Algoritmos Evolutivos aplicados ao Classificador baseado em Segmentos de Reta

Rosario Alejandra Medina Rodríguez

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação

Orientador: Prof. Dr. Ronaldo Fumio Hashimoto

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da FAPESP

São Paulo, julho de 2012

Algoritmos Evolutivos aplicados ao Classificador baseado em Segmentos de Reta

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original no trabalho, realizada em 03/07/2012. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Ronaldo Fumio Hashimoto (orientador) - IME-USP
- Prof. Dr. Roberto Hirata Jr. - IME-USP
- Prof. Dr. Anna Helena Reali Costa - EP-USP

Agradecimentos

Dedico meu maior agradecimento aos meus pais Leonor e Liborio, pelo amor, compreensão e apoio incondicional ao longo destes anos. A minha querida irmã Carola pela paciência, carinho e pelo apoio que encontrei nos momentos difíceis. Muito obrigada aos três por estar ao meu lado sempre apesar da distância e por ser minha inspiração e força.

Gostaria de expressar meus sinceros agradecimentos ao meu orientador Ronaldo Fumio Hashimoto, pela orientação deste trabalho, pelo incentivo e dedicação oferecida, e pela confiança em mim depositada. Sou muito grata aos professores Anna Helena Reali Costa e Roberto Hirata Jr. pelas contribuições e correções recebidas no exame de qualificação que ajudaram a melhorar este trabalho, e pela sua participação desta banca examinadora.

Agradeço a meus amigos que contribuíram de alguma forma na realização deste trabalho, aos colegas e professores do grupo de Visão Computacional do IME pelo enorme aprendizado. Muito obrigada ao João Henrique Burckas Ribeiro e Wellington Pinheiro dos Santos pela ajuda oferecida e ao Dan Pelleg e Andrew Moore pela disponibilização do código do algoritmo utilizado nesta dissertação.

Gostaria de agradecer à Maysa Macedo pelas palavras de ânimo, conselhos e companhia no momento preciso, pelas dicas de português e muita paciência comigo. Ao amigo e colega, Jesús Mena-Chalco pela grande ajuda e orientação, apoio e criticas construtivas oferecidas desde a graduação. A Andrea Britto, Isabel Hernández e Sílvia Pinto pela amizade, incentivo e por me permitir aprender muito delas. Também agradeço novamente à Maysa Macedo, Sílvia Pinto e ao Jesús Mena-Chalco pelas revisões do texto.

Aos meus companheiros de casa, em particular ao Carlos Herrera, Alfonso Phocco e Frank Julca, obrigada pela convivência e companheirismo durante estes anos. E ao Jorge Poco Medina, pelo carinho e por ter acreditado em mim nestes anos.

Finalmente, agradeço a FAPESP pelo apoio financeiro concedido durante os anos de mestrado, sem o qual eu não poderia ter me dedicado integralmente à essa pesquisa.

Resumo

MEDINA RODRIGUEZ, R. A. **Algoritmos Evolutivos aplicados ao Classificador baseado em Segmentos de Reta**. 2012. 89 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

Nos últimos anos o uso de técnicas de aprendizado computacional tornou se uma das tarefas comumente realizadas, pois tem inúmeras aplicações de reconhecimento de padrões, tais como: reconhecimento de voz, classificação de texto, reconhecimento facial, diagnóstico por imagens médicas, entre outras. Dessa forma, um grande número de técnicas que lidam com este tipo de problema tem sido desenvolvido até o momento. Neste trabalho apresentamos uma alternativa para melhorar a taxa acerto de classificação do classificador binário SLS, que apresentou resultados comparáveis com as SVMs. Nesse método, o Gradiente Descendente é utilizado para otimizar a posição final dos conjuntos de segmentos de reta que representarão cada classe. Embora convirja rapidamente a um valor ótimo, muitas vezes é possível o algoritmo parar em uma região de ótimos locais, que não representa o mínimo global. Dado esse problema, foram utilizados diferentes algoritmos evolutivos em combinação com o Gradiente Descendente a fim de melhorar a acurácia do classificador SLS. Adicionalmente à aplicação de algoritmos evolutivos na fase de treinamento do classificador SLS, foram exploradas duas propostas: (i) explorar o uso de diferente número de segmentos de reta para representar a distribuição de dados de cada classe. Dado que no algoritmo original do método SLS o número de segmentos de reta é igual para cada classe, o qual pode significar alguma perda de acurácia ou sobreposição dos segmentos de reta; (ii) estimar a melhor combinação de segmentos de reta a serem usados para cada classe. O uso de diferentes quantidades de segmentos de reta por classe pode ser de ajuda na obtenção de melhores porcentagens de acerto, mas determinar uma quantidade "ótima" que permita representar cada classe, é um trabalho difícil. Assim, usamos o algoritmo *X-Means*, que é um algoritmo de agrupamento, para estimar o número de segmentos de reta. As propostas exibiram bons resultados que possibilitam a aplicação do classificador SLS, com um algoritmo de treinamento híbrido, em problemas reais.

Palavras-chave: Segmentos de reta, aprendizado computacional supervisionado, algoritmos evolutivos, algoritmos de otimização, reconhecimento de padrões.

Abstract

MEDINA RODRIGUEZ, R. A. **Evolutive Algorithms applied to the Straight Line Segment Classifier**. 2012. 89 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

During the past years, the use of machine learning techniques have become into one of the most frequently performed tasks, due to the large amount of pattern recognition applications such as: voice recognition, text classification, face recognition, medical image diagnosis, among others. Thus, a great number of techniques dealing with this kind of problem have been developed until now. In this work, we propose an alternative training algorithm to improve the accuracy of the SLS binary Classifier, which produces good results that can be compared to Support Vector Machines. In that classifier, the Gradient Descent method has been used to optimize the final positions of two sets of straight line segments that represent each class. Although, this method quickly converges to an optimum, it is possible that the algorithm stops at a local optimum region, which does not guarantee a global minimum. Given that problem, we combine evolutive optimization algorithms with the gradient descent method to improve the accuracy of the SLS Classifier. In addition to our proposal of using evolutive algorithms, we also developed two proposals: (i) we explore the use of different number of straight line segments to represent the data distribution. Since the original SLS classifier algorithm uses the same number of segments for each class, which could lead to a loss of accuracy or straight line segments overlapping. So, using different number of segments could be the way to improve the accuracy; (ii) estimate the best combination of straight line segments to represent each class. Finding an "optimal" combination, can be a very difficult problem, so we propose the X-Means algorithm to determine the number of segments. The proposed methodology showed good results which can be used to solve some other real problems with the SLS classifier using the proposed hybrid training algorithm.

Keywords: straight line segments, supervised machine learning, evolutive algorithms, optimization algorithms, pattern recognition.

Sumário

Lista de Abreviaturas

SLS	Segmento de Reta (*Straight Line Segment*)
SLSs	Conjunto de Segmentos de Reta (*Straight Line Segments*)
SVM	Máquinas de Suporte Vetorial (*Support Vector Machines*)
BIC	Critério de Informação Bayesiano (*Bayesian Information Criterion*)
AIC	Critério de Informação de Akaike (*Akaike information Criterion*)
AD	Critério de Informação de Anderson-Darling (*Anderson-Darling Criterion*)
NFLT	(*No Free Lunch Theorem of Optimization*)
MSE	Erro Quadrático Médio (*Mean Squared Error*)
GD	Gradiente Descendente (*Gradient Descent*)
MDO	Método Dialético de Otimização (*Dialectical Optimization Method*)
AG	Algoritmos Genéticos (*Genetic Algorithms*)
KBM	K-Feixes (*K-Beams*)
nSLS	Número de Segmentos de Reta
sls-K	Número de segmentos de Reta obtidos pelo *K-Means*
sls-X	Número de segmentos de Reta obtidos pelo *X-Means*

Lista de Símbolos

$L_{p,q}$	Segmento de reta
$\mathcal{L}$	Conjunto de segmentos de reta
$\mathcal{C}$	Conjunto de classes
δ	contradição
$\mathbf{w}$	pólo
ω	conjunto de pólos
$\breve{w}$	antítese absoluta

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A tarefa de buscar algoritmos que a partir de instâncias fornecidas por um tutor externo produza hipóteses gerais que permitam fazer previsões futuras, é chamada de Aprendizado Supervisionado [Kotsiantis, 2007]. Em outras palavras, o objetivo do aprendizado supervisionado é construir um modelo conciso de predição de rótulos, geralmente uma função (chamada de *classificador*), a partir de um conjunto de dados de treinamento (*pares de exemplos*).

Neste tipo de aprendizagem, é um tutor quem fornece um rótulo ou um custo para cada exemplo de treinamento, sendo o objetivo a minimização da soma dos custos dos exemplos [Duda *et al*, 2001]; finalmente o classificador resultante é usado para atribuir rótulos a novas instâncias de teste. Cabe ressaltar que neste trabalho vamos nos focar só na classificação binária, ou seja dois rótulos $\{0\,,1\}$. Desta forma vamos procurar por uma função $f : \mathbb{R}^d \to \{0\,,1\}$ a partir de n pares de exemplos: $E_n = \{(x_i, y_i) \in \mathbb{R}^d \times \{0\,,1\} : i = 1, 2, ..., n\}$.

1.1 Motivação

A classificação supervisionada é uma das tarefas frequentemente realizadas e estudadas, pelas inúmeras aplicações tais como: reconhecimento de voz, reconhecimento de caracteres, mineração de dados, entre outras [Bishop, 2006; Duda *et al*, 2001]. Assim, entre os métodos comumente utilizados, podemos citar os Classificadores Lineares, Redes Neurais, Redes Bayesianas e Máquinas de Suporte Vetorial (SVM) [Kotsiantis, 2007]; sendo estas últimas consideradas como o estado da arte em máquinas de aprendizado, por ter um forte embasamento matemático e bons desempenhos em casos práticos [Abe, 2010].

No trabalho de Ribeiro e Hashimoto [2010], foi introduzido um novo método baseado em segmentos de reta, chamado de classificador SLS (*Straight Line Segments*), o qual consiste em encontrar uma função baseada na distância entre pontos e dois conjuntos de segmentos de reta para resolver problemas que envolvem separação de duas classes (classificação binária). Nesse trabalho foi mostrado que o classificador SLS é uma boa alternativa para ser usada em aplicações reais e comparável em termos de taxa de classificação com as SVM, para a classificação binária.

Este trabalho de dissertação explora a fase de treinamento do classificador SLS, a qual originalmente usa um número predeterminado de segmentos de reta e o método do Gradiente Descendente, como algoritmo de otimização, para encontrar as melhores posições dos conjuntos de segmentos de reta a fim de melhorar a acurácia do Classificador SLS. Assim propomos aplicar a combinação de diferentes algoritmos evolutivos com o gradiente descendente, na fase de treinamento, com o intuito de incrementar as possibilidades de encontrar um ótimo global ao gerar diferentes soluções como população inicial.

1.2 Objetivos

Tendo em consideração que nosso objetivo principal é melhorar a acurácia do Classificador SLS, este trabalho tem os seguintes objetivos específicos:

(a) Melhorar a taxa acerto de classificação do método SLS utilizando um outro método de otimização na fase de treinamento.

No método SLS, o gradiente descendente é utilizado para otimizar a posição final dos conjuntos de segmentos de reta que representam cada classe. Embora convirja rapidamente a um ótimo valor, muitas vezes é possível o algoritmo parar em uma região de ótimos locais que não representa o valor mínimo global. Dado esse problema, foram utilizados diferentes algoritmos de otimização em combinação com o Gradiente Descendente a fim de melhorar a acurácia do classificador SLS.

(b) Explorar o uso de diferente número de segmentos de reta para representar cada classe.

Dado que no algoritmo original do classificador SLS o número de segmentos de reta é igual para cada classe, o qual pode significar alguma perda de acurácia ou sobreposição dos segmentos de reta. Neste trabalho estudamos o desempenho do classificador quando são utilizadas diferentes quantidades de segmentos de reta por classe e apresentamos as taxas de acerto para as diferentes configurações do classificador SLS.

(c) Estimar o melhor número de segmentos a ser usado por classe.

O uso de diferente número de segmentos de reta por classe pode ajudar na obtenção de melhores resultados com relação à porcentagem de acerto do classificador SLS, mas determinar a número "ótimo" que permita representar cada classe, é um trabalho difícil. Assim, propomos o uso do algoritmo de *X-Means* [Pelleg e Moore, 1999], para estimar o número de segmentos de reta.

(d) Analisar o desempenho de classificação através da aplicação do classificador SLS a problemas reais.

Finalmente aplicar o método SLS a problemas reais incluindo a respectiva comparação com as SVM, pois é o método comumente utilizado na literatura.

Capítulo 2

Fundamentos Teóricos

Atualmente, o rápido crescimento do poder computacional tem permitido o processamento mais rápido de grandes conjuntos de dados, além de também, facilitar o uso de diversos e elaborados métodos para análise e classificação de dados. Ao mesmo tempo, a demanda de sistemas automáticos para o reconhecimento de padrões tem aumentado enormemente devido à grande disponibilidade de bases de dados e rigorosos requisitos de desempenho tal como: velocidade, precisão e custo.

Hoje em dia, lidamos com desafios do mundo das altas tecnologias como: detecção de faces, detecção de voz, recomendação de livros, leitura de manuscritos, entre outros; os quais podem ser resolvidos usando um computador. Mas para enfrentar estes desafios, é necessário programar o computador para que ele tenha capacidade de aprender a partir de um conjunto de exemplos fornecidos por um tutor além de treinar com eles.

Dada então a necessidade de programar o computador, criou-se o conceito do aprendizado computacional. No ano 1959, Arthur Samuel definiu o aprendizado computacional como a área de estudo que fornece aos computadores a capacidade de aprender sem serem explicitamente programados [Samuel, 1959].

2.1 Sistema de Reconhecimento de Padrões

Existem diversas maneiras de abordar reconhecimento de padrões, uma delas está representada conforme a Figura 2.1. A mais simples e intuitiva abordagem para modelar um classificador está baseada no conceito de similaridade: padrões que são similares devem ser atribuídos à mesma classe. Uma medida de similaridade muito utilizada é a medida de correlação [Kotsiantis, 2007]. Uma das abordagens mais estudadas na literatura, é a classificação estatística, na qual procura-se sempre saber qual é a probabilidade de um padrão pertencer a uma classe, associando a decisão a um risco de erro.

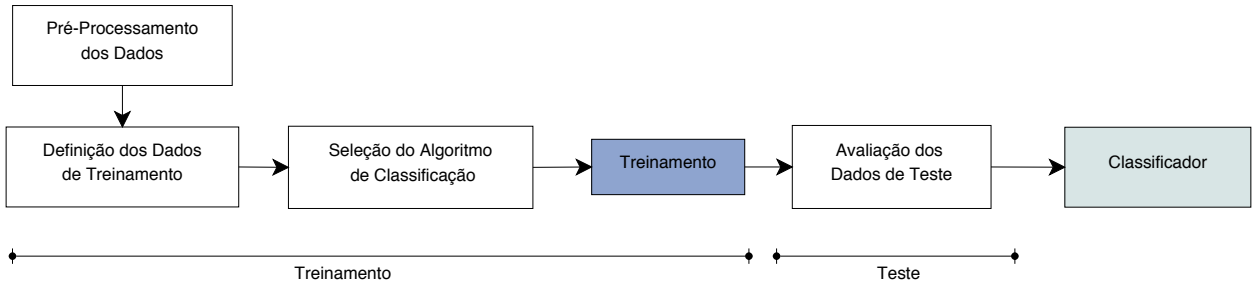


Figura 2.1: *Representação de um sistema baseado em Aprendizado Computacional Supervisionado. Figura adaptada do artigo apresentado por Kotsiantis [2007]*

No *survey* de Jain *et al.* [2000] é feita uma classificação das diferentes estratégias utilizadas para desenhar classificadores baseada no tipo de informação disponível sobre as funções de densidade de probabilidade. A Figura 2.2 apresenta uma a classificação das abordagens que são utilizadas em Reconhecimento de Padrões. A abordagem aplicada na construção do classificador, objeto de estudo, neste trabalho (classificador SLS - *Straight Line Segments*) está ressaltada em linhas pontilhadas, conhecida como abordagem geométrica, a qual frequentemente constrói as fronteiras de decisão a partir da otimização de determinadas funções de custo [Jain *et al*, 2000], e sem pressupor nenhuma forma de distribuição.

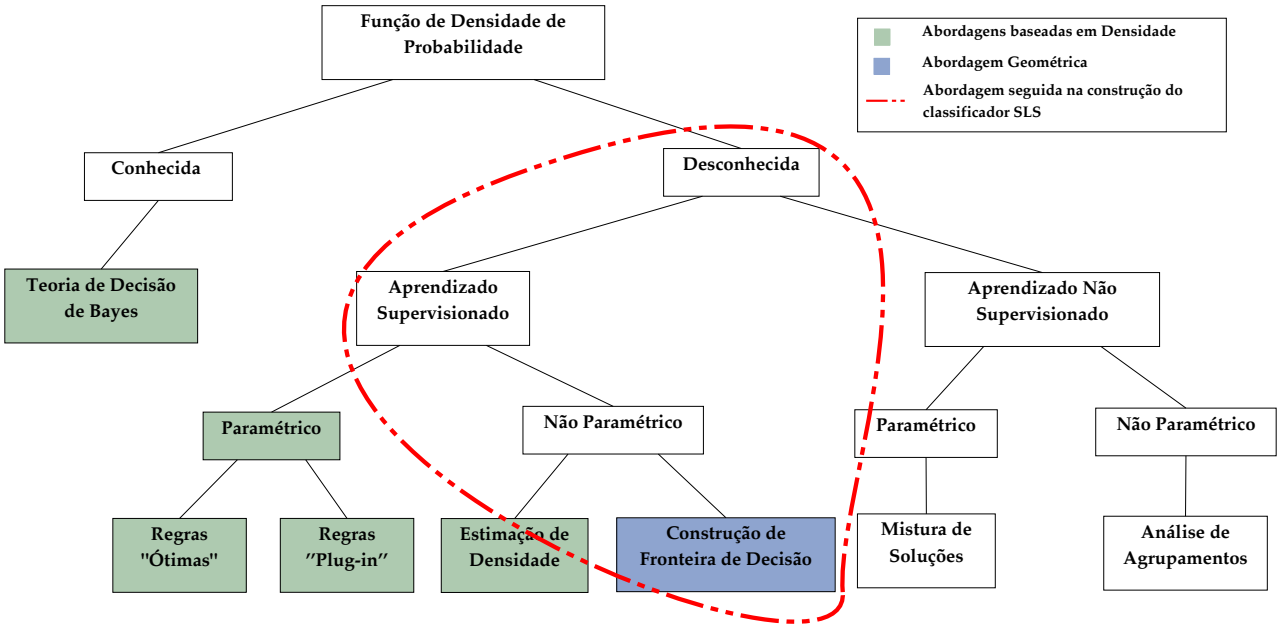


Figura 2.2: *Diferentes abordagens em Reconhecimento de Padrões. Diagrama adaptado do trabalho apresentado por Jain et al. [2000].*

Dadas as diferentes aplicações emergentes, é claro que não existe uma única abordagem para atingir uma classificação ótima e, é assim que múltiplos métodos e abordagens tem sido usados. Consequentemente, combinar vários classificadores é uma prática comum em Reconhecimento de Padrões [Rocha e Goldenstein, 2009].

A seguir apresentamos um breve resumo de técnicas de Aprendizado Computacional utilizadas na área de Reconhecimento de Padrões.

2.2 Aprendizado Computacional Supervisionado

O problema central da aprendizagem é induzir (construir) funções gerais a partir de exemplos de treinamento específicos. Informalmente, qualquer processo que é capaz de "aprender um conceito" a partir de exemplos que o ilustram é denominado **aprendizado**.

O objetivo do aprendizado computacional supervisionado, na classificação binária, é construir um modelo conciso de predição dos rótulos para duas classes, geralmente utilizando uma função $f : \mathbb{R}^d \rightarrow \{0, 1\}$ a partir de um conjunto de dados conhecidos, representado por pares de exemplos: $E_n = \{(x_i, y_i) \in \mathbb{R}^d \times \{0, 1\} : i = 1, 2, ..., n\}$. Neste tipo de aprendizagem é um tutor quem fornece um rótulo para cada categoria ou um custo para cada padrão no conjunto de treinamento e o objetivo é reduzir a soma dos custos desses padrões, onde o classificador resultante é usado para atribuir rótulos às instâncias de teste [Duda *et al*, 2001].

A eficácia do espaço de representação (conjunto de características) está determinado pela melhor forma em que os padrões pertencentes a diferentes classes podem ser separados. Dado um conjunto de padrões de treinamento para cada classe, o objetivo é estabelecer fronteiras de decisão no espaço de características, que separem os padrões em suas respetivas classes.

Considera-se que uma máquina de aprendizado é consistente quando a expectativa de erro na fase de teste, ou seja, quando os novos dados são apresentados à máquina já treinada, é bem próxima da expectativa de erro do classificador. Considera-se um erro quando a máquina de aprendizado rotula divergindo do tutor [Ribeiro, 2009].

Nas seguintes subseções iremos descrever brevemente três dos métodos de Reconhecimento de Padrões existentes, os quais tem alguma relação com o trabalho proposto nesta dissertação.

2.2.1 Classificador de Bayes

Sendo que a probabilidade de um evento discreto A ocorrer é $P(A)$, e dada uma variável aleatória contínua (x) a função de densidade de probabilidade é definida como $p(x)$. Para um vetor de variáveis aleatórias $\mathbf{x}$, a função de densidade de probabilidade é definida como $p(\mathbf{x})$, e a probabilidade condicional de A dado B é $P(A|B)$. Para o vetor $\mathbf{x}$, $p(\mathbf{x}|A)$ é definida como a probabilidade condicional de A dado $\mathbf{x}$ [Jain *et al*, 2000; Kotsiantis, 2007].

A abordagem Bayesiana supõe que as probabilidades de cada classe $P(\omega_i)$ e as densidades de probabilidade condicionais $p(\mathbf{x}|\omega_i)$ de $\mathbf{x}$ com respeito a cada uma das classes $\omega_i, i = 1, 2, ..., \mathcal{C}$, são conhecidas [Figueiredo, 2004]. Sabendo o valor das probabilidades condicionais, podemos utilizar o teorema de Bayes para calcular a probabilidade a posteriori, ou seja $P(\omega_i|\mathbf{x})$, definida como:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}, \tag{2.1}$$

onde:

- $p(\mathbf{x}|\omega_i)$ é a probabilidade condicional de classe;
- $P(\omega_i)$ é a probabilidade a priori;
- $p(\mathbf{x}) = \sum_{j=1}^{c} P(\omega_j)p(\mathbf{x}|\omega_j)$ que é a evidência;
- $P(\omega_i|\mathbf{x})$ é a probalidade a posteriori.

A regra de decisão de Bayes que resulta em uma mínima taxa de erro é dada como: "Decidir ω_i se $P(\omega_i|\mathbf{x}) \geq P(\omega_j|\mathbf{x})$ para todo $j \neq i$". Conhecendo-se as distribuições de probabilidades, classifica-se sempre com a classe que tem maior probabilidade de ocorrer para um dado $\mathbf{x}$, conforme:

$$f_B(\mathbf{x}) = \arg_i \max_{\omega_i \in \mathcal{C}} P(\omega_i|\mathbf{x}). \tag{2.2}$$

2.2.2 Máquinas de Suporte Vetorial

As máquinas de suporte vetorial (***SVM***), apresentadas pela primeira vez no trabalho de Vapnik [1995], são as técnicas mais usadas de aprendizado computacional supervisionado [Abe, 2010].

Basicamente é um classificador binário, onde o critério de otimização é a largura da margem entre as classes como, por exemplo, a área vazia ao redor da fronteira de decisão definida pela distância aos padrões de treinamento mais próximos. Estes padrões são chamados de padrões de suporte e definem a função de classificação. O número de padrões é reduzido através da maximização da margem [Kotsiantis, 2007].

O problema de otimização no treinamento das SVMs, como pode-se observar na Figura 2.3, consiste em encontrar um hiperplano ótimo, aquele que fique mais longe dos exemplos de treinamento mais próximos. Os vetores de suporte estão representados pelos pontos preenchidos. Na fase de treinamento as SVMs atingem um mínimo global, evitando terminar em um mínimo local, o qual pode acontecer em outros algoritmos de busca como as redes neurais.

As SVMs permitem obter ótimos desempenhos de classificação em casos práticos. Um problema da utilização das SVMs, segundo Hammer *et al.* [2004], é a complexidade computacional, pois é proporcional ao número de vetores de suporte resultantes do treinamento, e esse número tende a crescer quando o tamanho da amostra aumenta. Contudo, as SVM são técnicas de classificação binária, assim no caso de problemas multi-classe este problema se reduz a um conjunto de vários problemas de duas classes [Abe, 2010].

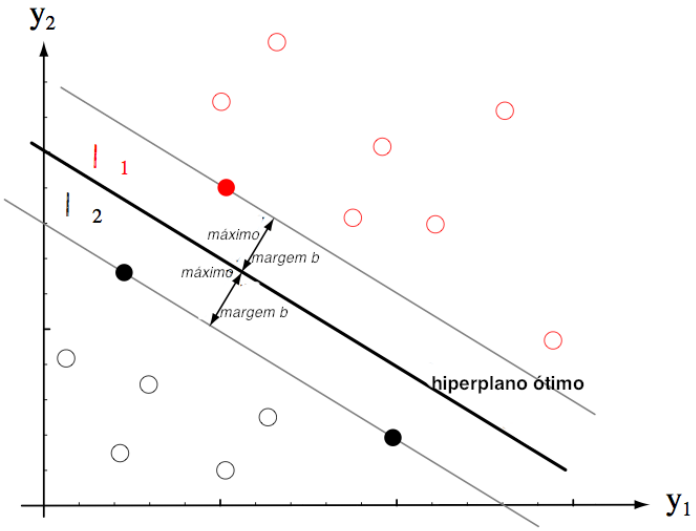


Figura 2.3: *Treinamento nas Máquinas de Suporte Vetorial. Figura extraída do livro* Pattern Classification *[Duda et al, 2001]*

2.2.3 Classificador SLS (*Straight Line Segments*)

Este classificador baseado em segmentos de reta, está centrado na classificação binária e foi apresentado nos trabalhos de Ribeiro e Hashimoto [2006, 2008]. Uma característica principal nele é que, ao contrário de usar distâncias a pontos (como no método K-vizinhos [Wu e Vipin, 2009]), usa distâncias a segmentos de reta, os quais representam infinitos pontos, incrementando a capacidade de aprendizagem. Um fluxograma do classificador é apresentado na Figura 2.4, onde podemos observar duas fases: treinamento e teste, as quais serão descritas nesta subseção.

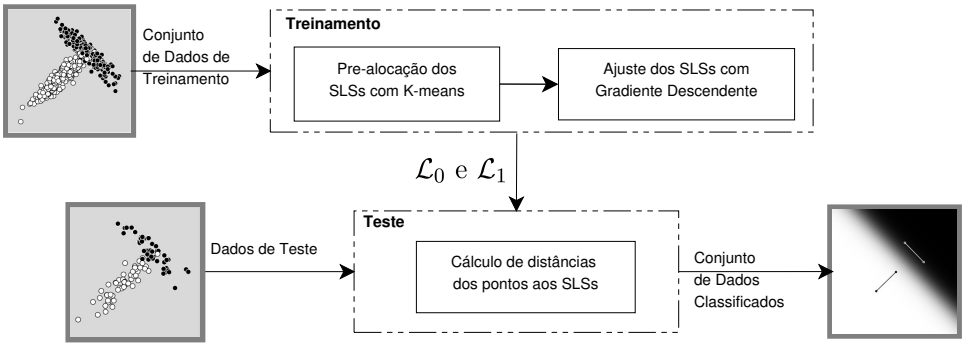


Figura 2.4: *Diagrama do fluxo de dados do Classificador SLS, para as fases de treinamento e teste.*

Definições Matemáticas Básicas [Ribeiro e Hashimoto, 2010]

(a) Dado que o classificador SLS é binário, são definidas duas classes: $\{0, 1\}$;

(b) Sejam $p, q \in \mathbb{R}^{d+1}$, o segmento de reta (SLS) $L_{p,q}$ com extremidades $\boldsymbol{p}$ e $\boldsymbol{q}$ é definido como:

$$L_{p,q} = \{x \in \mathbb{R}^{d+1} : x = p + \lambda \cdot (q - p), 0 \leq \lambda \leq 1\}\,; \tag{2.3}$$

(c) Dado um ponto $x \in \mathbb{R}^d$, a extensão do ponto para $\mathbb{R}^{d+1}$ é definida como $x_e = (x, 0)$; e dado $L_{p,q} \subseteq \mathbb{R}^{d+1}$, representado na Figura 2.5(a), a pseudo-distância entre x e L é definida como:

$$distP(x, L) = \frac{dist(x_e, p) + dist(x_e, q) - dist(p, q)}{2}\,, \tag{2.4}$$

onde $dist(a, b)$ é a distância Euclidiana entre os pontos $a, b \in \mathbb{R}^{d+1}$;

(d) Uma coleção de segmentos de reta (SLSs) será denotada por:

$$\mathcal{L} = \{L_{p_i,q_i} : p_i, q_i \in \mathbb{R}^{d+1}, i = 1, ..., m\}\,, \tag{2.5}$$

onde m representa o número de segmentos de reta para cada classe $\{0, 1\}$;

(e) Dado um ponto $x \in \mathbb{R}^d$, definimos a distância de x a uma coleção de segmentos de reta $\mathcal{L}$ como:

$$distL(x, \mathcal{L}) = \lim_{\varepsilon \to 0} \left(\sum_{L \in \mathcal{L}} \frac{1}{distP(x, L) + \varepsilon} \right)^{-1}\,, \tag{2.6}$$

onde ε é um valor utilizado para evitar divisão por zero;

(f) Seja $\mathcal{C}$ um conjunto de duas classes $\mathcal{C} = \{0\,, 1\}$, $\mathcal{L}_c$ o conjunto de segmentos de reta que definem a região da classe c; e seja $\mathcal{S}$ uma sequência de conjuntos de segmentos de reta, tal que, $\mathcal{S} = (\mathcal{L}_0\,, \mathcal{L}_1)$. Então, a função de classificação $y_{\mathcal{S}} : \mathbb{R}^d \to \mathcal{C}$ deve associar um ponto em $\mathbb{R}^d$ à classe do conjunto de segmentos de reta mais próximo; portanto, deve ser equivalente à seguinte equação:

$$y_{\mathcal{S}}(x) = \arg_c\{ \min_{\mathcal{L}_c \in \mathcal{S}} distL(x, \mathcal{L}_c)\}\,; \tag{2.7}$$

(g) Dados dois conjuntos de segmentos de reta $\mathcal{L}_0$ e $\mathcal{L}_1$, representados na Figura 2.5(b), para definir a função de classificação é necessário definir a função discriminante $T_{\mathcal{L}_0,\mathcal{L}_1} : \mathbb{R}^d \to \mathbb{R}$ como:

$$T_{\mathcal{L}_0,\mathcal{L}_1}(x) = \lim_{\varepsilon \to 0} \left(\sum_{L \in \mathcal{L}_1} \frac{1}{distP(x, L) + \varepsilon} - \sum_{L \in \mathcal{L}_0} \frac{1}{distP(x, L) + \varepsilon} \right)\,; \tag{2.8}$$

(h) Considerando a função discriminante, a função de classificação **binária** é definida como uma função sigmóide, denotada por:

$$y_{\mathcal{L}_0,\mathcal{L}_1}(x) = \frac{1}{1 + e^{-g(T_{\mathcal{L}_0,\mathcal{L}_1}(x))}}\,, \tag{2.9}$$

onde g é uma constante real positiva. Esta função de classificação é diferenciável, e por isso é possível usar o método do gradiente descendente [Wright, 2010] na fase de treinamento do classificador SLS.

