se mostrou eficiente. Como consequência, os estudos foram redirecionados no sentido de mitigar este problema, gerando a segunda parte da pesquisa. Nesta segunda abordagem, focou-se no estudo de métodos capazes de remover as instâncias não relevantes do problema sem que isto afete negativamente a qualidade da solução obtida. Esta forma de pré-processamento seria de grande interesse não somente para os classificadores desenvolvidos neste trabalho, mas também para qualquer método cuja complexidade esteja relacionada à quantidade de instâncias no conjunto de dados.

Ao estudar este problema, percebeu-se uma forte relação entre as áreas de Aprendizado de Máquinas, Álgebra Linear e Geometria Computacional. Mais especificamente, foi descoberta uma relação entre o problema de encontrar uma solução factível para o problema de classificação binária, representado por um sistema de inequações lineares, e a determinação de pontos extremos. A partir deste resultado vários desdobramentos ocorreram, dentre os quais, a criação de um método capaz de determinar em tempo finito se uma determinada instância, representada por um vetor de características no espaço de entrada, é relevante para a geometria do espaço de versões. Também, foi possível estender esta análise para a formulação dual, no contexto do espaço de características, possibilitando a escolha de uma função kernel apropriada, de menor complexidade, capaz de gerar uma solução factível para o respectivo problema.

Os conceitos teóricos abordados foram referentes ao problema de interesse, classificação binária, os quais serão apresentados ao longo desta tese devido a sua importância para o entendimento do funcionamento dos métodos desenvolvidos. Primeiramente, será definido o problema de classificação binária, e em seguida, serão apresentadas as métricas utilizadas para prever a eficácia dos algoritmos desenvolvidos. Em seguida, será apresentado o conceito de espaço de versões, caracterizado por sua forte conexão com a área de Geometria Computacional, e sendo também fundamental para o entendimento dos algoritmos aqui apresentados.

Após a fundamentação teórica, serão apresentadas as contribuições desta tese, estas divididas em duas linhas de pesquisa: a primeira teve como objetivo gerar métodos eficientes que possuam um tempo de execução linear por meio de computação evolucionista, e a segunda como objetivo a análise da relevância de instâncias e a análise da separabilidade de um conjunto de dados por meio de uma perspectiva baseada em geometria computacional. No total este trabalho apresenta oito contribuições descritas nos parágrafos a seguir, sendo as três primeiras contribuições relacionadas a primeira linha de pesquisa, e as outras cinco

relacionadas a segunda.

A primeira contribuição deste trabalho é a proposta da utilização de coordenadas hiperesféricas em problemas de classificação. Este sistema de coordenadas apresenta vantagens em relação ao tradicional sistema Cartesiano por facilmente representar pontos na superfície de uma hiper-esfera, formato associado ao espaço de versões. Para a utilização deste sistema de coordenadas foi necessário desenvolver dois algoritmos, um para converter coordenadas Cartesianas para hiper-esféricas, e outro para o mapeamento inverso também. Estes algoritmos possuem tempo linear de execução e são simples de serem implementados. Este sistema de coordenadas foi fundamental, provendo diversas vantagens, como uma distribuição uniforme para os operadores de recombinação e mutação, além de garantir a norma unitária para os indivíduos.

A segunda contribuição se refere ao desenvolvimento de um método para classificação binária, o *Algoritmos Evolucionista de Larga Margem (AELM)* [56], que utiliza operadores específicos de recombinação e mutação visando evoluir a população para otimizar o valor de margem de seu melhor indivíduo. Estes operadores são aplicados diretamente no vetor de pesos de seus indivíduos.

A terceira contribuição deste trabalho é o desenvolvimento de um segundo método, o *Algoritmo Evolucionista de Centro Analítico (EACA)* [57], o qual se baseia na evolução de uma população de classificadores para otimizar o valor de centro analítico associado a base em questão de seu melhor indivíduo. Seu funcionamento geral é similar ao *EALM*, a menos de algumas mudanças. Devido a função objetivo deste algoritmo ser o centro analítico, a técnica de balanceamento de viés não pode ser utilizada. Para contornar esta restrição foi necessário gerar uma nova técnica de balanceamento, baseada em busca binária, especifica para o problema do centro analítico.

Apesar dos resultados promissores para a formulação primal, os algoritmos evolucionistas pecaram quando aplicados na formulação dual, não se mostrando eficientes para problemas não lineares onde é necessário a utilização do *kernel trick*. Este problema foi abordado em profundidade em nossos estudos, levando a uma mudança na direção da pesquisa a qual passou a abordar dois problemas importantes. Primeiro relacionado a análise de relevância de amostras, e o segundo a análise de separabilidade. Como resultado destes novos estudos, mais cinco contribuições surgiram.

A quarta contribuição desta tese é a redução do problema de classificação binária ao problema de detecção de pontos extremos. Sendo esta a primeira contribuição da segunda linha de pesquisa, relacionada a geometria computacional, e a base teórica para a criação das contribuições seguintes.

A quinta é o desenvolvimento de um algoritmo para resolver um problema da área de computação geométrica, a detecção de pontos extremos em uma nuvem de pontos. Apesar de já existirem algoritmos eficientes para duas e três dimensões, o algoritmo

desenvolvido se destaca por ser eficiente em altas dimensões.

A sexta contribuição desta tese é referente ao problema de remoção de análise de relevância de instâncias. Uma instância considerada como não relevante pode ser removida do conjunto de dados, o que acarreta em um aumento da velocidade de execução de algoritmos de classificação. sem que isto afete o conjunto de classificadores factíveis. Este problema é trabalhado por meio de métodos heurísticos, como por exemplo o Apagoge [24, 28] e o SVM-KM [15], que ou encontram uma solução aproximada ou não tem garantia de encontrar a solução ótima em tempo finito. Nesta tese foi desenvolvido um método, denominado *ADePE*, capaz de definir, em tempo finito, se uma instância é relevante a um problema de classificação binária. Este método é aplicável tanto na formulação primal quanto na formulação dual, onde ele pode ser particularmente útil, devido ao custo quadrático em relação ao quantidade de instâncias desta formulação.

A sétima contribuição é uma forma indireta de representar dados na formulação dual, onde cada instância é representada como um vetor de alphas. Esta forma de representação foi necessária para o funcionamento do *ADePE* na formulação dual. Além disto, esta forma de representação permite, por meio de combinações lineares, acesso a todos os pontos espaço implícito, sendo isto algo com alto potencial a ser explorado por trabalhos futuros.

A oitava, e última, contribuição desta tese é um método capaz de determinar, em tempo finito, se um problema de classificação binária é separável tanto no espaço de entrada quanto no espaço de características. Este método é aplicável tanto na formulação primal quanto na formulação dual. Uma aplicação para este método é determinar se um problema pode ser resolvido pela formulação primal ou se é requerido usar uma solução não linear. Além disto, ele também pode ser usado para determinar se um kernel menos complexo poderia ser usado, o que é útil para evitar casos de *overfitting*.

A organização do conteúdo desta tese em capítulos pode ser resumida a seguir, a saber:

- **Capítulo 2 – Fundamentação Teórica:** Neste capítulo, explica-se detalhadamente a teoria basilar do problema de classificação, teoria esta necessária para o entendimento dos algoritmos desenvolvidos. Também descreve-se os métodos que auxiliaram no desenvolvimento, dos modelos apresentados, assim como os métodos mais utilizados, sendo estes utilizados para comparação nas seções de resultados.
- **Capítulo 3 – Coordenadas Hiperesféricas:** Apresenta-se neste capítulo a motivação para a adoção de coordenadas hiperesféricas e é mostrada sua importância para o desenvolvimento dos algoritmos evolucionistas. É introduzido o conceito de coordenadas hiper-esféricas e sua importância para o problema de classificação.

- **Capítulo 4 – Algoritmo Evolucionista de Larga Margem:** Neste capítulo descreve-se o primeiro método desenvolvido bem como os experimentos realizados e resultados obtidos. Este método baseia-se na construção de um algoritmo evolucionista no espaço primal com a finalidade de obter uma solução aproximada da margem ótima, denominada de solução de larga margem.

- **Capítulo 5 – Algoritmo Evolucionista de Centro Analítico:** Neste capítulo descreve-se o segundo método desenvolvido bem como os experimentos realizados e resultados obtidos. Este método baseia-se na construção de um algoritmo de computação evolucionista no espaço primal, porém com o propósito de obter uma solução aproximada para o centro de massa do espaço de versões.

- **Capítulo 6 – Análise de Relevância:** Este capítulo apresenta a teoria desenvolvida a qual une as áreas de Aprendizado de Máquinas e Geometria Computacional. Apresenta uma contribuição teórica importante para o entendimento do funcionamento dos métodos apresentados nos próximos capítulos.

- **Capítulo 7 – Detecção de Pontos Extremos:** Este capítulo apresenta um algoritmo, eficiente em altas dimensões, para detectar se um ponto é extremo em uma nuvem de pontos. A partir do conhecimento teórico do capítulo anterior, este algoritmo é utilizado para eliminar as instâncias não relevantes de um conjunto de dados.

- **Capítulo 8 – Análise de Separabilidade:** Neste capítulo apresenta-se um método para determinar em tempo finito a separabilidade do problema de classificação binária. Ao ser aplicado no domínio dual, é possível determinar se uma dada função kernel torna linearmente separável um problema no respectivo espaço de características, possibilitando a escolha uma função kernel factível de menor complexidade.

- **Capítulo 9 – Conclusões:** Apresenta-se um resumo das principais limitações e contribuições da tese bem como uma discussão a respeito dos resultados obtidos. Também, sugere-se algumas possibilidades de trabalhos futuros no sentido de dar continuidade a este estudo, principalmente relacionados a segunda parte da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão introduzidos conceitos fundamentais para o entendimento deste trabalho. Primeiro será definido o problema de classificação binária junto com a definição do arquétipo de algoritmos usados para resolver este problema, os classificadores binários. Em seguida será definido o conceito de espaço de versões, necessário para entender o funcionamento de algoritmos encontrados na literatura e que também possibilita uma visão geométrica do problema de classificação binária, essencial para os algoritmos desenvolvidos neste trabalho. Por fim serão apresentados alguns métodos de classificação que exploram a geometria do espaço de versões e que são utilizados como métodos comparativos

2.1 CLASSIFICAÇÃO

Neste problema procura-se encontrar o rótulo de um nova instância baseado na similaridade com os membros de um conjunto de instâncias cujos rótulos são conhecidos. O conjunto de rótulos é comumente chamado de *classes*, sendo este o modo como será referido no restante deste trabalho. Para o problema de classificação binária, os rótulos são valores discretos. Neste trabalho, foi estudado em detalhes o caso de problemas de classificação com duas classes, chamados de *Problema de Classificação Binária*. É provado que problemas com mais de duas classes podem ser reduzidos a problemas binários [6] por meio de técnicas como a *One-Against-All* que consiste em treinar um classificador binário em que um rótulo é relacionado a uma classe, e as instâncias das outras classes são consideradas como o segundo rótulo. Deste modo, ao estudar problemas de classificação binária é possível aplicar o conhecimento gerado aos problemas com mais classes. A sub-seção a seguir apresentará em detalhes o problema de classificação binária.

2.1.1 Classificação binária

O problema binário pode ser definido formalmente como: Seja $\mathcal{X} \subseteq \mathbb{R}^d$ o conjunto de instâncias e $\mathcal{Y} = \{-1, 1\}$ o conjunto de rótulos relacionado às respectivas de classes. Denota-se $Z_m = \{X_i, Y_i\}_{i=1}^m$ como o conjunto de treinamento de cardinalidade m. Procura-se encontrar uma função de decisão $f : \mathcal{X} \to \mathcal{Y}$ que seja capaz de inferir corretamente as classes das instâncias que pertençam ao conjunto de treinamento, e que tenha bom desempenho ao inferir instâncias não pertencentes a ele. Considerando a existência de apenas duas classes, pode-se utilizar como discriminante um hiperplano d-dimensional, responsável por dividir o espaço em dois semi-espaços, sendo cada um relacionado a uma das classes. As instância que possuem rótulo -1 terão valor funcional negativo, e estarão localizadas dentro de um mesmo semi-espaço. Por outro lado, as instâncias de rótulo $+1$ terão valores funcionais positivos. A superfície desta função, um hiperplano, pode ser descrita pela seguinte equação:

$$\langle W, X_i \rangle + b = 0 \tag{2.1}$$

onde W é o vetor normal e b é o parâmetro de viés.

Deste modo, a função discriminante realiza uma inferência ao substituir em sua equação o argumento X_i pelo vetor característico representativo de uma instância. A regra seguida para realizar a inferência é: caso o valor retornado pela equação seja positivo, ou seja, a instância está acima do hiperplano, ela é classificada como pertencente a classe $+1$; caso contrário, esteja abaixo ou em cima do hiperplano, é classificada como pertencente a classe -1. Formalmente isto pode ser representado pela função de decisão descrita a seguir:

$$f(W, b, X_i) = \begin{cases} +1, \text{ Se } \langle W, X_i \rangle + b > 0 \\ -1, \text{ Caso contrário.} \end{cases} \tag{2.2}$$

onde $f(W, b, X_i)$ é a classe inferida pelo hiperplano W para a instância X_i.

Considerando um conjunto de instâncias $\mathcal{X}$ e o conjunto de rótulos $\mathcal{Y}$, um hiperplano é definido como factível para o conjunto $\mathcal{X}$ caso ele infira corretamente as classes de todas as instâncias, ou seja, $f(W, b, X_i) = Y_i$ seja verdadeiro para cada instância X_i pertencente ao conjunto. A verificação se uma instância X_i está sendo classificada corretamente pode ser simplificada como demonstrado na equação abaixo:

$$Y_i \cdot (\langle W, X_i \rangle + b) \geq 0 \tag{2.3}$$

Caso a equação acima seja verdadeira para todas as instâncias conjunto $\mathcal{X}$, então o hiperplano (W, b) é dito como factível.

Algoritmos que encontrem hiperplanos factíveis são de extrema importância para o problema de classificação. Entretanto, apesar do tempo de processamento e da factibilidade de um algoritmo para encontrar a solução ser relevante, existe um outro fator com maior importância a ser considerado. Este fator está relacionado a capacidade de generalização da solução encontrada. Desta forma, diferentes soluções podem ser geradas por diferentes algoritmos e, apesar de classificarem corretamente todas as instâncias do conjunto de treinamento, possuirão poder de generalização diferentes. Neste sentido, é fundamental encontrar classificadores que tenham uma acurácia elevada quando testados para novas instâncias não pertencentes a este conjunto.

2.2 MARGEM E DISTÂNCIAS

Considerando a existência de mais de um classificador factível para um mesmo problema, é necessária uma métrica para estimar qual deles obterá menos erros em dados

desconhecidos. Uma das métricas usadas para estimar a generalização de um classificador é a margem. Ela é definida em função da distância da instância mais próxima ao hiperplano separador.

Seja uma instância X_i e um hiperplano separador em que W é seu vetor normal e b o seu parâmetro de viés. Sendo assim, a distância funcional da instância ao hiperplano separador é calculada pela seguinte equação:

$$\delta_F(W, b, X_i) = \langle W, X_i \rangle + b \tag{2.4}$$

Logo o valor da distância funcional da instância mais próxima multiplicado pelo respectivo rótulo, chamada de margem funcional, pode ser determinada a partir de:

$$\gamma_F = \inf_{X_i \in X, Y_i \in Y} Y_i \cdot \delta_F(W, b, X_i) \tag{2.5}$$

Entretanto, como o cálculo da margem envolve uma multiplicação de vetores, a norma do vetor W influencia no resultado da margem. Dado que cada classificador pode ter um vetor W de tamanho diferente, para se realizar uma comparação justa entre as margens de dois classificadores, devemos dividir a distância funcional pela norma Euclidiana dos classificadores, sendo esta função de distância chamada de *distância geométrica*:

$$\delta_G = \frac{\delta_F}{||W||_2} \tag{2.6}$$

Assim, para calcular a margem geométrica de um classificador deve-se utilizar a *distância geométrica*, exemplificada pela seguinte equação:

$$\gamma_G = \inf_{X_i \in X, Y_i \in Y} Y_i \cdot \delta_G(W, b, X_i) \tag{2.7}$$

Classificadores com um alto valor de margem geométrica tendem a possuir um alto poder de generalização. Nesse sentido, um dos algoritmos desenvolvidos neste trabalho, que será apresentado no capítulo 4, visa gerar um classificador com um alto valor de margem por meio da utilização de uma estratégia evolucionista.

2.3 ESPAÇO DE VERSÕES

Dado um conjunto Z_m de exemplos de treinamento, seu espaço de versões é definido como o conjunto de todos os classificadores possíveis que inferem corretamente o rótulo de cada instância. Para entender melhor o conceito do espaço de versões, considerando um hiperplano com vetor normal $\tilde{W} = (W, b)'$ e definindo $A_i = (Y_i.X_i, Y_i)'$, pode-se escrever a Equação 2.3 como:

$$\langle \tilde{W}, A_i \rangle \geq 0. \tag{2.8}$$

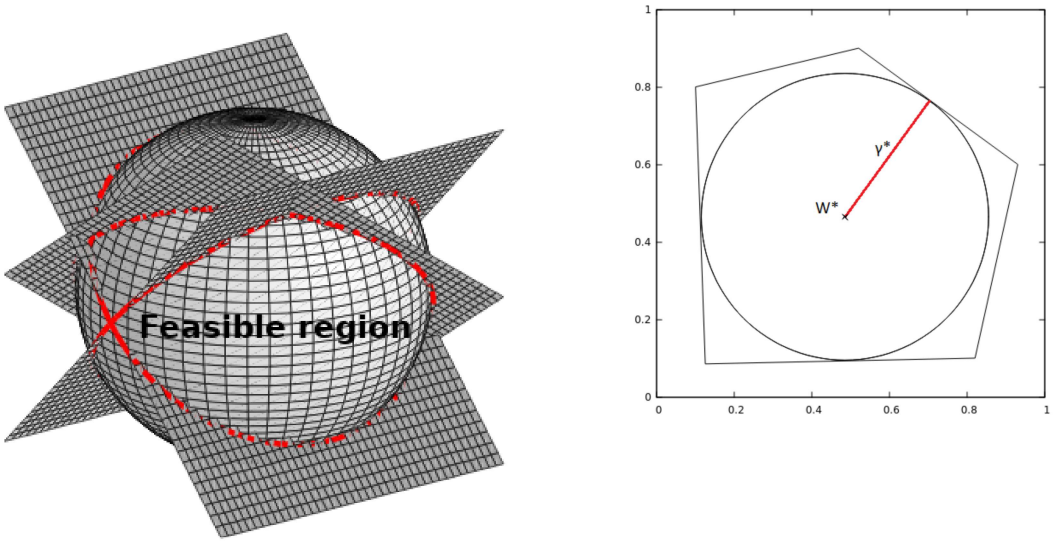
Então, pode-se verificar que cada instância X_i, acompanhada de seu rótulo Y_i implica um semi-espaço de $\mathbb{R}^{d+1}$ onde o vetor normal que define o semi-espaço é dado por A_i. Note que este é um semi-espaço que passa pela origem. Portanto, cada um desses semi-espaços define faces de um poliedro de forma que as equações de restrição para o conjunto Z_m sejam totalmente satisfeitas. Observe que a interseção desses semi-espaços, onde todos passam pela origem, forma um cone poliédrico. Este cone fica vazio apenas no caso de os dados não serem linearmente separáveis. Se for considerada uma suposição padrão adicional de que $\tilde{W}$ tem norma unitária, então o espaço de versões é a interseção de tal cone poliédrico com a superfície da esfera unitária. Usando S^{d+1} para denotar a esfera unitária em $\mathbb{R}^{d+1}$, formalmente o espaço de versões pode ser definido como:

$$V_z = \left\{ (W, b) \in S^{d+1} : \; Y_i \cdot (\langle W, X_i \rangle + b) \geq 0, \; \forall (X_i, Y_i) \in Z_m \right\}. \tag{2.9}$$

O ponto relativo ao vetor normal do hiperplano de margem máxima é o centro da maior hiper-esfera que pode ser circunscrita no cone poliédrico [30]. A Figura 1 mostra um exemplo do espaço de versões projetado em duas dimensões para um problema de classificação binária, cuja solução tem três suportes, cada um representado por uma face tangente ao círculo inscrito.

Figura 1a Gráfico de um espaço de versões (região viável) em um espaço tridimensional. Figura 1b Espaço de versões de um problema de classificação binária cuja solução possui três vetores de suporte, o centro do círculo indica a solução de margem máxima.

(a) (b)

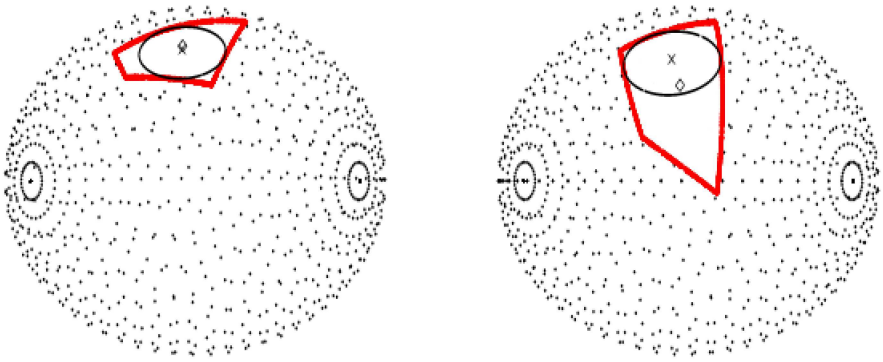


Fonte: **Figura 1a** adaptada de [31]. **Figura 1b** Própria produção.

2.4 CENTRO ANALÍTICO

É conhecido que o centro de massa do espaço de versões é uma boa aproximação do Ponto de Bayes [31], o qual possui o maior poder de generalização. Nesse sentido, apesar da solução *SVM* aproximar o centro de massa, existem casos, como em que o espaço de versões é alongado ou assimétrico, nos quais a solução *SVM* não é uma boa aproximação comparada à solução do centro de massa. Este caso é exemplificado pela Figura 2.

Figura 2 - Em vermelho, espaço de versões para dois problemas tridimensionais. O diamante é o centro de massa e a cruz o centro da maior esfera inscrita. Quanto mais alongado for o espaço de versões, maior será a distância de seu centro até o centro da esfera. Imagem retirada de [31] página 261.



Fonte: Própria produção.

Para tais casos, estimar o centro de massa levaria a uma solução de maior generalização. Porém, o algoritmo capaz de calcular com precisão o volume de um poliedro n-dimensional, e assim consequentemente seu centro de massa, é um procedimento recursivo [43] cuja complexidade é exponencial, da ordem $O(d^m)$, sendo d a dimensão e m o número de faces, ou restrições, relacionada à quantidade de instâncias. Devido a isto, um método capaz de eficientemente aproximar o centro de massa em tempo hábil é de grande utilidade.

Sendo assim, uma alternativa eficiente para aproximar o centro de massa seria calcular seu centro analítico. Esta forma de aproximação do centro de massa de um poliedro é calculada ao minimizar o uma soma de funções potenciais, denominadas função barreira. Para o problema de classificação binária, é utilizado como função barreira o negativo do logaritmo da distância de uma instância ao hiperplano [68]. Deste modo, procura-se minimizar o somatório do negativo destes logaritmos, assim limita a solução dentro do espaço de versões e afastando a mesma das faces do cone poliédrico que o forma. Formalmente o cálculo do centro analítico é definido por:

$$\min f(W,b) = -\sum_{i=1}^{m} \ln(Y_i \cdot (W \cdot X_i + b)). \tag{2.10}$$

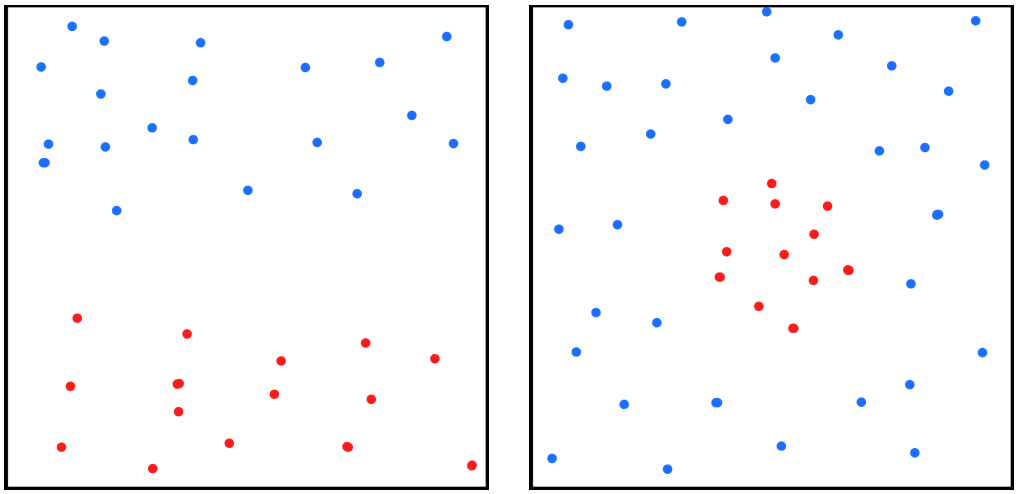
É importante mencionar que os argumentos da função logarítmica não devem ter valores negativos, ou seja, caso o classificador não seja factível o mesmo deve ser descartado.

2.5 FORMULAÇÃO DUAL

A formulação apresentada até então é chamada de primal. Ela é utilizada para gerar explicitamente classificadores lineares, ou seja, que formam um hiperplano separador no espaço de entrada. Os problemas que esse tipo de classificador consegue resolver corretamente são chamados de linearmente separáveis. As bases de dados que não podem ser separadas por um hiperplano no espaço de entrada são denominadas não linearmente separáveis. A Figura 3 mostra exemplo destes dois tipos de bases de dados.

Figura 3 - Exemplos de bases de dados bidimensionais. Os pontos azuis pertencem a classe positiva, e os pontos vermelhos a classe negativa.

(a) Base linearmente separável. (b) Base não-linearmente separável.



Fonte: Própria produção.

A formulação dual permite tratar bases não-linearmente separáveis. Esta formulação possibilita a geração de classificadores que, ao invés de serem apenas hiperplanos, tenham forma não linear no espaço de entrada, podendo assim separar corretamente as duas classes. Esta formulação atua aplicando uma função kernel para calcular implicitamente o produto interno de dois vetores do espaço de características. A restrição para uma função poder ser usada como kernel é ela atender a condição de Mercer [49]. A utilização do *kernel trick* permite simular implicitamente uma função de mapeamento para um espaço de dimensionalidade maior, tornando assim possível trabalhar em um espaço onde o conjunto de amostras seja linearmente separável, e assim gerar um classificador equivalente que seja não linear no espaço de entrada original.

Um classificador dual opera em um espaço dual de dimensão m, onde m se refere a quantidade de instâncias. Ele utiliza um vetor de variáveis duais também chamado de vetor de alphas. Para cada componente deste vetor atribui-se um peso ao valor da similaridade entre a nova instância que está sendo analisada e uma amostra do conjunto de treinamento. No espaço dual, o cálculo da distância funcional de uma instância X_i até o hiperplano (α, b), sendo α o vetor de variáveis duais e b o seu viés, para uma função kernel K, depende de todo o conjunto de treinamento Z_m sendo dado por:

$$\delta_F(\alpha, b, X_i, K()) = \sum_{j=1}^{m} \alpha_j \cdot Y_j \cdot K(X_i, X_j) \tag{2.11}$$

Entretanto, para o cálculo da distância geométrica δ_G é necessário determinar a norma euclidiana do vetor normal W em função do vetor α. Considerando a representação dual do vetor W, a sua norma pode ser avaliada pelo seguinte duplo somatório:

$$||W||_2 = \langle W, W^t \rangle^{\frac{1}{2}} = (\sum_{i=1}^{m} \alpha_i \cdot Y_i \cdot X_i \cdot \sum_{j=1}^{m} \alpha_j \cdot Y_j \cdot X_j)^{\frac{1}{2}} \tag{2.12}$$

Que é equivalente a:

$$(\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \cdot \alpha_j \cdot Y_i \cdot Y_j \cdot K(X_i, X_j))^{\frac{1}{2}} \tag{2.13}$$

Na formulação dual, uma amostra $(X_i, Y_i) \in Z_m$ é classificada corretamente para uma função kernel K se:

$$Y_i \cdot \sum_{j=1}^{m} \alpha_j \cdot Y_j \cdot K(X_i, X_j) \geq 0 \tag{2.14}$$

Apesar da vantagem de se utilizar a formulação dual para classificação em problemas não linearmente separáveis, a mesma tem como ponto negativo seu custo computacional. Este custo para avaliar a classe de uma nova instância é proporcional a quantidade de amostras do conjunto de treinamento. Para a avaliação de uma nova instância na formulação primal este custo é proporcional à dimensão do espaço de entrada. Desta forma a formulação primal é mais indicada para problemas linearmente separáveis, principalmente caso possuam uma grande quantidade de amostras e uma menor dimensão. Assim, para avaliação da norma Euclidiana do vetor normal tem-se custo quadrático em relação a quantidade de amostras enquanto que na formulação primal este custo é linear em relação à dimensão do problema. Para o caso de problemas não linearmente separáveis, a formulação dual possui a vantagem inerente de conseguir classificar corretamente todas as instâncias, porém a formulação primal ainda pode ser útil pela sua velocidade caso a quantidade de erros seja tolerável dado o contexto da aplicação. Deste modo, o desenvolvimento de algoritmos que operem na formulação primal são de grande importância para problemas com

grande quantidade de amostras. A seguir será detalhado as características e propriedades do espaço dual de versões.

2.5.1 Espaço Dual de Versões

Neste caso o espaço de versões é definido entre todos os classificadores possíveis que inferem corretamente o rótulo de cada instância pertencente a Z_m para uma dada função kernel K. Seja o hiperplano representado $\tilde{\alpha} = (\alpha, b)'$ e definindo $A_i = (Y_i \cdot G_i, Y_i)'$ onde $G_i = \sum_{j=1}^{m} \alpha_j \cdot Y_j \cdot K(X_i, X_j)$, pode-se reescrever a Equação 2.14 como:

$$\langle \tilde{\alpha}, A_i \rangle \geq 0 \tag{2.15}$$

Da mesma forma que o espaço primal de versões, a interseção dos semi-espaços definidos pela Equação 2.15 formam um cone poliédrico de m faces que passam pela origem. Entretanto seu espaço dimensional é $\mathbb{R}^{m+1}$ sendo também proporcional a quantidade de amostras. Ao contrário do espaço primal não é possível definir a restrição de norma unitária do vetor normal como uma casca esférica pelo simples fato desta norma ser computada pela Equação 2.13. Portanto, na implementação dual, não faz sentido a adoção de uma norma unitária para o vetor α e a sua consequente representação pelo sistema de coordenadas esféricas.

2.6 MÉTODOS RELACIONADOS

Nesta seção serão apresentadas as técnicas de Aprendizado de Máquinas usadas como referência durante o desenvolvimento deste trabalho, enquanto algumas são clássicas, outras são recentes e fazem parte do estado da arte atual; todas elas essenciais tanto para o processo de desenvolvimento dos algoritmos aqui apresentados, quanto para entender seu funcionamento. Estas técnicas serão utilizadas como base de comparação para os resultados obtidos pelos algoritmos desenvolvidos.

2.6.1 Perceptron

Um dos mais antigos e conhecidos algoritmos de Aprendizado de Máquinas é o modelo Perceptron [62]. Este algoritmo foi desenvolvido a partir de um modelo matemático para os neurônios biológicos que compõem os cérebros de seres vivos, devido a isto ele é chamado de neurônio artificial [48]. Ele é definido por um hiperplano d-dimensional, onde d se refere a dimensão do espaço de entrada. O processo de classificação de um neurônio é definido nas Equações 2.2 e 2.3.

Uma instância é classificada corretamente quando sua classe é igual a prevista pela saída do neurônio. No caso do valor previsto ser diferente do desejado, então a classificação é considerada como errada, e os parâmetros do hiperplano são atualizados de modo a

corrigir o erro. Esta atualização, na formulação primal, ocorre no vetor de pesos e no viés do hiperplano (W, b) por meio de equações de correção dadas por:

$$W_{t+1} = W_t + r \cdot Y_i \cdot X_i, \tag{2.16}$$

$$b_{t+1} = b_t + r \cdot Y_i \tag{2.17}$$

onde W_{t+1} é o vetor de pesos atualizado, W_t o vetor de pesos antes da correção, b_{t+1} o viés atualizado, b_t o viés antes da correção, r a taxa de aprendizado e Y_i o rótulo associado à instância X_i classificada erroneamente.

Este processo de aprendizado, também chamado de treinamento, é executado até que todas as instâncias possam ser classificadas corretamente. A convergência do algoritmo é garantida para um número finito de iterações caso o conjunto de treinamento seja linearmente separável e caso a taxa de aprendizado seja pequena o suficiente [59].

Para a formulação dual, o hiperplano é representado por um vetor de variáveis duais α e um valor de viés b. O processo de aprendizado é similar ao da formulação primal, porém são utilizadas equações de correção diferentes, sendo dadas a seguir:

$$\alpha_{t+1}[i] = \alpha_t[j] + r, \text{ Se } i = j. \tag{2.18}$$

$$\alpha_{t+1}[i] = \alpha_t[j], \text{ Se } i \neq j. \tag{2.19}$$

$$b_{t+1} = b_t + r \cdot Y_i \tag{2.20}$$

onde j é o índice da instância classificada erroneamente.

A simplicidade destas funções de atualização fazem com que o Perceptron seja um algoritmo de classificação de baixo custo computacional. Apesar disto, a solução encontrada por ele pode estar bem longe da considerada de máxima margem.

2.6.2 Máquina de Vetores Suporte

Nesta subseção, será discutido a formulação primal da Máquina de Vetores Suporte (SVM). Este algoritmo tem como objetivo encontrar a solução de margem máxima, ou seja, que maximiza a distância do hiperplano às instâncias mais próximas de ambas as classes. Serão apresentadas diferentes formulações primais do problema, as quais levam a diferentes interpretações geométricas que guiarão o desenvolvimento do algoritmo evolutivo de larga margem apresentado no Capítulo 4.

Em sua formulação primal, considerando um problema linearmente separável, o SVM retorna um hiperplano que não só classifica corretamente todas as instâncias, mas também maximiza a distância às instâncias mais próximas de ambas as classes. Tal solução é definida formalmente como a solução do seguinte problema maxmin:

$$\max_{W\in\mathbb{R}^d,\, b\in\mathbb{R}} \min_{\{X_i,Y_i\}\in Z_m} \frac{Y_i \cdot (\langle W, X_i\rangle + b)}{\|W\|_2} \tag{2.21}$$

$$\text{sujeito a } Y_i \cdot (\langle W, X_i\rangle + b) \geq 0, \quad i = 1...m. \tag{2.22}$$

Esta formulação, segundo [12], pode ser reescrita como o seguinte problema de minimização de norma:

$$\min_{W\in\mathbb{R}^d,\, b\in\mathbb{R}} \frac{1}{2}\|W\|_2^2 \tag{2.23}$$

$$\text{sujeito a } Y_i \cdot (\langle W, X_i\rangle + b) \geq 1, \quad i = 1...m. \tag{2.24}$$

Equivalentemente, [44] reescreveu o problema maxmin como o seguinte problema de maximização:

$$\max_{W\in\mathbb{R}^d,\, b\in\mathbb{R},\, \gamma\in\mathbb{R}_+} \gamma \tag{2.25}$$

$$\text{sujeito a } Y_i \cdot (\langle W, X_i\rangle + b) \geq \gamma \|W\|_2, \quad i = 1...m. \tag{2.26}$$

Sejam $W^\star$ e $\gamma^\star$ as soluções ótimas para seus respectivos problemas de otimização. Pode-se conectar estas duas soluções pela seguinte relação:

$$\|W^\star\|_2 = \frac{1}{\gamma^\star}.$$

A seguir, será mostrado as diferentes interpretações geométricas e a forma como isto conduziu ao desenvolvimento do algoritmo evolutivo proposto no Capítulo 4.

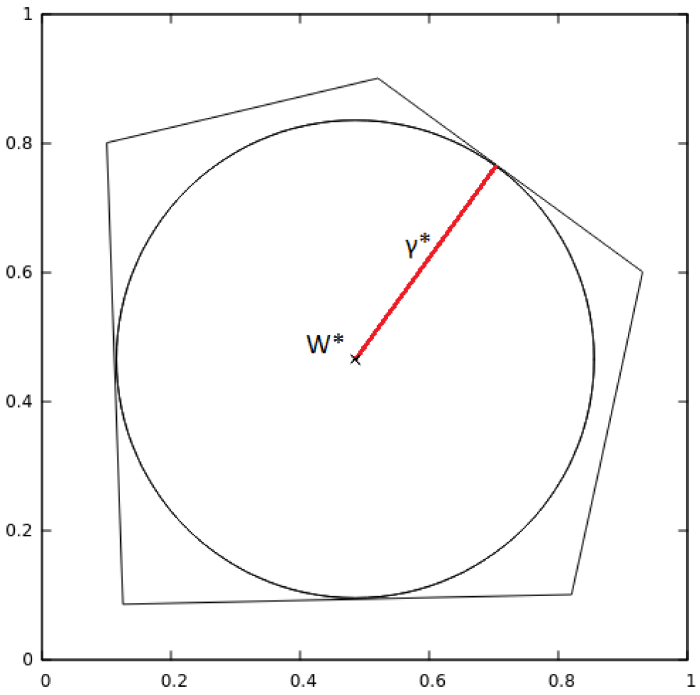
2.6.2.1 Interpretação geométrica da solução SVM

A solução SVM pode ser interpretada geometricamente como sendo o centro da maior hiperesfera que pode ser circunscrita dentro espaço de versões. A Figura 4 exemplifica esta interpretação em um domínio bidimensional.

A partir disto, é possível analisar sob a perspectiva geométrica o funcionamento das formulações previamente apresentadas, o que revela diferentes modos de lidar com a solução deste problema no espaço de versões. Compreender as diferenças entre estas formulações, cada uma com suas peculiaridades, é fundamental para projetar o funcionamento de um novo algoritmo que vise obter a solução SVM de modo eficiente.

A primeira é a formulação de Vapnik [12], onde a solução é o centro da hiperesfera unitária circunscrita no espaço de versões, onde a margem funcional dos suportes é igual a um, o que minimiza a norma euclidiana do vetor W.

Figura 4 - Visualização geométrica da solução SVM.



Fonte: Própria produção.

A segunda é a formulação apresentada em [44], onde a solução é o centro da maior hiperesfera que pode ser inscrita em um espaço de versões irrestrito. A solução é garantida pelo uso do conceito de margem geométrica, que controla implicitamente a norma do vetor W, evitando assim que a mesma escalone.

