Pós-Graduação em Ciência da Computação

TELMO DE MENEZES E SILVA FILHO

CLASSIFICAÇÃO BASEADA EM PROTÓTIPOS DE DECISÃO MAIS PRÓXIMOS E DISTÂNCIAS ADAPTATIVAS

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

Telmo de Menezes e Silva Filho

CLASSIFICAÇÃO BASEADA EM PROTÓTIPOS DE DECISÃO MAIS PRÓXIMOS E DISTÂNCIAS ADAPTATIVAS

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

ORIENTADORA: **PROFA. DRA. RENATA MARIA C. R. DE SOUZA**
CO-ORIENTADOR: **PROF. DR. RICARDO B. C. PRUDÊNCIO**

RECIFE
2017

Catalogação na fonte
Bibliotecária Denise Figueiredo Mendes CRB4-1368

S586c Silva Filho, Telmo de Menezes e.
Classificação baseada em protótipos de decisão mais próximos e distâncias adaptativas/ Telmo de Menezes e Silva Filho – 2017.
127 f.: fig., tab.

Orientadora: Renata Maria C.R. de Souza.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da computação, 2017.
Inclui apêndices e referências.

1. Inteligência computacional. 2. Aprendizado de computador. 3. Inteligência artificial. I. Souza, Renata Maria C.R. de. (Orientadora). II. Titulo.

006.3 CDD (22. ed.) UFPE-MEI 2018-12

Telmo de Menezes e Silva Filho

Classificação Baseada em Protótipos de Decisão Mais Próximos e Distâncias Adaptativas

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação

Aprovado em: 22/09/2017.

_____

Orientadora: Profa. Dra. Renata Maria Cardoso Rodrigues de Souza

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Lorena Inacio de Oliveira
Centro de Informática / UFPE

_____

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
Instituto de Ciências Matemáticas e de Computação / USP

_____

Prof. Dr. Carmelo José Albanez Bastos Filho
Escola Politécnica de Pernambuco / UPE

_____

Prof. Dr. Byron Leite Dantas Bezerra
Escola Politécnica de Pernambuco / UPE

_____

Prof. Dr. Carlos Wilson Dantas de Almeida
Centro de Engenharia Elétrica e Informática / UFCG

Aos meus pais e ao povo brasileiro,
por me sustentar enquanto me divirto fazendo ciência.

Agradecimentos

Minha gente, que alegria
É chegado o grande dia
Uma bela de uma conquista
Agora torno-me cientista

Muito tenho a agradecer
Obrigado irei sempre dizer
Reconheço Deus, senhor
Por meus pais e seu amor

Obrigado, família querida
Renata e Ricardo, orientadores
Todos que estavam na torcida
Inspiração que merece flores

Sempre lembro dos amigos
Ofereço gratidão
Carrego portanto comigo
Rios de apoio e superação

Essa etapa em minha vida
Será ponto de partida
Continuarei me esforçando
E com vocês, sempre avançando

Para Luíza, falarei
Realmente, eu tive sorte
Infinita sorte, eu sei
Nós dois juntos somos fortes

Cantei aqui felicidade
E o apoio de vocês
São parceiros de verdade
Assim vencemos mais uma vez

O indivíduo nasce, cresce
E adentra o mundo social e político,
Filosófico e artístico.
Fica danado, letrado,
Inteligente e sabido.

—MARCONDES FALCÃO MAIA

Resumo

A aprendizagem de máquina é um ramo da inteligência artificial, cujo objetivo é desenvolver algoritmos capazes de aprender a partir de dados a fim de realizar diferentes tarefas, como por exemplo, classificação e estimação de probabilidades de classe supervisionadas e semi-supervisionadas. Essas tarefas podem ser realizadas de forma intuitiva e com predições interpretáveis pelos métodos baseados em protótipos. Quanto a esses métodos, é preciso considerar dois pontos importantes: (i) são suscetíveis a mínimos locais causados pela má inicialização dos protótipos e (ii) são sensíveis à distância escolhida para comparar protótipos e instâncias, pois essa precisa ser capaz de modelar a variabilidade interna dos protótipos e classes para alcançar um bom desempenho. Assim, este trabalho visa a explorar a versatilidade dos métodos baseados em protótipos para apresentar soluções para as tarefas de classificação supervisionada e semi-supervisionada, ao mesmo tempo em que apresenta soluções para os dois pontos mencionados acima, principalmente na forma de novas distâncias adaptativas. Para a primeira tarefa, este trabalho introduz um novo método que apresenta uma solução para o problema dos mínimos locais e usa uma distância generalizada aplicada a dados intervalares, capaz de modelar classes desbalanceadas e sub-regiões de classe de diferentes formas e tamanhos. Esse algoritmo também é capaz de eliminar protótipos inativos e selecionar atributos automaticamente. Para a tarefa de classificação semi-supervisionada, este trabalho propõe um algoritmo de propagação de rótulos através de grafos que, ao contrário dos métodos presentes na literatura, não foca apenas na classificação de instâncias não-rotuladas, mas sim na predição de probabilidades de classe apropriadas. Este trabalho também provê uma análise de desempenho dos dois métodos propostos, comparando-os a métodos existentes, em termos de taxa de erro de classificação (primeiro método) e funções de escore apropriadas (segundo método), usando conjuntos de dados reais e sintéticos. Experimentos mostram que ambos os métodos apresentam desempenhos significativamente superiores ao estado da arte.

Palavras-chave: Classificação supervisionada. Classificação semi-supervisionada. Estimação de probabilidades de classe. Protótipos. Distâncias adaptativas.

Abstract

Machine learning is a subfield of artificial intelligence, whose goal is to develop algorithms that are able to learn from data in order to perform different tasks, such as supervised and semi-supervised classification and probability estimation. These tasks can be performed intuitively and with interpretable predictions by prototype-based methods. Regarding these methods, one needs to consider two important points: (i) they are susceptible to local minima due to poor prototype initialization and (ii) they are sensible to the distance that is chosen to compare prototypes and samples, because it has to be able to model the internal variability of prototypes and classes to perform well. Therefore, this work aims at exploring the versatility of prototype-based methods to provide solutions to the tasks of supervised and semi-supervised classification, while also presenting solutions to both points mentioned above, especially regarding new adaptive distances. For the first task, this work introduces a new method that provides a solution to the local minima problem and uses a generalized distance applied to interval data, which is capable of modeling imbalanced classes and class subregions with different shapes and sizes. This algorithm is also capable of eliminating inactive prototypes and automatically selecting features. For the semi-supervised classification task, this work proposes a graph-based label propagation algorithm, which, in contrast to existing methods from literature, does not focus only on unlabeled instance classification, but on the prediction of proper class probabilities. This work also provides a performance analysis of the two proposed methods, comparing them to existing algorithms, in terms of classification error rate (first method) and proper scoring rules (second method), using real and synthetic datasets. Experiments show that both methods perform significantly better than the state of the art.

Keywords: Supervised classification. Semi-supervised classification. Class-probability estimation. Prototypes. Adaptive distances.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

4Pros	Método *proper probability propagation on prototypes*	90
ABC	Método *artificial bee colony*	52
ADS	Análise de dados simbólicos	55
AGR	Método *anchor graph regularization*	31
AK-médias	Método K-médias adaptativo	93
BS	Escore de Brier	92
CE	Entropia cruzada	92
EAGR	Método *efficient anchor graph regularization*	32
GMM	Método de misturas de Gaussianas	28
IDPSO	Método *improved self-adaptive particle swarm optimization*	67
ISOPC	Método *interval swarm-optimized prototype classifier*	52
IVABC	Método *interval velocity-based artificial-bee classifier*	52
KDE	Estimação de densidades usando kernels	28
kNN	Método k-vizinhos mais próximos	21
LVQ	Família de métodos *learning vector quantization*	28
LVQ1	Método *learning vector quantization* original	28
LVQ2	Segunda variante do método *learning vector quantization*	28
LVQ3	Terceira variante do método *learning vector quantization*	28
LVQ-Batch	Método *batch learning vector quantization*	28
OLVQ1	*learning vector quantization* com taxa de aprendizado adaptativa	28
PSO	Método *particle swarm optimization*	45
SVM	Máquina de vetores de suporte	28
VABC	Método *velocity-based artificial-bee colony*	52
WFLVQ	Método *weighted interval fuzzy learning vector quantization*	39

Lista de Símbolos

$\Im$	Conjunto de treinamento	20
N	Número de instâncias de treinamento	20
i	i-ésimo objeto	20
x_i	Vetor de atributos do i-ésimo objeto	20
y_i	Rótulo do i-ésimo objeto	20
p	Número de variáveis	20
g	g-ésima classe	20
G	Número de classes	20
k	Número de vizinhos do kNN	21
Ω_g	Conjunto de protótipos da g-ésima classe	27
M_g	Número de protótipos da g-ésima classe	27
Ω	Conjunto de protótipos	27
m	m-ésimo protótipo	27
M	Número total de protótipos	27
n_l	Número de instâncias rotuladas	31
n_u	Número de instâncias não-rotuladas	31
$\mathbf{Z}$	Matriz de pesos locais	31
$\mathbf{E}$	Matriz de adjacências	31
Λ	Matriz diagonal da soma dos pesos locais dos protótipos	32
$\mathbf{Y}_{n_l}$	Matriz de rótulos para as instâncias rotuladas	32
$\mathbf{A}$	Matriz de predições de rótulos para os protótipos	32
$Q(\mathbf{A})$	Critério da propagação de rótulos	32
ξ	Peso para a importância do segundo termo do critério $Q(\mathbf{A})$	32
$\tilde{\mathbf{L}}$	Matriz Laplaciana reduzida do método AGR	32
$\mathbf{D}$	Matriz diagonal da soma das linhas da matriz $\mathbf{E}$	33
dp_m	Distância usada pelo método WFLVQ	39
$\mathrm{d}_{m,L}$	Distância Euclidiana quadrática adaptativa entre os limites inferiores	39
$\mathrm{d}_{m,U}$	Distância Euclidiana quadrática adaptativa entre os limites superiores	39

$\lambda_{m,j}$	Peso da j-ésima variável do m-ésimo protótipo (WFLVQ)	39
$\Delta_{m,j}$	soma das diferenças quadráticas (WFLVQ)	39
u_{im}	grau de pertinência difuso (WFLVQ)	40
$\gamma_{g,m}$	Peso do m-ésima protótipo da g-ésima classe (WFLVQ)	39
β_g	Peso da g-ésima classe (WFLVQ)	40
l	l-ésima partícula	46
$\mathbf{W}_l$	Posição da l-ésima partícula	46
$\mathbf{V}_l$	Velocidade da l-ésima partícula	46
t	Iteração de treinamento	46
SN	Número de partículas	46
ω	Inércia para o cálculo da velocidade	46
c_1	Coeficiente de aceleração para o cálculo da velocidade	46
c_2	Coeficiente de aceleração para o cálculo da velocidade	46
r_1	Valor aleatório local para o cálculo da velocidade	46
r_2	Valor aleatório global para o cálculo da velocidade	46
ψ_1	Critério da taxa de erro para métodos de enxames	47
ψ_2	Critério da soma das distâncias para métodos de enxames	47
ψ_3	Critério híbrido para métodos de enxames	47
$N_{erros,l}$	Número de erros da l-ésima partícula	47
$\delta(m,l,i)$	Função que indica igualdade de rótulos	47
p_l	Probabilidade de seleção da l-ésima partícula	54
π	Número de variáveis selecionadas pelo método IVABC	60
$\psi_{IVABC}(l)$	Critério de avaliação do método IVABC	61
$J_{IVABC}(l)$	Soma das distâncias do método IVABC	61
$d\pi_{m,l}$	Distância adaptativa do método IVABC	61
$N_{acertos,l}$	Número de acertos da l-ésima partícula	62
Υ_l	Conjunto de variáveis usadas pela l-ésima partícula	62
$max(j_U)$	Máximo limite superior dos intervalos da j-ésima variável	62
$min(j_L)$	Mínimo limite inferior dos intervalos da j-ésima variável	62
$\lambda_{m,l,j}$	Peso da j-ésima variável do m-ésimo protótipo da l-ésima partícula	62
$\gamma_{m,l,g}$	Peso do m-ésimo protótipo da l-ésima partícula na g-ésima classe	63
$\beta_{l,g}$	Peso da g-ésima classe	63
$\Theta_{g,i}$	Peso da g-ésima classe para a i-ésima instância	64
$\theta_{m,l,i}$	Peso do m-ésimo protótipo da l-ésima partícula para a i-ésima instância	64
s	Função de escore apropriada	91

Sumário

1 INTRODUÇÃO

Não podemos ter medo de não saber. O que devemos recear é o não termos inquietação para passarmos a saber.

—MIA COUTO

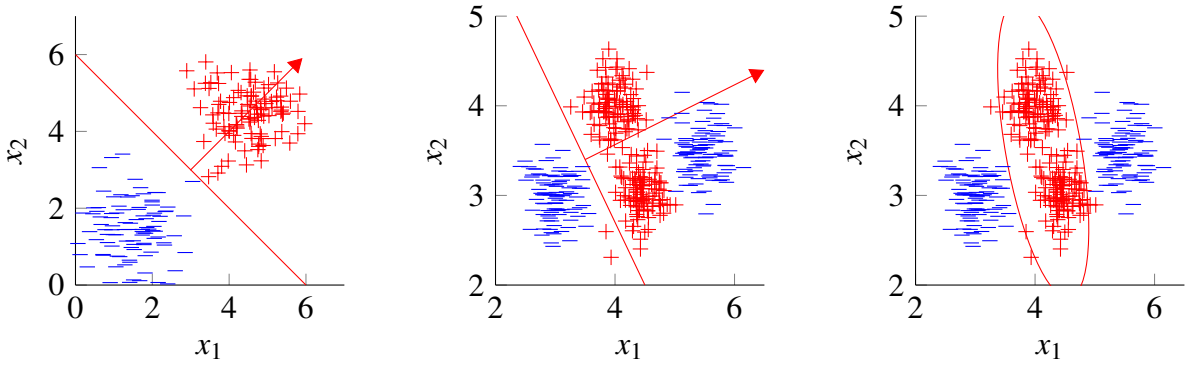
1.1 Motivação

Este trabalho busca explorar a versatilidade dos métodos baseados em protótipos e o uso das distâncias adaptativas para resolver diferentes tarefas de aprendizagem de máquina. Aprendizagem de máquina é um campo da ciência da computação, responsável por criar algoritmos capazes de fazer um computador aprender de forma autônoma a partir de dados[48]. A aprendizagem de máquina tem três componentes básicos: a tarefa a ser resolvida, o modelo gerado através de um algoritmo e os atributos dos quais obtém-se o aprendizado incorporado pelo modelo para a solução da tarefa[24].

Os tipos de tarefas que os algoritmos de aprendizagem de máquina podem resolver são inúmeros, incluindo classificação (binária, multi-classe ou multi-rótulo), ranqueamento, estimação de probabilidades de classe, regressão e abordagens não-supervisionadas[24].

Este trabalho foca nas tarefas de classificação e estimação de probabilidades de classe. A tarefa de classificação pode ser formalizada como segue: seja um conjunto de treinamento $\Im = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_i, y_i), \ldots, (\vec{x}_N, y_N)\}$ com N objetos, cujo i-ésimo objeto é uma tupla $(\vec{x}_i, y_i)$, composta por um vetor $\vec{x}_i$ com p valores obtidos de $\mathscr{X} \in \mathscr{R}^p$ e um rótulo de classe y_i, que toma valores de $\mathscr{Y} = \{1, \ldots, g, \ldots, G\}$; o objetivo é usar o conjunto de treinamento $\Im$ para aprender uma função $f : \mathscr{X} \to \mathscr{Y}$ que atribua rótulos de classe y a novos vetores $\vec{x}$.

Quando $G = 2$, a tarefa de classificação é binária (positivos e negativos). A forma mais simples de resolver essa tarefa é encontrar uma função f tal que, para $f(\vec{x}) < 0$, $y = -$, para $f(\vec{x}) > 0$, $y = +$ e, se $f(\vec{x}) = 0$, y é escolhido aleatoriamente. Como mostra a Figura 1.1a, na prática, essa solução equivale a traçar um hiperplano que separa as classes, posicionado com o objetivo de minimizar o número de objetos classificados erroneamente.



(a) Conjunto linear (b) Conjunto não-linear (c) Conjunto não-linear

Figura 1.1: **Conjuntos linearmente e não-linearmente separáveis** - (a) classes podem ser separadas de maneira ótima por uma reta; (b) classes não podem ser separadas de maneira ótima por uma reta; os objetos da classe negativa que estão à direita e acima da reta serão classificados como positivos; (c) classe positiva pode ser modelada por uma elipse; objetos fora da elipse são classificados como negativos

É fácil ver, porém, que métodos lineares não são capazes de modelar certos conjuntos de dados de forma satisfatória. A Figura 1.1b apresenta um simples conjunto de dados que não pode ser separado por uma reta de forma trivial. Nesse caso, uma das formas de resolver o problema é aplicar um método não-linear. Como mostra a Figura 1.1c, métodos não-lineares são capazes de encontrar formas complexas para separar as classes.

De interesse especial para este trabalho são os classificadores que empregam distâncias para encontrar a função f, entre os quais o mais conhecido e mais antigo é o algoritmo dos k-vizinhos mais próximos (kNN)[1]. Quando uma nova instância $\vec{x}$ é apresentada ao kNN, ele calcula as distâncias entre $\vec{x}$ e todas as N instâncias do conjunto de treinamento, $d(\vec{x},\vec{x}_i), \forall i \in \{1,\ldots,N\}$, e atribui $y = y_c$, onde y_c é a classe mais frequente entre os k-vizinhos mais próximos de $\vec{x}$. O único parâmetro do kNN é o número k de vizinhos, mas seu valor é importante. Valores pequenos de k podem levar a sobreajuste e valores altos podem sujeitar o modelo à influência de ruídos nos dados.

Algoritmos baseados em protótipos [3, 35, 34, 38, 39, 65, 61, 62, 63, 64, 65] compartilham a característica do kNN de atribuir classes a novas instâncias de acordo com seus vizinhos mais próximos. No entanto, ao invés de usar o conjunto de dados inteiro para classificar novos objetos, eles selecionam instâncias, os protótipos, para representar as classes, tarefa similar à dos algoritmos de agrupamento. A Figura 1.2 apresenta um exemplo de conjunto de dados com protótipos posicionados entre as instâncias.

Os protótipos são iterativamente reposicionados de forma que fiquem mais próximos dos

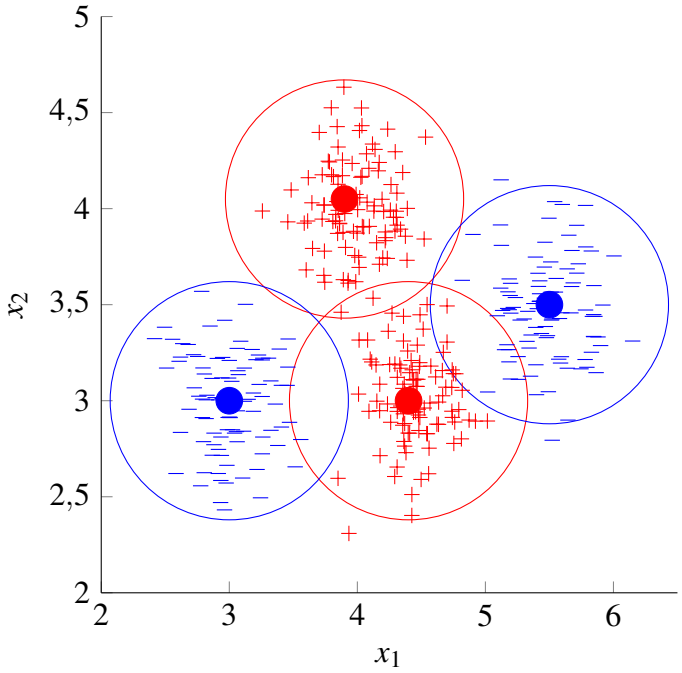


Figura 1.2: **Conjunto bidimensional com protótipos** - os protótipos são representados pelos discos preenchidos; círculos representam áreas de influência dos protótipos; os objetos são rotulados com a classe do protótipo mais próximo

objetos das suas classes e mais distantes dos objetos das demais classes [39]. Para classificar novas instâncias, o método determina a classe vencedora de acordo com a(s) classe(s) do(s) protótipo(s) mais próximo(s). Portanto, as predições feitas por esses algoritmos tendem a ser intuitivas e de fácil interpretação. Esses métodos são suscetíveis a atingir mínimos locais devido à inicialização dos protótipos, pois, caso o conjunto inicial de protótipos seja mal posicionado, o algoritmo pode não conseguir evitar estacionar em um estado ruim. Assim, é importante investigar alternativas para mitigar esse problema [64].

Uma alternativa para o problema da inicialização é usar métodos populacionais de otimização, e.g. métodos baseados em enxames [21, 36, 37] ou computação evolucionária [49], que inicializam várias soluções para um problema simultaneamente e buscam melhorar cada solução iterativamente. Esses algoritmos tem uma boa capacidade de evitar mínimos locais e podem ser adotados como métodos para treinar protótipos, resultando em maior robustez contra má inicialização dos protótipos [15].

Além da questão dos mínimos locais, outro fator importante para a qualidade da solução encontrada tanto pelo kNN quanto por métodos baseados em protótipos é a distância adotada. A distância mais usada para esses métodos é a distância Euclidiana. No entanto, essa distância não é capaz de modelar diferenças nas variabilidades das variáveis. No exemplo apresentado

pela Figura 1.2, cada protótipo é responsável por uma região esférica dos dados. Em várias aplicações do mundo real, isso pode ser insuficiente para modelar o problema. Assim, vários trabalhos vêm sendo dedicados a desenvolver distâncias capazes de modelar regiões de dados de diferentes formas e tamanhos [14, 17, 18, 27, 61, 62, 63, 64, 65].

A versatilidade dos protótipos permite que eles sejam aplicados a tarefas diferentes. Por exemplo, ao contrário dos conjuntos de dados apresentados nas Figuras 1.1-1.2, nos quais todas as instâncias tem rótulos conhecidos, em certas tarefas de classificação, há um grande volume de instâncias não-rotuladas e apenas algumas instâncias são rotuladas. Para incorporar a informação contida nas instâncias não rotuladas, surgiu a aprendizagem semi-supervisionada [5, 11, 33, 83, 28, 29, 45, 69, 70, 71, 77, 81]. Dentro do ramo da aprendizagem semi-supervisionada, fazem parte do escopo deste trabalho os algoritmos que constroem grafos usando protótipos e propagam rótulos de classe entre as instâncias usando o grafo [9, 43, 74].

Esses métodos de propagação de rótulos através de grafos de protótipos foram desenvolvidos para obter soluções eficientes, sem perder o bom desempenho de classificação obtido por outros algoritmos de propagação de rótulos. Porém, o seu objetivo é apenas obter uma boa classificação das instâncias não-rotuladas. Portanto, tais métodos não fornecem uma boa solução para a tarefa de estimação de probabilidades de classe. Dada a importância dessa tarefa em várias aplicações do mundo real, é necessário desenvolver um novo algoritmo que seja capaz de realizar propagação de probabilidades de classe.

1.2 Objetivos

O objetivo geral deste trabalho é apresentar novos métodos de classificação baseados em protótipos e distâncias adaptativas. A versatilidade desses métodos permite fornecer novas soluções para duas importantes tarefas da aprendizagem de máquina: classificação supervisionada e semi-supervisionada, essa última envolvendo estimação de probabilidades. O uso das distâncias adaptativas permite modelar a variabilidade presente nos dados e melhorar o desempenhos de classificação.

Nesse contexto de classificação, os objetivos específicos dessa tese são:

- Para a tarefa supervisionada, propor um método baseado em protótipos usando um algoritmo de otimização de enxames a fim de mitigar o problema da má inicialização dos protótipos que leva a mínimos locais.

- Para a tarefa de classificação semi-supervisionada com estimação de probabilidades de

classe, introduzir um novo algoritmo de propagação de probabilidades através de grafo de protótipos.

- Propor novas distâncias adaptativas que solucionem os problemas de distâncias propostas anteriormente, como perda de informação e ambiguidade de interpretação do pesos.

1.3 Contribuições

Para atingir seus objetivos, este trabalho apresenta uma série de contribuições listadas abaixo:

- Para a tarefa supervisionada:
  - Uma nova codificação de partículas, para treinamento de protótipos usando métodos populacionais de otimização, que também permite que o algoritmo realize seleção automática de variáveis.
    - A nova codificação é aplicada a dados intervalares.
  - Uma nova distância generalizada com três níveis de pesos. A vantagem é que essa distância é capaz de modelar a variabilidade de classes desbalanceadas e compostas de sub-regiões de tamanhos e formas diferentes. A nova distância também resolve problemas encontrados com as versões de distâncias adaptativas propostas anteriormente na literatura [63], como a perda de informação no cálculo dos pesos e ambiguidade na interpretação dos valores dos mesmos.
    - Como consequência da forma como os pesos da nova distância são calculados, o novo método de classificação é também capaz de remover protótipos inativos.
- Para a tarefa semi-supervisionada de estimação de probabilidades:
  - Um novo algoritmo que altera o critério de propagação usado nos métodos existentes [9, 43, 74] para incorporar uma função de escore apropriada. O novo algoritmo é flexível quanto aos métodos usados para selecionar os protótipos e construir o grafo e é aplicado a dados numéricos.
  - Uma nova distância adaptativa, cujos pesos são calculados sem informação de rótulos, durante a fase de inicialização dos protótipos.
- Outras contribuições:

  - Uma análise sobre as probabilidades obtidas a partir de um método de propagação de rótulos.
  - Uma estudo que mostra ganho de performance de um método de propagação de rótulos ao usar uma distância adaptativa
  - Uma breve análise sobre o benefício de métodos baseados em enxames para evitar mínimos locais

Adicionalmente, este trabalho também irá realizar estudo comparativo de desempenho entre os métodos introduzidos e outros presentes na literatura. Os métodos propostos têm objetivos e aplicações diferentes, portanto eles serão analisados separadamente. O primeiro método supervisionado terá seu desempenho medido em termos de taxa de erro de classificação e comparado ao de outros algoritmos usando conjuntos de dados sintéticos e reais. O segundo método terá seu desempenho avaliado usando funções de escore apropriadas, para verificar a qualidade das suas estimativas de probabilidades de classe.

1.4 Organização do documento

Esta tese de doutorado está organizada em duas partes como segue:

FUNDAMENTAÇÃO TEÓRICA

- **2 APRENDIZADO BASEADO EM PROTÓTIPOS**
  Esta Seção apresenta o estado-da-arte dos métodos de classificação supervisionada e semi-supervisionada usando protótipos, apontando também as suas limitações já discutidas na literatura relevante, como a necessidade de escolher uma métrica que modele bens os dados e o problema dos mínimos locais. Quanto à tarefa de classificação semi-supervisionada, a Seção também apresenta uma análise experimental que justifica a necessidade de um novo método de propagação de probabilidades, como o proposto na Seção 6.

- **3 DISTÂNCIAS ADAPTATIVAS**
  Esta Seção explora a literatura de métodos baseados em protótipos que usam distâncias adaptativas, com foco em uma distância adaptativa generalizada, cujas limitações, como perda de informação e ambiguidade de interpretação dos pesos (demonstrada graficamente), são resolvidas pela distâncias propostas nas Seções 5 e 6. A Seção também

apresenta experimentos que mostram que a construção de matrizes de adjacências usando distâncias adaptativas pode contribuir para um melhor desempenho dos métodos de propagação de rótulos.

- **4 TREINANDO PROTÓTIPOS COM OTIMIZAÇÃO DE ENXAMES**
  Esta Seção discute métodos de otimização de enxames para treinamento de protótipos e suas vantagens, focando na sua capacidade de fuga de mínimos locais, demonstrada por meio de um breve estudo de robustez.

MÉTODOS PROPOSTOS

- **5 PROTÓTIPOS TREINADOS POR ENXAMES APLICADOS A CLASSIFICAÇÃO DE DADOS INTERVALARES**
  Esta Seção apresenta a aplicação para dados intervalares de um algoritmo que usa uma distância adaptativa generalizada e diferente para cada protótipo, além de um método de otimização de enxames para tentar encontrar os melhores protótipos e as melhores variáveis para cada problema. A Seção também contem avaliações de convergência e de performance de acordo com valores de erro médio de classificação usando conjuntos de dados intervalares sintéticos e reais, além de avaliar a robustez do algoritmo contra má inicialização de protótipos e suas capacidades de eliminar protótipos inativos e selecionar atributos automaticamente.