(i) Utilizando um limiar em 0.5 considerando que $\mathbf{x}$ está na classe 0 quando $y_{\mathcal{L}_0,\mathcal{L}_1}(x) < 0.5$ e, de outro modo, está na classe 1. A Equação 2.9 é equivalente à Equação 2.7, pela prova extraída

do trabalho de Ribeiro e Hashimoto [2010] descrita a seguir:

Prova: Se $distL(x, \mathcal{L}_1) > distL(x, \mathcal{L}_0) \neq 0$

$$\begin{array}{rcl} distL(x, \mathcal{L}_1) & > & distL(x, \mathcal{L}_0) \ , \\ distL(x, \mathcal{L}_1)^{-1} & < & distL(x, \mathcal{L}_0)^{-1} \ , \\ distL(x, \mathcal{L}_1)^{-1} & - & distL(x, \mathcal{L}_0)^{-1} < 0 \ , \\ T(x, \mathcal{S}) & < & 0 \ , \\ y_{\mathcal{S}}(x) & < & 0.5 \ , \end{array} \tag{2.10}$$

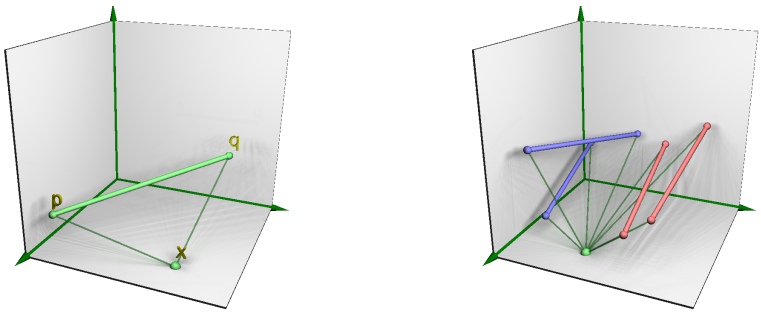
se $distL(x, \mathcal{L}_0) = 0$, então pelas Equações **??** e 2.9: $y_{\mathcal{S}}(x) = 0 < 0.5$;

Prova: Se $distL(x, \mathcal{L}_1) < distL(x, \mathcal{L}_0) \neq 0$

$$\begin{array}{rcl} distL(x, \mathcal{L}_1) & < & distL(x, \mathcal{L}_0) \ , \\ distL(x, \mathcal{L}_1)^{-1} & > & distL(x, \mathcal{L}_0)^{-1} \ , \\ distL(x, \mathcal{L}_1)^{-1} & - & distL(x, \mathcal{L}_0)^{-1} > 0 \ , \\ T(x, \mathcal{S}) & > & 0 \ , \\ y_{\mathcal{S}}(x) & > & 0.5 \ , \end{array} \tag{2.11}$$

e do mesmo modo, se $distL(x, \mathcal{L}_1) = 0$, então pelas Equações 2.8 e 2.9: $y_{\mathcal{S}}(x) = 1 > 0.5$;

■



(a) (b)

Figura 2.5: *(a) Pseudo-distância entre x e $L_{p,q}$, (b) Duas coleções de SLSs: $\mathcal{L}_0$(azul) e $\mathcal{L}_1$(vermelho). Figuras extraídas de [Ribeiro 2009].*

O algoritmo de treinamento deste classificador consiste em duas fases, as quais estão apresentadas conforme a Figura 2.6:

1. **Pre-alocação dos segmentos de reta:** Consiste em encontrar as posições iniciais dos segmentos de reta para cada classe. Para atingir este objetivo usa o algoritmo *K-Means* [Meilă 2006] duas vezes:

   (1) O conjunto de amostras é dividido em duas classes $\{0, 1\}$ e depois é aplicado *K-Means* com K igual ao número de segmentos de reta com os quais vão se representar as classes. Veja na Figura 2.6(b) um exemplo para $K = 2$;

   (2) O algoritmo *K-Means* é aplicado pela segunda vez, em cada agrupamento encontrado no item anterior, com $K = 2$, como na Figura 2.6(c), pois cada centróide representará uma extremidade de um segmento de reta conforme a Figura 2.6(d).

2. **Ajuste dos segmentos de reta:** O objetivo desta fase é minimizar uma função de erro, neste caso o Erro Quadrático Médio, definido nas Equações 2.12 e 2.13 (veja a Figura 2.6(h)). Para conseguir este objetivo é usado o método de Gradiente Descendente. Apesar do método de gradiente descendente não garantir o mínimo global e fazer com que a posição final dos segmentos dependa da posição inicial dos mesmos, este método foi aplicado com sucesso por Ribeiro e Hashimoto [2008].

$$MSE(y_{\mathcal{L}_0,\mathcal{L}_1}) = \frac{1}{N}\sum_{i=1}^{N}[err_i(y_{\mathcal{L}_0,\mathcal{L}_1})]^2 \tag{2.12}$$

$$err_i(y_{\mathcal{L}_0,\mathcal{L}_1}) = y_{\mathcal{L}_0,\mathcal{L}_1}(x_i) - y_i, \tag{2.13}$$

onde N representa o número de amostras no conjunto de treinamento.

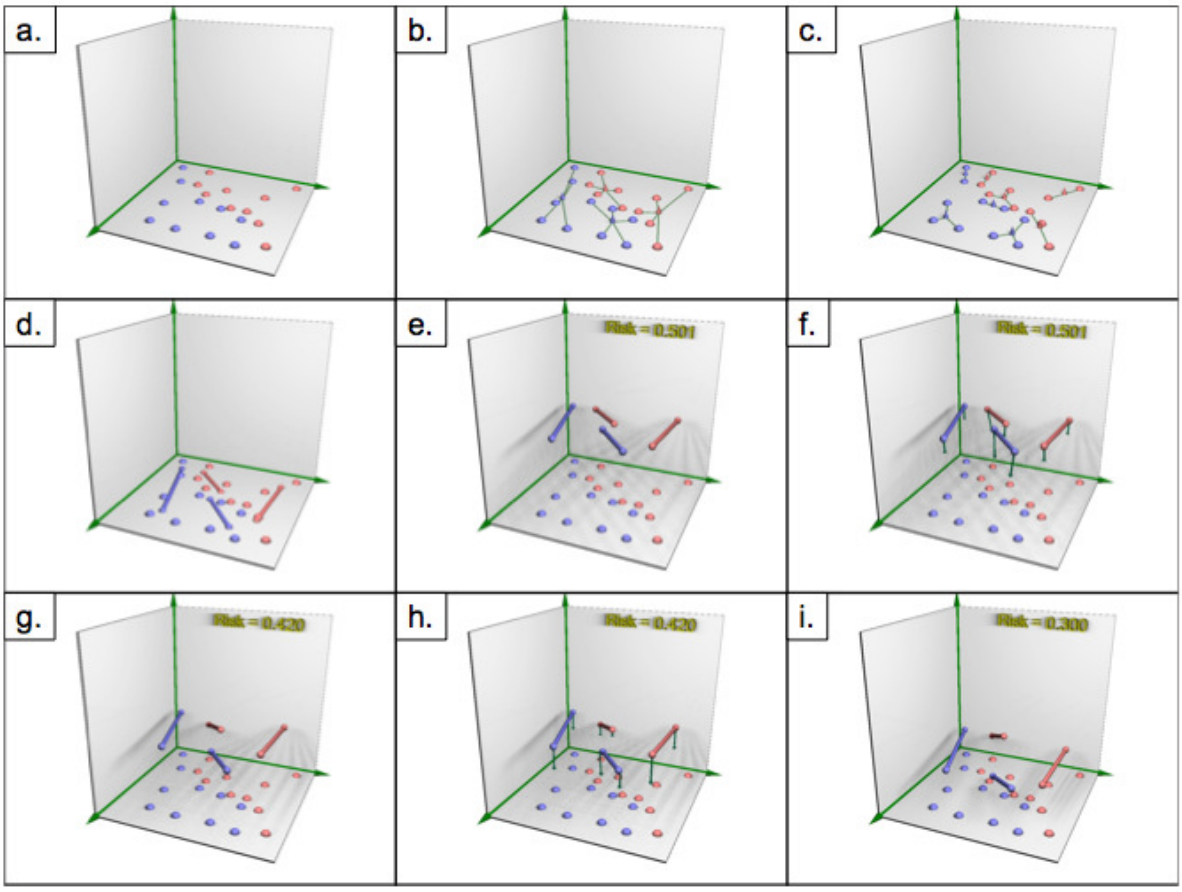


Figura 2.6: *Pré-alocação dos Segmentos de Reta (a-e) , Ajuste dos Segmentos de Reta (f-i). Figura extraída do trabalho de Ribeiro [2009].*

2.3 Aprendizado Computacional Não Supervisionado

O aprendizado computacional não supervisionado refere-se a situações onde o objetivo é construir fronteiras de decisão baseadas em dados de treinamento não rotulados, com o objetivo de agrupar os dados que são próximos ou semelhantes entre si. O objetivo do agrupamento de dados, também chamado de *"clustering"* é descobrir agrupamentos naturais dentro de um conjunto de padrões, pontos ou objetos. Pois, em várias aplicações de Reconhecimento de Padrões é difícil, caro ou impossível rotular com plena certeza, um conjunto de treinamento com sua verdadeira classe (*"ground truth"*).

O análise por agrupamento é uma técnica muito importante e útil. A velocidade, confiança e consistência com a qual cada algoritmo de clusterização pode organizar grandes quantidades de dados constituem boas razões para usá-los em aplicações como mineração de dados, recuperação de informação, segmentação de imagens, compressão e codificação de sinais e aprendizado computacional [Kotsiantis, 2007].

A seguir descrevemos, brevemente, dois tipos de algoritmos de classificação não supervisionada, ambos são utilizados no atual trabalho a fim de comparar qual deles oferece uma melhor representação dos dados de treinamento.

2.3.1 *K-Means*

A idéia do algoritmo *K-Means* (K-Médias, em português) [Meilă, 2006]) é fornecer uma classificação de informações de acordo com os próprios dados. Esta classificação é baseada na análise e comparações entre os valores numéricos dos dados.

Esta técnica consiste em associar ao padrão x, a ser classificado, à classe que ocorre com maior freqüência dentre as K amostras mais próximas a ele, segundo uma distância adotada. O algoritmo atribui cada ponto ao agrupamento cujo centróide está mais próximo desse ponto. Algumas das principais características deste algoritmo são descritas a seguir:

- *K-Means* é um algoritmo simples que pode ser adaptado a vários tipos de problemas;
- Identifica agrupamentos hiper-esféricos, pode ser modificado para encontrar agrupamentos hiper-elipsoidais usando a distância de Mahalanobis, computacionalmente eficiente;
- É necessário especificar o valor de K e os centros iniciais dos agrupamentos;
- São necessários parâmetros adicionais para criar novos agrupamentos e fusão de agrupamentos;
- Embora pode ser provado que o algoritmo sempre termina, o algoritmo nem sempre encontra uma configuração ótima, correspondente à minimização da função objetivo global [MacQueen, 1967];
- O algoritmo *K-Means* é significativamente sensível aos centros dos agrupamentos iniciais selecionados aleatoriamente. Para reduzir este efeito, o algoritmo pode ser executado várias vezes;
- Segundo Jiawei e Kamber [2001], o algoritmo é incapaz de lidar com dados ruidosos, pois é muito sensível a pontos isolados (*outliers*).

Sendo assim, o algoritmo apresenta os seguintes passos:

1. Colocar K pontos no espaço representado pelos objetos a serem agrupados. Estes pontos representam o grupo inicial de centróides;
2. Atribuir cada objeto ao grupo que contêm o centróide mais próximo;
3. Quando todos os objetos foram atribuídos, recalcular as posições dos K centróides;
4. Repetir os passos 2 e 3 até a convergência, ou seja, até quando os centróides não se deslocarem mais. Isto produz uma separação dos objetos em grupos a partir dos quais a métrica a ser minimizada pode ser calculada.

2.3.2 *X-Means*

Um problema do algoritmo *K-Means* é decidir qual é o valor de K, ou seja, o número de agrupamentos. Nesse contexto, Pelleg e Moore [2000] apresentaram o algoritmo *X-Means* (X-Médias, em português), o qual determina o valor de K no final da execução. Este algoritmo começa com um agrupamento e logo continua adicionando agrupamentos até os centróides selecionados modelarem os dados "bem o suficiente".

Segundo Bies *et al.* [2009], formalmente, o algoritmo consiste em:

1. Dado o número inicial de agrupamentos, executar o algoritmo *K-Means* no conjunto de dados até a convergência;
2. Dividir os centróides de cada agrupamento resultante em dois filhos e executar *K-Means* localmente em cada agrupamento e em cada par de filhos;
3. Para cada agrupamento, comparar o valor de BIC[1] do agrupamento pai com o valor do BIC para os dois agrupamentos filhos, e manter o agrupamento que obtiver o valor mais alto (o agrupamento que parece modelar melhor os dados);
4. Voltar para o passo 1.

O algoritmo *K-Means* usado nesta técnica é, na verdade, um algoritmo acelerado, o qual foi desenvolvido pelos mesmos autores do *X-Means* e apresentado no trabalho [Pelleg e Moore, 1999]. A ideia é usar *kd-trees* para acelerar o algoritmo quando é aplicado para pontos em baixas dimensões, salvando nos nós da árvore estatísticas sobre os pontos contidos no hiper-retângulo, com o qual as iterações do *K-Means* são executadas mais rapidamente quando grandes quantidades de pontos estão mais próximos de um centróide do que dos outros.

Além disso, o algoritmo *X-Means* mantêm em uma lista todos os agrupamentos que não foram divididos ou que perderam pontos para outros agrupamentos, desta forma o número de divisões de agrupamentos em cada iteração pode ser diminuído. Retornando depois de todas as iterações o melhor modelo que representa os dados [Bies *et al*, 2009].

2.4 Algoritmos de Otimização aplicados no Aprendizado Computacional

Na fase de treinamento dos classificadores supervisionados é comumente necessário usar um algoritmo de otimização de maneira que, uma determinada função de risco seja minimizada. Dessa forma, existem na literatura múltiplos classificadores que usam diferentes algoritmos de otimização, existindo até uma combinação de alguns deles, para assim tentar melhorar a classificação.

A otimização pode ser dividida em duas classes: global e local. A otimização global encontra a melhor solução do conjunto de "todas" as soluções possíveis. No entanto, a otimização local encontra a melhor solução dentro de um conjunto de soluções que está próximo a outro. Na otimização local, a solução encontrada depende do ponto de início do processo de busca de otimização. Ao contrário de otimização global que sempre encontrará a melhor solução possível, independentemente das condições de início do processo de busca, porém, geralmente, requisita um maior poder computacional.

[1] É um critério para a seleção de modelos entre um conjunto finito de modelos (apresentado no trabalho de [Schwarz , 1978]) e está baseado na função de verossimilhança e estreitamente relacionado com AIC (*Akaike Information Criterion*) [Akaike, 1976].

Teorema *"No Free Lunch"* para Algoritmos de Otimização

"Não há almoço grátis"[2], é uma frase popular que expressa a ideia de que é impossível conseguir algo sem dar nada em troca. Foi demonstrado pela primeira vez no trabalho de Wolpert e Macready [1997], onde é estabelecido que para qualquer algoritmo, qualquer "excelente" desempenho para algum tipo de problema é exatamente pago em desempenho com relação a outro tipo de problema.

Segundo Ho e Peyne [2001], este teorema pode ser resumido como: "é impossível, criar uma estratégia de otimização que seja de uso universal". Além disso, a única forma de uma estratégia superar outra é, se é especializada na estrutura de um problema específico em questão.

Considerando a ideia apresentada por este teorema como motivação, pode-se encontrar na literatura diferentes tipos de algoritmos de otimização, cada um deles inspirado em diferentes áreas como: Biologia, Ciências Sociais, Filosofia, etc. Nas subseções a seguir serão descritos, de maneira breve, os algoritmos utilizados neste trabalho.

2.4.1 Gradiente Descendente

O gradiente descendente é um algoritmo de otimização que usa as derivadas da função objetivo $f(x)$. Este método inicia em um ponto P_0 e tantas vezes quanto seja necessário se desloca desde P_i até P_{i+1} através da minimização ao longo da linha que se estende a partir de P_i na direção de $- \bigtriangledown f(P_i)$, a descida do gradiente local, a fim de encontrar o ponto com maior inclinação.

Portanto, seja $f : \mathbb{R} \to \mathbb{R}$ uma função uni-dimensional, o procedimento é iterativo e pode ser descrito como:

1. Calcular a derivada $\mathrm{d}f(x)/\mathrm{d}x$ da função objetivo em relação à variável independente x;
2. Atualizar o valor da variável x de acordo com :

$$x_i = x_{i-1} - \alpha_k \frac{\mathrm{d}f(x_{i-1})}{\mathrm{d}x}, \tag{2.14}$$

   onde α é a taxa de aprendizado e i o número de iteração;
3. Repetir os passos anteriores até a convergência ser atingida.

No caso multi-dimensional o método de gradiente descendente pode ser resolvido como segue:

1. Seja $f : \mathbb{R} \to \mathbb{R}$ uma função real de n variáveis. O gradiente da função objetivo f avaliada no ponto x e definida por $\nabla f(x)$ é uma função com valores representados por um vetor cuja i-ésima componente é a derivada parcial de f com respeito de x_i, $\partial f(x)/\partial x_i$;
2. Sabemos que, dado um ponto x onde o valor do vetor gradiente é diferente de zero, o gradiente negativo $-\nabla f(x)$ aponta na direção com maior inclinação para os menores valores da função objetivo f.

   Na verdade, $-\nabla f(x)$ representa a descida mais íngreme da função f, no sentido que o valor da função diminui mais rápido ao longo da direção do gradiente negativo do que no sentido contrário;
3. A partir de um ponto inicial x_0 cada solução aproximada sucessiva será definida por:

$$x_{k+1} = x_k - \alpha_k \bigtriangledown f(x_k), \tag{2.15}$$

[2]Tradução literal de *"There is no free lunch"*

onde α_k é um parâmetro da estratégia *line search* [Heath, 2002] que determina até onde ir na direção dada;

4. Conhecida a direção de descida, tal como o gradiente negativo, determinar o valor apropriado para α_k é um problema de minimização em uma dimensão:

$$\min_{\alpha} f(x_k - \alpha \bigtriangledown f(x_k)) \tag{2.16}$$

2.4.2 *K-Beams*

Algumas vezes, podem existir vários mínimos locais na otimização de funções não convexas, e assim, a otimização global não é garantida. Uma prática comum é procurar vários mínimos locais, executando o algoritmo com diferentes pontos de inicio e salvando a melhor solução encontrada.

O algoritmo *K-Beams* (K-Feixes, em português) é uma variação do algoritmo *Local Search* (Busca Local) [Russell e Norvig 2002], o qual mantém um estado atual único e procura uma melhor solução se deslocando aos pontos vizinhos. Se bem usar um único estado significa um mínimo uso de memória, também gera uma falta de diversidade nas soluções. Este algoritmo lida com esse problema, controlando um valor de k estados e não um único estado atual, como no *Local Search*.

Assim, a ideia geral do algoritmo, segundo Russell e Norvig [2002] é a seguinte:

1. Inicia com k estados gerados aleatoriamente;
2. Em cada estado, são gerados sucessores para cada um;
3. Avaliar a função objetivo;
4. Se não for atingido o valor ideal, seleciona os melhores k estados do conjunto e repete os passos 1 e 2, caso contrário, o algoritmo deve parar.

Diferente do algoritmo *Random Search* [Russell e Norvig 2002], no qual cada processo de busca é executado independentemente dos outros, o algoritmo descrito nesta subseção compartilha informação útil entre as buscas paralelas, ou seja, os estados que geram os melhores sucessores dizem para os outros que nessa região os valores podem ser melhores. Dessa forma são abandonadas buscas infrutíferas e re-dirige todos os seus recursos até o lugar mais promissório. Informalmente, é uma forma de busca estocástica na qual a busca é às "cegas", porque não usa informação das soluções anteriores [Houck *et al*, 1996].

No entanto, pode-se considerar uma desvantagem a falta de diversidade que o algoritmo pode sofrer, pois pode ficar preso em uma pequena região do espaço. Para resolver este problema foi criado o algoritmo *Stochastic K-Beams*, ao invés de escolher os melhores sucessores, a escolha é feita aleatoriamente com a probabilidade de escolher o próximo sucessor dada por uma função crescente do seu valor [Russell e Norvig 2002].

2.4.3 Algoritmos Genéticos

Podemos dizer que os algoritmos genéticos foram inspirados pela teoria da evolução de Darwin: "os melhores indivíduos tendem a sobreviver transferindo seu material genético para as próximas gerações; enquanto que o menos adaptados tendem a desaparecer" [Goldberg 1989]. Estas técnicas são comumente aplicadas para encontrar soluções aproximadas em problemas de otimização e pertencem a uma classe particular de algoritmos evolutivos, a qual está inspirada na biologia evolutiva, tal como: seleção natural, mutação e hereditariedade.

Segundo Russell e Norvig [2002], um algoritmo genético é uma variante do algoritmo *Stochastic K-Beams*, no qual os sucessores dos estados são gerados pela combinação de dois estados "pais" em

vez de modificar um estado só. Consiste de um método de busca baseado no conceito de evolução por seleção natural. Este algoritmo inicia com uma população inicial e depois a população evolui em um determinado número de gerações (iterações), fornecendo uma população evoluída (solução final).

Na implementação dos algoritmos genéticos, um cromossomo ou indivíduo é uma representação codificada (normalmente em uma sequência de zeros e uns) de uma possível solução do problema de otimização. Uma população é um conjunto de indivíduos, ou seja, ela contém algumas possíveis soluções do problema. Uma população inicial é gerada aleatoriamente, apesar de conhecimentos a priori do domínio do problema podem ser incorporados (pelo uso de heurísticas) para gerar potenciais soluções iniciais do problema [Wright, 2010].

A evolução é realizada por meio de sucessivas iterações conhecidas como gerações, como pode-se observar na Figura 2.7. A cada geração, a adaptação de cada indivíduo é analisada e alguns cromossomos são selecionados para a próxima geração, e recombinados e mutados para formar uma nova população, que por sua vez, é utilizada para a próxima iteração do algoritmo. O processo de geração termina quando não há mais nenhuma melhora na qualidade dos indivíduos da população ou um número máximo de iterações é alcançado. A saída do algoritmo é o melhor indivíduo da população final [Haupt e Haupt, 2004].

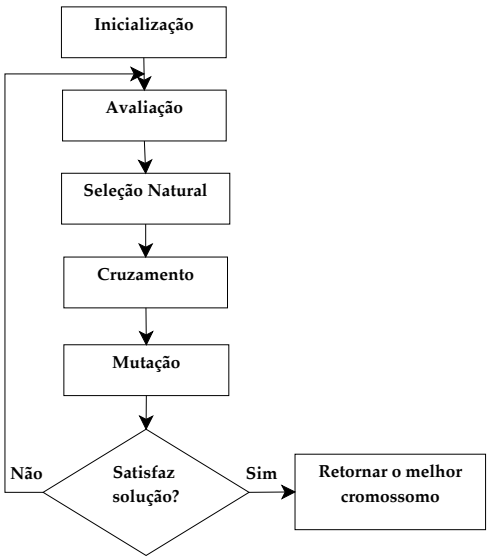


Figura 2.7: *Fluxograma dos processos na construção de um algoritmo genético*

Uma breve descrição sobre os operadores dos algoritmos genéticos, descritos no livro de Weise [2008], é apresentada a seguir:

(a) *Seleção Natural:* Avalia a função objetivo, e escolhe o melhor cromossomo. Existem várias formas de seleção: seleção por classificação, por roleta, por torneio, uniforme e seleção estocástica.

(b) *Cruzamento:* São gerados novos cromossomos a partir de um intercâmbio de material genético dos pais. cruzamento de um ponto, multi-ponto, segmentado, uniforme e por combinação parcial.

(c) *Mutação:* A importância deste operador reside no fato de que uma vez bem escolhido seu modo de atuar, garante que diversas alternativas serão exploradas, pois realiza um câmbio aleatório nos cromossomos. Os tipos mas conhecidos de mutação são: inversão e troca.

Os algoritmos genéticos diferem das clássicas técnicas de otimização, através das seguintes características:

- Usam parâmetros codificados e não os parâmetros próprios;
- Trabalham com uma população de pontos e não com um ponto só;
- Utiliza apenas os valores da função objetivo, não suas derivadas ou outro conhecimento auxiliar [Fogel, 1998];
- Usam funções de transição probabilísticas e não deterministas;
- Não ficam presos por mínimos locais;
- Se usarmos apenas os algoritmos genéticos, a convergência será excessivamente demorada;
- Em geral, os autores concluem que os algoritmos genéticos são uma poderosa ferramenta de busca, em problemas com pouca interação entre variáveis, pode convergir a uma solução muito perto do valor ideal, se não achar o ideal, muito rápido;
- Em problemas com alto nível de iteração entre variáveis, os algoritmos genéticos precisam ser adaptados adicionando algumas outras heurísticas, geralmente alguma que depende do domínio do problema [Weise, 2008].

2.4.4 Método Dialético de Busca e Otimização

Este método de otimização é inspirado na Filosofia, especificamente na Dialética Materialista, cujas origens estão conectadas às filosofias das antigas civilizações da Grécia, da China e da Índia. O automovimento de matéria no mundo perfaz-se de acordo com as leis da dialética. Essas leis afirmam que todo existente (**tese**) encerra em si *contradições* que impelem à luta e, consequentemente, ao desenvolvimento ou **evolução**[3]. A súbita mudança da tese é preparada por modificações quantitativas que podem ir-se produzindo, até um certo ponto máximo, natural para cada coisa [Brugger *et al*, 1972].

O Método Dialético de Otimização, foi introduzido no trabalho de Santos *et al.* [2009] como um método evolutivo baseado na dialética materialista para resolver problemas de busca e otimização, onde um problema de otimização está representado por um conjunto de pólos, sendo cada pólo a representação de uma solução candidata ao problema.

A busca de possíveis soluções está intrinsecamente relacionada com o movimento dos pólos e suas contradições, por isso, os pólos estão envolvidos em *lutas de pólos* e são afetados por *crises revolucionárias*, processos nos quais alguns pólos podem sumir ou ser absorvidos por outros. E assim, novos pólos podem surgir a partir de novos períodos de crise revolucionária.

Estes dois processos fazem o sistema ter uma tendência à estabilidade [Santos, 2009], e são similares às mutações na programação evolutiva e sua função é incluir diversidade ao processo de busca.

Definições Básicas da Dialética [Santos, 2009]

(a) *Pólo:* A unidade fundamental do sistema dialético que corresponde a uma solução candidata ao problema e está representado por um *vetor de condições.*
Dado um conjunto de pólos $\Omega = \{w_1, w_2, ..., w_m\}$, cada polo w_i é associado com um *vetor de pesos* definido como:

$$\mathbf{w}_i = (w_{i,1}, w_{1,2}, ..., w_{i,n})^T, \tag{2.17}$$

onde m é o número de pólos e n é a dimensionalidade do sistema.

[3]Entende-se por Evolução ao processo de conversão de mudanças quantitativas em qualitativas, segundo a Dialética Materialista [Cooper, 2002].

(b) *Força Social:* Associada à função objetivo do problema de otimização para cada pólo $\mathbf{w}_i$, representada como $f(\mathbf{w}_i)$. Esta é a ideia fundamental do método dialético.

(c) *Contradição:* A contradição entre dois pólos $\mathbf{w_p}$ e $\mathbf{w}_q$ é definida como:

$$\delta_{p,q} = dist(\mathbf{w}_p, \mathbf{w}_q)\,, \tag{2.18}$$

onde $dist(a, b)$ define uma medida de distância determinada. Neste caso foi escolhida a distância Euclidiana entre dois pontos a e b.

(d) *Hegemonia:* No processo da *luta de pólos*, o pólo $\mathbf{w}_i$ tem a *hegemonia* no instante t quando:

$$f_P(t) = f(\mathbf{w}_i(t)) = \max_{1\leq j\leq m} f(\mathbf{w}_j(t)), \tag{2.19}$$

onde $f_P(t)$ é chamada de *força hegemônica presente* no instante t. Igualmente, a *força hegemônica histórica* no instante t, $f_H(t)$, é definida como o máximo valor de força social obtido para cada fase histórica no instante t.

(e) *Antítese Absoluta:* Dado um pólo $\mathbf{w}_i$ tal que $a \leq \mathbf{w}_i \leq b$ onde $a, b \in \mathbb{R}$, o *oposto* ou *antítese absoluta* de $\mathbf{w}_i$ é definido segundo Shahryar *et al.* [2007] como:

$$\breve{\mathbf{w}}_i = b - \mathbf{w}_i + a\,. \tag{2.20}$$

Este conceito de antítese absoluta é muito importante na dialética pois expressa quantitativamente a máxima contradição entre dois pólos.