A terceira formulação refere-se a segunda porém mantendo a norma unitária. A solução é a maior hiperesfera que pode ser circunscrita no espaço de versões restrito, onde o centro da mesma deve estar na superfície hiperesférica definida pelo fato de $||W||_2 = 1$. Neste caso específico a margem funcional e geométrica apresentam os mesmos valores.

A quarta formulação é apresentada em [5], esta formulação utiliza uma relação entre as áreas de Aprendizado de Máquinas e Geometria Computacional. A partir do conjunto de amostras, é gerado uma casca convexa para as instâncias de cada classe. O hiperplano de máxima margem é encontrado a partir da menor distancia entre as cascas convexas.

É importante destacar que estas quatro soluções embora se situem em locais diferentes do espaço de versões possuem a mesma direção em relação a origem do mesmo, se posicionando portanto no mesmo raio. Para o desenvolvimento do algoritmo de larga margem optou-se pela terceira formulação, a qual mantém a norma do vetor normal unitária.

2.6.3 Bayes Point Machine

É reconhecido que o ponto de Bayes supera consistentemente o SVM em poder de generalização. Uma aproximação para o ponto de Bayes, chamada de Bayes Point Machine (BPM), foi proposta por Herbrich, Graepel e Campbell [31]. O BPM é um método de amostragem que consiste em usar como classificador a média de um conjunto de Perceptrons consistentes com o espaço de versões. Para estimar o ponto de Bayes no espaço de versões, é utilizado um método de Monte Carlo, ou seja, em vez de calcular exatamente a expectativa, ela é aproximada por uma média sobre os vetores. No entanto, é muito difícil amostrar uniformemente o espaço de versões, pois o mesmo consiste de todo o conjunto de pontos que define um poliedro restrito pela esfera unitária. Para obter uma amostragem aproximada, dois métodos foram propostos.

O primeiro se desenvolve a partir de uma ideia que se baseia em jogar *bilhar* no espaço de versões [65, 63]. As analogias são as seguintes: um algoritmo de aprendizado simples, como o Perceptron, é considerado como a bola; dado que o espaço de versões é um poliedro convexo, ele é representado como sendo uma mesa. O classificador começa como um ponto interno qualquer do espaço de versões, a cada iteração ele se movimenta em uma direção, caso saia do espaço de versões ele tem a sua direção de movimento modificada e a iteração é recalculada. A cada iteração, a sua posição é utilizada para atualizar a aproximação do centro de massa.

Se este bilhar é ergódico em relação à distribuição uniforme sobre o espaço de versões, ou seja, o tempo de viagem da bola de bilhar despendido em um subconjunto ou igual ou menor ao espaço de versões é proporcional a seu volume, então a média sobre a trajetória da bola de bilhar leva, no limite de um número infinito de saltos, ao centro de massa do espaço de versões.

O segundo método tenta superar as grandes demandas computacionais do método do bilhar por meio de uma aproximação da amostragem uniforme do espaço de versões. A ideia é cada ponto da amostragem ser gerador a partir do algoritmo Perceptron em variáveis duais, porém para gerar cada novo ponto, o Perceptron é treinado com uma diferente permutação na ordem dos dados de treinamento, o que leva a obter diferentes classificadores consistentes [71]. Devido ao número de diferentes amostras obtidas ser finito é impossível atingir a solução exata mesmo ao considerar todas as permutações. No entanto em certas bases de dados, em particular para a tarefa de reconhecimento de dígitos manuscritos, o desempenho alcançado por este método foi comparável aos melhores algoritmos de sua época.

2.6.4 Version Space Reduction Machine

Classificadores de baixa acurácia podem ser combinados para se comportarem como se fossem um classificador de alta acurácia [23]. Uma das técnicas para realizar isto é

denominada de comitê. Ele consiste em um conjunto de hipóteses que foram induzidas separadamente, porém cujas saídas são combinadas, como, por exemplo, por meio de uma votação que retorna a saídas mais comum dentre os membros, sendo esta solução a retornada pelo comitê. Este método tem como objetivo gerar soluções com mais capacidade de generalização do que cada membro do comitê seria capaz individualmente [69, 18, 38]. Para um comitê ser eficaz é necessário que seus membros sejam diversos.

O algoritmo VSRM [22] tem como objetivo gerar um comitê, e a partir deste, aproximar o centro de massa do espaço de versões. O comitê é formado por um conjunto de classificadores Perceptrons. Para garantir a diversidade entre seus membros, cada candidato a participar é inicializado com pesos aleatórios e treinado com uma diferente ordem dos dados de entrada. Também é utilizado uma medida de dissimilaridade, sendo ela calculada como a menor distância euclidiana entre o candidato e os membros do comitê:

$$\min_{h_i \in H} \{||h_i, h_c||2\} > \epsilon \tag{2.27}$$

onde H é o conjunto de classificadores pertences ao ensemble, h_i os classificadores pertencestes ao comitê, h_c o Perceptron candidato. Para o candidato ser admitido no comitê, ele precisa de ter uma dissimilaridade maior que um valor mínimo ϵ. Caso a dissimilaridade do candidato seja maior que o valor estipulado para ϵ, ele é adicionado ao comitê, caso contrário, um novo candidato é gerado e testado. Este processo se repete até que o comitê tenha uma certa quantidade de membros. Após o comitê ser formado, ele é utilizado como guia em um processo iterativo de reduzir o espaço de versões.

O objetivo final do VSRM é aproximar o centro de massa do espaço de versões. Isto é feito por meio de sucessivos cortes, que diminuem o tamanho do espaço de versões retirando sempre o semiespaço de menor volume. A decisão de qual semiespaço será descartado é feita por um oráculo formado pelo comitê de perceptrons. O hiperplano de corte é gerado a cada iteração sendo definido por uma direção escolhida aleatoriamente que consiste do vetor normal de uma face de um ponto viável dentro de cada espaço reduzido gerado por uma adaptação do algoritmo Perceptron [44]. Caso os membros do comitê estejam bem distribuídos no espaço de versões, o semiespaço que possui a maioria deles será o de maior volume. Este processo é repetido até que o espaço de versões restrito convirja para um ponto, sendo este ponto uma boa aproximação do centro de massa.

3 COORDENADAS HIPERESFÉRICAS

Como demonstrado na subseção 2.3 (Espaço de Versões), o problema de classificação binária tem fortes relações com a área de geometria, podendo ser modelado como um problema de encontrar um ponto específico no espaço de versões. Este capítulo apresenta a primeira contribuição deste trabalho: a introdução e utilização de um sistema diferente de coordenadas para problemas de classificação, as coordenadas hiperesféricas, e dois algoritmos de conversão entre coordenadas hiperesféricas e Cartesianas. Na literatura, é tradicionalmente utilizado o sistema de coordenadas Cartesianas para tratar o problema de classificação, sendo uma dos poucos trabalhos relacionados uma trabalho, desenvolvido simultaneamente e independente a este, sobre a utilização de coordenadas hiperesféricas para redes neurais [64].

3.1 INTRODUÇÃO E DEFINIÇÃO

O sistema de coordenadas hiperesféricas é uma extensão do sistema de coordenadas esféricas para dimensões maiores que três, assim primeiro será feito uma apresentação do sistema esférico.

O sistema de coordenadas esféricas é uma forma de representar uma posição vetorial que se encontra em um espaço tridimensional. O sistema esférico utiliza três valores para representar uma posição: o raio r que indica a distância o ponto a origem; o ângulo polar ϕ que pode variar de 0 a π; o ângulo azimutal θ que pode variar de 0 a 2π [54].

Para facilitar seu entendimento, duas comparações podem ser feitas com o sistema cartesiano. A primeira é em relação a como um ponto em cada um desses sistemas pode ser interpretado. Enquanto no sistema cartesiano uma posição é representada a partir de três vetores ortogonais, ou seja, como se cada posição fosse um vértice de um cubo imaginário, no sistema esférico os valores de ϕ e θ representam uma posição em uma esfera centrada na origem, enquanto o valor de r representa o raio desta esfera. A segunda comparação é sobre a relação dos ângulos ϕ e θ com as coordenadas cartesianas: o angulo θ representa uma rotação em torno do eixo Z das coordenadas cartesianas enquanto o angulo ϕ é o responsável pela posição do ponto ao longo do eixo Z. As Figuras 5a e 5b exemplificam estas duas comparações.

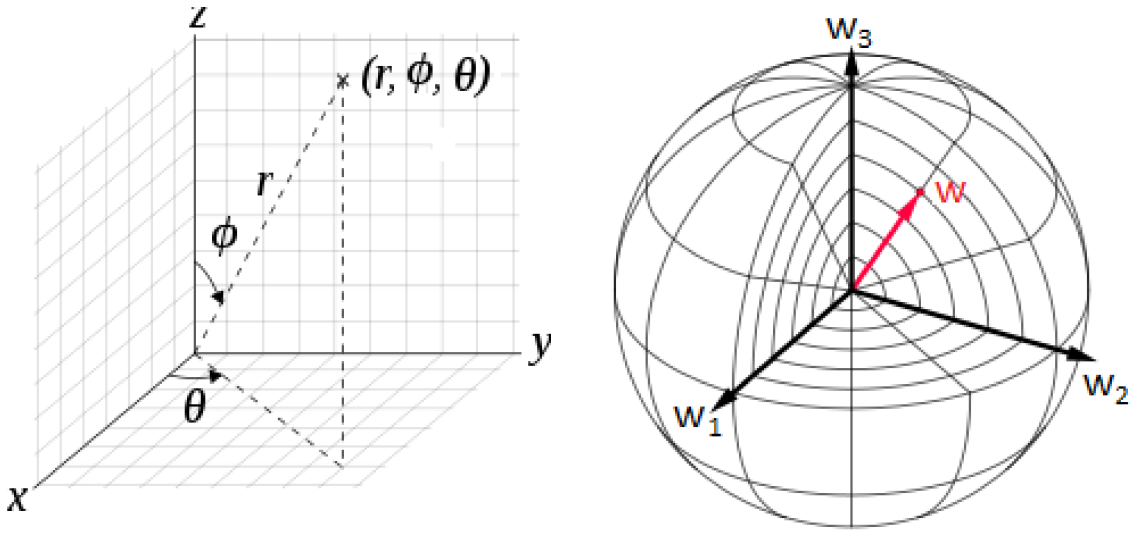
A conversão de um vetor W de coordenadas cartesianas para um vetor em coordenadas esféricas, pode ser feito aplicando as seguintes fórmulas:

$$r = ||W|| \tag{3.1}$$

$$\phi = \arctan \frac{W_Y}{W_X} \tag{3.2}$$

Figura 5 - **(a)** Sistema de coordenadas esféricas em um espaço 3-dimensional. **(b)** Representação de um vetor unitário tridimensional em uma esfera unitária.

(a) (b)



Fonte: Figura 5(a) retirada de: https://en.wikipedia.org/wiki/File:3D_Spherical_2.svg. Figura 5(b) Modificação da imagem retirada de: https://en.wikipedia.org/wiki/File:Spherical_coordinate_system.svg. Ambas sob a licença Creative Commons Attribution-Share Alike 4.0 International: https://creativecommons.org/licenses/by-sa/4.0/deed.en .

$$\theta = \arccos W_Z \tag{3.3}$$

em que W_X é a coordenada referente ao eixo X, W_Y é a coordenada referente ao eixo Y e W_Z é a coordenada referente ao eixo Z.

Como este sistema esférico intrinsecamente representa vetores na superfície de uma esfera, e como o espaço de versões da terceira formulação primal SVM pode ser representado como parte da superfície de uma esfera unitária, ele é apropriado para representar o vetor unitário W de um classificador. Porém, este sistema é limitado a apenas 3 dimensões, o torna necessário extende-lo a mais dimensões para poder ser aplicado em problemas d-dimensionais, onde $d > 3$. Assim, a extensão do sistema de coordenadas esféricas para dimensões maiores é chamado de *sistema de coordenadas hiperesféricas*. Enquanto o sistema esférico considera que os pontos estão na superfície de uma esfera, este sistema considera que os pontos estão na superfície de uma esfera d-dimensional (hiperesfera).

3.2 ALGORITMOS DE CONVERSÃO

Como as bases de dados tem, em quase sua totalidade, pontos representados de modo Cartesiano, é essencial ter um modo de converter um ponto representado em coordenadas Cartesianas para o sistema hiperesférico. Considerando que, por razões

práticas, pode ser necessário ter um classificador descrito em coordenadas cartesianas, também é essencial ter um modo de fazer a conversão de volta.

Após uma pesquisa nos métodos existentes para conversão percebeu-se que estes possuem uma tendência a gerarem erros numéricos devido a realizarem sucessivas operações trigonométricas para o cálculo de seno e arco tangente. Visando resolver estas duas necessidades, dois algoritmos para realizar estas conversões tiveram que ser desenvolvidos, sendo eles a primeira contribuição deste trabalho. Ambos requerem tempo computacional linear em relação a dimensionalidade do problema, e são de implementação simples, requerendo apenas operações trigonométricas. O pseudo-código 1 descreve a conversão para coordenadas hiperesféricas, e o pseudo-código 2 descreve a conversão de volta para coordenadas Cartesianas. É importante observar que quando ocorre um aumento da dimensionalidade, somente a quantidade e ângulos polares aumenta proporcionalmente permanecendo somente o angulo azimutal.

Algorithm 1: Algoritmo para converter um vetor de coordenadas cartesianas em um vetor de coordenadas hiperesféricas.

Vetor Cartesiano: W;
Dimensão de W: d;
Vetor de ângulos polares: $\phi[d-2]$;
Angulo azimutal: θ;
$W \leftarrow \frac{W}{||W||}$;
$squared\_Sum \leftarrow 1$;
for $(i = 0; i < d-2; i++)$ **do**
    $\phi_i \leftarrow \arccos\left(\frac{W_i}{\sqrt{squared\_Sum}}\right)$;
    $squared\_Sum \leftarrow squared\_Sum - W_i \cdot W_i$;
end
if $(W_{d-1} \geq 0)$ **then**
    $\theta \leftarrow \arccos\left(\frac{W_{d-2}}{\sqrt{squared\_Sum}}\right)$;
else
    $\theta \leftarrow 2 \cdot \pi - \arccos\left(\frac{W_{d-2}}{\sqrt{squared\_Sum}}\right)$;
end
return ϕ, θ;

Este sistema de coordenadas apresentou diversas vantagens quando utilizado nos classificadores evolutivos que foram desenvolvidos neste trabalho, tanto relacionadas a velocidade de execução do algoritmo devido a dispensar a necessidade de realizar procedimentos de normalização que eram constantes quando utilizado o sistema Cartesiano, quanto de proporcionar uma distribuição uniforme do espaço, necessária para a execução destes métodos. Essas vantagens serão discutidas com mais profundidade nos dois capítulos seguintes.

Algorithm 2: Algoritmo para converter um vetor de coordenadas hiperesféricas em vetor de coordenadas cartesianas.

Vetor de ângulos polares: ϕ;
Dimensão de ϕ: d_h;
Angulo azimutal θ;
Vetor Cartesiano: $W[d_h + 2]$;
$squared\_Sum \leftarrow 1$;
for $(i = 0; i < d_h - 1; i++)$ **do**
    $W_i \leftarrow \cos(\phi_i) \cdot \sqrt{squared\_Sum}$;
    $squared\_Sum \leftarrow squared\_Sum - W_i \cdot W_i$;
end
$W_{d_h-1} \leftarrow \cos(\theta) * \sqrt{squared\_Sum}$;
$W_{d_h} \leftarrow |\sqrt{squared\_Sum - W_{d_h-1} \cdot W_{d_h-1}}|$;
if $\theta > \pi$ **then**
    $W_{d_h} \leftarrow -1 \cdot W_{d_h}$;
end
return W;

4 ALGORITMO EVOLUCIONISTA DE LARGA MARGEM (AELM)

Até recentemente, foram realizados poucos estudos sobre a utilização de técnicas evolutivas para o desenvolvimento de classificadores de larga margem. A partir disto, viu-se potencial para explorar o uso de algoritmos evolutivos neste problema para render resultados competitivos aos das técnicas atuais. Outro fator importante que será apresentado neste capítulo está relacionado à utilização de coordenadas hiper-esféricas no problema SVM. Durante o desenvolvimento do algoritmo, percebeu-se que esse sistema apresenta grandes vantagens em relação ao sistema de coordenadas cartesianas ao retirar a necessidade de realizar operações de normalização nos hiperplanos e ao permitir os operadores evolutivos operarem em uma distribuição uniforme no espaço de versões.

O algoritmo tem um funcionamento simples descrito brevemente a seguir: cada indivíduo da população é um hiperplano representado por seu respectivo vetor normal, de norma unitária, e parâmetro de viés. O algoritmo tem como população inicial um conjunto de indivíduos factíveis treinados pelo algoritmo Perceptron. Para a função objetivo, é utilizado o valor de margem dos indivíduos da população, sendo o objetivo do algoritmo maximizar a melhor margem dentre a população, ou seja, encontrar um indivíduo que seja próximo da solução SVM. Novos indivíduos são gerados aplicando operadores de recombinação e mutação [53]. A substituição de indivíduos ocorre após uma população intermediária ser gerada, seus membros são então utilizados para substituir os indivíduos da população principal, caso passem pelos critérios de substituição. Este processo de geração e substituição de indivíduos é repetido por um número fixo de vezes.

As coordenadas hiper-esféricas são utilizadas para aumentar a eficiência as operações de recombinação e mutação, provendo uma distribuição uniforme na variedade dos indivíduos filhos, garantindo sua factibilidade e preservando a norma de um modo computacionalmente eficiente.

Uma visão geral do processo evolutivo de uma geração pode ser vista no diagrama da Figura 6.

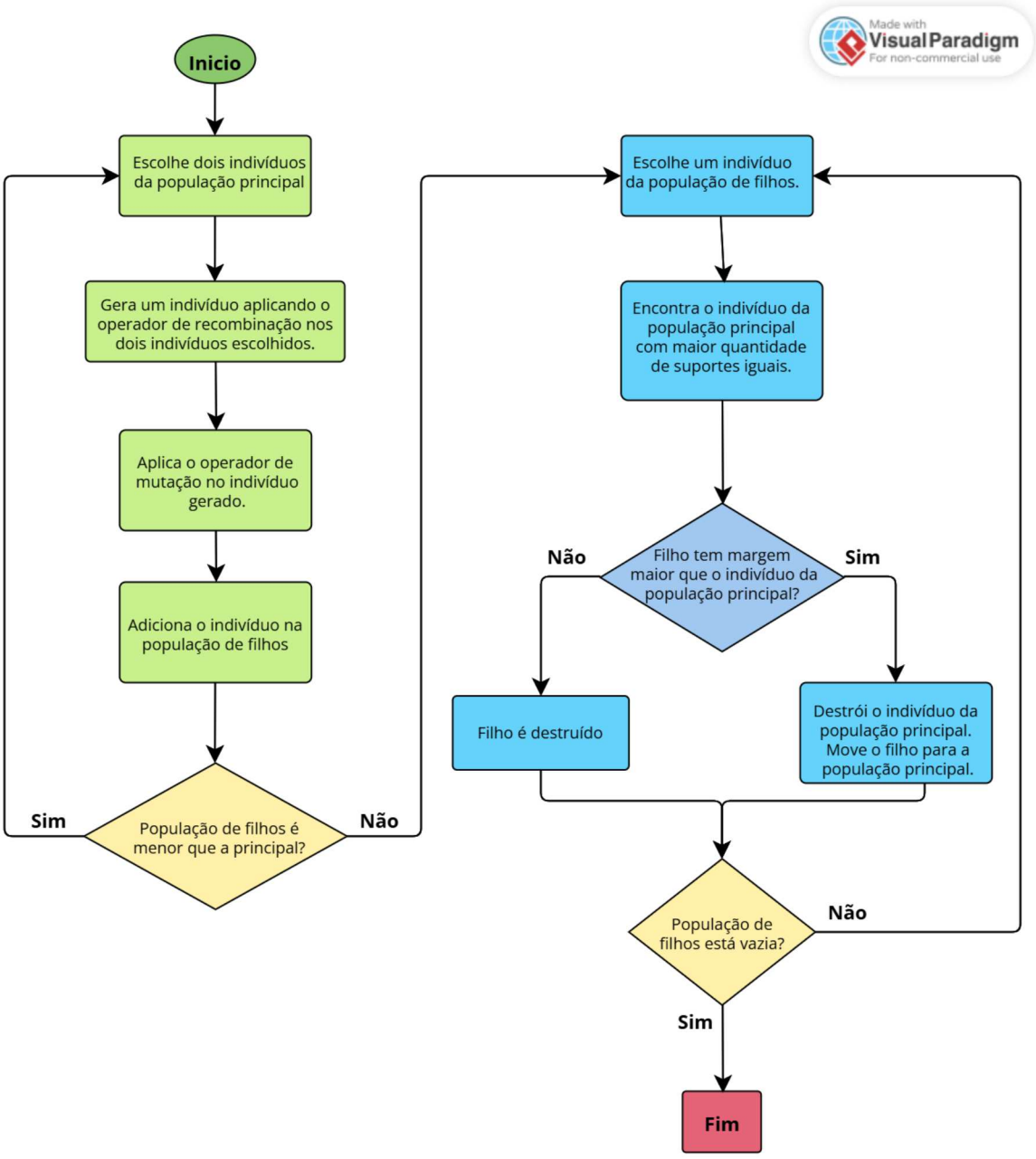
4.1 POPULAÇÃO INICIAL

A população inicial consiste em um conjunto de indivíduos gerados pelo algoritmo Perceptron. Esta é uma população viável porque cada indivíduo é representado pelo respectivo vetor de peso e parâmetro de viés contido no espaço de versões, definido por:

$$V_z = (\{W, b\} \forall (X_i, Y_i) \in Z_m : Y_i \cdot (\langle W, X_i \rangle + b) \geq 0 \wedge ||W||_2 = 1) \tag{4.1}$$

Caso o problema não seja linearmente separável, o espaço de versões V_Z será vazio. Porém, nesta situação, uma população inicial pode ser gerada usando uma variante do

Figura 6 - Processo evolucionário para uma geração.



Fonte: Própria produção.

algoritmo Perceptron, que permite a flexibilidade da margem [70]. Assim, a ocorrência de um erro somente é verificada se:

$$Y_i \cdot (\langle W, X_i \rangle + b) + \frac{1}{C} \cdot \alpha_i < 0, \tag{4.2}$$

onde C é uma constante de penalidade e α_i a variável dual associada à amostra (X_i, Y_i). A subseção a seguir discute o processo de atualização dessa variável.

Como a população inicial é gerada por meio do sistema de coordenadas cartesianas, precisa-se converter cada indivíduo inicial para o sistema de coordenadas hiperesféricas.

Algorithm 3: Procedimento para gerar a população inicial.

Gerar n Perceptrons ;
Treinar cada Perceptron até classificar corretamente todas as instâncias r até que um certo número de gerações tenha passado ;
Converter cada Perceptron de coordenadas cartesianas para as coordenadas hiperesféricas ;

4.2 FUNÇÃO OBJETIVO

A função objetivo usada para a avaliação de um indivíduo é baseada no valor da margem. Logo, a avaliação da aptidão de um indivíduo representado por (W, b) é dada por:

$$f(W, b) = \min Y_i \cdot (\langle W, X_i \rangle + b), \forall \{X_i, Y_i\} \in Z_m \tag{4.3}$$

dado que a norma do vetor W é sempre unitária.

Esta função de avaliação permite que os valores de margem sejam utilizados como função de avaliação, mesmo quando o hiperplano não está classificando corretamente todas as instâncias, neste caso, deixando a função com valores negativos.

Para otimizar o processo de avaliação do indivíduo, foi utilizado um vetor α de variáveis duais, associando cada componente a uma amostra do conjunto de treinamento e sua respectiva restrição de classificação. Ao avaliar a margem de um indivíduo, os valores α associados às amostras classificadas erroneamente, possíveis vetores de suporte, têm seus valores incrementados. Assim, as amostras que são sempre classificadas corretamente, que provavelmente não serão suportes, têm seus valores *alpha* decrementados até zero podendo ser descartados durante o processo de avaliação da margem. A fórmula de atualização é dada pela Equação 4.4:

$$\alpha_i = \begin{cases} \alpha_i + r, \text{Se } Y_i \cdot (\langle W, X_i \rangle + b) < 0 \\ \alpha_i - r, \text{ caso contrário.} \end{cases} \tag{4.4}$$

onde r é a taxa de aprendizagem e α_i é a variável dual associada a amostra (X_i, Y_i).

O vetor α é mantido em ordem decrescente. Assim, ao realizar a avaliação da margem seguindo a ordenação deste vetor, pode-se descartar amostras com valores *alpha* iguais a zero. É importante destacar que a função objetivo controla implicitamente a viabilidade dos indivíduos filhos em relação às restrições de que formam o espaço de versões. Consequentemente, é possível avaliar a função de aptidão dada pela Equação 4.3 somente em função dos vetores de suporte. Assim a função de avaliação otimizada pode ser computada como:

$$f(W, b) = \min Y_i \cdot (\langle W, X_i \rangle + b), \forall \{X_i, Y_i\} \in S_V, \tag{4.5}$$

onde S_V é o conjunto de vetores de suporte.

4.3 OPERADOR DE RECOMBINAÇÃO

Um indivíduo filho é inicializado pelo operador de recombinação. Os pais são dois indivíduos da população escolhida de acordo com um processo de seleção de roleta, onde a probabilidade de cada indivíduo é relativa à sua colocação em uma lista ordenada de modo crescente pelos valores de aptidão da população. A probabilidade de um indivíduo ser escolhido é dada por:

$$P_i = \frac{i^2 - (i-1)^2}{n^2}, \tag{4.6}$$

onde P_i é a probabilidade para o i-ésimo indivíduo da população e n é o tamanho da população. Cada filho tem seus pesos inicializados por uma combinação linear convexa dos pesos de seus pais. Ao considerar a propriedade convexa do espaço de versões e a viabilidade da população inicial, esses indivíduos gerados cumprirão as restrições de classificação. O processo de recombinação usando coordenadas cartesianas é dado por:

$$W_{f,i} = \lambda \cdot W_{p1,i} + (1 - \lambda) \cdot W_{p2,i}, \ \lambda \in (0, 1), \tag{4.7}$$

onde $W_{f,i}$, $W_{p1,i}$ e $W_{p2,i}$ são, respectivamente, os i-ésimos pesos dos hiperplanos do filho e dos pais. O parâmetro λ é escolhido a partir de uma distribuição de probabilidade uniforme. É importante destacar a necessidade de normalização dos indivíduos pais antes de aplicar a recombinação, a fim de preservar a viabilidade da restrição hiperesférica de raio unitário. Formalmente, isso pode ser demonstrado como:

Teorema 4.3.1. *No sistema de representação de coordenadas cartesianas o operador de recombinação preserva a viabilidade da restrição de raio unitário da hiperesfera do espaço de versões se os vetores tiverem norma unitária, isso é* $||W_f||_2 \leq 1$.

Demonstração. Sejam W_{p1} e W_{p2} os vetores pais com norma unitária $||W_{p1}||_2 = ||W_{p2}||_2 = 1$. Seja o vetor filho representado por uma combinação linear convexa de W_{p1} e W_{p2}. Então:

$$||\lambda \cdot W_{p1} + (1 - \lambda) \cdot W_{p2}||_2 \leq \lambda \cdot ||W_{p1}||_2 + (1 - \lambda) \cdot ||W_{p2}||_2 \leq \lambda + (1 - \lambda) = 1 \tag{4.8}$$

□

A combinação linear convexa dada pela Equação 4.7 também pode ser usada para calcular as coordenadas do filho no sistema de coordenadas hiperesférico, representadas

pelo raio, os ângulos azimutais e polares. Além disso, pode-se provar que se os vetores dos pais têm norma unitária, então, após a recombinação, os vetores filhos também a terão, satisfazendo a restrição da norma unitária do espaço de versões. Formalmente, isso pode ser demonstrado como:

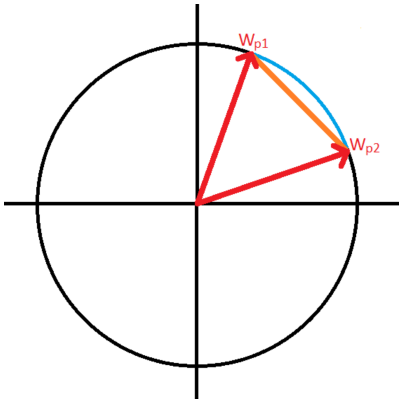
Teorema 4.3.2. *O operador de recombinação preserva a viabilidade da restrição de norma unitária do espaço de versões no sistema de representação de coordenadas hiperesféricas se os vetores pais tiverem norma unitária, isso é* $r_{hf} = 1$.

Demonstração. Sejam r_{hp1} e r_{hp2} o raio dos vetores pais com norma unitária. Então $r_{hp1} = 1$ e $r_{hp2} = 1$. Logo, $r_{hf} = r_{hp1} \cdot \lambda + r_{hp2} \cdot (1 - \lambda),\ \lambda \in [0, 1] = \lambda + (1 - \lambda) = 1$. □

Corolario 4.3.2.1. *Para qualquer vetor filho* W_f *sendo convertido do sistema hiperesférico para o sistema cartesiano, tem-se, independente dos valores das coordenadas,* $||W_f||_2 = r_{hf} = 1$.

Esses resultados demostram que os vetores filhos gerados no sistema hiperesférico preservam a norma unitária, portanto não é necessário a normalização dos novos indivíduos, ao contrário do que ocorreria no sistema cartesiano. A Figura 7 exemplifica essa propriedade de manter a norma unitária usando coordenadas hiperesféricas ao invés de se usar coordenadas cartesianas.

Figura 7 - Representação geométrica do operador de recombinação. A linha reta laranja mostra o conjunto de possíveis indivíduos gerados a partir da combinação direta dos vetores de peso. A linha curva azul mostra o possível indivíduo gerado pelo uso de coordenadas hiperesféricas.



Fonte: Própria produção.

Considerando que cada indivíduo representa uma direção, o operador de recombinação pode ser interpretado geometricamente como a escolha de uma nova direção entre as direções dos pais. Ao realizar a combinação convexa de peso sem essa representação não existiria a garantia de uma distribuição uniforme para todos os ângulos possíveis. Por exemplo, em um problema 2-dimensional, suponha que o primeiro pai seja um ângulo de

150° , enquanto o segundo pai é um ângulo de 0°. Se λ for definido como $0,75$, espera-se que o indivíduo filho tenha o ângulo de 120°. No entanto, se a recombinação fosse aplicada diretamente aos valores de peso, o ângulo resultante seria de aproximadamente 136°.

O operador de recombinação tem como objetivo fazer com que os indivíduos se assemelhem aos melhores da população, agindo assim como um acelerador, propagando informações relevantes de indivíduos antigos. Isso pode ser considerado uma forma primária de exploração. Por outro lado, os operadores de mutação, introduzidos na subseção seguinte, são responsáveis por explorar o espaço de busca, introduzindo uma diversidade aleatória, gerando novos indivíduos imparciais [21].

4.4 OPERADOR DE MUTAÇÃO

Cada mudança no vetor de pesos pode ser interpretada como uma rotação do hiperplano. O operador de mutação faz uma pequena alteração, causando assim uma pequena rotação no hiperplano. O sistema de coordenadas hiperesférico também é necessário para fazer uma mutação válida a fim de manter a norma unitária no indivíduo. Se os pesos fossem alterados diretamente, uma nova mutação poderia interferir em uma mutação anterior, resultando na perda da garantia de que quanto mais mutações aplicadas a um indivíduo, mais ela mudará. Portanto, para evitar esta solução de problemas, a representação de coordenadas hiperesférica deve ser usada.

Depois de aplicar o operador de mutação, a norma do vetor filho W_f permanece constante. Isto é comprovado pelo fato de que apenas os ângulos são modificados e o raio permanece o mesmo. Portanto, independentemente dos valores das coordenadas:

$$||W_f||_2 = r_{hf} = 1. \tag{4.9}$$

A intensidade da mutação é importante, se muito pequena, resultará em pequenas mudanças nos indivíduos, necessitando uma quantidade maior de iterações durante o processo de busca. No entanto, se for muito grande, maior do que o dobro da distância do indivíduo até o ponto ideal, fará com que o operador de mutação diminua o valor da margem. Por conta disso, foi utilizado um método para adaptar o valor da intensidade da mutação. Isto é feito usando um contador (ω). Se um filho tem uma margem melhor do que pelo menos um de seus pais, o contador é decrementado em 1 até o limite de zero ($\omega \geq 0$). Se um filho resultar em uma margem pior do que a de seus pais, o contador será incrementado em 1. Se o contador for maior que um valor limite, ele é zerado e o valor de intensidade de mutação é diminuído da seguinte maneira:

$$\mu_{i+1} \leftarrow \mu_i^{\tau} \tag{4.10}$$

com μ_{i+1} sendo a intensidade de mutação atualizada, μ_{i+1} a intensidade de mutação anterior, ambos em radianos pertencente ao intervalo $(0, 1)$, e τ um valor maior que 1. Isso permite uma redução rápida quando μ está perto de 1 e diminui a medida que μ se aproxima de 0. Experimentos indicaram que $1,8$ é um bom valor geral para determinar o parâmetro τ.

Diferentemente do operador de recombinação, o operador de mutação pode violar uma restrição de do espaço de versões tornando o vetor filho inviável. No entanto, em problemas linearmente separáveis, esse indivíduo produzirá um valor de margem negativa e será descartado da população. Em problemas quasi-linearmente separáveis, nos quais o espaço de versões está vazio, essa violação é relaxada e o algoritmo evolutivo selecionará os melhores indivíduos com menores valores de margem negativa, mais próximas de zero.

4.5 BALANCEAMENTO DO VIÉS

Os operadores de recombinação e mutação discutidos anteriormente não são aplicados ao parâmetro de viés. Após estes operadores serem aplicados a um indivíduo, o viés é calculado por uma técnica de balanceamento [22] capaz de encontrar o viés ótimo uma dada normal. Este procedimento utiliza da definição de margem, que depende da distância do hiperplano à instância mais próxima: se a distância do hiperplano à instância mais próxima de uma classe for diferente da distância da instância mais próxima à classe oposta, tem-se um hiperplano considerado *desbalanceado* e sua margem pode ser melhorada ao modificar o valor do viés para transladar o hiperplano de modo que a distância entre as duas classes seja a mesma. Isso é feito equacionando as duas distâncias mencionadas por meio de um procedimento que atualiza o parâmetro de viés, ou seja:

$$b = -\frac{1}{2} \cdot \langle W, S_+ + S_- \rangle \tag{4.11}$$

onde W é o vetor normal do hiperplano e S_+ e S_- as instâncias de cada classe que estão mais próximas a ele.

As figura 8 apresenta um exemplo da técnica de balanceamento sendo aplicada.

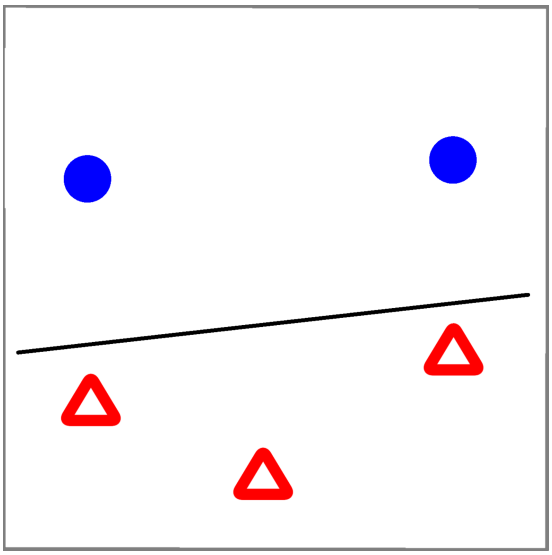
4.6 SUBSTITUIÇÃO DE INDIVÍDUOS

Para um indivíduo antigo ser substituído por um novo, ele deve concordar com três critérios a fim de manter a diversidade na população:

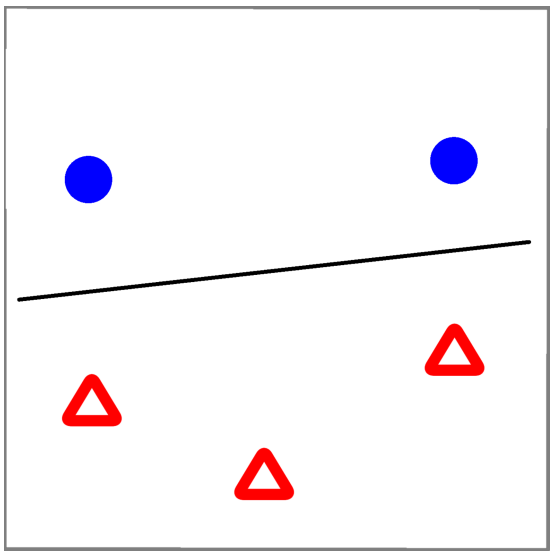
- Ter uma aptidão pior que o novo indivíduo.
- Para cada classe respectiva, o número de indivíduos que possuem a mesma instância mais próxima do indivíduo gerado, deve ser menor que o número de indivíduos que possuem a mesma instância mais próxima do indivíduo a ser substituído.

Figura 15 - Exemplo da técnica de balanceamento sendo aplicada a um conjunto de amostras, os triângulos vermelho são da classe negativa e as bolas azuis da classe positiva.

(a) Hiperplano inicialmente desbalanceado. A distância a classe negativa é menor que a distância a classe positiva.



(b) Hiperplano balanceado. O valor do viés foi modificado de modo que a distância da instância mais próxima da classe negativa ao hiperplano é a mesma que a distância da classe positiva ao mesmo.



Fonte: Própria produção.

- Ser aquele com a pior aptidão entre os que atendem à primeira e segunda condições.

Como o domínio do problema de maximização de margem é convexo, a velocidade do processo evolutivo pode aumentar ao aplicar o operador de recombinação numa população inicial diversa. Portanto, essa estratégia de substituição permite que o algoritmo tenha uma diversidade maior em sua população do que se apenas indivíduos de margem inferior fossem substituídos.

4.7 CONDIÇÃO DE PARADA

É importante estudar a relação entre a qualidade da margem de parada, γ^s, e o número de gerações, n, realizadas pelo algoritmo evolutivo para avaliar sua precisão. Considerando a margem ótima, γ^*, a qualidade da aproximação pode ser definida como:

$$\frac{\gamma^s}{\gamma^*} \cdot 100\% \tag{4.12}$$

Seja $\beta \in (0, 1)$ o parâmetro beta que controla uma β-aproximação da margem ótima, γ^*. Nesse sentido, uma margem de parada pode ser estimada considerando uma aproximação de β da margem ótima. Assim, o critério de parada do algoritmo deve satisfazer:

$$\gamma^s \geq (1-\beta)\cdot\gamma^*, \beta \in (0,1) \tag{4.13}$$

Para estimar o número de gerações necessárias para atingir a precisão desejada, precisa-se definir uma função de limite inferior que se aproxime assintoticamente das curvas de margem experimental produzidas pelo algoritmo evolutivo.