- **6 PROPAGAÇÃO DE PROBABILIDADES USANDO PROTÓTIPOS E DISTÂNCIAS ADAPTATIVAS**
  Esta Seção propõe um algoritmo com o objetivo de gerar boas probabilidades de classe num cenário semi-supervisionado, usando propagação de probabilidades através de um grafo de protótipos, construído usando uma distância generalizada com dois níveis de adaptabilidade, capaz de modelar sub-regiões dos dados de diferentes formas e tamanhos. O algoritmo proposto é submetido a uma análise de desempenho que avalia a qualidade das suas probabilidades de classe estimadas usando funções de escore apropriadas.

7 CONCLUSÃO

Por último, esta Seção apresenta as considerações finais, os artigos publicados no decorrer do desenvolvimento deste trabalho (diretamente relacionados ou não) e possíveis trabalhos futuros.

2 APRENDIZADO BASEADO EM PROTÓTIPOS

Os nossos conhecimentos são a reunião do raciocínio e experiência de numerosas mentes.

—RALPH EMERSON

2.1 Introdução

A literatura de aprendizagem de máquina possui uma boa quantidade de trabalhos dedicados a métodos baseados em protótipos, com algoritmos aplicados a várias tarefas diferentes, como classificação (incluindo semi-supervisionada), agrupamento, regressão, etc. Como dito na Seção 1.1, este trabalho foca nas tarefas de classificação supervisionada e semi-supervisionada com estimação de probabilidades de classe. Assim, esta Seção explora o estado-da-arte dos métodos baseados em protótipos para classificação supervisionada (Seção 2.2) e semi-supervisionada (Seção 2.3), expondo métodos relacionados e discutindo suas limitações, que motivam o desenvolvimento dos métodos propostos nas Seções 5 e 6.

2.2 Aprendizado supervisionado

Os métodos descritos nesta Seção assumem que um conjunto de protótipos Ω_g de cardinalidade M_g é selecionado para a g-ésima classe e que o conjunto com os protótipos de todas as classes $\Omega = \{(\vec{w}_1, y_1), \ldots, (\vec{w}_m, y_m), \ldots, (\vec{w}_M, y_M)\}$ tem cardinalidade M, onde $\sum_{g=1}^{G} M_g = M$. Assim como as instâncias de treinamento, o m-ésimo protótipo é composto por um vetor $\vec{w}_m$ com p valores obtidos de $\mathscr{X} \in \mathscr{R}^p$ e um rótulo de classe y_m, que toma valores de $\mathscr{Y} = \{1, \ldots, g, \ldots, G\}$.

A Figura 1.2 mostra um conjunto de dados e quatro protótipos posicionados no centro de cada sub-região. Os objetos são rotulados de acordo com seu protótipo mais próximo. Assim, cada protótipo tem uma ”zona de influência”.

Métodos baseados em protótipos são capazes de realizar as mais diversas tarefas, como agrupamento [4, 10, 39, 46, 55, 23], estimação de densidade [53, 57], regressão [51] e classifi-

cação [3, 13, 35, 34, 38, 39, 57, 58, 61, 62, 63, 65].

Métodos de estimação de densidade usando kernels (KDE) [53] e misturas de Gaussianas (GMM) [57] não foram originalmente desenvolvidos para a tarefa de classificação, mas podem ser usados para realizar essa tarefa. O KDE estima densidades posicionando um kernel sobre cada instância de dados do conjunto de treinamento. Já o GMM posiciona kernels sobre protótipos previamente selecionados por um algoritmo de agrupamento, como o k-médias [46]. Para realizar as tarefas de classificação e/ou estimação de probabilidades de classe com esses métodos, é preciso usá-los para estimar as densidades de cada classe do conjunto de dados. Para classificar uma nova instância, basta calcular a sua densidade em cada classe e extrair probabilidades de classe usando o teorema de Bayes.

Outro algoritmo que envolve a noção de protótipos é a máquina de vetores de suporte (SVM) [13]. Esse método de classificação binária busca encontrar um plano que discrimine as duas classes com uma margem de separação máxima. A margem de separação é definida como a distância entre os elementos das duas classes mais próximos do plano de separação, chamados vetores de suporte. Como os vetores de suporte são instâncias especiais, selecionadas com o objetivo de discriminar melhor as classes, eles podem ser considerados protótipos.

Entre os classificadores puramente baseados em protótipos, os mais conhecidos pertencem à família *learning vector quantization* (LVQ) [38, 39]. Essa família incluía originalmente o algoritmo LVQ clássico, também conhecido como LVQ1, e suas variantes OLVQ1, LVQ2, LVQ3, além do LVQ-*Batch*.

Essas versões do LVQ tem formas diferentes de treinar os protótipos, mas atribuem rótulos a instâncias de teste da mesma forma. Dado um vetor de teste $\vec{x}$, o algoritmo encontra o protótipo mais próximo, representado pelo índice c, tal que $c = \arg\min_{m \in \{1,\ldots,M\}} d(\vec{x}, \vec{w}_m)$, e atribui o rótulo $y = y_c$. A função de dissimilaridade d mais comumente usada é a distância Euclidiana padrão.

Quanto ao algoritmo usado para treinar os protótipos, o LVQ1 segue dois passos simples. No instante t, dada a i-ésima instância de treinamento, o algoritmo encontra o índice c do protótipo mais próximo. A seguir, se $y_i = y_c$, o protótipo é atualizado de acordo com a Equação (2.1), caso contrário, adota-se a Equação (2.2). Nessas equações, $\alpha(t)$ é a taxa de aprendizado (que pode decrescer monotonicamente) no instante t. O resultado desse processo é a aproximação entre protótipos e instâncias das suas classes e o afastamento entre protótipos e instâncias de outras classes.

$$\vec{w}_c(t+1) = \vec{w}_c(t) + \alpha(t)\left(\vec{x}_i - \vec{w}_c(t)\right) \tag{2.1}$$

$$\vec{w}_c(t+1) = \vec{w}_c(t) - \alpha(t)\,(\vec{x}_i - \vec{w}_c(t)) \tag{2.2}$$

O OLVQ1 difere do LVQ1 por adotar taxas de aprendizado diferentes para cada protótipo. Já o LVQ2 atualiza dois protótipos a cada iteração: o mais próximo que pertence à mesma classe do objeto e o mais próximo que pertence a alguma outra classe. O LVQ3 é similar ao LVQ2, mas atualiza os dois protótipos mais próximos, sejam eles da mesma classe ou não.

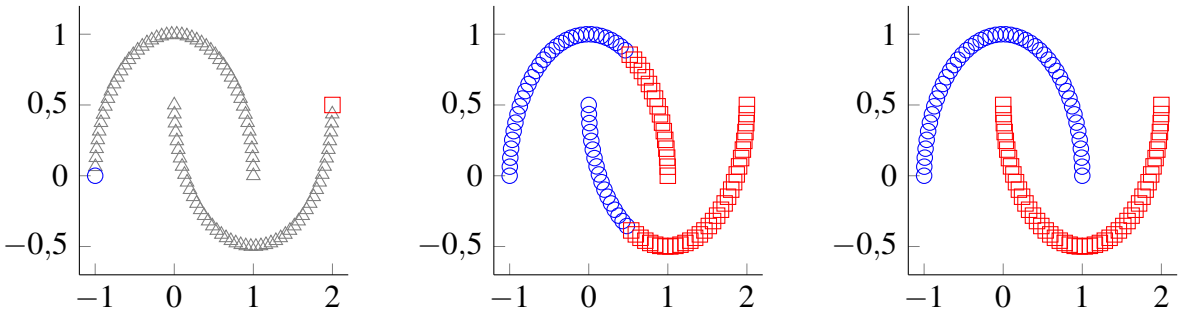
O LVQ-*Batch* é diferente dos outros por não ser baseado em aprendizado incremental. No LVQ-*Batch*, as instâncias são todas apresentadas aos protótipos e atribuídas aos seus protótipos mais próximos. A seguir, as classes dos protótipos são determinadas como as classes mais frequentes entre as suas instâncias. Por último, os protótipos são atualizados para a média das suas instâncias que pertencem à sua classe.

Essas primeiras versões do LVQ foram criadas por Kohonen como simples heurísticas para aproximar soluções Bayes-ótimas de forma intuitiva [35]. Portanto, Sato e Yamada [58] introduziram o *generalized learning vector quantization* (GLVQ), que fornece uma solução para a tarefa do LVQ baseada em otimização de uma função objetivo usando o método gradiente descendente.

Algumas adições recentes à família do LVQ têm o objetivo de melhorar certos aspectos do classificador, como interpretabilidade, tamanho do modelo e garantias de classificação [3, 34, 60]. Além disso, a métrica usada para calcular a distância entre os protótipos e as instâncias é muito importante para o desempenho dos métodos baseados em protótipos. Por isso, alguns trabalhos já foram dedicados a encontrar métricas adaptativas que modelem bem os dados para a tarefa de classificação supervisionada [61, 62, 63]. Outra questão crítica dos métodos baseados em protótipos é a inicialização. Esses métodos podem cair em mínimos locais, caso o conjunto inicial de protótipos seja mal posicionado. De Falco et al. [15] investigou uma solução para esse problema no cenário supervisionado, usando um método de otimização baseado em enxames. Essas duas questões receberão mais atenção mais adiante neste trabalho.

2.3 Aprendizado semi-supervisionado: propagação de rótulos

Em muitas aplicações de classificação, o praticante ou pesquisador de aprendizagem de máquina se depara com conjuntos de dados com poucas instâncias rotuladas. Para lidar com esse tipo de situação, a aprendizagem semi-supervisionada foi desenvolvida, com métodos capazes de aproveitar a informação contida nos dados não-rotulados para melhorar o desempenho de classificação [82].



(a) Conjunto de treinamento (b) kNN (c) Propagação de rótulos

Figura 2.1: **Conjunto duas luas** - (a) os dados são distribuídos em dois grupos com apenas duas instâncias rotuladas; (b) resultado se kNN for usado para rotular as demais instâncias; (c) resultado se um método de propagação de rótulos em grafos for usado

O interesse em métodos de aprendizagem semi-supervisionada tem sido grande, o que resultou em vários tipos de algoritmos diferentes [5, 11, 33, 83] e várias aplicações [28, 29, 45, 69, 70, 71, 77, 81].

Nesse trabalho, os algoritmos de maior interesse são os baseados em grafos [83]. Esses métodos são construídos a partir de duas suposições: (i) uma boa função de classificação não deve mudar (ou deve mudar o mínimo possível) os rótulos originais e (ii) indivíduos próximos devem ter rótulos similares [80]. Sendo assim, os métodos baseados em grafos fazem uma propagação de rótulos entre os objetos através do grafo [83].

A Figura 2.1 apresenta um conjunto de dados parcialmente rotulados como exemplo. O conjunto é formado por duas ”luas” cruzadas e apenas os dois objetos das extremidades são rotulados. Uma abordagem bastante simples e intuitiva para rotular as demais instâncias seria calcular as distâncias entre os dois objetos rotulados e todos os outros elementos e atribuir rótulos de acordo com o objeto rotulado mais próximo. O resultado dessa abordagem pode ser visto na Figura 2.1b. Se um método de propagação de rótulos em grafos for usado, a estrutura implícita dos dados deve ser levada em consideração na montagem do grafo. O resultado, visto na Figura 2.1c, captura a distribuição das classes em duas luas.

A maior parte dos trabalhos feitos sobre esse tipo de algoritmos focou em melhorar a acurácia da classificação através da construção de modelos de propagação ou de grafos melhores. Zhu et al. [83] propuseram uma propagação baseada em campos aleatórios Gaussianos e Zhou et al. [80] introduziram outra função de classificação suave na estrutura revelada pelos dados rotulados e não-rotulados. Quanto à construção do grafo, Wang e Zhang [72] desenvolveram uma abordagem que assume que cada ponto pode ser reconstruído a partir de uma combinação linear dos seus vizinhos e Tian e Kuang [68] propuseram a construção de um grafo de posto

baixo não-negativo para capturar as vizinhanças lineares globais.

Todos esses métodos demonstram boa acurácia em problemas de classificação, mas não são escaláveis, devido à estratégia de construção do grafo via kNN e ao cálculo da inversa da matriz Laplaciana na otimização. Recentemente, alguns trabalhos tentaram resolver essa limitação. Wang et al. [73] introduziram uma abordagem dividir-para-conquistar para construir uma aproximação do grafo de vizinhança. Para reduzir o custo computacional de propagação, Huang et al. [30] propuseram uma abordagem que começa propagando os rótulos das instâncias rotuladas para as não-rotuladas e depois continua propagando apenas entre as não-rotuladas até atingir um estado estável.

Dada a capacidade de representação apresentada pelos protótipos, alguns trabalhos procuraram usá-los na construção dos grafos, às vezes chamando-os de âncoras ou pontos de referência. Liu et al. [43] introduziram a regularização de grafo de âncoras (*anchor graph regularization* – AGR). Esse grafo é construído usando os protótipos e os objetos de treinamento. Desde sua introdução, o grafo de âncoras já foi usado em várias aplicações, como um método de agrupamento espectral mais eficiente [9], um método de hashing com geração de códigos compactos em tempo praticável [44], um método rápido de recuperação de imagens [75], etc.

Seja $\Im = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_i, y_i), \ldots, (\vec{x}_{n_l}, y_{n_l}), \vec{x}_{n_l+1}, \ldots, \vec{x}_N\}$ um conjunto de treinamento com N instâncias totais, sendo n_l rotuladas e $n_u = N - n_l$ não-rotuladas. O primeiro passo do método AGR é encontrar M protótipos não-rotulados $\Omega = \{\vec{w}_1, \ldots, \vec{w}_m, \ldots, \vec{w}_M\}$, usando algum método de agrupamento. O passo seguinte é encontrar uma matriz de pesos locais $\mathbf{Z}$ tal que $\forall_i \sum_{m=1}^{M} Z_{im} = 1$ e $Z_{im} \geq 0$. A matriz $\mathbf{Z}$ contém os pesos usados para a reconstrução dos elementos de treinamento a partir de combinações lineares dos protótipos e é encontrada pela solução da Equação (2.3) usando um solucionador de problemas quadráticos.

$$\arg\min_{\vec{Z}_i} \left|\left| \vec{x}_i - \Omega_{\langle i \rangle} \vec{Z}_{\langle i \rangle} \right|\right|^2, \tag{2.3}$$

$$\text{tal que } \sum \vec{Z}_{\langle i \rangle} = 1 \text{ e } \vec{Z}_{\langle i \rangle} \succcurlyeq 0,$$

onde $\langle i \rangle$ denota o conjunto dos índices dos k protótipos mais próximos à i-ésima instância.

A partir da matriz de pesos locais $\mathbf{Z}$, é possível construir uma matriz de adjacências $\mathbf{E}$, dada pela Equação (2.4), assumindo que objetos que são próximos aos mesmos protótipos devem ser similares.

$$\mathbf{E} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^T, \tag{2.4}$$

onde a matriz diagonal Λ é definida como $\Lambda_{mm} = \sum_{i=1}^{N} Z_{im}$. Essa matriz de adjacências é usada para fazer a propagação dos rótulos dos objetos rotulados para os protótipos e, por consequência, para os objetos não-rotulados.

Seja $\mathbf{Y}_{n_l} = [\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_{n_l}] \in \mathscr{R}^{n_l \times G}$ uma matriz de rótulos para as instâncias rotuladas, onde $Y_{ig} = 1$ se $y_i = g$, caso contrário $Y_{ig} = 0$. Seja também $\mathbf{A} = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_M] \in \mathscr{R}^{M \times G}$ uma matriz de predições de rótulos para os protótipos. O objetivo final da AGR é encontrar a matriz $\mathbf{A}$ que minimize o erro quadrático de reconstrução dos rótulos das instâncias rotuladas, ao mesmo tempo em que respeita, em todo o conjunto de treinamento, a condição de que objetos próximos devem ter rótulos similares. Para isso, é preciso encontrar $\mathbf{A}$ que minimiza a Equação (2.5).

$$Q(\mathbf{A}) = \sum_{i=1}^{n_l} \left|\left| \vec{Z}_i A - \vec{y}_i \right|\right|^2 + \frac{\xi}{2} \sum_{i',i''=1}^{N} E_{i'i''} \left|\left| \vec{Z}_{i'} A - \vec{Z}_{i''} A \right|\right|^2, \tag{2.5}$$

onde ξ é um parâmetro que determina a importância do termo de suavização dos rótulos através de todo o conjunto de treinamento. A Equação (2.5) tem uma solução analítica dada pela Equação (2.6).

$$\mathbf{A} = (\mathbf{Z}_{n_l}^T \mathbf{Z}_{n_l} + \xi \tilde{\mathbf{L}})^{-1} \mathbf{Z}_{n_l}^T \mathbf{Y}_{n_l}, \tag{2.6}$$

onde $\mathbf{Z}_{n_l}$ é a submatriz de $\mathbf{Z}$ referente aos objetos rotulados e $\tilde{\mathbf{L}} = \mathbf{Z}^T(\mathbf{I} - \mathbf{E})\mathbf{Z}$ é uma matriz Laplaciana reduzida. O último passo da AGR é a propagação dos rótulos dos protótipos para os objetos não-rotulados usando a Equação (2.7).

$$\hat{y}_i = \arg \max_{g \in \{1,\ldots,G\}} \frac{\vec{Z}_{i.} \times \vec{A}_{.g}}{\sum \mathbf{Z} \vec{A}_{.g}}, \; i = n_l + 1, \ldots, N, \tag{2.7}$$

onde $\vec{Z}_{i.}$ denota a i-ésima linha da matriz $\mathbf{Z}$, $\vec{A}_{.g}$ é a g-ésima coluna da matriz $\mathbf{A}$ e o denominador $\sum \mathbf{Z} \vec{A}_{.g}$ é um fator de normalização para lidar com classes desbalanceadas [83].

O cálculo dos pesos locais e da matriz de adjacências tem uma grande importância no desempenho da AGR. Além disso, $\mathbf{Z}$ é encontrada por um processo iterativo, o que pode ser computacionalmente custoso em grandes bases de dados. Wang et al. [74] propuseram modificações tanto no cálculo dos pesos locais, quanto na matriz Laplaciana normalizada que resultaram em um método AGR mais eficiente chamado *efficient anchor graph regularization* (EAGR).

No EAGR, as linhas da matriz de pesos locais $\mathbf{Z}$ são encontradas analiticamente e sua matriz de adjacências dos protótipos $\mathbf{E}$ (no AGR, $\mathbf{E}$ é uma matriz de adjacências de instâncias) é calculada pela Equação (2.8).

$$\mathbf{E} = \mathbf{Z}^T\mathbf{Z}, \tag{2.8}$$

Dadas a matriz de pesos locais $\mathbf{Z}$ e a matriz de adjacências $\mathbf{E}$, construída a partir de $\mathbf{Z}$ usando a Equação (2.8), e sejam $\mathbf{Y}_{n_l} = [\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_{n_l}] \in \mathscr{R}^{n_l \times G}$ e $\mathbf{A} = [\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_M] \in \mathscr{R}^{M \times G}$ matrizes de predições de rótulos para as instâncias rotuladas e os protótipos, respectivamente, o método EAGR realiza propagação de rótulos através da minimização da função descrita na Equação (2.9).

$$\arg\min_{\mathbf{A}} \sum_{i=1}^{n_l} ||\vec{z}_i\mathbf{A} - \vec{y}_i||^2 + \frac{\xi}{2} \sum_{m,r=1}^{M} E_{mr} \left|\left| \frac{\vec{a}_m}{\sqrt{D_{mm}}} - \frac{\vec{a}_r}{\sqrt{D_{rr}}} \right|\right|^2, \tag{2.9}$$

onde $\mathbf{D}$ é uma matriz diagonal cujos elementos são definidos como $D_{mm} = \sum_{r=1}^{M} E_{mr}$. O primeiro termo da função é responsável por encontrar os valores de $\mathbf{A}$ que minimizam o erro de classificação das instâncias rotuladas. O segundo termo (multiplicado pelo parâmetro ξ) minimiza a diferença entre os rótulos de protótipos relacionados. O nível de similaridade entre o m-ésimo e o r-ésimo protótipos é dado pelo peso E_{mr}, que leva em consideração os pesos locais dos protótipos em relação a todos os objetos do conjunto de dados.

2.3.1 Obtendo probabilidades do EAGR

Uma vez que a matriz $\mathbf{A}$ é encontrada e dada a matriz de pesos locais $\mathbf{Z}$, é possível rotular novas instâncias de teste. Para isso, é preciso encontrar os pesos locais das instâncias de teste, concatená-los à matriz $\mathbf{Z}$ e usar a Equação (2.7) para obter os rótulos. Apesar de os valores indicativos das G classes, usados para rotular as instâncias na Equação (2.7), não serem probabilidades, é possível aplicar a função *softmax*, dada pela Equação (2.10), aos mesmos para obter probabilidades de classe.

$$p(y_i = g) = \frac{e^{\frac{\vec{Z}_{i.} \times \vec{A}_{.g}}{\Sigma \mathbf{Z}\vec{A}_{.g}}}}{\sum\limits_{h \in \{1,\ldots,G\}} e^{\frac{\vec{Z}_{i.} \times \vec{A}_{.h}}{\Sigma \mathbf{Z}\vec{A}_{.h}}}} \tag{2.10}$$

Apesar de ser possível obter probabilidades de classe do EAGR, a otimização das Equações (2.5) e (2.9) não tem o objetivo de produzir boas probabilidades de clase, mas sim de minimizar o erro de classificação após a propagação dos rótulos. A matriz $\mathbf{A}$ pode tomar quaisquer valores reais para realizar essa minimização. A propagação de rótulos é feita através da otimização do erro quadrático de reconstrução de valores indicativos de rótulos, com a única preocupação de

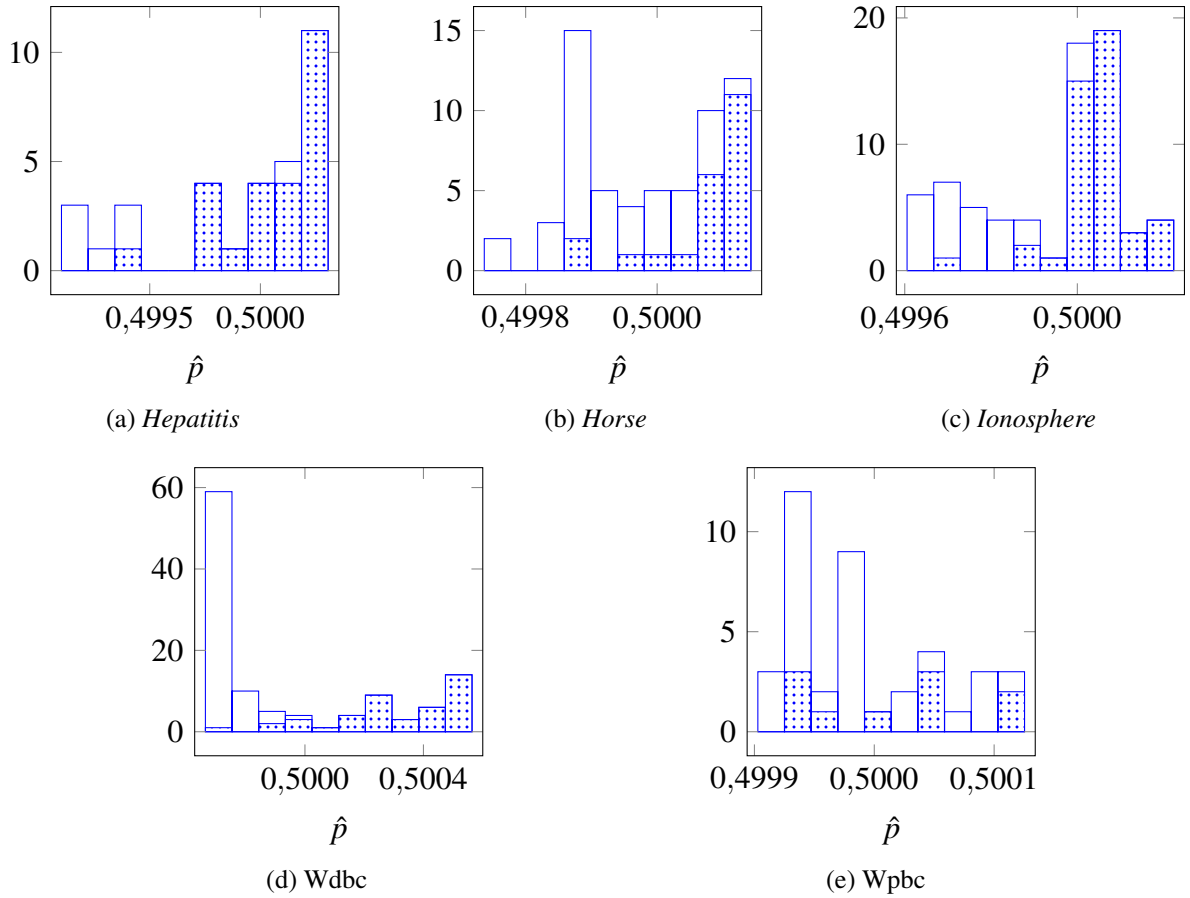
produzir rótulos corretos.

2.3.2 O EAGR produz modelos bem calibrados?

Segundo Kull et al. [40], um modelo probabilístico pode ser chamado de *bem calibrado* se, entre as instâncias que recebem uma probabilidade estimada $\hat{p}$, a proporção de instâncias da classe positiva se aproxima de $\hat{p}$. A calibragem de um modelo é importante para tomada de decisões ótimas e classificação sensível a custos de erro, pois é possível escolher um limiar ótimo para as predições de um classificador suficientemente bem calibrado para minimizar os custos de erro. Além disso, o limiar pode ser obtido de forma ótima para se adaptar a mudanças na proporção das classes e/ou nos custos.

Assim, é interessante avaliar o quão calibradas são as probabilidades produzidas pelo EAGR usando a função *softmax*. Para isso, o EAGR foi treinado com cinco conhecidos conjuntos de dados reais extraídos do repositório UCI [41] e descritos na Tabela 6.1, na página 97: *hepatitis*, *horse*, *ionosphere*, wdbc e wpbc. Todos os conjuntos representam problemas binários, i.e., possuem apenas duas classes (positiva e negativa). Por uma questão de simplicidade, os mesmos valores de parâmetros foram usados para todos os conjuntos: 15 protótipos por classe, $k = 3$, parâmetros para construção do grafo $\lambda = 10$ e $\sigma = 10$ e parâmetro regularizador $\xi = 100$. Os conjuntos foram separados em 5 partições, mantendo as proporções das classes, sendo 4 partições usadas para treinar e 1 para testar o modelo. Esses conjuntos de dados são todos completamente rotulados, então 80% dos rótulos do conjunto de treinamento foram apagados para simular cenários semi-supervisionados.

A Figura 2.2 apresenta as distribuições das probabilidades da classe positiva produzidas pelo EAGR para as instâncias de teste. Os histogramas sem preenchimento representam as distribuições das probabilidades para todo o conjunto de teste e os histogramas pontilhados representam as distribuições das probabilidades apenas para as instâncias da classe positiva. Para todos os conjuntos, as probabilidades ficaram bastante concentradas ao redor de $\hat{p} = 0,5$. Portanto, se os modelos obtidos fossem calibrados, as barras dos histogramas das instâncias positivas deveriam ter aproximadamente metade da altura das barras correspondentes nos histogramas de todas as instâncias, ou seja, a proporção de instâncias positivas deveria se aproximar de $0,5$, o que não ocorreu. Portanto, os modelos probabilísticos produzidos pelo EAGR, usando a função *softmax*, para os cinco conjuntos analisados podem ser considerados mal calibrados. Esse resultado motiva o desenvolvimento de um método de propagação de probabilidades em grafo bem calibrado.



(a) *Hepatitis* (b) *Horse* (c) *Ionosphere*

(d) Wdbc (e) Wpbc

Figura 2.2: **Histogramas das probabilidades de classe produzidas pelo EAGR** - $\hat{p}$ representa as probabilidades para a classe positiva, obtidas após aplicar a Equação (2.10). Os histogramas não preenchidos representam as distribuições das probabilidades para todas as instâncias e os histogramas pontilhados representam as distribuições das probabilidades para as instâncias da classe positiva.

2.4 Considerações finais

Este Capítulo apresentou o estado da arte de métodos de aprendizagem baseados em protótipos, focando nas tarefas de classificação supervisionada e semi-supervisionada. Algoritmos dos mais variados tipos usam o conceito de protótipos de alguma forma, como os não-supervisionados k-médias, GMM e KDE, os supervisionados SVM e LVQ e os semi-supervisionados AGR e EAGR. Apesar das suas vantagens, os métodos baseados em protótipos tem algumas limitações que podem afetar seu desempenho em qualquer tarefa à qual eles forem aplicados, como a influência da métrica escolhida e a tendência de atingir mínimos locais. Além disso, mais especificamente na tarefa de classificação semi-supervisionada, os métodos

discutidos não tem o objetivo de produzir bons modelos probabilísticos, o que os torna inadequados em alguns cenários, como por exemplo classificação sensível a custos.