(f) *Síntese:* De acordo com o método dialético, a síntese de dois pólos é a resolução da contradição entre eles, "tese e antítese" [Vazquez, 2007], representada pela função $g(\mathbf{w}_p, \mathbf{w}_q)$. Como resultado é obtido um novo pólo com características herdadas de ambos pólos.
Sejam dois pólos $\mathbf{w}_p$ e $\mathbf{w}_q$. Uma forma de calcular as possíveis sínteses entre eles, representadas por $\mathbf{w}_u$ e $\mathbf{w}_v$, é apresentada como segue:

$$w_{u,i} \quad \begin{cases} w_{p,i}, & \text{se } (i \bmod 2) = 0, \\ w_{q,i}, & \text{caso contrário } (i \bmod 2) = 1; \end{cases} \tag{2.21}$$

$$w_{v,i} \quad \begin{cases} w_{p,i}, & \text{se } (i \bmod 2) = 1, \\ w_{q,i}, & \text{caso contrário } (i \bmod 2) = 0\,. \end{cases} \tag{2.22}$$

(g) *Fase Histórica:* Composta por duas fases, *Evolução* e *Crise Revolucionária*. Sendo esses processos os que correspondem com a dinâmica entre pólos através de avaliações das contradições.

- ***Evolução:*** É executada enquanto o número máximo de iterações (n_H) não for atingido, e enquanto a força hegemônica histórica for menor que o valor mínimo da função objetivo ($f_H(t) < f_{sup}$). Ambas forças hegemônicas são calculadas: *(i)* Força Hegemônica Presente (Equação 2.24); *(ii)* Força Hegemônica Histórica (Equação 2.25).

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \Delta\mathbf{w}_{P,i}(t) + \Delta\mathbf{w}_{H,i}(t), \tag{2.23}$$

para:

$$\Delta\mathbf{w}_{P,i}(t) = \eta_P(t)(1 - \mu_{P,i}(t))^2(\mathbf{w}_C(t) - \mathbf{w}_i(t)) \quad e \tag{2.24}$$

$$\Delta\mathbf{w}_{H,i}(t) = \eta_H(t)(1 - \mu_{H,i}(t))^2(\mathbf{w}_H(t) - \mathbf{w}_i(t)); \tag{2.25}$$

$$0 < \eta_P(t) < 1 \;\; e \tag{2.26}$$

$$0 < \eta_H(t) < 1; \tag{2.27}$$

onde $\Delta\mathbf{w}_{P,i}(t)$ e $\Delta\mathbf{w}_{H,i}(t)$ modelam as influências das hegemonias Presente e Histórica; η_C e η_H são os respectivos passos de atualização dos pólos.

Baseado nestes valores é feita uma atualização dos pólos atuais: *(i)* Ao final de cada iteração (Equação 2.28); *(ii)* Ao final de cada fase histórica (Equação 2.29).

$$\eta_P(t+1) = \alpha\eta_P(t) \quad e \tag{2.28}$$

$$\eta_H(t+1) = \alpha\eta_H(t)\,, \tag{2.29}$$

para $\alpha \leq 1$ (tipicamente, $\alpha = 0,9999$).

- ***Crise Revolucionária:*** É aplicado sobre os pólos; neste processo todas as contradições $\delta(\mathbf{w}_i)$ são calculadas e aqueles pólos cujas contradições sejam menores do que uma *mínima contradição* (δ_{min}), são fusionados:

$$\delta_{i,j}(t) > \delta_{min} \Rightarrow \mathbf{w}_i(t),\, \mathbf{w}_j(t) \in \Omega(t+1) \quad e \tag{2.30}$$

$$\delta_{i,j}(t) \leq \delta_{min} \Rightarrow \mathbf{w}_i(t) \in \Omega(t+1)\,, \tag{2.31}$$

para $i \neq j\, \forall i, j$ onde $1 \leq i, j \leq m(t)$, e $\Omega(t+1)$ o novo conjunto de pólos.

A partir das contradições previamente calculadas, a maior delas é considerada como a *contradição principal* do sistema (δ_{max}) e os pólos que a compõem serão chamados de tese-antítese:

$$\delta_{i,j}(t) > \delta_{max} \Rightarrow g(\mathbf{w}_i(t), \mathbf{w}_j(t)) \in \Omega(t+1)\,, \tag{2.32}$$

para $i \neq j$, $\forall i, j$ onde $1 \leq i, j \leq m(t)$.

Finalmente, dada uma *máxima crise* (χ_{max}), o *efeito de crise* (Equação 2.33) é adicionado a todos os pólos do sistema dialético $\Omega(t+1)$, gerando um novo conjunto de pólos $\Omega(t+2)$ de forma que $\mathbf{w}_k(t+2) \in \Omega(t+2)$, desde que:

$$w_{k,i}(t+2) = w_{k,i}(t+1) + \chi_{max}\, G(0,1)\,, \tag{2.33}$$

para $1 \leq k \leq m(t+1)$ e $1 \leq i \leq n$, onde $G(0,1)$ é um número aleatório de distribuição Gaussiana com esperança 0 e variância 1.

Caso algum critério de parada não tenha sido atingido, é gerado um novo conjunto de pólos, por meio da adição dos pólos em antítese antagônica aos pólos já existentes:

$$\mathbf{w}_i(t+2) \in \Omega(t+2) \Rightarrow \breve{\mathbf{w}}_i(t+2) \in \Omega(t+2)\,, \tag{2.34}$$

para $1 \leq i \leq m(t+2)$, onde $m(t+2) = 2m(t+1)$.

Embora o algoritmo dialético de otimização seja complexo na sua definição e com muitas iterações no processo (veja o fluxograma da Figura 2.8), possui a propriedade de ter um custo computacional decrescente [Santos e Assis, 2009], em cada fase histórica pois em cada uma delas o algoritmo está mais próximo de encontrar o valor mínimo, ou seja, as contradições entre os pólos diminuem, ficando mais perto uns dos outros permitindo que alguns pólos sejam eliminados, reduzindo assim o número de operações.

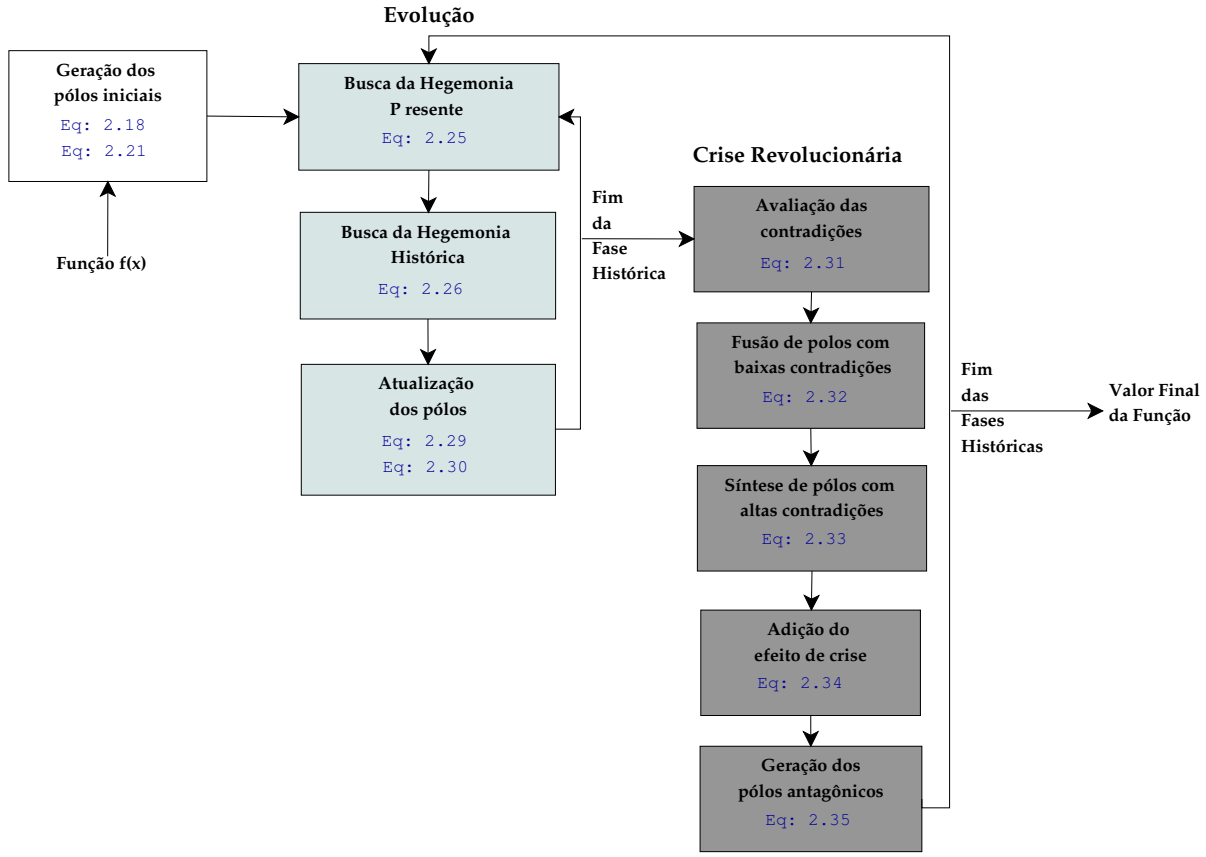


Figura 2.8: *Fluxograma do Método Dialético de Busca e Otimização. As duas etapas principais que o compõem: Evolução e Crise Revolucionária, estão ressaltadas em diferentes cores.*

Capítulo 3

Método Híbrido de Treinamento para o Classificador SLS

No aprendizado computacional supervisionado existem vários métodos de classificação, dentre eles podemos destacar um classificador apresentado nos trabalhos de Ribeiro e Hashimoto [2008, 2010], conhecido como classificador SLS (*Straight Line Segments*), baseado na distância do ponto a ser classificado a conjuntos de segmentos de reta, que ainda não foi completamente explorado e é objeto de estudo desde trabalho. Este classificador faz uso do método de otimização de Gradiente Descendente na fase de treinamento, o qual tem a vantagem de convergir a um ponto mínimo rapidamente, mas como todo algoritmo de otimização também pode atingir ao invés de um mínimo global, um mínimo local.

A ideia principal que propomos neste trabalho é combinar a rapidez do método do Gradiente Descendente para encontrar um mínimo local e a diversidade de soluções que pode ser proporcionada pela aplicação de Algoritmos Evolutivos, a fim de melhorar a acurácia do classificador SLS. Dessa forma, os Algoritmos Evolutivos procuram pela região do mínimo global e depois é trabalho do Gradiente Descendente procurar pelo valor ótimo [Heistermann, 1992]. Assim, a partir de pontos iniciais melhorados, a busca converge rapidamente.

Neste capítulo, descrevemos as diferentes combinações, entre Algoritmos Evolutivos e Gradiente Descendente, que foram realizadas a fim de encontrar o melhor método híbrido de treinamento que permita melhorar a acurácia do classificador SLS. É interessante notar que o *tempo computacional*, não será um fator decisivo na escolha da melhor combinação de algoritmos de otimização, pois sabemos que os algoritmos evolutivos têm uma tendência a incrementar o custo computacional devido as múltiplas iterações no seu projeto.

3.1 Considerações Iniciais

Usando como motivação o Teorema de *"No Free Lunch"* para *Aprendizado Computacional Supervisionado*, apresentado e provado nos trabalhos de Wolpert [1996, 2001], nos quais foi mostrado que em um cenário livre de ruído onde a função de perda é representada pelo erro de classificação, em termos do erro de teste (o erro de classificação dos elementos não presentes no conjunto de treinamento), não existe uma distinção entre algoritmos de aprendizagem supervisionados, quando comparados os desempenhos médios.

Formalmente, temos:

- S é o conjunto de treinamento;
- m é o número de exemplos no conjunto de treinamento, ou seja $m = |S|$;

- f é a função de classificação;
- h é a hipótese (conjunto de parâmetros para obter a função de classificação f para S) e
- C é o valor de perda do conjunto de teste, associado a f e h (erro de generalização).

Todos os algoritmos são equivalentes, em média, quando são comparados usando qualquer uma das seguintes medidas de risco: $\mathcal{E}(C|S)$, $\mathcal{E}(C|m)$, $\mathcal{E}(C|f,S)$ ou $\mathcal{E}(C|f,m)$. [Wolpert, 1996]

Em Wolpert [1996, 2001], foi mostrado também, que todos os algoritmos que procuram por um valor extremo para uma determinada função de custo, têm um desempenho exatamente igual, quando é calculada a média para todas as possíveis funções de custo. Em particular, se o algoritmo A supera ao algoritmo B em algumas funções de custo, então devem existir muitas outras funções de custo onde B supera a A. Assim, podemos dizer que nossa proposta de melhorar um classificador já existente é válida pois poderá haver situações em que este classificador terá um bom desempenho.

Dadas duas características, a primeira correspondente aos Algoritmos Evolutivos, que é sua capacidade de escapar de um mínimo local pela busca estocástica multiponto [Xuefeng *et al*, 2009]; e a segunda que é própria do Gradiente Descendente, e refere-se à habilidade para encontrar *ótimos locais* apontando na direção que maximiza ou minimiza a função objetivo, temos o intuito de combiná-las para melhorar a qualidade da solução do problema de otimização.

Sabemos que um classificador SLS possui duas fases: treinamento e teste. Como nosso objetivo está centrado na fase de treinamento, dado um conjunto de dados, serão executados os seguintes passos:

1. Gerar soluções iniciais (função própria de cada *algoritmo evolutivo*);
2. Aplicar o método de *gradiente descendente* em cada solução inicial;
3. Aplicar os processos de recombinação do *algoritmo evolutivo* desejado;
4. Se atingir o fim das iterações, aplicar mais uma vez o *gradiente descendente*, para garantir um valor mínimo no final da execução do algoritmo.
5. Caso contrário, repetir os passos 2 e 3.
6. Devolver os dois conjuntos de segmentos de reta, $\mathcal{L}_0$ e $\mathcal{L}_1$.

No fluxograma da Figura 3.1, apresentamos um resumo da ideia geral desta primeira proposta, a ser descrita neste capítulo; onde se pode observar que o objetivo principal da combinação de diferentes tipos de algoritmos de otimização, é proporcionar ao *Gradiente Descendente* um novo conjunto de posições iniciais, correspondentes à saída dos *Algoritmos Evolutivos*, evitando assim parar em um ótimo local, a fim de otimizar a posição dos conjuntos de segmentos de reta: $\mathcal{L}_0$ e $\mathcal{L}_1$.

Assim, lembrando o problema de otimização do classificador SLS, temos as seguintes definições:

- **Objetivo:** Encontrar dois conjuntos de segmentos de reta ($\mathcal{L}_0$ e $\mathcal{L}_1$), que representem as classes $\{0,1\}$ e minimizem uma função.
- **Função de Classificação:** apresentada no Capítulo 2, esta função é diferenciável, por isso é possível aplicar o gradiente descendente.

$$y_{\mathcal{L}_0,\mathcal{L}_1}(x) = \frac{1}{1+e^{-g(T_{\mathcal{L}_0,\mathcal{L}_1}(x))}} \tag{3.1}$$

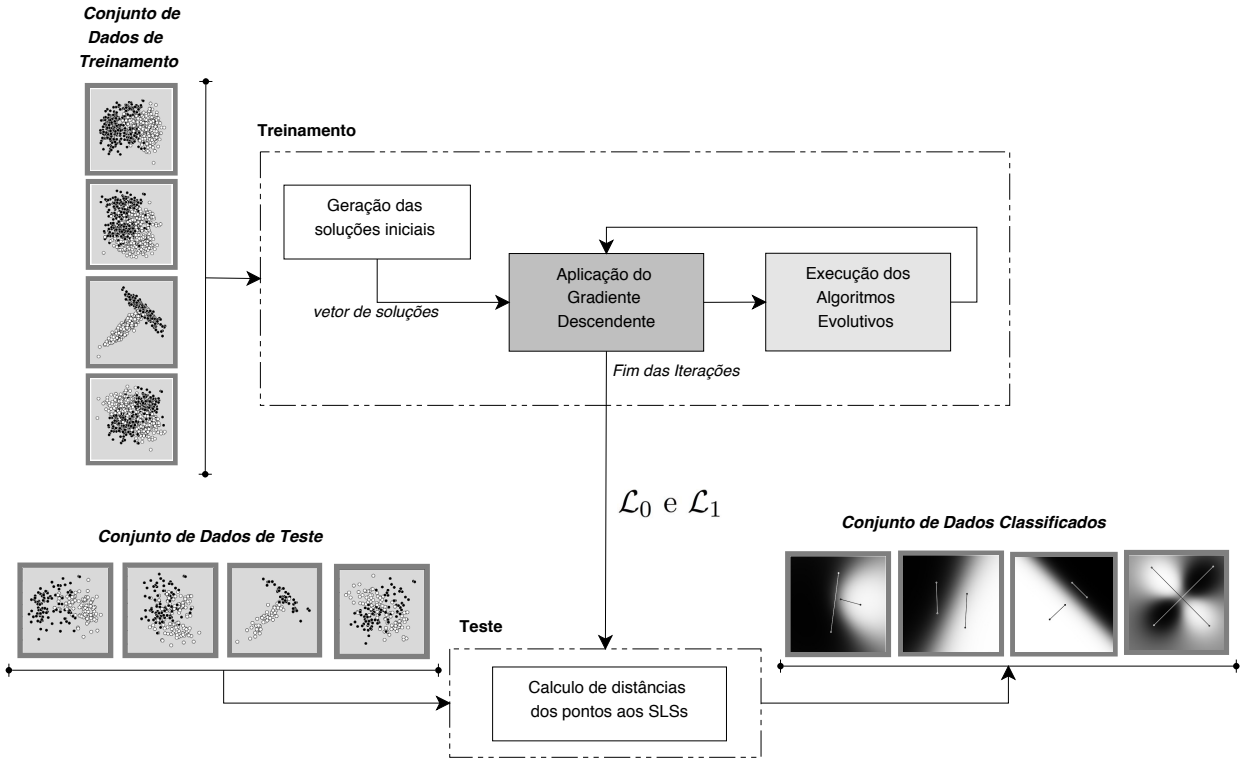


Figura 3.1: *Fluxograma Geral das Combinações de Algoritmos Evolutivos com Gradiente Descendente. Neste fluxograma estão representadas as duas fases do classificador SLS, o treinamento e a fase de teste.*

- **Função a Minimizar:** O Erro Quadrático Médio,

$$MSE(y_{\mathcal{L}_0,\mathcal{L}_1}) = \frac{1}{N}\sum_{i=1}^{N}[err_i(y_{\mathcal{L}_0,\mathcal{L}_1})]^2 \tag{3.2}$$

onde

$$err_i(y_{\mathcal{L}_0,\mathcal{L}_1}) = y_{\mathcal{L}_0,\mathcal{L}_1}(x_i) - y_i, \tag{3.3}$$

e N representa o número de amostras no conjunto de treinamento.

Nas próximas seções, serão descritas três combinações do método Gradiente Descendente com diferentes algoritmos de otimização: (i) *K-Beams*, que é um algoritmo de busca local; (ii) Algoritmos Genéticos e (iii) Método Dialético de Otimização.

3.2 *K-Beams* - Gradiente Descendente

Começamos descrevendo esta combinação de algoritmos por ser o algoritmo *K-Beams* o mais simples na implementação e definição. Como foi descrito nas considerações iniciais, a ideia é ajudar ao método do Gradiente Descendente a escapar dos mínimos locais. Com esse objetivo, utilizaremos este algoritmo para gerar novos estados, os quais representam um conjunto de soluções iniciais a ser melhoradas pelo gradiente descendente.

No Algoritmo 1, listado a seguir, é resumida a fase de treinamento do classificador SLS, utilizando a combinação (**KBM+GD**). As variáveis utilizadas são: S, o conjunto de dados de treinamento; $\mathcal{P}$, um conjunto de parâmetros que contém: (i) $estados_{max}$ que é o número de estados que serão gerados aleatoriamente; (ii) $sucessores_{max}$ representa o número de sucessores para cada estado e (iii) $iterações_{max}$ representa o número de iterações do laço inicial. E as variáveis **estados** e **valores** são

vetores que armazenam as soluções iniciais e os valores resultantes da avaliação da função objetivo para cada estado, respectivamente.

A função GERAR-ESTADOS gera os estados iniciais aleatoriamente e GERAR-SUCESSORES é uma heurística para gerar novas soluções a partir do ponto inicial; também são definidas, as funções CONVERTER-A-SLSS e CONVERTER-A-ESTADOS que permitem a conversão entre o vetor de estados e os conjuntos de segmentos de reta. A função ENCONTRAR-MIN-SUCESSOR permite encontrar o sucessor com o valor mínimo após a avaliação da função objetivo e ATUALIZAR-ESTADOS define como novos estados iniciais aos melhores sucessores, sendo nesta função onde acontece o "intercâmbio de informação" entre estados.

Algoritmo 1 TREINAMENTO-KBM-GD$(S, \mathcal{P})$

Entrada: S, $\mathcal{P} = \{estados_{max},\ sucessores_{max},\ iterações_{max}\}$
Saída: $SLSs = \{\mathcal{L}_0 \, e \, \mathcal{L}_1\}$

1: **estados** ← GERAR-ESTADOS$(S, estados_{max})$
2: $melhor_{valor} \leftarrow 0$
3: **para** $it = 0$ **até** $iterações_{max}$ **faça:**
4: **estados** ← GERAR-SUCESSORES(**estados**, $sucessores_{max}$)
5: **SLSs** ← CONVERTER-A-SLSS(**estados**)
6: **SLSs** ← GRADIENTE-DESCENDENTE(**SLSs**)
7: **estados** ← CONVERTER-A-ESTADOS(**SLSs**)
8: **valores** ← AVALIAR-FUNÇÃO-OBJETIVO$(S,$ **SLSs**$)$
9: $\{melhor_{valor}, estado_{pos}\}$ ← ENCONTRAR-MÍNIMO-SUCESSOR(**valores**)
10: **estados** ← ATUALIZAR-ESTADOS$(estado_{pos},$ **estados**$)$
11: **end para**
12: $SLSs$ ← CONVERTER-A-SLSS(**estados**)
13: $SLSs$ ← GRADIENTE-DESCENDENTE$(SLSs)$
14: **return** $\{SLSs\,,\, melhor_{valor}\}$

Uma representação do algoritmo híbrido de treinamento (**KBM+GD**), pode-se observar no fluxograma da Figura 3.2, onde é exibida a ideia geral da combinação destes algoritmos: após gerar os estados iniciais e seus respetivos sucessores, imediatamente é aplicado o gradiente descendente em cada um, para logo avaliar a função objetivo e voltar ao laço inicial o qual depende do número de iterações definido no inicio do algoritmo.

Adaptação dos conceitos de *K-Beams*

A fim de adaptar a definição de dois conjuntos de segmentos de reta, que representam uma solução para nosso problema, foram modificadas algumas definições do algoritmo *K-Beams* (veja a Seção 2.4.2), e serão descritas a seguir:

- *Estado:* Representa uma possível solução ao problema, nesse caso, representa uma possível configuração de dois conjuntos de segmentos de reta: $\mathcal{L}_0$ e $\mathcal{L}_1$, para nosso problema. Definido pelo vetor:

$$\textbf{estado} = [SLSs_{0,1},\ ...\, SLSs_{0,a}\ ,\ SLSs_{1,1},\ ...\, SLSs_{1,b}] \tag{3.4}$$

  onde, a e b são os números de segmentos de reta para a classe 0 e 1, respectivamente.

- *Sucessor:* Os sucessores são definidos baseados em diferentes heurísticas, neste caso poderíamos usar a estratégia mais simples. Dado um ponto no espaço $\mathbb{R}^2$, para poder explorar algumas possíveis direções, temos 2^2 opções: Norte, Sul, Leste e Oeste.

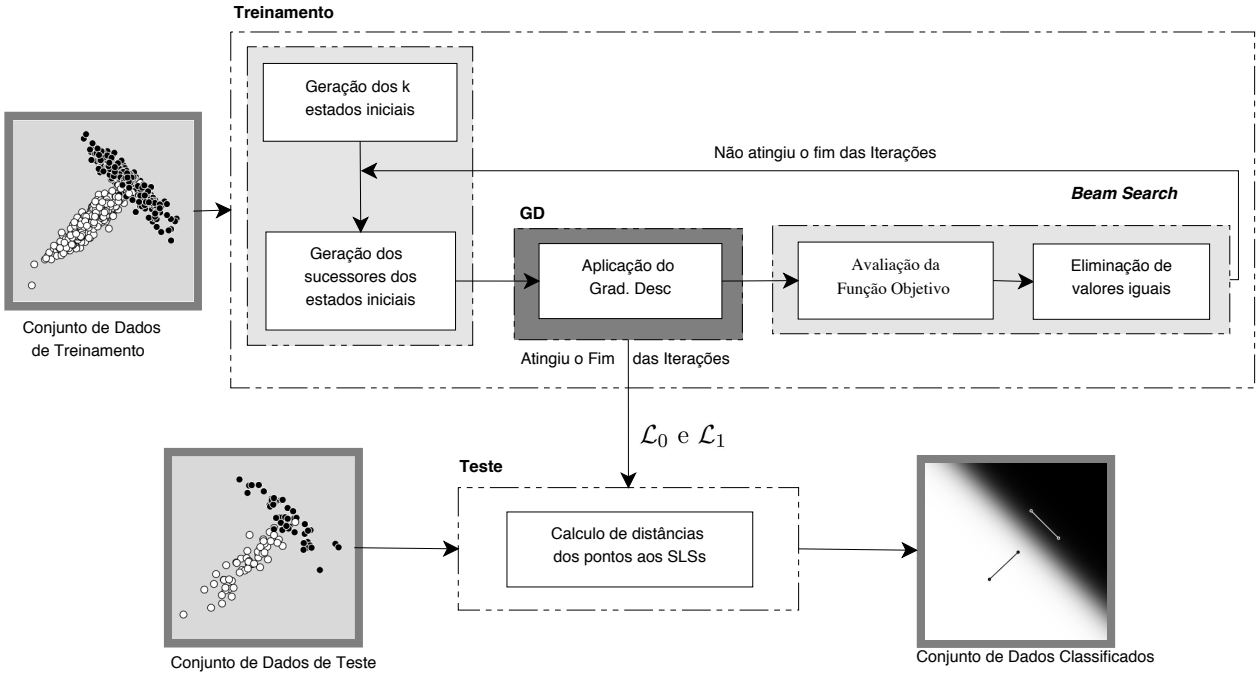


Figura 3.2: *Diagrama de fluxo de dados da combinação* ***KBM+GD****. Neste fluxograma estão representadas as duas fases do classificador SLS, o treinamento e a fase de teste.*

Essa heurística, por exemplo, teria um bom funcionamento para nosso problema, se os dados de treinamento estivessem no espaço $\mathbb{R}^1$. Pois, se lembrarmos a teoria do Classificador SLS, os pontos p e q que representam um segmento de reta são estendidos em uma dimensão, ou seja: $p_e = (x, y)$ e $q_e = (x, y)$. Nesse caso, para explorar as direções mencionadas acima, devemos adicionar a cada estado, cada um dos vetores da Tabela 3.1 que representam as 2^2 direções mais o ponto inicial.

No contexto dos segmentos de reta e de dados em dimensões maiores que um, essa heurística não proporcionaria uma solução boa e eficiente pois, se os dados estão no espaço $\mathbb{R}^{12}$ então os segmentos de reta estariam no $\mathbb{R}^{13}$. Assim, teríamos 2^{13} possíveis direções.

Dado esse problema, decidimos utilizar outra heurística para gerar os sucessores. Neste caso, os sucessores serão gerados seguindo uma estratégia aleatória, baseados em uma distribuição gaussiana ou em uma distribuição uniforme, que permita também obter valores negativos e simular diferentes "direções", definida como:

$$sucessor = estado + \delta \tag{3.5}$$

onde δ é o deslocamento, obtido a partir de um número aleatório, de tal forma que *sucessor* ficará dentro do intervalo $< +\delta, -\delta >$.

Tabela 3.1: *Tabela de vetores que representam as* 5 *direções a serem exploradas pelas soluções.*

$[\ 0\ ,\ +\delta\ ]$	Deslocamento-Norte
$[\ 0\ ,\ -\delta\ ]$	Deslocamento-Sul
$[\ 0\ ,\ 0\ ]$	Sem deslocamento
$[\ +\delta\ ,\ 0\ ]$	Deslocamento-Leste
$[\ -\delta\ ,\ 0\ ]$	Deslocamento-Oeste

3.3 Algoritmos Genéticos - Gradiente Descendente

Nesta seção é descrita a combinação do gradiente descendente com os Algoritmos Genéticos, os quais podem ser considerados uma variação do Algoritmo *K-Beams*; mantemos sempre a ideia geral, a qual é proporcionar novos pontos iniciais para o gradiente descendente.

Na implementação dos algoritmos genéticos, um cromossomo ou indivíduo é uma representação codificada (normalmente em uma seqüência de zeros e uns, $\{1\,0\,0\,1\,1\,0\,1\}$) de uma possível solução ao problema de otimização. Uma população é um conjunto de indivíduos, ou seja, ela contém possíveis soluções do problema. A população inicial é gerada aleatoriamente, podendo ser incorporado conhecimento a priori a respeito do problema (por heurísticas) para gerar potenciais soluções iniciais ao mesmo (veja a Seção 2.4.3).

No Algoritmo 2, é descrito o processo deste algoritmo híbrido a ser aplicado na fase de treinamento do classificador SLS. Onde S, representa o conjunto de dados treinamento e $\mathcal{P}$ um conjunto de parâmetros. Os principais parâmetros são: $população_{max}$ que define o número de cromossomos que formará parte da população (gerada pela função GERAR-POPULAÇÃO); e $gerações_{max}$ que representa o número de iterações do algoritmo.

As funções: CONVERTER-A-SLSS e CONVERTER-A-POPULAÇÃO permitem mudar de vetores de soluções (população) para conjuntos de segmentos de reta; AVALIAR-FUNÇÃO-OBJETIVO calcula o valor do Erro Quadrático Médio para cada cromossomo; e finalmente RECOMBINAÇÃO que realiza os processos de cruzamento e mutação. Nos vetores **população** e **valores** são armazenadas as soluções iniciais e os valores resultantes da avaliação da função objetivo para cada cromossomo, respectivamente.