Em [44], os autores apresentam um algoritmo incremental, denominado *Incremental Margin Algorithm* (IMA), que aproxima a margem ótima resolvendo subproblemas sucessivos para aumentar os valores da margem reduzindo de forma monótona o hiper-volume do espaço de versões. As curvas de margem experimentais derivadas do IMA têm uma grande similaridade com as curvas de margem derivadas de nossos experimentos. A função de limite inferior apresentada em [44], é baseada no teorema de convergência do Perceptron de Margem Fixa. Esta função é definida como:

$$1 - \frac{C(Z_m)}{\sqrt{t}}, \tag{4.14}$$

onde t é o número acumulado de atualizações e $C(Z_m)$ um valor constante que depende do conjunto de treinamento. Deste modo, pode-se adaptar esse limite inferior para estimar o número de gerações, n, necessário para atingir uma aproximação de β. Portanto, o critério de parada deve satisfazer:

$$\gamma^s \geq (1-\beta)\cdot\gamma^* \geq 1 - \frac{C(Z_m)}{\sqrt{n}}\cdot\gamma^* \tag{4.15}$$

Para avaliar a constante $C(Z_m)$, realizamos experimentos extensivos usando conjuntos de dados artificiais variando a cardinalidade e a dimensionalidade de cada respectivo conjunto de treinamento. Por padrão, consideramos o valor constante igual a 1 (um). Este valor varia proporcionalmente com a dificuldade do problema. Portanto, com valores maiores que um, mais gerações são necessárias para obter uma aproximação β predefinida. Inversamente, com valores inferiores a um, menos gerações são necessárias.

Além disso, para uma melhor estimativa da precisão do algoritmo temos que definir um valor apropriado para ela . Nesse sentido, quanto mais alta é a curva do limite inferior, mais preciso e menor é o número de gerações necessárias para satisfazer a aproximação β.

Por exemplo, considerando $C(Z_M) = 2$ e $\beta = 0,1$, devemos ter uma margem de parada $\gamma^s \geq 0.9\cdot\gamma^*$. Então, deve-se ter o número de gerações $n \geq 400$.

4.8 EXPERIMENTOS

Os experimentos feitos para avaliar este algoritmo tem como objetivo mostrar o quão próximo da margem ótima ele consegue chegar, e quantas gerações são necessárias

dependendo da escala do problema, tanto em relação ao número de instâncias quanto de gerações. A próxima subseção apresenta as bases dedados utilizadas nos experimentos.

4.8.1 Bases De Dados

Com o objetivo de avaliar a eficiência em aproximar a margem ótima foram usados cinco conjuntos de dados linearmente separáveis. Estes conjuntos de dados foram retirados do *UCI Machine Learning Repository* [19]. A Tabela 1 mostra o número de amostras e dimensões para os conjuntos de dados linearmente separáveis.

Tabela 1 – Bases de dados linearmente separáveis.

Base de dados	Amostras	Dimensões
Sonar	208	60
Robot	117	90
Divorce	170	54
Binary Iris	150	4
Binary Synthetic	400	60

Devido a função objetivo utilizada aceitar margens negativas, foi também analisado como o algoritmo se comportaria em bases não linearmente separáveis. Para isto foram usados cinco conjuntos de dados quase linearmente separáveis, ou seja, cujo número de erros na fase de treinamento foi inferior a 20%. Este tipo de base de dados é ideal para avaliar como o algoritmo se comportaria em problemas que não sejam linearmente separáveis, porém na qual uma solução linear ainda tem utilidade. A Tabela 2 mostra o número de amostras e dimensões para os conjuntos de dados quase linearmente separáveis.

Tabela 2 – Bases de dados quase linearmente separáveis.

Base de dados	Amostras	Dimensões
Heart	270	13
Fertility	100	10
Synthetic	250	2
Ionosphere	351	34
Congressional	435	16

Além disso, foi construído um conjunto de dados artificiais para a análise empírica da precisão do algoritmo. Cada base teve suas amostras geradas de acordo com uma distribuição de probabilidade uniforme dentro de uma região hiper-quadrada centrada na origem. Em seguida, um hiperplano que passa pela origem, cuja normal fora gerada aleatoriamente, foi gerado e usado para rotular as amostras.

4.8.2 Aproximação da Margem

Os parâmetros usados nos experimentos foram: População inicial de 50 indivíduos; o valor de μ foi inicializado como $0,005$ radianos; τ foi definido como $1,8$; o limite ω foi definido como o número de amostras vezes 200.

O algoritmo evolutivo foi capaz de aproximar os resultados obtidos pelo *SVMLight* [35]. Os resultados para os as bases de dados linearmente separáveis são exibidos na Tabela 3, que mostra os valores de margem médio, máximo e mínimo após dez execuções, e o número de gerações e seu respectivo valor β.

Tabela 3 – Bases de dados linearmente separáveis.

Base de dados	**SVM**	**Evo-Med**	**Evo-Max**	**Evo-Min**	**n**	β
Sonar	0.00108	-0.02267	-0.01795	-0.02922	500	0.208
Robot	6.06832	3.972789	4.403806	3.810328	5000	0.274
Divorce	1.76787	1.493276	1.535260	1.429533	500	0.131
Binary Iris	0.82899	0.817424	0.817207	0.816898	50	0.013
Binary Synt	16.26809	15.650749	15.889338	15.453779	500	0.023

Para conjuntos de dados *quasi-linearmente separáveis*, utilizou-se o número de amostras classificadas incorretamente para medir a eficácia do algoritmo na classificação. A Tabela 4 mostra o número médio, mínimo e máximo de amostras classificadas incorretamente após 10 execuções de 200 gerações cada. O algoritmo SVM usava um kernel linear com margem flexível e o parâmetro de penalidade C foi escolhido automaticamente por *SVMLight*.

Tabela 4 – Bases de dados quase linearmente separáveis.

Base de dados	**SVM**	**Evo-Avg**	**Evo-Min**	**Evo-Max**
Heart	87	38.5	38	39
Fertility	63	10.5	8	12
Synthetic	124	63.1	63	64
Ionosphere	131	37.9	35	40
Congressional	81	17	16	19

Os experimentos mostram que o método proposto obteve boas aproximações do valor ótimo do SVM em todas as bases de dados linearmente separáveis, o que o revela como sendo uma alternativa viável para problemas cuja quantidade de instâncias pode tornar métodos duais inviáveis.

Em relação ás bases *quasi-linearmente separáveis* o método evolucionista obteve resultados superiores aos encontrados pelo SVM para todas as bases de dados. Estes excelentes resultados mostram que o método é bastante eficiente para este tipo de problema não linear, indicando um maior poder de generalização em sua solução.

4.8.3 Incremento de margem

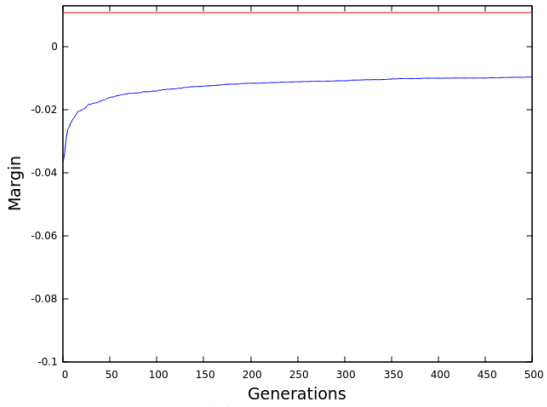
O número de gerações que o algoritmo executará é um parâmetro importante a ser analisado. Poucas gerações podem produzir um resultado ruim, enquanto muitas gerações será tempo despendido para pouca melhora no resultado. Devido a isso, analisou-se o

comportamento do algoritmo, à medida que as gerações progrediam, a fim de encontrar a melhor forma de definir o número ideal de gerações.

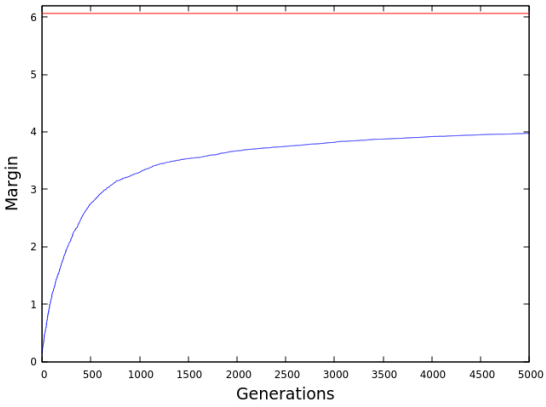
A análise consistiu em traçar as curvas experimentais relacionadas ao melhor indivíduo para cada geração nos conjuntos de dados linearmente separáveis. A Figura 9 mostra o incremento da margem dos melhores indivíduos em relação ao número de gerações em relação à margem ótima calculada pelo *SVMLight* ao final de sua execução.

Figura 9 - Incremento da margem em relação a encontrada pelo *SVMLight*.

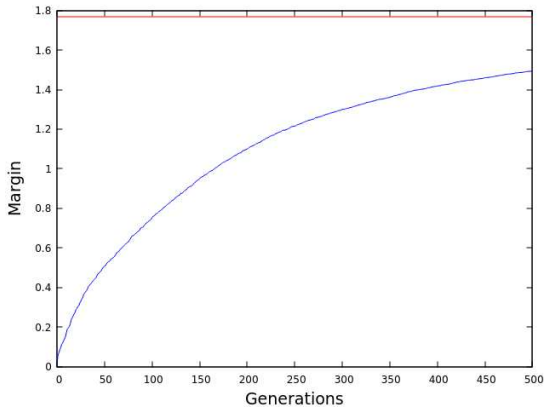
(a) Sonar



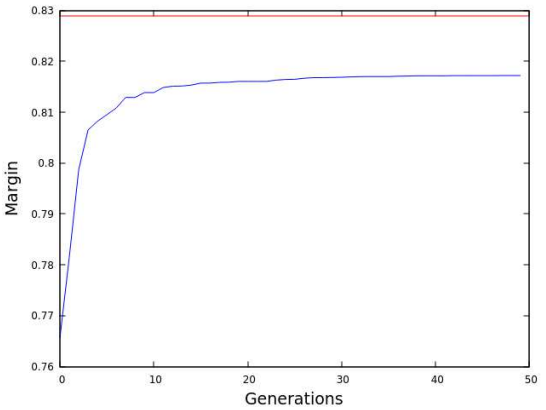
(b) Robot



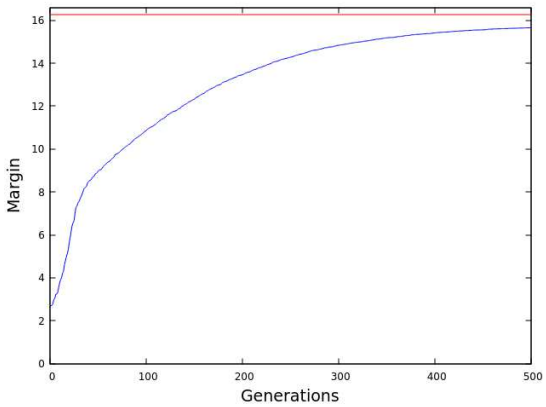
(c) Divorce



(d) Iris



(e) Synthetic



Fonte: Própria produção.

Com base nas curvas apresentadas pela Figura 9, pode-se estimar o número de gerações necessário para obter um aproximação esperada de margem ótima. Por exemplo, na base de dados Iris apresentado pela Figura 9(d), foi possível chegar a um resultado próximo ao do encontrado pelo SVM após apenas dez gerações. Em conjuntos de dados de mais dimensões, foram necessárias mais gerações para atingir uma margem próxima ao SVM, exceto para a base de dados Robot, Figura 9(b), onde a aproximação máxima foi em torno de 73%. Este valor é próximo ao valor obtido pelo algoritmo IMA [44] de 80%. Os gráficos mostram que durante as primeiras gerações, há um aumento significativo no valor da margem, e esse crescimento diminui à medida que a margem se aproxima do valor ótimo.

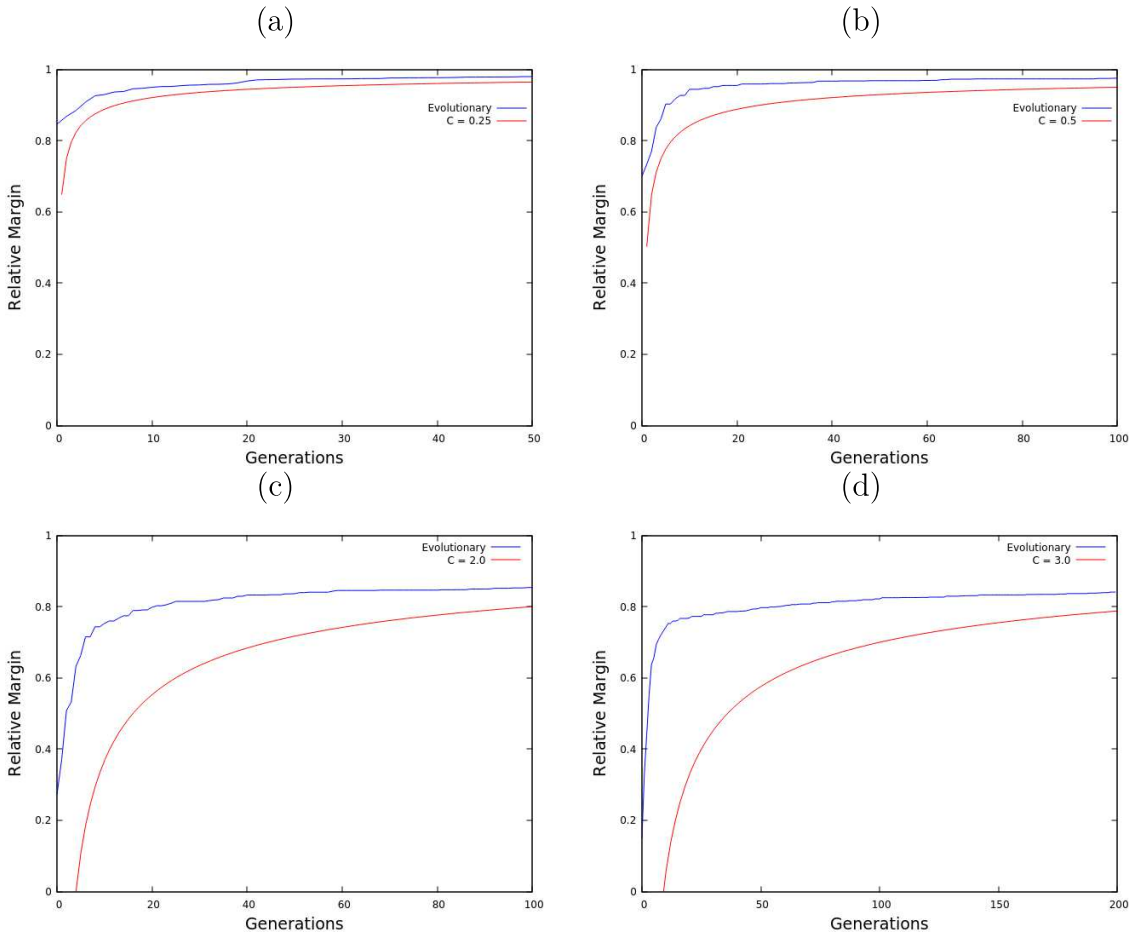
4.8.4 Análise de convergência

Ao analisar as curvas da Figura 9 é possível verificar que seguem um padrão semelhante em relação à sua convergência. Fazendo uso disso, foram feitos testes para verificar a influência do número de instâncias e/ou do número de dimensões na convergência e se uma função de limite inferior poderia ser usada para estimar as gerações necessárias para se obter uma boa aproximação.

A Figura 10 mostra a convergência do algoritmo para os conjuntos de dados artificiais com diferentes valores de dimensões para o mesmo número de instâncias. O eixo vertical representa a β-aproximação da margem ótima descrita pela função de limite inferior para cada conjunto de treinamento. A constante $C(Z_m)$ foi definida como o menor valor que assintoticamente se aproxima das curvas de margem experimental. Pode-se observar que o número necessário de gerações para atingir a β-aproximação aumenta proporcionalmente ao tamanho da dimensão também resultando em um incremento das constantes $C(Z_m)$.

A Tabela 5 mostra os valores $C(Z_m)$ relacionados à convergência do algoritmo nas bases de dados artificiais para um número diferente de amostras (linhas) e dimensões (colunas). A constante tende a diminuir conforme o número de amostras aumenta, refletindo o espaço de versões reduzindo. Além disso, em relação ao número de dimensões, ele tende a aumentar à medida que o número de dimensões aumenta, refletindo o espaço de versões expandindo. Assim, esta análise empírica indica que o método proposto lida melhor com problemas que possuem um número elevado de amostras. Isto indica que o método desenvolvido possui maior eficiência para bases de dados nas quais os métodos duais necessitam de um maior tempo de execução devido ao seu maior custo assintótico de tempo em relação a quantidade de amostras.

Figura 10 - **(a)** Base de dados de 1000 amostras com 5 dimensões. $C(Z_m)$ constante foi definido como $0,25$. **(b)** Base de dados de 1000 amostras com 10 dimensões. A constante $C(Z_m)$ foi definida como $0,5$. **(c)** Base de dados de 1000 amostras com 50 dimensões. A constante $C(Z_m)$ foi definida como $2,0$. **(d)** Base de dados de 1000 amostras com 250 dimensões. A constante $C(Z_m)$ foi definida como $3,0$.



Fonte: Própria produção.

Tabela 5 – Valores de $C(Z_m)$ encontrados para conjuntos de dados artificiais que variam pelo tamanho de dimensões e de amostras.

Amostras \ **Dimensões**	50	250	500	1000
50	1.1	1.75	3.0	3.25
250	1.4	1.6	2.2	2.7
500	1.1	1.5	2.0	2.4
1000	1.1	1.3	1.6	2.0

5 ALGORITMO EVOLUCIONISTA DE CENTRO ANALÍTICO (AECA)

A terceira contribuição deste trabalho é o desenvolvimento de um algoritmo evolutivo para aproximar a solução do problema do Centro Analítico. A partir do estudo realizado com o Algoritmo Evolucionista de Larga Margem, apresentado no capítulo anterior, viu-se potencial em aplicar uma estratégia evolutiva para aproximar a solução do problema do Centro Analítico. Assim desenvolveu-se um algoritmo para os casos em que a máxima margem não é recomendada, como mostrado na seção 2.4.

Uma breve descrição do algoritmo é apresentada a seguir. Cada indivíduo da população é um hiperplano representado por um vetor de pesos e parâmetro de viés. A população inicial é composta por um conjunto de hiperplanos factíveis treinados pelo algoritmo Perceptron. A função objetivo utiliza a função do centro analítico, apresentada em 2.10, sendo o objetivo do algoritmo minimizar este valor para o melhor indivíduo de sua população, ou seja, encontrar um que seja uma boa aproximação do centro de massa do espaço de versões. Novos indivíduos são gerados aplicando operadores de recombinação e mutação [53] para gerar seu vetor de pesos enquanto o viés tem seu valor otimizado por meio de um método de busca binária desenvolvido neste trabalho. A substituição de indivíduos ocorre após uma população intermediária ser gerada, seus membros são então utilizados para substituir os indivíduos da população principal, caso passem pelos critérios de substituição. Este processo de geração e substituição de indivíduos é repetido por um número fixo de vezes.

A visão geral do processo evolutivo de uma geração é similar ao do AELM, indicado previamente na Figura 6.

5.1 POPULAÇÃO INICIAL

A população inicial consiste em um conjunto de hiperplanos viáveis e normalizados, cada um gerado pelo algoritmo Perceptron. Uma vez que a população inicial é gerada através do sistema de coordenadas cartesianas, precisa-se converter cada indivíduo inicial para o sistema de coordenadas hiperesférico, isso é feito usando o algoritmo 1 apresentado anteriormente na seção 3.2.

5.2 FUNÇÃO OBJETIVA

A função objetivo usada para a avaliação de um indivíduo é baseada na função do centro analítico dada pela Equação 2.10. Portanto, a complexidade computacional para avaliar um indivíduo (W, b) é linear em relação ao número de exemplos do conjunto de treinamento. Esta função é dada por:

$$f(W,b) = -\sum_{i=1}^{m} \ln(Y_i \cdot (W \cdot X_i + b)), \tag{5.1}$$

considerando que $||W||_2 = 1$ para todos os indivíduos da população. É importante destacar que, neste caso, o argumento da função logaritmo não pode ter valores negativos, ou seja, o hiperplano necessita ser factível para que a função objetivo possa ser devidamente calculada. Desta forma, não é permitida a ocorrência de margens negativas.

5.3 OPERADOR DE RECOMBINAÇÃO

Um indivíduo filho é inicializado pelo operador de recombinação. Os pais são dois indivíduos da população, escolhidos de acordo com um processo de seleção por ranqueamento, onde a probabilidade de cada indivíduo é relativa à sua colocação em uma lista ordenada pelos valores de aptidão da população. A probabilidade de um indivíduo ser escolhido é dada por:

$$P_i = \frac{i^2 - (i-1)^2}{n^2}, \tag{5.2}$$

onde P_i é a probabilidade para o i-ésimo indivíduo da população e n é o tamanho da população.

Cada filho tem seus pesos inicializados por uma combinação linear convexa dos pesos de seus pais. Ao considerar a propriedade convexa do espaço de versões e a viabilidade da população inicial, esses indivíduos gerados também serão viáveis. O processo de recombinação é dado por:

$$W_{f,i} = \lambda \cdot W_{p1,i} + (1-\lambda) \cdot W_{p2,i},\ \lambda \in [0,1], \tag{5.3}$$

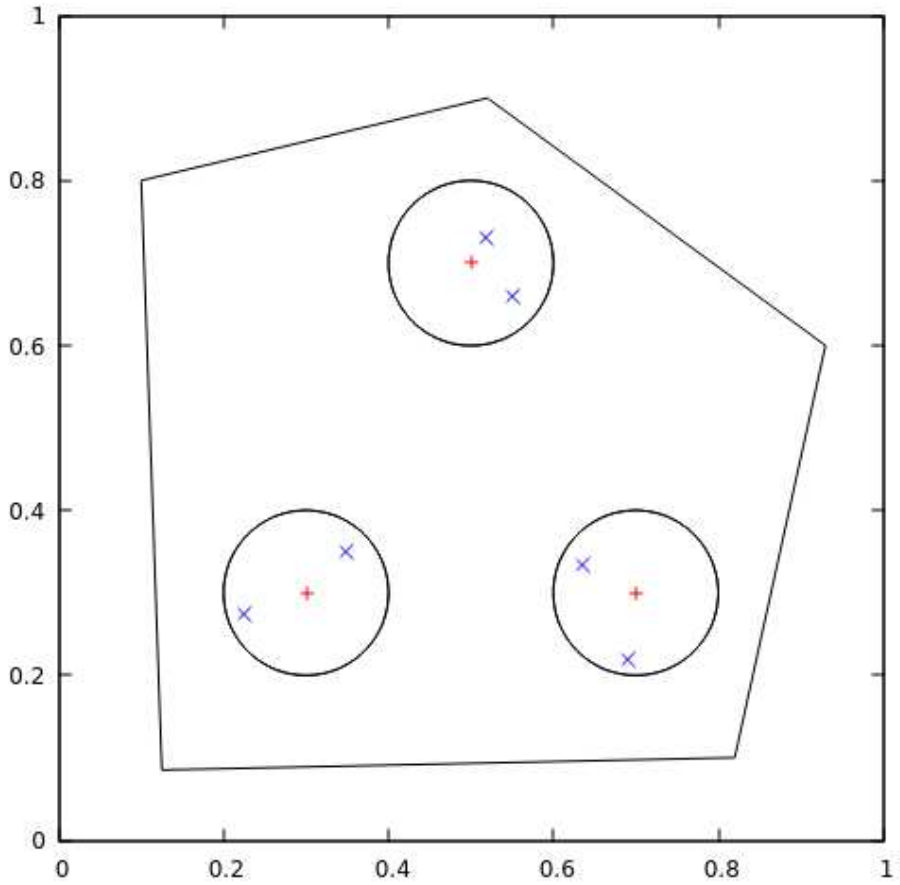
onde $W_{f,i}$, $W_{p1,i}$ e $W_{p2,i}$ são, respectivamente, os i-ésimos pesos dos hiperplanos da criança e dos pais. O parâmetro λ é escolhido a partir de uma distribuição de probabilidade uniforme. É importante destacar a necessidade de normalização dos indivíduos pais antes de aplicar a recombinação, a fim de preservar a viabilidade da restrição hiperesférica de raio unitário.

5.4 OPERADORES DE MUTAÇÃO

Para expandir o espaço de busca de modo a não ficar limitado pelo operador de recombinação foram criados operadores de mutação. Geometricamente eles podem ser interpretados no espaço de versões como um volume em torno do individuo sofrendo a mutação. a Figura 11 exemplifica isso.

Dois operadores de mutação são usados, um para realizar uma mudança pequena no hiperplano e outro para uma mudança maior. O primeiro operador de mutação causa

Figura 11 - Volume potencial de busca da operação mutação.



Fonte: Própria produção.

uma pequena rotação no hiperplano. A intensidade dessa mudança não tem um valor fixo, em vez disso, é definida como uma fração das distâncias de um progenitor para o outro progenitor. Isso permite que a intensidade da mutação seja escalonada conforme a população se aproxima do ponto ideal do espaço de versões. O primeiro procedimento do operador de mutação pode ser expresso pela seguinte equação:

$$\mu = \begin{cases} \tau \cdot (\sqrt{(\phi_{hp1} - \phi_{hp2})^2 + \sum (\theta_{hp1} - \theta_{hp2})^2}) & \text{se } |\phi_{hp1} - \phi_{hp2}| \leq \pi, \\ \tau \cdot (\sqrt{(2\pi - |\phi_{hp1} - \phi_{hp2}|)^2 + \sum (\theta_{hp1} - \theta_{hp2})^2}) & \text{caso contrário.} \end{cases} \tag{5.4}$$

com μ sendo a intensidade da mutação, em radianos, e τ o tamanho da fração. Como o ângulo ϕ é circular, a diferença entre os ângulos ϕ dos pais pode estar relacionada à direção oposta para a qual estão voltados, o que é o caso se a diferença resultar em um valor superior a π, portanto o valor correto é o seu ângulo de explicação.

O segundo operador de mutação também trabalha com as coordenadas hiperesféricas. Ele cria um vetor subtraindo as coordenadas hiperesféricas do pai com função objetivo inferior do outro, após isso, o indivíduo filho tem uma probabilidade de 2/3 de permanecer

o mesmo e uma probabilidade de 1/3 de ter este vetor adicionado a suas coordenadas hiperesféricas. Este operador permite explorar uma região maior do espaço de versões e ao mesmo tempo ter uma grande chance de manter a viabilidade devido a sua direção ser a mesma do operador de recombinação.

Ambos utilizam o sistema de coordenadas hiperesféricas devido as duas vantagens. A primeira é relacionada a norma do vetor de pesos, utilizar o sistema hiperesférico garantimos que ela não será alterada por quaisquer mudanças no vetor de ângulos. A segunda é para evitar que que uma mutação possa afetar em uma anterior, resultando na perda da garantia de que quanto mais mutações aplicadas a um indivíduo, mais ela mudará. Impedindo que um indivíduo, apesar de receber mutações, continue parecido a como era antes delas.

5.5 OTIMIZAÇÃO DO VIÉS

O parâmetro de viés é otimizado por um procedimento de pesquisa binária em seu intervalo viável, sendo este procedimento uma adaptação do procedimento de balanceamento do hiperplano separador utilizado no capítulo anterior, dado pela Equação 4.11. Como a função objetivo é uma soma de funções estritamente crescentes, qualquer mudança no valor de viés aumentará a soma das funções potenciais para uma classe enquanto diminui a soma da outra. Portanto, o valor de viés ideal é o ponto em que a soma das derivadas das funções potenciais para uma classe é igual à outra. Isso é resolvido considerando a minimização de uma função convexa uni-variada quando tomamos o conjunto de valores dos componentes do vetor W. Portanto, para um determinado indivíduo W_f, temos o objetivo:

$$\min_b f(b) = -\sum_{i=1}^{m} \ln(Y_i \cdot (W_f \cdot X_i + b)). \tag{5.5}$$

Derivando a Equação 5.5 em relação ao parâmetro b e considerando m^+ e m^- a cardinalidade das respectivas classes, é possível deduzir a condição de primeira ordem:

$$\sum_{i=1}^{m^+} \frac{1}{Y_i \cdot (W_f \cdot X_i + b)} = \sum_{i=1}^{m^-} \frac{1}{Y_i \cdot (W_f \cdot X_i + b)}, \tag{5.6}$$

que representa o equilíbrio entre o inverso das distâncias entre uma classe e outra em relação ao hiperplano de separação.

5.6 SUBSTITUIÇÃO DE INDIVÍDUOS

Para um indivíduo antigo ser substituído por um novo, ele deve concordar com os mesmos três critérios, apresentados no capítulo anterior, a fim de manter a diversidade na população:

- Ter uma aptidão menor que o novo indivíduo.
- Para cada classe respectiva, o número de indivíduos que possuem a mesma instância mais próxima do indivíduo gerado, deve ser menor que o número de indivíduos que possuem a mesma instância mais próxima do indivíduo a ser substituído.
- Ser aquele com menor aptidão entre os que atendem à primeira e segunda condições.

5.7 EXPERIMENTOS

Como apresentado na seção 2.4, o centro analítico representa uma boa aproximação do centro de massa, o que leva a classificadores com alto poder de generalização. Devido a isto, foram realizados experimentos no sentido de validar a qualidade da aproximação no que tange ao poder de generalização do classificador, neste caso, o Algoritmo Evolucionista de Centro Analítico. Seus resultados foram comparados com o atual estado da arte, apresentada pelos métodos descritos na Seção 2.6, utilizando bases de dados reais para tanto.

5.7.1 Bases de dados

Para analisar o desempenho do algoritmo foram utilizados seis conjuntos de dados linearmente separáveis. Estas bases foram retiradas de aplicações oriundas de analises de *microarray*. Essas bases de dados são referenciados por [3], [26], [66] e [73]. A Tabela 6 mostra as informações relativas a estas bases:

Tabela 6 – Descrição das bases de dados utilizadas.

Bases	Dimensões	Amostras		
		$+1$	-1	Total
Prostate	12600	50	52	102
Breast	12625	10	14	24
Colon	2000	22	40	62
Leukemia	7129	47	25	72
DLBCL	5468	58	19	77
CNS	7129	21	39	60

5.7.2 Resultados

Os algoritmos SVM [35], BPM [31] e VSRM [22] empregaram uma validação cruzada de 10x10-10 vezes, devido a aleatoriedade envolvida nestes métodos. A única exceção foi o SVM, onde foi empregada uma validação cruzada de 1x10-10 vezes. Esses testes foram adotados a fim de reduzir o enviesamento nos resultados que poderia ocorrer devido a aleatoriedade deles, e assim conseguir um resultado próximo da média. Foi

utilizada uma estratégia de validação cruzada estratificada, onde cada partição mantém a porcentagem de pontos de dados de cada classe. Para comparações mais precisas foram sempre selecionados, para cada algoritmo, os mesmos conjuntos de treinamento e teste e também os mesmos 10 subconjuntos para validações cruzadas, preservando a semente geradora associada ao processo aleatório.

Os resultados para SVM, BPM e VSRM foram retirados de [22], com a semente aleatória usada nos experimentos sendo a mesma. Para o Algoritmo Evolucionista de Centro Analítico (AECA), foi utilizada uma população de 100 indivíduos evoluídos por 50 gerações, com o primeiro operador de mutação tendo uma probabilidade de 10% de ser aplicado em cada dimensão. A Tabela 7 mostra os resultados obtidos nos experimentos. Os melhores resultados para cada conjunto de dados são destacados em negrito.

Tabela 7 – Média, e desvio padrão, de erros de classificação.

Bases	SVM	BPM	VSRM	EACC|
Prostate	9.58 ± 1.35	10.46 ± 1.57	9.46 ± 1.77	**8.75 ± 1.05**
Breast	20.67 ± 2.49	20.35 ± 3.42	19.83 ± 2.39	**19.33 ± 1.54**
Colon	18.69 ± 2.32	15.68 ± 2.35	15.44 ± 2.29	**15.29 ± 2.57**
Leukemia	**2.75 ± 0.88**	5.50 ± 1.32	3.39 ± 1.30	4.91 ± 1.52
DLBCL	3.81 ± 0.68	3.86 ± 0.79	**3.49 ± 0.87**	4.38 ± 1.20
CNS	33.50 ± 1.99	33.33 ± 2.68	32.87 ± 2.15	**32.33 ± 3.02**

Os resultados mostram que o AECA obteve os melhores resultados em 4 das 6 bases de dados usadas, sendo superado pelos *SVM* e *VSRM* apenas duas das seis bases da dados. Deste modo, o algoritmo proposto conseguiu melhores resultados que as outras abordagens na maioria das bases de dados, o que indica ele como uma boa opção a ser utilizada em um problema real desconhecido, quanto para possivelmente obter resultados superiores aos produzidos por outros métodos anteriormente.

Os próximos capítulos tem como objetivo abordarem o segundo tópico desta tese, relacionado a análise de relevância das amostras e análise de separabilidade do problema de classificação binária. Antes, entretanto, serão feitas considerações a respeito da implementação dual dos algoritmos AELM e a AECA ressaltando as suas limitações no contexto da representação esférica e da computação evolucionista.

5.8 IMPLEMENTAÇÃO DUAL

Como comentado anteriormente, a versão dual dos algoritmos AELM e AECA, não apresentaram bons resultados em termos de acurácia. Isto se deve principalmente a dois fatores:

- O maior custo computacional no valor da função de aptidão dos indivíduos.
- Pior distribuição dos indivíduos no espaço dual de versões.

Primeiramente, será discutido o maior custo computacional. Como o espaço dual de versões para formulação adotada não possui uma casca esférica, no sentido de preservar a norma unitária do vetor normal, não faz sentido representar o vetor α dos indivíduos da população pelo sistema de coordenadas hiper-esféricas. Consequentemente, para o cálculo de margem geométrica de cada indivíduo, tornou-se necessário o cálculo da norma em função do vetor α utilizando-se a Equação 2.13. Esta complexidade é quadrática em relação à quantidade de amostras dado a existência do duplo somatório. Além disso, não é possível inferir diretamente a norma de um indivíduo filho da norma de seus respectivos pais, devendo a mesma sendo computada para cada indivíduo gerado pelo processo de recombinação.

O segundo ponto está relacionado a representatividade das operações realizadas no espaço de entrada no espaço de características. Mesmo que fosse utilizado um sistema de coordenadas cartesianas para representação dos indivíduos, a operação de recombinação feita no espaço de entrada não seria reproduzida corretamente no espaço de características. A reprodução do produto interno dos vetores nos espaços de características por uma função kernel é uma operação válida desde que esta função atenda a condição de Mercer [49]. Entretanto, a execução da operação linear entre vetores necessários ao operador de recombinação não é uma operação válida no espaço de características, provocando uma má distribuição da população neste espaço. Essas duas situações são demonstradas numericamente a seguir:

Sejam α_1, $\alpha_1 \in \mathbf{R}^2$ vetores de uma população representados respectivamente pelas coordenadas cartesianas (x_1, y_1) e (x_2, y_2). Seja $\alpha_f \in \mathbf{R}^2$ um individuo gerado a partir da aplicação do operador de recombinação nos vetores pais α_1 e α_2. Portanto, no espaço de entrada, tem-se:

$$\alpha_f = \lambda\alpha_1 + (1-\lambda)\alpha_2, \text{ onde } \lambda \in (0,1) \tag{5.7}$$

Na respectiva representação vetorial no espaço $\mathbb{R}^2$ tem-se:

$$\alpha_f = \lambda(x_1, y_1) + (1-\lambda)(x_2, y_2) = (\lambda x_1 + (1-\lambda)x_2, \lambda y1 + (1-\lambda)y_2), \text{ onde } \lambda \in (0,1) \tag{5.8}$$

Seja, também, uma função kernel quadrática aplicada em $\mathbb{R}^2$, cujo mapeamento pode ser representado explicitamente no espaço $\mathbb{R}^3$ pelos seguintes vetores β característicos:

$$\beta_1 = (x_1^2, \sqrt{2} \cdot x_1 \cdot y_1, y_1^2) \text{ e } \beta_2 = (x_2^2, \sqrt{2} \cdot x_2 \cdot y_2, y_2^2) \ , \tag{5.9}$$

os quais produzem, no respectivo espaço, o seguinte vetor filho, fruto da aplicação do operador de recombinação:

$$\beta_f = \lambda(x_1^2, \sqrt{2} \cdot x_1 \cdot y_1, y_1^2) + (1-\lambda)(x_2^2, \sqrt{2} \cdot x_2 \cdot y_2, y_2^2) \text{ , onde } \lambda \in (0,1) \tag{5.10}$$

Como afirmado anteriormente, a reprodução do produto interno dos vetores característicos no espaço $\mathbb{R}^3$, para uma função kernel válida, pode ser realizada implicitamente no espaço de entrada considerando, como argumento da função kernel, o produto interno dos vetores associados. Operação esta denominada *kernel trick*. De fato, para uma função kernel quadrática:

$$\begin{aligned} \langle(\beta_1, \beta_2)\rangle = \langle(\alpha_1, \alpha_2)\rangle^2 = \langle(x_1^2, \sqrt{2} \cdot x_1 \cdot y_1, y_1^2), (x_2^2, \sqrt{2} \cdot x_2 \cdot y_2, y_2^2)\rangle = \\ = \langle(x_1, y_1), (x_2, y_2)\rangle^2 = x_1^2 \cdot x_2^2 + 2 \cdot x_1 \cdot x_2 \cdot y_1 \cdot y_2 + y_1^2 \cdot y_2^2 \text{ , onde } \lambda \in (0,1) \end{aligned} \tag{5.11}$$

Entretanto, esta operação implícita não é válida para o operador de recombinação. Se for realizada esta operação no espaço de entrada e o consequente mapeamento deste vetor filho, dado ela Equação 5.8, para o espaço de características, tem-se:

$$\beta_f = ((\lambda \cdot x_1 + (1-\lambda)x_2)^2, \sqrt{2} \cdot (\lambda x_1 + (1-\lambda)x_2) \cdot (\lambda y_1 + (1-\lambda)y_2)), ((\lambda y_1 + (1-\lambda)y_2)^2), \tag{5.12}$$

produzindo uma representação incorreta diferente da representação correta descrita pela Equação 5.10. Consequentemente, a geração de uma nova população não ocorre da maneira adequada, levando a uma distorção da distribuição da mesma no espaço de características.

Por outro lado, se fosse necessário avaliar o produto interno do vetor filho com um outro vetor no espaço de características isto poderia ser feito de forma implícita pois operações envolvendo a soma de funções kernel ou a multiplicação de uma função kernel por um escalar são operações válidas.