Tendo em vista o que foi exposto neste Capítulo, os Capítulos seguintes discutirão distâncias adaptativas, que podem melhorar a modelagem dos dados de treinamento, e métodos de otimização baseados em enxames, para evitar o problema dos mínimos locais, com o objetivo de apresentar soluções para as tarefas de classificação supervisionada e semi-supervisionada.

3 DISTÂNCIAS ADAPTATIVAS

Nós podemos ver apenas uma pequena distância a nossa frente, mas o suficiente para perceber que há muito a fazer.

—ALAN TURING

3.1 Introdução

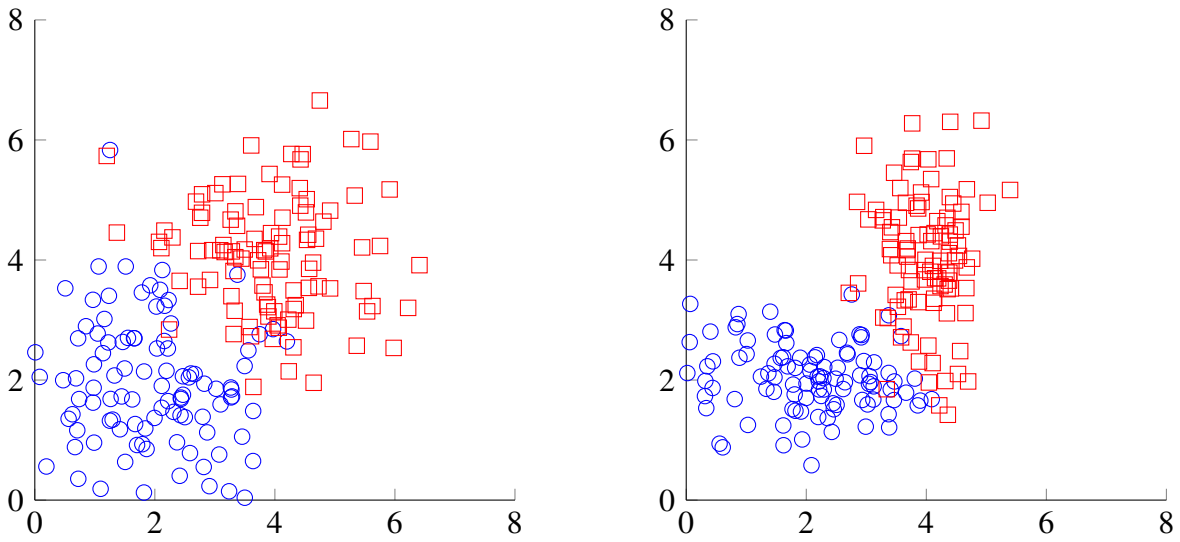
A escolha da distância usada pelo método baseado em protótipos tem uma grande influência no seu desempenho. A distância Euclidiana padrão é a mais utilizada por métodos da família LVQ e sua principal característica é que objetos equidistantes a um ponto formam uma esfera ao redor do mesmo, ou seja, ela tem a característica de modelar regiões de dados esféricas.

Como mostra a Seção 3.2, a suposição de que os dados se distribuem em regiões esféricas pode não ser ideal em aplicações práticas e muitos trabalhos vêm buscando soluções para essa questão. Uma dessas soluções é discutida na Seção 3.3, que apresenta uma distância Euclidiana generalizada com três níveis de adaptatividade, capaz de modelar regiões de dados compostas por sub-regiões de variadas formas e tamanhos. Por fim, a Seção 3.4 mostra que o uso de uma distância generalizada para construir a matriz de adjacências pode melhorar o desempenho de métodos de propagação de rótulos.

3.2 Quando a distância Euclidiana falha

A Figura 3.1 mostra exemplos de distribuições de dados. No lado esquerdo (Figura 3.1a), as duas classes tem formato esférico, ou seja, as variáveis tem variâncias iguais e as covariâncias são 0. No lado direito (Figura 3.1b), as classes tem formato elíptico, as variáveis tem variâncias diferentes e as covariâncias são 0. No segundo caso, é interessante que o método baseado em protótipos use uma distância capaz de modelar classes não-esféricas.

Pensando nisso, vários estudos surgiram com o objetivo de encontrar distâncias adaptativas, capazes de modelar diferentes distribuições dos dados. Esse interesse não é recente:



(a) Classes esféricas (b) Classes elípticas

Figura 3.1: **Dados distribuídos em duas classes** - (a) as duas classes tem formatos esféricos, caso que pode ser bem modelado pela distância Euclidiana padrão e (b) as duas classes são elípticas, apresentando diferentes variabilidades de suas variáveis, caso que não pode ser bem modelado pela distância Euclidiana padrão

Diday e Govaert (1977) [18] propuseram um método de agrupamento que usa uma distância que se adapta à estrutura local dos dados. Após esse trabalho, outros algoritmos de agrupamento adaptativo foram propostos. De Carvalho et al. (2006) [14] introduziram algoritmos de agrupamento difuso que usam duas distâncias diferentes: uma modela a dispersão das variáveis no conjunto de treinamento inteiro e a outra modela a dispersão em cada grupo.

Além dos métodos de agrupamento adaptativos, também foram propostos métodos supervisionados, pertencentes à família LVQ, que usam distâncias adaptativas. Bojer et al. [6] propuseram um método para determinar a relevância das variáveis, introduzindo pesos, calculados segundo a teoria Hebbiana, que resultam em duas vantagens: além de melhorar o desempenho de classificação, os pesos permitem selecionar as variáveis mais relevantes. Schneider et al. [59] expandiram o trabalho de Bojer et al. [6], introduzindo uma matriz de relevância que leva em consideração as correlações das variáveis. No entanto, esse método tinha uma tendência a sofrer sobreajuste, portanto Schneider et al. [60] propuseram a introdução de um fator de regularização para o cálculo da matriz de relevância.

3.3 Distância Euclidiana generalizada

Souza e Silva Filho [65] introduziram uma distância que modela a dispersão global das variáveis para dados do tipo ponto. Essa distância foi estendida em duas novas distâncias por Silva Filho e Souza [61], a primeira com pesos capazes de representar a dispersão das variáveis em cada classe e a segunda com pesos para as sub-regiões das classes, representadas pelos protótipos. Essa última distância foi adaptada para dados intervalares em [62]. Em outro trabalho, Silva Filho e Souza [63] introduziram uma distância intervalar, chamada dp_m, com pesos que representam três níveis de adaptabilidade: a importância de cada classe no conjunto de treinamento, a importância de cada protótipo em sua classe e as dispersões das variáveis na sub-região representada por cada protótipo.

Em [63], a distância dp_m, representada pela Equação (3.1), foi usada em um algoritmo LVQ difuso, chamado *weighted interval fuzzy learning vector quantization* (WFLVQ), que mostrou bom desempenho, em comparação com o LVQ difuso que usa a distância Euclidiana padrão.

$$\mathrm{dp}_m(\vec{x}_i, \vec{w}_m) = \mathrm{d}_{m,L}(\vec{x}_i, \vec{w}_m) + \mathrm{d}_{m,U}(\vec{x}_i, \vec{w}_m), \tag{3.1}$$

onde $\mathrm{d}_{m,L}(\vec{x}_i, \vec{w}_m)$ e $\mathrm{d}_{m,U}(\vec{x}_i, \vec{w}_m)$ são as distâncias Euclidianas quadráticas adaptativas entre os limites inferiores e superiores, respectivamente, das variáveis intervalares da i-ésima instância e do m-ésimo protótipo. Essas distâncias Euclidianas quadráticas adaptativas são calculadas de acordo com a Equação (3.2).

$$\mathrm{d}_{m,L}(\vec{x}_i, \vec{w}_m) = \sum_{j=1}^{p} \lambda_{m,j}(x_{i,j,L} - w_{m,j,L})^2. \tag{3.2}$$

O vetor de pesos $\vec{\lambda}_m$ modela a dispersão das variáveis na sub-região de dados representada pelo m-ésimo protótipo e é responsável pela capacidade de modelar sub-regiões de diferentes formatos. O j-ésimo peso do m-ésimo protótipo é calculado pela Equação (3.3).

$$\lambda_{m,j} = \frac{\{\gamma_{g,m} \prod_{h=1}^{p} \Delta_{m,h}\}^{\frac{1}{p}}}{\Delta_{m,j}}, \tag{3.3}$$

onde $\gamma_{g,m}$ é o peso do m-ésimo protótipo na sua classe $y_m = g$ e $\Delta_{m,j}$ é a soma das diferenças quadráticas entre os limites inferiores e superiores do m-ésimo protótipo e dos padrões corretamente afetados por ele, dada pela Equação (3.4).

$$\Delta_{m,j} = \sum_{i \in m} u_{im} \left[(x_{i,j,L} - w_{m,j,L})^2 + (x_{i,j,U} - w_{m,j,U})^2 \right] \delta(m,i), \tag{3.4}$$

onde u_{im} é o grau de pertinência difuso do i-ésimo padrão para o m-ésimo protótipo, i indica as instâncias afetadas pelo m-ésimo protótipo e $\delta(m,i) = 1$ se $y_i = y_m$, senão $\delta(m,i) = 0$.

O peso $\gamma_{g,m}$ na Equação (3.3) modela a importância do m-ésimo protótipo em sua classe $y_m = g$. Isso permite que a distância modele classes com sub-regiões de diferentes tamanhos em suas estruturas. O peso é atualizado usando a Equação (3.5).

$$\gamma_{g,m} = \frac{\{\beta_g \prod_{r=1}^{M_g} (\sum_{i \in r} u_{ir} \ \mathrm{dp}_r(\vec{x}_i, \vec{w}_r)\delta(r,i))\}^{\frac{1}{M_g}}}{\sum_{i \in m} u_{im} \ \mathrm{dp}_m(\vec{x}_i, \vec{w}_m)\delta(m,i)}, \tag{3.5}$$

onde M_g é o número de protótipos que pertence à g-ésima classe. A Equação (3.5) resulta em erro se $\sum_{i \in m} u_{im} \ \mathrm{dp}_m(\vec{x}_i, \vec{w}_m)\delta(m,i) = 0$, ou seja, se o protótipo não afetar instância de treinamento alguma. Portanto, todos os protótipos que cumpram essa condição são removidos do conjunto de protótipos.

O terceiro tipo de peso, β_g, representa a importância da g-ésima classe no banco de dados. Esse peso permite que o algoritmo modele classes desbalanceadas e é computado de acordo com os protótipos que pertencem a cada classe e os padrões que foram afetados por eles, segundo a Equação (3.6).

$$\beta_g = \frac{\left[\prod_{a=1}^{G} (\sum_{m \in a} \sum_{i \in m} u_{im} dp_m(\vec{x}_i, \vec{w}_m)\delta_(m,i))\right]^{\frac{1}{G}}}{\sum_{m \in g} \sum_{i \in m} u_{im} dp_m(\vec{x}_i, \vec{w}_m)\delta_(m,i)}. \tag{3.6}$$

Caso $\sum_{m \in g} \sum_{i \in m} u_{im} dp_m(\vec{x}_i, \vec{w}_m)\delta_(m,i) = 0$, o que significa que nenhum protótipo da g-ésima classe afetou instâncias, os pesos de todas as classes recebem o valor 1.

O WFLVQ sofre de duas limitações principais devido a sua distância. A primeira limitação acontece porque WFLVQ é um algoritmo online, assim os vetores de pesos $\vec{\lambda}_m$ devem ser recalculados para cada padrão de treinamento, levando à perda de informação nas somas das distâncias. A segunda limitação é causada pelas suas equações dos pesos: como mostra a Figura 3.2, as somas nessas equações podem ter valores altos tanto porque um protótipo afetou muitas instâncias próximas quanto porque as instâncias estão localizadas longe dos seus protótipos. Essa ambiguidade é problemática, porque valores mais altos dessas somas levam a valores mais baixos para os pesos e pesos menores resultam em distâncias menores.

Se é desejável que protótipos tenham pesos menores quando estão longe das suas instâncias, o mesmo não é verdade para protótipos que já estão bem posicionados e afetam muitas instâncias de treinamento. Esse problema leva a um pequeno número de protótipos muito ativos e um grande número de protótipos inutilizados.

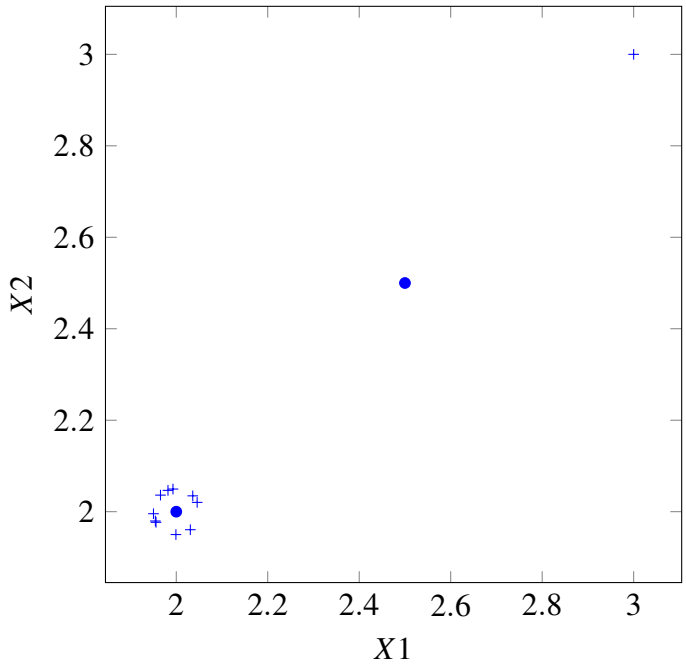


Figura 3.2: **Protótipos (círculos) e suas instâncias (cruzes)** - a soma das distâncias da nuvem de instâncias à esquerda para o protótipo no seu centro é igual à distância da instância à direita para o seu protótipo mais próximo

3.4 Matriz de adjacências para propagação de rótulos usando uma distância generalizada

Como discutido na Seção 2.3, o cálculo dos pesos locais e da matriz de adjacências é muito importante para o desempenho de métodos de propagação de rótulos baseados em protótipos, como o AGR e o EAGR. Portanto, é interessante avaliar o impacto do uso de uma distância adaptativa ou generalizada na construção dessa matriz. Nesta Seção, duas versões do EAGR serão comparadas. Essas versões são idênticas em todos os passos, exceto na distância usada para montar a matriz de adjacências.

A distância generalizada usada nesta análise, dada pela Equação (3.7), é uma distância para dados do tipo ponto, baseada na distância do WFLVQ, discutida na Seção 3.3, porém ela é calculada sem informação de rótulos e de graus de pertinência, tendo, portanto, dois níveis de adaptabilidade: a importância de cada protótipo no conjunto de treinamento e as dispersões das variáveis na sub-região representada por cada protótipo.

$$\mathrm{d}_m(\vec{x}_i, \vec{w}_m) = \sum_{j=1}^{p} \lambda_{m,j}(x_{i,j} - w_{m,j})^2 \tag{3.7}$$

Usando o método dos multiplicadores de Lagrange, com $\lambda_{m,j} > 0$ e $\prod_{j=1}^{p} \lambda_{m,j} = \gamma_m$, o j-ésimo valor do vetor de pesos $\vec{\lambda}_m$ do m-protótipo é calculado pela Equação (3.8).

$$\lambda_{m,j} = \frac{\{\gamma_m \prod_{h=1}^{p} (x_{i,h} - w_{m,h})^2\}^{\frac{1}{p}}}{(x_{i,j} - w_{m,j})^2}, \tag{3.8}$$

onde γ_m é o peso do m-ésimo protótipo no conjunto de treinamento e é atualizado usando a Equação (3.9), obtida pelo método dos multiplicadores de Lagrange, com $\gamma_m > 0$ e $\prod_{m=1}^{M} \gamma_m = 1$

$$\gamma_m = \frac{\left[\prod_{r=1}^{M} (\sum_{i \in r} \mathrm{d}_m(\vec{x}_i, \vec{w}_r))\right]^{\frac{1}{M}}}{\sum_{i \in m} \mathrm{d}_m(\vec{x}_i, \vec{w}_m)}, \tag{3.9}$$

onde M é o número de protótipos. A Equação (3.9) resulta em erro se $\sum_{i \in m} \mathrm{d}_m(\vec{x}_i, \vec{w}_m) = 0$, ou seja, se o protótipo não afetar instância de treinamento alguma. Para evitar esse problema, todos os protótipos que cumpram essa condição são removidos do conjunto de protótipos.

O algoritmo K-means clássico foi escolhido para selecionar os protótipos. Sua versão adaptativa segue os passos do Algoritmo 1. Dados os protótipos e seus pesos, os demais passos do método EAGR seguem conforme proposto por Wang et al. [74].

Algoritmo 1 K-means adaptativo

Inicialize k centroides aleatoriamente;
Enquanto O critério de parada não é atingido **Faça**:
    Determine o centroide mais próximo de cada instância;
    Atualize os centroides como as médias das instâncias afetadas;
    Atualize os pesos da distância, usando as Equações (3.8) e (3.9).
Fim Enquanto
Retorne centroides e seus pesos.

Para avaliar o efeito da distância generalizada d_m as versões adaptativa e padrão do EAGR foram treinadas com 10 conjuntos retirados do repositório online UCI [41]: *balance-scale*, *cleveland*, *hepatitis*, *horse*, *ionosphere*, *iris*, *vehicle*, *vowel*, wdbc e wpbc. Assim como na Seção 2.3.1, os mesmos valores de parâmetros foram selecionados para todos os conjuntos: 15 protótipos por classe (centroides do K-means), $k = 3$, parâmetros para construção do grafo $\lambda = 10$ e $\sigma = 10$ e parâmetro regularizador $\xi = 100$. Os experimentos foram executados em 10 repetições de validações cruzadas com 5 partições. Como os conjuntos de dados são todos completamente rotulados, 80% dos rótulos do conjunto de treinamento foram apagados para simular cenários semi-supervisionados.

A Tabela 3.1 apresenta os resultados em termos de taxa de acerto (em %), com valores

	Distância generalizada	Distância padrão
balance-scale	67,665	67,884
	(5,501)	(4,969)
cleveland	50,977	50,065
	(4,822)	(4,975)
hepatitis	76,984	76,961
	(6,777)	(8,206)
horse	63,684	**69,491**
	(8,823)	(**6,496**)
ionosphere	81,481	80,742
	(4,372)	(5,789)
iris	**91,133**	86,000
	(**5,817**)	(5,471)
vehicle	**45,301**	40,430
	(**4,226**)	(3,556)
vowel	**35,374**	33,465
	(**3,393**)	(3,622)
wdbc	93,883	93,482
	(2,256)	(2,312)
wpbc	55,231	55,385
	(7,510)	(7,784)

Tabela 3.1: **Resultados de taxa de acerto percentual do EAGR** - Valores em **negrito** denotam significância estatística, segundo o teste dos postos sinalizados de Wilcoxon, com $\alpha = 0,05$.

em **negrito** denotando significância estatística, segundo o teste dos postos sinalizados de Wilcoxon, com $\alpha = 0,05$. A distância generalizada apresentou melhores médias em 7 dos 10 conjuntos, com significância em 3. Já a distância padrão foi significativamente melhor em apenas 1 conjunto. Esses resultados mostram que o uso de distâncias adaptativas para construir a matriz de adjacências pode contribuir para melhorar o desempenho dos métodos de propagação de rótulos.

3.5 Considerações finais

Esta Seção apresentou o conceito de distâncias adaptativas como uma alternativa à distância Euclidiana padrão em casos em que a suposição de que os dados se distribuem em regiões esféricas pode não se aplicar. Essa é uma área que recebe bastante atenção na literatura, pois o desempenho de métodos baseados em protótipos depende do quanto a distância escolhida é

adequada aos dados do problema.

A Seção 3.3 apresentou uma distância generalizada com três níveis de adaptabilidade, para modelar sub-regiões de variadas formas e tamanhos. Apesar da sua capacidade de modelar diversas situações, essa distância tem algumas limitações, como o cálculo dos pesos em etapas diferentes, ocasionando perda de informação, e a ambiguidade de interpretação do valor dos seus pesos. Essas limitações são resolvidas pelas distâncias usadas pelos algoritmos propostos nas Seções 5 e 6.

Já a Seção 3.4 apresentou uma análise de desempenho de um método de propagação de rótulos, usando uma distância generalizada para construir a matriz de adjacências. Os resultados mostraram que o uso da distância generalizada pode contribuir para um melhor desempenho, justificando o uso de uma distância generalizada pelo método proposto na Seção 6.

4 TREINANDO PROTÓTIPOS COM OTIMIZAÇÃO DE ENXAMES

Nunca subestime o poder de pessoas estúpidas em grandes grupos.
—AUTOR DESCONHECIDO

4.1 Introdução

O desempenho dos métodos baseados em protótipos tende a depender bastante da etapa de inicialização. Se protótipos mal-posicionados forem selecionados, o algoritmo pode não se recuperar e terminar caindo em um mínimo local. Portanto, métodos que inicializam e treinam uma população de soluções, como os baseados em enxames, tornam-se uma possível solução para esse problema.

Esta Seção aborda o treinamento de protótipos usando enxames. A Seção 4.2 explica o conhecido método de enxames *Particle Swarm Optimization* (PSO), a Seção 4.3 detalha a codificação das partículas e a Seção 4.4 analisa a robustez do método PSO, mostrando que os métodos baseados em enxames podem mitigar o problema dos mínimos locais.

4.2 PSO: um método de enxames simples e eficaz

O PSO é baseado no comportamento social de bandos de pássaros e cardumes de peixes [37]. O algoritmo busca pela melhor solução possível fazendo várias partículas ”voarem” pelo espaço de busca em direção a melhores posições. Para isso, o PSO usa as Equações (4.1) e (4.2).

$$\mathbf{V}_l(t+1) = \omega \mathbf{V}_l(t) + c_1 \mathbf{R}_1 \cdot (pbest_l(t) - \mathbf{W}_l(t)) + c_2 \mathbf{R}_2 \cdot (gbest(t) - \mathbf{W}_l(t)), \tag{4.1}$$

$$\mathbf{W}_l(t+1) = \mathbf{W}_l(t) + \mathbf{V}_l(t), \tag{4.2}$$

onde $\mathbf{W}_l(t)$ e $\mathbf{V}_l(t)$ são, respectivamente, a posição e a velocidade da l-ésima partícula no instante t (aqui representadas como matrizes), ω é o valor de inércia, c_1 e c_2 são coeficientes de aceleração, $pbest_l(t)$ é a melhor posição obtida pela l-ésima partícula até o instante t, $gbest(t)$ é a melhor posição obtida por todo o enxame até o instante t e $\mathbf{R}_1$ e $\mathbf{R}_2$ são matrizes de valores aleatórios sorteados do intervalo $[0;1]$.

Após cada iteração, as qualidades das SN partículas são calculadas e as melhores posições locais ($pbest_l$) e global ($gbest$) são atualizadas. O PSO atualiza as partículas usando as Equações (4.1) e (4.2) até que um critério de parada seja atingido.

Algoritmo 2 PSO

1: Inicialize SN partículas e suas velocidades;
2: **Enquanto** O critério de parada não é atingido **Faça**:
3: Calcule a qualidade de cada partícula;
4: Ajuste o $pbest_l$ para cada partícula ($l = 1, \ldots, SN$) e o $gbest$ do enxame;
5: **Para** $l = 1 \rightarrow SN$ **Faça**:
6: Atualize a velocidade $\mathbf{V}_l$, usando a Equação (4.1);
7: Calcule a nova posição $\mathbf{W}_l$, usando a Equação (4.2);
8: **Fim Para**
9: **Fim Enquanto**

Retorne $gbest$.

Muitas novas versões do PSO original foram introduzidas para obter melhores resultados ou convergência mais rápida. Por uma questão de simplicidade, esta Seção se aterá ao PSO original, mas a Seção 5 aborda uma versão mais recente do PSO.

4.3 Codificação das soluções

Classificadores baseados em protótipos são treinados através da otimização de um critério, como a soma das distâncias entre protótipos e seus objetos afetados. Isso permite o uso de algoritmos de otimização, como os métodos de enxames, para treinar os protótipos. Trabalhos anteriores exploraram classificadores baseados em protótipos treinados por enxames, com o objetivo de escapar de mínimos locais, minimizando o problema da má inicialização [15, 12, 66, 20].

Para treinar protótipos usando métodos de otimização de enxames, as soluções podem ser codificadas como segue. Para um conjunto com G classes e p variáveis, o objetivo é encontrar as posições ótimas de M protótipos no espaço p-dimensional. Assim, em uma população com SN partículas/soluções, o l-ésimo indivíduo é uma matriz $M \times p$, $\mathbf{W}_l = \{\vec{w}_{1,l}, \ldots, \vec{w}_{M,l}\}$, cujas linhas representam protótipos cujas coordenadas são $\vec{w}_{m,l} = \{W_{m,l,1}, \ldots, W_{m,l,p}\}$. Assim, cada partícula é composta por $M \cdot p$ valores reais. Em métodos que usam a "velocidade" das partículas, como o PSO, essa velocidade tem a mesma composição das partículas, ou seja, a velocidade da l-ésima partícula é uma matriz $M \times p$, $\mathbf{V}_l = \{\vec{v}_{1,l}, \ldots, \vec{v}_{M,l}\}$, cuja m-ésima linha representa a velocidade do m-ésimo protótipo $\vec{v}_{m,l} = \{v_{m,l,1}, \ldots, v_{m,l,p}\}$.

Uma parte importante dos algoritmos de enxames é o critério usado para avaliar a qualidade das partículas, pois ele tem um grande impacto no resultado final. Cervantes et al. [12] propuseram dois algoritmos que adotam critérios diferentes. O primeiro usa a taxa de sucesso de classificação, enquanto o segundo foca nas distâncias dos protótipos para os elementos corretamente e incorretamente classificados. De forma similar ao segundo algoritmo de Cervantes et al. [12], Dilmac e Korurek [20] usaram a distância entre os elementos e seus centroides de classe mais próximos.

Quando um método usa apenas a taxa de erro/sucesso de classificação (ψ_1) no conjunto de treinamento como critério, as partículas podem terminar resultando em excelentes classificadores para o conjunto de treinamento, mas não para objetos de teste. Por outro lado, se apenas a soma das distâncias (ψ_2) for considerada, os protótipos podem ficar muito bem posicionados para representar as classes, mas não para resolver conflitos nas fronteiras das classes. Portanto, a função de qualidade ψ_3, proposta por De Falco et al. [15], é interessante por tentar evitar a ocorrência de sobreajuste do classificador ao conjunto de treinamento, fazendo uma combinação linear entre ψ_1 e ψ_2, como mostra a Equação (4.3).

$$\psi_3(l) = \frac{1}{2}\left(\psi_1(l) + \psi_2(l)\right), \tag{4.3}$$

onde $\psi_1(l) = N_{erros,l}/N$ é a taxa de erro de classificação, $N_{erros,l}$ é o número de erros de classificação cometidos pela l-ésima partícula e N é o número de objetos do conjunto de treinamento. Os valores de $\psi_1(l)$ situam-se no intervalo $[0;1]$. O segundo componente, ψ_2, é dado pela Equação (4.4)).

$$\psi_2(l) = \frac{\sum_{i=1}^{N} d\left(\vec{x}_i, \vec{w}_{m,l}\right) \delta(m,l,i)}{\sum_{i=1}^{N} \delta(m,l,i)}, \tag{4.4}$$

onde $\vec{w}_{m,l}$ é o protótipo da l-ésima partícula mais próximo do i-ésimo objeto e $\delta(m,l,i) = 1$ se $y_i = y_{m,l}$, caso contrário $\delta(m,l,i) = 0$. ψ_2 é a soma das distâncias Euclidianas quadráticas

clássicas entre objetos corretamente classificados e seus protótipos mais próximos. No trabalho de De Falco et al. [15], os protótipos são os centroides das classes, mas neste trabalho, cada classe pode ter vários protótipos.

Os valores de ψ_2 devem ser mantidos no intervalo $[0;1]$ (assim como os da função ψ_1). Portanto, as somas das distâncias devem ser normalizadas com respeito a $\sum_{i=1}^{N} \delta_(m,l,i)$, que corresponde ao número de classificações corretas. Além disso, os p componentes da distância são normalizados com respeito aos seus valores máximos e mínimos e a soma dos componentes é dividida por p.