Algoritmo 2 TREINAMENTO-AG-GD$(S, \mathcal{P})$

Entrada: S, $\mathcal{P} = \{\ população_{max},\quad gerações_{max}\}$
Saída: $SLSs = \{\mathcal{L}_0\, e\, \mathcal{L}_1\}$

1: **população**← GERAR-POPULAÇÃO(S, pop_{max})
2: $melhor_{valor} \leftarrow 0$
3: **para** $geração = 0$ **até** $gerações_{max}$ **faça:**
4: **SLSs** ← CONVERTER-A-SLSS(**população**)
5: **SLSs** ← GRADIENTE-DESCENDENTE(**SLSs**)
6: **população** ← CONVERTER-A-POPULAÇÃO(**SLSs**)
7: **valores** ← AVALIAR-FUNÇÃO-OBJETIVO$(S,$ **SLSs**$)$
8: { **população**, $melhor_{valor}$} ← RECOMBINAÇÃO()
9: **end para**
10: $SLSs$ ← CONVERTER-A-SLSS(**população**)
11: $SLSs$ ← GRADIENTE-DESCENDENTE$(SLSs)$
12: **return** $\{SLSs\,,\, melhor_{valor}\}$

Veja na Figura 3.3 o fluxograma do algoritmo híbrido de treinamento proposto nesta seção, o qual está composto por um laço de execução o qual depende do conjunto de parâmetros iniciais. O processo pode-se resumir como: Geração da população (conjunto de cromossomos representando possíveis soluções do problema); aplicação do gradiente descendente em cada solução; e finalmente a recombinação de cromossomos, processos responsáveis por adicionar diversidade às soluções prévias obtidas pelos Algoritmos Genéticos.

Adaptação dos conceitos de Algoritmos Genéticos

Dadas as definições sobre este algoritmo de otimização, apresentadas na Seção 2.4.3, para

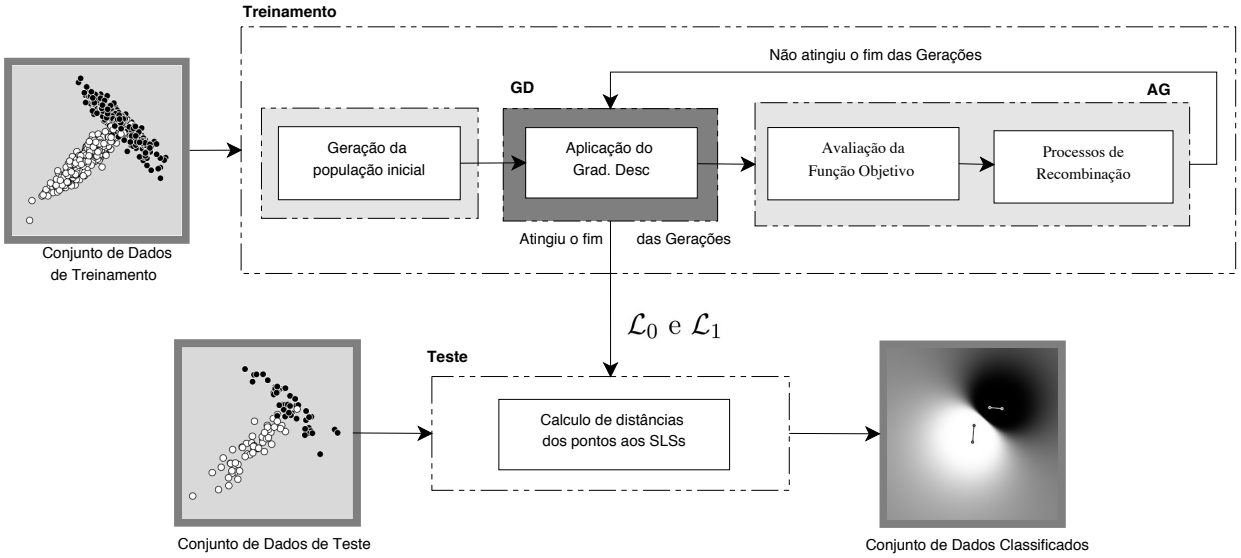


Figura 3.3: *Diagrama de fluxo de dados da combinação **AG+GD**. Neste fluxograma estão representadas as duas fases do classificador SLS, o treinamento e a fase de teste.*

serem aplicados no nosso problema foram realizadas certas modificações, descritas a seguir:

- *Cromossomo:* Para nosso problema vamos considerar como indivíduo um vetor real α de dimensão k. Dadas duas coleções de segmentos de retas $\mathcal{L}_0$ e $\mathcal{L}_1$ representando às classes $\{0, 1\}$, um cromossomo é formado pela composição das extremidades de cada segmento de reta.

- *População Inicial:* Consiste em um conjunto de m vetores $\{\alpha_1, \alpha_2, ..., \alpha_m\}$, onde cada vetor α_i representa uma possível solução de uma configuração dos segmentos de reta distribuídos nos conjuntos $\mathcal{L}_0$ e $\mathcal{L}_1$; e m é um valor definido no início do algoritmo.

- *Mutação:* Neste processo, dado um cromossomo, a mutação dele está definida pela substituição das extremidades dos segmentos de reta desse cromossomo, por valores aleatórios extraídos da população inicial.

3.4 Método Dialético de Otimização - Gradiente Descendente

Por último, nesta seção apresentamos o método híbrido utilizando o Método Dialético de Otimização (MDO) para tentar escapar do ótimo local, e assim encontrar um outro valor (ainda melhor do que o anterior) o qual pode-se tornar em um *ótimo global*. Deste modo, no momento que o gradiente descendente para em um mínimo local, o MDO proporciona um novo conjunto de pontos de partida ajudando-o a fugir dessa região de mínimos locais.

Resumindo o processo do método híbrido **MDO+GD** [Medina e Hashimoto, 2011], apresentamos o Algoritmo 3, o qual pode ajudar no entendimento do método proposto. Onde $\mathcal{P}$, representa um conjunto de parâmetros: m é o número de pólos; max_{fh} o máximo número de fases históricas; max_{it} define o número de iterações, α é o valor usado para a atualização dos pólos; e f_p é o valor mínimo que pode alcançar a função objetivo. As funções Gerar-Polos-Iniciais, Crise-Revolucionária e Evolução são descritas a detalhe na Seção 2.4.4.

Como pode-se observar no fluxograma da Figura 3.4, os passos principais que compõem a ideia do método híbrido são: primeiro, gerar a população inicial como é proposto pelo MDO; em seguida, para cada fase histórica no MDO aplicar o método de gradiente descendente em

Algoritmo 3 TREINAMENTO-MDO-GD$(S, \mathcal{P})$

Entrada: S, $\mathcal{P} = \{m,\ max_{fh},\ max_{it},\ \alpha,\ f_p\}$
Saída: $SLSs = \{\mathcal{L}_0\, e\, \mathcal{L}_1\}$

```
w ← GERAR-POLOS-INICIAIS(S, m)
n_fh ← 0
melhor_valor ← 0
while (n_hf < max_fh) e (best_value < f_p) faça:
    SLSs ← CONVERTER-A-SLSS(w)
    SLSs ← GRADIENTE-DESCENDENTE(SLSs)
    w ← CONVERTER-A-PÓLOS(SLSs)
    para it = 0 até max_it faça:
        w ← EVOLUÇÃO(w, α)
    end para
    {w, melhor_valor} ← CRISE-REVOLUCIONÁRIA(w)
    n_fh ← n_fh + 1
end while
SLSs ← CONVERTER-A-SLSS(w)
SLSs ← GRADIENTE-DESCENDENTE(SLSs)
return {SLSs , melhor_valor}
```

cada pólo da população a fim de obter um *ótimo local* para cada um; finalmente, executar os seguintes passos do MDO (evolução, crise revolucionária, etc.)

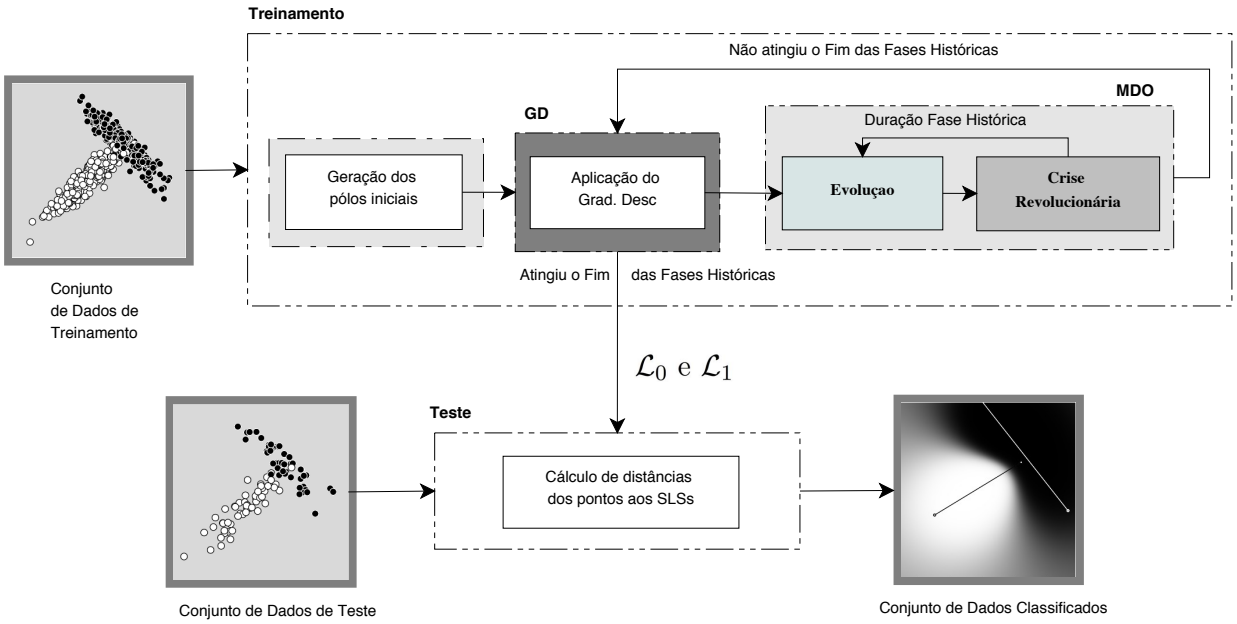


Figura 3.4: *Diagrama de fluxo de dados da combinação* ***MDO+GD****. Neste fluxograma estão representadas as duas fases do classificador SLS, o treinamento e a fase de teste.*

Adaptação dos conceitos do Método Dialético de Otimização

Algumas definições do método de otimização dialética, foram adaptadas ao domínio de nosso problema a fim de encontrar as melhores posições dos SLSs em $\mathcal{L}_0$ e $\mathcal{L}_1$, e são descritas a seguir:

- *Pólo:* Como um pólo corresponde a uma possível solução, então definimos um pólo

como um vetor composto pelas extremidades dos SLSs pertencentes a $\mathcal{L}_0$ e $\mathcal{L}_1$. Deste modo, podemos representar um pólo como a concatenação de $\mathcal{L}_0$ com $\mathcal{L}_1$. Por exemplo: $[\mathcal{L}_0|\mathcal{L}_1]$.

$$\mathbf{w}_i = (SLSs_{0,1},\ ...\ SLSs_{0,a}\ ,\ SLSs_{1,1},\ ...\ SLSs_{1,b}) \tag{3.6}$$

onde, a e b representam o número de segmentos de reta para a classe 0 e 1, respectivamente.

- *Antítese:* Visto que nosso problema faz uso de conjuntos de segmentos de reta, que dividem duas classes de pontos, o pólo oposto ou *antítese* é equivalente a inverter as posições dos segmentos de reta de cada classe, o que significa que os segmentos em $\mathcal{L}_0$ e $\mathcal{L}_1$, representarão a classe 1 e 0, respectivamente.

  Assim, a função antítese representada pela Equação 2.20, foi redefinida de modo que, dado um pólo $[\mathcal{L}_0|\mathcal{L}_1]$ a antítese seja $([\mathcal{L}_1|\mathcal{L}_0])$:

$$\breve{\mathbf{w}}_i = (SLSs_{i,1},\ ...\ SLSs_{i,\mathbf{b}}\ ,\ SLSs_{i,1},\ ...\ SLSs_{i,\mathbf{a}}) \tag{3.7}$$

- *População Inicial:* A população inicial é gerada da seguinte forma: metade da população inicial é gerada aleatoriamente, com dados contidos dentro do intervalo composto pelo mínimo e máximo valor do conjunto de dados de treinamento. Enquanto que a outra metade da população inicial é gerada a partir dos pólos opostos (pólos antítese) da primeira metade.

$$\Omega = \{\mathbf{w}_1, \mathbf{w}_2,\ ...\ \mathbf{w}_{\frac{m}{2}},\ \breve{\mathbf{w}}_{\frac{m}{2}+1},\ ...\ \breve{\mathbf{w}}_m\} \tag{3.8}$$

- *Força Social:* A força social será representada pela função a minimizar descrita nas considerações iniciais (Equação 3.2), ou seja o Erro Quadrático Médio

- *Contradição:* A contradição entre dois pólos foi definida na Equação 2.18 como a distância Euclidiana entre eles. Em nosso trabalho foi redefinida como o valor absoluto da diferença entre o Erro Quadrático Médio (MSE) obtido para cada pólo envolvido ($\mathbf{w}_p$ e $\mathbf{w}_q$), representada pela Equação 3.9. Portanto, a contradição significa a diferença dos erros de classificação entre dois pólos.

$$\delta_{p,q} = |MSE_{\mathbf{w}_p}(\mathcal{L}_0, \mathcal{L}_1) - MSE_{\mathbf{w}_q}(\mathcal{L}_0, \mathcal{L}_1)| \tag{3.9}$$

No Capítulo 5 apresentamos os resultados obtidos aplicando cada um destes métodos híbridos na fase de treinamento do Classificador SLS, analisando o desempenho do classificador nos baseando na taxa de acerto na classificação, quando é aplicado sobre distribuições de dados conhecidas.

Capítulo 4

Estimando o Número de Segmentos de Reta

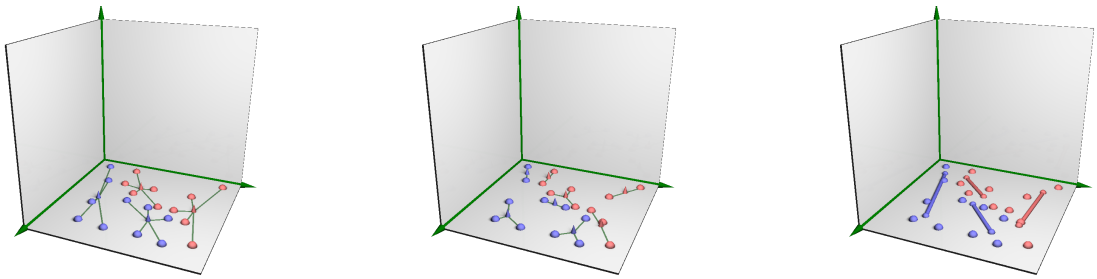
Como foi descrito anteriormente, o algoritmo *K-Means* precisa de um valor K, que representa o número de agrupamentos a serem encontrados, este valor deve ser definido no início do algoritmo. Para escolher este valor precisamos de alguma informação prévia sobre o conjunto de dados; e segundo Vatsavaia *et al.* [2011], este algoritmo tem a desvantagem de ser muito influenciável por diferentes configurações iniciais. Assim, quando os dados têm muitas dimensões torna-se um problema difícil, mesmo quando os agrupamentos encontram-se bem separados [Hamerly e Elkan, 2003].

Visto o problema de encontrar o valor de K e, dado que em nosso problema a taxa de acerto do classificador depende das posições iniciais dos segmentos de reta, encontradas na fase de *Placing* (veja a Seção 2.2.3), onde é usado o algoritmo de *K-Means* com um valor de K predefinido. Decidimos usar um algoritmo de agrupamento que nos proporcione o número de *"clusters"* que existem em cada classe, a fim de estimar o número de segmentos de reta a ser usado. Pois, ao limitar o número de segmentos de reta, mantemos a complexidade computacional baixa e evitamos o problema de sobreposição [Ribeiro e Hashimoto, 2010].

Os algoritmos de agrupamento desempenham um papel fundamental na mineração de dados e em muitas áreas da ciência computacional. Estes algoritmos são comumente usados para analisar grandes, multivariados e complexos volumes de dados; como um primeiro passo na obtenção de conhecimentos sobre a estrutura ou os agrupamentos naturais dentro dos dados. Deste modo, determinar o número de agrupamentos é um dos principais desafios dos algoritmos de agrupamento [Vatsavaia *et al*, 2011].

Na fase de treinamento do Classificador SLS, o número de segmentos de reta que representam cada classe, além de ser igual para $\mathcal{L}_0$ e $\mathcal{L}_1$, é definido no início do algoritmo. Este número de segmentos de reta tem influência na taxa de acerto do classificador SLS, apesar dos bons resultados obtidos no trabalho de Ribeiro e Hashimoto [2010], onde o número de segmentos necessários para representar as diferentes distribuições foi igual para cada classe; consideramos que explorar o classificador SLS quando o número de segmentos é diferente, pode fazer com que o resultado tenha uma tendência a melhorar.

Assim, neste capítulo, vamos explorar qual é o comportamento e o desempenho do classificador SLS quando o número de segmentos de reta, que representam cada classe, é diferente. A segunda ideia que propomos neste capítulo, consiste em evitar a definição a priori do número de segmentos de reta a serem usados na configuração do algoritmo de treinamento.



(a) (b) (c)

Figura 4.1: *Os diferentes passos a seguir na fase de Pre-Alocação de Segmentos de Reta.*

Para atingir este objetivo, decidimos usar o algoritmo de agrupamento *X-means*, descrito no Capítulo 2 e apresentado por Pelleg e Moore [2000], que encontra automaticamente o valor de K, otimizando um determinado critério de informação como: *Akaike Information Criterion* (AIC) ou *Bayesian Information Criterion* (BIC) [Jain, 2010].

4.1 Considerações Iniciais

Sabemos que, o algoritmo de treinamento do classificador SLS está composto por duas fases: (i) Pre-alocação de segmentos de reta ou *Placing* e (ii) Ajuste dos segmentos de reta ou *Tunning*. Neste capítulo vamos nos focar na primeira fase (*Placing*), a qual tem por objetivo colocar os segmentos de reta em uma posição inicial, de maneira que os segmentos de reta fiquem próximos a locais que minimizem o risco empírico [Ribeiro, 2009].

No algoritmo original do classificador SLS, os passos a seguir na fase de Pre-alocação, são os seguintes:

(a) O conjunto de amostras é dividido em dois conjuntos, as que pertencem à classe 0 e as que pertencem à classe 1. Depois é aplicado *K-Means* com o valor de K igual ao número de segmentos de reta com os quais vão se representar às classes ($nSLS$). Por exemplo, na Figura 4.1.a, $K = 2$.

(b) Em seguida, *K-Means* é aplicado pela segunda vez, agora em cada agrupamento encontrado no item anterior com $K = 2$ (veja a Figura 4.1.b), pois cada centróide dos novos agrupamentos, representará uma extremidade de um segmento de reta, conforme a Figura 4.1.c.

Uma vez descrito o algoritmo para a Pre-Alocação de Segmentos de Reta, as principais definições a serem usadas neste capítulo, são as seguintes:

- *Número de Segmentos de Reta (nSLS):* valor que determina quantos segmentos de reta representarão cada classe, ou seja, o número de elementos do conjunto $\mathcal{L}_0$ e $\mathcal{L}_1$. Neste caso, vamos usar $1, 2, 3$ e 4 segmentos de reta para cada classe.

- *Conjunto de Dados Artificiais:* Usamos conjuntos de dados desenhados para quatro funções de densidade de probabilidade, descritas no trabalho de Ribeiro e Hashimoto [2008], apresentadas na Figura 4.2 e chamadas de F, S, Simples e X, respectivamente. Estes conjuntos de dados estão compostos de várias regiões onde cada uma delas tem uma função de densidade de probabilidade uniforme. Esta propriedade torna possível a aplicação do classificador de Bayes [Duda *et al*, 2001], para obter a taxa ideal de classificação.

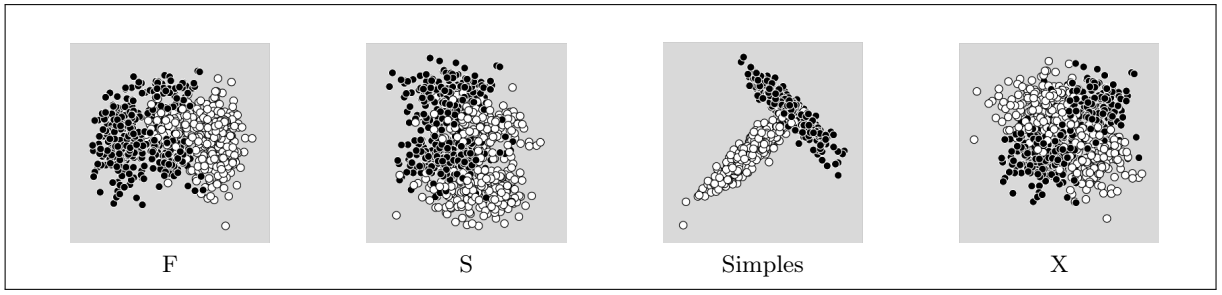


Figura 4.2: *Distribuições de Dados Artificiais.*

- *Algoritmo de Otimização:* Vamos usar como referência o algoritmo de Treinamento original do classificador SLS, ou seja, para realizar o ajuste das posições dos segmentos de reta, será usado o método de Gradiente Descendente. Cabe ressaltar que todas as figuras e porcentagens, neste capítulo, pertencem aos resultados obtidos usando este método.
- *Número de Exemplos por Amostra:* O tamanho das amostras é de 800 exemplos, e os erros de classificação serão comparados com o acerto de Bayes, para cada distribuição.

Nas próximas seções são descritas as duas alternativas propostas a fim de procurar pelo número "ótimo" de segmentos de reta.

4.2 Busca Exaustiva do Número de Segmentos de Reta

Dada uma determinada distribuição de dados (F, S, Simples e X) e as quatro possíveis opções de segmentos de reta que foram definidas nas considerações iniciais, $nSLS = \{1, 2, 3, 4\}$, será realizada uma busca exaustiva do número ótimo de segmentos de reta, que representarão às classes. Com este objetivo, vamos explorar cada uma das possíveis combinações dos elementos do conjunto $nSLS$, representadas na Tabela 4.1. Pode-se observar que o número de treinamentos a ser realizados para conhecer qual foi a melhor combinação de segmentos de reta a serem usados para cada classe é $2^4 = 16$ treinamentos.

Tabela 4.1: *Possíveis Combinações de Segmentos de Reta para cada classe.*

NSLS	1	2	3	4
1	**1-1**	1-2	1-3	1-4
2	2-1	**2-2**	2-3	2-4
3	3-1	3-2	**3-3**	3-4
4	4-1	4-2	4-3	**4-4**

Deste modo, conforme as Figuras 4.3 e 4.4, onde mostramos os agrupamentos encontrados pelo algoritmo *K-Means* na fase de treinamento do classificador SLS, usando o Gradiente Descendente, para cada uma das combinações apresentadas na tabela acima. As figuras pertencentes a cada distribuição artificial, estão ordenadas conforme a tabela de combinações 4.1, onde o número de agrupamentos da classe 0 (cor rosa) vai se incrementando em cada linha; e o número de agrupamentos da classe 1, representada pela cor verde, se incrementa em cada coluna. Como pode-se observar nas figuras, algumas vezes não é necessário usar o mesmo número de segmentos de reta por classe para obter uma boa representação dos dados. Veja a Distribuição F por exemplo (Fig. 4.3.a), onde as configurações **2-3** ou **3-2** e **2-2** ou **3-3**, destacadas em requadro, dão a impressão de serem as configurações que melhor representam os dados.

Para a Distribuição S, estimar o número de segmentos de reta foi mais difícil, como pode-se observar na Fig. 4.3(b) pois existem várias configurações que a representam. Assim, à primeira vista, a melhor configuração usa o mesmo número de segmentos por classe **2-2**, mas também as configurações **2-3** e **3-2** conseguem representar os dados. No caso da Distribuição X (Fig. 4.4(b)), onde sabemos que é uma distribuição tipo "XOR", estimar o número de segmentos pode ser fácil, o número ideal de segmentos de reta para cada classe é **2-2**; e como pode-se ver na figura outras configurações poderiam gerar uma sobreposição dos segmentos de reta, com o qual o erro se incrementaria, contudo as configurações **1-1** e **2-1** poderiam funcionar bem. Para a Distribuição Simples (Fig. 4.4(a)), o número de segmentos de reta necessários diminui, pois a configuração mais simples **1-1** é suficiente para representar os dados. Porém, neste caso, qualquer configuração de segmentos de reta encontrada pelo *K-Means* é válida.

Embora tenha sido possível determinar qual poderia ser o número de segmentos de reta observando uma representação gráfica da distribuição dos dados, quando o objetivo é lidar com problemas reais, onde não conhecemos a distribuição e nem conseguimos representá-la, surgem os seguintes problemas:

(a) Precisamos de uma representação gráfica da distribuição dos dados para poder determinar o número de segmentos;

(b) Se a distribuição de dados não for conhecida (caso mais comum), necessitaríamos executar o algoritmo de treinamento repetidas vezes, avaliando todas as combinações possíveis, até encontrar um valor bom o suficiente para a distribuição;

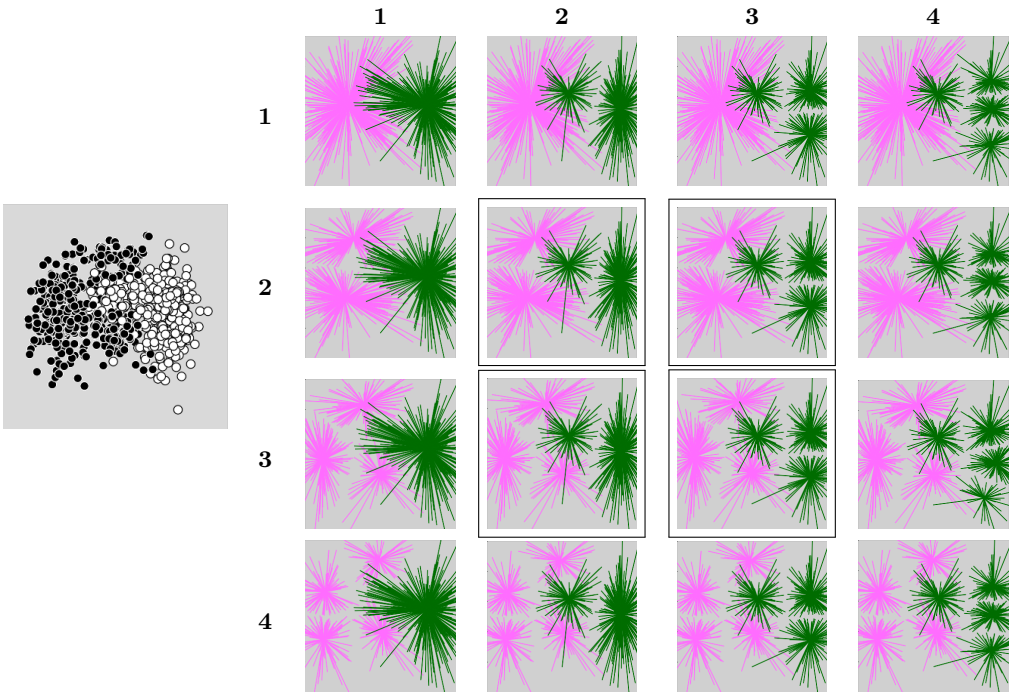
(c) Nossa proposta de algoritmo de treinamento híbrido apresentada no capítulo prévio, embora consiga melhorar um pouco a acurácia do classificador, incrementa o tempo de execução (pelos múltiplas iterações dos algoritmos evolutivos); o qual dificulta a busca exaustiva, para poder conhecer o número "ótimo" de segmentos de reta;

(d) Finalmente, se tentássemos adivinhar o número de segmentos de reta a serem usados, não vamos garantir o mínimo erro do classificador.

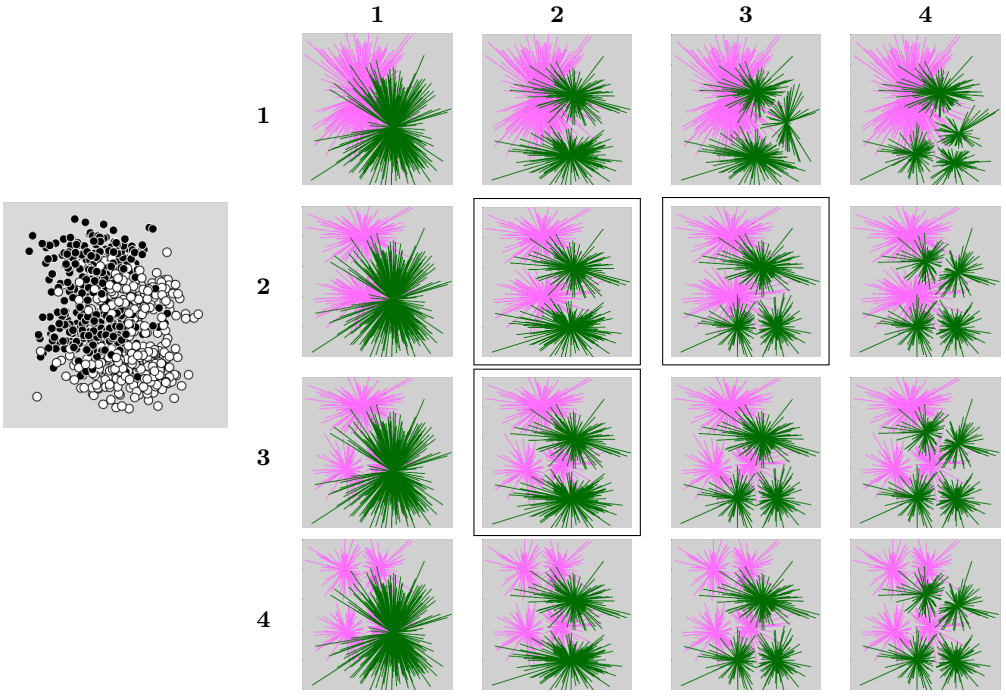
Nas seguintes figuras: 4.5 e 4.6, são apresentados os resultados obtidos após o treinamento do classificador, ou seja, mostramos as posições finais dos segmentos de reta. Cada figura consta de 4 linhas e 4 colunas, nas quais novamente o número de segmentos para a classe zero se incrementa para baixo e os segmentos da classe um, à direita, respectivamente.