Sejam, por exemplo, um vetor $\beta_3 \in \mathbf{R^2}$ com coordenadas (x_3, y_3), o vetor filho $\beta_f \in \mathbf{R^2}$ e uma função kernel quadrática. A realização do produto interno entre estes dois vetores no espaço de características produz como resultado:

$$\begin{aligned} \langle \alpha_3, \alpha_f \rangle = \langle (x_3^2, \sqrt{2} \cdot x_3 \cdot y_3, y_3^2), ((\lambda^2 \cdot x_1^2 + (1-\lambda)^2 \cdot x_2^2 + 2 * \lambda \cdot x_1 \cdot (1-\lambda) \cdot x_2), \\ \lambda \cdot \sqrt{2} \cdot x_1 \cdot y_1 + (1-\lambda) \cdot \sqrt{2} \cdot x_2 \cdot y_2, ((\lambda^2 \cdot y_1^2 + (1-\lambda)^2 \cdot y_2^2 + 2 \cdot \lambda \cdot y_1 \cdot (1-\lambda) \cdot y_2))\rangle \end{aligned} \tag{5.13}$$

Equivalentemente, este resultado pode ser obtido implicitamente a partir do espaço de entrada. Para tanto, considera-se, inicialmente, a avaliação do produto interno no espaço de entrada entre o individuo filho α_f , dado pela Equação 5.8, e o vetor α_3 com coordenadas (x_3, y_3). Assim:

$$\langle \alpha_3, \alpha_f \rangle = \langle (x_3, y_3), (\lambda \cdot x_1 + (1-\lambda) \cdot x_2, \lambda \cdot y_1 + (1-\lambda) \cdot y_2) \rangle \tag{5.14}$$

Que é igual a:

$$\lambda \cdot (x_1 \cdot x_3 + y_1 \cdot y_3) + (1-\lambda) \cdot (x_2 \cdot x_3 + y_2 \cdot y_3) \tag{5.15}$$

Aplicando a este argumento a função quadrática tem-se:

$$\langle \alpha_3, \alpha_f \rangle^2 = (\lambda \cdot (x_1 \cdot x_3 + y_1 \cdot y_3) + (1-\lambda) \cdot (x_2 \cdot x_3 + y_2 \cdot y_3))^2, \tag{5.16}$$

o qual resultará no mesmo resultado dado pela Equação 5.13. Entretanto, esta forma de avaliação implícita que envolve operações lineares no espaço de entrada não se aplica ao algoritmo evolutivo onde é necessário a representação dos indivíduos pelos seus respectivos vetores de coordenadas para a geração de uma nova população.

6 ANÁLISE DE RELEVÂNCIA

A quantidade de amostras no conjunto de treinamento é um fator importante para o tempo de execução de um algoritmo de classificação. Portanto, a remoção de amostras não relevantes produz uma redução no tempo de execução, principalmente na formulação dual, cuja complexidade é quadrática em relação a quantidade de amostras. Neste sentido, a segunda parte desta pesquisa tem como objetivo estudar formas de remover amostras de modo a não afetar negativamente o resultado final dos classificadores. Este estudo também gerou desdobramentos como a descoberta de uma nova relação com a área de Geometria Computacional, e a uma nova forma de análise da separabilidade de um conjunto de amostras em problemas de classificação binária.

Detectar quais amostras são as mais importantes para gerar um bom classificador foi um tema importante deste estudo, porém abordado de forma superficial durante o desenvolvimento dos algoritmos evolutivos com implementação primal apresentados nos capítulos anteriores. Entretanto, percebeu-se as vantagens em reduzir o tamanho do conjunto de treinamento, tanto por diminuir o tempo de execução do algoritmo, quanto para aprimorar o processo de geração produzindo indivíduos com melhor aptidão.

Ao analisar a evolução da população no algoritmo *AELM*, percebeu-se que um sub-conjunto das amostras de treinamento aparecia frequentemente como responsáveis pela definição da margem dos indivíduos, enquanto outro sub-conjunto de amostras raramente aparecia. A partir desta observação foi utilizada uma heurística para remover as amostras pertencentes a este segundo sub-conjunto. Apesar de eficiente, esta heurística não garante a remoção de todos as amostras que poderiam ser eliminados, além de existir um risco de remover amostras que não deveriam sair do conjunto de treinamento.

Neste sentido, o processo de desenvolvimento destes dois algoritmos levou ao interesse e ao entendimento da importância do problema de remoção de amostras não relevantes do conjunto de treinamento. Portanto, teve-se como foco o desenvolvimento de heurísticas de remoção que não possuíssem o risco de remover amostras relevantes para a formação do espaço de versões. Contudo, após uma série de estudos e experimentos, obteve-se uma solução mais apropriada para o problema culminando com o desenvolvimento de um novo método. Esta solução foi criada por meio da utilização de conceitos das áreas de Aprendizado de Máquinas e Geometria Computacional.

Foram observadas novas relações entre os problemas de classificação binária e de detecção de pontos extremos em uma nuvem de pontos. Estas relações levaram a formulação de conceitos, a serem apresentados neste capítulo, que culminaram em novas perspectivas para a abordagem e solução de problemas de ambas as áreas.

Inicialmente, serão apresentados os conceitos e definições necessários para classificar as amostras como relevantes ou não relevantes. Em seguida, serão apresentadas as defini-

ções matemáticas de pontos extremos, seguido de formulações matemáticas comumente utilizadas para detectar se determinada amostra é relevante ou não. Também, será apresentada uma formulação matemática para detectar se um determinado ponto é extremo ou não. Finalmente, será apresentada a relação entre os problemas de classificação binária e de detecção de pontos extremos por meio de uma redução de um problema ao outro. Ou seja, será mostrado que ao se resolver um dos problemas, também consegue-se resolver o outro produzindo a mesma solução. A partir desta redução é possível induzir um novo problema de classificação sem rótulos, ou seja, apresentando somente uma classe. Este novo problema, que consiste em uma nuvem de pontos, possibilitou o desenvolvimento de um algoritmo determinístico e finito, denominado de Oráculo Ponto Extremo, denominado Oráculo PE, a ser utilizado para a detecção de amostras não relevantes.

Será também apresentado uma aplicação adicional para este algoritmo, sendo ela a geração de uma população inicial para métodos populacionais ou um comitê de classificadores, devido a sua capacidade de gerar, como um subproduto, um classificador viável relacionado a cada amostra relevante encontrada.

Porém, devido a quantidade de novos conceitos a serem introduzidos, e a complexidade do procedimento desenvolvido, será apresentado inicialmente uma visão geral do funcionamento do mesmo.

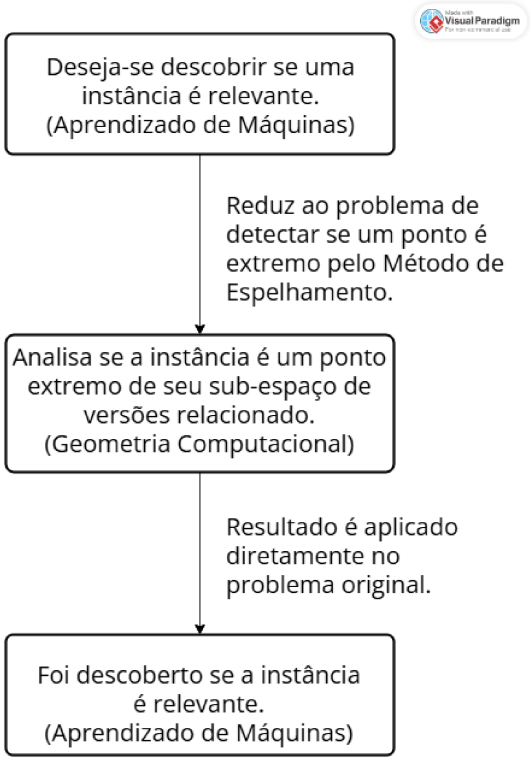
6.1 VISÃO GERAL DO PROCEDIMENTO

O problema a ser resolvido consiste em: dado um problema de classificação binária; definir se uma amostra pertencente ao conjunto de treinamento é necessária, ou não, para a formação do espaço de versões. O procedimento desenvolvido para resolver este problema é complexo, sendo composto por vários etapas envolvendo conceitos de duas áreas: Aprendizado de Máquinas e Geometria Computacional. Devido a isto, entender como o procedimento funciona de modo geral facilita a compreender à importância de cada etapa e a relação entre estas duas áreas. Esta seção visa apresentar uma visão geral do procedimento ao explicar resumidamente o funcionamento de cada etapa. O diagrama da Figura 12 apresenta uma abstração das etapas do procedimento.

Formalmente deseja-se descobrir, a partir de um conjunto de instâncias $\mathcal{X}$, se uma instância arbitrária $X_p \in \mathcal{X}$ é não relevante na definição do espaço de versões do conjunto $\mathcal{X}$. Uma instância ser não relevante no espaço de versões significa que ao ser retirada do conjunto, isto não influenciará no conjunto de classificadores factíveis, e por consequência, não afetará negativamente a qualidade de um classificador que maximize uma das métricas previamente discutidas como margem ou centro analítico. Deste modo, ao retirar do conjunto de treinamento uma instância não relevante, é possível diminuir o tempo de execução de um algoritmo sem que isto afete negativamente o resultado ótimo.

A execução do procedimento pode ser resumida em três etapas, a primeira sendo

Figura 12 - Visão geral do funcionamento do procedimento de detecção de pontos relevantes, relevando a relação entre as áreas de Aprendizado de Máquinas e Geometria Computacional.



Fonte: Própria produção.

uma redução do problema de detectar a redundância de uma instância no problema de classificação binária ao problema de detectar se um ponto pertencente a uma nuvem de pontos é extremo. Após a redução ser feita, a segunda etapa usa um algoritmo desenvolvido neste trabalho para detectar se o ponto é extremo. A terceira etapa então usa o resultado encontrado na segunda parte como solução do problema inicial de detectar se o ponto é não relevante na formação do espaço de versões.

O primeiro passo do procedimento é reduzir o problema de classificação ao de descobrir se um ponto pertencente a uma nuvem é extremo nela, este segundo problema pertence a área de Geometria Computacional. A principal diferença entre estes dois problemas está no fato de o problema de classificação binária ser formado por um conjunto de instâncias que podem pertencer a duas classes diferentes, enquanto no problema de definição de pontos extremos não existirem classes. Para realizar a redução de um problema ao outro foi criado uma técnica chamada de *espelhamento*, que será descrita em detalhes na seção 6.4 e sua prova no Teorema 6.4.1. Resumidamente, esta técnica permite, a partir de um conjunto de instâncias $\mathcal{X}$, e uma instância qualquer $X_p \in \mathcal{X}$ definida como pivô, retirar do conjunto uma instância cuja classe é oposta a do pivô e adicionar outra que possua a mesma classe dele, porém garantindo que esta mudança não afete o conjunto de hiperplanos factíveis que passam pela posição do pivô. A redução consiste, de forma

resumida, em inicialmente aplicar esta técnica a todas as instâncias de classe diferente do pivô, de modo que o conjunto resultante de pontos seja formado apenas por instâncias de mesma classe, assim removendo a principal diferença entre os dois problemas, enquanto garantindo que um subconjunto dos hiperplanos factíveis continue o mesmo. A redução prossegue utilizando as definições de ponto extremo apresentadas na Seção 6.3. A partir delas pode-se garantir que caso o ponto pivô seja um ponto extremo do conjunto após o espelhamento, então existe um hiperplano que o possui, e que agrupa todos os outros pontos em um mesmo semi-espaço. Como o conjunto de hiperplanos que possuem o pivô não é alterado pela técnica de espelhamento, então pode-se afirmar que caso o pivô seja um ponto extremo, então existe um hiperplano factível que o possui, logo ele é relevante para a formação do espaço de versões do conjunto original.

Após a técnica de espelhamento ser realizada em todas as instâncias de classe diferente do pivô, o próximo passo é descobrir se ele é um ponto extremo do novo conjunto. A área de Geometria Computacional possui algoritmos para resolver este problema, porém como as aplicações de Geometria Computacional envolvem tradicionalmente domínios de apenas duas ou três dimensões, os algoritmos atuais não escalam bem com aumento da dimensionalidade, o que os torna impraticáveis para uso em problemas de Aprendizado de Máquinas. Devido a isto foi necessário criar um algoritmo eficiente para altas dimensões. Este algoritmo será explicado em detalhes na Seção 7.2. O algoritmo desenvolvido utiliza a propriedade de convexidade do espaço de versões para verificar a existência de um hiperplano que atenda as duas seguintes condições: Conter o pivô, e separar todos os pontos em um mesmo semi-espaço. O algoritmo itera entre hiperplanos que contenham d pontos do conjunto, sendo um deles o pivô, procurando minimizar a cada iteração a distância ao hiperplano dos pontos localizados no semi-espaço vazio. Este processo ocorre por uma busca gulosa até encontrar uma face da nuvem de pontos que contenha o pivô, ou até chegar no ótimo local, o que pela propriedade de convexidade, garante não existir um hiperplano que contém o pivô e que separa todos os pontos num mesmo semi-espaço.

O passo final do procedimento envolve retornar ao problema original de classificação binária, onde o objetivo é definir se a instância escolhida como pivô é relevante. Assim, como será demonstrado nas seções seguintes: Caso no passo anterior seja encontrado um hiperplano que atenda as duas condições mencionadas no parágrafo anterior, então o ponto escolhido como pivô é relevante, e o hiperplano é uma solução factível do problema original de classificação. Para o caso de não existir um hiperplano factível que passe pelo pivô, então o mesmo é redundante, ou seja, caso o pivô seja removido do conjunto de instâncias, isto não afetará a existência de nenhum dos hiperplanos factíveis, ou seja, não afetará o espaço de versões.

As próximas seções detalharão o funcionamento de cada parte do procedimento. Primeiro serão formulados os conceitos e definições necessárias para o entendimento do funcionamento do procedimento, após isto será descrito em detalhes o processo de redução

e o funcionamento do algoritmo de detectar pontos extremos.

6.2 DEFINIÇÃO DE AMOSTRAS RELEVANTES E NÃO RELEVANTES

Primeiramente é necessário definir formalmente quais as condições que devem ser atendidas para que uma instância possa ser removida seguramente do conjunto de treinamento. O principal aspecto a considerar numa remoção é se isto afetará a qualidade do resultado final. Deste modo, procura-se remover apenas aquelas que não interfiram na qualidade do melhor classificador, ou seja, as instâncias que não afetem a geometria do espaço de versões. Considerando que o espaço de versões é formado por um conjunto de restrições, cada uma relacionada a uma instância, não queremos remover as instâncias responsáveis por formar as faces do espaço de versões. Com isso define-se uma instância relevante como:

Definição 6.2.1. *Dado um espaço de versões não vazio V formado por um conjunto de instâncias $\mathcal{X}$, uma instância $X_i \in \mathcal{X}$ é considerada relevante em $\mathcal{X}$ quando sua existência afeta o espaço de versões V.*

Sendo a definição do sub-conjunto de amostras relevantes dada por:

Definição 6.2.2. *Dado um conjunto de instâncias $\mathcal{X}$, o sub-conjunto formado por todas as instâncias pertences a $\mathcal{X}$ que sejam relevantes em $\mathcal{X}$ é definido como sub-conjunto $\mathcal{X}_r$*

Equivalentemente podemos definir uma amostra não relevante por:

Definição 6.2.3. *Dado um espaço de versões não vazio V formado por um conjunto de instâncias $\mathcal{X}$, uma instância $X_i \in \mathcal{X}$ é considerada não relevante em $\mathcal{X}$ quando sua existência não afeta o espaço de versões formado por $\mathcal{X}$.*

Definição 6.2.4. *Dado um espaço de versões vazio V formado por um conjunto de instâncias $\mathcal{X}$, todas as instâncias $X_i \in \mathcal{X}$ são consideradas não relevantes.*

E para o subconjunto de amostras não relevantes:

Definição 6.2.5. *Dado um conjunto de instâncias $\mathcal{X}$, o sub-conjunto formado por todas as instâncias pertences a X que sejam não relevantes em $\mathcal{X}$ é definido como sub-conjunto $\mathcal{X}_n$*

É importante ressaltar que estes subconjuntos são complementares em relação ao conjunto total de amostras. Ou seja: $\mathcal{X}_r \cup \mathcal{X}_n = \mathcal{X}$ e $\mathcal{X}_r \cap \mathcal{X}_n = \emptyset$

A partir da interpretação geométrica do espaço de versões, pode-se definir um oráculo para detectar, de uma forma mais restrita e eficiente, se uma instância é relevante ou não. Considerando que a representação geométrica do espaço de versões consiste de

um cone poliédrico, com cada uma de suas faces associada a uma amostra relevante do conjunto de treinamento, cada ponto ou vetor deste cone representa uma solução viável do problema de classificação associado, constituindo o vetor normal de um hiperplano separador. Assim, pode-se assegurar que todos pontos pertencentes as faces delimitadoras também representam soluções viáveis do problema. Sendo assim, pode-se concluir que cada amostra relevante do conjunto de treinamento possuirá pelo menos um hiperplano viável que contém a mesma.

Deste modo, para uma instância ser relevante, é necessário que exista ao menos um classificador viável cujo hiperplano contenha a mesma.

Seguindo esta análise, constata-se que, para uma dada amostra, caso seja verificado que não exista pelo menos um classificador ou hiperplano viável que a contenha, então a restrição imposta pela mesma não forma uma das faces que delimita o espaço de versões. Neste sentido, pode-se afirmar que todas amostras suportes são amostras relevantes seguindo a definição do oráculo. Isto é facilmente verificado pelo fato das amostras suportes estarem contidas nos respectivos hiperplanos canônicos que delimitam as margens de separação.

No contexto de um sistema de inequações lineares, relacionado a formulação do algoritmo Perceptron, a verificação se uma amostra é não relevante para o sistema e, portanto, redundante, pode ser feita com o uso da técnica de relaxação para a solução da viabilidade de sistemas de inequações lineares [24, 28]. Seja a amostra (X_k, Y_k). Para verificar se a mesma é redundante para o sistema invertemos o valor do rótulo e acrescentamos a nova restrição, como indicado pela Equação 6.1, no sistema de inequações que define o espaço de versões retirando a inequação original referente a amostra (X_k, Y_k).

$$-Y_k \cdot (< W, X_k > +b) \geq 0 \tag{6.1}$$

Se este novo sistema possuir pelo menos uma solução então a amostra não é redundante e a sua face associada delimita o espaço de versões. Caso contrário, se o sistema não tiver solução, a mesma é redundante podendo ser eliminada. Do ponto de vista da geometria do espaço de versões pode-se considerar esta solução como a inversão do vetor normal da face definida pelo hiperplano: $Y_k \cdot (< W, X_k > +b) = 0$. Assim, se não houver pelo menos um ponto W para o qual: $-Y_k \cdot (< W, X_k > +b) \geq 0$, a região de viabilidade associada ao espaço de versões será vazia. Esta solução por contradição é mais comumente conhecida como “Apagoge” e possui um grande problema relacionado ao fato deste oráculo não ser finito. Tratando-se da solução de viabilidade de um sistema de inequações por um processo de relaxação, caso o sistema não possua solução, o oráculo entrará em loop não fornecendo a resposta desejada. Para solucionar este problema pode ser considerado o uso de um oráculo baseado em programação linear. Para tanto, basta remover do sistema de inequações a restrição: $Y_k \cdot (< W, X_k > +b) \geq 0$. Em seguida,

resolver o problema de PL indicado pela Equação 6.2 sem a restrição removida que passa a ser a função objetivo.

$$\begin{aligned} &\text{Max}_{(W,b)} f(X_k, Y_k) = Y_k \cdot (< W, X_k > +b) \\ &\text{Sujeito a:} \\ &Y_1 \cdot (< W, X_1 > +b) \geq 0 \\ &... \\ &Y_m \cdot (< W, X_m > +b) \geq 0 \end{aligned} \tag{6.2}$$

Se $f^*(X_k, Y_k) < 0$ então a amostra (X_k, Y_k) pode ser removida . Caso contrário, se $f^*(X_k, Y_k) \geq 0$ então a amostra não é redundante.

Este segundo oráculo, denominado Oráculo PL, embora finito em tempo de execução, exige um grande esforço computacional, pois possui complexidade relacionada a matriz de coeficientes do conjunto de restrições a qual está associada não somente a quantidade de amostras, mas também a dimensão do problema.

Neste sentido, justifica-se a criação de um novo método ou oráculo para a solução do problema de detecção de amostras relevantes. Este novo oráculo será denominado Oráculo Ponto Extremo ou simplesmente Oráculo PE. O fato deste oráculo proposto ser mais restrito pode ser explicado pelo seguinte raciocínio:

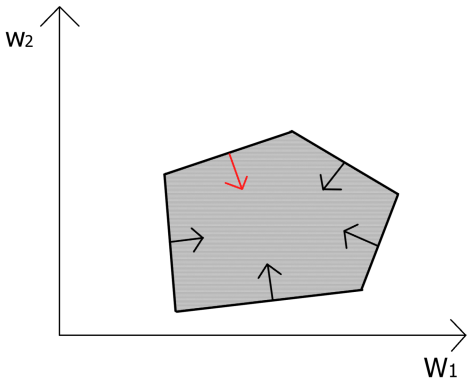
O Oráculo “Apagoge” responde que a amostra é relevante se qualquer solução viável for encontrada para o novo sistema. Isto inclui evidentemente as soluções da face e possivelmente outras soluções pelo fato de: $-Y_k.(< W, X_k > +b) \geq 0$. Entretanto, para o Oráculo PE procura-se encontrar um hiperplano ou solução que contenha a amostra, para o qual: $-Y_k.(< W, X_k > +b) = 0$, restringindo desta forma à região de viabilidade.

Contudo, caso exista uma solução que satisfaça a inequação: $-Y_k.(< W, X_k > +b) \geq 0$, também existirá uma solução que satisfaça a igualdade: $-Y_k.(< W, X_k > +b) = 0$. Isto significa que o oráculo proposto pode classificar corretamente todas as amostras não relevantes. Em contrapartida, para o Oráculo “Apagoge”, uma amostra considerada relevante, pode ser classificada como não relevante, caso o mesmo não execute em um tempo de computação suficiente. Neste sentido, pode-se concluir que o Oráculo PE é mais vantajoso pelo fato de ser finito e também de garantir a não eliminação de amostras suportes em um espaço de busca mais restrito. A seguir na Figura 13, é apresentada uma ilustração do espaço de versões mostrando como se procede a verificação da redundância de uma amostra para cada um dos três oráculos apresentados.

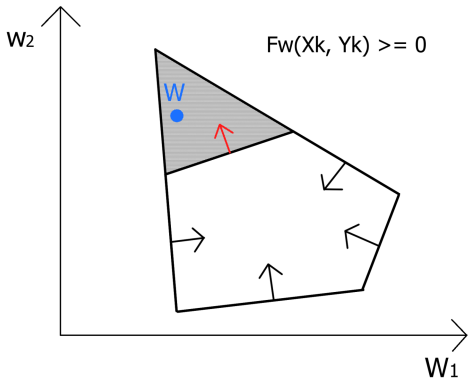
Além do problema do Oráculo "Apagoge"não garantir a redundância de uma instância, ele pode levar a rotular erroneamente uma instância redundante como relevante em um caso específico, no qual o conjunto de dados não é separável, porém se torna ao inverter a classe de uma instância. A Figura 14 exemplifica este problema.

Figura 13 - Processos de avaliação da relevância de uma amostra (X_k, Y_k) para cada um dos oráculos.

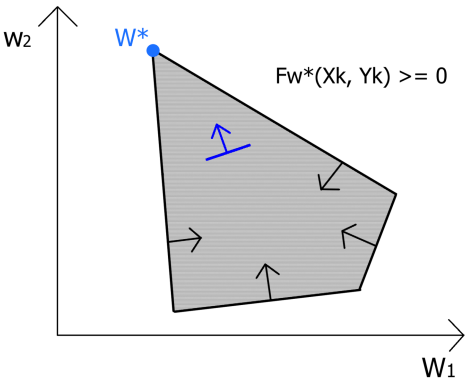
(a) Espaço de versões inicial.



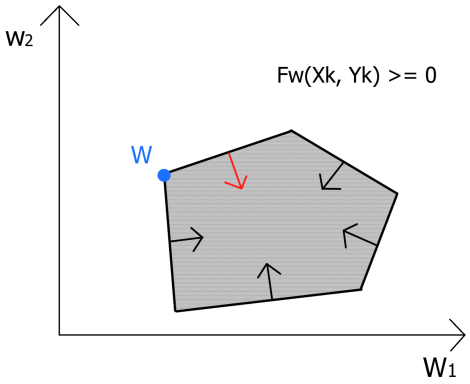
(b) Solução do Oráculo "Apagoge".



(c) Solução pela formulação de PL. Seta azul indica a direção do gradiente.



(d) Solução pelo Oráculo Ponto Extremo.



Fonte: Própria produção.

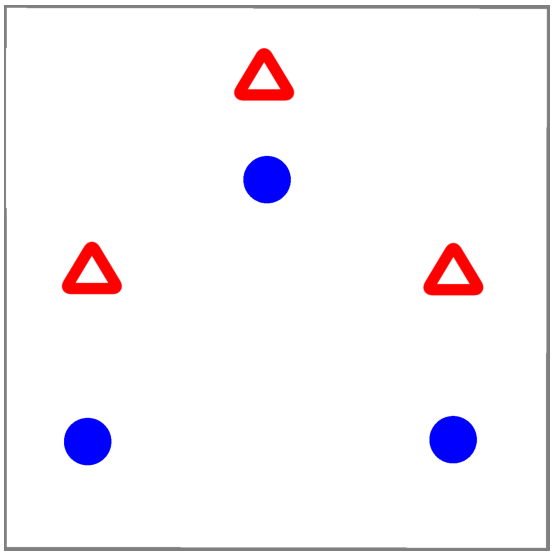
Este problema não ocorre ao usar a definição do Oráculo PE. Como não existe um hiperplano viável que contém a instância superior azul, ela não será rotulada corretamente como não relevante. Está é outra vantagem ao usar a definição proposta neste trabalho.

6.3 DEFINIÇÃO DE PONTOS EXTREMOS

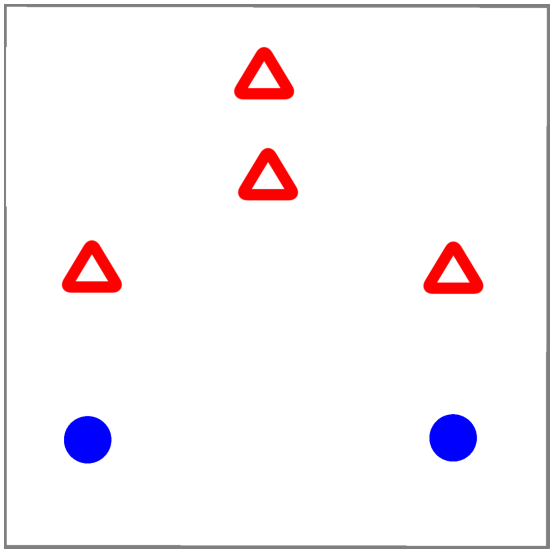
Durante este trabalho foi descoberto uma forte relação com um problema da área de Geometria Computacional, sendo ele o de detecção de pontos extremos [1]. Devido a isto, é importante conhecer este problema para melhor entendimento do trabalho produzido

Figura 14 - Exemplo onde a apagoge classifica erroneamente uma instância como relevante. A Figura 14b mostra a inversão de classe da apagoge sendo aplicada à instância azul que está mais acima, trocando a sua classe para vermelha. A Figura 14c mostra um hiperplano viável encontrado apos esta inversão, o que leva a apagoge a rotular incorretamente a instância como relevante.

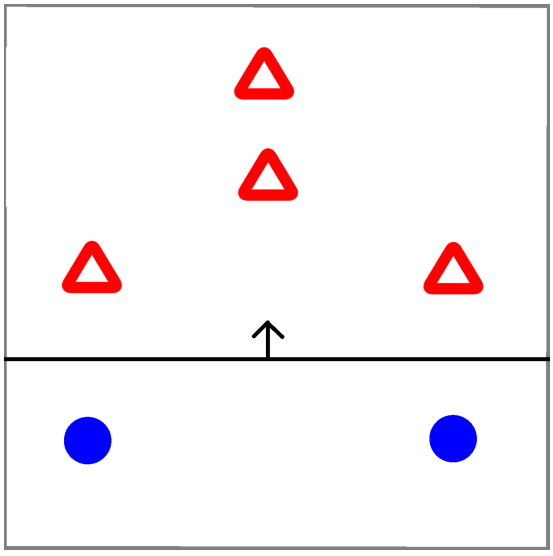
(a) Base de dados inicial.



(b) Inversão de classe aplicada à instância azul superior.



(c) Classificador factível que leva a apagoge a rotular erroneamente a instância invertida como relevante.



Fonte: Própria produção.

para a detecção de instâncias relevantes.

Devido a relação existente entre o problema: “detecção de amostras relevantes” ou, seu complemento: “remoção de amostras não relevantes”, e o problema de detecção de pontos extremos, torna-se importante conhecer este último para melhor entendimento do oráculo a ser proposto para a detecção de amostras relevantes. Também, é importante

ressaltar que o problema em questão se refere a um problema de decisão e não a determinação de uma envoltória convexa. Ou seja, saber se, para um dado conjunto convexo dado por uma nuvem de pontos, um determinado ponto é extremo ou não. A definição formal para um ponto extremo é dada a seguir:

Definição 6.3.1. *Dado um espaço vetorial real ou complexo K, para todo ponto $p, q, b \in K$, dizemos que p está entre q e b caso ele possa ser escrito como uma combinação afim de ambos.*

$$p = t \cdot q + (1 - t) \cdot b, \ t \in [0, 1], \tag{6.3}$$

Sendo K um sub-conjunto de $\mathcal{X}$, p é rotulado como ponto extremo de K caso ele, segundo a Equação 6.3, não esteja entre dois outros pontos distintos de K. O sub-conjunto de pontos de K que atendem a esta condição é dito como o conjunto de pontos extremos de K. O subconjunto de pontos de K que atendem a esta condição é denominado de conjunto de pontos extremos de K. A partir desta definição, o conjunto de pontos extremos pode ser interpretado geometricamente como o conjunto de vértices de um poliedro convexo formado por uma nuvem de pontos K.

Uma segunda definição para o conjunto de pontos extremos foi derivada a partir desta:

Definição 6.3.2. *Dado uma direção D, os pontos com a maior projeção nesta direção são pontos extremos de K.*

Considerando um espaço paramétrico que contenha a direção D como uma de suas dimensões, e que a Equação 6.3 é uma combinação afim, é fácil perceber que os pontos de maior projeção na direção D não podem ser escritos por uma combinação afim de dois outros pontos. Deste modo, os pontos que possuem a maior projeção em uma direção arbitrária D são pontos extremos. Por esta definição, cada ponto extremo possui ao menos uma direção em que ele é o de maior projeção. Essa propriedade possibilita um meio rápido de verificar se um dado ponto é extremo. Nesse sentido, esta segunda definição tem maior relação com o oráculo proposto. Entretanto, também é possível a construção de um oráculo finito baseado em programação linear.

Caso o vetor X_k de uma amostra (X_k, Y_k) seja um ponto extremo ele não pode ser representada por uma combinação linear convexa dos vetores de outras amostras. Matematicamente, esta condição pode ser formulada como um problema de otimização na forma:

$$Minf(X_k) = V_k \tag{6.4}$$

Sujeito a:

$$\begin{aligned} X_k &= \sum_{j=1}^{m} V_j \cdot X_j \\ \sum_{j=1}^{m} V_j &= 1 \\ V_j \geq 0,\ j &= 1, ..., k, ..., m \end{aligned} \tag{6.5}$$

Se $f^*(X_k) = 1$ então o vetor X_k é um ponto extremo. Caso contrário, se $f^*(X_k) < 1$, então X_k não é um ponto extremo. Este oráculo, baseado em Programação Linear (PL), também padece do mesmo problema do Oráculo PL anteriormente apresentado, relacionado ao grande esforço computacional. Pois, embora o sistema de restrições possua somente duas equações, a matriz de coeficientes do conjunto de restrições está associada a dimensão do problema e também a quantidade de amostras.

6.4 REDUÇÃO DO PROBLEMA DE DETECÇÃO DE PONTOS RELEVANTES AO DE DETECÇÃO DE PONTOS EXTREMOS

A forte relação entre os problemas de detecção de pontos extremos e de classificação binária será mostrada nesta seção por meio de uma redução de um problema ao outro. Ou seja, será provado que um algoritmo utilizado para solução de um problema também pode ser usado para solucionar o outro.

Primeiramente, deve-se analisar os dois problemas, suas similaridades e diferenças, para entender como essas diferenças podem ser contornadas para transformar uma instância de um problema no outro. Começando pelas similaridades, ambos possuem uma nuvem de pontos, sendo que no problema original de classificação binária tem-se uma nuvem composta por instâncias rotuladas como positivas ou negativas. Neste caso, o objetivo é encontrar um hiperplano que separe corretamente as duas classes. Já na definição do problema de detecção de pontos extremos temos uma nuvem de pontos, não rotulados, em que um ponto dessa nuvem está sendo analisado, onde, segundo a Definição 6.3.2, procura-se provar a existência de um hiperplano que não só contenha este ponto mas que agrupe todos os outros pontos no mesmo semi-espaço. Apesar de, neste problema não existirem rotulações, o fato de todos os pontos precisarem estar num mesmo semi-espaço pode ser usado para interpretar todos eles como se fossem instâncias que possuem um mesmo rótulo.

A equivalência destes dois problemas está baseada no seguinte pressuposto: Caso uma amostra seja não redundante, será possível, na nuvem de pontos, determinar um hiperplano que contenha a sua amostra correspondente e divida o espaço em dois semi-espaços. Um semi-espaço positivo que conterá toda nuvem de pontos e outro semi-espaço negativo que será vazio.

A definição de amostra correspondente depende da técnica de espelhamento a ser utilizada. Normalmente, para ser coerente com o resultado acima, mantém-se as amostras da classe positiva nas suas posições originais e rebate-se as amostras negativas, construindo desta forma uma nuvem de pontos sem rotulações. Este processo de rebatimento consiste em trocar o sinal do vetor e do rótulo associados a respectiva amostra, o que se torna parecido a um espelhamento.

Nesse sentido, foi criado um método para realizar este procedimento, sendo que seu funcionamento é dado a seguir. Inicia-se com uma instância X_p, pertencente a uma nuvem de instâncias $\mathcal{X}$, a qual se deseja verificar se ela é relevante ou não. Denomina-se esta instância X_p de pivô. Esta instância pivô será utilizada para gerar um sub-espaço do espaço de versões relacionada a ela, denotado por V_p. Este sub-espaço é formado ao adicionar-se uma nova restrição a definição do espaço de versões:

$$V_p = \{(W, b) \in \mathbb{R}^{d+1} : \ Y_i \cdot (\langle W, X_i \rangle + b) \geq 0, \ \forall (X_i, Y_i) \in Z_m \ \wedge \ \langle W, X_p \rangle + b = 0\} \quad (6.6)$$

Esta restrição adicional, dada pela equação de igualdade, limita os hiperplanos viáveis a somente aqueles que contém a instância pivô X_p. É importante notar que, segundo a Definição 6.2.4, caso o pivô não seja uma instância relevante, então o sub-espaço V_p é vazio. Ou seja, para mostrar que X_p é relevante, é necessário existir um hiperplano factível que a contenha, ou seja, pertencente a este sub-espaço de versões.

O próximo passo da redução está em gerar um conjunto de instâncias tais que todas elas possuam a mesma classe do pivô, e que, além disto, este novo conjunto não possua um sub-espaço de versões relacionado ao pivô diferente do sub-espaço original. Deste modo consegue-se remover a principal diferença entre os dois problemas, a diferença na quantidade de classes, sem afetar as soluções válidas.

Considerando o pivô escolhido e seu sub-espaço de versões, procura-se para cada instância $X_o \in \mathcal{X}$, de classe diferente a do pivô, gerar uma instância X_e que seja da mesma classe do pivô, porém que produza exatamente a mesma restrição no sub-espaço de versões que a instância original X_o. Deste modo esta instância de classe oposta pode ser removida do conjunto sem afetar a factibilidade do problema de classificação.

Este processo inicia-se definindo um vetor D a partir de uma instância X_o de classe diferente a classe do pivô:

$$D = X_o - X_p \quad (6.7)$$

assim, ao utilizar este vetor D pode-se reescrever a instância X_o em função de D e X_p:

$$X_o = X_p + D \quad (6.8)$$

Esta definição dada pela Equação 6.8 será utilizada para adicionar uma nova instância X_e ao conjunto $\mathcal{X}$, sendo sua posição dada por $X_e = X_p - D$ e sua classe sendo diferente da instância X_o, ou seja, a mesma classe da instância pivô. A adição desta instância não afeta o sub-espaço de versões associado ao pivô escolhido.

Teorema 6.4.1. *Dado um conjunto de instâncias $\mathbb{Z} = \{\mathcal{X}, \mathbb{Y}\}$; duas instâncias $X_p \in \mathcal{X}$ e $X_o \in \mathcal{X}$; e uma instância $X_e \notin \mathcal{X}$ de classe oposta a classe de X_o e com vetor de características definido por $X_p - D$, onde $D = X_o - X_p$. A adição da instância X_e ao conjunto $\mathcal{X}$, e remoção da instância X_o, não afeta o sub-espaço de versões V_p relacionado à instância X_p.*

Demonstração. Considerando a definição de sub-espaço de versões dado pela Equação 6.6, deve-se provar que as restrições impostas pelas instâncias X_o e X_e ao sub-espaço de versões relacionado à instância X_p são as mesmas. Para isto ocorrer, a seguinte igualdade deve ser verdadeira:

$$Y_o \cdot (\langle W, X_o \rangle + b) = Y_e \cdot (\langle W, X_e \rangle + b) \tag{6.9}$$

A prova de que a Equação 6.9 está correta começa com a restrição imposta pela instância X_o:

$$Y_o \cdot (\langle W, X_o \rangle + b) \tag{6.10}$$

Substituindo X_o por $X_p + D$, como definido pela Equação 6.8:

$$Y_o \cdot (\langle W, X_o \rangle + b) = Y_o \cdot (\langle W, X_p + D \rangle + b) \tag{6.11}$$

Colocando D em evidencia obtém-se:

$$Y_o \cdot (\langle W, X_p + D \rangle + b) = Y_o \cdot (\langle W, X_p \rangle + b + \langle W, D \rangle) \tag{6.12}$$

Utilizando o fato de que o ponto X_p está contido no hiperplano W:

$$Y_o \cdot (\langle W, X_p \rangle + b + \langle W, D \rangle) = Y_o \cdot (0 + \langle W, D \rangle) \tag{6.13}$$

Ao inverter o valor da classe Y_{co} e o sentido do vetor D obtém-se:

$$Y_o \cdot (0 + \langle W, D \rangle) = -Y_o \cdot (0 + \langle W, -D \rangle) \tag{6.14}$$

Utilizando o fato de que Y_{ci} tem valor inverso ao de Y_{co}:

$$-Y_o \cdot (0 + \langle W, -D \rangle) = Y_e \cdot (0 + \langle W, -D \rangle) \tag{6.15}$$

Utilizando novamente o fato de que o ponto X_p está contido no hiperplano (W, b):

$$Y_e \cdot (0 + \langle W, -D \rangle) = Y_e \cdot (\langle W, X_p \rangle + b + \langle W, -D \rangle) \tag{6.16}$$

Unindo os dois produtos vetoriais:

$$Y_e \cdot (\langle W, X_p \rangle + b + \langle W, -D \rangle) = Y_e \cdot (\langle W, X_p - D \rangle + b) \tag{6.17}$$

Utilizando a Definição 6.7 do vetor D:

$$Y_e \cdot (\langle W, X_p - D \rangle + b) = Y_e \cdot (\langle W, X_e \rangle + b) \tag{6.18}$$

Assim provamos que:

$$Y_e \cdot (\langle W, X_e \rangle + b) = Y_o \cdot (\langle W, X_o \rangle + b) \tag{6.19}$$

□

Isto demonstra que o valor das restrições aplicadas por estas instâncias ao sub-espaço de versões V_p são iguais, ou seja, caso a restrição imposta pela instância X_o seja atendida, então a restrição da instância X_e também estará, necessariamente, sendo atendida.