A codificação e as funções de qualidade citadas acima não dependem do método baseado em enxames escolhido, o que significa que eles podem ser usados com algoritmos de enxames bem conhecidos, como o PSO, ou mais recentes, como o VABC [31].

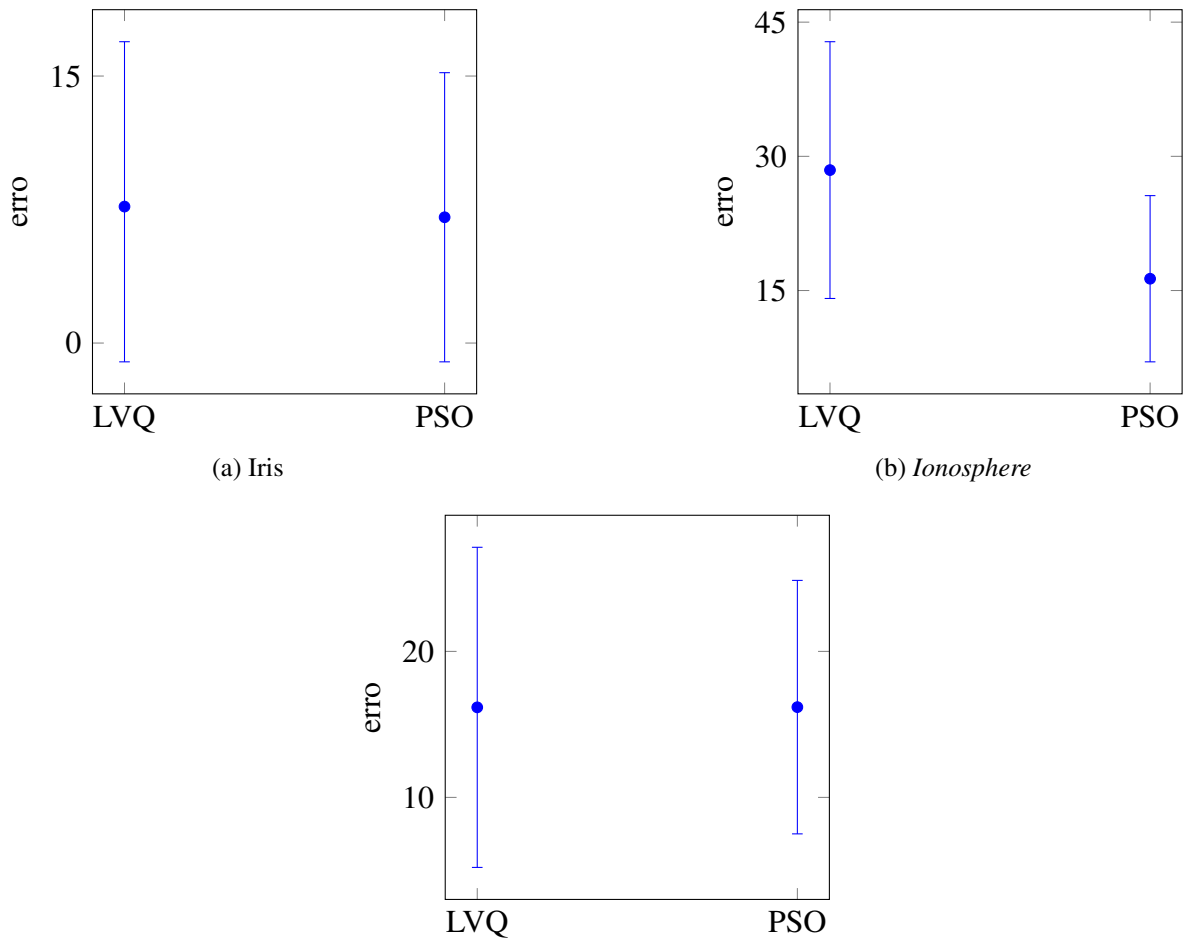
4.4 Análise de robustez

Esta Seção faz uma rápida demonstração experimental da capacidade de evitar mínimos locais demonstrada pelo PSO. Para isso, a versão original desse método será comparada ao algoritmo LVQ1 usando três conhecidos conjuntos de dados do repositório online UCI [41] e descritos na Tabela 6.1, na página 97: *iris*, *ionosphere* e *hepatitis*. O PSO utilizará a codificação discutida na Seção 4.3, minimizando o critério ψ_3, apresentado na Equação (4.3).

Os dois métodos foram treinados em 10 repetições de validações cruzadas com 5 partições (total de 50 execuções), por um máximo de 200 iterações (parando antes em caso de estagnação por 50 iterações). Por uma questão de simplicidade os mesmos valores de parâmetros foram usados para todos os conjuntos, sendo eles: 2 protótipos para cada classe, 25 partículas, inércia $\omega = 0,6$ e coeficientes de aceleração $c_1 = c_2 = 2$ para o PSO e taxa de aprendizado fixa de $0,3$ para o LVQ. Os algoritmos e seus respectivos parâmetros não foram escolhidos para otimizar os desempenhos das abordagens, mas apenas para obter modelos simples que demonstram o benefício do uso de métodos populacionais para reduzir a variância dos resultados.

A Figura 4.1 mostra intervalos de 95% de confiança para as médias dos erros dos dois métodos (em %), assumindo normalidade dos dados. Apesar de os intervalos mostrarem que as diferenças entre as médias dos erros não são significativas, os intervalos do PSO foram menores do que os intervalos do LVQ em todos os três conjuntos, indicando menor variância dos resultados, principalmente no conjunto *ionosphere*.

Essa menor variância ocorre porque o PSO tem mais facilidade de evitar mínimos locais do que o LVQ. Outros métodos de enxames compartilham essa habilidade, sendo boas alternativas



(a) Iris

(b) *Ionosphere*

(c) *Hepatitis*

Figura 4.1: **Intervalos de confiança com** $\alpha = 0,05$ - Os intervalos do PSO foram todos menores do que os do LVQ, indicando menor variância

para treinar protótipos.

4.5 Considerações finais

Esta Seção discutiu o uso de métodos de enxames para treinar protótipos. Essa tarefa envolve a codificação das partículas como matrizes de protótipos e a minimização de um critério composto de dois termos: o erro de classificação e a soma das distâncias dos protótipos para as instâncias corretamente afetadas por eles.

O interesse em usar métodos de enxames para treinar protótipos se deve a sua capacidade de evitar mínimos locais, o que foi demonstrado experimentalmente na Seção 4.4. Essa

capacidade é explorada pelo método proposto na Seção 5.

5 PROTÓTIPOS TREINADOS POR ENXAMES APLICADOS A CLASSIFICAÇÃO DE DADOS INTERVALARES

Todo o conhecimento que não leva a novas perguntas rapidamente morre: não consegue manter a temperatura necessária para a manutenção da vida.

—WISLAWA SZYMBORSKA

5.1 Introdução

Considerando os métodos existentes para classificação de dados intervalares usando protótipos [62, 63], três limitações podem ser identificadas. A primeira é a escolha de uma distância adaptativa apropriada. Como discutido na Seção 3, a distância intervalar adotada pelo método WFLVQ [63] demonstrou bom desempenho em relação à distância Euclidiana intervalar não-adaptativa. No entanto, essa distância ainda tem algumas limitações, como: perda de informação, devido ao fato de que os pesos são atualizados para cada exemplo no treinamento, e incapacidade de distinguir protótipos que afetam muitos objetos e protótipos que situam-se longe dos elementos afetados (dois cenários extremamente diferentes).

A segunda questão diz respeito à convergência para mínimos locais devido a inicializações sub-ótimas dos protótipos, ou seja, a qualidade das soluções encontradas por esses métodos depende da seleção de protótipos bem posicionados. Algoritmos adaptativos [62, 63] tem se mostrado capazes de obter melhor posicionamento dos protótipos, mas não resolvem a questão. Uma possível solução para esse problema envolve usar métodos de otimização de enxames aplicados à seleção de protótipos, o que já foi investigado para dados clássicos do tipo ponto em alguns trabalhos [15, 12, 66, 20].

Por último, a terceira questão identificada na literatura é a seleção de variáveis relevantes. Seleção de variáveis é um mecanismo comumente utilizado para obter melhor desempenho de

classificação. Assim como com a seleção de protótipos, a seleção de variáveis pode ser vista como um problema de otimização ao qual métodos populacionais de otimização podem ser aplicados. Essa solução já foi explorada diversas vezes para dados do tipo ponto [52, 42, 26, 25], mas nenhum desses trabalhos focou em classificação baseada em protótipos.

Com o objetivo de prover soluções para as três limitações citadas, esta Seção introduz um novo algoritmo aplicado à classificação de dados intervalares, chamado *interval swarm-optimized prototype classifier* (ISOPC). O método ISOPC usa uma nova distância adaptativa que resolve as limitações da distância usada pelo método WFLVQ [63]. Além disso, o ISOPC é capaz de selecionar protótipos e variáveis, usando um método de otimização de enxames para treinar os protótipos. Qualquer método de enxames (ou de otimização de populações de uma maneira geral) que se adeqüe à codificação definida na Seção 5.4 pode ser usado. Neste trabalho o método escolhido foi o *velocity-based artificial bee colony* (VABC) [31], por ser um algoritmo de simples implementação que combina as vantagens de dois tipos diferentes de algoritmos baseados em enxames: *particle swarm optimization* (PSO) [37, 21] e *artificial bee colony* (ABC) [36] para evitar mínimos locais, obtendo bons resultados. Deste ponto em diante, a versão do ISOPC que usa o algoritmo VABC para treinar os protótipos será chamada de *interval velocity-based artificial-bee classifier* (IVABC).

O Algoritmo 3 apresenta uma representação de alto nível do IVABC, com partes sublinhadas que correspondem às principais contribuições discutidas nesta Seção. O IVABC procede da seguinte maneira: novas partículas são selecionadas aleatoriamente e suas velocidades e valores de qualidade são inicializados (passos 1 a 3), a seguir, o método itera pelas fases de *abelhas empregadas* (passos 5 a 8), *observadoras* (passos 9 a 13) e *exploradoras* (passos 14 a 16), até um critério de convergência ser atingido.

Essas três fases do IVABC são assim chamadas por remeter ao comportamento de diferentes tipos de abelhas em uma colmeia. As abelhas empregadas continuam trabalhando em sua fonte de alimento, o que é representado no algoritmo por melhorias pontuais em cada uma das fontes de alimento memorizada. As abelhas observadoras recebem as informações de qualidade das fontes de alimento trabalhadas pelas outras abelhas e decidem qual fonte em qual fonte de alimento irão trabalhar, o que é representado no algoritmo por melhorias pontuais em fontes de alimento selecionadas aleatoriamente, com maior probabilidade para fontes com maior qualidade. Por fim, as abelhas exploradoras saem da colmeia em busca de novas fontes, o que o algoritmo representa como o abandono de fontes de alimento estagnadas e reinicialização de novas fontes.

Esta Seção aborda seis aspectos importantes do IVABC: os dados simbólicos intervalares (Seção 5.3); o algoritmo de enxames adotado (Seção 5.2); a codificação das partículas (Se-

Algoritmo 3 IVABC - pseudocódigo simplificado

1: Inicialize um total de *SN* fontes de comida;
2: Inicialize as velocidades;
3: Calcule a função de qualidade de cada partícula;
4: **Enquanto** o critério de parada não for atingido **Faça**:
Envie as abelhas empregadas:
5: Atualize as fontes de alimento;
6: Calcule os pesos da distância;
7: Calcule as novas funções de qualidade;
8: Memorize as melhores soluções entre as fontes de alimento originais e modificadas;
Envie as abelhas observadoras:
9: Calcule as probabilidades das fontes de alimento;
10: Atualize as fontes de alimento usando suas velocidades;
11: Calcule os pesos da distância;
12: Calcule as novas funções de qualidade;
13: Memorize as melhores soluções entre as fontes de alimento originais e modificadas;
Envie as abelhas exploradoras:
14: Reinicialize fontes de alimento estagnadas e suas velocidades;
15: Calcule os pesos da distância;
16: Calcule as novas funções de qualidade;
17: **Fim Enquanto**
Retorne Melhor partícula e seus pesos.

ção 5.4), responsável por representar os protótipos e pela habilidade de selecionar variáveis; a função de qualidade (Seção 5.5), desenvolvida para evitar sobreajuste; sua distância adaptativa e o procedimento para calcular seus pesos (Seção 5.6); e seu processo de atribuição de rótulos (Seção 5.7), baseado numa abordagem de k-protótipos mais próximos.

Essas contribuições estão presentes em passos importantes do Algoritmo 3. A nova codificação das partículas é parte integral dos passos de inicialização (passos 1 e 14) e aparece também na fase de abelhas empregadas (passo 5). Nos passos 6, 11 e 15, o algoritmo atualiza os pesos da distância relativos aos protótipos de uma partícula e calcula as distâncias entre instâncias de treinamento e esses protótipos. A seguir, o algoritmo atribui rótulos a essas instâncias, resultando numa taxa de erros de classificação. Essa taxa de erro e as distâncias são usadas para avaliar a qualidade da partícula.

5.2 *Velocity-based artificial bee colony*

Algoritmos de otimização de enxames são conhecidos por sua habilidade de escapar de mínimos locais, devido a sua versatilidade [32]. Existem muitos tipos diferentes de métodos

de enxames, mas o IVABC usa uma combinação do PSO [37, 21] e do ABC [36], chamado *velocity-based artificial bee colony* (VABC) [31].

Tanto o PSO quanto o ABC têm mostrado bons desempenhos de otimização e foram adotados em várias aplicações, apesar de terem algumas limitações. No caso do PSO, se a melhor partícula atingir um mínimo local, o algoritmo pode ser incapaz de evitar mover as outras partículas nessa direção, o que resulta em uma solução mal-otimizada. O ABC, ao abandonar fontes de alimento estagnadas, pode ser capaz de escapar desse cenário. Por outro lado, o ABC faz apenas pequenos ajustes a uma fonte de alimento antes de abandoná-la, o que pode levar a uma má exploração [31]. O método VABC foi proposto para superar essas limitações, ao combinar os pontos fortes de ABC e PSO.

Assim como o algoritmo ABC, o VABC consiste de fontes de alimento que codificam soluções e três tipos de abelhas: empregadas, observadoras e exploradoras. Todas as fontes de alimento são inicializada por abelhas exploradoras. Seja SN o número de fontes de alimento e seja a matriz $\vec{w}_l$, $(l = 1 \ldots SN)$ uma solução para o problema de otimização. A l-ésima partícula contém p variáveis $(w_{l,j}, j = 1 \ldots p)$ que devem ser ajustadas para otimizar (minimizar ou maximizar) uma função objetivo $f(\vec{w}_l)$. Depois que os vetores de soluções são inicializados, as abelhas empregadas vão se alimentar próximo às fontes existentes, o que é representado pela Equação (5.1).

$$w'_{l,h} = w_{l,h} + \phi * (w_{l,h} - w_{k,h}), \tag{5.1}$$

onde $\vec{w}_l$ é a fonte de alimento memorizada, $\vec{w}_k$ é uma fonte de alimento selecionada aleatoriamente, que deve ser diferente de $\vec{w}_l$, h é uma variável selecionada aleatoriamente e ϕ é um valor aleatório selecionado do intervalo $[-1; 1]$. A seguir, o critério de qualidade da nova fonte de alimento $\vec{w}'_l$ é calculado. A abelha irá memorizar ou $\vec{w}_l$ ou $\vec{w}'_l$, baseado numa seleção gulosa.

Após as abelhas empregadas terminarem sua busca, elas passam a informação das qualidades das fontes de alimento para as abelhas observadoras. Baseado nessa informação, cada fonte de alimento recebe uma probabilidade p_l de ser escolhida por uma abelha observadora. Essa probabilidade é calculada pela Equação (5.2), usando o critério de avaliação $f(\vec{w}_l)$.

$$p_l = \frac{f(\vec{w}_l)}{\sum_{k=1}^{SN} f(\vec{w}_k)} \tag{5.2}$$

No VABC, cada abelha observadora escolhe uma fonte de alimento $\vec{w}_k$ usando essas probabilidades e tenta achar uma melhor fonte $\vec{w}'_k$ usando as equações de atualização das partículas do PSO (Equações 5.3 e 5.4). A seguir, o critério da nova fonte de alimento $\vec{w}'_k$ é calculado e a abelha escolhe memorizar a fonte que tiver o melhor critério entre $\vec{w}_k$ e $\vec{w}'_k$.

$$\vec{v}_k(t+1) = \omega\vec{v}_k(t) + c_1\vec{r}_1 \cdot (pbest_k(t) - \vec{w}_k(t)) + c_2\vec{r}_2 \cdot (gbest(t) - \vec{w}_k(t)), \tag{5.3}$$

$$\vec{w}_k(t+1) = \vec{w}_k(t) + \vec{v}_k(t), \tag{5.4}$$

onde $\vec{w}_k(t)$ e $\vec{v}_k(t)$ são, respectivamente, a posição e a velocidade da k-ésima partícula ou fonte de alimento no instante t, ω é a inércia, c_1 e c_2 são coeficientes de aceleração, $pbest_k(t)$ é a melhor posição obtida pela k-ésima partícula até o instante t, $gbest(t)$ é a melhor posição obtida pelo enxame até o instante t e $\vec{r}_1$ e $\vec{r}_2$ são vetores, cujos valores são escolhidos aleatoriamente no intervalo $[0;1]$.

Na fase das abelhas exploradoras, cada fonte de alimento que falhou em melhorar sua qualidade mais do que um certo número de vezes, que é um parâmetro do algoritmo, é abandonada. Nesse caso, uma abelha exploradora busca uma nova fonte de alimento. O VABC itera por essas três fases (abelhas empregadas, observadoras e exploradoras) até atingir um critério de convergência. Como mencionado nesta Seção, o VABC será usado para treinar protótipos descritos por variáveis intervalares, como as descritas na Seção 5.3.

5.3 Dados simbólicos

Dados clássicos são definidos como vetores de variáveis que podem ser quantitativas ou qualitativas, não sendo, portanto, naturalmente capazes de representar variabilidade ou incerteza de dados complexos. Assim, a análise de dados simbólicos (ADS) introduziu diversos tipos de dados para representar melhor a variabilidade dos dados, como histogramas, listas de valores, intervalos e outros [19].

Como um exemplo simples, pode-se dizer que o comprimento de uma espécie de peixe é de 30 a 60 cm, mas não passa de 40 cm nas fêmeas. Esse tipo de informação não é facilmente representado por tabelas de pontos de dados clássicos. Esses peixes podem ser descritos corretamente da seguinte forma:

[Comprimento = [30, 60]], [Sexo = {macho, fêmea}]

e [Se {Sexo = fêmea} Então {Comprimento <= 40}].

A ADS tem o objetivo de pesquisar e introduzir novas ferramentas capazes de modelar esses tipos de dados. Um ponto de dados simbólicos pode representar um hipercubo ou um produto Cartesiano de distribuições no espaço p-dimensional ou uma mistura de ambos. Por outro lado, um ponto clássico é apenas um ponto no espaço p-dimensional [16].

Dados simbólicos surgem naturalmente de diversas formas, como a variação diária de temperatura de uma região ou como a lista de categorias de uma população e suas frequências. Esses dados também podem ser obtidos através da agregação de uma base de dados, usando categorias de níveis mais altos, e.g. uma base de moradores de um país pode ser agregada em uma base de cidades do país. Essa agregação pode ser feita tanto porque o pesquisador desejava analisar os dados a partir de uma visão de nível mais alto, quanto porque a base de dados era grande demais e precisava ser reduzida.

5.3.1 Tipos de dados

Seja um conjunto de dados, contendo os registros dos moradores do Brasil obtidos em um censo demográfico, tal que, para cada indivíduo, tem-se o registro de sua região (N, NE, CO, SE ou S), cidade (Recife, Manaus, etc), naturalidade (brasileiro ou não), sexo (M ou F), profissão, renda anual e idade. Esse típico conjunto de dados pode ser representado pela Tabela 5.2, contendo variáveis aleatórias $X_1, X_2, \ldots$ descritas na Tabela 5.1.

Tabela 5.1: **Exemplos de variáveis clássicas** - variáveis categóricas e quantitativas

X_j	Descrição: valores possíveis
X_1	Região: Norte (N), Nordeste (NE), Centro-Oeste (CO), Sudeste (SE), Sul (S)
X_2	Cidade
X_3	Brasileiro: sim (S), não (N)
X_4	Sexo: masculino (M), feminino (F)
X_5	Profissão
X_6	Renda anual (em milhares de reais): ≥ 0
X_7	Idade (em anos): ≥ 0

Cada linha da Tabela 5.2 corresponde a um ponto de dados clássicos, correspondendo a valores observados para as variáveis $X_1, \ldots, X_p$ para um indivíduo. É possível analisar esses dados considerando certas categorias, e.g. habitantes da região Sul, ou médicos da região Sudeste. Cada uma dessas categorias consiste de vários indivíduos, portanto o valor observado não é mais um simples ponto. Por exemplo, a idade dos habitantes da região Sul (segundo a Tabela 5.2) pode ser representada pela lista $\{24,\ 26,\ 28,\ 34,\ 36,\ 37,\ 37,\ 41,\ 42\}$, pelo intervalo $[24;42]$ ou pelo histograma $\{[20;30),\ 1/3;\ [30;40),\ 4/9;\ [40;50],\ 2/9\}$. Cada um desses valores representa um tipo de dado simbólico. A partir da variável profissão, é possível obter 11 categorias ou conceitos simbólicos. Esses conceitos formam a Tabela 5.3 e a Tabela 5.4.

Tabela 5.2: **Exemplo de base de dados clássicos** - as variáveis foram descritas na Tabela 5.1

i	X_1	X_2	X_3	X_4	X_5	X_6	X_7
1	N	Belém	S	M	Engenheiro	60	43
2	S	Porto Alegre	N	F	Advogada	84	36
3	S	Curitiba	S	M	Engenheiro	36	24
4	CO	Cuiabá	S	M	Agricultor	120	50
5	NE	Recife	S	F	Médica	140	47
6	N	Manaus	N	F	Médica	96	60
7	NE	Recife	S	M	Porteiro	8,4	25
8	NE	Fortaleza	S	F	Médica	108	40
9	N	Palmas	S	M	Estudante	3,6	18
10	SE	São Paulo	N	M	Advogado	240	55
11	SE	Vitória	S	F	Médica	120	38
12	SE	São Paulo	S	M	Médico	144	66
13	S	Curitiba	N	F	Engenheira	96	34
14	NE	Fortaleza	S	F	Engenheira	48	27
15	N	Belém	S	M	Médico	72	32
16	N	Belém	S	M	Porteiro	12	67
17	S	Porto Alegre	S	F	Engenheira	36	42
18	SE	São Paulo	S	F	Estudante	0	16
19	NE	Recife	N	M	Médico	180	58
20	S	Porto Alegre	N	M	Engenheiro	84	26
21	S	Curitiba	S	F	Porteira	24	41
22	NE	Recife	S	F	Médica	120	31
23	N	Manaus	S	M	Porteiro	8,4	25
24	SE	Vitória	S	M	Estudante	6	19
25	NE	Fortaleza	S	F	Estudante	0	20
26	N	Belém	S	F	Engenheira	96	42
27	SE	São Paulo	N	F	Engenheira	120	30
28	S	Porto Alegre	S	M	Médico	180	37
29	NE	Recife	S	F	Porteira	9,6	29
30	N	Belém	N	M	Médico	240	48
31	SE	Vitória	S	F	Engenheira	36	34
32	NE	Recife	S	M	Engenheiro	48	26
33	N	Manaus	S	F	Médica	108	51
34	S	Curitiba	S	M	Porteiro	24	28
35	SE	São Paulo	N	F	Advogada	120	64
36	NE	Recife	S	M	Advogado	120	41
37	NE	Fortaleza	S	F	Estudante	0	23
38	N	Belém	S	F	Advogada	84	33
39	S	Curitiba	S	M	Advogado	108	37
40	SE	Vitória	N	M	Advogado	72	29

Tabela 5.3: **Dados simbólicos** - variáveis dos tipos multivalorada e categórica

w_u	Profissão	X_2
w_1	Médico	{Recife,Manaus,Fortaleza,Vitória,São Paulo,Belém,Porto Alegre}
w_2	Engenheiro	{Belém,Curitiba,Fortaleza,Porto Alegre,São Paulo,Vitória,Recife}
w_3	Advogado	{Porto Alegre,São Paulo,Recife,Belém,Curitiba,Vitória}
w_4	Porteiro	{Recife,Belém,Curitiba,Manaus}
w_5	Estudante	{Palmas,São Paulo,Vitória,Fortaleza}
w_6	Agricultor	{Cuiabá}

Tabela 5.4: **Dados simbólicos de vários tipos** - duas variáveis do tipo multivalorada, uma modal e duas variáveis intervalares; n_u indica o número de indivíduos descritos pelo u-ésimo conceito simbólico

w_u	X_1	X_3	X_4	X_6	X_7	n_u
w_1	{N,NE,SE,S}	{S, N}	{M(5/11),F(6/11)}	[72;240]	[31;66]	11
w_2	{N,NE,SE,S}	{S, N}	{M(4/10),F(6/10)}	[36;120]	[24;43]	10
w_3	{N,NE,SE,S}	{S, N}	{M(4/7),F(3/7)}	[72;240]	[29;64]	7
w_4	{N,NE,S}	{S}	{M(4/6),F(2/6)}	[8,4;24]	[25;67]	6
w_5	{N,NE,SE}	{S}	{M(2/5),F(3/5)}	[0;6]	[16;23]	5
w_6	{CO}	{S}	{M(1/1)}	[120;120]	[50;50]	1

Uma observação simbólica tipicamente representa o conjunto de n_u indivíduos que satisfazem a descrição associada ao conceito simbólico, ou categoria, w_u [16]. Um valor clássico é um caso particular de um valor simbólico correspondente. Variáveis clássicas podem ser qualitativas (e.g., $x_{ij} = Recife$) ou quantitativas (e.g., $x_{ij} = 24$). Variáveis simbólicas podem ser multivaloradas, intervalares ou modais com ou sem lógica, taxonomia e regras de dependência hierárquica.

5.3.1.1 Variáveis multivaloradas

Uma variável aleatória simbólica multivalorada X toma valores compostos de uma quantidade finita de valores categóricos ou quantitativos obtidos do domínio da variável.

Por exemplo, no conjunto de dados de censo demográfico presente na Tabela 5.3, os valores da variável $X_2 = Cidade$ para a categoria Médico ($u = 1$) são {Recife, Manaus, Fortaleza, Vitória, São Paulo, Belém, Porto Alegre}. Da mesma forma, os valores da variável $X_3 = Brasileiro$ para a mesma categoria, segundo a Tabela 5.4 são {S,N}, o que significa que alguns desses indivíduos são brasileiros e outros não.

Uma observação qualitativa clássica é um caso particular de uma variável simbólica multivalorada. Por exemplo, o valor da variável X_3 para a categoria Estudante ($u = 5$) é $X_3 = \{S\}$, ou seja, todos os estudantes são brasileiros.

5.3.1.2 Variáveis modais

Seja uma variável X que toma seus valores do domínio $\{\eta_k; k = 1,2,\ldots\}$. Ela é uma variável modal se tomar a forma $X(w_u) = \{\eta_k, \pi_k; k = 1,\ldots,s_u\}$ para a u-ésima observação, onde π_k é uma medida não-negativa associada a η_k e s_u representa o número de valores do domínio utilizados.

Por exemplo, no conjunto de dados de censo demográfico presente na Tabela 5.4, os valores da variável $X_4 = Sexo$ para a categoria Engenheiro ($u = 2$) são $\{M(4/10), F(6/10)\}$, ou seja, 40% dos engenheiros são homens e 60% dos engenheiros são mulheres.

Um caso particular das variáveis modais são as variáveis do tipo histograma, que tomam seus valores de um número finito de intervalos não sobrepostos $\{[a_k; b_k), k = 1,2,\ldots\}$ onde $a_k \leq b_k$, assumindo a forma $X(w_u) = \{[a_k; b_k), p_{uk}; k = 1,\ldots,s_u\}$, onde $s_u < \infty$ é o número finito de intervalos que formam o valor de $X(w_u)$ para a observação w_u e p_{uk} é o peso para o intervalo $[a_k; b_k)$, $k = 1,\ldots,s_u$, com $\sum_{k=1}^{s_u} p_{uk} = 1$.