Desta forma, mostramos os resultados para a Distribuição F (Fig.4.5.a), foram destacadas em requadro, as configurações que consideramos como bons exemplos para representar os dados nas figuras anteriores (representações dos agrupamentos). Podemos dizer que as escolhas foram boas, pois os segmentos de reta se encaixam na distribuição dos dados. Além disso, as porcentagens de acerto, indicam que a configuração de segmentos de reta que gerou um erro mínimo foi **2-2**, com um acerto de 87.17%, sendo que o acerto de Bayes para essa distribuição é de 87.66%. Também foram obtidas outras porcentagens altas como de 86.99% e 86.55%, usando diferente número de segmentos para cada classe: **2-3** e **4-3**, respectivamente.

Para a Distribuição S, podemos ver que para as configurações de segmentos de reta iguais para cada classe, localizados na diagonal da matriz, a fronteira de decisão tem uma tendência a ser linear, ao contrario da configuração **3-2** destacada na Figura 4.5(b), onde os segmentos aparentemente representam melhor a distribuição dos dados. As porcentagens de acerto indicam que essa configuração apresenta um dos melhores resultados, igual a 63, 87%; contudo temos outras configurações (também com diferente número de segmentos por classe)



(a)

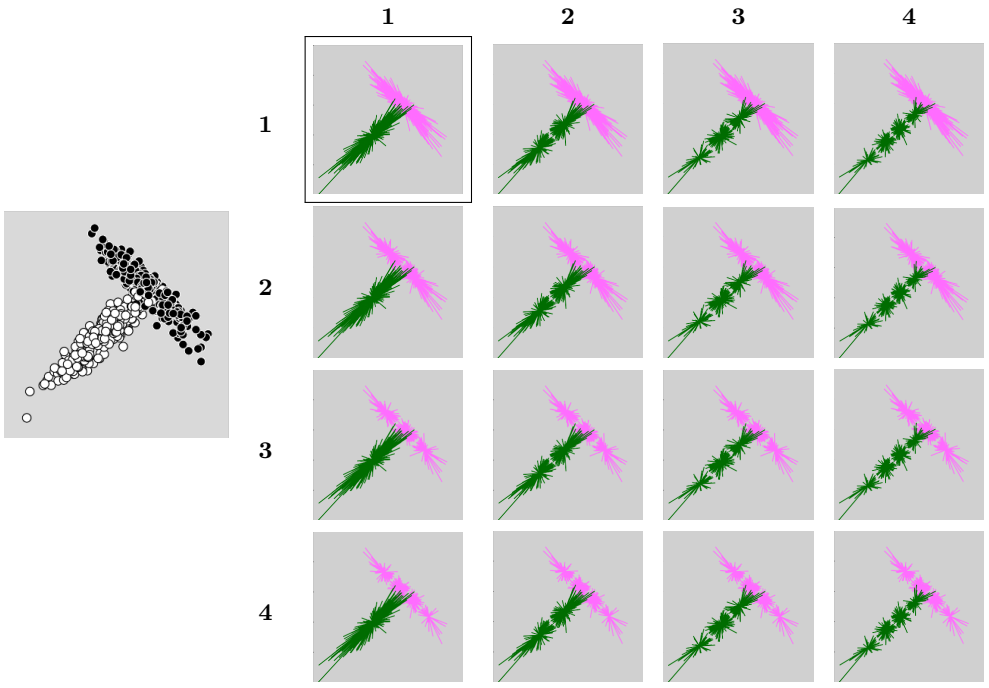


(b)

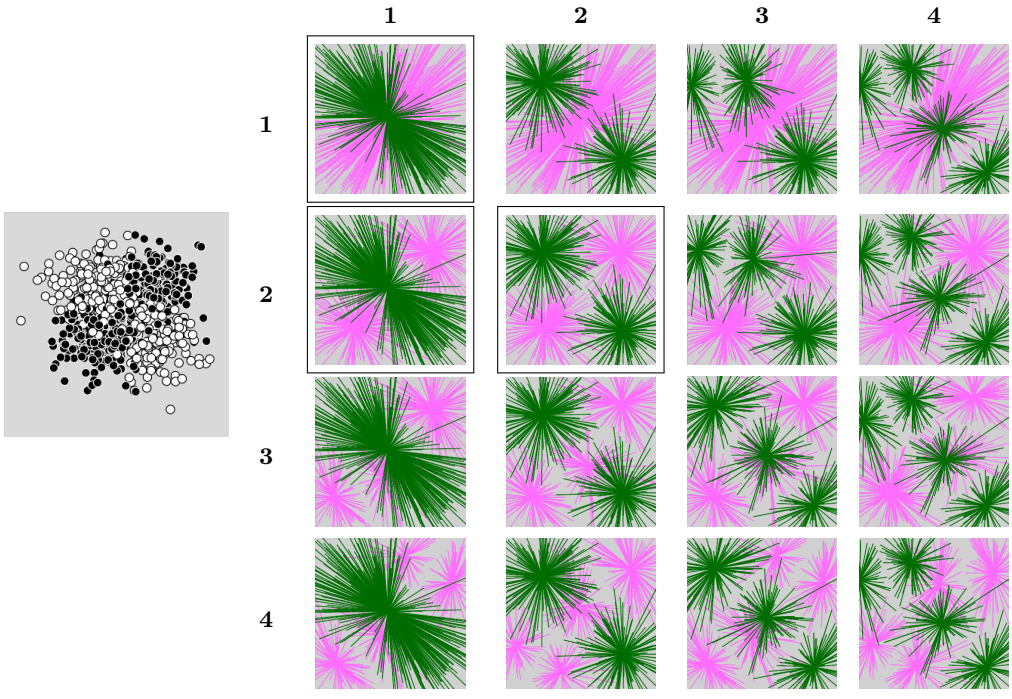
Figura 4.3: *Agrupamentos obtidos após a fase de* Pre-Alocação *do classificador SLS, para: (a) Distribuição F e (b) Distribuição S. Os agrupamentos em destaque (requadro), representam aqueles agrupamentos que visualmente conseguem dividir as duas classes tal como a distribuição da direita.*

onde a porcentagem de acerto foi maior, por exemplo: (*i*) **2-1** com $66,43\%$ e (*ii*) **4-3** com $65,75\%$. Também existem algumas configurações que ao invés de incrementar a porcentagem de acerto, a diminuem; é o caso das configurações **1-3**, **1-4**, **3-1** e **4-1**.

Como foi mencionado anteriormente, qualquer configuração de segmentos é boa para representar a Distribuição Simples. Novamente os segmentos de reta que estão na diagonal da



(a)



(b)

Figura 4.4: *Agrupamentos obtidos após a fase de* Pre-Alocação *do método SLS, para: (a) Distribuição Simples e (b) Distribuição X. Os agrupamentos em destaque (requadro), representam aqueles agrupamentos que visualmente conseguem dividir as duas classes tal como a distribuição da direita.*

matriz da Fig. 4.6(a), tendem a dividir as classes de forma linear, onde a porcentagem de acerto não atinge o 93%. Mas, usando as outras configurações de segmentos a porcentagem atinge no melhor dos casos 93.63% com **2-4** segmentos, a qual fica muito próximo do acerto de Bayes para essa distribuição (93.90%). As configurações que garantem um melhor resultado poderiam ser: **1-2**, **1-3**, **2-1**, **3-2**, **3-4**, **4-2** e **4-3**, todas geraram uma porcentagem de acerto maior a 93.0%.

Um caso aparte é a Distribuição X, cujos resultados estão apresentados na Figura 4.6(b), onde podemos observar que as maiores porcentagens de acerto pertencem a imagens na diagonal da matriz (configurações com igual número de segmentos por classe). Essas porcentagens são próximas do acerto de Bayes para X, que é 66.70%, sendo que a melhor porcentagem obtida dentre as diferentes configurações de segmentos é de 66.49% para **1-1** segmentos. Neste caso, utilizar um número diferente de segmentos de reta não foi de grande ajuda, ao contrário contribuiu à diminuição da taxa de acerto.

4.3 Aplicação do algoritmo *X-Means*

Como foi visto na seção anterior, usar um número diferente de segmentos de reta para representar às classes, proporciona uma melhora na representação dos dados. Nesta seção, propomos usar o algoritmo de clusterização *X-Means*[1] (Sec. 2.3.2), para evitar realizar múltiplos treinamentos usando todas as possíveis combinações de segmentos de reta, como foi mostrado na seção anterior; ou tentar adivinhar qual é o melhor número de segmentos de reta para cada classe. A vantagem deste algoritmo é que não precisamos de indicar o número de agrupamentos a serem encontrados (K), como acontece com o *K-Means*. Deste modo, podemos estimar o número de segmentos de reta que representarão cada classe usando o número de agrupamentos, resultado da aplicação do algoritmo de clusterização em cada classe.

O algoritmo de *X-Means* inicia a partir de um pequeno número de agrupamentos. Depois, vai decidindo localmente para cada centróide, se é necessário dividi-lo em dois novos centróides ou não. O critério de decisão padrão, para dividir os centróides, usado no algoritmo é o *Bayesian Information Criterion (BIC)* [Schwarz, 1978]. Porém, pode ser usado também o critério *Anderson-Darling Criterion (AD)*, ideia proposta no trabalho de [Hamerly e Elkan , 2003], pelo seu bom desempenho em agrupamentos que não possuem forma esférica. No nosso trabalho, decidimos usar o último critério mencionado *(AD)*, pois não esperamos que todas as distribuições de dados sejam esféricas.

Para demonstrar que este algoritmo consegue encontrar uma boa configuração de segmentos de reta, foi usado em cada uma das distribuições artificiais da seguinte forma:

1. Dividir o conjunto de dados em suas respetivas classes $\{0, 1\}$;
2. Aplicar o *X-Means* em cada classe, para encontrar $nSLS_0$ agrupamentos para a classe zero e $nSLS_1$ para a classe um;
3. Aplicar o algoritmo *K-Means* com valor $K = 2$, em cada agrupamento encontrado para cada classe, a fim de encontrar as extremidades dos segmentos de reta.

A seguir, apresentamos os resultados da aplicação do *X-Means* na fase de treinamento do classificador SLS nas Figuras 4.7-4.10, para cada uma das distribuições e todas relacionadas com sua respectiva tabela de porcentagens de acerto de classificação.

Por exemplo, para os resultados obtidos para a Distribuição F representados na Figura 4.7, podemos observar no lado esquerdo, quais foram os agrupamentos (representados pelos centróides) encontrados pelo algoritmo de clusterização. Assim, o número encontrado para a classe zero é 4, e para a classe um é 2; no lado direito, apresentamos os resultados mais

[1] O código do algoritmo *X-Means* foi proporcionado pelo autor Dan Pelleg http://www.cs.cmu.edu/~dpelleg/ para fins de pesquisa.

relevantes para esta distribuição ou seja, a posição final dos segmentos de reta após o treinamento e, como o número de segmentos de reta varia com respeito ao número de exemplos na amostra. Observando os resultados da tabela (no topo da mesma figura), a porcentagem de acerto se incrementa conforme o número de amostras cresce, sendo o melhor resultado (ressaltado em negrito e sublinhado) usando *X-Means* igual a 87.3% não muito diferente do resultado obtido pelo *K-Means* e muito próximo do acerto de Bayes.

A Distribuição S, é um exemplo onde o número de segmentos de reta não varia com respeito ao número de exemplos na amostra, pode-se observar na Fig. 4.8. O algoritmo de agrupamento encontrou 2 grupos para cada classe (lado esquerdo da figura) e no lado direito encontramos as posições finais dos segmentos de reta após o treinamento. Podemos ver que a diferença entre as diferentes posições dos segmentos de reta, é mínima, ou seja é provável que as respectivas porcentagens de acerto sejam similares. Para confirmar isso, mostramos as porcentagens da tabela, na mesma figura, onde o melhor resultado obtido a partir das combinações de segmentos de reta é 66.43% e usando o *X-Means* o resultado é menor aproximadamente em 1%.

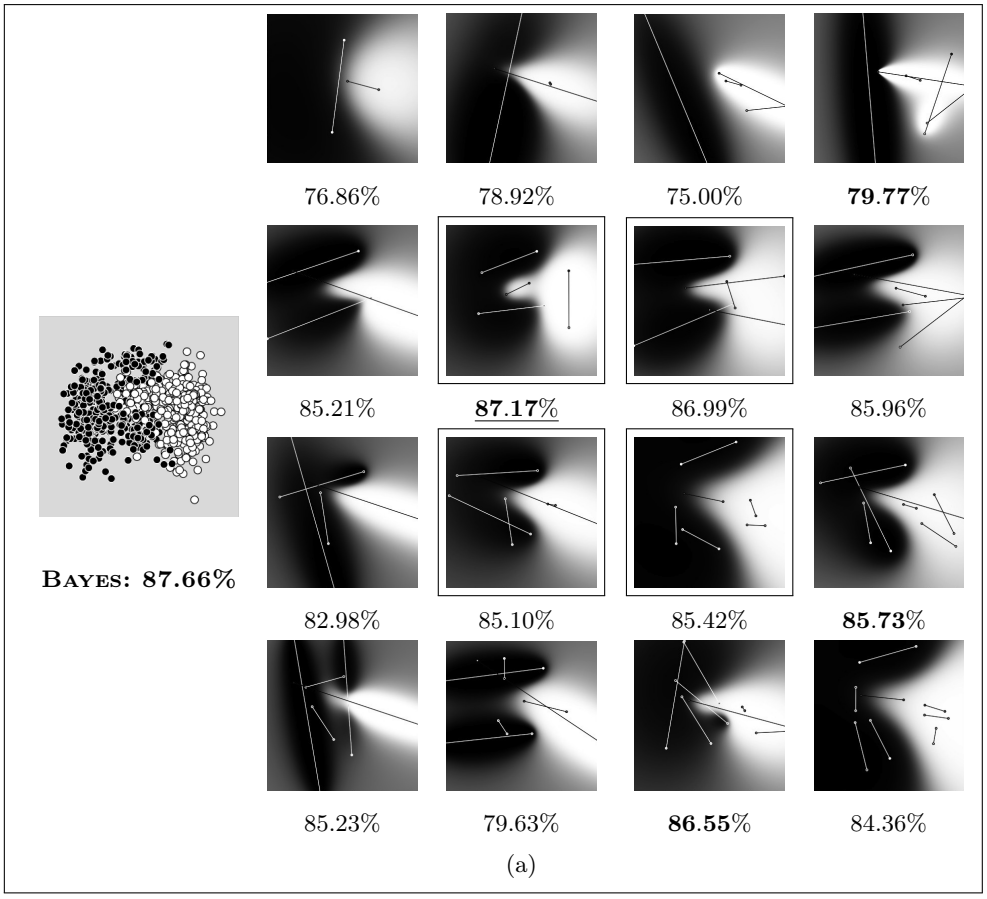
Para a Distribuição-Simples o número de agrupamentos encontrados pelo X-Means (veja Fig. 4.9) é de 4 para cada classe, e sim, varia com respeito ao tamanho da amostra. Podemos ver que a posição dos segmentos de reta tem a mesma orientação em todos os resultados mostrados. Enquanto aos resultados em porcentagens na tabela antes mencionada, a maior foi de 92.88% usando 3 segmentos por classe, no entanto foi obtido um melhor porcentagem usando **2-4** segmentos para obter 93.63%. É bom ressaltar que a posição dos segmentos de reta tem pouca variação nas figuras com amostras de tamanho 800 e 1600, assim as porcentagens de acerto são também similares.

Enquanto à última distribuição, a Dist. X (veja a Figura 4.10), o algoritmo encontrou 2 segmentos de reta para cada classe para todos os tamanhos de amostra testados e nesta distribuição acontece o mesmo que com a Distribuição S a posição dos segmentos de reta varia muito pouco e os resultados em porcentagens o confirmam. Desta forma, o melhor resultado obtido após o treinamento é 66.61% tanto para o *X-Means* como para o melhor resultado obtido das diferentes combinações de segmentos de reta, usando *K-Means*; ambas porcentagens quase atingiram o acerto de Bayes, a diferença é de 0.9%.

Neste capítulo foram descritas duas ideias para encontrar o número ideal de segmentos de reta a serem usados no classificador SLS: (*i*) Realizar uma busca exhaustiva e (*ii*) Aplicar o algoritmo *X-Means*. Para ambos casos foram apresentados os resultados tanto em imagens com a posição final dos segmentos de reta após o treinamento, como as porcentagens de acerto atingidas para cada um dos exemplos.

Baseando-nos nos resultados podemos concluir que para este tipo de distribuições artificiais (distribuições conhecidas) o acerto de classificação sempre fica muito próximo do acerto de Bayes, para ambas ideias. Por um lado, a principal desvantagem da primeira ideia é o espaço de busca, pois se o tamanho da amostra for maior e a distribuição desconhecida, precisaríamos usar mais de 4 segmentos de reta por classe, o qual incrementaria a tabela de combinações exponencialmente. Por outro lado, uma possível desvantagem da segunda ideia é que a porcentagem de classificação não é tão alta quanto o acerto obtido a partir da busca exaustiva. Contudo, conseguimos reduzir o tempo que levaria treinar se for usada a primeira ideia, o número de segmentos usados encontrados pelo *X-Means* só coincide com o número de segmentos que atingiram a melhor porcentagem após a busca exaustiva para a distribuição X, e a porcentagem de acerto foi menor para todas as simulações apresentadas neste capítulo.

No Capítulo 5 vamos a apresentar os resultados para todas as simulações feitas comparando os diferentes métodos de treinamento híbridos descritos no Capítulo 3 e aplicando as duas ideias apresentadas previamente, para assim, decidir qual delas é melhor nas distribuições artificiais. No caso dos problemas reais, será usado o *X-Means* pois não conhecemos qual é a distribuição dos dados e esperamos apresente uma diferença relevante na taxa de acerto de classificação quando seja comparada com resultados do classificador original.



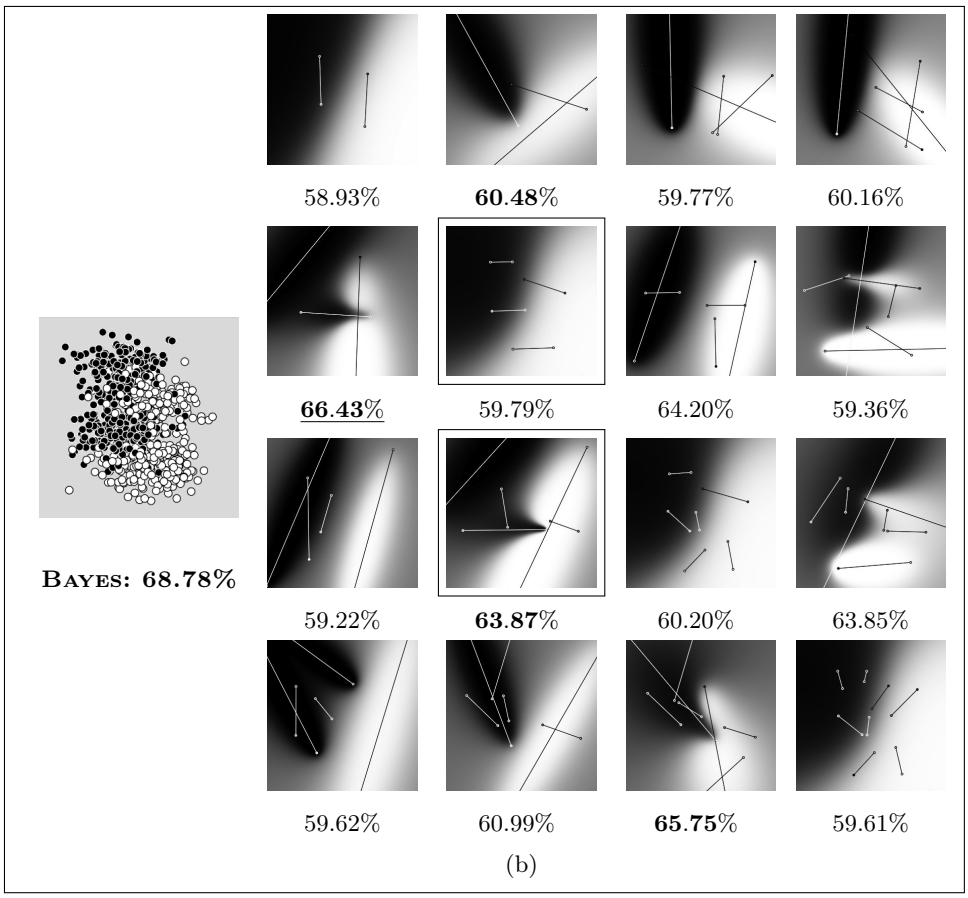


Figura 4.5: *Posição dos segmentos de reta, obtidos após a fase de treinamento, para: (a) Distribuição F e (b) Ditribuição S. Em negrito: as taxas de acerto para cada linha e sublinhado: a melhor taxa de acerto para cada distribuição.*

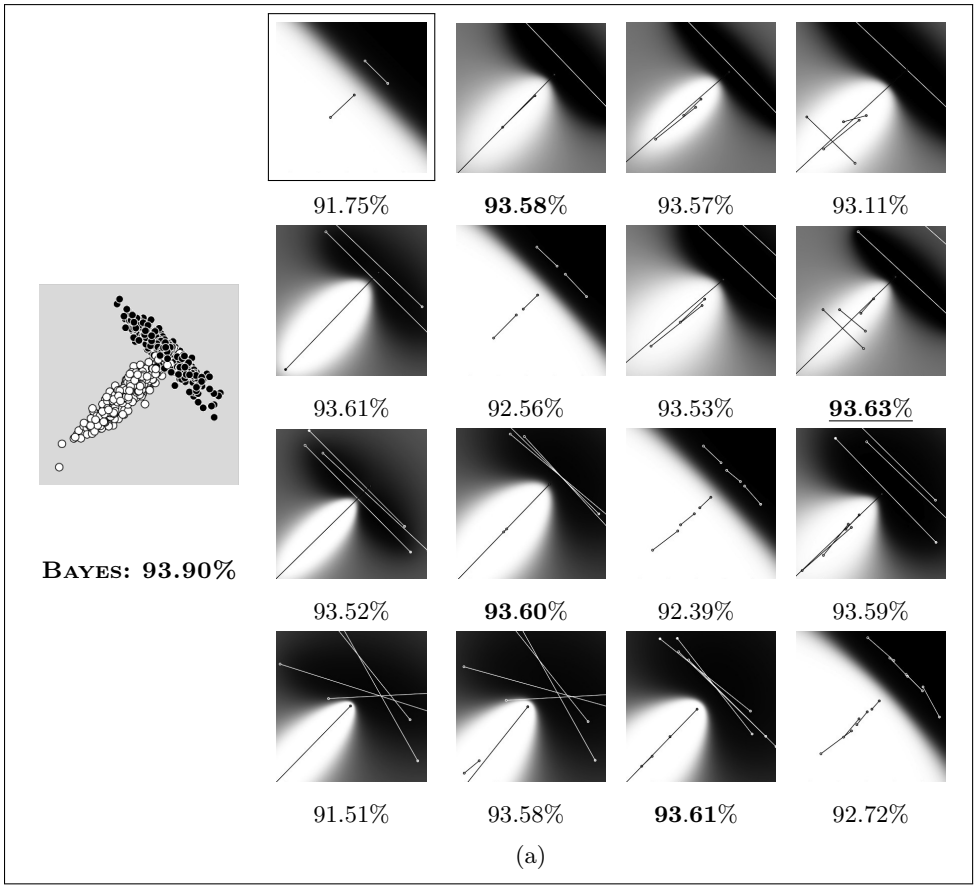


Figura 4.6: *Posição dos segmentos de reta, obtidos após a fase de treinamento, para: (a) Distribuição Simples e (b) Distribuição X. Em negrito: as taxas de acerto para cada linha e sublinhado: a melhor taxa de acerto para cada distribuição.*

	Usando X-Means				K-Means
Tx. Acerto Bayes	$2-2$ sls	$3-2$ sls	$4-2$ sls	$4-3$ sls	$2-2$ sls
	100 exemplos	800 exemplos	1600 exemplos	3200 exemplos	1600 exemplos
87.66%	85.89%	84.87%	83.22%	**87.03**%	**87.17**%

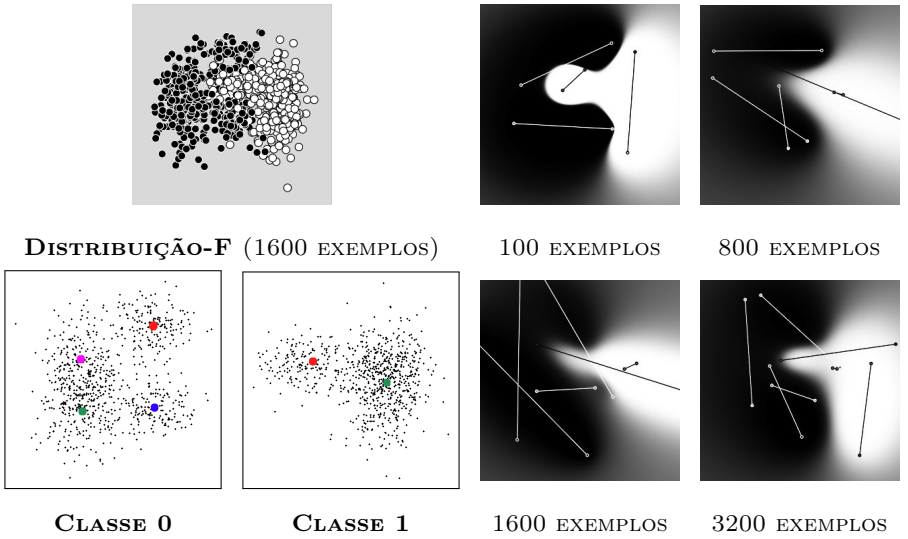


Figura 4.7: *Resultados da aplicação do* X-Means *na Distribuição F: no topo, a taxa de acerto de Bayes usando X-Means em comparação com K-Means e abaixo a representação gráfica das posições finais dos Segmentos de Reta após o treinamento, usando X-Means. Em negrito: o melhor valor para a busca exaustiva e sublinhado: o melhor valor para o X-Means.*

	Usando X-Means				K-Means
Tx. Acerto Bayes	$2-2$ sls	$2-2$ sls	$2-2$ sls	$2-2$ sls	$2-1$ sls
	100 exemplos	200 exemplos	800 exemplos	1600 exemplos	1600 exemplos
68.78%	**65.40**%	63.57%	59.79%	59.83%	**66.43**%

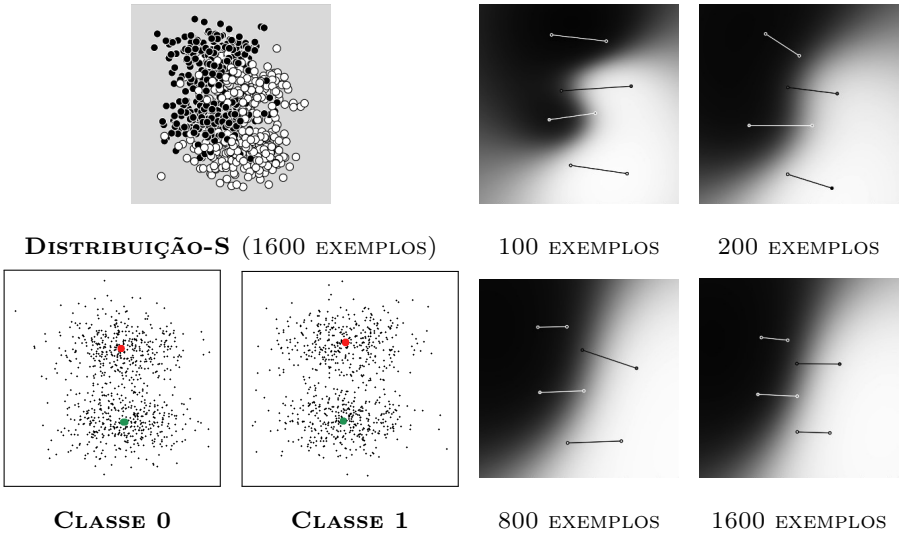


Figura 4.8: *Resultados da aplicação do* X-Means *na Distribuição S: no topo, a taxa de acerto de Bayes usando X-Means em comparação com K-Means e abaixo, a representação gráfica das posições finais dos Segmentos de Reta após o treinamento, usando X-Means. Em negrito: o melhor valor para a busca exaustiva e sublinhado: o melhor valor para o X-Means.*

	Usando X-Means				K-Means
Tx. Acerto Bayes	$2-2$ sls	$3-3$ sls	$4-4$ sls	$4-4$ sls	$2-4$ sls
	200 exemplos	400 exemplos	800 exemplos	1600 exemplos	1600 exemplos
93.90%	92.28%	**92.88**%	92.72%	92.75%	**93.63**%

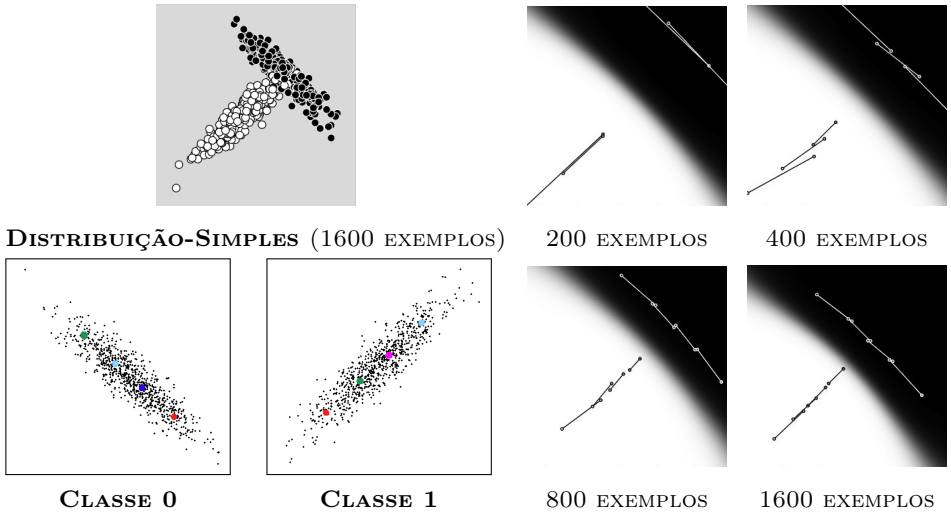


Figura 4.9: *Resultados da aplicação do* X-Means *na Distribuição Simples: no topo, a taxa de acerto de Bayes usando X-Means em comparação com K-Means e abaixo, a representação gráfica das posições finais dos Segmentos de Reta após o treinamento, usando X-Means. Em negrito: o melhor valor para a busca exaustiva e sublinhado: o melhor valor para o X-Means*

	Usando X-Means				K-Means
Tx. Acerto Bayes	$2-2$ sls	$2-2$ sls	$2-2$ sls	$2-2$ sls	$2-2$ sls
	100 exemplos	200 exemplos	400 exemplos	3200 exemplos	3200 exemplos
66.70%	66,07%	64,46%	65,27%	**66,61**%	**66.61**%

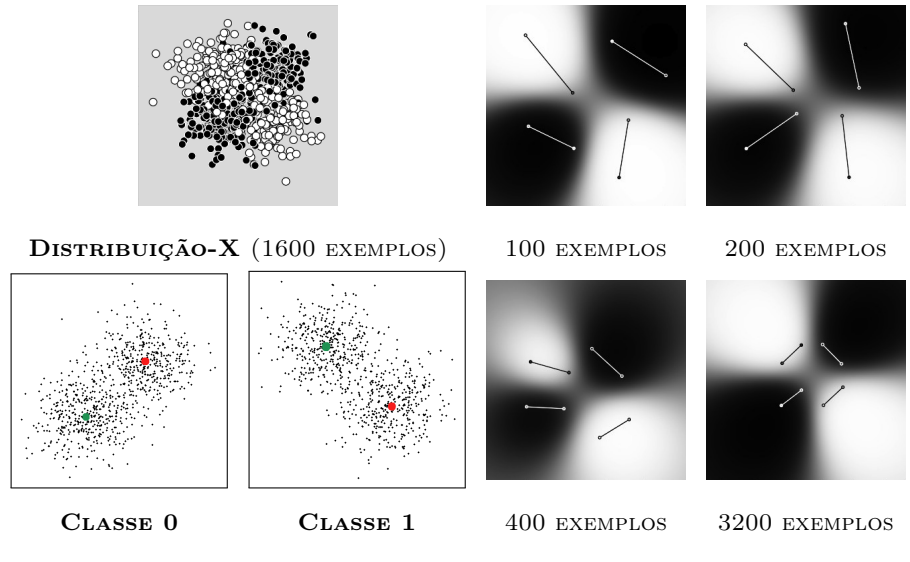


Figura 4.10: *Resultados da aplicação do* X-Means *na Distribuição X: no topo, a taxa de acerto de Bayes usando X-Means em comparação com K-Means e abaixo, a representação gráfica das posições finais dos Segmentos de Reta após o treinamento, usando X-Means. Em negrito: o melhor valor para a busca exaustiva e sublinhado: o melhor valor para o X-Means.*

Capítulo 5

Resultados e Discussões

Neste capítulo são apresentados os resultados obtidos a partir das diferentes implementações dos métodos híbridos de treinamento descritos no Capítulo 3. Estes métodos foram implementados para serem executados em paralelo em C++, e foram produzidos usando computadores com 16 processadores Intel 2.4 GHz com 5.8 GB de memória RAM sob o sistema operacional Ubuntu/Linux.