Devido a isto, ter ambas as instâncias X_o e X_e no conjunto $\mathcal{X}$ implica em redundância. Deste modo, uma delas pode ser eliminada sem que isto afete o sub-espaço de versões V_p. Pode-se então eliminar a instância original X_o, cuja classe é diferente da classe do pivô, e manter a instância X_e cuja classe é a mesma. Ao aplicar este mesmo procedimento para todas as outras instâncias cujas classes são diferentes da classe do pivô, termina-se com um conjunto de instâncias as quais todas pertencem a mesma classe.

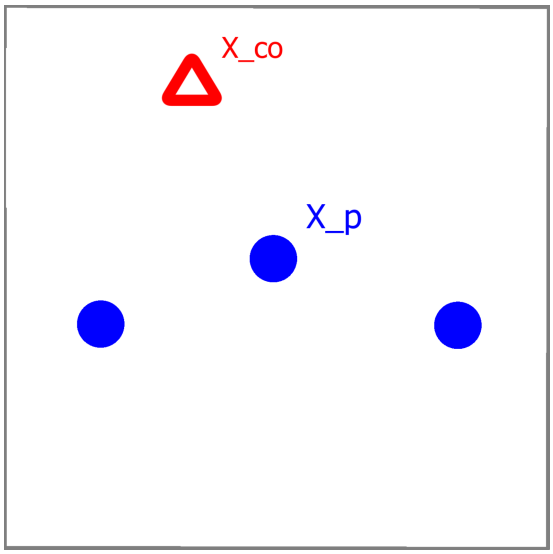
Considerando a restrição adicional usada para gerar o sub-espaço de versões, em que um hiperplano solução deve conter a instância pivô, e este novo conjunto de instâncias em que todas pertencem a mesma classe, tem-se que para existir um hiperplano factível, ele deve possuir todas as instâncias do conjunto no mesmo semi-espaço, além de conter o pivô. Estes requisitos impostos ao hiperplano para ele ser factível são os mesmos que, segundo a Definição 6.3.2, definem a posição da instância pivô como sendo um ponto extremo da nuvem formada por este conjunto de instâncias. Caso exista um hiperplano que atenda a estas restrições, o mesmo pode ser utilizado como uma solução do problema de classificação inicial. Caso não exista um hiperplano atendendo a estas restrições, então o instância utilizada como pivô pode ser marcada como sendo não relevante no problema de classificação, devido a não existir um hiperplano factível que a contém. É importante notar que caso seja demonstrado que nenhuma instância, ao ser utilizada como pivô, é

capaz de gerar um hiperplano que atenda a estas condições, isto significa que o espaço de versões é nulo, ou seja, o problema inicial de classificação não possui soluções. Deste modo, resolver o problema classificação pode ser reduzido a resolver o problema de definir se um ponto é extremo.

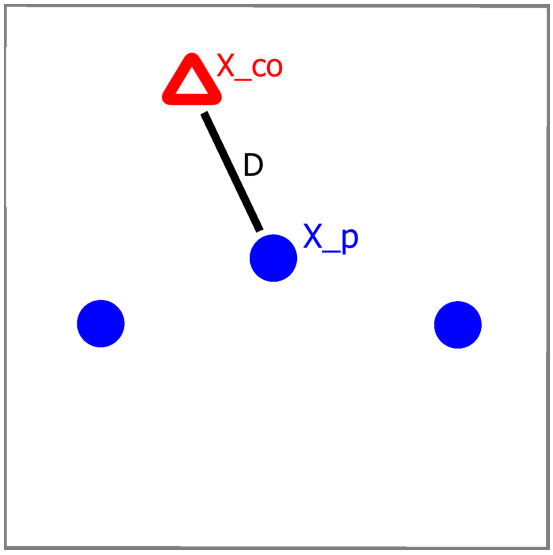
Um exemplo deste método para inversão de classes é dado na Figura 15:

Figura 15 - Exemplo do método de inversão de classes aplicado a uma instância.

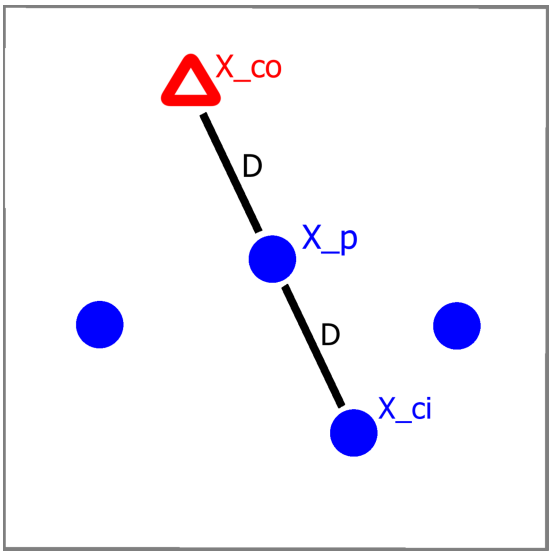
(a) Base de dados inicial. Procura-se gerar um conjunto de instâncias apenas azuis cujo sub-espaço de versões associado à instância P_v seja o mesmo que o deste conjunto inicial.



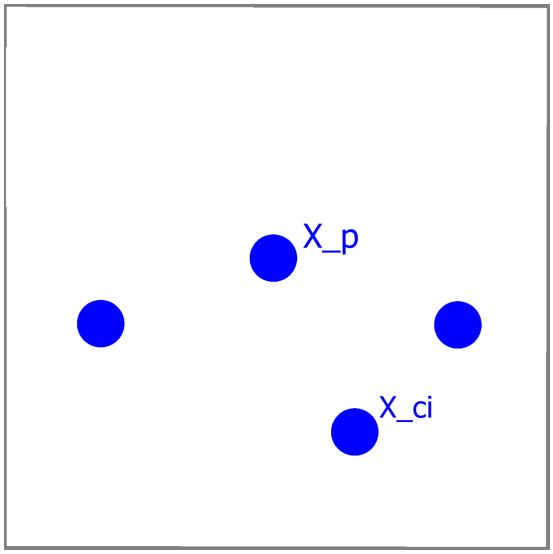
(b) Realiza-se o processo de inversão de classes para o ponto vermelho P_{co}. Começa-se calculando o Vetor D utilizando P_v e P_{co}.



(c) Gera-se o ponto P_{ci} sendo ele da mesma classe do pivô, e tendo como posição a do pivô menos o vetor D.



(d) Com o ponto P_{ci} adicionado ao conjunto, pode-se retirar o ponto P_{co} e garantir que o sub-espaço de versões associado ao ponto escolhido como pivô continua o mesmo. Caso existissem mais pontos vermelhos, este processo seria aplicado a todos eles até só existirem pontos azuis.



Fonte: Própria produção.

7 DETECÇÃO DE PONTOS EXTREMOS

A redução de problemas apresentada no capítulo anterior necessita um algoritmo capaz de definir se um dado ponto é extremo numa nuvem de pontos. Neste capítulo inicialmente será realizada uma revisão dos algoritmos existentes para este problema, e que serviram de inspiração para o desenvolvimento do algoritmo proposto, analisando suas limitações. Em seguida serão descritas as versões primal e dual do algoritmo desenvolvido, tendo este como pontos positivos escalar bem seu tempo de execução para dimensões altas e não precisar de parâmetros para sua execução. Adicionalmente, o método proposto produz uma solução viável para o problema de classificação associado, caso o ponto em análise seja extremo.

O funcionamento deste algoritmo necessita de um método para solucionar outro problema, o de dado um conjunto de no máximo **d** pontos, encontrar um hiperplano que tangencie todos eles. Sendo assim, foi também necessário desenvolver um algoritmo para este segundo problema. Este segundo algoritmo será explicado neste capítulo, após o detalhamento do algoritmo de detecção de pontos extremos, junto de sua relação com a solução *SVM* e potencial para desenvolvimento de outros métodos de classificação.

7.1 ALGORITMOS RELACIONADOS

Inicialmente foram estudadas abordagens relacionadas para o problema de detecção de pontos extremos, estas provenientes da área de computação gráfica. Estas abordagens envolvem a detecção da envoltória convexa de um poliedro. A definição de envoltória convexa é apresentada a seguir.

Um conjunto de pontos P é definido como convexo se para quaisquer pontos $P_x, P_y \in P$ existir um segmento de reta pertencente a P que una os dois pontos. A envoltória convexa de um conjunto de pontos P é definida como o menor conjunto convexo que contenha todos os pontos de P [36]. Determinar a envoltória convexa de um conjunto de pontos é um dos problemas principais da área de Geometria Computacional, tendo aplicações nas áreas de economia [33], estatística [20, 72, 29] e otimização [39, 52]. O problema de detectar o conjunto de pontos extremos está relacionado ao problema de detectar a envoltória convexa, como demonstrado pelo Teorema de Krein–Millman [42]:

Teorema 7.1.1. *Se* $\boldsymbol{P}$ *é um sub-conjunto compacto de um espaço vetorial localmente convexo Hausdorff, então o conjunto de pontos extremos de* $\boldsymbol{P}$ *tem a mesma envoltória convexa que* $\boldsymbol{P}$*.*

A partir deste resultado, é possível utilizar algoritmos de detecção de envoltória convexa para encontrar pontos extremos. Um ponto importante a ser notado é que como a

envoltória convexa consiste de todos os vértices, arestas, faces, d-hiperfaces de um poliedro, então encontrar o conjunto de pontos extremos é um sub-problema da envoltória convexa.

Foram estudados quais algoritmos são utilizados para a solução do problema da envoltória convexa, porém os métodos clássicos foram desenvolvidos especificamente para problemas de duas e três dimensões, os quais são os domínios mais comuns nas aplicações relacionadas a este problema. Exemplos destes algoritmos são Grahan [27], Gift-Wraping [34] e Quickhull [11]. Apesar de para até três dimensões existirem algoritmos eficientes, para dimensões superiores este problema possui custo assintótico, para n pontos em uma dimensão d, de ordem $O(n^d)$, o que os torna inviável para serem aplicados em bases de dados de Aprendizado de Máquinas, as quais frequentemente possuem uma dimensionalidade maior. Devido a este alto custo, pesquisas recentes neste problema levaram a criação de algoritmos que aproximam a envoltória convexa como por exemplo o Deephull [4]. Porém, como procura-se somente saber se um ponto especifico é extremo ou não, este tipo de algoritmo não seria eficiente. Assim, considerou-se inicialmente a possibilidade de adaptar algum dos métodos exatos para apenas solucionar este problema de decisão. Esta ideia de criar uma adaptação foi explorada por um tempo, porém não foi possível gerar uma adaptação que fosse eficiente. Apesar disto, analisar o funcionamento dos algoritmos clássicos e dos aproximativos ajudou a entender melhor o problema e a criar um método para a detecção de um ponto extremo.

O algoritmo que serviu de inspiração inicial para o desenvolvimento foi o *Quickhull* [11]. Este algoritmo encontra as faces, e por consequência os vértices, de um polígono convexo por meio de uma operação simples de expansão, em que dado um hiperplano, procura-se qual o vértice mais distante e o utiliza para gerar d novos hiperplanos. O pseudo-código simplificado para o caso de duas dimensões é dado a seguir:

Algorithm 4: Pseudo-código do Algoritmo Quickhull para $\mathbb{R}^2$.

Input = Um conjunto de n pontos.;
Casca Convexa: $\mathcal{C} \leftarrow$;
Adicionar os pontos $A = \text{Inf}_{X_i \in \mathcal{X}} X_i[0]$ e $B = \text{Sup}_{X_i \in \mathcal{X}} X[0]$ em $\mathcal{C}$;
Adicionar os pontos de um lado do segmento $\overline{AB}$ a um conjunto $\mathcal{P}_1$ e os pontos do outro lado a um conjunto $\mathcal{P}_2$. FindHull($\mathcal{C}$, $\mathcal{C}_1$, A, B);
FindHull($\mathcal{C}$, $\mathcal{C}_2$, B, A);
return $\mathcal{C}$;

Este algoritmo é usado devido a sua boa eficiência em problemas de duas e três dimensões. Porém, na área de Aprendizado de Máquinas, é comum encontrar bases de dados com uma grande quantidade de dimensões, neste caso o Quickhull não escala bem seu custo computacional de tempo devido ao mesmo ser de ordem $O(n^d)$, tornando inviável sua utilização. Isto ocorre por causa de dois motivos: primeiro, pela quantidade de subconjuntos que ele gera a cada iteração. Em um problema bidimensional, como o

Algorithm 5: FindHull() : Sub-rotina do Algoritmo Quickhull para $\mathbb{R}^2$.

Input: Um conjunto de pontos $\mathcal{C}_i$; um ponto P; um ponto Q;
if $S = \emptyset$ **then**
  return;
else
  Ponto $M = \max_{X_i \in \mathcal{X}} \text{Dist}(\overline{PQ}, X_i)$;
  Adiciona M a $\mathcal{C}$;
  Conjunto de pontos S1: Pontos que estejam a direita do segmento orientado PM;
  Conjunto de pontos S2: Pontos que estejam a esquerda do segmento orientado MQ;
  FindHull(S1, P, M) ;
  FindHull(S2, M, Q) ;
end

exemplificado no pseudo-código acima, ele duplica a quantidade de subconjuntos a serem analisados a cada iteração, para um problema de três dimensões ele triplicaria, e em um problema d-dimensional ele multiplicaria em d vezes [11]. Deste modo, apesar de ser eficiente para problemas de baixa dimensão ele não é eficiente para problemas com uma quantidade maior de dimensões.

O segundo motivo de sua ineficiência está relacionado a uma especificidade do que se procura. O Quickhull finaliza sua execução somente após encontrar todas as faces e vértices do poliedro. Porém, no problema que precisa-se resolver necessita-se somente determinar se um ponto específico pertence a envoltória convexa, ou seja, se é ponto extremo ou não. No caso bidimensional significaria que, a cada iteração, apenas uma das duas faces geradas necessitaria ser expandida para encontrar a solução. Infelizmente, este mesmo raciocínio não é aplicável para dimensões maiores que duas, o que levaria a um custo computacional exponencial.

Apesar das limitações encontradas ao estudar o funcionamento do Quickhull, elas possibilitaram perceber que iterar uma quantidade menor de faces tornaria a resolução do problema mais rápida. Isto se tornou a motivação ao estudo e desenvolvimento de um método que aproveitasse esta ideia. A próxima seção introduzirá o algoritmo desenvolvido.

7.2 ALGORITMO DETECTOR DE PONTO EXTREMO (ADePE)

Devido a alta dimensionalidade presente nas bases de Aprendizado de Máquinas, foi desenvolvido um algoritmo que fosse capaz de escalar seu custo computacional. Como será mostrado na sub-seção de experimentos 7.2.5, o algoritmo consegue definir se um ponto é extremo ou não em tempo factível, escalando bem tanto em relação a quantidade de pontos quanto em relação a quantidade de dimensões do problema. Esta vantagem em relação aos métodos de envoltória convexa foi possível devido ao escopo do problema abordado ser um

sub-problema da envoltória convexa, no qual ao invés de todas as hiperfaces precisarem ser definidas, busca-se apenas determinar os pontos extremos. As próximas sub-seções formalizam o problema a ser resolvido, a interpretação geométrica do algoritmo, seu pseudo-código e um diagrama para facilitar a visualização de todas as suas etapas.

7.2.1 Definição do Problema

Antes de explicar o funcionamento do algoritmo é necessário definir qual o problema geométrico que ele procura resolver. A partir da definição do problema e de sua análise, se torna mais fácil compreender o funcionamento do algoritmo em questão..

O problema pode ser definido da seguinte forma: A partir de um conjunto de pontos $\{P\} \subset \mathbb{R}^d$, determinar se um ponto $P_p \in \{P\}$ é um ponto extremo de $\{P\}$. Para P_p ser considerado um ponto extremo de $\{P\}$ ele deverá atender a Definição 6.3.1 e, consequentemente, à Definição 6.3.2 apresentados no capítulo anterior.

Dentre estas duas formas de verificar se um ponto é extremo ou não, a Definição 6.3.2 foi utilizada para determinar o funcionamento e a condição de parada do algoritmo. Para provar que o ponto P_p é extremo em $\{P\}$, o algoritmo procura um hiperplano W que atenda as seguintes condições de término:

Condição 1: P_p esteja contido nele.

Condição 2: Todos os outros elementos de $\{P\}$ estejam ou contidos em W ou estejam todos num mesmo semi-espaço definido por W.

Caso exista um hiperplano que satisfaça estas duas condições, então a direção apontada pela normal dele estará provando que P_p é extremo segundo a Definição 6.3.2. Caso seja provado que não existe um hiperplano que atenda a estas condições, então podemos afirmar que P_p não é extremo. O algoritmo tem como objetivo encontrar, ou provar a impossibilidade da existência de, um hiperplano que atenda estas condições. A próxima subseção analisará a detecção destas condições de parada e como ela pode ser detectada.

7.2.2 Critério de Parada e Hiperplano Inicial

O critério de parada este algoritmo deve satisfazer o seguinte condição: encontrar um hiperplano que atenda as duas condições definidas na seção anterior, ou provar que o mesmo não existe.

Considerando que, pela segunda condição, para um ponto ser extremo é necessário existir um semi-espaço vazio, se o vetor normal do hiperplano inicial apontar para este semi-espaço, podemos utilizar a métrica de margem como uma função de otimização. O quanto maior for a distância do ponto ao hiperplano, contando que satisfaça a primeira condição, em relação aos outros pontos do conjunto, mais próximo ele será de uma solução

viável. Caso seja aplicado um método de otimização a este hiperplano, ele eventualmente ou conseguirá separar os pontos em um mesmo semi-espaço, provando que o pivô em questão é um ponto extremo, ou chegará na margem ótima sem conseguir separar os pontos em semi-espaços diferentes. Deste modo, o requisito para a solução inicial é apontar para o semi-espaço vazio, caso ele exista.

Portanto, tratando-se de semi-espaços, é possível gerar um vetor aleatório, e testá-lo para ambas as direções. Se o semi-espaço vazio existir e não for detectado numa direção, então ele será detectado na outra. A partir desta analise é possível escolher uma direção inicial que exime a necessidade de testar seus dois sentidos. Isto pode ser feito definindo como direção inicial a posição do pivô menos a posição de um ponto qualquer escolhido.

7.2.3 Funcionamento do Algoritmo

Ao analisar o Quickhull foi notado que ele procura primeiramente gerar os conjuntos de faces da envoltória convexa da nuvem de pontos. A partir de uma face, é trivial encontrar as hiperfaces e vértices que a formam. Considerando este fato, foi definido como objetivo do algoritmo provar a existência de uma face da envoltória convexa da nuvem que contenha o pivô. Como toda face de um poliedro convexo é definida por um conjunto de d vértices, um modo de provar a existência de uma face da envoltória contendo o pivô seria testar as faces geradas a partir de todas as combinações possíveis de vértices que contenham o pivô. Porém, a quantidade de faces que podem ser geradas deste modo é de grandeza combinatorial, o que tornaria este modo *força bruta* inviável na prática. Porém, como discutido na sub-seção anterior, pode-se tratar o problema de maximização de margem como um problema convexo ao gerar a hipótese inicial pelo método especificado. A partir desta hipótese inicial, pode-se utilizar um método guloso para iterar a outra hipótese de maior margem, e assim aproveitar esta propriedade para podar a quantidade total de faces a serem testadas, gerando um método que seja viável na prática. O algoritmo desenvolvido aproveita disto para conseguir eficientemente verificar a existência de uma face que contenha o ponto pivô.

O algoritmo procura verificar a existência de um hiperplano que atenda as seguintes condições: conter o pivô, e manter todos os pontos no mesmo semi-espaço. Caso um hiperplano que atenda estas condições seja encontrado, então isto significa que o pivô é um ponto extremo. O algoritmo funciona por meio de três etapas que serão descritas brevemente a seguir, e em detalhes nos próximos parágrafos. A primeira etapa consiste em gerar um hiperplano inicial usando o método descrito na seção anterior, caso este não atenda as duas condições, o algoritmo prossegue para a segunda etapa. A segunda etapa procura qual o ponto mais distante positivamente, ou acima, do hiperplano, e o rotaciona para que este ponto fique contido nele. Este processo é realizado até um máximo de $d - 1$ vezes, sempre garantindo que o hiperplano após ser rotacionado ainda contenha

os pontos adicionados nas iterações anteriores. Após o hiperplano estar limitado a conter os d pontos adicionados, não é mais possível incrementar a quantidade de pontos que ele contém devido a não existir mais graus de liberdade para rotações. A terceira etapa busca retirar a restrição do hiperplano conter um destes pontos, e no lugar dela, adicionar a restrição do ponto mais acima dele. Esta troca é realizada para cada um dos pontos restringindo o hiperplano. Caso a margem do hiperplano aumente após uma troca, é atualizado qual o ponto mais acima do hiperplano e então esta etapa itera novamente até a margem se tornar positiva. Quando não houver aumento de margem em nenhuma troca, isto significa que o algoritmo chegou num ótimo local da margem, o que, pela propriedade de convexidade do problema, pode-se concluir que não existe um hiperplano que atenda as duas condições.

Para facilitar a execução do algoritmo, é realizado um pré processamento nos dados, sendo os pontos transladados para que o pivô fique na origem. Após isto, todos os pontos, com exceção do pivô que está na origem, são normalizados. Como estas duas mudanças não afetam a relação entre os ângulos dos pontos em relação a origem, isto também não afetará o conjunto de hiperplanos que atendem as duas condições. Estas mudanças facilitam compreender o funcionamento do algoritmo bem como sua implementação.

A primeira etapa do algoritmo procura gerar um hiperplano inicial que esteja no ápice do domínio da função objetivo margem, para que assim o problema possa ser tratado como um problema convexo. Como discutido anteriormente na Sub-seção 7.2.2, isto é realizado ao fazer com que a normal do hiperplano inicial W_i tenha a mesma direção, porém sentido oposto do vetor correspondente a posição de um ponto P_i diferente do pivô, ou seja:

$$W_i = -P_i \tag{7.1}$$

A partir deste passo inicial, caso o hiperplano resultante seja capaz de atender as duas condições de término apresentadas anteriormente, o algoritmo para e retorna que o pivô é um ponto relevante para o espaço de versões. Caso o hiperplano não atenda as condições, então inicia-se a próxima etapa, aproveitando esta hipótese inicial que permite o problema ser tratado como se fosse convexo.

A segunda etapa procura provar que existe uma hiperface formada pelo pivô. Como toda hiperface de um poliedro é formada por um sub-conjunto de pontos extremos do mesmo, caso seja provado que o pivô faz parte de uma hiperface, isto significa que ele é um ponto extremo. Este processo pode ser interpretado geometricamente como o algoritmo procurando um hiperplano que inicialmente contenha uma aresta do poliedro, depois uma hiperface-3, depois uma hiperface-4 e assim sucessivamente até chegar a um hiperplano que possa conter uma hiperface-d, ou seja, uma face do poliedro. Devido a isto, esta segunda etapa consiste de um processo que itera até um máximo de $d - 1$ vezes.

Cada iteração se inicia ao encontrar um ponto extremo da nuvem. Isto é feito utilizando a definição 6.3.2 de pontos extremos, ao encontrar o ponto com maior projeção na normal do hiperplano. Em seguida o hiperplano é rotacionado de modo que este ponto esteja contido nele, e de modo que os pontos encontrados nas iterações anteriores ainda estejam contidos no hiperplano. O viés do hiperplano sempre é mantido como zero para que o pivô também esteja contido no hiperplano. Assim, caso o hiperplano resultante atenda as condições de término, então significa que ele contém uma hiperface do poliedro que é formada pelo pivô, logo o pivô é um ponto extremo deste poliedro. Como uma hiperface máxima de um poliedro é uma face, e uma face é formada por d pontos, e considerando que o pivô sempre estará contido no hiperplano, esta etapa pode iterar por um máximo de $d-1$ vezes. Caso após esta quantidade de iterações ainda não seja encontrado um hiperplano que atenda as condições, então o algoritmo segue para a terceira etapa.

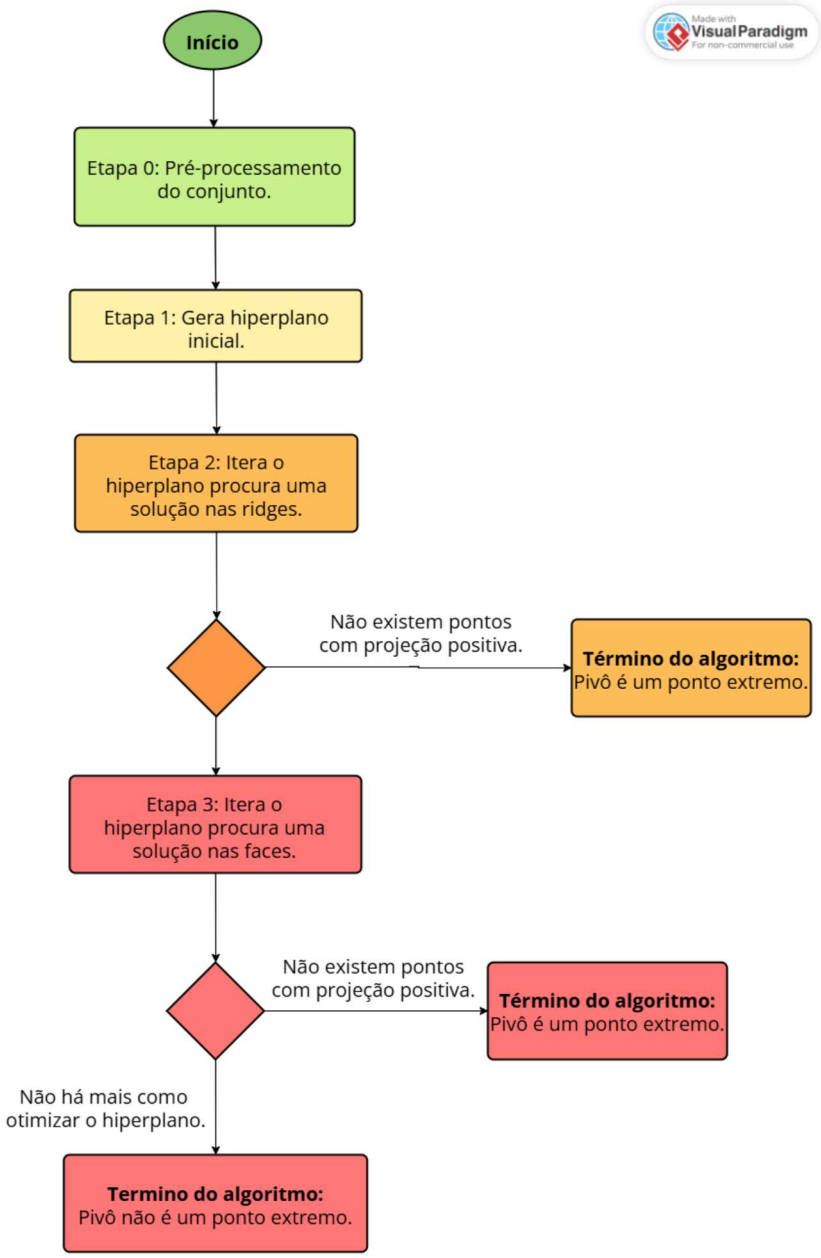
A terceira etapa continua iterando de modo similar à segunda etapa, porém com o objetivo de encontrar uma face do poliedro que seja formada pelo pivô, ou provar a inexistência de uma face assim. A cada iteração o algoritmo busca qual o ponto de maior projeção na normal do hiperplano. Em seguida, ele troca a restrição do hiperplano de conter um dos pontos encontrados nas primeiras $d-1$ iterações pela restrição do hiperplano conter este novo ponto. Após gerar o hiperplano que atende a este novo conjunto de restrições, a margem dele é comparada com a margem do hiperplano anterior. Caso ela tenha aumentado, o algoritmo termina esta iteração e passa para a próxima. Caso a margem tenha diminuído, o hiperplano é rotacionado de modo oposto para voltar a atender ao conjunto antigo de restrições, e então a troca de restrições é realizada para outra restrição. Caso todas as restrições tenham sido trocadas e nenhum hiperplano gerado tenha margem maior, esta iteração termina. No termino de uma iteração, caso a margem seja igual ou maior que zero, isto significa que uma face do poliedro que é formada pelo pivô foi encontrada, e assim o algoritmo retorna que o pivô é um ponto extremo. Caso a margem tenha aumentado porem ainda seja negativa, uma nova iteração da terceira etapa é realizada. Caso a margem não tenha aumentado, então isto significa que o algoritmo chegou no hiperplano de maior margem possível que contém o pivô, e portanto não existe uma face do poliedro que é formada por ele. Assim o algoritmo retorna que o pivô não é um ponto extremo.

Um diagrama resumindo as etapas do algoritmo está contido na Figura 16. Diagramas detalhados para cada uma das etapas estão nas Figuras 17, 18 e 19.

7.2.4 Hiperplano que Contém d Pontos

Encontrar um hiperplano que contenha um conjunto de pontos é uma parte essencial para o funcionamento do algoritmo. Um ponto P_i estar contido em um hiperplano significa que a distância do hiperplano W a este ponto é igual a zero, ou seja, que a seguinte

Figura 16 - Fluxograma resumido do algoritmo de detecção de pontos extremos. Em verde está o pré processamento. Em amarelo a primeira etapa. Em laranja a segunda etapa. Em vermelho a terceira etapa.



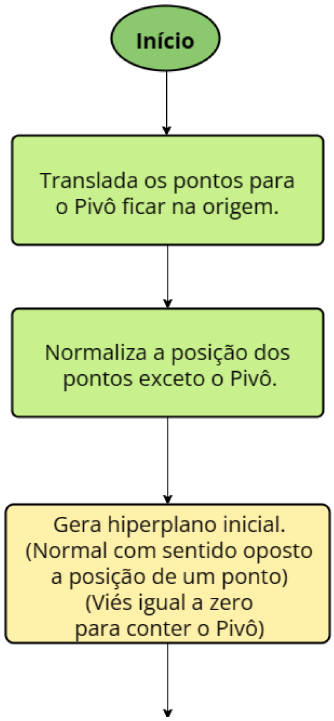
Fonte: Própria produção.

equação é satisfeita:

$$\langle W, P_i \rangle + b = 0 \tag{7.2}$$

Este problema possui similaridades em relação ao problema de encontrar uma base ortogonal a partir do mesmo conjunto de pontos. Considere que procura-se encontrar um hiperplano W que contenha todos os d pontos, $P_i \in P$, existentes no domínio $\mathbb{R}^d$. A partir deste conjunto de pontos é então criado um novo conjunto V, composto por d vetores,

Figura 17 - Fluxograma do pré-processamento, em verde, e da primeira etapa, em amarelo, do algoritmo de detecção de pontos extremos.



Fonte: Própria produção.

cujos primeiros $d-1$ elementos são formados pela seguinte equação:

$$V_i = P_i - P_0, \quad 0 < i \leq d \tag{7.3}$$

e o d-ésimo membro é formado pela equação:

$$V_d = W_0 - P_0 \tag{7.4}$$

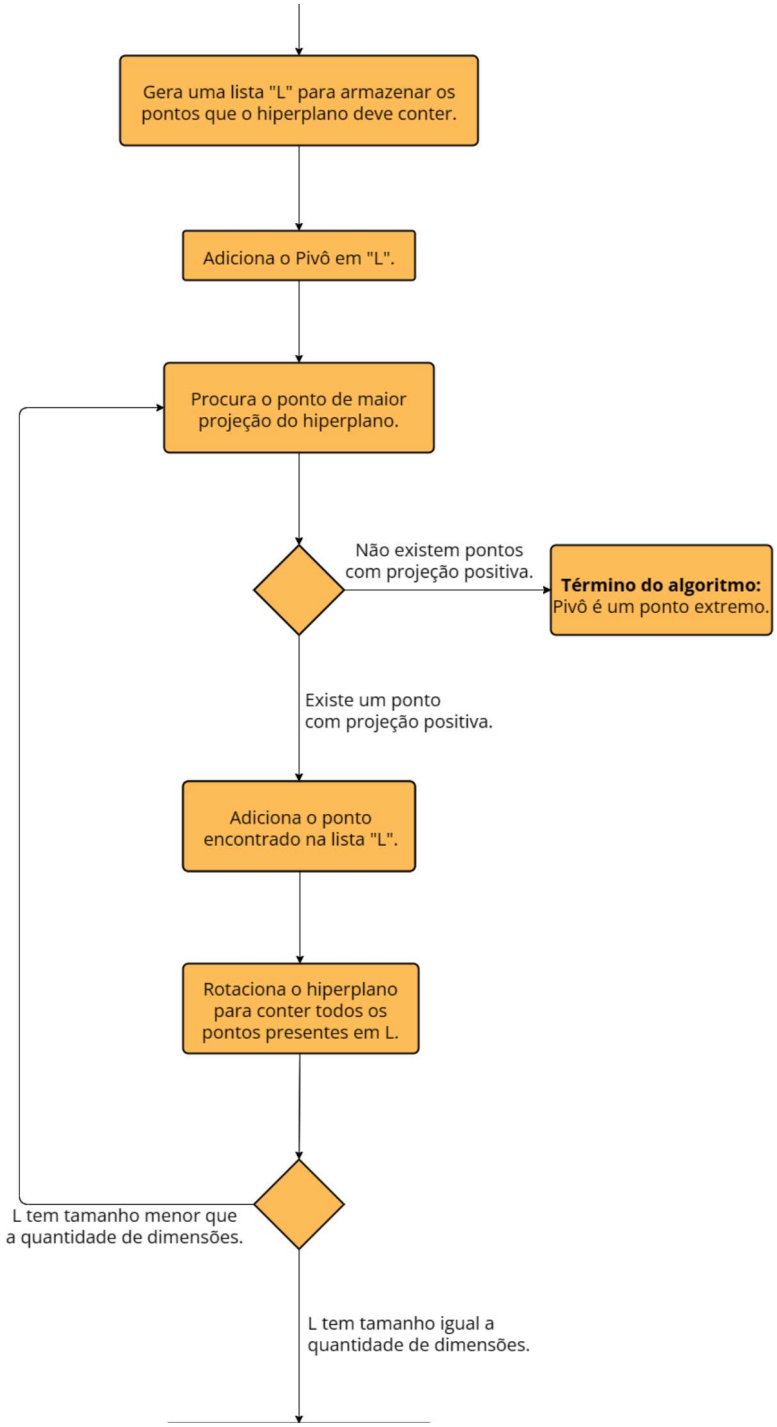
em que W_0 é a normal de um hiperplano qualquer.

Considere um caso especifico, em que todos os vetores de V são ortogonais entre si, ou seja, o conjunto V é uma base ortogonal de $\mathbb{R}^d$, isto implica em $\langle V_i, V_j \rangle = 0$ para quaisquer V_i e V_j pertencentes a V. A partir desta relação, é trivial encontrar um hiperplano W que contenha todos os pontos de P no caso do conjunto V ser uma base ortogonal. Para isto, a normal de W ser igual ao vetor V_d, pois assim tem-se a relação indicada pela Equação 7.5:

$$\langle W, V_i \rangle = \langle W, V_j \rangle \tag{7.5}$$

que é equivalente a:

Figura 18 - Fluxograma da segunda etapa do algoritmo de detecção de pontos extremos.



Fonte: Própria produção.

$$\langle W, P_0 - P_i \rangle = \langle W, P_0 - P_j \rangle \tag{7.6}$$

esta relação pode ser simplificada para:

$$W \cdot P_0 - W \cdot P_i = W \cdot P_0 - W \cdot P_j \rangle \tag{7.7}$$

o que leva a concluir que a distância entre um hiperplano cuja normal é igual a V_d a qualquer ponto pertencente a P é a mesma:

$$W \cdot P_i = W \cdot P_j\rangle \tag{7.8}$$

A partir desta relação, o viés do hiperplano é trivialmente definido segundo a seguinte equação:

$$b = -\langle W, P_0\rangle \tag{7.9}$$

Deste modo é obtido um hiperplano $\{W, -\langle W, P_0\rangle\}$ cuja distância a qualquer ponto de P é igual a zero. Porém, isto só foi possível devido aos vetores de V serem ortogonais entre si. Para tornar este procedimento aplicável a qualquer conjunto V, é necessário aplicar um algoritmo de ortogonalização no mesmo. O diagrama da Figura 20 mostra todas as etapas do processo de gerar um hiperplano que contenha todos os pontos de um conjunto P.

7.2.5 Experimentos

O método desenvolvido tem como objetivo determinar quais são as instâncias relevantes em um conjunto de dados. Para validar o seu funcionamento foram realizados dois experimentos. O primeiro com intuito de verificar se ele determina corretamente quais as instâncias relevantes. O segundo para analisar o tempo despendido para verificar a relevância de um ponto. Foram utilizadas oito bases de dados linearmente separáveis reais: Iris Binária [91], testNS, lp4 [92], Mushroom, Toy, Zoo [93], Colon e Divorce [94]. Foi realizado um pré-processamento nas bases de dados utilizadas para remover amostras duplicadas.

7.2.5.1 Análise da Corretude

Para validar a corretude do método é necessário analisar separadamente os dois casos. Se uma instância for relevante, o método retorna um hiperplano factível que a contém, assim a existência deste hiperplano é suficiente para verificar que a instância avaliada é relevante. Porém, para o caso da instância não ser relevante, o método apenas o rotula como não relevante. Deste modo, para validar a não relevância de um ponto, é necessário um oráculo externo. Para esta comparação foi escolhido o oráculo Apagoge, o qual usa o algoritmo perceptron. É importante ressaltar que o oráculo Apagoge fornece um limite inferior para o numero de instâncias relevantes.