A variável intervalar $X_6 = Renda\ anual$ para a categoria Engenheiro ($u = 2$) pode ser representada como $X_6 = \{[20;40),\ 0,3;\ [40;60),\ 0,3;\ [60;80),\ 0,0;\ [80;100),\ 0,3; [100;120],\ 0,1\}$, ou seja, 30% dos engenheiros ganham entre 20000 e 40000 reais, 30% ganham entre 40000 e 60000 reais, nenhum deles ganha entre 60000 e 80000 reais, 30% ganham entre 80000 e 100000 reais e 10% ganham entre 100000 e 120000 reais.

5.3.1.3 Variáveis intervalares

Uma variável aleatória simbólica intervalar X, toma valores em um intervalo, $X = [a; b] \subset \mathscr{R}^1$, onde $a \leq b$ e $a, b \in \mathscr{R}^1$. Quando os valores intervalares surgem da agregação de um conjunto de dados clássicos como na Tabela 5.2, os valores para a_{uj} e b_{uj} da j-ésima variável são $\min_{i\in\Omega_u} x_{ij}$ e $\max_{i\in\Omega_u} x_{ij}$, respectivamente, onde Ω_u é o conjunto de valores que constituem a categoria w_u.

No conjunto de dados de censo demográfico presente na Tabela 5.4, os valores da variável $X_6 = Renda\ anual$ para a categoria Porteiro ($u = 4$) são $X_6 = [8,4;24]$, caracterizando a variável X_6 como uma variável simbólica intervalar.

Uma variável quantitativa clássica é um caso particular de variável simbólica do tipo intervalo chamado de intervalo degenerado, sendo equivalente a um ponto em $\mathscr{R}^1$. No exemplo do conjunto de censo demográfico, as variáveis $X_6 = Renda\ anual$, e $Y_7 = Idade$ têm valo-

res $[120;120]$ e $[50;50]$, respectivamente, para a categoria Agricultor ($u = 6$). As variáveis intervalares são o foco do algoritmo proposto nesta Seção.

5.4 Uma nova codificação para partículas compostas por protótipos intervalares

Esta Seção estende a codificação discutida na Seção 4.3 para protótipos intervalares. Para um conjunto de dados com G classes e p variáveis intervalares, o IVABC tem o objetivo de achar as π variáveis mais úteis e posicionar M protótipos para achar a representação ótima do conjunto de treinamento $\Im$. Para isso, a l-ésima partícula é codificada como $\mathbf{W}_l = \{\vec{w}_{0,l}, \vec{w}_{1,l}, \ldots, \vec{w}_{M,l}\}$, com velocidade $\mathbf{V}_l = \{\vec{v}_{0,l}, \vec{v}_{1,l}, \ldots, \vec{v}_{M,l}\}$. Tanto a posição quanto a velocidade do m-ésimo protótipo são compostas por p coordenadas intervalares, e.g., $\vec{w}_{m,l} = \{w_{m,l,1}, \ldots, w_{m,l,p}\}$ e $\vec{v}_{m,l} = \{v_{m,l,1}, \ldots, v_{m,l,p}\}$, onde os valores $w_{m,l,j} = [w_{m,l,j,L}; w_{m,l,j,U}]$ e $v_{m,l,j} = [v_{m,l,j,L}; v_{m,l,j,U}]$ são intervalos.

O vetor $\vec{w}_{0,l} = \{w_{0,l,1}, \ldots, w_{0,l,p}\}$ é um importante componente da l-ésima partícula, que não representa um protótipo, mas um vetor de p valores reais situados no intervalo $[0,1]$, cujos componentes correspondem às p variáveis intervalares e indicam que variáveis são usadas pela partícula. Se $w_{0,l,j} \leq 0,5$, então a j-ésima variável correspondente é descartada por todos os protótipos da l-ésima partícula. Adicionalmente, $\vec{v}_{0,l}$ é o componente da velocidade que corresponde a $\vec{w}_{0,l}$. Assim, cada indivíduo da população é composto por $M \cdot p$ intervalos e p números reais.

O novo método inicializa as fontes de alimento como segue: para cada fonte de alimento $\vec{w}_l$, $(l = 1, \ldots, SN)$, para cada classe g, $(g = 1, \ldots, G)$, M_g protótipos são selecionados, resultando em M protótipos por partícula ($\sum_{g=1}^{G} M_g = M$). Além disso, para cada variável j, $(j = 1, \ldots, p)$, $w_{0,l,j}$ é aleatoriamente selecionado do intervalo $[0,1]$.

A fórmula da fase de abelhas empregadas tem que ser adaptada para refletir essa codificação. Para cada fonte de alimento $\mathbf{W}_l$, a abelha escolhe outra fonte de alimento aleatória $\mathbf{W}_k$. A seguir, um índice aleatório $0 \leq m \leq M$ é escolhido, onde o índice $m = 0$ indica o vetor que contem a importância de cada variável e os outros valores possíveis de m são associados aos protótipos. Por último, uma dimensão $(j = 1, \ldots, p)$ é escolhida aleatoriamente. A Equação (5.5) mostra como a atualização da fonte de alimento é feita, caso $m = 0$. Quando $m > 0$, a atualização usa a Equação (5.6).

$$w'_{0,l,j} = w_{0,l,j} + \phi * (w_{0,l,j} - w_{0,k,j}) \tag{5.5}$$

$$
\begin{aligned}
w'_{m,l,j,L} = &\quad w_{m,l,j,L} + \phi * (w_{m,l,j,L} - w_{m,k,j,L}) \\
w'_{m,l,j,U} = &\quad w_{m,l,j,U} + \phi * (w_{m,l,j,U} - w_{m,k,j,U})
\end{aligned} \tag{5.6}
$$

Antes que uma partícula seja atualizada, é fundamental checar se, para todos os componentes intervalares $w'_{m,l,j}$, $w'_{m,l,j,L} <= w'_{m,l,j,U}$, ou seja, se a definição de intervalo é respeitada (limite inferior menor ou igual ao limite superior). Caso essa condição não seja respeitada, os limites devem ser invertidos. Além disso, é preciso verificar se algum limite inferior $w'_{m,l,j,L} < min(j_L)$, ou se algum limite superior $w'_{m,l,j,U} > max(j_U)$, onde $min(j_L)$ e $max(j_U)$ são, respectivamente, os valores mínimos dos limites inferiores e os valores máximos dos limites superiores de todos os intervalos correspondentes à j-ésima variável. Adicionalmente, todos os $w'_{0,l,j}$ devem ser mantidos no intervalo $[0;1]$. Por último, as velocidades devem ser mantidas no intervalo $[v_{min}; v_{max}]$, onde v_{min} e $v_{max}]$ são parâmetros do método. Somente após essas verificações, o valor do critério de avaliação da partícula será calculado e o algoritmo decidirá se memorizará a nova partícula $\mathbf{W'}_l$ ou manterá a partícula atual $\mathbf{W}_l$.

5.5 Critério de avaliação

Como discutido na Seção 4.3, o critério de avaliação das partículas é muito importante para a capacidade de generalização do classificador treinado pelo IVABC. Em particular, o critério ψ_3 apresentado na Equação (4.3) é útil para evitar a ocorrência de sobreajuste. Por isso, esse critério foi adaptado para o IVABC através da função $\psi_{IVABC}(l)$, como mostra a Equação (5.7).

$$
\psi_{IVABC}(l) = \eta_1 \psi_1(l) + \eta_2 J_{IVABC}(l), \tag{5.7}
$$

onde η_1 e η_2 são pesos escolhidos do intervalo $[0;1]$, tal que $\eta_1 + \eta_2 = 1$ e $\psi_1(l) = N_{erros,l}/N$ é a taxa de erro de classificação da l-ésima partícula. A classificação é feita segundo o processo explicado na Seção 5.7. Os valores de $\psi_1(l)$ situam-se no intervalo $[0;1]$. A função $J_{IVABC}(l)$ é calculada pela Equação (5.8).

$$
J_{IVABC}(l) = \frac{\sum_{m=1}^{M} \sum_{i=1}^{N} \frac{d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right) \delta(m,l,k,i)}{ki}}{N_{acertos,l}}, \tag{5.8}
$$

onde $\delta(m,l,k,i) = 1$ se o m-ésimo protótipo da l-ésima partícula for um dos k protótipos mais próximos da i-ésima instância e $y_i = y_{m,l}$, caso contrário $\delta(m,l,k,i) = 0$. $d\pi_{m,l}$ é uma distância

Euclidiana quadrática intervalar entre os vetores $\vec{x}_i$ e $\vec{w}_{m,l}$. Essa distância é adaptativa e tem pesos diferentes para cada protótipo. É fundamental manter os valores da função J_{IVABC} no intervalo $[0;1]$ (mesmo intervalo de ψ_1). Portanto, é necessário seguir os seguintes passos: (i) as distâncias entre instâncias e seus protótipos mais próximos são divididos por *ki*, que representa quantos protótipos entre os k mais próximos ao i-ésimo objeto pertencem a mesma classe; (ii) o total da soma das distâncias é normalizado por $N_{acertos,l}$, que é o número de instâncias que possuem vizinhos mais próximos da mesma classe; e (iii) a distância $d\pi_{m,l}$ deve ser calculada usando a Equação (5.9).

$$d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)=\frac{\sum_{j\in\Upsilon_l}\lambda_{m,l,j}\left[\left(\frac{x_{i,j,L}-w_{m,l,j,L}}{max(j_U)-min(j_L)}\right)^2+\left(\frac{x_{i,j,U}-w_{m,l,j,U}}{max(j_U)-min(j_L)}\right)^2\right]}{2\cdot|\Upsilon_l|}, \tag{5.9}$$

onde Υ_l é o conjunto de índices de variáveis para os quais $W_{0,l,j}>0,5$, i.e., o conjunto de variáveis usadas pela l-ésima partícula, $max(j_U)$ e $min(j_L)$ são respectivamente os valores máximos dos limites superiores e os valores mínimos dos limites inferiores dos intervalos que correspondem à j-ésima variável e $\lambda_{m,l,j}$ é o valor que modela a dispersão da j-ésima variável na sub-região representada pelo m-ésimo protótipo da l-ésima partícula. Ao dividir os termos da distância por $(max(j_U)-min(j_L))$ e a soma dos termos por $2\cdot|\Upsilon_l|$, os valores da distância são mantidos no intervalo $[0,1]$.

O algoritmo IVABC tem o objetivo de minimizar a função $\psi_{IVABC}(l)$ apresentada pela Equação (5.7). Portanto, as probabilidades para a fase das abelhas observadoras podem ser calculadas como mostra a Equação (5.10).

$$p_l=\frac{f(l)}{\sum_{k=1}^{SN}f(k)} \tag{5.10}$$

onde $f(l)$ é calculada pela Equação (5.11).

$$f(l)=\frac{1}{\psi_{IVABC}(l)+1} \tag{5.11}$$

5.6 Pesos da distância adaptativa

Os pesos da distância $d\pi_{m,l}$ permitem-na modelar conjuntos de dados compostos de classes com várias formas e estruturas e são calculados pelo método dos multiplicadores de Lagrange. A distância tem três níveis de adaptabilidade: (i) os pesos λ representam o efeito das variáveis

sobre os protótipos; (ii) os pesos γ representam a influência do protótipo na sua classe g e (iii) os pesos β representam a importância da classe no conjunto de dados.

Considere que cada classe g, $(g = 1, \ldots, G)$, tem M_g protótipos. Sejam $\vec{\lambda}_{m,l}$, $\vec{\gamma}_{l,g}$ e $\vec{\beta}_l$ três vetores que satisfazem as seguintes condições:

1. $\lambda_{m,l,j} > 0$ e $\prod_{j \in \Upsilon_l} \lambda_{m,l,j} = \gamma_{m,l,g}$;
2. $\gamma_{m,l,g} > 0$ e $\prod_{m \in \Omega_g} \gamma_{m,l,g} = \beta_{l,g}$;
3. $\beta_{l,g} > 0$ e $\prod_{g=1}^{G} \beta_{l,g} = 1$;

onde Ω_g é o conjunto de protótipos que pertencem à classe g. A cardinalidade desse conjunto é M_g. Todos esses pesos são calculados de acordo com o método dos multiplicadores de Lagrange. Os vetores $\vec{\lambda}_{m,l}$ são dados pela Equação (5.12).

$$\lambda_{m,l,j} = \frac{\{\gamma_{m,l,g}[\prod_{h \in \Upsilon_l} \Delta_{m,l,h}]\}^{\frac{1}{|\Upsilon_l|}}}{\Delta_{m,l,j}}, \tag{5.12}$$

onde $\Delta_{m,l,j}$ é dado pela Equação (5.13).

$$\Delta_{m,l,j} = \sum_{i=1}^{N} \frac{\left[\left(\frac{x_{i,j,L} - w_{m,l,j,L}}{\max(j_U) - \min(j_L)}\right)^2 + \left(\frac{x_{i,j,U} - w_{m,l,j,U}}{\max(j_U) - \min(j_L)}\right)^2\right] \delta(m,l,k,i)}{2 \cdot |\Upsilon_l| \cdot ki \cdot N_{\text{acertos},l}} \tag{5.13}$$

Os valores dos pesos γ são calculados como mostra a Equação (5.14).

$$\gamma_{m,l,g} = \frac{\left[\beta_{l,g} \prod_{r \in \Omega_g} \left(\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{r,l}, \vec{\lambda}_{r,l}\right) \delta(r,l,k,i) N_{r,l,k,g}^{-1}\right)\right]^{\frac{1}{M_g}}}{\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right) \delta(m,l,k,i) N_{m,l,k,g}^{-1}}, \tag{5.14}$$

onde $N_{m,l,k,g}$ é o número de ocorrências do m-ésimo protótipo da l-ésima partícula como um dos k vizinhos mais próximos das instâncias de treinamento que pertencem a sua classe g.

A motivação dos pesos γ é que protótipos que estão em média localizados longe das suas instâncias afetadas tem pesos γ menores, o que resulta em pesos λ potencialmente menores e, portanto, tornam-se capazes de afetar mais instâncias.

A Equação (5.14) resulta em erro se $\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right) \delta(m,l,k,i) N_{m,l,k,g}^{-1} = 0$, i.e., se um protótipos não afetar corretamente instância alguma. Para resolver esse problema, todos os protótipos que satisfazem essa condição em cada iteração são removidos do conjunto de protótipos. Se uma classe g ficar sem protótipos na l-ésima partícula, um novo conjunto de protótipos é selecionado a partir do conjunto de treinamento para g.

Os pesos β são calculados usando a Equação (5.15).

$$\beta_{l,g} = \frac{\left[\prod_{a=1}^{G}\left(\sum_{m=1}^{M_a}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right)\delta(m,l,k,i)N_a^{-1}\right)\right]^{\frac{1}{G}}}{\sum_{m=1}^{M_g}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right)\delta(m,l,k,i)N_g^{-1}} \tag{5.15}$$

onde N_g é o número de ocorrências dos protótipos da classe g entre os k vizinhos mais próximos de instâncias da classe g. O Apêndice A apresenta a derivação completa dos pesos segundo o método dos multiplicadores de Lagrange.

A Equação (5.15) resulta em erro se $\sum_{m=1}^{M_g}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right)\delta(m,l,k,i)N_g^{-1} = 0$, i.e., se nenhum protótipo da classe g afetou instâncias corretamente. Nesse caso, todos os pesos β da l-ésima partícula são fixados em $1,0$ para essa iteração.

Os pesos da distância usada pelo algoritmo WFLVQ [63] eram calculados em dois passos. Isso poderia levar a perda de informação. Aqui, os pesos da distância $d\pi_{m,l}$ são calculados em um passo, imediatamente antes de calcular o critério $\psi_{IVABC}(l)$ para a l-ésima partícula (passos 4, 13, 30 e 42 do Algoritmo 4). Inicialmente, os pesos β são calculados, em seguida os pesos γ e por último os pesos λ.

5.7 Atribuição de rótulos

O processo de atribuição de rótulos usado pelo IVABC é baseado no método SO-NN [2]. Aqui, o i-ésimo objeto de teste é atribuído à g-ésima classe que tem o maior peso $\Theta_{g,i}$ entre as classes dos k vizinhos mais próximos. O vetor de pesos de classe pode ser calculado ponderando a contribuição de cada um dos M protótipos da l-ésima partícula de acordo com o inverso da dissimilaridade entre o protótipo e a instância, i.e., $\theta_{m,l,i} = 1/d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right)$. Com isso, o peso total $\Theta_{g,i}$ da g-ésima classe em relação à i-ésima instância é dado pela Equação (5.16).

$$\Theta_{g,i} = \sum_{m=1}^{k} \theta_{m,l,i}\nu(g,m,l), \tag{5.16}$$

onde $\nu(g,m,l) = 1$ se o m-ésimo protótipo da l-ésima partícula pertencem à g-ésima classe e $\nu(g,m,l) = 0$, caso contrário.

A classe estimada da i-ésima instância, $\hat{y}_i$, é então atribuída como g, tal que g é a classe com maior peso, i.e., $g = \arg\max(\Theta_{g,i}|1 \leq g \leq G)$. Se existem protótipos entre os k vizinhos mais próximos para os quais a distância é 0, então $\hat{y}_i = g$, onde g é a classe mais frequente entre esse protótipos. O método IVABC atribui rótulos para instâncias de treinamento antes de

calcular a qualidade de cada partícula (passos 5, 13, 30 e 42 do Algoritmo 4).

5.8 Pseudocódigo do IVABC

O Algoritmo 4 apresenta os passos detalhados do IVABC, considerando todas as contribuições discutidas nesta Seção. A nova codificação das partículas está presente nos passos de inicialização (passos 1 e 41) e na fase de abelhas empregadas (passo 12). O algoritmo atualiza os pesos da distância relativos aos protótipos de uma partícula e calcula as distâncias entre instâncias de treinamento e esses protótipos nos passos 4, 5, 13, 30 e 42. Além disso, para avaliar a qualidade da partícula (passos 5, 13, 30 e 42), o algoritmo atribui rótulos a essas instâncias, resultando numa taxa de erros de classificação.

Algoritmo 4 IVABC

Inicialize SN fontes de alimento e suas velocidades;
Atribua valores para os pesos η_1 e η_2 do critério e para o limite de tentativas;
Inicialize ω_{max} e ω_{min};
Calcule os pesos da distância para todas as fontes de alimento;
Calcule o valor do critério para cada fonte de alimento, usando Equação (5.7);
Inicialize $pbest_l$ para cada fonte de alimento ($l = 1, \ldots, SN$) e $gbest$ para a população;
Enquanto o critério de parada não for atingido **Faça**:
Envie as abelhas empregadas:
    Para $l = 1 \rightarrow SN$ **Faça**:
        Selecione uma fonte de alimentos aleatória $\mathbf{W}_k$, $k \neq l$;
        Selecione um índice aleatório $0 \leq m \leq M$;
        Selecione uma variável aleatória j;
        Use a Equação (5.5) ou a Equação (5.6) para gerar uma nova fonte de alimento $\mathbf{W'}_l$;
        Calcule os pesos da distância e o novo valor do critério $\psi'_{IVABC}(l)$;
        Se $\psi'_{IVABC}(l) < \psi_{IVABC}(l)$ **Então**
            $\mathbf{W}_l \leftarrow \mathbf{W'}_l$
            $pbest_l \leftarrow \mathbf{W'}_l$
            $tentativas_l \leftarrow 0$
        Senão
            $tentativas_l \leftarrow tentativas_l + 1$
        Fim Se
    Fim Para
    Atualize $gbest = \mathbf{W}_c$, onde $c = arg\min \psi_{IVABC}(l), \forall l \in (1, \ldots, SN)$.
Envie as abelhas observadoras:
    Calcule as probabilidades das fontes de alimento, usando a Equação (5.10);
    Calcule ω, usando a formula adotada pelo método VABC [31];
    Para $l = 1 \rightarrow SN$ **Faça**:
        Selecione aleatoriamente uma fonte de alimento $\mathbf{W}_k$, $k \neq l$, baseado nas probabilidades;
        Selecione aleatoriamente r_1 e r_2 do intervalo $[0, 1]$ e c_1 e c_2 do intervalo $[0.5, 1]$;
        Atualize a velocidade $\mathbf{V}_k$, usando a Equação (5.3);
        Calcule a nova posição $\mathbf{W'}_k$, usando a Equação (5.4);
        Calcule os pesos da distância e o novo valor do critério $\psi'_{IVABC}(k)$;
        Se $\psi'_{IVABC}(k) < \psi_{IVABC}(k)$ **Então**
            $\mathbf{W}_k \leftarrow \mathbf{W'}_k$
            $pbest_l \leftarrow \mathbf{W'}_k$
            $tentativas_l \leftarrow 0$
        Senão
            $tentativas_l \leftarrow tentativas_l + 1$
        Fim Se
    Fim Para
Envie as abelhas exploradoras:
    Para $l = 1 \rightarrow SN$ **Faça**:
        Se $tentativas_l > limit$ **Então**
            Reinicialize $tentativas_l$, $\mathbf{W}_l$ e sua velocidade;
            Calcule os pesos da distância e o novo valor do critério $\psi_{IVABC}(l)$;
        Fim Se
    Fim Para
    Atualize $gbest = \mathbf{W}_c$, where $c = arg\min \psi_{IVABC}(l), \forall l \in (1, \ldots, SN)$.
Fim Enquanto

Retorne $gbest$ e seus pesos λ.

5.9 Experimentos com o IVABC

Nesta Seção, o desempenho do IVABC será avaliado em termos de taxa de erro de classificação, além da sua habilidade de cumprir seus objetivos, i.e. reduzir o número de variáveis, demonstrar robustez com relação à inicialização dos protótipos e eliminar protótipos inutilizados. O IVABC foi comparado ao WFLVQ e a três classificadores baseados em protótipos treinados por diferentes algoritmos de enxames: (i) VABC [31]; (ii) *improved self-adaptive particle swarm optimization* (IDPSO), uma versão do PSO que ajusta automaticamente os seus parâmetros [79]; e (iii) o PSO básico com inércia decrescente.

Os três métodos baseados em enxames que serão comparados ao IVABC usarão a codificação de partículas explicada na Seção 4.3, que é uma generalização para múltiplos protótipos por classe da codificação proposta por De Falco et al. [15], modificada para permitir protótipos intervalares. Assim, cada partícula é uma matriz $\mathbf{W}_l = \{\vec{w}_{1,l}, \ldots, \vec{w}_{M,l}\}$, cuja m-ésima linha representa o m-ésimo protótipo, composto por p coordenadas intervalares, e.g., $\vec{w}_{m,l} = \{w_{m,l,1}, \ldots, w_{m,l,p}\}$. Estes experimentos usarão tanto conjuntos de dados sintéticos quanto reais.

5.9.1 Conjuntos sintéticos

Os conjuntos sintéticos serão usados nesta análise porque oferecem um melhor entendimento sobre o funcionamento dos algoritmos. Quatro deles são considerados neste trabalho.

1. O primeiro tem duas classes elípticas e duas variáveis, mas apenas uma é importante para separar as classes;
2. O segundo conjunto tem duas classes compostas de uma região esférica e outra elíptica e duas variáveis importantes;
3. O terceiro tem duas classes hiper-elípticas e três variáveis, mas apenas duas contribuem para separar as classes;
4. Por último, o quarto conjunto é similar ao segundo, mas com muito mais sobreposição entre as classes.

Cada sub-região de classe é obtida de uma distribuição Gaussiana com componentes não-correlacionados, como descrito em [8]. Cada ponto (z_1, z_2) é usado para gerar um vetor de intervalos (um hipercubo de p dimensões, onde p é o número de variáveis), definido como:

$[([z_1 - \varphi_1/2; z_1 + \varphi_1/2], [z_2 - \varphi_2/2; z_2 + \varphi_2/2])]$. Os parâmetros φ_1 e φ_2 são selecionados aleatoriamente do intervalo $[1; 10]$ e, para preservar a estrutura definida para cada sub-região, os dados intervalares de cada sub-região são obtidos usando o mesmo par (φ_1, φ_2).

Os parâmetros para geração dos conjuntos são mostrados nas Tabelas 5.5-5.8 e os conjuntos de dados resultantes podem ser vistos nas Figuras 5.1-5.4.

Tabela 5.5: **Conjunto 1** - Parâmetros das distribuições Gaussianas do primeiro conjunto

Classe 1				
$\mu_1 = 99$	$\mu_2 = 99$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$n = 200$
Classe 2				
$\mu_1 = 108$	$\mu_2 = 99$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$n = 200$

Figura 5.1: **Conjunto 1** - dados gerados pelas distribuições Gaussianas

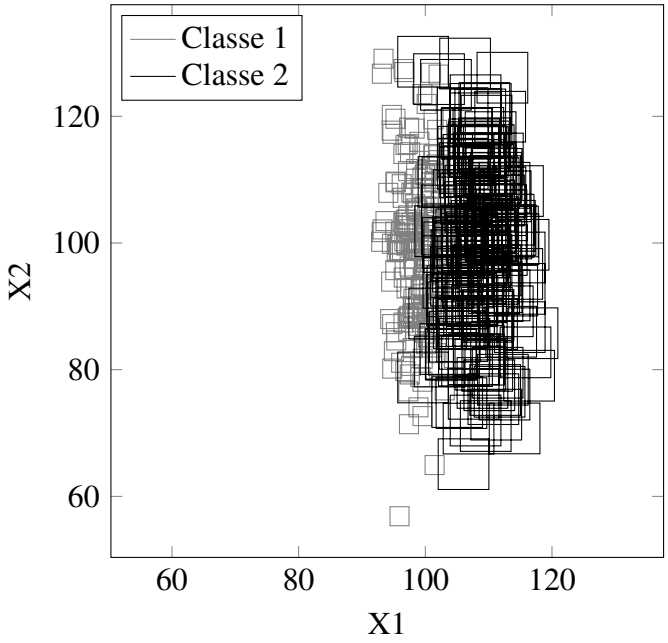


Tabela 5.6: **Conjunto 2** - Parâmetros para as distribuições Gaussianas do segundo conjunto

Classe 1				
$\mu_1 = 99$	$\mu_2 = 99$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$n = 200$
$\mu_1 = 104$	$\mu_2 = 138$	$\sigma_1^2 = 16$	$\sigma_2^2 = 16$	$n = 150$
Classe 2				
$\mu_1 = 108$	$\mu_2 = 99$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$n = 200$
$\mu_1 = 104$	$\mu_2 = 60$	$\sigma_1^2 = 16$	$\sigma_2^2 = 16$	$n = 150$

Figura 5.2: **Conjunto 2** - dados gerados pelas distribuições Gaussianas

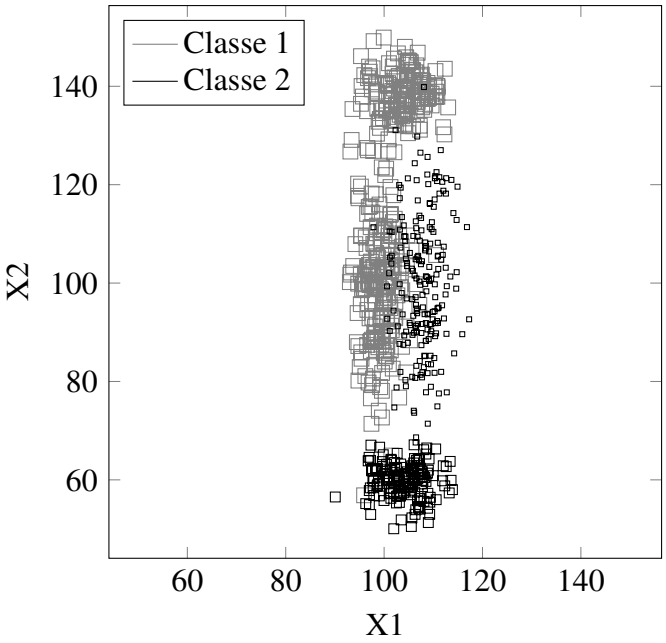


Tabela 5.7: **Conjunto 3** - Parâmetros para as distribuições Gaussianas do terceiro conjunto

Classe 1						
$\mu_1 = 99$	$\mu_2 = 99$	$\mu_3 = 44$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$\sigma_3^2 = 25$	$n = 200$
$\mu_1 = 104$	$\mu_2 = 138$	$\mu_3 = 44$	$\sigma_1^2 = 16$	$\sigma_2^2 = 16$	$\sigma_3^2 = 25$	$n = 150$
Classe 2						
$\mu_1 = 108$	$\mu_2 = 99$	$\mu_3 = 41$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$\sigma_3^2 = 25$	$n = 200$
$\mu_1 = 104$	$\mu_2 = 60$	$\mu_3 = 41$	$\sigma_1^2 = 16$	$\sigma_2^2 = 16$	$\sigma_3^2 = 25$	$n = 150$

Figura 5.3: **Conjunto 3** - dados gerados pelas distribuições Gaussianas

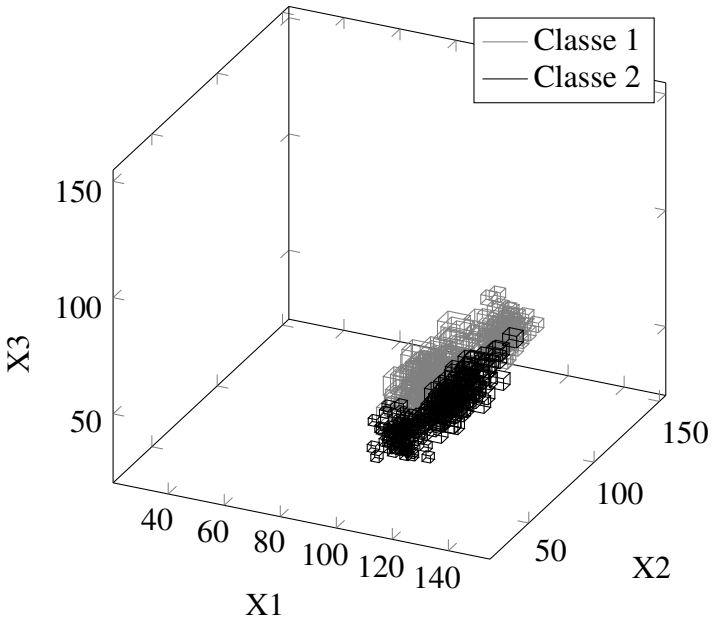
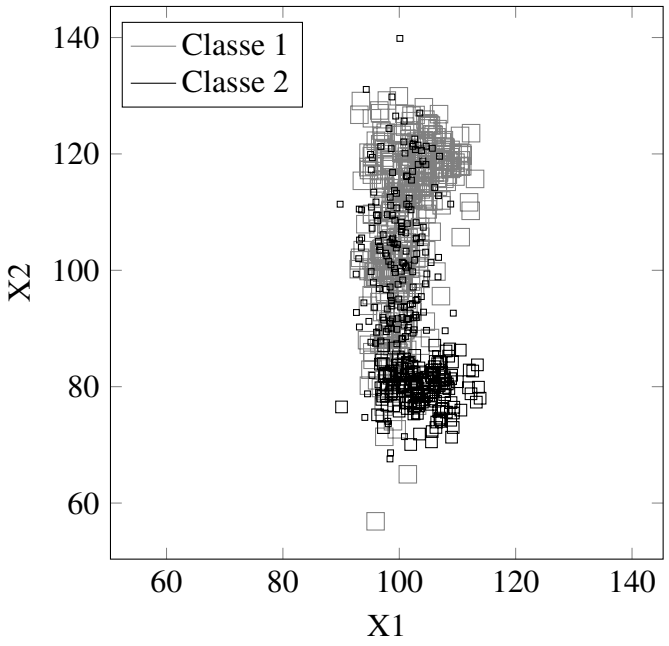


Tabela 5.8: **Conjunto 4** - Parâmetros para as distribuições Gaussianas do quarto conjunto

Classe 1				
$\mu_1 = 99$	$\mu_2 = 99$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$n = 200$
$\mu_1 = 104$	$\mu_2 = 118$	$\sigma_1^2 = 16$	$\sigma_2^2 = 16$	$n = 150$
Classe 2				
$\mu_1 = 100$	$\mu_2 = 99$	$\sigma_1^2 = 9$	$\sigma_2^2 = 169$	$n = 200$
$\mu_1 = 104$	$\mu_2 = 80$	$\sigma_1^2 = 16$	$\sigma_2^2 = 16$	$n = 150$

Figura 5.4: **Conjunto 4** - dados gerados pelas distribuições Gaussianas



5.9.2 Conjuntos de dados reais

Quatro conjuntos de dados reais são usados nesta análise de desempenho: um conjunto de cogumelos e três conjuntos de climas. O conjunto de cogumelos foi apresentado por Xu [76]. Ele continham 274 observações de várias espécies de fungos extraídas de MykoWeb [50]. O conjunto tem três variáveis intervalares: largura do píleo e comprimento e espessura do estipe. As instâncias foram rotuladas como comestíveis ou não-comestíveis de acordo com informações extraídas de MykoWeb [50]. As instâncias cujos rótulos não puderam ser determinados foram removidas do conjunto. O conjunto resultante tem 154 observações (114 comestíveis e 40 não-comestíveis). Esses dados estão disponíveis em http://goo.gl/GsBMTQ.