Desta forma, três diferentes tipos de experimentos foram realizados com o intuito de avaliar qual é o melhor algoritmo e qual é a melhor forma de estimar o número de segmentos para representar as classes, que apresentaram melhor desempenho tanto nas distribuições artificiais como nos problemas reais. Espera-se que os métodos híbridos de treinamento, algoritmos de otimização combinados com o Gradiente Descendente, funcionem melhor do que os mesmos algoritmos sem ser combinados.

5.1 Considerações Iniciais

Algumas definições previas devem ser consideradas antes de descrever os resultados obtidos, enquanto ao número de treinamentos realizados, número de exemplos por amostra, e são listadas a seguir:

(a) *Distribuições*: F , S, Simples e X ;

(b) *Amostras*: Foram usadas amostras com 6 diferentes tamanhos: 100, 200, 400, 800, 1600 e 3200 exemplos, 3 para cada quantidade de exemplos, em um total de 18 amostras para cada distribuição ;

(c) *Algoritmos de Treinamento*: Como foi descrito no Capítulo 3 onde propomos 3 métodos, também usamos outros algoritmos de otimização (em um total de 5 algoritmos), listados a seguir:

   (a) **GD** - Gradiente Descendente (algoritmo original) ,
   (b) **AG+GD** - Algoritmos Genéticos + Gradiente Descendente ,
   (c) **AG** - Algoritmos Genéticos ,
   (d) **MDO+GD** - Método Dialético de Otimização + Gradiente Descendente ,
   (e) **KBM+GD** - *K-Beams* + Gradiente Descendente.

(d) *Técnica para estimar os segmentos de Reta*: Usaremos as duas ideias apresentadas no Capítulo 4: (*i*) Busca Exaustiva e (*ii*) Aplicando o *X-Means*; para o caso (*i*), o espaço de busca estará entre 1 e 4 segmentos de reta por classe, ou seja: { 1-1, 1-2, 1-3, 1-4; 2-1, 2-2, 2-3, 2-4; 3-1, 3-2, 3-3, 3-4; 4-1, 4-2, 4-3, 4-4 }.

(e) *Parâmetros*: Consideramos 3 variações de parâmetros para cada algoritmo de treinamento, e são definidos na tabela a seguir:

Tabela 5.1: *Parâmetros para cada algoritmo de treinamento proposto.*

AG+GD	Parâmetros		
	Param. 1	Param. 2	Param. 3
população	2	10	50
gerações	2	5	3

AG	Parâmetros		
	Param. 1	Param. 2	Param. 3
população	200	500	1000
gerações	100	1000	500

MDO+GD	Parâmetros		
	Param. 1	Param. 2	Param. 3
pólos	30	15	50
fases	20	10	15
iterações	15	15	15

KBM+GD	Parâmetros		
	Param. 1	Param. 2	Param. 3
estados	2	5	10
sucessores	10	20	15
iterações	3	10	5

5.2 Resultados para Dados Artificiais

Na seção anterior comentou-se quais são as definições a serem usadas nos experimentos realizados, temos assim que, para cada distribuição (quatro no total) foram geradas ao todo 18 amostras, e para cada amostra foram realizados 16 treinamentos, seguindo a ideia da busca exaustiva. Além disso, foram testados todos os algoritmos de treinamento (**GD**, **AG+GD**, **AG**, **MDO+GD** e **KBM+GD**) e para esses algoritmos também foram avaliados os resultados obtidos para as três variações de parâmetros. Resumindo, foram geradas 14,976 treinamentos, para a busca exaustiva, e 1,080 treinamentos aplicando o *X-Means*, ou seja, ao todo foram realizados **16,056** treinamentos[1]. Foram realizados 3 treinamentos e 3 testes para cada amostra, para assim obter uma média dos resultados para os dados artificiais.

O primeiro experimento foi considerado com o intuito de investigar qual dos algoritmos é o melhor em desempenho baseando-nos na porcentagem de acerto de classificação. Um segundo experimento é realizado a fim de determinar qual é o conjunto de parâmetros que atinge um máximo valor de acerto na classificação e finalmente no último experimento temos por objetivo definir qual das ideias apresentadas no Capítulo 4, nos permite obter algum incremento na acurácia do classificador SLS quando os resultados são comparadas com o algoritmo original.

5.2.1 Algoritmos de Treinamento

Esta seção descreve o primeiro dos experimentos realizados, temos como objetivo determinar qual dos cinco algoritmos utilizados no trabalho, permite obter a melhor taxa de acerto de classificação. As médias percentuais, o desvio padrão e o número de segmentos de reta obtidos para cada execução podem ser encontrados na Tabela A.1. É importante ressaltar que para este experimento foi usada a busca exaustiva, ou seja, procurar pela melhor combinação de segmentos de reta, e os parâmetros usados foram Param.1 definidos na Tabela 5.1. A razão disto, foi comparar o desempenho dos algoritmos de treinamento.

Como mostra a tabela antes mencionada e o diagrama apresentado na Figura 5.2, o algoritmo de treinamento híbrido de Otimização Dialética e Gradiente Descendente (**MDO+GD**), apresentou bons resultados tanto para a Distribuições F e S como para a Distribuição X.

[1] Note-se que as tabelas que contêm as médias dos resultados obtidos, podem ser encontradas no Apêndice A.

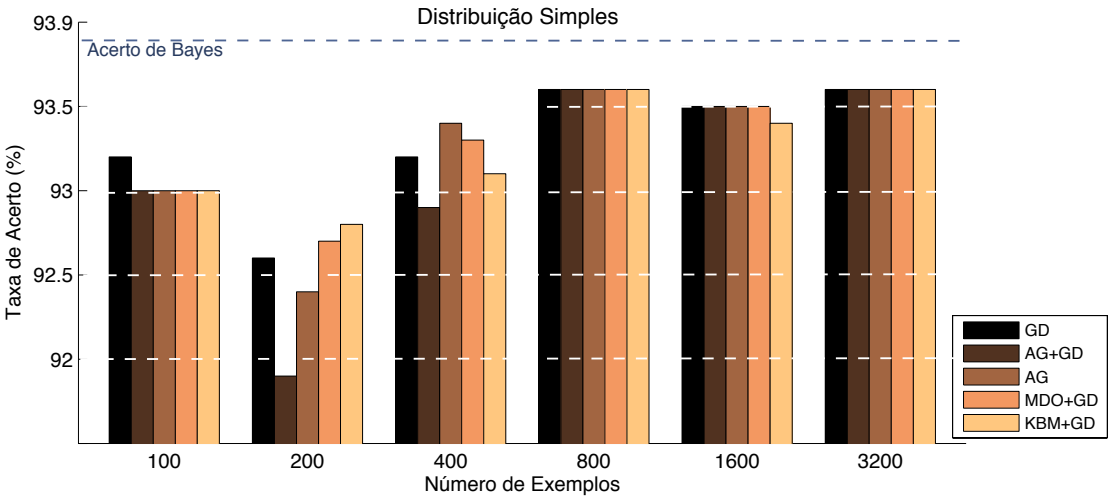


Figura 5.1: *Gráfico de barras representando a comparação entre os algoritmos de treinamento: (GD) Gradiente Descendente, (AG+GD) Alg. Genéticos e Gradiente Descendente, (AG) Alg. Genéticos, (MDO+GD) Otimização Dialética e Gradiente Descendente, e (KBM+GD) K-Beams e Gradiente Descendente; para a Distribuição Simples.*

As porcentagens de acerto atingidas para a distribuição F (veja a Figura 5.2(a)) foram no primeiro lugar 87.6% usando o treinamento **MDO+GD** e no segundo lugar 87.5% com o algoritmo (**KBM+GD**); sendo que o acerto de Bayes para essa distribuição é de 87.66%. A Figura 5.2(b) exibe os resultados para a Distribuição S, podemos ver que as maiores porcentagem, 68.5% e 68.3%, foram atingidas com 3200 exemplos usando novamente os algoritmos de treinamento **MDO+GD** e **KBM+GD**, respetivamente. Para a distribuição X, o gráfico na Fig. 5.2(c) apresenta o mesmo comportamento das duas distribuições antes mencionadas, a maior porcentagem atingida foi de 66.6% para os algoritmos **GD**, **MDO+GD** e **KBM+GD**; sendo a única diferença entre eles o desvio padrão. Para estas três distribuições os resultados foram próximas ao acerto de Bayes, a diferença foram só de décimas. Um caso especial é a distribuição Simples, como pode-se observar na Figura 5.1,os resultados para todos os algoritmos de treinamento atingiram um valor próximo do acerto de Bayes (93.90%), para amostras de tamanho maior a 800 .

Tomando como base os resultados previamente exibidos para este experimento, podemos concluir que:

(a) Podemos considerar o algoritmo de treinamento híbrido **Otimização Dialética e Gradiente Descendente**, como o melhor dentre os 5 algoritmos implementados. Embora não conseguiu as melhores porcentagens de acerto para duas distribuições, obteve as segundas melhores taxas de acerto. Um outro algoritmo que apresentou bons resultados foi o híbrido entre ***K-Beams* e Gradiente Descendente**. Pelos resultados apresentados nas gráficas de comparação de porcentagens, podemos dizer que todos os algoritmos obtêm taxas de acerto similares inclusive iguais pois a diferença às vezes é de 0.1%.

(b) Dado que, para este experimento foram usados só os melhores resultados obtidos a partir da busca exaustiva de segmentos de reta, podemos observar que para três das quatro distribuições a porcentagem de acerto foi obtida a partir do uso de um diferente número de segmentos de reta para cada classe; sendo a distribuição X a única que precisou de um mesmo número de segmentos de reta. Assim, podemos dizer que usar um número diferente de segmentos de reta permite incrementar a taxa de acerto.

(c) Finalmente, é comum que a porcentagem de acerto seja maior conforme o tamanho da amostra também é incrementado, como pode-se observar nas gráficas de resultados.

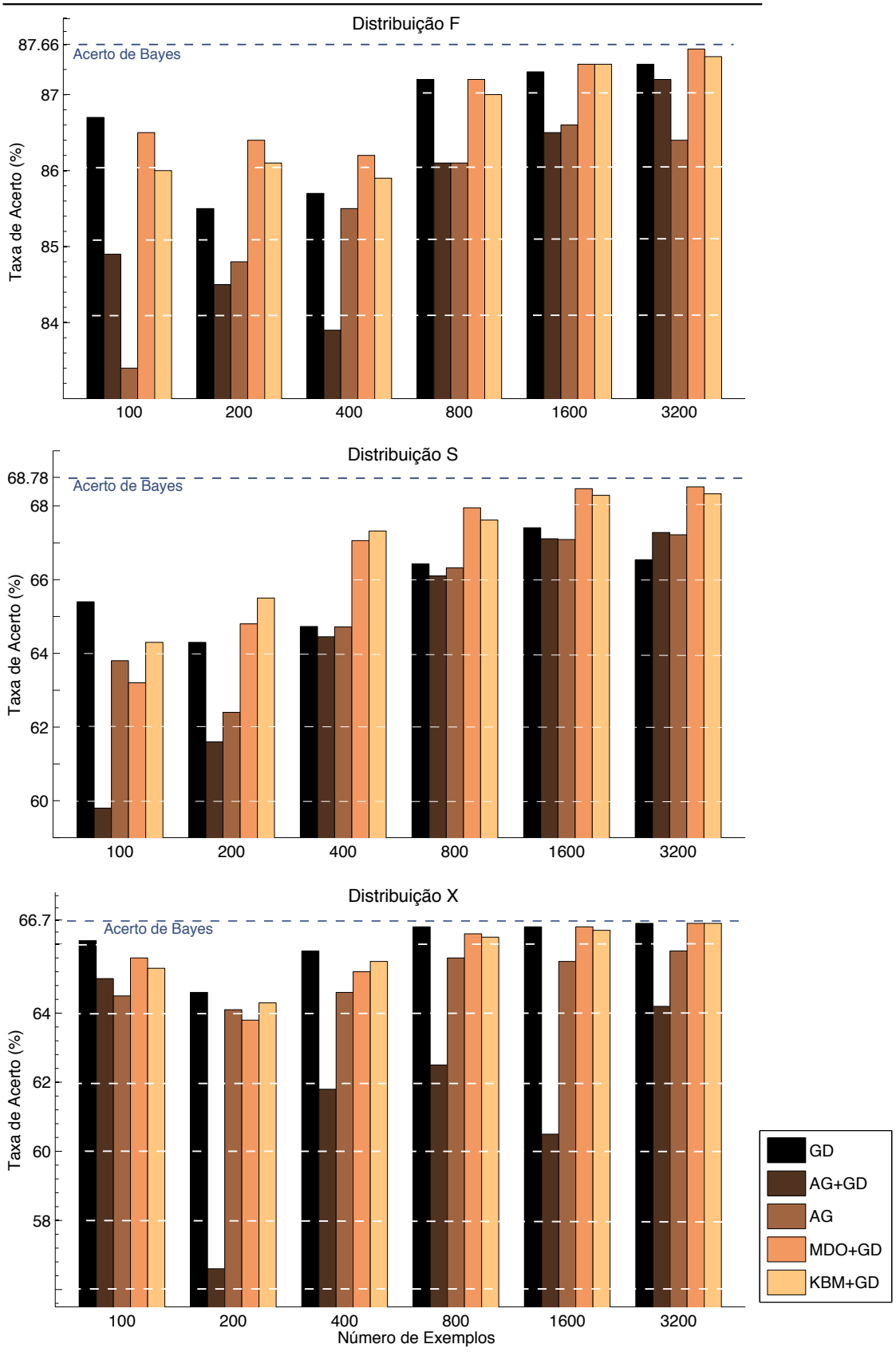


Figura 5.2: *Gráfico de barras comparando a porcentagem de acerto de classificação pelo número de amostras; para os cinco algoritmos de treinamento implementados: (GD) Gradiente Descendente, (AG+GD) Alg. Genéticos e Gradiente Descendente, (AG) Alg. Genéticos, (MDO+GD) Otimização Dialética e Gradiente Descendente e (KBM+GD) K-Beams e Gradiente Descendente, para as distribuições F, S e X.*

5.2.2 Variações de Parâmetros

Os resultados deste experimento, cujo objetivo é determinar quais das três variações de parâmetros permitem obter uma melhor taxa de acerto, são apresentados em detalhe nas Tabelas A.2-A.5. Assim, vamos resumir esses resultados em comparações da taxa de acerto de classificação, apresentadas nas Figuras 5.3 e 5.4, onde cada linha representa uma determinada distribuição, e cada coluna representa um algoritmo de treinamento, respectivamente. É importante ressaltar que o algoritmo **GD** não foi considerado neste experimento, dado que o Gradiente Descendente sempre foi usado com 1000 iterações. Cada curva nas figuras representa o desempenho do algoritmo para cada um dos parâmetros definidos no inicio do capítulo, sendo os Parâmetros-1 representados pelos círculos, Parâmetros-2 pelos quadrados e Parâmetros-3 pelos triângulos.

Para a combinação dos Algoritmos Genéticos e Gradiente Descendente (**AG+GD**), aqueles parâmetros que apresentaram um melhor desempenho quando foram aplicados nas quatro distribuições de dados, foram os Parâmetros-3. Para os Algoritmos Genéticos executados sem ser combinados (**AG**), também as maiores porcentagens de acerto foram atingidas usando os Parâmetros-3; no caso do algoritmo de treinamento da Otimização Dialética com Gradiente Descendente (**MDO+GD**), pode-se observar que para tamanhos de amostras maiores a 800 os parâmetros que atingiram uma alta porcentagem, foram também os Parâmetros-3. O mesmo acontece para o algoritmo (**KBM+GD**).

Segundo as tabelas antes mencionadas e os gráficos apresentados, podemos dizer que:

(a) Para a maioria dos algoritmos de treinamento, a execução deles usando os **Parâmetros-3**, conseguiram obter as máximas porcentagens de acerto.

(b) Para as duas primeiras distribuições as variações nos parâmetros podem ter um leve incremento na taxa de acerto, cerca de 1% para as 6 amostras. Já nas últimas distribuições as execuções dos algoritmos com diferentes parâmetros não apresenta uma diferença significativa, muitas vezes o valor é o mesmo, especificamente para amostras de tamanho maior a 800.

(c) Um caso especial foi o treinamento com Algoritmos Genéticos e Gradiente Descendente da distribuição X, onde a diferença entre os Parâmetros-1 e 3 foi grande, cerca de 6%.

(d) Podemos concluir que aplicar qualquer uma das variações de parâmetros vai produzir resultados bons, pois pelas gráficas as porcentagens de acerto ficam estáveis com amostras de tamanho 400 a mais, para todos os algoritmos de treinamento testados, e isso acontece também para as quatro distribuições utilizadas nos experimentos.

5.2.3 Estimação de Segmentos de Reta

Neste último experimento realizado, nosso objetivo foi determinar qual das duas formas de estimar o número de segmento de reta é a melhor alternativa a ser usada. Assim, nesta subseção apresentamos as taxas de acerto de classificação para cada distribuição e para todos os algoritmos de treinamento pois no Capítulo 4, unicamente nos focamos no Gradiente Descendente (**GD**). Deste modo, apresentamos na Figura 5.5, um gráfico de barras para cada distribuição e em cada um deles a comparação entre as taxas de acerto para cada algoritmo de treinamento considerado nos experimentos aplicando a busca exaustiva (em preto, usando *K-Means*) e aplicando o algoritmo *X-Means* (em branco).

Dos gráficos podemos dizer que a diferença entre uma e outra ideia para estimar o número de segmentos de reta é mínima para três distribuições (F, S e X) aplicando os diferentes algoritmos de treinamento, um caso especifico é a distribuição X onde o valor obtido foi 66.1%

para todos os algoritmos de treinamento, sendo a única diferença o desvio padrão, aplicando a busca exaustiva; as respectivas porcentagens de acerto podem ser encontradas na Tabela A.6.

Segundo os dados da tabela antes mencionada, para todos os tamanhos das amostras de todas as distribuições, a busca exaustiva foi a melhor; enquanto ao número de segmentos de reta, na Tabela 5.2 podemos ver um resumo dos segmentos de reta encontrados pelo algoritmo *X-Means* e o melhor resultado obtido para a busca exaustiva, ambos para amostras de tamanho 3200 e para cada distribuição. Na tabela foram ressaltados aqueles números de segmentos de reta que coincidiram para ambas ideias (busca exaustiva e *X-Means*), o qual aconteceu uma única vez para cada algoritmo de otimização testado e para diferentes distribuições, na única distribuição onde não se obtiveram coincidências foi na Distribuição Simples.

Tomando como base os resultados previamente exibidos, podemos concluir que:

(a) Tal como foi apresentado no Capítulo 4, onde propomos duas alternativas para estimar o número de segmentos de reta; podemos dizer que usar o *X-Means* como técnica para determinar quantos segmentos representarão às classes, é uma boa alternativa pois a diferença entre as taxas de classificação para ambas técnicas é mínima e é claro que o tempo de treinamento foi menor.

(b) Embora os números de segmentos encontrados para cada técnica é diferente na maioria dos casos, se encontraram algumas coincidências no número de segmentos e na taxa de acerto para: Dist. X (66.6%,**GD**) e Dist. F (87.5%, **KBM+GD**). Também existem outros casos onde a coincidência de número de segmentos não significou a mesma taxa de acerto, mas a diferença foi mínima como para a Dist. F (87.6% - 87.4% - **MDO+GD**) e finalmente para a Dist. S (67.3% - 63.1%) onde a diferença foi significativa.

(c) É bom destacar que, contudo o número de segmentos seja igual para cada técnica a taxa de acerto as vezes não é a mesma pois os centróides encontrados para cada técnica são diferentes, o qual pode gerar segmentos de reta em diferentes posições e isso produz uma diferente taxa de acerto.

(d) Enquanto ao tempo de treinamento, o algoritmo de *X-Means* representa uma boa alternativa para poupar tempo e conseguir resultados comparáveis com o melhor da busca exaustiva.

	GD		**AG+GD**		**AG**		**MDO+GD**		**KBM+GD**	
Distribuição	**SLS-K**	**SLS-X**	**SLS-K**	**SLS-X**	**SLS-K**	**SLS-X**	**SLS-K**	**SLS-X**	**SLS-K**	**SLS-X**
F	2-3	4-3	2-2	4-3	3-2	4-3	4-3	4-3	4-3	4-3
S	2-1	2-2	2-2	2-2	3-2	2-2	3-4	2-2	3-4	2-2
Simples	4-3	4-4	3-2	4-4	4-3	4-4	3-4	4-4	3-3	4-4
X	2-2	2-2	3-2	2-2	4-4	2-2	2-4	2-2	4-2	2-2

Tabela 5.2: *Comparação entre o número de segmentos de reta para:* SLS-K *que representa o melhor resultado obtido a partir da busca exaustiva e* SLS-X *que representa a aplicação do algoritmo X-Means.*

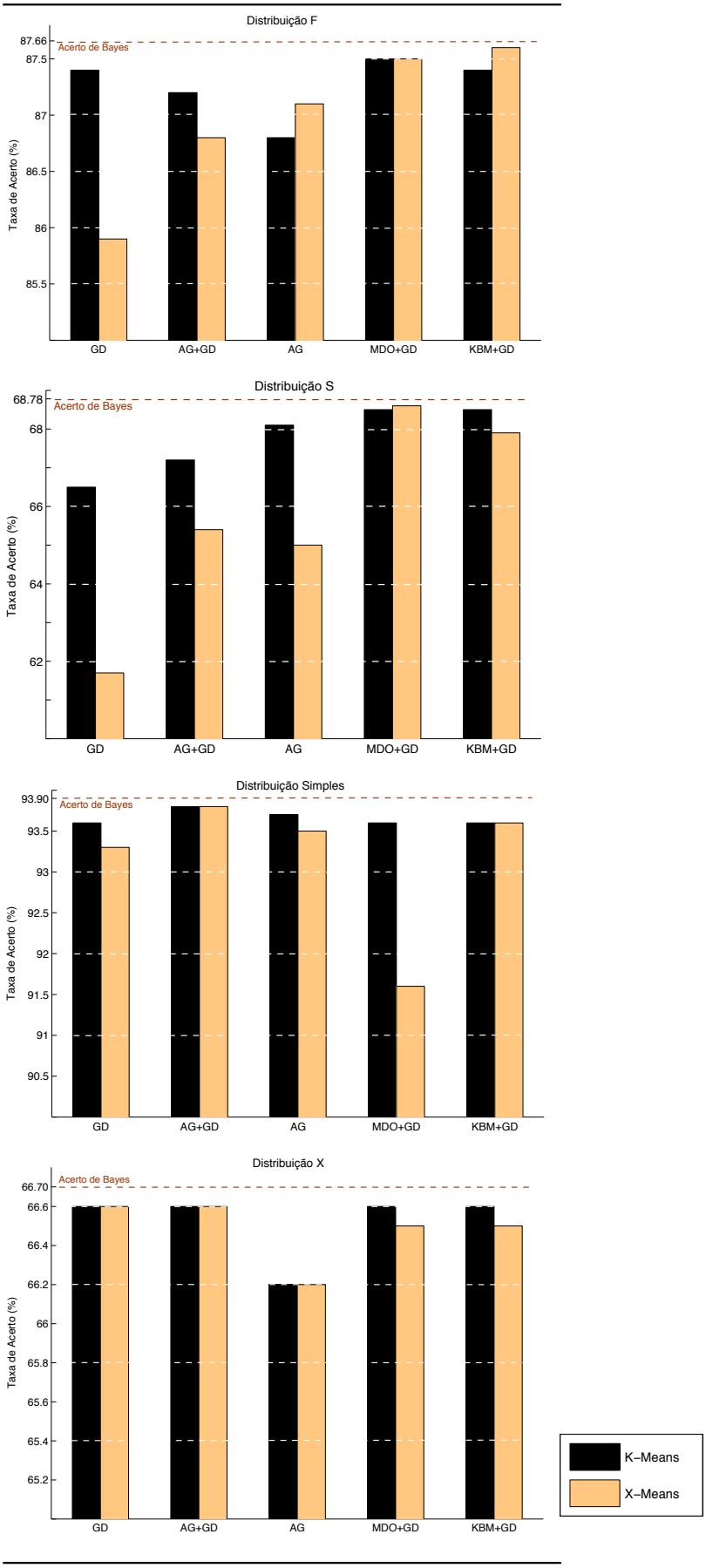


Figura 5.5: *Comparação da taxa de acerto obtida para: (i) Busca exaustiva aplicando o algoritmo de K-Means e (ii) Estimando o número de segmentos de reta aplicando o X-Means; para cada um dos cinco algoritmos de treinamento e as quatro distribuições consideradas no trabalho.*

Para completar os experimentos apresentamos um resumo deles, onde as taxas de acerto de classificação para todos os algoritmos usados neste trabalho são mostradas como uma matriz de valores (entre 0 e 1), que permite gerar uma figura em uma escala da cor verde, onde as cores mais claras são próximas de 0% de acerto e as regiões mais escuras, próximas de 100%. Assim, o resultado é exibido na Figura 5.6. A figura consta de quatro linhas e três colunas, onde cada coluna representa os diferentes parâmetros testados (Param1, Param2 e Param3) e cada linha representa cada uma das distribuições (Dist: F, S, Simples e X). Para cada matriz apresentada na figura grande antes mencionada, temos que: as colunas representam os diferentes algoritmos implementados e, as linhas representam todas as combinações de segmentos de reta testadas na busca exaustiva, já na última linha é apresentado o resultado obtido usando o *X-Means* com cada algoritmo. Enfatizamos que a escala de cores depende dos valores de cada matriz, sendo por isso que cores claras podem representar 0.2 para uma matriz e para outra representar 0.6; também o número de exemplos usados para gerar cada matriz é de 3200, pois para na maioria dos casos, foi com esse tamanho que conseguimos os melhores resultados .

A figura confirma as conclusões obtidas nas subseções anteriores, enquanto ao algoritmos de treinamento, a combinação do Método de Otimização Dialética e Gradiente Descendente (**MDO+GD**), pode ser considerado o melhor dentre todos seguido pela combinação do *K-Beams* e Gradiente Descendente (**KBM+GD**). Embora apresentem porcentagens de acerto similares ou comparáveis consideramos o **MDO+GD** como uma boa alternativa para o treinamento do classificador SLS pois o Método Dialético de Otimização vai reduzindo o número de soluções conforme o algoritmo é executado ao contrário do **KBM+GD** que mantêm sempre o mesmo número de soluções. Finalmente, para os parâmetros testados e considerando esses dois algoritmos de treinamento, poderiam ser usados qualquer uma das variações de parâmetros. Enquanto as duas formas de determinar o número de segmentos de reta, podemos observar na figura também que os resultados com *X-Means* sempre mantêm uma porcentagem alta e comparável com os melhores resultados da busca exaustiva. Para conferir o número de segmentos de reta obtidos pelo *X-Means* para cada uma das matrizes da figura pode-se ver a Tabela 5.2 antes mencionada.