A comparação entre os dois métodos foi realizada analisando a quantidade de instâncias relevantes que cada um retornou. Foi analisado também se o método desenvolvido foi capaz de detectar todas as instâncias encontradas pelo oráculo Apagoge. Como o

oráculo Apagoge não para de executar no caso de uma instância não ser relevante, ele foi executado por no máximo 1.000.000 épocas. Outro fator importante no Apagoge é a taxa de aprendizado que pode afetar o resultado final, para isto foram realizados testes com os seguintes valores: 1.0, 0.1, 0.05, 0.01, e 0.005, e escolhido a execução com os melhor resultado para cada base. A Tabela 8 contém os resultados da quantidade de instâncias relevantes de ambos os métodos.

Tabela 8 – Instâncias relevantes detectados, pelo ADePE e pelo oráculo Apagoge, em bases de dados linearmente separáveis.

Bases	**Iris Binária**	**testNS**	**lp4**	**Mushroom**	**Toy**	**Zoo**	**Colon**	**Divorce**
Dimensões	4	2	90	98	11	16	2000	54
Instâncias Totais	147	8	116	5644	12	59	60	150
Relevantes ADePE	37	5	116	5644	12	59	60	150
Relevantes Apagoge	36	5	116	5644	12	58	60	145

Estes experimentos revelam que o método desenvolvido foi capaz de encontrar uma quantidade igual ou maior de maior de instâncias relevantes que a apagoge, assim comprovando assim a sua eficácia.

7.2.5.2 Análise do Tempo de Processamento

Para avaliar a viabilidade do método, foi analisado o tempo necessário para executa-lo. Para realizar uma comparação precisa com o Apagoge, foi avaliado um valor aproximado do mínimo de épocas necessárias para obter uma quantidade próxima de instâncias relevantes das referidas Tabela 8. A Tabela 9 mostra o tempo despendido para analisar todas as instâncias de uma base pelo ADePE e pelo Apagoge, junto da quantidade de épocas máximas usadas como parâmetro no Apagoge. Pode-se perceber que o ADePE requer uma quantidade menor de tempo que a apagoge.

Considerando que a Apagoge despende uma quantidade maior de tempo nas instâncias não relevantes, é importante também realizar uma comparação do tempo despendido apenas nas instâncias relevantes. A Tabela 10 mostra o tempo médio, e desvio padrão, despendido para definir uma instância como relevante por cada um dos dois

Tabela 9 – Tempo total de execução despendido pelo ADePE e pelo Apagoge, em segundos.

Bases	**ADePE**	**Apagoge**	**Apagoge Epocas**
Iris Binária	0.01	460.76	500000
testNS	0.01	0.07	100
lp4	0.13	2.11	10000
Mushroom	457.20	1256.29	300
Toy	0.01	0.01	1000
Zoo	0.02	0.04	1000
Colon	0.94	1.70	5000
Divorce	0.09	193.70	500000

Tabela 10 – Tempo médio despendido ao avaliar uma instância como relevante pelo ADePE e pelo Apagoge, em segundos.

Bases	ADePE	Apagoge
Iris Binária	$5.41 \cdot 10^{-5} \pm 2.26 \cdot 10^{-4}$	$8.2 \cdot 10^{-2}$
testNS	$1 \cdot 10^{-16} \pm 1 \cdot 10^{-16}$	$1 \cdot 10^{-16} \pm 1 \cdot 10^{-16}$
lp4	$1.09 \cdot 10^{-3} \pm 6.01 \cdot 10^{-4}$	$1.78 \cdot 10^{-2} \pm 2.34 \cdot 10^{-2}$
Mushroom	$8.10 \cdot 10^{-2} \pm 9.72 \cdot 10^{-3}$	$2.15 \cdot 10^{-1} \pm 4.01 \cdot 10^{-3}$
Toy	$1 \cdot 10^{-16} \pm 1 \cdot 10^{-16}$	$8.34 \cdot 10^{-5} \pm 2.76 \cdot 10^{-4}$
Zoo	$3.22 \cdot 10^{-4} \pm 1.60 \cdot 10^{-3}$	$4.14 \cdot 10^{-4} \pm 6.17 \cdot 10^{-4}$
Colon	$1.57 \cdot 10^{-2} \pm 1.31 \cdot 10^{-3}$	$2.84 \cdot 10^{-2} \pm 5.35 \cdot 10^{-3}$
Divorce	$6.00 \cdot 10^{-4} \pm 5.00 \cdot 10^{-4}$	$7.50 \cdot 10^{-3} \pm 2.17 \cdot 10^{-2}$

Tabela 11 – Tempo médio despendido ao avaliar as instâncias relevante e não relevantes pelo ADePE, em segundos.

Bases	Relevantes	Não Relevantes
Iris Binária	$5.41 \cdot 10^{-5} \pm 2.26 \cdot 10^{-4}$	$2.36 \cdot 10^{-5} \pm 2.6 \cdot 10^{-3}$
testNS	$1 \cdot 10^{-16} \pm 1 \cdot 10^{-16}$	$1.50 \cdot 10^{-3} \pm 1.50 \cdot 10^{-3}$
lp4	$1.09 \cdot 10^{-3} \pm 6.01 \cdot 10^{-4}$	$--$
Mushroom	$8.10 \cdot 10^{-2} \pm 9.72 \cdot 10^{-3}$	$--$
Toy	$1 \cdot 10^{-16} \pm 1 \cdot 10^{-16}$	$--$
Zoo	$3.22 \cdot 10^{-4} \pm 1.60 \cdot 10^{-3}$	$--$
Colon	$1.57 \cdot 10^{-2} \pm 1.31 \cdot 10^{-3}$	$--$
Divorce	$6.00 \cdot 10^{-4} \pm 5.00 \cdot 10^{-4}$	$--$

métodos. É possível perceber que mesmo nas instâncias relevantes o método realiza a avaliação mais rapidamente que a Apagoge.

Por fim, a Tabela 11 mostra o tempo despendido pelo ADePE, separadamente, para as instâncias relevantes e para as não relevantes de uma mesma base. Como esperado, o tempo despendido para avaliar uma instância não relevante é maior que para avaliar uma relevante.

7.3 FORMULAÇÃO DUAL

A versão dual do método opera de modo parecido à sua versão primal, com a diferença entre as duas implementações estando no domínio do espaço onde pontos estão sendo manipulados. Na versão primal os pontos são manipulados no espaço de entrada. Na versão dual os pontos são manipulados no espaço implícito gerado pelo kernel. Deste modo, o algoritmo detecta os pontos relevantes na nuvem de pontos existente no espaço implícito. Devido a isto, para que o método possa executar corretamente as etapas de espelhamento e ortogonalização, é necessário um modo de representar combinações lineares no espaço implícito. Este problema, mencionado na subseção 5.8, apesar de parecer simples, possui solução não trivial. Na próxima sub-seção será descrito uma forma de representação indireta dos pontos que fora criada para solucionar este problema. Na

sub-seções subsequentes, será mostrado como esta forma de representação é usada para o processo inicial de espelhamento das instâncias e para o funcionamento correto do algoritmo em sua versão dual.

7.3.1 Manipulação de pontos no Espaço Implícito

A versão dual do algoritmo procura realizar os mesmos passos da versão primal, porém operando no domínio implícito. Em certas etapas, como o espelhamento, é necessário realizar combinações lineares para gerar novos pontos no domínio implícito, como descrito na Equação 7.10:

$$X_a = \beta \cdot X_c + \gamma \cdot X_d \tag{7.10}$$

onde β e γ são escalares. Além disso, é necessário que a distancia, calculada usando um kernel K, entre um hiperplano dual $W = ([\alpha_1, \alpha_2, ..., \alpha_n], b)$ e o novo ponto gerado pela combinação linear seja feita de forma correta. Seja inicialmente o cálculo da distância dual, baseado no uso de operações válidas de funções kernel:

$$Distancia(W, X_a, K) = b + \beta \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_c \rangle_K + \gamma \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_d \rangle_K \tag{7.11}$$

Entretanto, realizar a combinação linear dada pela Equação 7.6 diretamente no espaço de entrada e utilizar a mesma no cálculo da distância dual pode parecer correto, porém, ao se aplicar um kernel não linear produz-se um resultado final incorreto. Isto pode ser demonstrado introduzindo a relação da combinação linear dada pela Equação 7.6 diretamente na fórmula relacionada ao cálculo da distância dual para uma instância específica. Relembrando que o cálculo de distância dual, de um hiperplano $W = ([\alpha_1, \alpha_2, ..., \alpha_n], b)$ a uma instância qualquer X_j usando um kernel K, é dado por:

$$Distancia(W, X_j, K) = b + \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_j \rangle_K \tag{7.12}$$

Portanto, deseja-se mostrar que:

$$b + \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_a \rangle_K = b + \beta \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_c \rangle_K + \gamma \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_d \rangle_K \tag{7.13}$$

O que leva certamente a uma contradição. Introduzindo a relação 7.10 na parte esquerda da equação tem-se:

$$b + \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, \beta \cdot X_c + \gamma \cdot X_d \rangle_K = b + \beta \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_c \rangle_K + \gamma \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_d \rangle_K \tag{7.14}$$

Eliminando o viés, obtém-se:

$$\sum_{i=1}^{n} \alpha_i \cdot \langle X_i, \beta \cdot X_c + \gamma \cdot X_d \rangle_K = \beta \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_c \rangle_K + \gamma \cdot \sum_{i=1}^{n} \alpha_i \cdot \langle X_i, X_d \rangle_K \tag{7.15}$$

Para garantir que esta equação sempre seja verdadeira, precisa-se que para qualquer instância X_i a seguinte relação de igualdade seja verdadeira:

$$\langle X_i, \beta \cdot X_c + \gamma \cdot X_d \rangle_K = \beta \cdot \langle X_i, X_c \rangle_K + \gamma \cdot \langle X_i, X_d \rangle_K \tag{7.16}$$

Para o kernel linear, polinomial de grau 1, é fácil provar que a Equação 7.16 se mantém, porém para kernel não lineares esta equação não será sempre verdadeira. Um contra exemplo simples é demonstrado a seguir: Considerando kernel polinomial de grau dois K_2, dois pontos pontos $X_c = [2, 2]$ e $X_d = [3, 3]$, um ponto $X_a = 2 \cdot X_c + 1 \cdot X_d$ e uma instância $X_i = [1, 1]$ percebe-se que a relação de igualdade da Equação 7.16 não se satisfaz:

$$\langle [1, 1], 2 \cdot [2, 2] + 1 \cdot [3, 3] \rangle_{K_2} = 2 \cdot \langle [1, 1], [2, 2] \rangle_{K_2} + 1 \cdot \langle [1, 1], [3, 3] \rangle_{K_2} \tag{7.17}$$

$$\langle [1, 1], [7, 7] \rangle_{K_2} = 2 \cdot \langle [1, 1], [2, 2] \rangle_{K_2} + \langle [1, 1], [3, 3] \rangle_{K_2} \tag{7.18}$$

$$(1 \cdot 7 + 1 \cdot 7)^2 + 0 = 2 \cdot ((1 \cdot 2 + 1 \cdot 2)^2) + ((1 \cdot 3 + 1 \cdot 3)^2) \tag{7.19}$$

$$14^2 = 2 \cdot (4^2) + 6^2 \tag{7.20}$$

Que leva a um absurdo:

$$196 = 68 \tag{7.21}$$

Este contra exemplo mostra que, ao se utilizar kernel não lineares, não se pode realizar combinações lineares de pontos no espaço de entrada, pois isto não garante que a distância de um hiperplano ao ponto gerado seja uma combinação linear das distâncias dos pontos combinados.

A solução criada para este problema envolve uma forma de representação indireta para os pontos, e uma fórmula para cálculo da distância que utiliza esta representação. Esta solução possibilita que combinações lineares sejam feitas corretamente. Primeiro será definido a forma de representação. Nela, cada ponto é agora representado por um vetor de *alphas*. Dado um ponto X_a, seu vetor de *alphas* correspondente V_a tem seus valores iguais

a 0, exceto no índice a, onde possui valor igual a 1. A Equação 7.22 mostra exemplos dos vetores relacionados as instâncias X_1, X_2 e X_n:

$$\begin{aligned} V_1 &= [1, 0, ..., 0] \\ V_2 &= [0, 1, ..., 0] \\ V_n &= [0, 0, ..., 1] \end{aligned} \tag{7.22}$$

Em seguida é definido uma fórmula para o cálculo do produto interno entre dois vetores de alphas V_a e V_b que representam respectivamente os pontos X_a e X_b. Esta fórmula é definida pela seguinte equação:

$$\text{ProdInt}_\alpha(V_a, V_b, K) = \sum_{i=1}^{n}\{V_{a,i} \cdot \sum_{j=1}^{n}(V_{b,j} \cdot \langle X_j, X_i \rangle_K)\} \tag{7.23}$$

A partir desta equação define-se uma nova fórmula para o cálculo da distância entre um hiperplano dual $W = (W_\alpha = [\alpha_1, \alpha_2, ..., \alpha_n], b)$ e um vetor de alphas V_a que representa um ponto X_a. Esta é definida pela Equação 7.24:

$$Distancia_\alpha(W, V_a, K) = b + ProdInt_\alpha(W_\alpha, V_a, K) \tag{7.24}$$

Agora, deve-se provar que este cálculo de distância resulta no mesmo valor que o cálculo correto de distância dual utilizando-se a Equação 7.12 e a nova forma de representação por vetores de alphas. Esta demonstração inicia-se separando as parcelas do somatório externo:

$$Distancia_\alpha(W, V_a, K) = b + V_{a,1} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_1 \rangle_K) + ... + V_{a,n} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_n \rangle_K) \tag{7.25}$$

Como, pela definição dos vetores relacionados, todos os valores de V_a são iguais a 0, exceto o valor do índice a que vale 1, a Equação 7.25 é simplificada para:

$$Distancia_\alpha(W, V_a, K) = b + 1 \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_a \rangle_K) \tag{7.26}$$

Esta é a mesma equação dada pelo cálculo de distância dual utilizando-se a equação 7.12. Logo, pode-se afirmar que calcular uma distância entre um hiperplano dual W e uma instância X_a pelo método tradicional equivale a calcular a distância entre um hiperplano dual W e o vetor de *alphas* V_a relacionado á instância X_a.

A partir desta nova representação é possível solucionar o problema inicial relacionado à realização de combinações lineares no espaço de entrada. Considerando um vetor de alphas definido por $V_a = (\beta \cdot V_c + \gamma \cdot V_d)$, pode-se demonstrar que o novo cálculo de

distância dual baseado na utilização da Equação 7.12 e utilizando a representação por vetores de alphas se mostra correto. Seja inicialmente:

$$Distancia_{\alpha}(W, V_a, K) = b + \beta \cdot \sum_{i=1}^{n}\{V_{c,i} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_i \rangle_K)\} + \gamma \cdot \sum_{i=1}^{n}\{V_{d,i} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_i \rangle_K)\} \tag{7.27}$$

Considera-se agora o novo calculo da distância dual baseado nas equações 7.12 e 7.23:

$$Distancia_{\alpha}(W, V_a, K) = b + \sum_{i=1}^{n}(V_{a,i} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_i \rangle_K)) \tag{7.28}$$

Substituindo V_a por $(\beta \cdot V_c + \gamma \cdot V_d)$ obtém-se:

$$b + \sum_{i=1}^{n}((\beta \cdot V_{c,i} + \gamma \cdot V_{d,i}) \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_i \rangle_K)) \tag{7.29}$$

Aplicando a propriedade distributiva e reorganizando a Equação 7.29 em dois vetores V_c e V_d, tem-se:

$$b + \beta \cdot \sum_{i=1}^{n}\{V_{c,i} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_i \rangle_K)\} + \gamma \cdot \sum_{i=1}^{n}\{V_{d,i} \cdot \sum_{j=1}^{n}(\alpha_j \cdot \langle X_j, X_i \rangle_K)\} \tag{7.30}$$

Resultando no lado direito da Equação 7.27 a qual deseja-se provar. Portanto, ao se utilizar esta representação de vetores de alphas, pode-se realizar operações lineares em pontos do domínio implícito. Assim, permitindo o funcionamento do ADePE na formulação dual, como será mostrado nas próximas sub-seções.

Para exemplificar o funcionamento desta representação, considera-se o seguinte exemplo: Para um kernel polinomial, K, de grau 2; um conjunto $\mathcal{X}$ formado por três pontos $X_1 = [2, 2]$, $X_2 = [3, 3]$ e $X_3 = [1, 1]$, e suas respectivas representações por vetores de alphas $V_1 = [1, 0, 0]$ e $V_2 = [0, 1, 0]$ e $V_3 = [0, 0, 1]$; um vetor de alphas $V_{comb} = (2 \cdot V_1 + V_2)$ representando uma combinação linear realizada no espaço expandido; e um hiperplano dual $W = \{W_\alpha = [0, 0, 1], b\}$. Ao se utilizar fórmula para cálculo do produto interno entre dois vetores de alphas, dada pela Equação 7.23, será mostrado que a seguinte equação se satisfaz:

$$ProdInt_{\alpha}(W_\alpha, V_{comb}, K) = 2 \cdot ProdInt_{\alpha}(W_\alpha, V_1, K) + ProdInt_{\alpha}(W_\alpha, V_2, K) \tag{7.31}$$

Inicia-se calculando o valor numérico da parte direita da equação acima:

$$2 \cdot \text{ProdInt}_\alpha(W_\alpha, V_1, K) + \text{ProdInt}_\alpha(W_\alpha, V_2, K) \tag{7.32}$$

Ao substituir a fórmula de produto interno de vetores de alphas obtém-se:

$$2 \cdot \{\sum_{i=1}^{3}(W_{\alpha,i} \cdot \sum_{j=1}^{3}(V_{1,j} \cdot \langle X_j, X_i\rangle_K))\} + \sum_{i=1}^{3}(W_{\alpha,i} \cdot \sum_{j=1}^{3}(V_{2,j} \cdot \langle X_j, X_i\rangle_K)) \tag{7.33}$$

Novamente simplificando os somatórios externos devido a W_α ter valores iguais a zero exceto no terceiro índice:

$$2 \cdot \{W_{\alpha,3} \cdot \sum_{j=1}^{3}(V_{1,j} \cdot \langle X_j, X_3\rangle_K)\} + W_{\alpha,3} \cdot \sum_{j=1}^{3}(V_{2,j} \cdot \langle X_j, X_3\rangle_K) \tag{7.34}$$

$$2 \cdot \{1 \cdot \sum_{j=1}^{3}(V_{1,j} \cdot \langle X_j, X_3\rangle_K)\} + 1 \cdot \sum_{j=1}^{3}(V_{2,j} \cdot \langle X_j, X_3\rangle_K) \tag{7.35}$$

Simplificando os somatórios devido aos vetores V_c e V_d só possuírem valores não zero em um índice cada:

$$2 \cdot \langle X_1, X_3\rangle_K + \langle X_2, X_3\rangle_K \tag{7.36}$$

Substituindo os valores numéricos na Equação 7.36 obtém-se:

$$2 \cdot (2 \cdot 1 + 2 \cdot 1)^2 + 1 \cdot (3 \cdot 1 + 3 \cdot 1)^2 = 68 \tag{7.37}$$

Portanto pode-se afirmar que:

$$ProdInt_\alpha(W_\alpha, V_{comb}, K) = 68 \tag{7.38}$$

Agora será analisado a parte esquerda da equação. Onde será calculado o valor de $ProdInt_\alpha(W_\alpha, V_{comb}, K)$. Utilizando a definição dada pela Equação 7.19 obtém-se:

$$ProdInt_\alpha(W_\alpha, V_{comb}, K) = \sum_{i=1}^{3}\{W_{\alpha,i} \cdot \sum_{j=1}^{3}(V_{comb,j} \cdot \langle X_j, X_i\rangle_K)\} \tag{7.39}$$

Seguindo os mesmos passos anteriores, o somatório externo pode ser simplificado:

$$1 \cdot \sum_{j=1}^{3}(V_{comb,j} \cdot \langle X_j, X_3\rangle_K) \tag{7.40}$$

Expandindo os fatores do somatório obtém-se:

$$2 \cdot \langle X_1, X_3 \rangle_K + 1 \cdot \langle X_2, X_3 \rangle_K \tag{7.41}$$

Resultando na mesma fórmula obtida na Equação 7.36 e cujo valor já foi calculado:

$$2 \cdot \langle X_1, X_3 \rangle_K + 1 \cdot \langle X_2, X_3 \rangle_K = 68 \tag{7.42}$$

Assim é demonstrado, neste exemplo numérico, que ao se utilizar a representação por vetores de alphas pode-se realizar corretamente as combinações lineares diretamente no espaço de entrada.

Por fim, é importante ressaltar que esta forma de representação tem utilidade não somente para o oráculo ADePE dual, mas possui um potencial para ser usada na criação de outros métodos que envolvam a formulação dual. Uma utilização poderosa desta formulação é para criar uma base ortogonal do domínio implícito, o que permite acesso a todos os pontos pertencentes a ele, e não somente os pontos que são acessados diretamente a partir da aplicação do *kernel trick* nos pontos do espaço de entrada. Considerando que o domínio implícito tem, usualmente, dimensionalidade superior a do domínio de entrada, e que o *kernel trick* envolve um mapeamento de um ponto do domínio de entrada para apenas um ponto do domínio implícito, não se tem a garantia de que todos os pontos do domínio implícito possam ser acessados a partir dos pontos do espaço de entrada. Porém, ao utilizar esta representação, é possível gerar uma base ortogonal do domínio implícito, e a partir dela ter acesso a todos os pontos pertencentes a ele. Isto tem um grande potencial para a criação de novos métodos duais.

A seguir será mostrado um exemplo da utilização da representação por vetores de alphas para gerar uma base no domínio implícito, e poder acessar pontos que não seriam acessíveis usando a representação direta. Considere um conjunto de pontos unidimensionais $\mathcal{X} \in \mathbb{R}^1$ composto por $X_1 = [-2]$, $X_2 = [-1.5]$, $X_3 = [-1]$, $X_4 = [0]$, $X_5 = [1]$, $X_6 = [1.5]$ e $X_7 = [2]$; e um kernel K que mapeie um ponto unidimensional qualquer $[x]$ para um ponto bidimensional $[x, x^2]$. A Figura 21b mostra a visualização do kernel K mapeando o conjunto $\mathcal{X}$. É fácil perceber que todos os pontos mapeados no espaço implícito ficarão ao longo de uma parábola, como indicado pela linha pontilhada laranja pela Figura 21c:

A partir desta parábola é possível perceber que o kernel K não gera todos os pontos de $\mathbb{R}^2$. Deste modo, ao se efetuar as combinações lineares diretamente no espaço de entrada limita-se a quantidade de pontos do domínio implícito que se tem acesso a apenas os que estejam na parábola laranja, assim tornando impossível acessar certos pontos como por exemplo o ponto $p = [0, 2]$ como indicado na Figura 22a. Porém, ao utilizar a representação por vetores de alphas é possível gerar uma base ortogonal no espaço implícito, e usar a mesma para acessar qualquer ponto. Considere um conjunto de vetores de alphas $\mathcal{V}$ usado para representar os pontos do conjunto $\mathcal{X}$, ou seja, os pontos X_1, X_2,

X_3, X_4, X_5, X_6 e X_7 são representados respectivamente pelos vetores $V_1 = [1,0,0,0,0,0,0]$, $V_2 = [0,1,0,0,0,0,0]$, $V_3 = [0,0,1,0,0,0,0]$, $V_4 = [0,0,0,1,0,0,0]$, $V_5 = [0,0,0,0,1,0,0]$, $V_6 = [0,0,0,0,0,1,0]$ e $V_7 = [0,0,0,0,0,0,1]$. A partir destes vetores pode-se descrever uma base no espaço implícito formada pelos seguintes vetores de alphas: $V_a = V_3 - V_4$ e $V_b = V_5 - V_4$, ou seja: $V_a = [0,0,1,-1,0,0,0]$ e $V_b = [0,0,0,-1,1,0,0]$. A Figura 22b mostra graficamente estes dois vetores no espaço implícito. A partir destes dois vetores de alphas pode-se por meio de uma combinação linear gerar um vetor de alphas que representa o ponto $[0,2]$, sendo este representado por $V_p = V_a + V_b$, ou seja $V_p = [0,0,1,-2,1,0,0]$.

Esta forma de representação possibilita acesso a pontos que seriam impossíveis de acessar usando apenas o mapeamento de pontos do espaço de entrada. Isto é essencial para o funcionamento do ADePE dual, e além disto, oferece potencial para a criação de novos algoritmos para classificação.

7.3.2 Espelhamento

A operação de espelhamento tem o seguinte objetivo: Considerando uma instância X_p denominada de **p**ivô; o sub-espaço de versões de X_p, definido como o conjunto de todos os classificadores factíveis que possuem distância zero a X_p; e uma instância X_o de classe **o**posta a do pivô. A operação de espelhamento procura encontrar uma instância X_e, de classe igual a do pivô, tal que ela imponha a mesma restrição que X_o no subespaço de versões do pivô.

Na formulação primal, o espelhamento é realizado diretamente no domínio de entrada, como explicado na Sub-seção 6.4, ao gerar uma instância X_e aplicando a Equação 7.43:

$$X_e = X_p - D \tag{7.43}$$

onde $D = X_o - X_p$. Esta equação pode ser reescrita como uma combinação linear de X_p e X_o ao expandir D:

$$X_e = 2 \cdot X_p - X_o \tag{7.44}$$

Porém, como demonstrado na sub-seção anterior, na formulação dual não é possível realizar combinações lineares diretamente no espaço de entrada, caso esteja-se operando com um kernel não linear. Para que a operação de espelhamento funcione na formulação dual com qualquer kernel, é necessário usar a representação de vetores de alphas introduzida na sub-seção anterior. O Teorema 7.3.1 demonstra como o espelhamento é realizado no dual.

Teorema 7.3.1. *Sejam: uma instância X_p e seu vetor de alphas relacionado V_p; uma instância X_o, de classe oposta a I_p, e seu vetor de alphas relacionado V_o; e uma instância*

I_e, de classe igual a I_p, representada por um vetor de alphas $V_e = 2 \cdot V_p - V_o$. A restrição imposta pela instância X_e no sub-espaço de versões dual de X_p é a mesma restrição que a imposta pela instância X_o, independente do kernel K usado no cálculo de distância.

Demonstração. Deseja-se provar que para qualquer hiperplano dual $W = ([\alpha_1, \alpha_2, ..., \alpha_n], b)$, cuja distância à instância pivô X_p é igual a 0, a Equação 7.45 é verdadeira:

$$Y_o \cdot Distancia(W, V_o, K) = Y_e \cdot Distancia(W, V_e, K) \tag{7.45}$$

Como pela definição de V_e tem-se que $V_e = 2 \cdot V_p - V_o$, pode-se reescrever a distância de W a V_e da seguinte forma:

$$Distancia(W, V_e, M) = b + 2 \cdot \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_p[j] \cdot \langle X_j, X_i\rangle_K)\} - \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_o[j] \cdot \langle X_j, X_i\rangle_K)\} \tag{7.46}$$

Assim, é possível redefinir a restrição imposta por I_e como:

$$Y_e \cdot Distancia(W, X_e, M) = Y_e \cdot (b + 2 \cdot \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_p[j] \cdot \langle X_j, X_i\rangle_K)\} - \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_o[j] \cdot \langle X_j, X_i\rangle_K)\}) \tag{7.47}$$

Como, por definição do hiperplano W, tem-se que $Distancia(W, V_p, M) = 0$, conseuqentemente a restrição de I_e pode ser simplificada para:

$$Y_e \cdot (0 + \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_p[j] \cdot \langle X_j, X_i\rangle_K)\} - \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_o[j] \cdot \langle X_j, X_i\rangle_K)\}) \tag{7.48}$$

Considerando que $Distancia(W, V_p, M) = 0$, pode-se afirmar então que:

$$b = -\sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_p[j] \cdot \langle X_j, X_i\rangle_K)\} \tag{7.49}$$

Assim, pode-se aplicar a Equação 7.49 para reescrever a Equação 7.48 como:

$$Y_e \cdot (0 - b - \sum_{i=1}^{n}\{\alpha_i \cdot \sum_{j=1}^{n}(V_o[j] \cdot \langle X_j, X_i\rangle_K)\}) \tag{7.50}$$

Esta equação pode ser simplificada usando a notação de distância entre vetores de alphas definida em 7.24:

$$Y_e \cdot (-1 \cdot Distancia(W, V_o, M)) \tag{7.51}$$

Como I_e é da classe oposta a I_o:

$$Y_e \cdot (-1 \cdot Distancia(W, V_o, M)) = -Y_o \cdot (-1 \cdot Distancia(W, V_o, M)) \tag{7.52}$$

Simplificando os dois sinais negativos obtém-se:

$$-Y_o \cdot (-1 \cdot Distancia(W, V_o, M)) = Y_o \cdot Distancia(W, V_o, M)) \tag{7.53}$$

Que é a equação da restrição imposta pela instância X_o, assim provando que ambas as instâncias X_e e X_o impõem a mesma restrição no sub-espaço de versões dual relacionado à instância X_p, independente do kernel usado.

□

Isto prova que a técnica de espelhamento pode ser realizada na formulação dual, independentemente do kernel, por meio da utilização da representação dos vetores de alphas, ao aplicar a Equação 7.44 não nas instâncias, mas nos vetores de alphas relacionados a elas. Portanto, é importante perceber que esta operação de espelhamento dual gera o vetor de alphas da instância espelhada, sem precisar gerar as suas coordenadas do espaço de entrada. Isto é uma grande vantagem, pois, nem todos os pontos existentes no domínio implícito tem um correspondente no domínio de entrada. Portanto, caso a instância espelhada esteja em um destes pontos, isto não será um problema para realizar o espelhamento.

7.3.3 Ortogonalização no domínio implícito

Outra operação importante realizada pelo ADePE é a de encontrar um hiperplano que contém d pontos. Como explicado na sub-seção 7.2.4, encontrar este hiperplano é o mesmo que gerar uma base ortogonal a partir de um conjunto de d pontos. Podemos utilizar a representação por vetores de alphas para aplicar algoritmos de ortogonalização, como por exemplo o Procedimento de Gram-Schimdt [13], Transformações de Householder [32], no espaço implícito. A adaptação destes métodos para funcionarem nos vetores de alphas necessita da realização de uma multiplicação escalar entre dois vetores de alphas, para tanto deve ser usado a Equação 7.54,

$$ME(V_a, V_b, K) = \sum_{i=1}^{n}(V_a[i] \cdot \sum_{j=1}^{n}(V_b[j] \cdot \langle X_j, X_i \rangle_K)) \tag{7.54}$$

Esta equação é gerada a partir da Equação 7.24 de distância entre um hiperplano dual e um vetor de alphas, sendo ela a parte referente a multiplicação escalar entre dois vetores existentes no cálculo da fórmula de distância.

7.3.4 ADePE dual

A formulação dual do algoritmo segue as mesmas etapas de sua versão primal, apresentada na Figura 16, porém com algumas adaptações envolvendo a representação por vetores de alphas para operar no domínio implícito. Como demonstrado nas sub-seções anteriores, a etapa de espelhamento não pode ser realizada nos dados de entrada. Logo, o algoritmo dual não opera diretamente nos dados de entrada, ao invéz disso, ele gera vetores de alphas relacionados para os dados de entrada, e então os utiliza em todos as suas etapas do mesmo modo que a versão primal usaria os dados de entrada.

Outra etapa do algoritmo que necessita desta representação é a de encontrar um hiperplano que contenha um sub-conjunto de pontos. Ao utilizar esta forma de representação, tem-se que tanto o hiperplano quanto os pontos estarão sendo representados por um vetor de alphas. Apesar disto, é possível aplicar o mesmo algoritmo definido para o processo de ortogonalização da versão primal, havendo apenas uma troca na forma de representação dos vetores.

Por fim, a versão dual do algoritmo, por trabalhar no domínio implícito, necessita saber a quantidade de dimensões que formam este domínio. Devido a natureza do *kernel trick*, a quantidade de dimensões pode ser um valor desconhecido a priori. Porém, é possível descobrir este valor ao aplicar o algoritmo de ortogonalização de forma sucessiva. Começando com um hiperplano de direção qualquer que contenha o pivô, ele deve ser ortogonalizado para conter um dos pontos do conjunto espelhado, caso a normal não tenha se tornado igual ao vetor zero. Após este processo, ele é repetido para outro ponto do conjunto espelhado que não tenha sido escolhido ainda, e que não seja o pivô. A quantidade de dimensões do domínio implícito será igual a quantidade de vezes que a ortogonalização for realizada. Caso todos os pontos sejam escolhidos então isto significa que a quantidade de dimensões é maior que a quantidade de pontos do conjunto, o que implica numa solução trivial que pode ser formada ao gerar um hiperplano que contenha todos os pontos. O diagrama apresentado na Figura 23 detalha o processo deste método de calcular a dimensão no domínio implícito.

7.3.5 Experimentos da formulação dual

Para validar o funcionamento do método na formulação dual foram realizados dois experimentos. O primeiro com intuito de verificar se ele determina corretamente quais são os pontos relevantes. O segundo experimento analisou o tempo necessário para verificação de uma instância. Também foi gerado a visualização gráfica das instâncias relevantes de duas bases não linearmente separáveis geradas artificialmente: a base XOR-400 que contém 400 instâncias e duas dimensões, sendo ela inspirada no clássico problema não linear XOR; e a base Circle-200 contém 200 instâncias e duas dimensões, ela apresenta as instâncias de uma classe dispostas próximas do centro, e as instâncias da outra classe

ao redor da primeira classe, que a envolve. Além destas duas bases artificiais, também foram usadas outras 9 bases reais: Ionosphere [85], Pima [86], Sonar [87], wdbc [88], Wine [89], Bupa [74], Hepatitis [83], Heart [84], Fertility [90]. Nestes experimentos, o oráculo Apagoge versão dual foi executado por um total máximo de 10000 épocas.

7.3.5.1 Análise de Corretude

Para avaliar a eficácia da versão dual utilizou-se bases não linearmente separáveis e kernel quadrático. Foram realizadas comparações entre os resultados obtidos pelo ADePE Dual com o oráculo Apagoge aplicado no Perceptron Dual. A Tabela 12 apresenta os resultados. Pela tabela pode-se perceber que o ADePE Dual conseguiu rotular uma quantidade maior de instâncias como relevantes em cinco das bases utilizadas. Nas outras três foram rotuladas a mesma quantidade que o Apagoge Dual.

Tabela 12 – Instâncias relevantes detectadas, pelo ADePE Dual e pelo Apagoge Dual, em bases de dados não linearmente separáveis usando o kernel quadrático.

	Circle-200	**XOR-400**	**Ionosphere**	**Pima**	**Sonar**	**wdbc**	**Wine**	**Bupa**	**Hepatitis**	**Heart**	**Fertility**
Dimensões	2	2	34	8	60	30	13	6	19	13	10
Inst. Totais	200	400	350	768	208	569	178	341	155	270	100
ADePE Dual	27	40	350	0	208	569	177	0	155	46	100
Apagoge Dual	23	21	324	0	208	0	0	0	0	0	100

7.3.5.2 Tempo de execução

Para avaliar a viabilidade do método foi analisado seu tempo de processamento. A Tabela 13 apresenta os tempos despendidos pelo Apagoge Dual e o tempo despendido pelo ADePE Dual nas bases não linearmente separáveis. É possível perceber que os resultados obtidos pelo método proposto foram mais rápidos que os resultados obtidos pelo oráculo Apagoge Dual.

Também foi comparado o tempo necessário para definir uma instância como relevante. A Tabela 14 mostra que o método proposto obteve resultados superiores.

O tempo despendido pelo ADePE para avaliar uma instância relevante foi comparado com o tempo para avaliar uma instância não relevante. A Tabela 15 mostra os resultados obtidos.

Também foi analisado a etapa de descobrir quantas dimensões existem no espaço implícito. A Tabela 16 contém a quantidade e dimensões do espaço implícito, e o tempo despendido para descobrir este valor. É importante lembrar que este passo só precisa ser executado uma única vez, e seu resultado é aproveitado ao analisar cada instância.

7.3.5.3 Visualização gráfica

As imagens 24a e 24b mostram uma visualização das instâncias relevantes das bases Circle-200 e da base XOR-400 encontradas pelo ADePE Dual com kernel quadrático.

Tabela 13 – Tempo de execução total, em segundos, despendido pelo ADePE Dual e pelo Apagoge Dual, ambos com kernel quadrático.

Bases	ADePE Dual	Apagoge Dual
Circle-200	0.486	29.747
XOR-400	0.937	67.082
Iosnosphere	49.252	160.447
Pima	222.890	3807.471
Sonar	14.466	29.948
wdbc	646.972	919.152
Wine	27.262	14794.599
Bupa	347.523	1073.511
Hepatitis	14.455	376.163
Heart	8334.351	7927.939
Fertility	0.466	0.748

Tabela 14 – Tempo de execução médio para avaliar uma instância relevante, em segundos, despendido pelo pelo AdePe Dual e pelo Apagoge Dual.

Bases	ADePE Dual	Apagoge Dual
Circle-200	$2.59 \cdot 10^{-4} \pm 4.38 \cdot 10^{-4}$	$1.41 \cdot 10^{-2} \pm 2.50 \cdot 10^{-2}$
XOR-400	$8.75 \cdot 10^{-4} \pm 7.48 \cdot 10^{-4}$	$3.73 \cdot 10^{-2} \pm 3.57 \cdot 10^{-2}$
Iosnosphere	$4.76 \cdot 10^{-2} \pm 2.62 \cdot 10^{-2}$	$1.14 \cdot 10^{-1} \pm 1.12 \cdot 10^{-1}$
Sonar	$3.32 \cdot 10^{-2} \pm 8.87 \cdot 10^{-3}$	$1.40 \cdot 10^{-1} \pm 2.40 \cdot 10^{-2}$
Fertility	$4.20 \cdot 10^{-4} \pm 5.13 \cdot 10^{-4}$	$5.20 \cdot 10^{-3} \pm 9.24 \cdot 10^{-3}$

Tabela 15 – Tempo de execução médio despendido pelo pelo AdePe Dual, em segundos, para avaliar instâncias relevantes e não relevantes.