Os três conjuntos de dados de climas foram apresentados por Silva Filho e Souza [63]. O primeiro tem 1415 cidades/regiões e dez classes: continental úmido, deserto, equatorial, mediterrâneo, monção, oceânico, savana, semiárido, subártico e subtropical úmido. Os outros dois conjuntos são subconjuntos do primeiro: climas secos, com três climas (deserto, semiárido e savana) e 522 instâncias e climas da Europa ocidental (mediterrâneo e oceânico), com 324 cidades ou regiões. Esses três conjuntos tem as mesmas 16 variáveis intervalares: 12 variáveis que representam as temperaturas mínimas e máximas de cada mês (começando pelo primeiro mês de verão) e 4 variáveis que representam o nível de precipitação mínimo e máximo em cada estação (começando com verão).

5.9.3 Intervalos de confiança

Os cinco algoritmos (IVABC, WFLVQ, VABC, IDPSO and PSO) tiveram suas taxas de erros de classificação medidas usando os oito conjuntos de dados mencionados nessa Seção. Para fornecer resultados visualizáveis, essa análise utilizará intervalos de confiança não-paramétricos para as médias dos resultados, construídos por *bootstrap*.

Bootstrap é um método estatístico para reproduzir distribuições de probabilidades dos dados de forma aleatória e não-paramétrica. Após a criação dos intervalos para as médias, caso haja interseção de um ou mais intervalos, essas médias são consideradas iguais com um certo nível de confiança. Caso contrário, uma média é considerada maior do que a outra.

Seja $\varepsilon = \{e_1, e_2, \cdots, e_s\}$ um conjunto de valores de taxa de erro. Um intervalo de confiança para a média pode ser construído sorteando B instâncias de tamanho s de ε com reposição. Para cada instância, a média é computada. Essas médias são ordenadas para encontrar valores de percentis. Com um nível de confiança de $(1-\alpha)100\%$, $(0 \leq \alpha \leq 1)$, o intervalo de confiança é dado pelos elementos que correspondem aos índices round $\left(B \cdot \frac{\alpha}{2}\right)$ e round $\left(B \cdot (1-\frac{\alpha}{2})\right)$ [47]. O Algoritmo 5 apresenta uma forma compacta de construir esses intervalos [47].

Algoritmo 5 Construindo um intervalo de confiança para a média das taxas de erro usando bootstrap

Requer: Conjunto $\varepsilon = \{e_1, e_2, \cdots, e_s\}$ de valores de taxas de erro; número de execuções B; tamanho da amostra s

1: **Para** b=1:B **Faça**
2: instância$_b \leftarrow$ uma amostra de ε, com reposição e tamanho s;
3: m$_b \leftarrow$ média da b-ésima amostra;
4: **Fim Para**
5: Ordene as médias $m_1, m_2, \cdots, m_B$ em ordem crescente;
6: min $\leftarrow$ round $\left(B \cdot \frac{\alpha}{2}\right)$;
7: max $\leftarrow$ round $\left(B \cdot (1-\frac{\alpha}{2})\right)$;
8: $c_a \leftarrow m_{min}$;
9: $c_b \leftarrow m_{max}$;

Retorne Intervalo de confiança: $[c_a; c_b]$.

5.9.4 Metodologia

Os parâmetros dos algoritmos tem uma grande importância para seus desempenhos. Portanto, esses valores foram escolhidos empiricamente, através de validação-cruzada. Todos os métodos

baseados em enxames usaram 25 fontes de alimento para todos os conjuntos de dados. Outros valores de parâmetros foram:

- IVABC: $limit = 10$, $\omega_{max} = 0,9$, $\omega_{min} = 0,4$, $v_{max} = 1$, $v_{min} = -1$ e o peso η_1 do critério ψ_{IVABC} na Equação (5.7) toma valores do conjunto {0,0; 0,2; 0,4; 0,5; 0,6; 0,8; 1,0}, com η_2 variando de acordo;
- WFLVQ: $\alpha_m(0) = 0,3$;
- VABC: $limit = 10$, $\omega_{max} = 0,9$, $\omega_{min} = 0,4$, $v_{j,max} = (max(j_U) - min(j_L))$, $v_{j,min} = -(max(j_U) - min(j_L))$, $\forall j \in (1,\ldots,p)$;
- IDPSO: além dos mesmo valores de parâmetros usados para o VABC, $c_1(0) = c_2(0) = 2$;
- PSO: mesmos valores que os usados para o IDPSO, exceto $v_{max} = 0,05$ e $v_{min} = -0,05$.

Todos os quatro métodos de enxames foram executados por 200 iterações ou até o *gbest* atingir 50 iterações sem alterações. WFLVQ foi executado até o erro de validação aumentar por três iterações consecutivas. Os números de protótipos para todos os métodos e o valor de k usado pelo IVABC foram escolhidos através de validação-cruzada. Esses valores foram:

- Conjunto sintético 1: 20 protótipos para ambas as classes - $k = 3$ para o IVABC;
- Conjunto sintético 2: 20 protótipos para ambas as classes - $k = 21$ para o IVABC;
- Conjunto sintético 3: 20 protótipos para ambas as classes - $k = 2$ para o IVABC;
- Conjunto sintético 4: 20 protótipos para ambas as classes - $k = 21$ para o IVABC;
- Cogumelo: 7 para a classe comestível e 2 para a classe não-comestível - $k = 3$ para o IVABC;
- Dez climas: 34 para continental úmido, 28 para deserto, 12 para equatorial, 20 para mediterrâneo, 12 para monção, 42 para oceânico, 38 para savana, 34 para semiárido, 6 para subártico e 40 para subtropical úmido - $k = 8$ para o IVABC;
- Climas secos: 28 para deserto, 38 para savana e 34 para semiárido - $k = 21$ para o IVABC;
- Climas da Europa ocidental: 20 para mediterrâneo e 40 para oceânico - $k = 3$ para o IVABC.

Para os conjuntos sintéticos, os métodos foram repetidos 100 vezes num experimento Monte Carlo. Em cada replicação, um novo conjunto de dados foi gerado de acordo com as distribuições Gaussianas. Para os conjuntos de dados reais, os experimentos Monte Carlo consistem de 10 repetições de validação cruzada com 10 partições. Portanto, cada experimento gerou 100 taxas de erro.

5.9.5 Análise de convergência

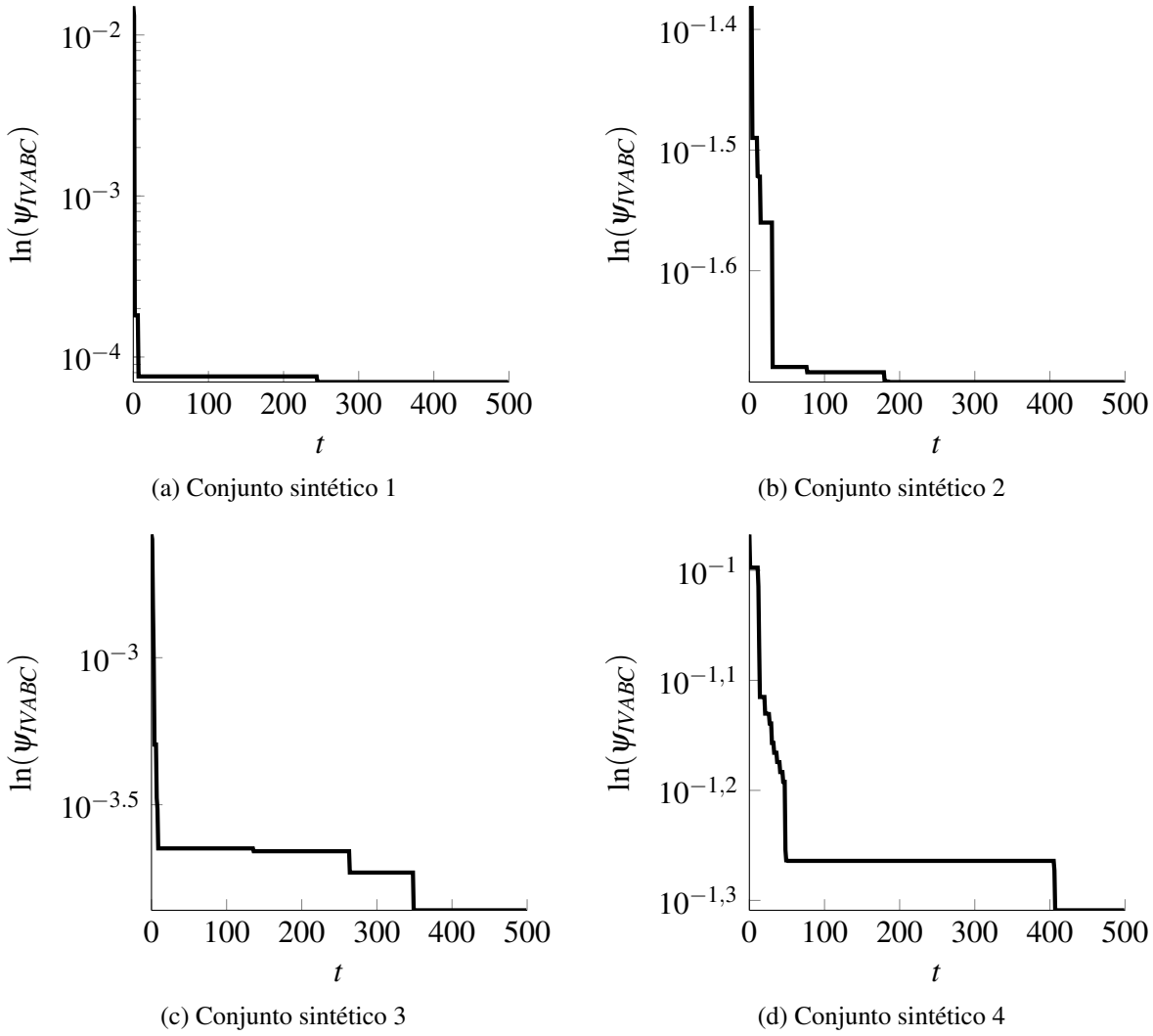
Antes da análise de performance, é importante verificar a convergência do critério ψ_{IVABC}. Para isso, o IVABC foi executado por 500 iterações com todos os oito conjuntos, usando os parâmetros mencionados na Seção 5.9.4. As Figuras 5.5 e 5.6 mostram que o valor do critério decresce rapidamente nas primeiras iterações ($t < 200$), com ajustes menores em iterações posteriores. Em quase todos os casos – a única exceção é o conjunto de climas secos–, o valor do critério estava estacionado por cerca de 100 ou mais iterações ao final do experimento. Esses resultados justificam os valores escolhidos para os parâmetros do IVABC.

5.9.6 Resultados

A Tabela 5.9 mostra os resultados e as Figuras 5.7-5.14 apresentam intervalos de confiança construídos com $B = 1000$ e $\alpha = 0,05\%$ para as médias das taxas de erro, em %. Em todas as figuras, a subfigura (a) apresenta os resultados do método IVABC para vários valores do peso η_1 do critério ψ_{IVABC}. Os pesos η_2 são calculados baseado nos valores de η_1: $\eta_2 = 1,0 - \eta_1$. Além disso, a subfigura (b) mostra a comparação entre o intervalo de confiança da melhor versão do IVABC com os intervalos dos outros métodos. Se ocorrerem interseções nos intervalos de confiança das versões do IVABC, aquela que tiver a menor média é escolhida para a comparação com os outros algoritmos.

A Figura 5.7 mostra que a melhor versão do algoritmo IVABC para o primeiro conjunto sintético usa os valores $\eta_1 = 0,6$ e $\eta_2 = 0,4$. Além disso, seu desempenho foi muito superior ao dos outros algoritmos. Como esperado, o IVABC e os outros métodos de enxames apresentaram intervalos menores do que o WFLVQ, com o IVABC mostrando o menor intervalo. Isso significa maior robustez contra a inicialização dos protótipos.

Para o conjunto sintético 2 (Figura 5.8), a melhor versão do IVABC foi $\eta_1 = 0,8$ e $\eta_2 = 0,2$ e os métodos de enxames tiveram intervalos de confiança de tamanhos similares e menores do que os do WFLVQ. IVABC e VABC tiveram os melhores resultados, com interseção dos intervalos, ou seja, seus desempenhos são considerados estatisticamente iguais.



(a) Conjunto sintético 1

(b) Conjunto sintético 2

(c) Conjunto sintético 3

(d) Conjunto sintético 4

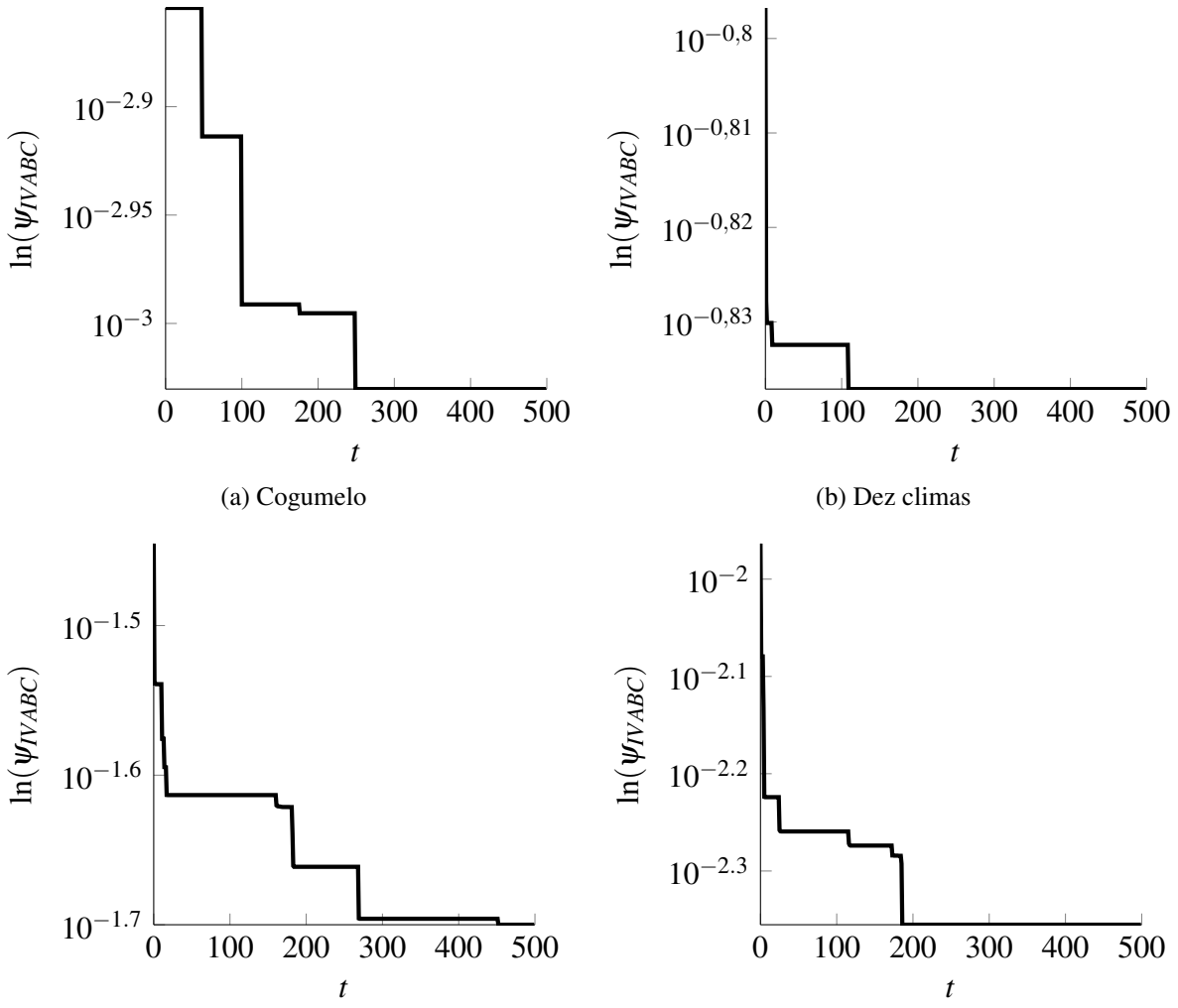
Figura 5.5: **Curvas de convergência do IVABC para os conjuntos sintéticos**

A melhor versão do algoritmo IVABC para o conjunto sintético 3 (Figura 5.9) usou os valores $\eta_1 = 0,2$ e $\eta_2 = 0,8$ e mostrou o melhor desempenho entre os métodos comparados. Quanto ao tamanho dos intervalos, VABC e IDPSO apresentaram os menores intervalos.

Para o conjunto sintético 4, a Figura 5.10 mostra que a melhor versão do IVABC usou $\eta_1 = 0,5$ e $\eta_2 = 0,5$ e apresentou o melhor desempenho e o menor intervalo entre os algoritmos comparados.

Para o conjunto cogumelo (Figura 5.11), a melhor versão do IVABC ($\eta_1 = 0,0$ e $\eta_2 = 1,0$) foi estatisticamente similar ao WFLVQ e ao IDPSO. O IDPSO obteve o menor intervalo.

A Figura 5.12 mostra que o melhor IVABC para o conjunto dos dez climas ($\eta_1 = 1,0$ e $\eta_2 = 0,0$) obteve a menor taxa de erro média e o menor intervalo de confiança entre os métodos



(a) Cogumelo

(b) Dez climas

(c) Climas secos

(d) Climas da Europa ocidental

Figura 5.6: **Curvas de convergência do IVABC para os conjuntos reais**

comparados.

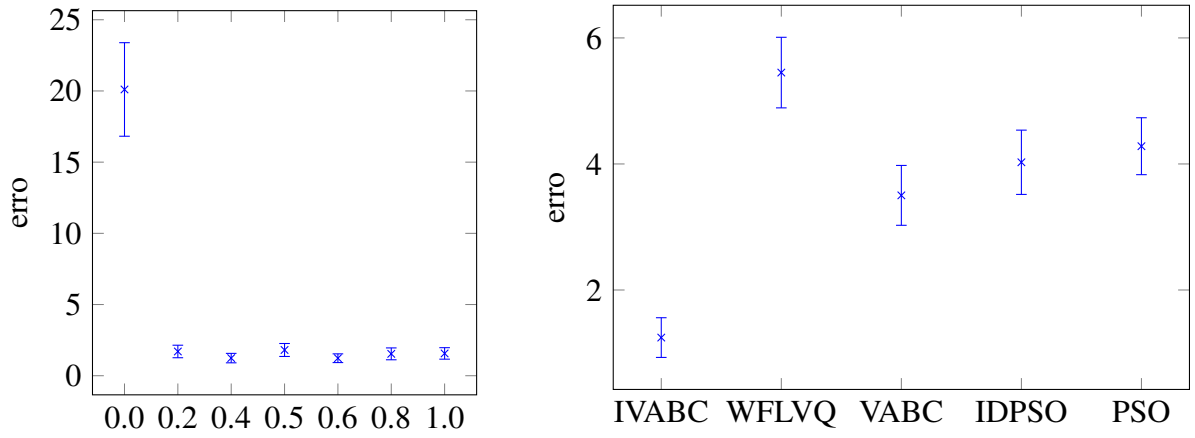
O IVABC com $\eta_1 = 0,5$ e $\eta_2 = 0,5$ teve o melhor desempenho para o conjunto dos climas secos, como mostra a Figura 5.13, com uma grande diferença para os outros métodos. Adicionalmente, ele apresentou o menor intervalo.

A Figura 5.14 apresenta os desempenhos para o conjunto dos climas da Europa ocidental. O IVABC com $\eta_1 = 0,2$ e $\eta_2 = 0,8$ teve desempenho estatisticamente igual ao WFLVQ e ambos foram melhores do que os outros algoritmos. O IVABC obteve o menor intervalo.

É importante observar que os valores de η_1 e η_2 mostraram que para cada conjunto de dados, um termo diferente de ψ_{IVABC} pode ser mais importante para obter uma melhor solução. Para três conjuntos de dados, o primeiro termo foi mais importante ($0,6 \leq \eta_1 \leq 1,0$), enquanto

Tabela 5.9: **Resultados de taxa de erro percentual** - A parte esquerda da tabela apresenta resultados do IVABC para diferentes valores de η_1; valores marcados em **negrito** representam versões do IVABC que foram significativamente melhores do que os outros algoritmos, de acordo com os intervalos de confiança apresentados pelas Figuras 5.7-5.14

	Versões do IVABC							Outros algoritmos			
Conjunto	0,0	0,2	0,4	0,5	0,6	0,8	1,0	WFLVQ	VABC	IDPSO	PSO
sintético 1	19,850	1,687	1,223	1,780	**1,223**	1,533	1,577	5,440	3,473	4,037	4,277
	(17,287)	(2,189)	(1,711)	(2,348)	**(1,592)**	(2,088)	(2,105)	(3,049)	(2,482)	(2,517)	(2,366)
sintético 2	17,032	3,781	2,745	2,970	2,733	2,709	2,762	6,540	2,543	3,482	3,362
	(17,412)	(2,179)	(1,669)	(1,364)	(1,492)	(1,436)	(1,402)	(2,311)	(1,382)	(1,524)	(1,453)
sintético 3	22,691	**2,101**	2,305	2,587	2,505	2,545	2.762	4,591	3,208	4,168	4,080
	(7,066)	**(1,415)**	(1,407)	(1,564)	(1,594)	(1,328)	(1,614)	(1,825)	(1,148)	(1,138)	(1,352)
sintético 4	23,036	7,695	7,011	**6,720**	7,798	7,699	7,339	16,551	12,693	15,608	15,914
	(9,811)	(6,199)	(5,784)	**(5,498)**	(5,887)	(6,644)	(5,104)	(8,109)	(7,576)	(6,806)	(7,052)
cogumelo	27,683	31,329	31,096	29,192	29,637	29,775	29,067	29,175	31,629	29,171	31,825
	(6,148)	(6,701)	(6,843)	(6,461)	(6,739)	(6,688)	(6,563)	(6,326)	(6,508)	(5,749)	(7,864)
dez climas	72,830	18,652	19,198	18,701	18,662	19,008	**18,560**	34,045	26,927	26,925	26,621
	(10,587)	(3,220)	(3,108)	(3,143)	(3,384)	(3,856)	**(2,912)**	(4,714)	(3,597)	(3,844)	(3,477)
climas secos	51,657	6,527	6,309	**5,890**	6,408	6,418	6,222	12,087	14,406	14,123	15,028
	(9,570)	(3,327)	(3,084)	**(2,951)**	(3,549)	(3,025)	(3,263)	(4,520)	(3,797)	(4,056)	(4,722)
Europeus	19,686	4,254	4,360	4,289	4,576	4,354	4,873	5,373	7,158	7,334	7,535
	(12,132)	(3,586)	(3,546)	(3,441)	(3,485)	(3,523)	(3,758)	(3,792)	(4,357)	(4,395)	(4,181)

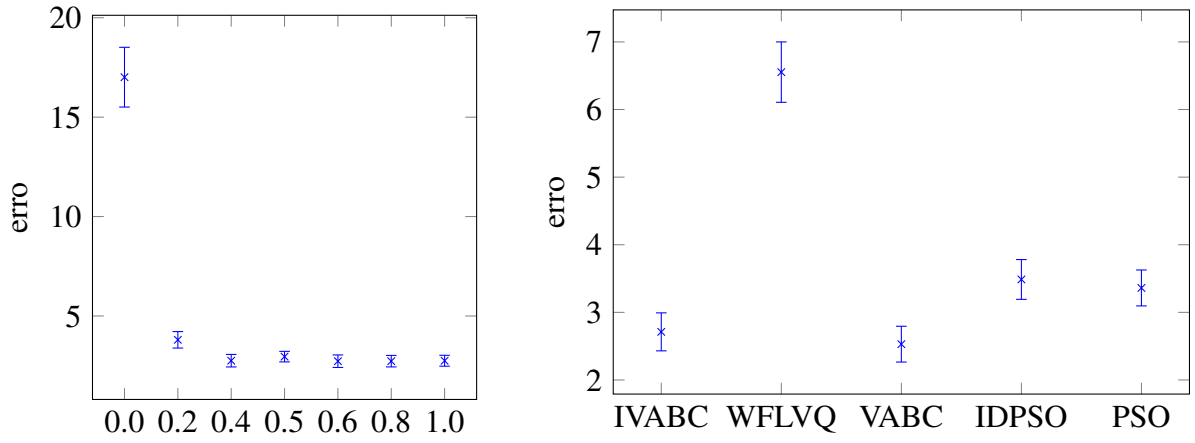


(a) Resultados para vários valores de η_1

(b) Comparação dos algoritmos

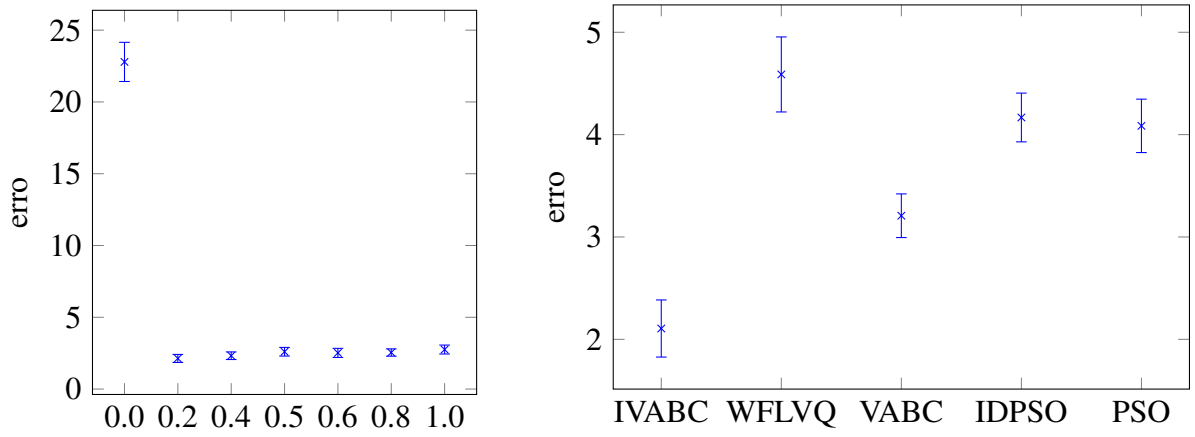
Figura 5.7: **Conjunto sintético 1** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 0,6$ e $\eta_2 = 0,4$) apresentou resultado superior aos dos outros métodos

que para três outros conjunto, o oposto foi verdadeiro ($\eta_2 = 0,8$ ou $\eta_2 = 1,0$). Para os dois conjuntos de dados restantes, ambos os termos foram igualmente importantes ($\eta_1 = \eta_2 = 0,5$). Além disso, enquanto a versão do IVABC que considera apenas o segundo termo do critério ($\eta_2 = 1,0$) foi a melhor para o conjunto cogumelo, ela foi sempre a pior em todos os outros conjuntos. Essa versão força o algoritmo a selecionar apenas a variável para a qual as diferenças entre os protótipos e suas instâncias afetadas são mínimas.