5.3 Resultados para Dados Reais

Para comparar o desempenho dos algoritmos de treinamento propostos neste trabalho, com os resultados publicados nos trabalhos de Ribeiro [2009] e Ribeiro e Hashimoto [2010], realizamos experimentos com oito bases de dados públicas, cujas características podem ser encontradas na Tabela 5.3, listadas a seguir:

1. *Australian Credit Approval* (australian),
2. *Breast Cancer Wisconsin* (breast-cancer),
3. *Pima Indians Diabetes* (diabetes),
4. *German Credit Data* (german),
5. *Heart*,
6. *Ionosphere*,
7. *Liver Disorders* e
8. *Sonar Mines vs Rocks*.

Os experimentos foram realizados da mesma forma descrita no trabalho de Ribeiro [2009]:(i) cada amostra foi separada aleatoriamente em duas partes: uma com 2/3 dos exemplos para o treinamento e outra com 1/3 para o teste; (ii) treinamos o classificador e se calcula a taxa de acerto. Para cada base de dados foi realizado esse processo 10 vezes.

Enquanto aos parâmetros usados nos treinamentos dos dados públicos, testamos os cinco algoritmos de treinamento propostos aplicando o *X-Means* somente para estimar o número de segmentos de reta a serem usados como representantes de cada classe, pois não conhecemos a distribuição dos dados; e usamos os Parâmetros-3 para os algoritmos **AG+GD**, **AG**, **MDO+GD** e **KBM+GD**; para o algoritmo **GD** (Gradiente Descendente) sempre foram usadas 1000 iterações.

A Tabela 5.3, está dividida em 3 partes: a primeira contêm as características dos dados públicos, na segunda parte estão os resultados representados pelas médias percentuais para cada algoritmo e o seu respetivo desvio padrão entre parênteses, seguido do número de segmentos de reta por classe encontrados pelo *X-Means*; e na terceira parte encontramos os melhores resultados obtidos pelo algoritmo original extraídos de Ribeiro [2009], assim mostramos os a média da porcentagem de acerto do Classificador SLS original usando o *K-Means* (indicando o número de segmentos de reta a usar), e o melhor dos resultados obtidos utilizando o SVM (também extraídos do mesmo trabalho), seguido do número de segmentos de reta usados nos testes.

Nesta tabela estão ressaltados em negrito as maiores porcentagens obtidas com nossa proposta, também estão sublinhados os melhores resultados entre o Classificador SLS original e o resultado das Máquinas de Suporte Vetorial. Finalmente foi destacada em um requadro, a melhor porcentagem obtida entre nossa proposta e o algoritmo original.

A partir dos resultados podemos concluir :

(a) Para os dados de *australian*, o melhor resultado foi de 76.0%-87.0% que ficou muito longe dos resultados obtidos com o algoritmo original, o mesmo acontece com *heart*(77.9%-85.1%) e com *breast-cancer*(76.9%-98.1%).

(b) Para os outros dados, os resultados amostraram um melhor comportamento, temos assim, para *diabetes*(81.5%-77.8%) onde os resultados obtidos com o algoritmo **MDO+GD**, atingiram uma porcentagem maior do que as SVM em um **3.7**%. Um outro exemplo é a base de dados *german*(80.9%-76.7%) onde o algoritmo **KBM+GD**, conseguiu uma porcentagem melhor do que o classificador SLS e o SVM em **4.2**%.

(c) O mesmo acontece para *liver disorders*(78.9%-72.7%) usando o algoritmo **AG+GD** com diferença de **4.2**% sobre as SVMs; e *sonar*(88.9%-88.4%) usando o algoritmo **AG+GD** também com diferença de **0.5**% também sobre às SVMs.

(d) Enquanto ao número de segmentos de reta, pode-se notar uma clara diferença entre aqueles problemas onde nossa proposta foi melhor e aqueles onde não foi muito bem, por exemplo, o número de segmentos sempre foi maior e usando diferente número de segmentos para três dos quatro resultados (*diabetes, german e liver-disorders*); mas para *sonar* o número de segmentos foi reduzido de 4-4 para 2-2, mas foi mantido o número de segmentos igual para cada classe. Para os outros dados, o número de segmentos de reta foi reduzido em dois casos, diminuiu de 10 a 2.

(e) Enquanto as conclusões dos dados artificiais, podemos dizer que o algoritmo **MDO+GD** apresentou um bom desempenho em três dos experimentos conseguiu a melhor taxa de

acerto dentre todos os algoritmos, e quando não foi o melhor a diferença entre um e outro não é tão significativa. E, em comparação com o **GD**, que é o Gradiente Descendente, os resultados foram melhores.

(f) Podemos dizer também que o número de segmentos de reta utilizados tem grande influencia nos resultados, pois como podemos observar na tabela de resultados para dados reais, a diferença entre nossos resultados estimando o número de segmentos de reta com o *X-Means* e os resultados obtidos no trabalho de Ribeiro [2009], é grande. Assim, podemos atribuir a baixa taxa de acerto a uma escolha errada do número de segmentos de reta, então não podemos garantir que o *X-Means* vai encontrar, para todos os casos, o número ótimo de segmentos de reta.

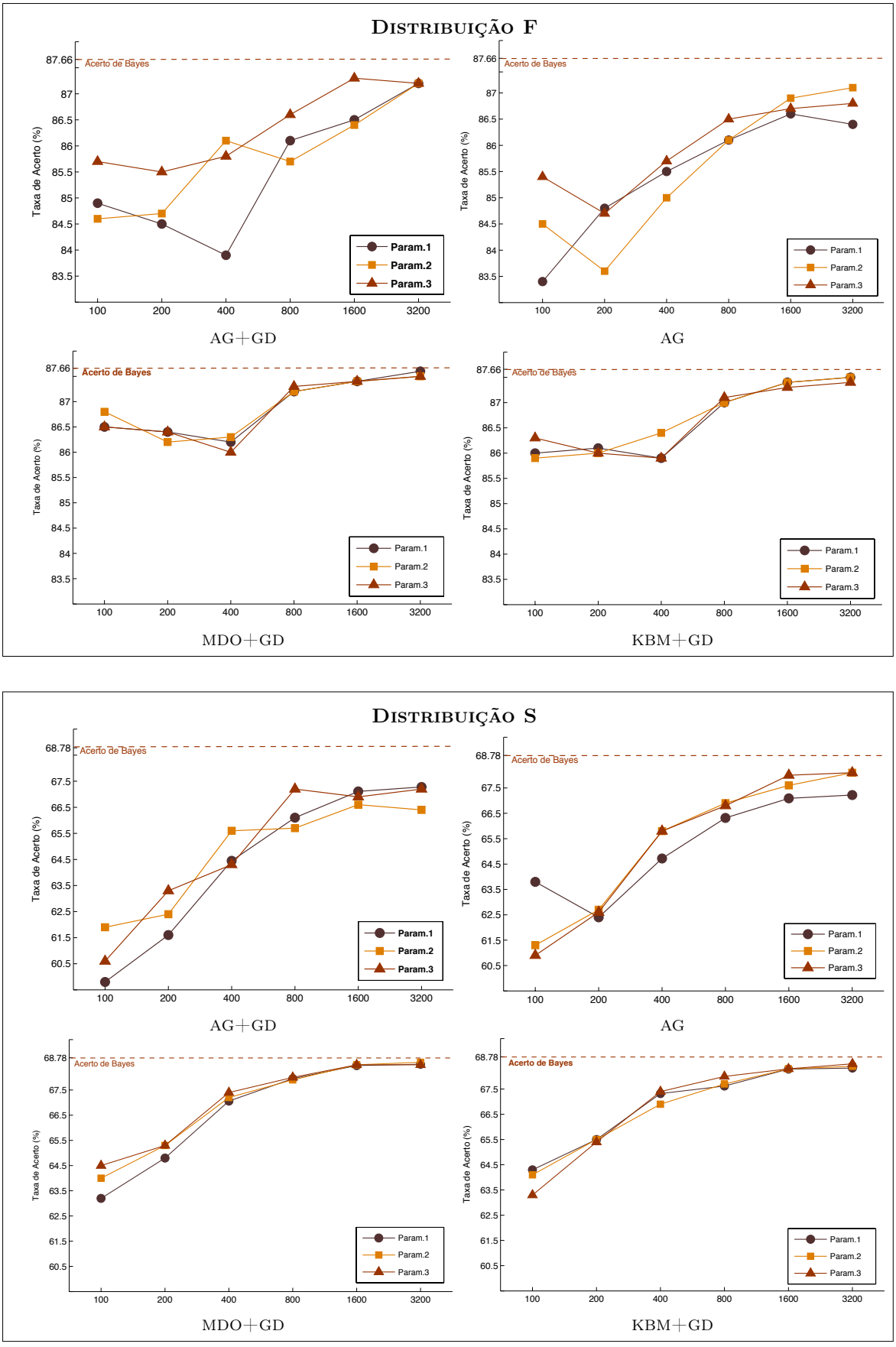


Figura 5.3: *Comparação das taxas de acerto de classificação pelo número de exemplos; dos diferentes algoritmos de treinamento aplicando as três variações de parâmetros propostas no experimento, nas distribuições F e S.*

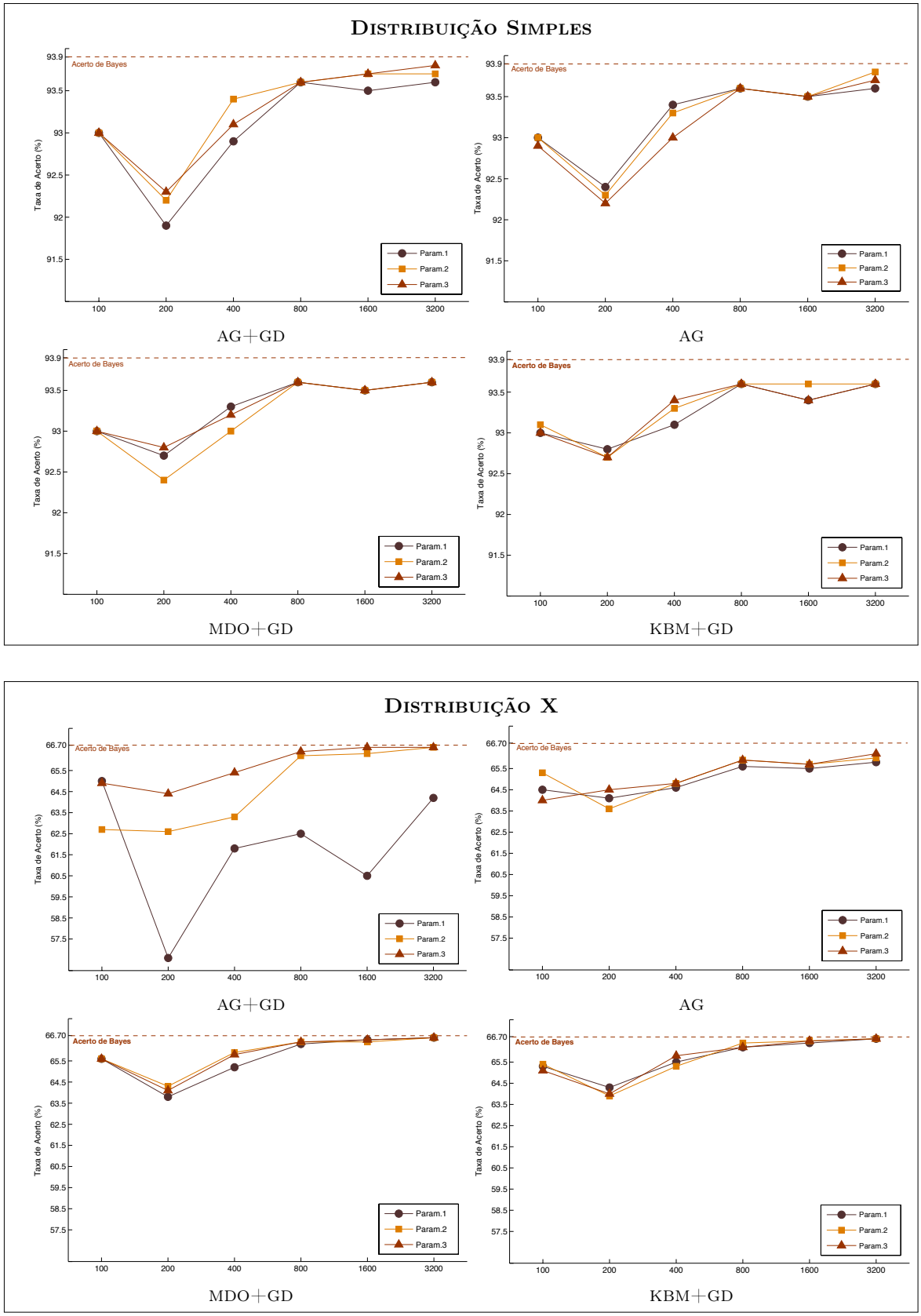


Figura 5.4: *Comparação das taxas de acerto de classificação pelo número de exemplos; dos diferentes algoritmos de treinamento aplicando as três variações de parâmetros propostas no experimento, nas distribuições Simples e X.*

Figura 5.6: *Matrizes de cores representando as porcentagens de acerto para cada distribuição, para cada algoritmo e para cada variação de parâmetros utilizadas nos experimentos realizados, desta forma é possível realizar uma comparação visual dos resultados.*

Tabela 5.3: *Resultados para os Dados Públicos*

				Proposta						Algoritmo Original		
				Algoritmos de Treinamento					X-Means	Melhores %		K-Means
Dados Públicos	**Ref.**	**# Ex.**	**Dim.**	GD	AG+GD	AG	MDO+GD	KBM+GD	SLSS	Class. SLS	SVM	SLSS
AUSTRALIAN	[Quinlan, 1993]	690	14	**76.9** (0.6)	75,0(2,7e-04)	75,0 (2,7e-04)	75,0 (1,9e-04)	75,4 (0,6)	9-1	87.0 (1.8)	87.4 (1.6)	8-8
BREAST-CANCER	[Ryszard *et al*, 1986]	683	9	**76.9** (2.4)	75.2 (0.4)	75.0 (0.0)	75.0 (0.0)	75,0 (0,0)	3-1	98.1 (0.7)	97.9 (0.9)	4-4
DIABETES	[Smith *et al*, 1988]	768	8	75.9 (0.2)	81.2 (0.9)	75.1 (0.03)	**81.5** (1.1)	81.4 (0.7)	**3-5**	76.4 (1.8)	77.8 (1.8)	1-1
GERMAN	[Eggermont *et al*, 2004]	1000	20	76.5 (0.5)	80.5 (0.8)	75.2 (0.03)	80.7 (1.1)	**80.9** (0.7)	**5-3**	76.7 (2.2)	77.3 (0.5)	2-2
HEART	[Smirnov *et al*, 2004]	270	13	75.5 (0.5)	77.0 (1.1)	75.0 (0.01)	**77.9** (1.7)	77.1 (0.9)	3-2	82.2 (3.3)	85.1 (3.3)	10-10
IONOSPHERE	[Sigillito *et al*, 1989]	351	33	81.0 (3.5)	94.7 (1.1)	74.5 (3.7)	**95.1** (1.1)	93.1 (1.8)	2-8	95.2 (2.6)	96.0 (2.1)	10-10
LIVER-DISORDERS	[Bagirov *et al*, 2003]	345	6	75.8 (0.5)	**78.9** (1.9)	75.5 (0.3)	77.3 (1.8)	76.3 (1.2)	**4-2**	70.1 (2.8)	72.7 (2.7)	3-3
SONAR	[Gorman e Sejnowski, 1988]	208	60	80.9 (1.8)	**88.9** (0.8)	75.3 (0.09)	87.6 (2.8)	87.8 (1.4)	**2-2**	86.3 (4.1)	88.4 (4.2)	4-4

Os resultados na tabela representam a média percentual da taxa de acerto e entre parênteses o desvio padrão para 10 testes realizados. Para cada banco de dados público, ou seja, cada linha há um valor em negrito que é o melhor resultado obtido dentre todos os algoritmos de treinamento. Estão sublinhados os melhores resultados entre o algoritmo original do classificador SLS e as SVM. Finalmente, estão destacados as máximas porcentagens de acerto obtidas para cada banco de dados, entre todos os algoritmos testados.

Capítulo 6

Considerações Finais

Neste trabalho, descrevemos a aplicação de algoritmos de otimização diferentes ao Gradiente Descendente, na fase de treinamento do classificador SLS a fim de incrementar a acurácia do mesmo. Para atingir este objetivo, apresentamos no Capítulo 3, três algoritmos híbridos de treinamento que combinam o método de Gradiente Descendente com Algoritmos Genéticos, Método de Otimização Dialética e o *K-Beams*, para encontrar a melhor posição dos SLSs, a fim de minimizar o erro quadrático.

Uma contribuição deste trabalho, apresentado no Capítulo 3, foi o método híbrido de treinamento do GD-MDO (Gradiente Descendente e Método Dialético de Otimização) cujos resultados parciais foram apresentados em 2011, no *SIBGRAPI 2011. 24 Conference on Graphics, Patterns and Images* [Medina e Hashimoto, 2011], onde os bons resultados em quanto ao incremento na acurácia do classificador SLS usando um diferente algoritmo de treinamento, foram atingidos, para distribuições artificiais.

No Capítulo 4 apresentamos duas formas para estimar o número de segmentos de reta a serem usados no Classificador SLS, aplicando uma busca exaustiva ou usando o algoritmo de agrupamento *X-Means*. Mostramos que a diferença do algoritmo original de treinamento do classificador SLS, o uso de diferentes números de segmentos de reta para representar cada classe, pode incrementar a acurácia do mesmo. A pesar de não ter obtido resultados iguais aplicando as duas formas, a segunda opção permite obter porcentagens similares ao melhor resultado da busca exaustiva em pouco tempo. Espera-se agora conseguir uma publicação com os resultados obtidos para esta segunda opção.

No Capítulo 5 apresentamos os resultados experimentais, tanto para dados artificiais como para dados públicos. Os experimentos realizados mostraram que para as distribuições artificiais a taxa de acerto do classificador SLS ficou muito próxima do acerto de Bayes, sendo que para duas distribuições foi possível conseguir uma diferença de menos de 0.5% e para uma delas conseguiu a mesma porcentagem de acerto. Também mostramos que os parâmetros dos algoritmos de otimização utilizados foram estáveis para Parâmetros-3, cujos valores são maiores para iterações e população; mas a diferença entre as três variações de parâmetros apresentadas foi menor a 1%.

Os resultados dos experimentos realizados com bases de dados públicas nos permitiu comparar os resultados do algoritmo original apresentados em Ribeiro [2009], e compará-los com nossos resultados. Assim, mostramos que para dados públicos os métodos híbridos de treinamentos apresentaram uma melhora na acurácia do classificador original, sendo a diferença no melhor dos casos de 6% comparado com o resultado das SVM. Também podemos dizer que o número de segmentos de reta usados, aplicando nossa proposta, é menor do que

o número de segmentos de reta usados no algoritmo original para atingir um erro mínimo.

6.1 Trabalhos futuros

Uma vez apresentados os resultados do nosso trabalho de mestrado, alguns dos trabalhos a futuro que poderiam ser realizados a fim de melhorar o estender projeto são:

1. Usar outros tipos de cálculo de distância.

   Investigar se o uso de diferentes tipos de distâncias além da Euclidiana, até agora utilizada, tem alguma influência significativa na classificação e ao mesmo tempo procurar pelos tipos de problemas nos quais a distância Euclidiana não apresenta um bom desempenho.

2. Estender o classificador SLS a problemas de multiclassificação.

   Um aspecto teórico do problema de multiclassificação é investigar a existência de uma função matemática diferenciável (como feito no caso binário) na qual uma limiarização produzisse como saída da função, a classe a qual pertence um certo dado de entrada. Esta função poderia ser utilizada diretamente com o Gradiente Descendente ou o algoritmo híbrido **MDO+GD**.

3. Avaliar as possíveis aplicações do classificador SLS em outros problemas reais, além dos dados públicos obtidos do UCI.

   Tentar aplicar o classificador em problemas reais com dados obtidos de outros trabalhos do grupo, ou procurar alguma trabalho em colaboração com outras áreas.

4. Investigar e estudar o possível uso dos segmentos de reta como descritores de formas, dadas as propriedades dos segmentos, de ficar posicionados nas regiões de maior concentração de pontos.

Apêndice A

Tabelas de Resultados

Neste capítulo serão descritas todas as tabelas de resultados obtidas após os múltiplos treinamentos e testes, apresentados no Capítulo 5.

Assim, primeiro temos a Tabela A.1, onde são apresentadas as médias percentuais e o desvio padrão para cada uma dos 6 tamanhos das amostras, e a sua vez para cada algoritmo de treinamento implementado. O objetivo principal desta tabela é apresentar uma comparação entre os diferentes algoritmos de treinamento a fim de conhecer qual é o melhor de todos.

Nesta tabela, são apresentados somente os resultados para uma variação de parâmetros (Parâmetros-1) e a forma de estimar o número de segmentos de reta, foi a busca exaustiva. Apresentamos os melhores resultados da busca exaustiva para cada tamanho da amostra e os respectivos números que representam os segmentos de reta a serem usados na representação das classes.

Nos resultados, para cada distribuição, podemos ver ressaltadas em negrito, as duas maiores porcentagens de acerto que foram encontradas dentre todos os algoritmos de treinamento, para cada linha (tamanho de amostra).E encontra-se sublinhado o máximo valor de acerto. Finalmente, está destacado em requadro aquela porcentagem máxima de todas as amostras e algoritmos de treinamento.

A partir destes resultados podemos dizer que o algoritmo de treinamento com melhor desempenho é o **MDO+GD** pois obteve a melhor porcentagem para três das quatro distribuições, para a distribuição atingiu sim, um máximo valor; a única diferença foi o desvio padrão. O segundo melhor algoritmo, foi o **KBM+GD** para três distribuições (F, S e Simples), mas para a distribuição X o segundo melhor foi o **GD**.

Tabela A.1: *Taxas de classificação obtidas aplicando:* (GD) *Gradiente Descendente;* $(AG{+}GD)$ *Algoritmos Genéticos e Gradiente Descendente;* (AG) *Algoritmos Genéticos;* $(MDO + GD)$ *Método de Otimização Dialética e Gradiente Descendente; e* $(KBM + GD)$ *K-Beams e Gradiente Descendente.*

		Porcentagem de Acerto (%) para os diferentes Algoritmos de Treinamento														
		GD			**AG+GD**			**AG**			**MDO+GD**			**KBM+GD**		
	Exemplos	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss
	100	**86.7**	0.04	3-3	84.9	0.67	4-4	83.4	4.60	3-2	**86.5**	0.25	1-4	85.9	0.70	2-4
	200	85.5	0.26	2-3	84.5	1.09	2-2	84.8	1.86	2-2	**86.4**	0.54	2-3	**86.1**	0.85	4-3
F	400	85.7	0.99	3-4	83.9	1.47	4-4	85.5	0.19	4-4	**86.2**	0.58	1-2	**85.9**	0.59	3-3
(87.66%)	800	**87.2**	0.05	2-2	86.1	0.48	2-2	86.1	0.77	4-3	**87.2**	0.07	3-4	**87.0**	0.19	4-4
	1600	**87.3**	0.01	2-3	86.5	0.31	3-3	86.6	0.19	3-3	**87.4**	0.06	2-3	**87.4**	0.01	4-3
	3200	87.4	0.01	2-3	87.2	0.34	2-2	86.4	0.37	3-2	**87.6**	0.02	4-3	**87.5**	0.07	4-3
	100	**65.4**	0.28	2-2	59.8	0.07	1-1	63.8	3.60	3-2	63.2	4.20	1-1	**64.3**	0.70%	2-3
	200	64.3	0.28	3-4	61.6	0.67	3-3	62.4	3.47	4-3	**64.8**	1.88	2-4	**65.5**	0,47	4-3
S	400	64.7	3.81	1-2	64.5	2.36	2-2	64.7	1.50	4-4	**67.1**	0.98	1-2	**67.3**	0.82	2-4
(68.78%)	800	66.4	0.02	2-1	66.1	0.39	3-3	66.3	0.15	2-4	**67.9**	0.11	3-4	**67.6**	0.10	4-3
	1600	67.4	0.61	3-2	67.1	0.84	3-3	67.1	0.69	2-4	**68.5**	0.13	3-2	**68.3**	0.21	3-3
	3200	66.5	0.02	2-1	67.3	1.68	3-3	67.2	0.90	3-3	**68.5**	0.04	3-4	**68.3**	0.17	3-4
	100	**93.2**	0.01	1-2	92.9	0.01	1-1	93.0	0.13	2-1	**93.0**	0.03	4-4	**93.0**	0.06	2-3
	200	92.6	0.25	4-3	91.9	0.69	1-1	92.4	0.10	1-1	**92.7**	0.20	4-3	**92.8**	0.08	4-4
Simples	400	93.2	0.29	4-3	92.9	0.35	3-2	**93.4**	0.18	2-1	**93.3**	0.13	2-1	93.1	0.06	3-2
(93.90%)	800	**93.6**	0.01	2-4	**93.6**	0.01	2-2	**93.6**	0.04	2-2	**93.6**	0.01	3-4	**93.6**	0.10	3-3
	1600	**93.5**	0.02	2-3	**93.5**	0.28	2-2	**93.5**	0.24	4-3	**93.5**	0.11	4-3	93.4	0.06	3-3
	3200	**93.6**	0.01	4-3	**93.6**	0.03	3-2	**93.6**	0.14	4-3	**93.6**	0.01	3-4	**93.6**	0.01	3-3
	100	**66.1**	0.10	2-2	64.9	0.23	1-1	65.1	0.64	1-1	**65.5**	0.09	2-1	65.3	0.54	1-1
	200	**64.6**	1.41	1-1	56.6	1.85	2-3	64.1	0.89	2-2	63.8	1.36	2-2	**64.3**	1.13	1-2
X	400	**65.8**	0.08	1-1	61.8	1.90	4-2	64.6	0.97	2-4	65.2	0.24	2-1	**65.5**	0.11	4-1
(66.70%)	800	**66.5**	0.00	1-1	62.5	1.12	2-3	65.6	0.28	2-4	**66.3**	0.25	1-2	66.2	0.20	3-1
	1600	**66.5**	0.01	1-1	60.5	3.56	4-3	65.5	0.42	4-2	**66.5**	0.08	1-3	66.4	0.05	1-1
	3200	**66.6**	0.01	2-2	64.2	2.47	3-2	65.8	0.12	4-4	**66.6**	0.06	2-4	**66.6**	0.02	4-2

Na próxima Tabela A.2, mostramos uma tabela comparativa do algoritmo híbrido **AG+GD** quando foi executado para as diferentes distribuições artificiais de dados usando as três variações de parâmetros propostas na Tabela 5.1. Dos resultados podemos dizer que a diferença do desempenho do classificador com esse algoritmo de treinamento para diferentes parâmetros não significou melhoras importantes, pois para várias das amostras a porcentagem foi a mesma. Um caso especial foi a distribuição X, onde a diferença entre usar o algoritmo **AG+GD** com Parâmetros-1 e Parâmetros-2 foi de 6%.

Tabela A.2: *Porcentagem de Acerto para o Algoritmo Híbrido de Treinamento (**AG+D**) - Algoritmos Genéticos e Gradiente Descendente. Os valores em negrito representam as maiores porcentagens para cada Parâmetro e destacado em requadro o melhor de todos os resultados.*

	Parâmetros descritos na Tabela 5.1									
		Parâmetros 1			**Parâmetros 2**			**Parâmetros 3**		
	Exemplos	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss
F **(87.66%)**	100	84.9	0.67	4-4	84.6	3.01	4-3	**85.7**	0.28	**2-1**
	200	84.5	1.09	2-2	84.7	0.81	3-3	**85.5**	0.44	**4-2**
	400	83.9	1.47	4-4	**86.1**	0.87	**3-2**	85.8	0.83	3-4
	800	86.1	0.48	2-2	85.7	1.11	2-2	**86.6**	0.31	**4-3**
	1600	86.5	0.31	3-3	86.4	0.07	4-4	**87.3**	0.06	**2-2**
	3200	87.2	0.34	2-2	**87.2**	0.16	**2-2**	**87.2**	0.17	**3-4**
S **(68.78%)**	100	59.8	0.07	2-2	**61.9**	2.00	**1-4**	60.6	1.84	2-3
	200	61.6	0.67	3-2	62.4	0.30	3-2	**63.3**	2.04	**4-2**
	400	64.5	2.36	2-1	**65.6**	1.13	**3-4**	64.3	1.53	3-2
	800	66.1	0.39	1-2	65.7	0.89	3-2	**67.2**	0.93	**4-4**
	1600	**67.1**	0.84	**4-4**	66.6	0.41	4-3	66.9	0.67	4-4
	3200	**67.2**	1.68	**4-4**	66.4	1.52	4-4	**67.2**	0.56	**4-4**
Simples **(93.90%)**	100	**93.2**	0.01	**1-1**	93.0	0.02	2-1	93.0	0.01	4-2
	200	**92.6**	0.25	**1-1**	92.2	0.42	1-3	92.3	0.42	2-2
	400	93.2	0.29	3-2	**93.4**	0.17	**3-3**	93.1	0.47	2-3
	800	**93.6**	0.00	**2-2**	93.6	0.10	4-2	**93.6**	0.01	**2-2**
	1600	93.5	0.02	2-2	93.7	0.15	4-2	**93.7**	0.01	**4-4**
	3200	93.6	0.00	3-2	93.7	0.17	3-2	**93.8**	0.06	**1-1**
X **(66.70%)**	100	**65.0**	0.23	**1-1**	62.7	0.92	3-3	64.9	1.09	1-4
	200	56.6	1.85	2-3	62.6	1.27	1-4	**64.4**	0.69	**1-1**
	400	61.8	1.90	4-2	63.3	1.84	3-3	**65.4**	0.18	**2-2**
	800	62.5	1.12	2-3	66.2	0.10	2-3	**66.4**	0.03	**2-2**
	1600	60.5	3.58	4-3	66.3	0.20	2-2	**66.6**	0.01	**2-3**
	3200	64.2	2.47	3-2	**66.6**	0.02	**2-2**	66.6	0.89	4-2

Os resultados apresentados na Tabela A.3, representam a comparação de taxas de acerto do classificador SLS usando só Algoritmos Genéticos (**AG**), quando foi executado com diferentes parâmetros, podemos observar que os melhores resultados foram obtidos executando o algoritmo como os Parâmetros-2 e Parâmetros-3, e na média a diferença entre os três parâmetros é ao redor de 1% para amostras com tamanhos a partir de 1600.