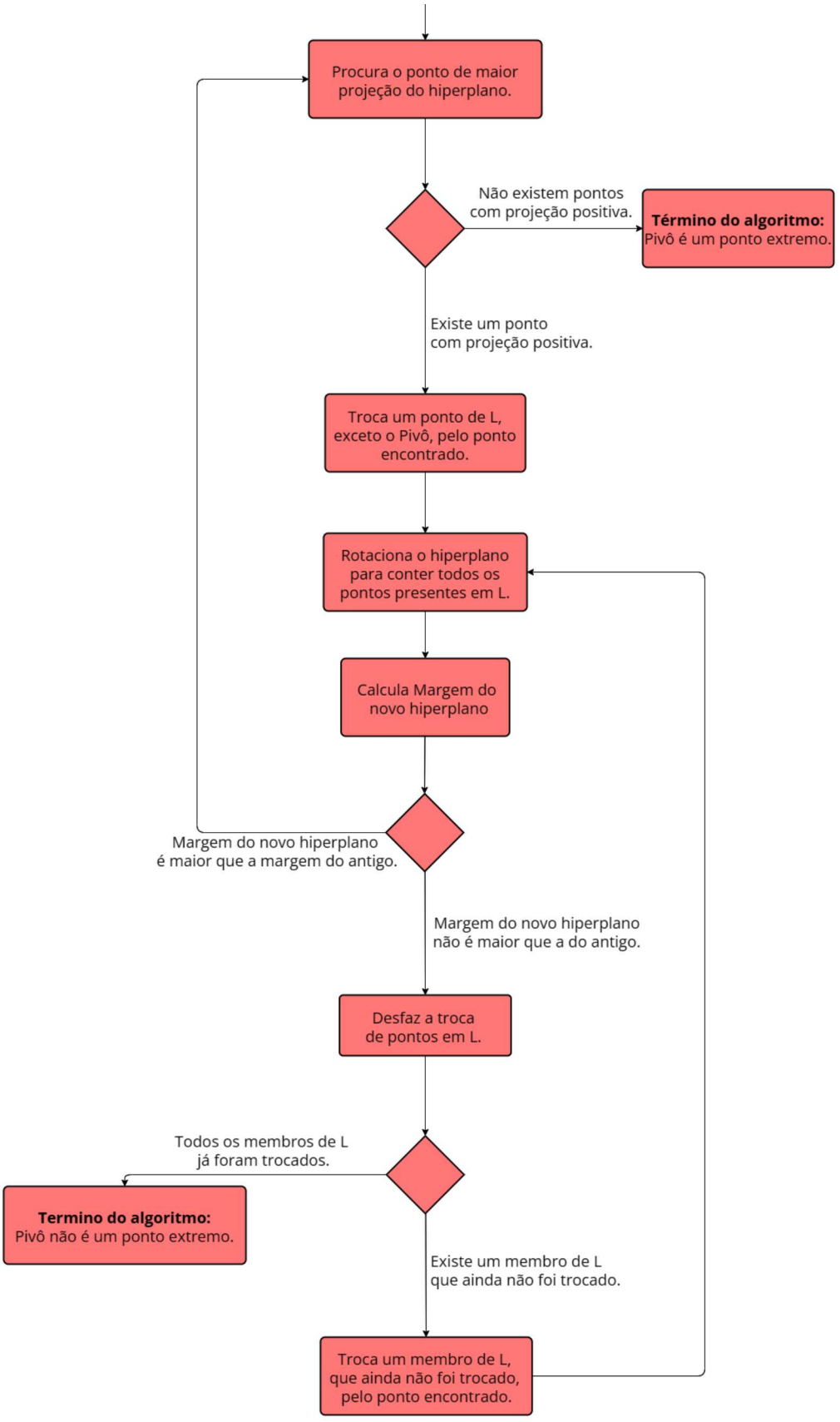
Bases	Relevantes	Não Relevantes
Circle-200	$2.59 \cdot 10^{-4} \pm 4.38 \cdot 10^{-4}$	$2.43 \cdot 10^{-4} \pm 4.55 \cdot 10^{-4}$
XOR-400	$8.75 \cdot 10^{-4} \pm 7.48 \cdot 10^{-4}$	$7.33 \cdot 10^{-4} \pm 7.31 \cdot 10^{-4}$
Iosnosphere	$4.76 \cdot 10^{-2} \pm 2.62 \cdot 10^{-2}$	$---$
Pima	$---$	$2.86 \cdot 10^{-1} \pm 1.58 \cdot 10^{-1}$
Sonar	$3.32 \cdot 10^{-2} \pm 8.87 \cdot 10^{-3}$	$---$
wdbc	$9.98 \cdot 10^{-1} \pm 1.59 \cdot 10^{-0}$	$---$
Wine	$1.20 \cdot 10^{-2} \pm 1.08 \cdot 10^{-2}$	$21.52 \cdot 10^{-0} \pm 0.00 \cdot 10^{-0}$
Bupa	$---$	$1.02 \cdot 10^{0} \pm 8.79 \cdot 10^{-1}$
Hepatitis	$8.40 \cdot 10^{-2} \pm 1.83 \cdot 10^{-2}$	$---$
Heart	$1.96 \cdot 10^{-1} \pm 1.57 \cdot 10^{-2}$	37.16 ± 59.35
Fertility	$4.20 \cdot 10^{-4} \pm 5.13 \cdot 10^{-4}$	$---$

Tabela 16 – Tempo de execução despendido pelo ADePE Dual, em segundos, para contar a quantidade de dimensões no espaço implícito, e a quantidade de dimensões contadas do espaço implícito comparadas com a quantidade de dimensões do espaço de entrada.

Bases	**Tempo**	**Dimensão Implícita**	**Dimensão Entrada**
Circle-200	0.002	3	2
XOR-400	0.006	3	2
Ionosphere	28.496	348	34
Pima	0.353	36	8
Sonar	3.595	206	60
wdbc	81.999	346	30
Wine	0.657	92	13
Bupa	0.045	39	6
Hepatitis	1.047	151	20
Heart	1.199	92	13
Fertility	$8.2 \cdot 10^{-2}$	55	100

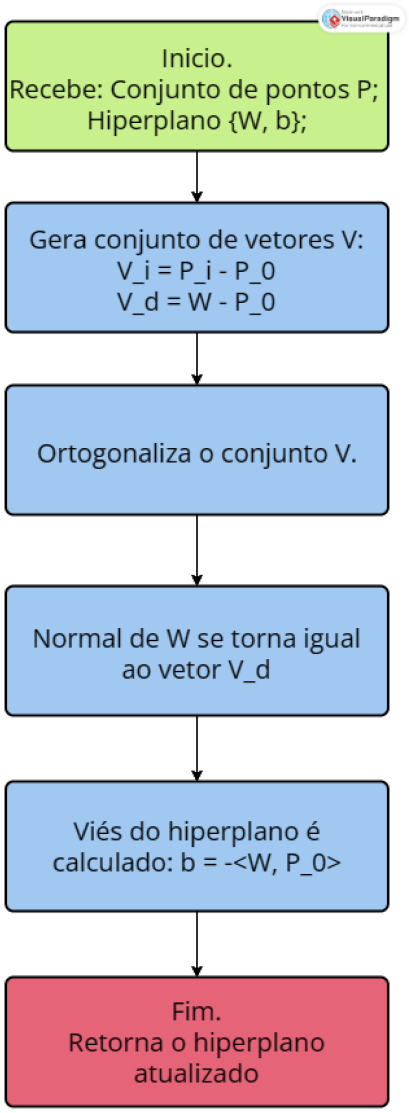
Essas imagens mostram que os pontos relevantes, como esperado, são aqueles próximos da fronteira entre as duas classes.

Figura 19 - Fluxograma da terceira etapa do algoritmo de detecção de pontos extremos.



Fonte: Própria produção.

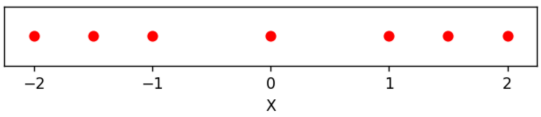
Figura 20 - Diagrama do algoritmo de ortogonalização de pontos extremos.



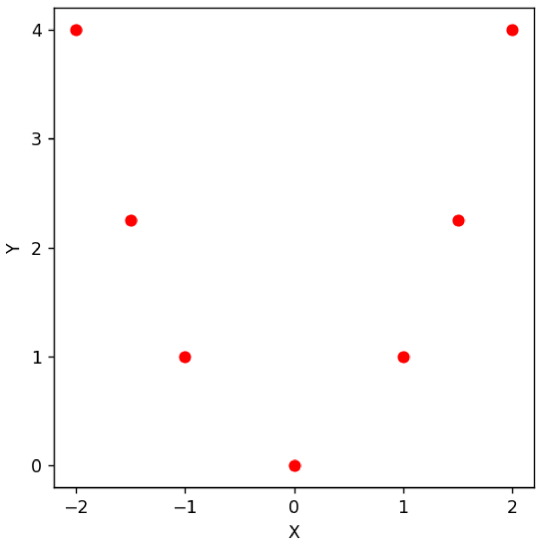
Fonte: Própria produção.

Figura 21 - Mapeamento do espaço de entrada pelo kernel K.

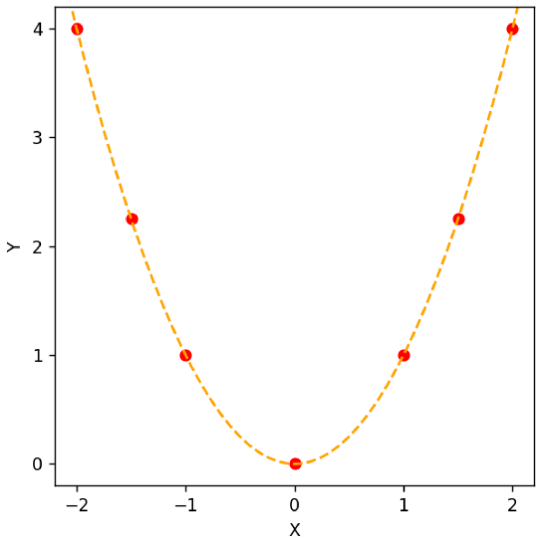
(a) Amostras do conjunto unidimensional $\mathcal{X}$.



(b) Amostras do conjunto $\mathcal{X}$ mapeadas pelo kernel K para um espaço bidimensional.



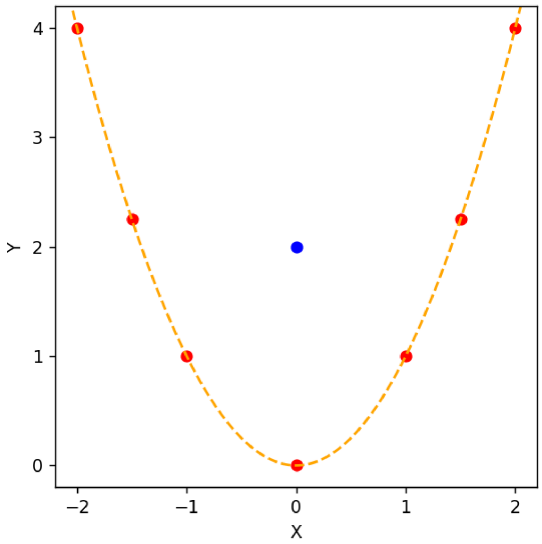
(c) Linha pontilhada laranja indicando a parábola que contém as possíveis coordenadas que o kernel K pode gerar.



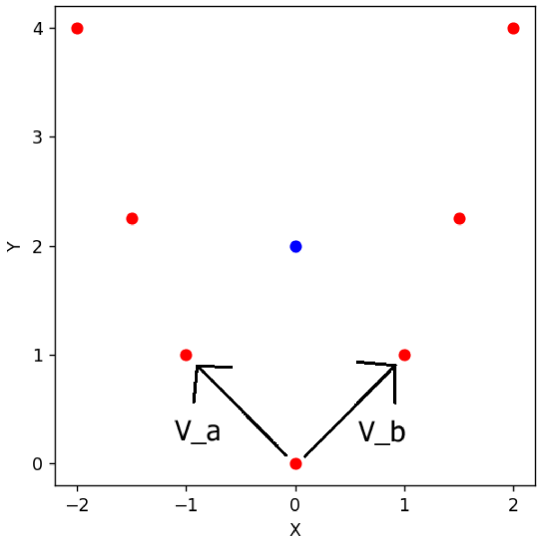
Fonte: Própria produção.

Figura 22 - Exemplo da utilização da representação por vetores de alphas para gerar uma base no espaço implícito.

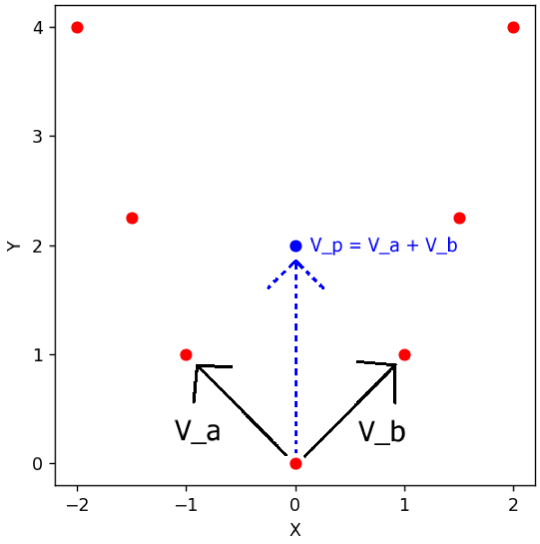
(a) Como o ponto $p = [0, 2]$ não está contido na parábola, logo ele não pode ser acessado ao mapear diretamente um ponto do espaço de entrada.



(b) Os vetores de alphas V_a e V_b formam uma base do espaço implícito de K.

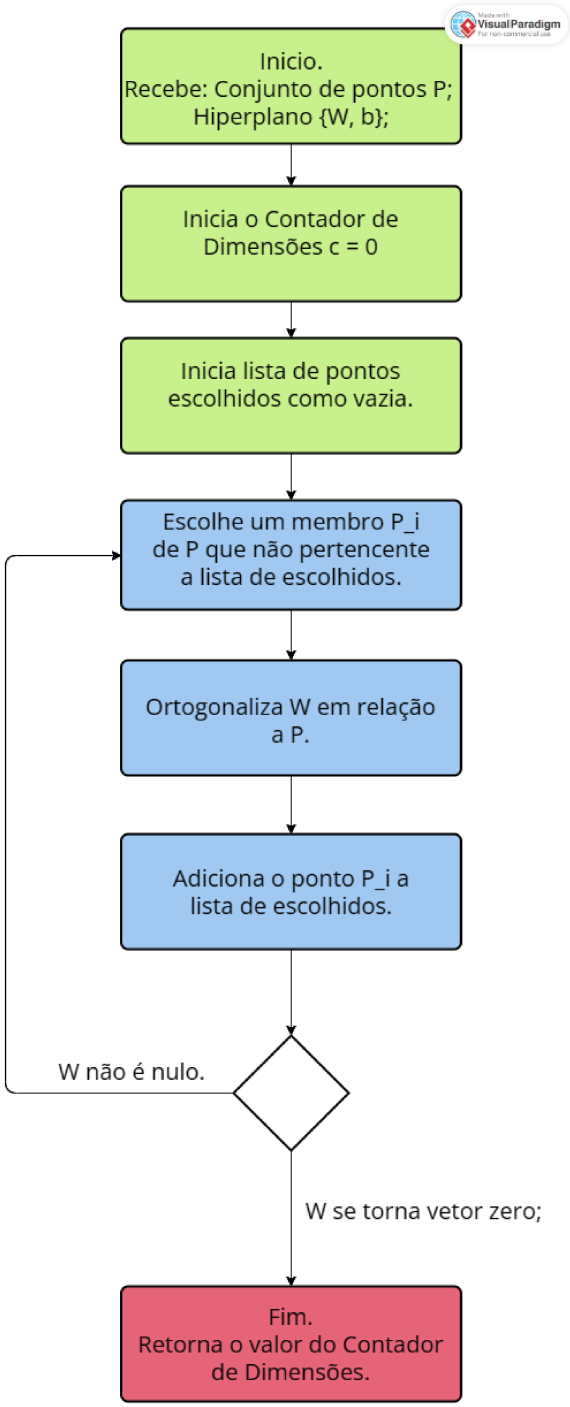


(c) A base formada por V_a e V_b é usada para gerar um vetor de alphas que consegue representar o ponto p.



Fonte: Própria produção.

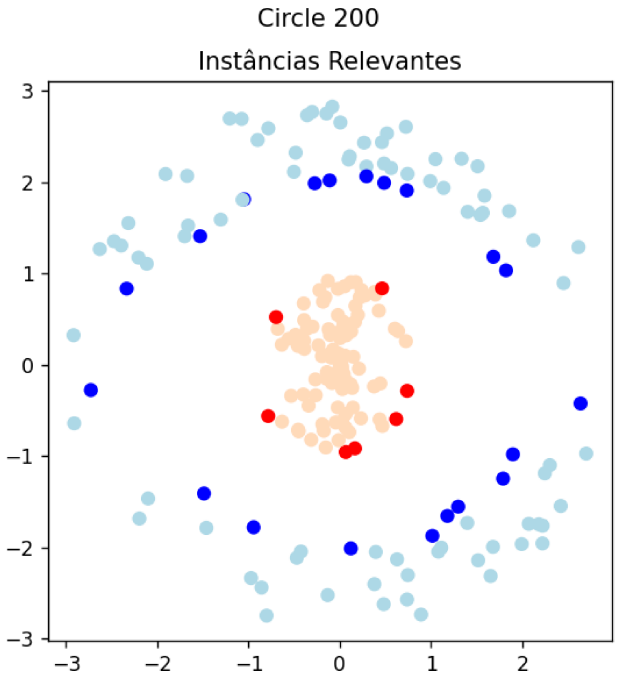
Figura 23 - Diagrama do algoritmo para descobrir as dimensões do domínio implícito na formulação dual. Em verde está a inicialização. Em azul o processo iterativo para contar as dimensões. Em vermelho a condição de para e retorno.



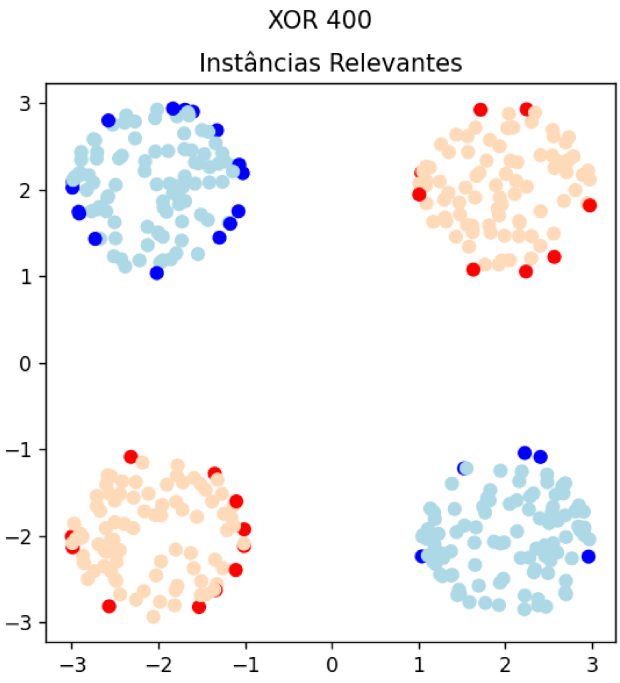
Fonte: Própria produção.

Figura 24 - Em azul claro estão as instâncias não relevantes da classe positiva, em azul escuro as instâncias relevantes da classe positiva. Em vermelho claro estão as instâncias não relevantes da classe negativa, em vermelho escuro as instâncias relevantes da classe negativa.

(a) Base de dados Circle-200.



(b) Base de dados XOR-400.



Fonte: Própria produção.

8 ANÁLISE DE SEPARABILIDADE

Detectar em tempo finito se um problema de classificação binária é linearmente separável, seja no espaço de entrada ou no espaço de características, é um importante problema de Aprendizado de Máquinas. Este capítulo apresenta dois métodos para solucionar este problema. Estas métodos utilizam a base teórica introduzida nos dois capítulos anteriores. A primeira solução, denominada de trivial, aproveita-se da definição de pontos relevantes para executar de modo similar a uma estratégia de força bruta. A segunda solução, chamada de Classificador de Espaço Expandido, opera em um espaço de dimensão maior onde é possível resolver o problema de modo mais eficiente. As próximas seções descrevem, em detalhes, o funcionamento das duas soluções para as formulações primal e dual.

8.1 SOLUÇÃO TRIVIAL

A solução trivial é baseada em uma constatação simples: caso exista um hiperplano viável $H = (W, b)$, logo também existirá um hiperplano viável dado por $H_p = (W, b - D_p)$, em que D_p é definido por:

$$D_p = -1 \cdot (W \cdot X_p + b) \tag{8.1}$$

onde X_p é a instância mais próxima do hiperplano H. Ou seja, para todo hiperplano viável H, também existe ao menos um hiperplano viável H_p que contém a instância mais próxima de H. Como o método ADePE, descrito no Capítulo 7, é capaz de encontrar um hiperplano viável que contenha uma dada instância, caso exista tal hiperplano, pode-se executar este procedimento para todas as instâncias do conjunto de treinamento. Caso existam hiperplanos factíveis, ao menos um deles será encontrado ao executar o método em uma das instâncias relevantes do problema. Caso nenhuma instância seja rotulada como relevante, isso significa que não existem hiperplanos factíveis, e por consequência, o problema não é separável.

Esta solução pode se usada tanto na formulação primal quanto dual. Seu custo está relacionado, no pior caso, à quantidade total de instâncias, o que ocorre quando o problema é não linearmente separável. Neste caso, é necessário testar a relevância de todas as instâncias do problema. No melhor caso, se a primeira instância testada foi rotulada como relevante, resultando em um hiperplano factível, prova-se, testando somente uma instância, que o conjunto é linearmente separável.

Como esta solução testa iterativamente todas as instâncias até encontrar alguma que seja relevante, este método de busca lembra o funcionamento de métodos chamados *força bruta*. Porém, apesar dos experimentos mostrarem que esta solução demanda pouco

tempo de processamento para ser executada, durante o processo de pesquisa foi considerado a possibilidade de um método mais eficiente, ou seja, uma solução que não precise testar todas as instâncias. A próxima seção detalha o funcionamento de um método capaz de operar deste modo.

8.2 CLASSIFICADOR DE ESPAÇO EXPANDIDO (CEE)

O objetivo da criação do Classificador de Espaço Expandido (CEE) foi otimizar o método da força bruta, para que, ao invés de executar o ADePE em todos as instâncias, fosse possível executar em apenas um ponto. Isto é feito ao transpor o problema para um espaço expandido, de uma dimensão adicional, onde é possível escolher um ponto pivô comum que seja válido para analisar todas as instâncias na mesma execução. Para trabalhar neste espaço são gerados dois conjuntos. A partir do conjunto de instâncias $\mathcal{X}$ existentes no espaço de entrada E, é gerado um primeiro conjunto de instâncias $\mathcal{X}E \in EE$ denominado *Instâncias Expandidas*. A partir do conjunto $\mathcal{H}$ formado por todos os hiperplanos possíveis em E, é gerado um segundo conjunto de hiperplanos $\mathcal{H}E$ existentes em EE denominado *Hiperplanos Expandidos*. Estes conjuntos expandidos devem possuir com os conjuntos do espaço de entrada a relação de equivalência definida a seguir:

Definição 8.2.1. *Dado dois conjuntos de instâncias $\mathcal{X}$ e $\mathcal{X}E$, e dois conjuntos de hiperplanos $\mathcal{H}$ e $\mathcal{H}E$, a dupla $\{\mathcal{H}E, \mathcal{X}E\}$ é dita equivalente da dupla $\{\mathcal{H}, \mathcal{X}\}$ caso para todo hiperplano $H_i \in \mathcal{H}$ exista um hiperplano $HE_i \in \mathcal{H}E$ tal que para todo produto interno $\langle H_i, X_j \rangle, \forall X_j \in \mathcal{X}$, o produto interno $\langle HE_i, XE_j \rangle$, em que $XE_j \in \mathcal{X}E$, tenha mesmo valor.*

A condição de equivalência garante que, para qualquer hiperplano do conjunto $\mathcal{H}E$, existe um hiperplano do espaço de entrada que possui o mesmo valor de margem. Isto garante que caso um hiperplano HE_i seja viável no espaço expandido, então existe um hiperplano H_i viável no espaço de entrada. Esta condição também garante que caso nenhum hiperplano do conjunto $\mathcal{H}E$ seja viável no espaço expandido, então não existe um hiperplano viável no espaço de entrada. Assim, esta condição permite que a avaliação da margem no espaço expandido seja a mesma que no espaço de entrada, e assim possibilitando trabalhar no espaço expandido e aproveitar sua solução no problema original.

Além desta relação de equivalência, é necessário uma propriedade adicional no conjunto HE:

- Todo hiperplano HE_i do Conjunto de Hiperplanos Expandidos $\mathcal{H}E$ deve conter um mesmo ponto PE_v do Espaço Expandido EE.

Esta propriedade garante que caso o ponto PE_v seja usado como pivô pelo ADePE, então todos os hiperplanos do conjunto $\mathcal{H}E$ serão analisados. Assim, caso PE_v seja um

ponto extremo do conjunto de Instâncias Expandidas $\mathcal{X}E$, então o ADePE retornará um Hiperplano Expandido viável. Caso PE_v não seja extremo, então provará que não existe um Hiperplano Expandido viável.

Assim, ao gerar estes conjuntos expandidos e executar o ADePE usando como pivô um ponto Pe_v que é contido por todos os Hiperplanos Expandidos, é possível resolver o problema de separabilidade em apenas uma execução do ADePE. O funcionamento deste método é simples, necessitando de três etapas:

- Na primeira etapa, a partir de um conjunto de instâncias $\mathcal{X}$ existentes num espaço E, é gerado um conjunto de instâncias expandidas $\mathcal{X}E$ pertencente a um espaço EE de dimensão mais alta que E. Os conjuntos $\mathcal{X}$ e $\mathcal{X}E$ devem obedecer a condição de equivalência dada pela Definição 8.2.1.
- A segunda etapa envolve executar o ADePE no conjunto $\mathcal{X}E$ utilizando um ponto específico PE_v do espaço EE como pivô.
- A terceira etapa analisa o retorno do ADePE. Caso ele retorne um hiperplano, o mesmo é usado para gerar um hiperplano que seja viável no espaço de entrada. Caso o ADePE não retorne solução, isto significa que não existem soluções viáveis no problema de entrada.

As próximas sub seções descrevem este processo para a formulação primal e também para a formulação dual, e apresentam experimentos validando a funcionalidade e analisando o desempenho computacional. O Diagrama 25 exemplifica o funcionamento do algoritmo.

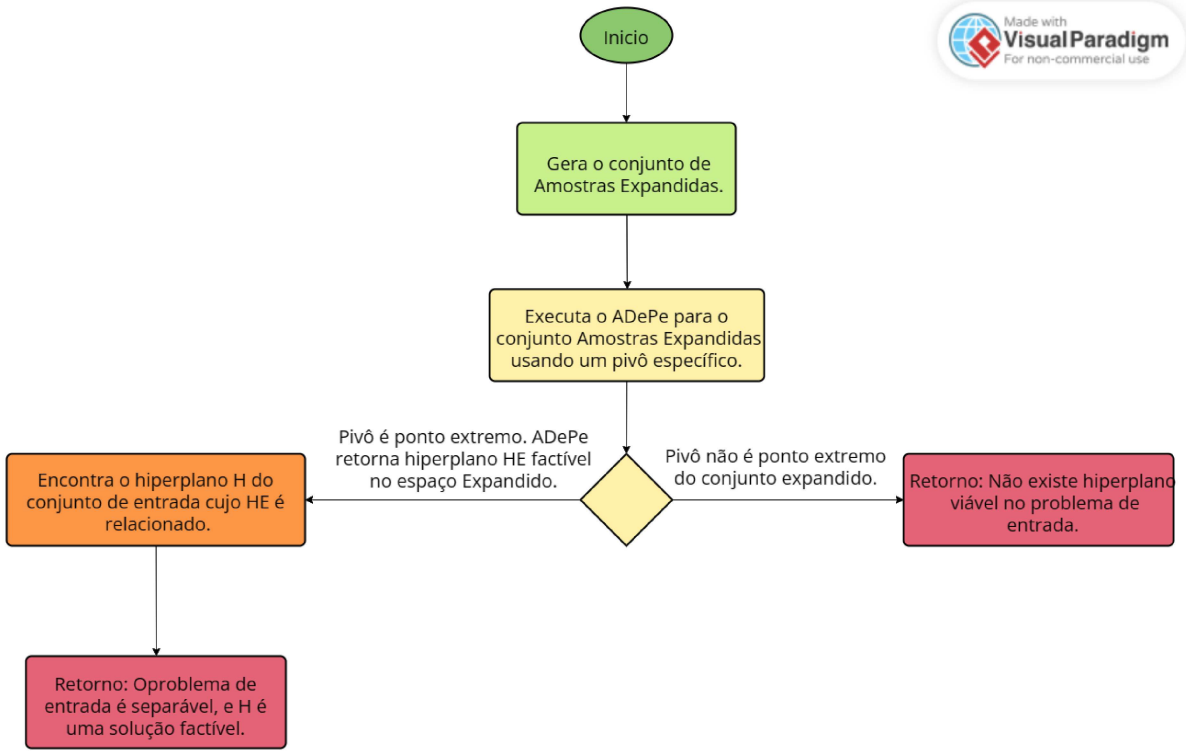
8.2.1 Espaço Expandido Primal

A geração das equivalências e do espaço expandido para a formulação primal envolve uma mudança simples, mas que possibilita as duas condições serem atendidas. As regras para geração do espaço expandido e dos conjuntos são listadas no teorema a seguir:

Teorema 8.2.1. *Dado uma instância Pi e um hiperplano W existentes em um espaço de entrada E, eles possuem relação de equivalência, na representação primal, respectivamente com uma instância PE_i e um hiperplano WE existentes em um espaço expandido EE, gerados a partir das regras listadas a seguir:*

- *Dado um espaço de entrada E de dimensionalidade d, o seu Espaço Expandido EE possui dimensionalidade $d + 1$.*
- *Dado um hiperplano $W = \{[w_1, w_2, .., w_d], b\}$ existente no espaço de entrada, o seu hiperplano equivalente no Espaço Expandido é dado por $WE = \{[w_1, w_2, .., w_d, b], 0\}$. Ou seja: o hiperplano expandido possui seus d primeiros pesos iguais aos d primeiros*

Figura 25 - Diagrama do Classificador de Espaço Expandido. Em verde claro a primeira etapa. Em Amarelo e laranja segunda etapa. Em vermelho a terceira etapa.



Fonte: Própria produção.

pesos de seu hiperplano relacionado no espaço original; o valor do último peso equivale ao valor do viés do hiperplano relacionado do espaço original; o valor do viés do hiperplano expandido é igual a 0.

- *Dado uma instância do espaço original* $P_i = [p_1, p_2, .., p_d]$*, a sua instância relacionada no Espaço Expandido é definida por* $PE_i = [p_1, p_2, .., p_d, 1]$*. Ou seja: os d primeiros atributos da instância expandida são iguais aos d primeiros atributos de sua instância relacionada no espaço original; o último atributo é igual a* 1.

Demonstração. Dados uma instância $P_i = [p_1, p_2, .., p_d]$ existente num espaço de entrada E de dimensão d, sua instância equivalente $PE_i = [p_1, p_2, .., p_d, 1]$ existente num Espaço Expandido EE de dimensão $d+1$; um hiperplano $W = \{[w_1, w_2, .., w_d], b\}$, e seu hiperplano equivalente em EE dado por $WE = \{[w_1, w_2, .., w_d, b], 0\}$.

Para as regras de equivalência estarem corretas, elas deve atender as duas condições descritas na Sub-Seção 8.2. A primeira condição se refere a distância entre um hiperplano e instância ser igual a distância entre suas suas versões equivalentes. Ou seja, deseja-se que a Equação 8.2 seja verdadeira:

$$\langle W, P_i \rangle = \langle WE, PE_i \rangle \tag{8.2}$$

Expandindo os termos, segundo as Regras 2 e 3 do Teorema 8.2.1, tem-se:

$$\langle\{[w_1, w_2, .., w_d], b\}, [p_1, p_2, .., p_d]\rangle = \langle\{[w_1, w_2, .., w_d, b], 0\}, [p_1, p_2, .., p_d, 1]\rangle \tag{8.3}$$

Calculando os produtos internos obtém-se:

$$w_1 \cdot p_1 + w_2 \cdot p_2 + ... + w_d \cdot p_d + b = w_1 \cdot p_1 + w_2 \cdot p_2 + ... + w_d \cdot p_d + b \cdot 1 + 0 \tag{8.4}$$

Simplificando nos dois lados resulta na relação de igualdade:

$$w_1 \cdot p_1 + w_2 \cdot p_2 + ... + w_d \cdot p_d + b = w_1 \cdot p_1 + w_2 \cdot p_2 + ... + w_d \cdot p_d + b \tag{8.5}$$

Logo, a distância é a mesma e a primeira condição é satisfeita.

Para atender a segunda condição, é necessário que todo hiperplano expandido contenha um mesmo ponto do Espaço Expandido. Isto é garantido pela segunda regra de geração, que limita os hiperplanos Expandidos a terem sempre seu viés igual a 0. O que por consequência faz a origem sempre estar contida neles. Consequentemente, escolhendo a origem como pivô, constituí-se um novo ponto de interseção comum a todos os hiperplanos. Assim, ambas as condições são cumpridas pelas três regras de geração.

□

A partir destas regras é possível gerar um novo conjunto de amostras que mantém relação de equivalência com o conjunto original, e, a partir de um hiperplano Expandido, gerar um hiperplano equivalente a ele no espaço de entrada. Estas regras permitem a utilização da origem do Espaço Expandido como o pivô do ADePE.

Um exemplo numérico da aplicação destas regras de equivalência é apresentado a seguir: Considerando um conjunto de pontos $\mathcal{X}$ constituído por três pontos $X_1 = [2, 2]$, $X_2 = [3, 3]$ e $X_3 = [1, 1]$ e um hiperplano $W = \{[2, 1], 4\}$; ao aplicar as regras no conjunto $\mathcal{X}$ obtém-se o conjunto de pontos expandidos X_e constituído pelos seguintes de pontos expandidos $X_{1e} = [2, 2, 1]$, $X_{2e} = [3, 3, 1]$ e $X_{3e} = [1, 1, 1]$ respectivamente relacionados aos pontos X_1, X_2 e X_3; ao aplicar as regras no hiperplano W obtém-se seu hiperplano expandido relacionado $W_e = \{[2, 1, 4], 0\}$. Consequentemente, a distância entre os membros de $\mathcal{X}$ e W é dada a seguir: $D(W, X_1) = (2 \cdot 2 + 1 \cdot 2 + 4)$; $D(W, X_2) = (2 \cdot 3 + 1 \cdot 3 + 4)$; $D(W, X_3) = (2 \cdot 1 + 1 \cdot 1 + 4)$; estas distâncias são as mesmas entre o hiperplano W_e e os respectivos pontos relacionados de X_e como demonstrado a seguir: $D(W_e, X_{1e}) = (2 \cdot 2 + 1 \cdot 2 + 4 \cdot 1 + 0)$; $D(W_e, X_{2e}) = (2 \cdot 3 + 1 \cdot 3 + 4 \cdot 1 + 0)$; $D(W_e, X_{3e}) = (2 \cdot 1 + 1 \cdot 1 + 4 \cdot 1 + 0)$.

Tabela 17 – Tempo de execução do Classificador de Espaço Expandido primal e do Perceptron primal, em segundos.

Bases	CEE	Perceptron
Iris Binária	$1.00 \cdot 10^{-3}$	$9.50 \cdot 10^{-5} \pm 2.97 \cdot 10^{-4}$
testNS	$2.00 \cdot 10^{-3}$	$1.40 \cdot 10^{-5} \pm 1.17 \cdot 10^{-4}$
lp4	$4.00 \cdot 10^{-3}$	$1.06 \cdot 10^{-2} \pm 1.30 \cdot 10^{-3}$
Mushroom	$8.40 \cdot 10^{-2}$	$7.68 \cdot 10^{-2} \pm 1.49 \cdot 10^{-2}$
Toy	$2.00 \cdot 10^{-3}$	$9.40 \cdot 10^{-5} \pm 2.92 \cdot 10^{-4}$
Zoo	$1.00 \cdot 10^{-3}$	$2.30 \cdot 10^{-4} \pm 4.26 \cdot 10^{-4}$
Colon	$1.80 \cdot 10^{-2}$	$2.86 \cdot 10^{-2} \pm 1.81 \cdot 10^{-3}$
Divorce	$1.00 \cdot 10^{-3}$	$9.58 \cdot 10^{-4} \pm 4.61 \cdot 10^{-4}$

8.2.1.1 Experimentos

Para validação do método foram realizados experimentos analisando se ele é capaz de encontrar classificadores viáveis nas bases linearmente separáveis. O algoritmo conseguiu encontrar soluções para todas as bases linearmente separáveis utilizadas nos experimentos do capítulo anterior.

Também foi analisado o tempo de execução, comparando ao tempo despendido pelo algoritmo Perceptron versão primal. Foram realizados experimentos variando a taxa de aprendizado com os valores: 1.0, 0.1, 0.05 e 0.01; o melhor resultado foi escolhido para ser utilizado na comparação. Foram realizadas 1000 execuções variando aleatoriamente a normal do hiperplano inicial do perceptron. A Tabela 17 compara o tempo despendido pelo Classificador de Espaço Expandido com o tempo do Perceptron primal. É possível perceber que o método desenvolvido obteve resultados competitivos com o Perceptron em algumas bases.

8.2.2 Espaço Expandido Dual

A formulação dual do Classificador de Espaço Expandido (CEE) utiliza regras de equivalência que, assim como as regras de equivalência da formulação primal, fazem com que todos os hiperplanos expandidos passem pela origem ao forçar o valor do viés ser igual a zero. Foram desenvolvidas regras de equivalência tanto para a representação tradicional quanto para a representação por vetores de alphas. O Classificador de Espaço Expandido Dual (CEED) utiliza a representação por vetores de alphas, porém para provar a corretude das regras desta representação é necessário primeiro compreender as regras para a representação dual tradicional. As regras para as duas formas de representação são descritas nas sub-seções seguintes.

Teorema 8.2.2. *Dados uma instância P_i e um hiperplano H existentes em um espaço de entrada E, eles possuem relação de equivalência, na representação dual, respectivamente com uma instância PE_i e um hiperplano HE existentes em um espaço de entrada expandido*

EE gerados a partir das regras listadas a seguir:

- *1) Seja um conjunto de instâncias no espaço de entrada P. O conjunto de instâncias expandidas PE relacionadas ao conjunto P é formado por todas as instâncias pertencentes a P adicionado uma instância PE_v que atenda as seguintes condições: $\langle PE_j, PE_v \rangle_K = 0, \forall (PE_j \neq PE_v) \in PE$ e $\langle PE_v, PE_v \rangle_K = 1$.*
- *2) Dado um espaço de alphas E de dimensionalidade n, o seu Espaço Expandido EE possui dimensionalidade $n+1$. O índice $n+1$ é referente à instância PE_v.*
- *3) Dado um vetor de alphas $H = \{[\alpha_1, \alpha_2, .., \alpha_n], b\}$ existente no espaço de alphas E, o seu vetor de alphas equivalente em EE é dado por $HE = \{[\alpha_1, \alpha_2, .., \alpha_n, b], 0\}$.*
- *4) A distância entre um hiperplano HE e uma instância PE_i é calculada da seguinte forma: $DE(HE, PE_i) = \sum_{j=1}^{n+1} (\alpha_j \cdot \langle PE_j, PE_i \rangle_K) + \sum_{j=1}^{n+1} (\alpha_j \cdot \langle PE_j, PE_v \rangle_K)$.*

Demonstração. Para a equivalência estar correta, ela deve atender as duas condições descritas na Sub-Seção 8.2. A primeira condição se refere a distância entre um hiperplano e instância ser igual a distância entre suas versões equivalentes.