(a) Resultados para vários valores de η_1 (b) Comparação dos algoritmos

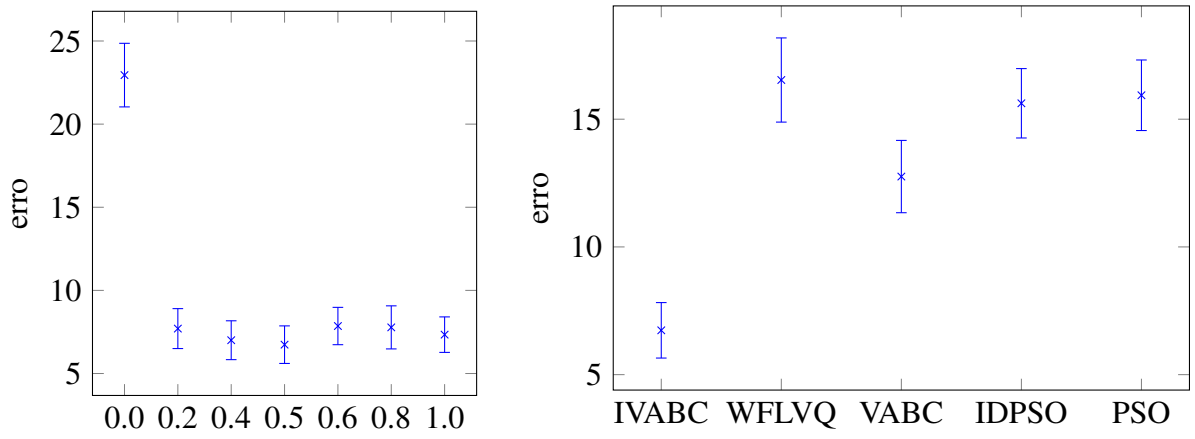
Figura 5.8: **Conjunto sintético 2** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 0,8$ e $\eta_2 = 0,2$) apresentou resultado equivalente ao do VABC e superior aos dos outros métodos



(a) Resultados para vários valores de η_1 (b) Comparação dos algoritmos

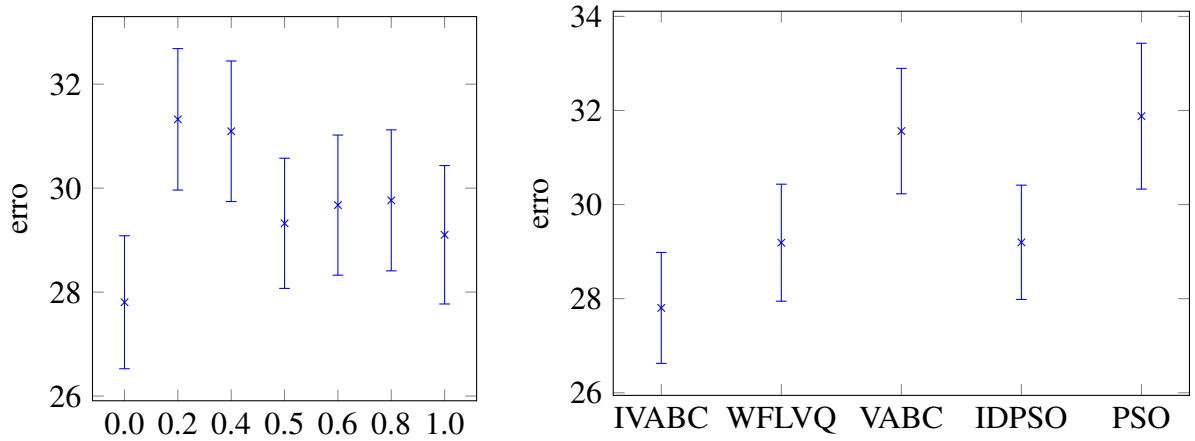
Figura 5.9: **Conjunto sintético 3** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 0,2$ e $\eta_2 = 0,8$) mostrou desempenho melhor do que os outros algoritmos

Os resultados apresentados confirmam a versatilidade e a utilidade do IVABC como um novo classificador para dados intervalares. Ele obteve as melhores taxas médias de erro em cinco de oito conjuntos de dados. Nos outros três conjuntos para os quais o IVABC não foi o único melhor algoritmo (conjunto sintético 2, cogumelo e climas da Europa ocidental), ele foi estatisticamente equivalente aos melhores algoritmos. Além disso, o IVABC obteve os menores intervalos de confiança em cinco conjuntos de dados, estando entre os menores nos outros



(a) Resultados para vários valores de η_1 (b) Comparação dos algoritmos

Figura 5.10: **Conjunto sintético 4** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 0,5$ e $\eta_2 = 0,5$) mais uma vez teve melhor desempenho

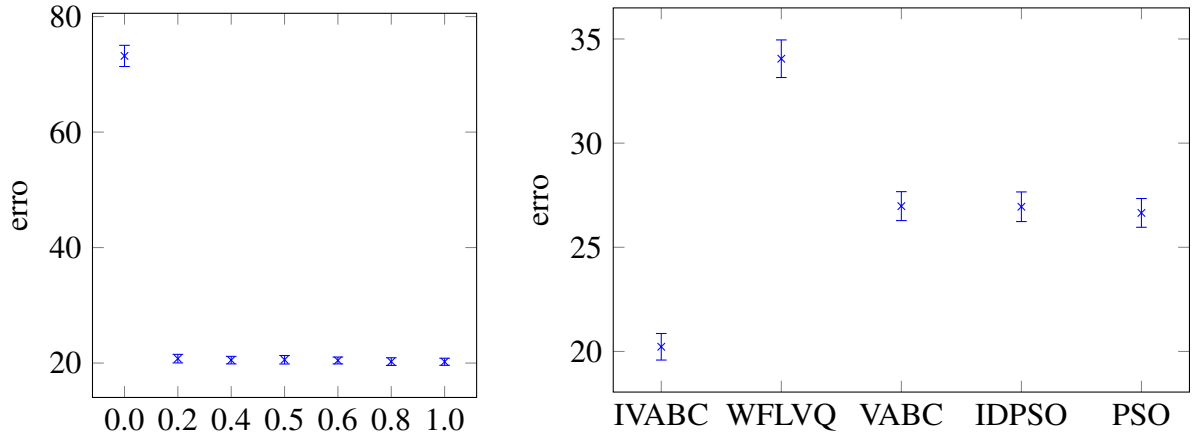


(a) Resultados para vários valores de η_1 (b) Comparação dos algoritmos

Figura 5.11: **Cogumelo** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 0,0$ e $\eta_2 = 1,0$) teve desempenho similar ao WFLVQ e ao IDPSO

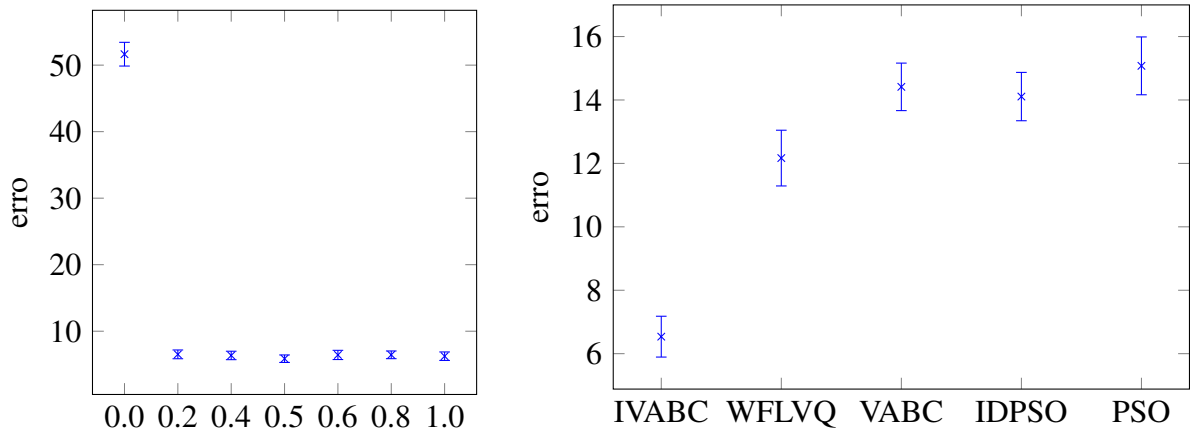
conjuntos. Ainda sobre o tamanho dos intervalos de confiança, o WFLVQ nunca apresentou o menor intervalo, como esperado. Isso significa que métodos baseados em enxames são mais capazes de escapar de mínimos locais causados pela inicialização dos protótipos.

WFLVQ foi o segundo melhor algoritmo (depois do IVABC) para o conjunto de climas secos e empatou com o IVABC para os conjuntos cogumelo e climas da Europa ocidental. Isso é devido a sua distância adaptativa, que é capaz de modelar as diferenças entre as escalas das variáveis de temperatura e precipitação. O IVABC teve desempenho ainda melhor porque sua distância não sofre de perda de informação como a distância usada pelo WFLVQ.



(a) Resultados para vários valores de η_1 (b) Comparação dos algoritmos

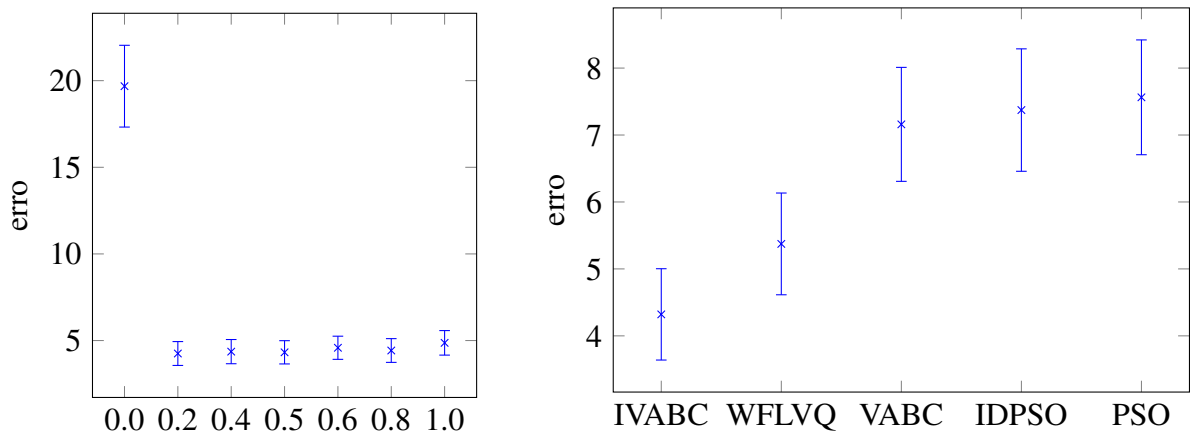
Figura 5.12: **Dez climas** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 1,0$ e $\eta_2 = 0,0$) teve desempenho médio muito superior ao de todos os outros métodos



(a) Resultados para vários valores de η_1 (b) Comparação dos algoritmos

Figura 5.13: **Climas secos** - intervalos de confiança para as médias das taxas de erro; assim como com o conjunto de dez climas, a melhor versão do IVABC ($\eta_1 = 0,5$ e $\eta_2 = 0,5$) obteve a menor taxa de erro média

Além disso, o IVABC também se beneficiou da sua habilidade de selecionar variáveis dinamicamente e reduzir o espaço do problema. Para observar essa habilidade, ao final de cada repetição Monte Carlo r, cada variável recebeu um valor $\delta_j^r = 1$, se ela foi considerada importante pelo algoritmo, ou $\delta_j^r = 0$, se ela foi descartada. Ao final de cada experimento, a porcentagem de vezes que uma variável foi usada é obtida usando $(\sum_{r=1}^{100} \delta_j^r)/100$. As seguintes variáveis foram consideradas importantes pela melhor versão do IVABC para cada conjunto de



(a) Resultados para vários valores de η_1

(b) Comparação dos algoritmos

Figura 5.14: **Climas da Europa ocidental** - intervalos de confiança para as médias das taxas de erro; melhor versão do IVABC ($\eta_1 = 0,2$ e $\eta_2 = 0,8$) teve desempenho similar ao WFLVQ e superior aos outros algoritmos

dados (com a porcentagem em parênteses):

- Conjunto sintético 1: X1 (95%);
- Conjunto sintético 2: X1 (97%) e X2 (81%);
- Conjunto sintético 3: X1 (77%) e X2 (71%);
- Conjunto sintético 4: X1 (79%);
- Cogumelo: comprimento do estipe (90%);
- Dez climas: temp-2 (69%), temp-8 (71%), prec-verao (99%), prec-outono (99%), prec-inverno (99%) e prec-primavera (99%);
- Climas secos: prec-verao (86%), prec-outono (100%), prec-inverno (96%) e prec-primavera (100%);
- Climas da Europa ocidental: prec-verao (100%) e prec-primavera (59%).

Esses números confirmam o que era esperado para os três primeiros conjuntos sintéticos. Apenas uma variável foi considerada importante para o conjunto sintético 1, ambas as variáveis foram consideradas para o conjunto sintético 2 e duas de três foram usadas para o conjunto sintético 3. Quanto ao conjunto sintético 4, a sobreposição entre as duas classes foi feita aproximando as sub-regiões do conjunto sintético 2, modificando as médias e mantendo as variâncias.

As mudanças foram maiores em X2, tornando essa variável menos discriminativa do que X1. Assim, o IVABC descartou X2.

Quanto aos conjuntos de dados reais, para o conjunto cogumelo, apenas o comprimento do estipe foi considerado importante. Os resultados para os conjuntos de climas foram interessantes, porque eles se encaixam de maneira intuitiva às discriminações de climas de K*ö*ppen [22]. Em se tratando de climas secos (deserto, semiárido e savana), suas temperaturas podem ser similares, sendo que a quantidade de precipitação ao longo do ano é o que realmente separa esses classes. Quanto aos climas da Europa ocidental, um clima mediterrâneo é definido por meses de verão muito secos, ao contrário de chuva regular em todas as estações em regiões que apresentam clima oceânico.

Mais variáveis foram consideradas para o conjunto dez climas, incluindo temperaturas no segundo mês de verão (temp-2) e de inverno (temp-8), mas nenhuma variável de temperatura de outono e primavera foi considerada. Isso ocorreu porque temperaturas são muito importantes para separar climas tropicais dos subtropicais, mas o outono e a primavera tendem a apresentar temperaturas amenas (não-discriminativas) não maioria dos climas. Portanto, a maioria desses atributos de temperatura foi descartada.

O IVABC e o WFLVQ compartilham a habilidade de remover protótipos inutilizados, embora o IVABC seja capaz de selecionar novos protótipos para uma classe, se a mesma terminar sem protótipos para representá-la. A Tabela 5.10 apresenta o número médio de protótipos removidos pela melhor versão do IVABC e pelo WFLVQ para cada conjunto de dados.

Tabela 5.10: **Eliminação de protótipos** - Número de protótipos removidos pelo IVABC e pelo WFLVQ e número total de protótipos M

Conjunto de dados	IVABC	WFLVQ	M
Conjunto sintético 1	0,96	7,21	40
Conjunto sintético 2	0,0	7,72	40
Conjunto sintético 3	1,79	5,96	40
Conjunto sintético 4	0,0	7,01	40
Cogumelo	0,63	0,2	9
Dez climas	2,99	75,37	266
Climas secos	0,02	24,31	100
Climas da Europa ocidental	1,48	4,65	60

Esse valores mostram que o WFLVQ remove muito mais protótipos. Sabendo que esses algoritmos descartam protótipos quando eles não afetam padrões de treinamento das suas clas-

ses, era esperado que o IVABC removesse menos protótipos. De fato, como o IVABC procura pelos k vizinhos mais próximos de cada objeto de treinamento, seu comportamento leva a menos protótipos inutilizados. Além do mais, o IVABC seleciona novos protótipos para qualquer classe que fique sem protótipos.

5.9.7 Matrizes de confusão para o conjunto de dez climas

Como a distância generalizada do IVABC tem a vantagem de permitir modelar classes desbalanceadas, é interessante analisar as matrizes de confusão produzidas pelos métodos analisados para um conjunto que apresente classes de tamanhos variados, como o conjunto de dez climas, cujas classes equatorial, monção e subártico são menos representadas do que as demais.

Para obter as matrizes de confusão apresentadas nesta Seção, os métodos foram treinados e avaliados 30 vezes com os mesmos conjuntos de treinamento e de teste, usando os mesmos valores de parâmetros mencionados na Seção 5.9.4. Dessa forma, cada instância de teste recebeu 30 predições de cada método, tomando-se a média dessas 30 predições (arredondada para o inteiro mais próximo) como a predição final para a construção da matriz.

As Tabelas 5.11-5.15 apresentam as matrizes de confusão dos métodos analisados. A g-ésima linha representa as instâncias da g-ésima classe e a g-ésima coluna representa as instâncias que foram atribuídas à g-ésima classe. Assim, a diagonal representa as instâncias corretamente classificadas.

Entre as instâncias que pertencem às três menores classes (equatorial, monção e subártico) o IVABC classificou 10 corretamente, de um total de 18, enquanto WFLVQ e PSO classificaram 9 e VABC e IDPSO atribuíram 8 às suas classes corretas. Além de atribuir mais instâncias das classes menos representadas às suas classes corretas, o IVABC também teve melhor desempenho de classificação geral, com 99 instâncias corretamente classificadas, contra 65 do WFLVQ, 73 do VABC, 72 do IDPSO e 78 do PSO. Apesar do pior desempenho do WFLVQ para classificação geral, quando se leva em consideração apenas as instâncias das classes menores, o WFLVQ só foi pior do que o IVABC. Isso pode se dever a sua distância generalizada, capaz de modelar classes desbalanceadas, como a distância do IVABC.

Tabela 5.11: **Matriz de confusão do IVABC para o conjunto com dez climas** - As classes são: continental úmido (1), deserto (2), equatorial (3), mediterrâneo (4), monção (5), oceânico (6), savana (7), semiárido (8), subártico (9) e subtropical úmido (10)

	1	2	3	4	5	6	7	8	9	10	Total
1	14	1	0	1	2	0	0	0	0	0	18
2	0	11	1	2	0	1	0	0	0	0	15
3	0	0	6	1	0	0	0	0	0	0	7
4	0	0	0	7	4	0	0	0	0	0	11
5	0	0	1	1	2	1	2	0	0	0	7
6	0	0	0	0	2	19	0	0	1	0	22
7	0	0	0	0	0	1	18	1	0	0	20
8	1	0	0	2	0	0	5	9	1	0	18
9	0	0	1	0	0	0	1	0	2	0	4
10	0	0	0	0	0	2	2	4	4	11	23
Predições	15	12	9	14	10	24	28	14	8	11	145

Tabela 5.12: **Matriz de confusão do WFLVQ para o conjunto com dez climas** - As classes são: continental úmido (1), deserto (2), equatorial (3), mediterrâneo (4), monção (5), oceânico (6), savana (7), semiárido (8), subártico (9) e subtropical úmido (10)

	1	2	3	4	5	6	7	8	9	10	Total
1	7	6	2	3	0	0	0	0	0	0	18
2	0	7	1	3	0	4	0	0	0	0	15
3	0	0	5	1	1	0	0	0	0	0	7
4	0	0	0	3	6	2	0	0	0	0	11
5	0	0	0	1	2	4	0	0	0	0	7
6	0	0	0	0	1	19	1	1	0	0	22
7	0	0	0	1	4	5	9	1	0	0	20
8	0	0	1	3	1	2	6	5	0	0	18
9	0	0	0	1	0	0	1	0	2	0	4
10	0	0	0	0	0	1	4	6	6	6	23
Predições	7	13	9	16	15	37	21	13	8	6	145

Tabela 5.13: **Matriz de confusão do VABC para o conjunto com dez climas** - As classes são: continental úmido (1), deserto (2), equatorial (3), mediterrâneo (4), monção (5), oceânico (6), savana (7), semiárido (8), subártico (9) e subtropical úmido (10)

	1	2	3	4	5	6	7	8	9	10	Total
1	10	2	4	1	1	0	0	0	0	0	18
2	0	6	4	1	2	0	2	0	0	0	15
3	0	0	3	4	0	0	0	0	0	0	7
4	0	0	0	4	2	3	2	0	0	0	11
5	0	0	0	2	2	1	2	0	0	0	7
6	0	0	1	1	2	17	0	0	1	0	22
7	0	0	0	0	0	3	17	0	0	0	20
8	1	1	0	0	5	2	6	2	1	0	18
9	0	0	0	0	0	0	1	0	3	0	4
10	0	0	0	0	0	1	3	5	5	9	23
Predições	11	9	12	13	14	27	33	7	10	9	145

Tabela 5.14: **Matriz de confusão do IDPSO para o conjunto com dez climas** - As classes são: continental úmido (1), deserto (2), equatorial (3), mediterrâneo (4), monção (5), oceânico (6), savana (7), semiárido (8), subártico (9) e subtropical úmido (10)

	1	2	3	4	5	6	7	8	9	10	Total
1	12	1	2	2	1	0	0	0	0	0	18
2	0	7	2	3	1	2	0	0	0	0	15
3	0	0	5	2	0	0	0	0	0	0	7
4	0	0	0	2	4	2	3	0	0	0	11
5	0	0	0	2	1	2	2	0	0	0	7
6	0	0	1	1	3	16	0	0	1	0	22
7	0	0	0	0	0	5	14	1	0	0	20
8	0	2	0	1	3	3	4	4	1	0	18
9	0	0	0	0	0	0	1	1	2	0	4
10	0	0	0	0	0	1	3	6	4	9	23
Predições	12	10	10	13	13	31	27	12	8	9	145

Tabela 5.15: **Matriz de confusão do PSO para o conjunto com dez climas** - As classes são: continental úmido (1), deserto (2), equatorial (3), mediterrâneo (4), monção (5), oceânico (6), savana (7), semiárido (8), subártico (9) e subtropical úmido (10)

	1	2	3	4	5	6	7	8	9	10	Total
1	11	3	2	1	0	1	0	0	0	0	18
2	0	6	3	3	1	0	2	0	0	0	15
3	0	0	5	1	1	0	0	0	0	0	7
4	0	0	0	3	4	2	2	0	0	0	11
5	0	0	0	2	2	2	1	0	0	0	7
6	0	0	1	2	1	17	0	0	1	0	22
7	0	0	0	0	0	2	17	1	0	0	20
8	1	1	0	2	3	3	3	5	0	0	18
9	0	0	0	0	0	0	1	1	2	0	4
10	0	0	0	0	0	0	5	5	3	10	23
Predições	12	10	11	14	12	27	31	12	6	10	145

5.10 Considerações finais

Esta Seção introduziu um classificador adaptativo para dados intervalares baseado em protótipos treinados por enxames, capaz de selecionar variáveis automaticamente e de descartar protótipos inativos. Os protótipos são treinados usando o algoritmo VABC [31], que ajuda a evitar mínimos locais causados pela inicialização dos protótipos. O novo método também usa uma distância Euclidiana quadrática adaptativa para dados intervalares que é capaz de modelar classes desbalanceadas e compostas de sub-regiões de diferentes formas e tamanhos. O método proposto (IVABC) também generalizou o critério de treinamento das partículas introduzido por De Falco et al. [15], para dar diferentes pesos aos termos, no entanto, os resultados obtidos mostraram que os valores $\eta_1 = \eta_2 = 0.5$ podem ser sempre escolhidos, sem prejuízo de desempenho.

Os experimentos reportados nesta Seção consideraram quatro conjuntos de dados intervalares sintéticos e quatro conjuntos reais e compararam o IVABC ao WFLVQ e a três classifi-

cadores baseados em protótipos treinados por diferentes métodos de enxames (VABC, IDPSO e PSO) e que usam a codificação de partículas proposta por De Falco et al. [15] adaptada para dados intervalares. Nos experimentos, o IVABC atingiu resultados melhores de taxa de erro de classificação para cinco dos oito conjuntos. Nos outros três conjuntos (conjunto sintético 2, cogumelo e climas da Europa ocidental), o IVABC foi estatisticamente igual ao melhor algoritmo. Esses resultados foram confirmados por intervalos de confiança não-paramétricos construídos com $\alpha = 0,05$.

Os resultados dos experimentos também mostraram que: (i) o IVABC foi capaz de selecionar variáveis, mantendo apenas as mais importantes; (ii) em comparação com o WFLVQ, o IVABC descartou menos protótipos, demonstrando melhor utilização dos protótipos; (iii) o uso de pesos diferentes para os termos do critério ψ_{IVABC} levou a soluções melhores para cada conjunto de dados, porém a escolha de valores iguais para os pesos não acarreta perda significativa de desempenho; (iv) a distância adaptativa do IVABC deu-lhe vantagem sobre os outros métodos de enxames, como visto nas matrizes de confusão para o conjunto de dez climas; (v) todos os quatro métodos de enxame tiveram intervalos de confiança menores do que os do WFLVQ, mostrando que são mais robustos contra mínimos locais (o IVABC apresentou os menores intervalos para cinco conjuntos). Quanto ao custo do IVABC em termos de tempo de processamento, seu treinamento é mais lento do que o do WFLVQ, o que era esperado, pois ele usa um método populacional para treinar os protótipos. Isso pode ser mitigado se um método populacional paralelizável for usado. Já em tempo de teste, o custo é o mesmo do WFLVQ.

6 PROPAGAÇÃO DE PROBABILIDADES USANDO PROTÓTIPOS E DISTÂNCIAS ADAPTATIVAS

Todas as verdades são fáceis de perceber depois de terem sido descobertas; o problemas é descobri-las.

—GALILEU

6.1 Introdução

Os métodos semi-supervisionados de propagação de rótulos usando protótipos (AGR), discutidos na Seção 2.3, têm o objetivo de diminuir o custo de construção do grafo e de propagação dos rótulos sem prejudicar o desempenho de classificação. No entanto, em muitas aplicações modernas, e.g. auxílio à tomada de decisões e classificação sensível a custos de erro, o praticante de Aprendizagem de Máquina pode estar mais interessado em saber as probabilidades de classe para uma nova instância do que um simples rótulo.