Tabela A.3: *Porcentagem de Acerto para o Algoritmo de Treinamento (**AG**) - Algoritmos Genéticos. Os valores em negrito representam as maiores porcentagens para cada Parâmetro e destacado em requadro o melhor de todos os resultados.*

	Parâmetros descritos na Tabela 5.1									
		Parâmetros 1			**Parâmetros 2**			**Parâmetros 3**		
	Exemplos	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss
F **(87.66%)**	100	82.7	4.60	3-2	84.5	0.72	3-3	**85.4**	1.14	**3-2**
	200	83.9	1.85	2-2	83.6	2.61	3-4	**84.7**	1.96	**4-2**
	400	85.2	0.19	4-4	85.0	0.43	3-3	**85.7**	0.94	**2-3**
	800	83.4	0.76	4-3	86.1	0.51	4-3	**86.5**	0.17	**4-2**
	1600	86.0	0.19	3-3	**86.9**	0.26	**2-3**	86.7	0.63	2-4
	3200	86.1	0.37	3-2	**87.1**	0.24	**4-3**	86.8	0.33	4-3
S **(68.78%)**	100	**63.8**	3.60	**3-2**	61.3	2.80	2-1	60.9	2.21	2-1
	200	62.4	3.47	4-3	**62.7**	2.94	**1-2**	62.6	3.98	2-3
	400	64.7	1.50	4-4	65.8	0.68	3-3	**65.8**	0.49	**3-2**
	800	66.3	0.15	2-4	**66.9**	0.60	**2-3**	66.8	0.78	1-2
	1600	67.1	0.69	2-4	67.6	0.51	2-3	**68.0**	0.39	**4-4**
	3200	67.2	0.90	3-3	68.1	0.39	3-4	**68.1**	0.07	**2-4**
Simples **(93.90%)**	100	93.0	0.13	4-4	**93.0**	0.03	**2-1**	92.9	0.10	3-3
	200	**92.4**	0.09	**4-3**	92.3	0.26	1-1	92.2	0.43	3-4
	400	**93.4**	0.18	**2-1**	93.3	0.17	3-4	93.0	0.49	4-3
	800	93.6	0.04	3-4	93.6	0.05	4-4	**93.6**	0.04	**1-1**
	1600	93.5	0.23	4-3	93.5	0.21	2-4	**93.5**	0.18	**3-4**
	3200	93.6	0.14	3-4	**93.8**	0.06	**1-1**	93.7	0.09	3-4
X **(66.70%)**	100	64.5	0.64	2-1	**65.3**	0.65	**1-2**	64.0	0.19	1-2
	200	64.1	0.89	2-2	63.6	0.94	1-1	**64.5**	1.55	**2-2**
	400	64.6	0.97	2-1	64.8	0.71	1-1	**64.8**	0.63	**2-2**
	800	65.6	0.28	1-2	**65.9**	0.40	**1-1**	65.9	0.40	3-2
	1600	65.5	0.42	1-3	**65.7**	0.41	**1-1**	65.7	0.48	2-2
	3200	65.8	0.12	2-4	66.0	0.37	1-1	**66.2**	0.24	**1-1**

Para a seguinte Tabela A.4, apresentamos os resultados do mesmo jeito que para os algoritmos de treinamento anteriores; neste caso os melhores resultados também foram obtidos quando foram executados com os Parâmetros-2 e Parâmetros-3, novamente a diferença entre as execuções com diferentes parâmetros não foram significativas, sempre foi em média de 1%, para todas as amostras.

Tabela A.4: *Porcentagem de Acerto para o Algoritmo Híbrido de Treinamento (**MDO+GD**) - Método de Otimização Dialética e Gradiente Descendente. Os valores em negrito representam as maiores porcentagens para cada Parâmetro e destacado em requadro o melhor de todos os resultados.*

	Parâmetros descritos na Tabela 5.1									
		Parâmetros 1			**Parâmetros 2**			**Parâmetros 3**		
	Exemplos	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss
F **(87.66%)**	100	86.5	0.25	1-4	**86.8**	0.29	**1-4**	86.5	0.50	3-4
	200	86.4	0.54	2-3	86.2	0.77	2-4	**86.4**	0.49	**3-2**
	400	86.2	0.58	1-2	**86.3**	0.35	**1-2**	86.0	0.,08	1-2
	800	87.2	0.07	3-4	87.2	0.03	4-2	**87.3**	0.15	**3-2**
	1600	**87.4**	0.06	**2-3**	87.4	0.08	3-3	87.4	0.09	3-4
	3200	**87.6**	0.02	**4-3**	87.5	0.04	4-3	87.5	0.02	3-3
S **(68.78%)**	100	63.2	4.20	1-1	64.0	3.50	4-1	**64.5**	1.79	**1-1**
	200	64.8	1.88	2-4	65.3	1.60	2-2	**65.3**	0.97	**3-2**
	400	67.1	0.98	1-2	67.2	0.27	2-4	**67.4**	0.35	**1-2**
	800	67.9	0.11	3-4	67.9	0.11	3-4	**68.0**	0.07	**3-2**
	1600	68.5	0.13	3-2	68.5	1.03	4-1	**68.5**	0.06	**4-3**
	3200	68.5	0.04	3-4	**68.6**	0.05	**2-4**	68.5	0.05	4-4
Simples **(93.90%)**	100	93.0	0.03	4-4	**93.0**	0.01	**3-2**	93.0	0.03	3-3
	200	92.7	0.20	4-3	92.4	0.24	1-1	**92.8**	0.10	**3-4**
	400	**93.3**	0.13	**2-1**	93.0	0.60	3-4	93.2	0.30	4-3
	800	93.6	0.01	3-4	**93.6**	0.00	**4-4**	93.6	0.02	1-1
	1600	93.5	0.11	4-3	**93.5**	0.03	**2-4**	93.5	0.13	3-4
	3200	93.6	0.01	3-4	**93.6**	0.01	**1-1**	93.6	0.04	3-4
X **(66.70%)**	100	65.6	0.09	2-1	65.6	0.46	1-2	**65.6**	0.07	**2-1**
	200	63.8	1.36	2-2	**64.3**	1.55	**1-1**	64.1	0.86	1-1
	400	65.2	0.24	2-1	**65.9**	0.12	**1-1**	65.8	0.02	1-1
	800	66.3	0.25	1-2	66.4	0.07	1-1	**66.4**	0.01	**1-1**
	1600	66.5	0.08	1-3	66.4	0.08	1-1	**66.5**	0.01	**1-1**
	3200	66.6	0.06	2-4	66.6	0.03	1-1	**66.6**	0.02	**1-1**

Os resultados para o algoritmo de treinamento híbrido **KBM+GD** são apresentados na Tabela A.5, mostramos aqui também que a diferença entre a aplicação de diferentes parâmetros no algoritmo não gera porcentagens de acerto muito diferentes e acontece o mesmo com todas as amostras. Novamente a diferença foi menor de 1%.

Tabela A.5: *Porcentagem de Acerto para o Algoritmo Híbrido de Treinamento (**KBM+GD**) - KBeams e Gradiente Descendente. Os valores em negrito representam as maiores porcentagens para cada Parâmetro e destacado em requadro o melhor de todos os resultados.*

	Conjuntos de Parâmetros descritos na Tabela 5.1									
		Parâmetros 1			**Parâmetros 2**			**Parâmetros 3**		
	Exemplos	Média	Desvio	slss	Média	Desvio	slss	Média	Desvio	slss
F **(87.66%)**	100	86.0	0.70	2-4	85.9	1.20	4-3	**86.3**	0.57	**2-3**
	200	**86.1**	0.85	**4-3**	86.0	0.85	2-2	86.0	0.90	3-4
	400	85.9	0.59	3-3	**86.4**	0.42	**2-3**	85.9	1.10	2-3
	800	87.0	0.19	4-4	87.0	0.06	2-2	**87.1**	0.12	**3-3**
	1600	**87.4**	0.01	**4-3**	87.4	0.06	3-4	87.3	0.10	2-2
	3200	87.5	0.06	4-3	**87.5**	0.02	**3-4**	87.4	0.08	3-3
S **(68.78%)**	100	**64.3**	0.70	**2-3**	64.1	2.31	3-2	63.3	1.39	3-2
	200	**65.5**	0.47	**4-3**	65.5	1.55	2-4	65.4	1.45	3-3
	400	67.3	0.82	2-4	66.9	0.79	2-2	**67.4**	0.54	**4-3**
	800	67.6	0.10	4-3	67.7	0.29	3-4	**68.0**	0.09	**3-2**
	1600	68.3	0.21	3-3	68.3	0.22	3-3	**68.3**	0.09	**3-3**
	3200	68.3	0.17	3-4	68.4	0.29	3-4	**68.5**	0.08	**4-4**
Simples **(93.90%)**	100	93.0	0.06	2-3	**93.1**	0.04	**2-1**	93.0	0.02	2-1
	200	**92.8**	0.08	**4-4**	92.7	0.12	4-3	92.7	0.05	4-3
	400	93.1	0.06	3-2	93.3	0.11	4-3	**93.4**	0.17	**4-3**
	800	93.6	0.10	3-3	93.6	0.02	3-3	**93.6**	0.09	**2-4**
	1600	93.4	0.06	3-3	**93.6**	0.21	**3-4**	93.4	0.06	3-4
	3200	**93.6**	0.01	**3-3**	93.6	0.16	2-3	93.6	0.02	3-3
X **(66.70%)**	100	65.3	0.54	1-1	**65.4**	0.30	**2-3**	65.1	0.49	2-2
	200	**64,3**	1.13	**1-2**	63,9	1.55	4-1	64,0	1.24	1-3
	400	65.5	0.11	4-1	65.3	0.82	1-2	**65.8**	0.05	**1-1**
	800	66.2	0.20	3-1	**66.4**	0.16	**2-1**	66.2	0.10	1-3
	1600	66.4	0,05	1-1	**66.5**	0,01	**1-1**	66.5	0,02	1-1
	3200	66.6	0.02	4-2	66.6	0.02	1-2	**66.6**	0.02	**1-1**

Na Tabela A.6 apresentamos os resultados para todos os algoritmos de treinamento comparando a duas técnicas de estimação de segmentos de reta; é bom ressaltar que na tabela estão os melhores resultados obtidos após uma busca exaustiva e os respetivos números de segmentos de reta para cada classe com os quais conseguiu-se esse resultado (valores em negrito); e os resultados obtidos após aplicar o *X-Means* (valores sublinhados). Mostramos que os resultados aplicando uma busca exaustiva são melhores do que os obtido pelo treinamento com o algoritmo de agrupamento, mas ao mesmo tempo, ambos são comparáveis.

Tabela A.6: *Taxas de classificação obtidas aplicando:* (GD) *Gradiente Descendente;* $(AG{+}GD)$ *Algoritmos Genéticos e Gradiente Descendente;* (AG) *Algoritmos Genéticos;* $(MDO + GD)$ *Método de Otimização Dialética e Gradiente Descendente; e* $(KBM + GD)$ *K-Beams e Gradiente Descendente, comparando as duas técnicas de estimação do número de segmentos de reta.* ***[Acerto de Bayes:(Dist. F: 87.66%, Dist. S: 68.78%, Dist. X: 66.70%)]****.*

		Porcentagem de Acerto (%) para os diferentes Algoritmos de Treinamento														
		GD			**AG+GD**			**AG**			**MDO+GD**			**KBM+GD**		
Ex	**sls-X**	Méd-X	Méd-K	sls-K	Méd-X	Méd-K	sls-K	Méd-X	Méd-K	sls-K	Méd-X	Méd-K	sls-K	Méd-X	Méd-K	sls-K
100	2-2	85.9	86.7	3-3	80.3	84.9	4-4	82.7	83.4	3-2	85.3	86.5	1-4	84.0	85.9	2-4
200	3-2	84.7	85.5	2-3	76.0	84.5	2-2	83.9	84.8	2-2	85.3	86.4	2-3	84.0	86.1	4-3
400	3-3	83.9	85.7	3-4	80.5	83.9	4-4	85.2	85.5	4-4	85.3	86.2	1-2	84.6	85.9	3-3
800	3-2	84.9	87.2	2-2	73.5	86.1	2-2	83.4	86.1	4-3	87.1	87.2	3-4	86.9	87.0	4-4
1600	4-2	83.2	87.3	2-3	82.5	86.5	3-3	86.0	**86.6**	**3-3**	87.4	87.4	2-3	87.2	87.4	4-3
3200	4-3	87.0	**87.4**	**2-3**	81.2	**87.2**	**2-2**	86.1	86.4	3-2	87.4	**87.6**	**4-3**	87.4	**87.5**	**4-3**
100	2-2	63,9	65.4	2-2	45.1	59.8	1-1	58.8	63.8	3-2	62.6	63.2	1-1	62.4	64.3	2-3
200	2-2	63.9	64.3	3-4	44.8	61.6	3-3	62.1	62.4	4-3	65.9	64.8	2-4	64.4	65.5	4-3
400	2-2	59.5	64.7	1-2	63.0	64.5	2-2	61.0	64.7	4-4	66.1	67.1	1-2	65.2	67.3	2-4
800	2-2	59.8	66.4	2-1	45.8	66.1	3-3	65.5	66.3	2-4	67.6	67.9	3-4	67.3	67.6	4-3
1600	2-2	59.8	**67.4**	**3-2**	58.9	67.1	3-3	65.3	67.1	2-4	67.9	68.5	3-2	68,0	68.3	3-3
3200	2-2	59.9	66.5	2-1	63.1	**67.3**	**3-3**	67.1	**67.2**	**3-3**	68.3	**68.5**	**3-4**	66.8	**68.3**	**3-4**
100	2-2	92.8	93.2	1-2	92.9	92.9	1-1	92.0	93.0	2-1	92.9	93.0	4-4	93.0	93.0	2-3
200	2-2	92.3	92.6	4-3	92.4	91.9	1-1	92.0	92.4	1-1	92.5	92.7	4-3	92.2	92.8	4-4
400	3-4	93,1	93.2	4-3	91.4	92.9	3-2	92.8	93.4	2-1	93.2	93.3	2-1	92.9	93.1	3-2
800	4-4	92.7	93.6	2-4	93.5	93.6	2-2	93.3	93.6	2-2	93.5	93.6	3-4	93.5	93.6	3-3
1600	4-4	92.7	93.5	2-3	92.1	93.5	2-2	93.2	93.5	4-3	93.5	93.5	4-3	93.4	93.4	3-3
3200	4-4	92.9	**93.6**	**4-3**	93.4	**93.6**	**3-2**	93.2	**93.6**	**4-3**	93.6	**93.6**	**3-4**	93.5	**93.6**	**3-3**
100	2-2	66.1	66.1	2-2	60.3	**64.9**	**1-1**	63.8	65.1	1-1	65.1	65.5	2-1	65.4	65.3	1-1
200	2-2	64.5	64.6	1-1	46.2	56.6	2-3	60.6	64.1	2-2	63.1	63.8	2-2	61.9	64.3	1-2
400	2-2	65.3	65.8	1-1	59.1	61.8	4-2	63.3	64.6	2-4	64.9	65.2	2-1	64.8	65.5	4-1
800	2-2	66,3	66.5	1-1	55,6	62.5	2-3	64,7	65.6	2-4	66,2	66.3	1-2	66,2	66.2	3-1
1600	2-2	66.3	66.5	1-1	58.2	60.5	4-3	64.2	65.5	4-2	66.3	66.5	1-3	66.3	66.4	1-1
3200	2-2	66.6	**66.6**	**2-2**	48.4	64.2	3-2	65.3	**65.8**	**4-4**	66.6	**66.6**	**2-4**	66.5	**66.6**	**4-2**

Referências Bibliográficas

Abe(2010) S. Abe. *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*. Springer-Verlag New York, Inc., 2a. edição. Citado na pág. 1, 5, 6

Akaike(1976) H. Akaike. An Information Criterion (AIC). *Math Sci*, 14(153):5–9. Citado na pág. 11

Bagirov *et al.***(2003)** A.M. Bagirov, A.M. Rubinov, N.V. Soukhoroukova e J. Yearwod. Unsupervised and Supervised Data Classification via Nonsmooth and Global Optimization. *Top*, 11:1–75. Citado na pág. 56

Bies *et al.***(2009)** B. Bies, K. Dabbs e H. Zou. On Determining the Number of Clusters: A Comparative Study. Citado na pág. 11

Bishop(2006) C.M. Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer New York. Citado na pág. 1

Brugger *et al.***(1972)** W. Brugger, J. M. V. Cantarell e R. Gabás. *Dicionário de Filosofia*. Herder Barcelona. Citado na pág. 15

Cooper(2002) D. E. Cooper. *As Filosofias do Mundo: Uma Introdução Histórica*. Ediçoes Loyola. Citado na pág. 15

Duda *et al.***(2001)** R. O. Duda, P. E. Hart e D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2a. edição. Citado na pág. 1, 4, 6, 30

Eggermont *et al.***(2004)** J. Eggermont, N. K. Joost e W. A. Kosters. Genetic Programming for Data Classification: Partitioning the Search Space. Em *Proceedings of the 2004 Symposium on applied computing (ACM SAC'04)*, páginas 1001–1005. ACM. Citado na pág. 56

Figueiredo(2004) M.A.T. Figueiredo. Lecture notes on bayesian estimation and classification. October 2004. Citado na pág. 5

Fogel(1998) D. B. Fogel. *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1a. edição. Citado na pág. 15

Goldberg(1989) D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1a. edição. Citado na pág. 13

Gorman e Sejnowski(1988) R. P. Gorman e T. J. Sejnowski. Analysis of hidden units in a Layered Network trained to classify sonar targets. *Neural Networks*, 1(1):75–89. Citado na pág. 56

Hamerly e Elkan(2003) G. Hamerly e C. Elkan. Learning the K in K-Means. Em *Neural Information Processing Systems*. MIT Press. Citado na pág. 29, 35

Hammer *et al.***(2004)** B. Hammer, M. Strickert e T. Villman. Relevance LVQ vs. SVM. *Artificial-Intelligence and Soft-Computing - ICAISC 2004, Lecture Notes in Artificial Intelligence*, páginas 592–597. Citado na pág. 6

Haupt e Haupt(2004) R.L. Haupt e S.E. Haupt. *Practical Genetic Algorithms*. J. Wiley & Sons. Citado na pág. 14

Heath(2002) M.T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, 2a. edição. Citado na pág. 13

Heistermann(1992) J. Heistermann. A Mixed Genetic Approach to the Optimization of Neural Controllers. Em *Proceedings of Computer Systems and Software Engineering*, páginas 459–464. IEEE. Citado na pág. 19

Ho e Peyne(2001) Y.C. Ho e D.L. Peyne. Simple Explanation of the No Free Lunch Theorem of Optimization. Em *Proceedings of the 40th IEEE Conference on Decision and Control*, volume 5, páginas 4409–4414. IEEE. Citado na pág. 12

Houck *et al.***(1996)** C.R. Houck, J.A. Joines e M.G. Kay. Comparison of Genetic Algorithms, Random Restart and Two-opt Switching for Solving Large Location-allocation Problems. *Computers Operations Research*, 23(6):587–596. Citado na pág. 13

Jain(2010) A.K. Jain. Data Clustering: 50 years beyond K-Means. *Pattern Recognition Letters*, 31(8):651–666. Citado na pág. 30

Jain *et al.***(2000)** A.K. Jain, R.P.W. Duin e M. Jianchang. Statistical Pattern Recognition: A Review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4 –37. Citado na pág. 4, 5

Jiawei e Kamber(2001) H. Jiawei e M. Kamber. Data Mining: Concepts and Techniques. *San Francisco, CA, itd: Morgan Kaufmann*, 5. Citado na pág. 10

Kotsiantis(2007) S.B. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. Em *Proceeding of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, páginas 3–24. IOS Press. Citado na pág. 1, 3, 5, 10

MacQueen(1967) J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. Em *Proceedings of the 5th Berkeley Symposium on Mathematics Statistics and Probability*. Citado na pág. 10

Medina e Hashimoto(2011) R.A. Medina e R.F. Hashimoto. Combining Dialectical Optimization and Gradient Descent methods for Improving the Accuracy of Straight Line Segment Classifiers. Em *Sibgrapi 2011. 24 Conference on Graphics, Patterns and Images.*, páginas 321–328. IEEE. Citado na pág. 25, 57

Meilă(2006) M. Meilă. The Uniqueness of a Good Optimum for K-Means. Em *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, páginas 625–632. ACM. Citado na pág. 8, 10

Pelleg e Moore(1999) D. Pelleg e A. Moore. Accelerating exact K-Means Algorithms with Geometric Reasoning. Em *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 277–281. ACM. Citado na pág. 2, 11

Pelleg e Moore(2000) D. Pelleg e A. Moore. X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters. Em *Proceeding of the 17th International Conference on Machine Learning*, páginas 727–734. Morgan Kaufmann. Citado na pág. 11, 30

Quinlan(1993) J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1a. edição. Citado na pág. 56

Ribeiro(2009) J.H.B. Ribeiro. *Aprendizado Computacional Baseado em Distância a Segmentos de Reta*. Tese de Doutorado, Instituto de Matemática e Estatística, Universidade de São Paulo Brasil. Citado na pág. 5, 8, 9, 30, 50, 51, 52, 57

Ribeiro e Hashimoto(2006) J.H.B. Ribeiro e R.F. Hashimoto. A New Machine Learning Technique Based on Straight Line Segments. Em *ICMLA 2006: 5th International Conference on Machine Learning and Applications, Proceedings*, páginas 10–16. IEEE Computer Society. Citado na pág. 6

Ribeiro e Hashimoto(2008) J.H.B. Ribeiro e R.F. Hashimoto. A New Training Algorithm for Pattern Recognition Technique Based on Straight Line Segments. Em *Proceedings of the 2008 XXI Brazilian Symposium on Computer Graphics and Image Processing*, páginas 19–26. IEEE Computer Society. Citado na pág. 6, 9, 19, 30

Ribeiro e Hashimoto(2010) J.H.B Ribeiro e R.F. Hashimoto. *Pattern Recognition, Recent Advances*, chapter Pattern Recognition Based on Straight Line Segments. I-Tech. Book Chapter. Citado na pág. 1, 7, 8, 19, 29, 50

Rocha e Goldenstein(2009) A. Rocha e S. Goldenstein. Multi-class from Binary - Divide to Conquer. Em *VISAPP 2009 - Proceedings of the Fourth International Conference on Computer Vision Theory and Applications, Lisboa, Portugal, February 5-8, 2009 - Volume 1*, páginas 323–330. Citado na pág. 4

Russell e Norvig(2002) S. Russell e P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2a. edição. Citado na pág. 13

Ryszard *et al.***(1986)** S. Ryszard, I. Mozetic, J. Hong e N. Lavrac. The Multi-Purpose Incremental System AQ15 and Its Testing Application to Three Medical Domains. Em *AAAI'86*, páginas 1041–1047. Citado na pág. 56

Samuel(1959) A.L. Samuel. Some Studies in Machine Learning using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):211–229. Citado na pág. 3

Santos(2009) W.P. Dos Santos. *Método Dialético de Busca e Otimização para Análise de Imagens de Ressonancia Magnetica*. Tese de Doutorado, Engenharia Eletrica, Universidade Federal de Campina Grande, Brasil. Citado na pág. 15

Santos e Assis(2009) W.P. Dos Santos e F.M. De Assis. Optimization Based on Dialectics. Em *Proceedings of the 2009 International Joint Conference on Neural Networks*, IJCNN'09, páginas 1095–1102. IEEE Press. Citado na pág. 17

Santos *et al.***(2009)** W.P. Dos Santos, F.M. De Assis, R.E. De Souza, P.B. Mendes, H.S.S Monteiro e H.D. Alves. Dialectical Non-Supervised Image Classification. Em *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*, CEC'09, páginas 2480–2487. IEEE Press. Citado na pág. 15

Schwarz(1978) G. Schwarz. Estimating the Dimension of a Model. *The annals of statistics*, 6(2):461–464. Citado na pág. 11, 35

Shahryar *et al.***(2007)** R. Shahryar, H.R. Tizhoosh e M.A. Salama. A Novel Population Initialization Method for Accelerating Evolutionary Algorithms. *Computers Mathematics with Applications*, 53(10):1605 – 1614. Citado na pág. 16

Sigillito *et al.***(1989)** V. G. Sigillito, S. P. Wing, L. V. Hutton e K. B. Baker. Classification of Radar returns from the Ionosphere using Neural Networks. *Johns Hopkins APL Technical Digest*, páginas 262–266. Citado na pág. 56

Smirnov *et al.***(2004)** E. Smirnov, I. Sprinkhuizen-Kuyper e G. Nalbantov. Unanimous Voting using Support Vector Machines. Em *BNAIC-2004: Proceedings of the Sixteenth Belgium-Netherlands Conference on Artificial Intelligence*, páginas 43–50. Citado na pág. 56

Smith *et al.***(1988)** J. W. Smith, J. E. Everhart, W.C. Dickson, W.C. Knowler e R.S. Johannes. Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, páginas 261–265. Citado na pág. 56

Vapnik(1995) V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc. Citado na pág. 5

Vatsavaia *et al.***(2011)** R.R. Vatsavaia, C.T. Chandolaa, V. Junb e G. Junb. GX-Means: A Model-Based Divide and Merge Algorithm for Geospatial Image Clustering. *Procedia Computer Science*, 4:186–195. Citado na pág. 29

Vazquez(2007) A.S. Vazquez. Filosofia da Práxis. *São Paulo: Expressão Popular*, página v02. Citado na pág. 16

Weise(2008) T. Weise. Global Optimization Algorithms - Theory and Application. *URL: http://www.it-weise.de, Abrufdatum*, 1:24. Citado na pág. 14, 15

Wolpert(1996) D. H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390. Citado na pág. 19, 20

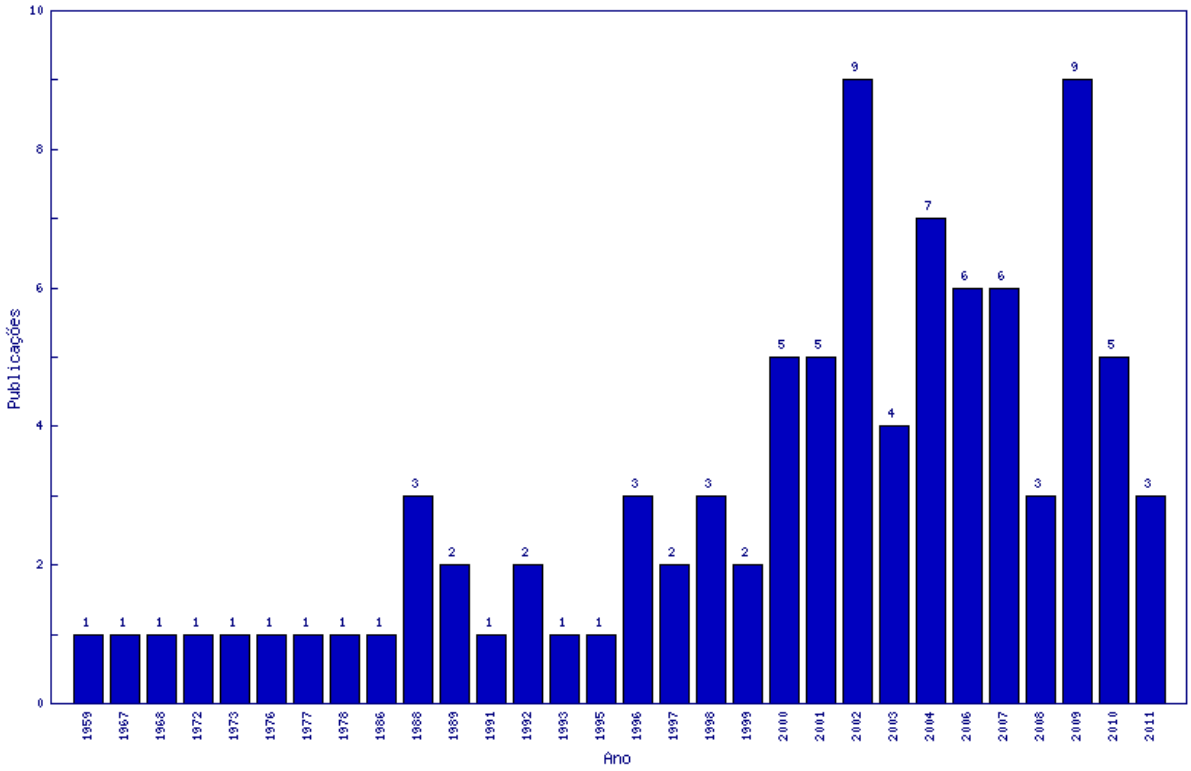
Wolpert(2001) D. H. Wolpert. The Supervised Learning No-Free-Lunch Theorems. *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, 6(1):1–20. Citado na pág. 19, 20

Wolpert e Macready(1997) D.H. Wolpert e W.G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82. Citado na pág. 12

Wright(2010) S. Wright. Optimization Algorithms in Machine Learning. *NIPS Tutorial*. Citado na pág. 7, 14

Wu e Vipin(2009) X. Wu e K. Vipin. *The Top Ten Algorithms in Data Mining*. Chapman Hall. Citado na pág. 6

Xuefeng *et al.***(2009)** C. Xuefeng, L. Xiabi e J. Yunde. Combining Evolution Strategy and Gradient Descent Method for Discriminative Learning of Bayesian Classifiers. Em *Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation*, GECCO '09, páginas 507–514. ACM. Citado na pág. 20


Publicações
Ano
0
2
4
6
8
10
1959 1967 1968 1972 1973 1976 1977 1978 1986 1988 1989 1991 1992 1993 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2006 2007 2008 2009 2010 2011
1 1 1 1 1 1 1 1 1 3 2 1 2 1 1 3 2 3 2 5 5 9 4 7 6 6 3 9 5 3