Deseja-se provar a igualdade $DE(HE, PE_i) = D(H, P_i)$. Pela Regra 4 que define o cálculo de distância no espaço expandido tem-se:

$$DE(HE, PE_i) = \sum_{j=1}^{n+1} (\alpha_j \cdot \langle PE_j, PE_i \rangle_K) + \sum_{j=1}^{n+1} (\alpha_j \cdot \langle PE_j, PE_v \rangle_K) \tag{8.6}$$

Pela Regra 1 do Teorema 8.2.2, como $\langle PE_j, PE_v \rangle_K = 0, \forall (PE_j \neq PE_v) \in PE$, pode-se simplificar a equação diminuindo a quantidade de itens do primeiro somatório eliminando o ultimo elemento, e transformando o segundo somatório em apenas um termo:

$$DE(HE, PE_i) = \sum_{j=1}^{n} (\alpha_j \cdot \langle PE_j, PE_i \rangle_K) + \alpha_{n+1} \cdot \langle PE_v, PE_v \rangle_K \tag{8.7}$$

Como pela Regra 2 do Teorema 8.2.2 tem-se que $\alpha_{n+1} = b$ e pela Regra 1 tem-se que $\langle PE_v, PE_v \rangle_K = 1$:

$$DE(HE, PE_i) = \sum_{j=1}^{n} (\alpha_j \cdot \langle PE_j, PE_i \rangle_K) + b \cdot 1 \tag{8.8}$$

Portanto pode-se simplificar para:

$$DE(HE, PE_i) = \sum_{j=1}^{n} (\alpha_j \cdot \langle PE_j, PE_i \rangle_K) + b \tag{8.9}$$

a qual é igual a fórmula de distância da formulação dual apresentada anteriormente na Equação 7.12:

$$D(H, P_i) = \sum_{j=1}^{n} (\alpha_j \cdot \langle PE_j, PE_i \rangle_K) + b \tag{8.10}$$

Consequentemente pode-se afirmar que:

$$DE(HE, PE_i) = D(H, P_i) \tag{8.11}$$

Concluindo que a primeira condição está satisfeita.

A segunda condição a ser satisfeita diz que todos os hiperplanos equivalentes devem conter um mesmo ponto. Isto pode ser verificado de forma trivial devido a Regra 3 do Teorema 8.2.2, que impões todos os hiperplanos a possuírem o valor de seus viés igual a 0, consequentemente todos eles contém a origem do Espaço Expandido de Alphas EE em que são definidos.

□

Existem dois aspectos importantes a serem analisados a partir destas regras. O primeiro ponto importante é relacionado a escolha do ponto a ser analisado pelo ADePE dual, sendo este a origem do Espaço Expandido devido a ele atender a segunda condição.

O segundo ponto importante está relacionado à instância PE_v definida pelas regras de equivalência dual. Como esta instância não existe no espaço de entrada, ela não precisa ser avaliada durante a execução do ADePE Dual. Apenas as instâncias relacionadas as existentes no espaço de entrada precisam ser consideradas pelo ADePE ao analisar se a origem é um ponto extremo. Isto leva a outra constatação importante: como a instância PE_v não precisa ter mensurada sua distância ao classificador HE, e o valor do kernel aplicado a mesma já é definido a priori, não é necessário saber seus componentes nem seu rótulo. Deste modo, mesmo que seja impossível existir uma instância cujos valores de seu Kernel com as outras instâncias sejam iguais aos definidos pela Regra 1 do Teorema 8.2.2, isto não impede o funcionamento do algoritmo.

8.2.2.1 Representação por Vetores de Alphas

A representação por vetores de alphas introduzida na Sub-seção 7.3.1 é necessária para o Classificador de Espaço Expandido Dual devido ao ADePE dual ser executado em uma de suas etapas. Para utilizar esta representação é necessário gerar um segundo conjunto de regras de equivalência. As regras estão descritas no teorema a seguir:

Teorema 8.2.3. *Seja uma instância P_i e um hiperplano H definidos num espaço de entrada E, eles possuem relação de equivalência, na representação dual, respectivamente com uma instância PE_i e um hiperplano HE definidos num espaço de entrada expandido EE gerados a partir das regras listadas a seguir:*

- *1) Dado um conjunto de instâncias P do espaço de entrada, o conjunto de instâncias expandidas PE é formado por todas as instâncias pertencentes a P adicionado uma instância PE_v que atende as seguintes condições: $\langle PE_j, PE_v \rangle_K = 0, \forall (PE_j \neq PE_v) \in PE$ e $\langle PE_v, PE_v \rangle_K = 1$.*
- *2) Dado um espaço de alphas E de dimensionalidade n, o seu Espaço Expandido EE possui dimensionalidade $n+1$. O índice $n+1$ é referente à instância adicional PE_v.*
- *3) Dada uma instância P_i do espaço de entrada, a sua representação como vetor de alphas no espaço EE é definida por um vetor PE_i, cujos valores nos índices i e $n+1$ são iguais a 1, e os valores nos demais índices são iguais a 0.*
- *4) Dado um vetor de alphas $H = \{[\alpha_1, \alpha_2, .., \alpha_n], b\}$ existente no espaço de alphas E, o seu vetor de alphas equivalente no Espaço de Alphas Expandido é dado por $HE = \{[\alpha_1, \alpha_2, .., \alpha_n, b], 0\}$.*
- *5) A distância entre um hiperplano HE e uma amostra PE_i é calculada da seguinte forma: $DEA(HE, PE_i) = \sum_{j=1}^{n+1} \sum_{u=1}^{n+1} \alpha_j \cdot PE_i[u] \cdot \langle PE_j, PE_u \rangle_K$.*

Demonstração. Para a equivalência estar correta, ela deve atender as duas condições descritas na Sub-Seção 8.2. A primeira condição se refere ao calculo da distância entre um hiperplano e uma instância, o qual deve ser igual a distância da versão equivalente. Deseja-se provar que $DEA(HE, PE_i) = D(H, P_i)$.

Começando pela definição de $DEA(HE, PE_i)$ tem-se:

$$DEA(HE, PE_i) = \sum_{j=1}^{n+1} \sum_{u=1}^{n+1} \alpha_j \cdot PE_i[u] \cdot \langle PE_j, PE_u \rangle_K \tag{8.12}$$

Pela Regra 3 tem-se que os únicos índices não nulos de PE_i são o índice i e o índice $n+1$, logo pode-se simplificar a equação:

$$DEA(HE, PE_i) = \sum_{j=1}^{n+1} \alpha_j \cdot PE_i[i] \cdot \langle PE_j, PE_i \rangle_K + \sum_{j=1}^{n+1} \alpha_j \cdot PE_i[n+1] \cdot \langle PE_j, PE_{n+1} \rangle_K \tag{8.13}$$

Como pela Regra 3 tem-se que $PE_i[i] = 1$ e $PE_i[n+1] = 1$:

$$DEA(HE, PE_i) = \sum_{j=1}^{n+1} \alpha_j \cdot 1 \cdot \langle PE_j, PE_i \rangle_K + \sum_{j=1}^{n+1} \alpha_j \cdot 1 \cdot \langle PE_j, PE_{n+1} \rangle_K \tag{8.14}$$

Como o índice $n+1$ é referente à instância adicional PE_v, pode-se reescrever a equação como:

$$DEA(HE, PE_i) = \sum_{j=1}^{n+1} \alpha_j \cdot \langle PE_j, PE_i \rangle_K + \sum_{j=1}^{n+1} \alpha_j \cdot \langle PE_j, PE_v \rangle_K \tag{8.15}$$

Que é igual a fórmula de distância do espaço expandido dual apresentada na Regra 4 da sub-seção 8.2.2, logo:

$$DEA(HE, PE_i) = DE(HE, PE_i) \tag{8.16}$$

Como pelo Teorema 8.2.2 tem-se que $DE(HE, PE_i) = D(H, P_i)$, assim pode-se afirmar que:

$$DEA(HE, PE_i) = D(H, P_i) \tag{8.17}$$

A segunda condição a ser satisfeita diz que todos os hiperplanos equivalentes devem conter um mesmo ponto. Isto pode ser verificado de forma trivial devido a Regra 3 definir que todos os hiperplanos devem ter o valor do viés igual a 0. Logo todos eles contém a origem do Espaço Expandido de Alphas EE.

□

Esta forma de representação, como abordado na Sub-seção 7.3.1 possibilita a implementação do algoritmo ADePE na formulação dual. Assim, estas regras de equivalência permitem que o ADePE seja utilizado no Classificador de Espaço Expandido introduzido neste capítulo.

8.2.2.2 Experimentos

Para validar a formulação dual do método, foram realizados experimentos analisando se o mesmo conseguiria, usando o kernel linear, determinar corretamente a separabilidade no conjunto de bases linearmente separáveis utilizadas no capítulo anterior. Para este conjuntos de bases, o algoritmo foi capaz de determinar corretamente que todas são linearmente separáveis com o kernel linear.

Para analisar a viabilidade do método, foi analisado o tempo de execução, comparando ao tempo do algoritmo Perceptron Dual. A Tabela 18 contém os resultados destes experimentos para bases linearmente separáveis.

Também foi comparado a quantidade de suportes existentes no classificador gerado. Um hiperplano classificador com uma quantidade menor de suportes apresenta uma vantagem pois isto implica que menos tempo será despendido ao classificar uma nova instância. A Tabela 19 contém os resultados para as bases linearmente separáveis.

Para bases não linearmente separáveis também foram realizados experimentos comparando o tempo de execução e a quantidade de suportes nas soluções. A Tabela

Tabela 18 – Tempo de execução do classificador dual, com kernel linear, comparado ao tempo de um Perceptron dual com kernel linear.

Bases	CEED	Perceptron
Iris Binária	$1 \cdot 10^{-16}$	$1 \cdot 10^{-16}$
testNS	$1 \cdot 10^{-16}$	$1 \cdot 10^{-16}$
lp4	$1.6 \cdot 10^{-1}$	$1.5 \cdot 10^{-2}$
Mushroom	$1.9 \cdot 10^{-0}$	$1.94 \cdot 10^{-1}$
Toy	$1 \cdot 10^{-16}$	$1 \cdot 10^{-16}$
sc	$5.3 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$
Zoo	$1.6 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$
Colon	$5.30 \cdot 10^{-2}$	$1.30 \cdot 10^{-2}$
Divorce	$1.17 \cdot 10^{-1}$	$1.00 \cdot 10^{-3}$

Tabela 19 – Quantidade de suportes na solução do classificador comparado a quantidade de suportes da solução do Perceptron dual, ambos com kernel linear.

Bases	CEED	Perceptron
Iris Binária	2	2
testNS	3	3
lp4	35	63
Mushroom	39	66
Toy	6	9
Zoo	9	7
Colon	17	36
Divorce	19	19

20 mostra o tempo comparado com o Perceptron Dual. As bases Pima, Bupa, Hepatitis e Heart demonstram a vantagem do Classificador de Espaço Expandido terminar sua execução em tempo finito, pois o mesmo definiu estas bases como não separáveis por um kernel quadrático em tempo viável, diferentemente do Perceptron Dual que executou por uma grande quantidade de tempo. A Tabela 21 mostra a comparação da quantidade de suportes. Ela demonstra que os classificadores encontrados pelo método obtiveram aproximadamente metade da quantidade de suportes dos classificadores encontrados pelo Perceptron Dual.

Os tabelas mostram que o Classificador de Espaço Expandido em sua formulação dual consegue gerar classificadores viáveis com uma menor quantidade de suportes e requerendo tempo próximo ao do Perceptron dual.

Tabela 20 – Tempo de execução em segundos, do Classificador de Espaço Expandido Dual e do Perceptron dual, ambos usando kernel quadrático.

Bases	CEED	Perceptron Dual
Circle	$1 \cdot 10^{-16}$	$1 \cdot 10^{-16}$
XOR	$7.0 \cdot 10^{-3}$	$1 \cdot 10^{-16}$
Ionosphere	13.45	$2.20 \cdot 10^{-2}$
Pima	0.52	10364.55
Sonar	1.75	0.13
wdbc	1.79	181
Wine	0.19	1548.64
Bupa	0.03	1692.69
Hepatitis	1.10	1211.41
Heart	1.85	141.80
Fertility	$9.2 \cdot 10^{-2}$	$2 \cdot 10^{-3}$

Tabela 21 – Quantidade de suportes na solução do CEED comparado a quantidade de suportes da solução do Perceptron, ambos usando kernel quadrático.

Bases	CEED	Perceptron Dual
Circle	4	9
XOR	2	3
Ionosphere	54	107
Pima	—-	—-
Sonar	63	114
wdbc	62	—-
Wine	24	52
Bupa	—-	—-
Hepatitis	75	—-
Heart	93	—-
Fertility	13	25

9 CONCLUSÕES

Durante esta pesquisa foram realizados estudos relacionados ao problema de classificação binária. Seguiram-se duas linhas de abordagem: a primeira explorando a geometria do espaço de versões. Como resultados desenvolveram-se dois algoritmos de computação evolucionistas que utilizam a representação de coordenadas hiper-esféricas. A segunda, também inspirada na geometria do espaço de versões e no problema de envoltória convexa. Como resultado desenvolveram-se duas técnicas importantes que são executadas em tempo finito. Uma para análise de relevância de amostras e outra para análise de separabilidade. Estas técnicas foram desenvolvidas tanto na formulação primal quanto na dual, permitindo a sua extensão a análise dos problemas no espaço de característica. Na próxima sub-seção será apresentado um breve resumo das contribuições desenvolvidas neste trabalho. Depois, será listado os artigos publicados ao longo destes anos e discutido as futuras publicações. Por fim, será comentado trabalhos futuros que podem ser gerados partir do conhecimento produzido nesta tese.

9.1 RESUMO DO CONTEÚDO APRESENTADO

Nesta tese foi apresentado o conhecimento desenvolvido durante o processo de pesquisa do Doutorado. Primeiramente, foram estudados os conceitos teóricos relacionados ao problema de classificação binária, conceitos estes fundamentais para o desenvolvimento de novos algoritmos e para o entendimento profundo do assunto, como exemplo a definição do problema de classificação binária, de distância geométrica e funcional, de factibilidade e de poder de generalização.

Um dos objetivos mais importantes em classificação binária é a geração de classificadores mais eficientes, sendo essa eficiência relacionada a capacidade de acertar consistentemente o rótulo de instâncias desconhecidas, capacidade esta denominada de poder de generalização. Para o desenvolvimento de novos métodos foram estudadas métricas para estimar o poder de generalização de um classificador. Duas métricas foram avaliadas, sendo a primeira baseada no conceito de margem, definido como a distância do classificador até a instância mais próxima. A segunda métrica foi o centro analítico, calculado por meio de o somatório de funções potencias relacionadas a todas as instâncias do problema.

Também foi estudado o conceito de espaço de versões, definido como o conjunto de todos os classificadores viáveis. Esta interpretação do espaço de versões é essencial para o entendimento avançado do funcionamento dos algoritmos desenvolvidos, e das duas métricas utilizadas para encontrar classificadores mais eficientes.

Entre as contribuições geradas pela primeira parte da pesquisa, está a adoção de um sistema de coordenadas hiperesféricas para representação dos dados, e o desenvolvimento de dois métodos de computação evolucionista. A utilização deste sistema foi fundamental

para desenvolvimento destes métodos, sendo uma forma mais eficiente de representação vetorial no respectivo espaço de versões. O primeiro método desenvolvido, denominado de *Algoritmo Evolucionista de Larga Margem (AELM)*, está relacionado à obtenção de uma solução aproximada para o problema de maximização da margem. O segundo método, denominado de *Algoritmo Evolucionista de Centro Analítico (AECA)*, está relacionado à obtenção de uma solução aproximada para o centro de massa do espaço de versões utilizando o conceito de centro analítico. Ambos algoritmos foram publicados em periódicos e conferências relevantes da área de Aprendizado de Máquinas [57] [56]. Os resultados obtidos em termos de acurácia, foram equivalentes ou superiores aos métodos de referência utilizados como comparação.

Como contribuições relacionadas a segunda parte da pesquisa, pode-se destacar o desenvolvimento de um método eficiente, e de tempo finito, relacionado a análise da relevância de instâncias do conjunto de dados. Nesse sentido, foi mostrada a possibilidade de redução deste problema ao problema de determinação de pontos extremos. A partir desta redução foi possível o desenvolvimento da solução tanto em sua formulação primal quanto em sua formulação dual. Finalmente foi apresentado uma técnica eficiente para a analise de separabilidade de um problema do classificação binária permitindo consequentemente a escolha de uma função kernel menos complexa que seja capaz de garantir esta separabilidade linear no respectivo espaço de características.

9.2 ARTIGOS PUBLICADOS

Durante esse período de pesquisa foram publicados artigos em conferências e periódicos tanto nacionais quanto internacionais. Algumas das publicações foram relacionadas a primeira parte da tese, outras foram de aplicações de Aprendizado de Máquinas não relacionados aos temas da tese. Abaixo, segue uma lista com a descrição das publicações e seu respectivo Qualis:

Periódicos:

- Lélis, C.A.S.; Motta Goulart, R. **A Complete Diabetes Management and Care System.** Latifi, S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 738. Springer. 2018. **Qualis B5.**
- Lélis, C.A.S.; Motta Goulart, R. **A Diabetes Management Information System with Glucose Prediction.** Information. 2019. **Qualis: B2.**
- Motta Goulart, R; Leite, S.d.C.; Fonseca Neto, R. **A kernel based learning method for non-stationary two-player repeated games.** Knowledge-Based Systems. 2020. **Qualis: A1.**

- Motta Goulart, R.; Villela, S.M.; Borges, C.C.H.; Fonseca Neto, R. (2020). **An Evolutionary Algorithm for Large Margin Classification.** Soft Computing. 25(3). Pages 7593–7607. **Qualis: A2.**

  Conferências:

- Motta Goulart, R.; Villela, S.M.; Borges, C.C.H.; Fonseca Neto, R. (2020). **An Evolutionary Analytic Center Classifier.** 9th Brazilian Conference on Intelligent Systems. **Qualis: B2.**

9.3 TRABALHOS FUTUROS

Como trabalhos futuros é planejado publicar ao menos dois artigos relacionados a segunda parte da tese. Um primeiro artigo sobre a redução do problema de classificação ao de detecção de pontos extremos, contendo os algoritmos criados para ambos e também a representação por vetores de alphas usada na formulação dual. Em seguida será publicado um segundo artigo sobre a analise de separabilidade, explicando o método de classificação desenvolvido, e estendendo também a escolha de funções kernel menos complexas.

O conhecimento gerado nesta tese abre muitas possibilidades para novas pesquisas. Dentre elas está em desenvolvimento uma modificação do classificador apresentado no Capítulo 8 para se tornar um maximizador de margem. Outro trabalho futuro seria a investigar se a representação por vetores de alphas conseguiria fazer os algoritmos evolucionistas obterem resultados satisfatórios na formulação dual.

9.4 CONSIDERAÇÕES FINAIS

A pesquisa realizada para o desenvolvimento desta tese de Doutorado possibilitou o desenvolvimento de métodos eficientes para o problema de classificação binária. Esta produção resultou em artigos publicados em periódicos e conferências. No futuro próximo, planeja-se publicar o conteúdo da segunda parte desta tese em mais artigos.

Mais desenvolvimentos relacionados ao conteúdo desta tese estão sendo realizados, que podem futuramente render mais publicações. Além disto, espera-se disponibilizar *online* o código dos métodos implementados para que isto ajude outros pesquisadores em futuras pesquisas.

Espera-se também que o conhecimento desenvolvido nesta tese, em especial o relacionado a segunda parte, possa ser aplicado futuramente para o desenvolvimento de novos métodos de classificação e ajudem a avançar a área de Aprendizado de Máquinas.

REFERÊNCIAS

[1] ADASCH, NORBERT; ERNST, B.; KEIM, D. **Topological Vector Spaces: The Theory Without Convexity Conditions**. Lecture Notes in Mathematics, Berlin New York: Springer-Verlag, v. 639, 1978. ISBN 978-3-540-08662-8. OCLC 297140003.

[2] AGMON S. **The relaxation method for linear inequalities**. Canadian Journal of Mathematics, v. 6, p. 382-392, 1954. DOI 10.4153/CJM-1954-037-2.

[3] ALON, U.; BARKAI, N.; NOTTERMAN, D.A.; GISH, K.; YBARRA, S.; MACK, D.; LEVINE, A.J. **Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays.** Proceedings of the National Academy of Sciences of the United States of America, v. 96, n. 12, p. 6745-6750, 1999.

[4] BALESTRIERO, R.; WANG, Z.; BARANIUK, R.G. **DeepHull: Fast Convex Hull Approximation in High Dimensions**. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), p. 3888–3892, 2022. DOI: 10.1109/ICASSP43922.2022.9746031.

[5] BENNETT, K.P.; BREDENSTEINER, E.J. **Duality and Geometry in SVM Classifiers.** Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 57–64. ISBN 1558607072.

[6] BISHOP, C.M. **Pattern Recognition and Machine Learning**. Springer, p. 183-186, 2006.

[7] BOLDRINI J.L. **Algebra Linear**. Harbra, 1986.

[8] BORGES, C; FONSECA NETO, R.; and HACEN, S. **A Regularized Evolutionary Algorithm for Constrained Optimization.** In. XXXVI Ibero-Latin American Congress on Computational Methods in Engineering, 2015. DOI 10.20906/CPS/CILAMCE2015-0514.

[9] BONYADI, Mr; REUTENS, D. **Optimal-margin evolutionary classifier.** IEEE Transactions on Evolutionary Computation, 2018.

[10] BOUSSAID, I.; LEPAGNOT, J.; SIARRY, P. **A survey on optimization metaheuristics.** Inf Sci 237:82–117, DOI 10.1016/j.ins.2013.02.041, 2013. `https://doi.org/10.1016/j.ins.2013.02.041`.

[11] BRADFORD B.C.; DOBKIN, D.P.; Huhdanpaa, Hannu (1996). **The quickhull algorithm for convex hulls**. ACM Trans. Math. Softw. v. 22, n. 4, p. 469–483, 1996. `https://doi.org/10.1145/235815.235821`.

[12] CORTES, C.; VAPNIK, V. **Support-vector networks**. Mach Learn v. 20, n. 3, p. 273–297, 1995. DOI 10.1023/A:1022627411411, `https://doi.org/10.1023/A:1022627411411`.

[13] CHENEY, W.; KINCAID, D. **Linear Algebra: Theory and Applications.** Sudbury, Ma: Jones and Bartlett, p. 544-558, 2009. ISBN 978-0-7637-5020-6.

[14] CRISTIANINI, N.; SHAWE-TAYLOR, J. **Kernel Methods for Pattern Analysis**, Cambridge University Press, 1st edn, 2004.

[15] de ALMEIDA, M. B.; BRAGA A.d.P.; BRAGA J.P. **SVM-KM: speeding SVMs learning with a priori cluster selection and k-means.** Proceedings Sixth Brazilian Symposium on Neural Networks, Rio de Janeiro, Brazil, v. 1, pp. 162-167, 2000. DOI: 10.1109/SBRN.2000.889732.

[16] DEB, K., KALYANMOY, D. **Multi-Objective Optimization Using Evolutionary Algorithms.** John Wiley and Sons, Inc., USA, 2001.

[17] DIAS, M.; ROCHA NETO, A. **Evolutionary support vector machines: A dual approach.** Congress on Evolutionary Computation, 2016. DOI 10.1109/CEC.2016.7744058.

[18] DIETTERECH, T.G. **Ensemble methods in machine learning.** Multiple classifier systems, p. 1-15, 2000.

[19] DUA, D.; GRAFF, C. **UCI machine learning repository.**, 2017. `http://archive.ics.uci.edu/ml`.

[20] EDDY, W.F. **Convex hull peeling**. Physica-Verlag HD, COMPSTAT 1982 5th Symposium held at Toulouse 1982, Toulouse, Heidelberg, p. 42–47, 1952.

[21] EIBEN, A., SCHIPPERS, C. **On evolutionary exploration and exploitation.** Fundam Inform, v.35, p. 35–50, 1998.

[22] ENES, K.B.; VILLELA, S.M.; FONSECA NETO, R. **Version Space Reduction Based on Ensembles of Dissimilar Balanced Perceptrons.** Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, p. 1448-1454, 2016.

[23] FREUND, Y.; SCHAPIRE, R.E. **A desicion-theoretic generalization of on-line learning and an application to boosting.** Computational learning theory, 1995.

[24] GOBERNA, M.A.; LÓPEZ, M.A. **A Theory os Linear Inequality Systems.** Linear Algebra and its Applications, Elsevier Science Publishing Co., vol. 106, p. 77-115, 1988.

[25] GOLDBERG, D. (1988) **Genetic Algorithms in Search Optimization and Machine Learning.** Addison-Wesley Publishing Companing, Inc Herbrich R., Graepel T., Campbell C., Bayes point machines. J Mach Learn Res v. 1, p. 245–279, 2001. DOI 10.1162/153244301753683717, `https://doi.org/10.1162/153244301753683717`.

[26] GOLUB, T.R.; SLONIM, D.K.; TAMAYO, P.; HUARD, C.; GAASENBEEK, M.; MESIROV, J.P.; COLLER, H.; LOH, M.L.; DOWNING, J.R.; CALIGIURI, M.A.; BLOOMFIELD, C.D; LANDER, E.S. **Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.** Science, 1999.

[27] GRAHAM, R.L. (1972). **An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set.** Inf. Process. Lett. v. 1, p. 132–133. `https://doi.org/10.1016/0020-0190(72)90045-2`.

[28] GREENBERG, H.J. **Consistency, redundancy, and implied equalities in linear systems.** Annals of Mathematics and Artificial Intelligence, v. 17, p. 37-83, 1996.

[29] HARTMANN, A.K.; MAJUMDAR, S.N.; SCHAWE, H.; SCHEHR, G. (2020). **The convex hull of the run-and-tumble particle in a plane.** J. Statistical Mechanics Theory and Experiment, v. 5, 2020.

[30] HERBRICH, R. **Learning Kernel Classifiers: Theory and Algorithms.** MIT Press, 2001.

[31] HERBRICH, R., Graepel T., Campbell C. (2001) **Bayes point machines.** J Mach Learn Res, v. 1, p. 245–279. DOI 10.1162/153244301753683717, https://doi.org/10.1162/153244301753683717.

[32] HOUSEHOLDER, A.S. **Unitary Triangularization of a Nonsymmetric Matrix.** J. ACM, v. 5, n. 4, p. 339–342, October, 1958. https://doi.org/10.1145/320941.320947.

[33] HUA, B.; BALDRICK, R. **A convex primal formulation for convex hull pricing.** IEEE Trans. Power Syst., v. 32, n. 5, p. 3814–3823, 2016.

[34] JARVIS, R.A. (1973). **On the identification of the convex hull of a finite set of points in the plane.** Information Processing Letters, v. 2, n. 1, p. 18–21. https://doi.org/10.1016/0020-0190(73)90020-3.

[35] JOACHIMS, T. **Making Large-Scale Support Vector Machine Learning Practical.** MIT Press, Cambridge, MA, USA, p. 169–184, 1999.

[36] JOSWIG, M. **Beneath-and-Beyond revisited.** Algebra, Geometry and Software Systems, 2002.

[37] KAPP, M.N., SABOURIN, R., MAUPIN, P. **A dynamic model selection strategy for support vector machine classifiers.** Applied Soft Computing, v. 12, n. 8, p. 2550–2565, 2012. DOI https://doi.org/10.1016/j.asoc.2012.04.001, URL http://www.sciencedirect.com/science/article/pii/S1568494612001615.

[38] KUNCHEVA, L.I. **Combining pattern classifiers: methods and algorithms.**, 2004.

[39] LACHAND-ROBERT, T.; OUDET, E. **Minimizing within convex bodies using a convex hull method.** SIAM J. on Optim., v. 16, n. 2, p. 368–379, 2005.

[40] LÉLIS, C.A.S.; MOTTA GOULART, R. **A Complete Diabetes Management and Care System.** Latifi, S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, Springer, v. 738, p. 651–658, 2018.

[41] LÉLIS, C.A.S.; MOTTA GOULART, R. **A Diabetes Management Information System with Glucose Prediction.** Information, v. 9, n.12, p. 319, 2019.

[42] KREIN, M.; MILMAN, D. **On extreme points of regular convex sets.**. Studia Mathematica, v. 9, p. 133-138, 1940.

[43] LASSERRE, J.B. **An analytical expression and an algorithm for the volume of a convex polyhedron in R^n.** Journal of Optimization Theory and Applications, v. 39, p. 363-377, 1983.

[44] LEITE S.d.C., FONSECA NETO, R. (2008) **Incremental margin algorithm for large margin classifiers.** Neurocomputing, v. 71, p. 1550–1560. DOI 10.1016/j.neucom.2007.05.002.

[45] LIAO P., ZHANG X., LI, K., FU, Y., WANG, M., WANG, S. (2016) **Parameter optimization for support vector machine based on nested genetic algorithms.** Journal of Automation and Control Engineering, v. 4, n. 1, p. 78–83.

[46] LIU, Y., ZHANG, Z., LUO, Y., WU, X. **Pso based on cartesian coordinate system.** Intelligent Computing in Bioinformatics. International Conference on Intelligent Computing(ICIC) 2014. Springer, Cham, Lecture Notes in Computer Science, v. 8590, p. 363–370, 2014.

[47] LOVISOLO, L., DA SILVA, E. **Uniform distribution of points on a hyper-sphere with applications to vector bit-plane encoding.** IEE Proceedings - Vision Image and Signal Processing, p. 187–193, 2001. DOI 10.1049/ip-vis:20010361.

[48] McCULLOCH, W.S., PITTS, W. **A Logical Calculus of the Ideas Immanent in Nervous Activity.** MIT Press, Cambridge, MA, USA, p 15–27, 1988.

[49] MERCER, J. *Functions of positive and negative type, and their connection with the theory of integral equations.* 587 Philosophical Transactions of the Royal Society, London A 209. p. 415–446, 1909.

[50] MICHALSKI, R.S., CARBONELL, J.G., MITCHELL, T.M. **Machine Learning: An Artificial Intelligence Approach.** Springer Publishing Company, Incorporated, 2013.

[51] MINSHU, M. **Genetic algorithms designed for solving support vector classifier.** International Symposium on Data, Privacy, and E-Commerce, p. 167–169, 2007.

[52] MISSOUM, S.; RAMU, P.; HAFTKA, R.T. **A convex hull approach for the reliability based design optimization of nonlinear transient dynamic problems.** Comput. Methods Appl. Mechanics Eng., v. 196, n. 29-30, p. 2895–2906, 2007.

[53] MITCHELL, M. **An Introduction to Genetic Algorithms.** MIT Press, Cambridge, MA, USA, 1998.

[54] MOON, P.H., SPENCER, D.E. **Field theory handbook : including coordinate systems, differential equations, and their solutions.** Springer-Verlag, Berlin; New York, 1988.

[55] MOTTA GOULART, R.; LEITE, S.d.C.; FONSECA NETO, R. **A kernel based learning method for non-stationary two-player repeated games.** Knowledge-Based Systems, v. 196, 2020.

[56] MOTTA GOULART, R.; VILLELA, S.M.; BORGES, C.C.H.; FONSECA NETO, R. **An Evolutionary Algorithm for Large Margin Classification.** Soft Computing, v. 25, n. 3, p. 7593–7607, 2021.

[57] MOTTA GOULART, R.; VILLELA, S.M.; BORGES, C.C.H.; FONSECA NETO, R. (2020). **An Evolutionary Analytic Center Classifier**. 9th Brazilian Conference on Intelligent Systems (BARCIS), 2020. https://doi.org/10.1007/978-3-030-61377-8_3.

[58] NARICI, L.; BECKENSTEIN, E. **Topological Vector Spaces. Pure and applied mathematics (Second ed.)**. Boca Raton, FL: CRC Press, 2011. ISBN 978-1584888666. OCLC 144216834.

[59] NOVIKOFF, A. **On convergence proofs on Perceptrons.** Proceedings of the Symposium on the Mathematical Theory of Automata, v. 12, p.615–622, 1962.

[60] PAQUET, U., ENGELBRECHT, A. **Training support vector machines with particle swarms.** In: Proceedings of the International Joint Conference on Neural Networks, v. 2, p. 1593–1598, 2003. DOI 10.1109/IJCNN.2003.1223937.

[61] RITUPARNA, D., KALYANMOY, D. (eds) **Evolutionary Constrained Optimization.** Springer India, 2015.

[62] ROSENBLATT **The Perceptron, a perceiving and recognizing automaton.** Project Para. Cornell Aeronautical Laboratory, 1957. URL referencematerial/rosenblatt1957Perceptron.pdf.

[63] RUJÁN, P.; MARCHAND, M. **Computing the Bayes Kernel Classifier.** The MIT Press, Setembro, 2000. ISBN 9780262283977, DOI 10.7551/mitpress/1113.003.0025.

[64] ROCHA H.P.; COSTA M. A.; BRAGA A.P. **Neural Networks Multiobjective Learning With Spherical Representation of Weights.** IEEE Transactions on Neural Networks and Learning Systems, v. 31, n. 11, p. 4761-4775, 2020. DOI: 10.1109/TNNLS.2019.2957730.

[65] RUJÁN, P. **Playing Billiards in Version Space.** Neural Computation, v. 9, n. 1, p. 99-122, 1997. ISSN 0899-7667, DOI 10.1162/neco.1997.9.1.99.

[66] SINGH, D.; FEBBO, P.G.; ROSS, K.; JACKSON, D.G.; MANOLA, J.; LADD, C.; TAMAYO, P.; RENSHAW, A.A.; D'AMICO, A.V.; RICHIE, J.P. **Gene expression correlates of clinical prostate cancer behavior.** Cancer Cell, v. 1, n. 2, p. 203-209, 2002.

[67] SONNEVEND, G. **An Analytical Center for Polyhedrons and then New Classes of Global Algorithms for Linear (Smooth, Convex) Programming.** Lectures Notes in Control Information Sciences, Springer-Verlag, New-York, p. 866–876, 1985.

[68] TRAFALIS, T.; MALYSCHEFF, A. **An Analytic Center Machine.** Machine Learning, v. 46. p. 203-223, 2002.

[69] TUMER, K.; GHOSH, J. **Analysis of decision boundaries in linearly combined neural classifiers.** Pattern Recognition, Elsevier, 1996.

[70] VILLELA, S.M.; LEITE. S.d.C.; FONSECA NETO, R. **Incremental p-margin algorithm for classification with arbitrary norm.** Pattern Recogn, v. 55, p. 261–272, 2016. DOI10.1016/j.patcog.2016.01.016, https://doi.org/10.1016/j.patcog.2016.01.016.

[71] WATKIN, T.L.H. **Optimal learning with a neural network.** EPL (Europhysics Letters), v. 21, p. 871, Março, 1993. DOI 10.1209/0295-5075/21/8/013.

[72] WILDERJANS, T.F.; CEULEMANS, E.; MEERS, K. **Chull: A generic convex-hull-based model selection method.** Behav. Res. Methods, v. 45, n. 1, p. 1–15, 2013.

[73] ZHU, Z.; ONG, Y.S.; DASH, M. **Markov blanket-embedded genetic algorithm for gene selection.** Pattern Recognition, 2007.

[74] BUPA Medical Research Ltd. **Bupa dataset.** https://archive.ics.uci.edu/ml/datasets/Liver+Disorders

[75] **Breast Cancer dataset.** ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin.

[76] YÕNTEM, M.; ADEM, K.; ILHAN, T. and KILIÇARSLAN, S. **Divorce Prediction Using Correlation Based Feature Selection and Artificial Neural Networks.** https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set.

[77] **Statlog (Heart) [Dataset]**. UCI Machine Learning Repository. https://doi.org/10.24432/C57303.

[78] **Hepatitis [Dataset]**. UCI Machine Learning Repository. 1983. https://doi.org/10.24432/C5Q59J.

[79] **Credit Screening dataset.** ftp://ftp.ics.uci.edu/pub/machine-learning-databases/credit-screening.

[80] AHA D.W. **Tic-Tac-Toe dataset** https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame.

[81] SCHLIMMER, J. **Congressional Voting Records Data Set.** https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records.

[82] THRUN, S. **Monk-2 dataset.** https://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems.

[83] Unknown. **Hepatitis Data Set.** https://archive.ics.uci.edu/ml/datasets/Hepatitis.

[84] Unknown **Statlog (Heart).** http://archive.ics.uci.edu/ml/datasets/statlog+(heart).

[85] SIGILLITO, V. **Ionosphere dataset.** https://archive.ics.uci.edu/ml/datasets/ionosphere.

[86] SMITH, J.W.; EVERHART, J.E.; DICKSON, W.C.; KNOWLER, W.C.; JOHANNES, R.S. **Using the ADAP learning algorithm to forecast the onset of diabetes mellitus.** Proceedings of the Symposium on Computer Applications and Medical Care, IEEE Computer Society Press, p. 261-265, 1988.

[87] SEJNOWSKI, T.; GORMAN, R. **Connectionist Bench (Sonar, Mines vs. Rocks) [Dataset].** UCI Machine Learning Repository, 1988. https://doi.org/10.24432/C5T01Q.

[88] WOLBERG, W.; MANGASARIAN, O.; STREET, N.; STREET, W. **Breast Cancer Wisconsin (Diagnostic) [Dataset].** UCI Machine Learning Repository, 1993. https://doi.org/10.24432/C5DW2B.

[89] AEBERHARD, S.; FORINA, M. **Wine [Dataset]**. UCI Machine Learning Repository, 1992. https://doi.org/10.24432/C5PC7J.

[90] GIL, D.; GIRELA, J. **Fertility [Dataset].** UCI Machine Learning Repository, 2012. https://doi.org/10.24432/C5Z01Z.

[91] FISHER, R. **Iris [Dataset].** UCI Machine Learning Repository, 1936. https://doi.org/10.24432/C56C76.

[92] LOPES, L.; CAMARINHA-MATOS, L. **Robot Execution Failures [Dataset].** UCI Machine Learning Repository, 1998. https://doi.org/10.24432/C5M89N.

[93] FORSYTH, R. **Zoo [Dataset].** UCI Machine Learning Repository, 1990. https://doi.org/10.24432/C5R59V.

[94] **Divorce Predictors data set [Dataset].** UCI Machine Learning Repository, 2019. https://doi.org/10.24432/C53W5P.