Os algoritmos de propagação de rótulos são projetados para respeitar duas condições: (i) os rótulos das instâncias rotuladas devem ser mantidos (ou minimamente modificados) e (ii) objetos similares ou próximos devem ter rótulos similares. Para tanto, o critério otimizado na etapa de propagação é composto de um termo para cada condição. Para transformar a propagação de rótulos em propagação de probabilidades, uma mudança precisa ser feita no critério a ser otimizado. O algoritmo resultante dessa adaptação é chamado propagação de probabilidades apropriadas usando protótipos (*proper probability propagation on prototypes* – 4Pros).

Outro ponto importante considerado nesta Seção é a importância do cálculo dos pesos locais para a construção do grafo de âncoras (protótipos) [74]. Considerando as distâncias discutidas na Seção 3 e a distância proposta na Seção 5, seria interessante explorar o uso de uma distância adaptativa para construir o grafo de protótipos.

6.2 Algoritmo

De forma resumida, o treinamento do método 4Pros é composto de 3 passos: (i) inicialização dos protótipos; (ii) construção do grafo; e (iii) propagação das probabilidades. Os dois primeiros passos são flexíveis: o praticante pode escolher quaisquer distâncias, algoritmos de agrupamento para inicialização dos protótipos e métodos de construção do grafo. O Algoritmo 6 mostra os passos do 4Pros e lista os seus parâmetros.

Algoritmo 6 Algoritmo 4Pros

Requer: Conjunto de treinamento $\mathfrak{I} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_i, y_i), \ldots, (\vec{x}_{n_l}, y_{n_l}), \vec{x}_{n_l+1}, \ldots, \vec{x}_N\}$; número de protótipos M; número de vizinhos k; peso do segundo termo da otimização ξ; função de escore apropriada s;

1: Obtenha um conjunto de protótipos Ω (e seus pesos λ se a distância for adaptativa);
2: Construa a matriz de pesos locais $\mathbf{Z}$ e a matriz de adjacências normalizadas $\mathbf{E}$;
3: Minimize a Equação (6.2) com a função s escolhida e obtenha e matriz $\mathbf{A}$;
4: Obtenha as probabilidades das instâncias não-rotuladas usando a Equação (6.1);

Retorne Protótipos Ω, matriz de probabilidades de classe dos protótipos $\mathbf{A}$ e, se a distância for adaptativa, os pesos λ.

6.3 Propagação de probabilidades

Como discutido na Seção 2.3.1, apesar de os métodos de propagação de rótulos não produzirem probabilidades, é possível obter probabilidades de classe, usando a função *softmax*. No entanto, a Seção 2.3.2 mostrou que essas probabilidades não são bem calibradas, pois o método não é treinado para otimizá-las. Portanto, se o objetivo for obter probabilidades de classe como resultado da propagação, o critério de propagação deve ser adaptado.

No algoritmo 4Pros, a propagação de probabilidades deve respeitar duas condições: (i) as probabilidades das instâncias rotuladas não devem ser modificadas (ou devem ser minimamente modificadas) e (ii) objetos próximos devem ter probabilidades de classe similares. Para isso, é necessário incorporar funções de escore apropriadas (do inglês *proper scoring rules*) ao critério a ser minimizado.

Uma função de escore $s(p,q)$ é apropriada se ela tiver seu valor mínimo (ou máximo, dependendo da função) quando as probabilidades corretas ($p = q$) forem fornecidas. Como essas funções avaliam probabilidades, seu domínio é o intervalo $[0;1]$. Portanto, a matriz de

valores preditivos de rótulos das instâncias rotuladas $\mathbf{Y}_{n_l}$ passa a tomar seus valores de $[0;1]^{n_l \times G}$ e o somatório da sua *i*-ésima linha (que corresponde ao vetor de probabilidades de classe da *i*-ésima instância) deve ser igual a 1.

Por ser uma matriz de probabilidades, $\mathbf{Y}_{n_l}$ tem um grau de liberdade para a definição das suas linhas, i.e. em um problema de classificação binária, basta estimar as probabilidades da classe positiva $\hat{p}(y = 1|x)$, pois as probabilidades da classe negativa são o complemento $\hat{p}(y = 0|x) = 1 - \hat{p}(y = 1|x)$. Assim, em um problema de classificação binário, as probabilidades da classe positiva para as instâncias rotuladas podem ser representadas por um vetor $\vec{y}_{n_l}$, que toma seus valores de $[0;1]^{n_l \times 1}$. No restante desta Seção, o 4Pros será desenvolvido para problemas com duas classes. A generalização do 4Pros para o cenário multiclasse ficará como trabalho futuro.

Para problemas binários, a matriz de valores preditivos de classe dos protótipos $\mathbf{A}$ pode ser descrita como um vetor $\vec{a} \in \mathscr{R}^{M \times 1}$ de valores preditivos para a classe positiva. Assim, a probabilidade para a classe positiva, dada a *i*-ésima instância pode ser estimada pela Equação (6.1), onde σ é a função sigmoide. Percebe-se que os valores de $\vec{a}$ indicam a "afinidade" dos protótipos correspondentes com a classe positiva. Quanto maiores esses valores, mais esses protótipos representam a classe positiva e quanto menores esses valores, mais esses protótipos representam a classe negativa. Adicionalmente, valores de $\vec{a}$ muito próximos de 0 significam que os protótipos correspondentes não contribuem para a estimação das probabilidades, indicando a possibilidade de removê-los.

$$\hat{p}(y_i = 1|\vec{z}_i) = \sigma(\vec{z}_i, \vec{a}) = \frac{1}{1 + e^{-\vec{z}_i \vec{a}}}. \tag{6.1}$$

Dada uma função de escore apropriada qualquer *s*, o critério $Q(\vec{a})$, cuja minimização permite a propagação de probabilidades, é dado pela Equação (6.2).

$$Q(\vec{a}) = \arg\min_{\vec{a}} \sum_{i=1}^{n_l} \mathrm{s}(\sigma(\vec{z}_i, \vec{a}), \vec{y}_i) + \frac{\xi}{2} \sum_{i',i''=1}^{N} E_{i'i''} \mathrm{s}(\sigma(\vec{z}_{i'}, \vec{a}), \sigma(\vec{z}_{i''}, \vec{a})). \tag{6.2}$$

A formulação do critério $Q(\vec{a})$ com uma função *s* genérica é simples e poderosa, pois permite que qualquer função de escore apropriada seja usada. Duas dessas funções que são bastante comuns na literatura são o escore de Brier (*Brier score* – BS)[7] e a entropia cruzada (*cross entropy* – CE). Caso o BS seja usado, o critério toma a forma da Equação (6.3). Caso a CE seja usada, o critério toma a forma da Equação (6.4).

$$Q_{BS}(\vec{a}) = \arg\min_{\vec{a}} \sum_{i=1}^{n_l} ||\sigma(\vec{z}_i, \vec{a}) - \vec{y}_i||^2 + \frac{\xi}{2} \sum_{i',i''=1}^{N} E_{i'i''} ||\sigma(\vec{z}_{i'}, \vec{a}) - \sigma(\vec{z}_{i''}, \vec{a})||^2, \tag{6.3}$$

$$Q_{CE}(\vec{a}) = \arg\min_{\vec{a}} \sum_{i=1}^{n_l} -\vec{y}_i \log \sigma(\vec{z}_i, \vec{a}) + \frac{\xi}{2} \sum_{i',i''=1}^{N} -E_{i'i''} \sigma(\vec{z}_{i'}, \vec{a}) \log \sigma(\vec{z}_{i''}, \vec{a}). \tag{6.4}$$

É interessante destacar que essa formulação do 4Pros equivale a uma regressão logística semi-supervisionada regularizada, tendo como entrada a matriz de pesos locais $\mathbf{Z}$ e como pesos estimados o vetor $\vec{a}$. Caso o valor do peso regularizador seja $\xi = 0$, o 4Pros reverte para uma regressão logística supervisionada treinada sobre os pesos locais $\mathbf{Z}$ dos dados rotulados. Esse é um importante resultado, pois conecta o 4Pros a um método muito conhecido e estudado da literatura de Aprendizagem de Máquina. O segundo termo dos critérios promove uma regularização sobre todos os elementos de treinamento, respeitando a segunda condição de propagação (objetos próximos devem ter probabilidades de classe similares).

Esses critérios podem ser minimizados por qualquer otimizador capaz de executar métodos de gradiente descendente. Neste trabalho, a biblioteca Theano [67] da linguagem Python foi escolhida, devido a sua capacidade de derivação automática.

Uma vez que o vetor $\vec{a}$ é obtido, é possível atribuir probabilidades de classe às instâncias não-rotuladas usando a Equação (6.1).

6.4 Inicialização dos protótipos

Antes de construir o grafo usado pelo método 4Pros é preciso obter um conjunto de protótipos não-rotulados. Para isso, é possível empregar qualquer algoritmo de agrupamento. Neste trabalho, o algoritmo escolhido foi o K-médias com uma distância generalizada, pois, como a Seção 3.4 mostrou, o uso de uma distância adaptativa pode melhorar o desempenho dos métodos de propagação em grafo. No restante desta Seção, o algoritmo K-médias adaptativo será chamado de AK-médias.

O AK-médias adiciona um passo de cálculo dos pesos da distância às iterações do K-médias tradicional. O método inicia com um conjunto $\Omega = \{\vec{w}_1, \ldots, \vec{w}_m, \ldots, \vec{w}_M\}$ de protótipos selecionados aleatoriamente. A cada iteração, o método calcula as distâncias das instâncias de treinamento para os protótipos, reposiciona os protótipos para os centros das suas instâncias afetadas e calcula os pesos da distância.

Os pesos da distância generalizada da_m usada pelo AK-médias nesta Seção são similares aos pesos λ da distância $\mathrm{d}\pi_{m,l}$ do IVABC (Seção 5), porém é preciso adaptá-los, devido à ausência de rótulos na etapa de seleção de protótipos. Assim, essa distância tem dois níveis de adaptabilidade, pois ela modela as dispersões das variáveis nas sub-regiões dos dados representadas pelos protótipos e a importância de cada sub-região no conjunto de dados. Além disso, a distância da_m é aplicada a dados do tipo ponto, como mostra a Equação (6.5).

$$\mathrm{da}_m(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m) = \sum_{j=1}^{p} \lambda_{m,j}(x_{i,j} - w_{m,j})^2, \tag{6.5}$$

onde o peso $\lambda_{m,j}$ representa a dispersão da j-ésima variável do m-ésimo protótipo e é calculado de acordo com a Equação (6.6), com $\lambda_{m,j} > 0$ e $\prod_{j=1}^{p} \lambda_{m,j} = \gamma_m$.

$$\lambda_{m,j} = \frac{\left(\gamma_m \prod_{h=1}^{p} \Delta_{m,h}\right)^{\frac{1}{p}}}{\Delta_{m,j}}, \tag{6.6}$$

onde $\Delta_{m,j}$ é dado pela Equação (6.7) e γ_m é o peso do m-ésimo protótipo no conjunto de treinamento e é atualizado usando a Equação (6.8).

$$\Delta_{m,j} = \frac{\sum_{i=1}^{N} \left(x_{i,j} - w_{m,j}\right)^2 \delta(m,k,i)}{N_{m,k}}, \tag{6.7}$$

onde $N_{m,k}$ é o número de instâncias das quais o m-ésimo protótipo é um dos k vizinhos mais próximos e $\delta(m,k,i) = 1$, se o m-ésimo protótipo for um dos k vizinhos mais próximos da i-ésima instância, caso contrário $\delta(m,k,i) = 0$.

$$\gamma_m = \frac{\left[\prod_{r=1}^{M} \sum_{i \in r} \mathrm{da}_r(\vec{x}_i, \vec{w}_r)\delta(r,k,i)N_{r,k}^{-1}\right]^{\frac{1}{M}}}{\sum_{i \in m} \mathrm{da}_m(\vec{x}_i, \vec{w}_m)\delta(m,k,i)N_{m,k}^{-1}}, \tag{6.8}$$

onde M é o número de protótipos. A Equação (6.8) resulta em erro se $\sum_{i \in m} \mathrm{d}_m(\vec{x}_i, \vec{w}_m) = 0$, ou seja, se o protótipo não afetar instância de treinamento alguma. Para evitar esse problema, todos os protótipos que cumprem essa condição são removidos do conjunto de protótipos. O Apêndice B apresenta a derivação completa dos pesos da distância da_m, segundo o método dos multiplicadores de Lagrange.

No passo de reposicionamento dos protótipos, o m-ésimo protótipo tem a sua posição alterada para o centro das instâncias das quais ele foi um dos k vizinhos mais próximos, como mostra a Equação (6.9).

$$\vec{w}_m = \frac{\sum_{i=1}^{N} \vec{x}_i \delta(m,k,i)}{N_{m,k}} \tag{6.9}$$

O Algoritmo 7 detalha os passos do método AK-médias, considerando as equações de pesos apresentadas nesta Seção.

Algoritmo 7 Algoritmo AK-médias

Requer: Conjunto de treinamento $\Im = \{\vec{x}_1, \ldots, \vec{x}_i, \ldots, \vec{x}_N\}$; número de protótipos M; número de vizinhos k; número máximo de iterações t_{max};

1: $t \leftarrow 0$;
2: Selecione um conjunto inicial de protótipos $\Omega(0) = \{\vec{w}_1(0), \ldots, \vec{w}_m(0), \ldots, \vec{w}_M(0)\}$;
3: Inicialize os pesos $\forall_m \forall_j \lambda_{m,j} = 1$;
4: **Faça**
5: $t \leftarrow t + 1$;
6: Calcule as distâncias entre as instâncias e os protótipos usando a Equação (6.5);
7: Defina os k vizinhos mais próximos de cada instância;
8: Reposicione os protótipos usando a Equação (6.9), obtendo $\Omega(t)$;
9: Calcule as distâncias entre as instâncias e os protótipos usando a Equação (6.5);
10: Defina os k vizinhos mais próximos de cada instância;
11: Calcule os pesos usando as Equações (6.6) e (6.8);
12: **Enquanto** $t < t_{max}$ **e** $\Omega(t) != \Omega(t-1)$

Retorne Protótipos $\Omega(t)$ e seus respectivos pesos λ.

6.5 Construção do grafo

Depois de obter os protótipos e seus pesos λ usando o AK-médias, é necessário construir o grafo usado que representa a estrutura intrínseca dos dados. Neste trabalho, o grafo kNN será usado. Para construí-lo, primeiro, é preciso obter uma matriz $\mathbf{R} \in \mathscr{R}^{N \times M}$ de distâncias entre as N instâncias e os M protótipos que toma a seguinte forma:

$$\mathbf{R} = \begin{bmatrix} \mathrm{da}_m(\vec{x}_1, \vec{w}_1, \vec{\lambda}_1)\delta(1,k,1) & \mathrm{da}_m(\vec{x}_1, \vec{w}_2, \vec{\lambda}_2)\delta(2,k,1) & \ldots & \mathrm{da}_m(\vec{x}_1, \vec{w}_M, \vec{\lambda}_M)\delta(M,k,1) \\ \mathrm{da}_m(\vec{x}_2, \vec{w}_1, \vec{\lambda}_1)\delta(1,k,2) & \mathrm{da}_m(\vec{x}_2, \vec{w}_2, \vec{\lambda}_2)\delta(2,k,2) & \ldots & \mathrm{da}_m(\vec{x}_2, \vec{w}_M, \vec{\lambda}_M)\delta(M,k,2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{da}_m(\vec{x}_N, \vec{w}_1, \vec{\lambda}_1)\delta(1,k,N) & \mathrm{da}_m(\vec{x}_N, \vec{w}_2, \vec{\lambda}_2)\delta(2,k,N) & \ldots & \mathrm{da}_m(\vec{x}_N, \vec{w}_M, \vec{\lambda}_M)\delta(M,k,N) \end{bmatrix}$$

Ou seja, a posição $R_{im} = 0$ se o m-ésimo protótipo não for um dos k vizinhos mais próximos da i-ésima instância, caso contrário, $R_{im} = \mathrm{da}_m(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m)$. A partir da matriz de distâncias, é possível obter a matriz de pesos locais $\mathbf{Z}$ da seguinte forma:

$$\mathbf{Z} = \begin{bmatrix} \frac{(R_{11}+1)^{-1}}{\sum_{m=1}^{M}(R_{1m}+1)^{-1}} & \frac{(R_{12}+1)^{-1}}{\sum_{m=1}^{M}(R_{1m}+1)^{-1}} & \cdots & \frac{(R_{1M}+1)^{-1}}{\sum_{m=1}^{M}(R_{1m}+1)^{-1}} \\ \frac{(R_{21}+1)^{-1}}{\sum_{m=1}^{M}(R_{2m}+1)^{-1}} & \frac{(R_{22}+1)^{-1}}{\sum_{m=1}^{M}(R_{2m}+1)^{-1}} & \cdots & \frac{(R_{2M}+1)^{-1}}{\sum_{m=1}^{M}(R_{2m}+1)^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{(R_{N1}+1)^{-1}}{\sum_{m=1}^{M}(R_{Nm}+1)^{-1}} & \frac{(R_{N2}+1)^{-1}}{\sum_{m=1}^{M}(R_{Nm}+1)^{-1}} & \cdots & \frac{(R_{NM}+1)^{-1}}{\sum_{m=1}^{M}(R_{Nm}+1)^{-1}} \end{bmatrix}$$

Assim, quanto menor a distância entre o m-ésimo protótipo e a i-ésima instância, maior o peso local Z_{im}. Por fim, a matriz de adjacências $\mathbf{E}$ é obtida usando a Equação (6.10).

$$\mathbf{E} = \mathbf{Z}\mathbf{Z}^T. \tag{6.10}$$

Na matriz de adjacências $\mathbf{E}$, duas instâncias de treinamento possuem um valor alto de adjacência se compartilharem muitos protótipos similares.

Embora outras formas de construção de grafo possam ser usadas, como os grafos do AGR e do EAGR, o grafo kNN foi escolhido por simplificar o processo de predição de probabilidades de novas instâncias de teste, pois os grafos do AGR e do EAGR são construídos pela solução de um problema de otimização e os mesmos passos devem ser seguidos para novas instâncias. Por outro lado, com o grafo kNN usado pelo 4Pros, o vetor de pesos locais $\vec{z}$ de uma nova instância é obtido diretamente a partir das distâncias entre a instância e os protótipos.

6.6 Experimentos com o 4Pros

Esta Seção apresenta uma análise de desempenho do método 4Pros, em sua versão 4Pros_{BS}, que minimiza a Equação (6.3). O 4Pros_{BS} será comparado a um algoritmo de propagação de rótulos em grafo de protótipos recentemente introduzido, chamado EAGR [74].

6.6.1 Conjuntos de dados

Dez conjuntos de dados do repositório online UCI [41] serão considerados neste estudo. Quando os conjuntos estavam disponíveis no repositório de conjuntos `mldata.org`, a biblioteca *Python scikit-learn* [54] foi usada para obtê-los diretamente. Caso contrário, os dados foram retirados da página do UCI. A Tabela 6.1 descreve os conjuntos de dados usados nesta

Nome	Instâncias	Atributos	Classes	Proporção
balance-scale	625	4	3	46,08%
cleveland	297	13	5	53,87%
hepatitis	155	19	2	79,35%
horse	300	27	2	36,33%
ionosphere	351	34	2	64,10%
iris	150	4	3	33,33%
vehicle	846	18	4	25,77%
vowel	990	10	11	9,09%
wdbc	569	30	2	37,26%
wpbc	194	33	2	23,71%

Tabela 6.1: **Conjuntos de dados** - Descrição dos dez conjuntos de dados do repositório UCI usados nos experimentos. A coluna da direita apresenta a proporção de instâncias positivas. Os conjuntos multi-classe foram binarizados tomando a classe majoritária como classe positiva e formando a classe negativa com as instâncias das demais classes

análise. Como o 4Pros foi desenvolvido para problemas binários, os conjuntos com mais de duas classes são transformados em conjuntos binários considerando a classe majoritária como classe positiva e agrupando todas as outras classes como classe negativa. A coluna da direita da Tabela 6.1 apresenta as proporções de instâncias positivas em cada conjunto.

6.6.2 Metodologia

Todos os métodos foram implementados usando a linguagem de programação Python. A biblioteca Theano [67] foi adotada para realizar a minimização do critério de propagação para o 4Pros_{BS}. Para isso, foi necessário definir dois parâmetros extras: o número máximo de iterações t_{max} e a taxa de aprendizado α, que decresce monotonicamente a cada iteração t, como mostra a Equação (6.11), onde $\alpha(0)$ é o valor inicial da taxa de aprendizado.

$$\alpha(t) = \alpha(0)\frac{t_{max} - t}{t_{max} - 1}. \tag{6.11}$$

Os experimentos foram executados em dez repetições de validações cruzadas de cinco partições (quatro partições para treinamento e uma para teste), totalizando cinquenta execuções para cada método e cada conjunto de dados. Para selecionar os parâmetros, foi adotado o seguinte procedimento:

1. o conjunto de dados é dividido em cinco partições, preservando as proporções das classes;

2. quatro dessas partições são usadas para construir o conjunto que será usado para seleção

de parâmetros (conjunto de ajuste);

3. o conjunto de ajuste é dividido em três partições:

   (a) duas partições são usadas para treinar o método com uma certa configuração dos parâmetros, a outra é usada para testar o modelo obtido;

   (b) o resultado segundo alguma métrica (neste caso, entropia cruzada) é comparado com o melhor resultado anterior e, se for melhor, é armazenado.

Para os três métodos o melhor número de protótipos M foi buscado no intervalo $[2, N/10]$, onde N é o tamanho do conjunto de ajuste. O melhor valor do número de vizinhos k foi obtido do intervalo $[1, k_{max}]$, onde $k_{max} = M/2$, se $M \leq 30$, senão $k_{max} = 15$. O outro parâmetro que os dois métodos compartilham é o peso do critério de propagação ξ (chamado γ no artigo que propôs o EAGR, mas aqui renomeado), cujos valores foram obtidos do intervalo $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100]$. Os demais parâmetros foram buscados nos seguintes intervalos:

1. EAGR:

   (a) parâmetros λ e σ para construção do grafo: $[1, 100]$;

2. 4Pros_{BS} (parâmetros para a otimização do critério):

   (a) taxa de aprendizado α: $[0.1, 1]$;

   (b) número máximo de iterações t_{max}: $[5, 75]$.

As Tabelas 6.2 e 6.3 apresentam os parâmetros selecionados para o EAGR e o 4Pros_{BS}, respectivamente.

conjunto	M	k	ξ	λ	σ
balance-scale	9	3	1	100	1
cleveland	15	4	10	10	1
hepatitis	8	1	1	100	100
horse	4	2	1	100	1
ionosphere	18	1	100	100	1
iris	9	1	100	1	1
vehicle	56	5	1	1	10
vowel	77	5	1	1	10
wdbc	30	1	100	100	1
wpbc	14	4	100	10	1

Tabela 6.2: **EAGR** - parâmetros selecionados

conjunto	M	k	ξ	$\alpha(0)$	t_{max}
balance-scale	15	7	10^{-3}	1	75
cleveland	5	1	10	0,775	75
hepatitis	6	2	10^{-1}	1	10
horse	20	2	10^{-3}	0,55	25
ionosphere	15	7	10^{-3}	1	75
iris	9	2	10^{-3}	0,775	75
vehicle	64	4	1	0,775	75
vowel	55	2	10^{-1}	1	75
wdbc	12	1	10	1	25
wpbc	12	3	10^{-1}	0,325	75

Tabela 6.3: **4Pros$_{\mathbf{BS}}$** - parâmetros selecionados

O método EAGR não fornece probabilidades de classe de forma nativa. Para extrair essas probabilidades com o objetivo de comparar os métodos, a matriz $\mathbf{A}_{EAGR}$ de predições de rótulos dos protótipos precisa ser normalizada, tal que $\sum \vec{a}_m = 1$ e $\vec{a}_m \succcurlyeq 0$, $\forall_m \in \{1,\ldots,M\}$.

6.6.3 Resultados

As Tabelas 6.4, 6.5 e 6.6 apresentam os resultados dos experimentos em termos de entropia cruzada, escore de Brier e taxa de acerto, respectivamente. Valores em negrito indicam resultados com significância de acordo com o teste dos postos sinalizados de Wilcoxon, com nível de significância $\alpha = 0,05$.

Como esperado, em termos de entropia cruzada e escore de brier, o 4Pros, em sua versão 4Pros_{BS}, tem desempenho médio superior ao EAGR, com valores significativamente melhores em 9 de 10 conjuntos. O EAGR apresentou médias de escore de Brier e entropia cruzada em apenas um conjunto (*balance-scale*), mas o resultado não foi significativo. O EAGR apresenta valores muito próximos de entropia cruzada e escore de Brier em todos os conjuntos. Isso ocorre porque ele tem uma forte tendência de estimar probabilidades muito concentradas em torno de $0,5$, como mostra a Seção 2.3.2. Um fato interessante é que o 4Pros_{BS} e o EAGR apresentaram desempenhos similares em termos de taxa de acerto, o que surpreende, pois o EAGR é treinado especificamente para otimizar essa métrica. As predições de rótulos do 4Pros_{BS} foram feitas considerando como positivas as instâncias com probabilidade de classe positiva maior ou igual a $0,5$, mas outro limiar poderia ser escolhido, otimizando a taxa de acerto. Os resultados de entropia cruzada e escore de Brier confirmam que o método proposto 4Pros atinge seu objetivo de estimar probabilidades apropriadas de classe num cenário semi-supervisionado.

conjunto	4Pros_{BS}	EAGR
balance-scale	$0,81805$	$0,69235$
	$(0,62191)$	$(0,00015)$
cleveland	$\mathbf{0,52030}$	$0,69181$
	$(\mathbf{0,07922})$	$(0,00026)$
hepatitis	$\mathbf{0,44924}$	$0,69102$
	$(\mathbf{0,08506})$	$(0,00128)$
horse	$\mathbf{0,62805}$	$0,69234$
	$(\mathbf{0,05643})$	$(0,00040)$
ionosphere	$\mathbf{0,43905}$	$0,69157$
	$(\mathbf{0,07825})$	$(0,00026)$
iris	$\mathbf{0,07432}$	$0,68629$
	$(\mathbf{0,00927})$	$(0,00041)$
vehicle	$\mathbf{0,38026}$	$0,69247$
	$(\mathbf{0,02624})$	$(0,00004)$
vowel	$\mathbf{0,19926}$	$0,69270$
	$(\mathbf{0,02238})$	$(0,00004)$
wdbc	$\mathbf{0,23606}$	$0,69167$
	$(\mathbf{0,06094})$	$(0,00011)$
wpbc	$\mathbf{0,66234}$	$0,69299$
	$(\mathbf{0,08456})$	$(0,00029)$

Tabela 6.4: **Entropia cruzada** - médias e desvios padrões entre parênteses; valores em negrito indicam significância estatística ao nível de $\alpha = 0,05$

conjunto	4Pros_{BS}	EAGR
balance-scale	$0,51621$	$0,49920$
	$(0,07791)$	$(0,00015)$
cleveland	$\mathbf{0,32730}$	$0,49866$
	$(\mathbf{0,05411})$	$(0,00026)$
hepatitis	$\mathbf{0,28438}$	$0,49787$
	$(\mathbf{0,06396})$	$(0,00128)$
horse	$\mathbf{0,43514}$	$0,49920$
	$(\mathbf{0,05010})$	$(0,00040)$
ionosphere	$\mathbf{0,26537}$	$0,49842$
	$(\mathbf{0,05528})$	$(0,00026)$
iris	$\mathbf{0,01207}$	$0,49314$
	$(\mathbf{0,00384})$	$(0,00040)$
vehicle	$\mathbf{0,23574}$	$0,49932$
	$(\mathbf{0,02087})$	$(0,00004)$
vowel	$\mathbf{0,10180}$	$0,49956$
	$(\mathbf{0,01681})$	$(0,00004)$
wdbc	$\mathbf{0,13346}$	$0,49852$
	$(\mathbf{0,04196})$	$(0,00011)$
wpbc	$\mathbf{0,46301}$	$0,49984$
	$(\mathbf{0,07179})$	$(0,00029)$

Tabela 6.5: **Escore de Brier** - médias e desvios padrões entre parênteses; valores em negrito indicam significância estatística ao nível de $\alpha = 0,05$

dataset	$4Pros_{BS}$	EAGR
balance-scale	53,367	**81,596**
	(3,078)	(**5**,**111**)
cleveland	78,146	78,670
	(5,042)	(6,103)
hepatitis	**80**,**699**	75,209
	(**6**,**365**)	(12,182)
horse	66,863	**71**,**845**
	(7,062)	(**8**,**930**)
ionosphere	84,107	82,386
	(5,107)	(6,127)
iris	100,000	99,800
	(0,00000)	(0,800)
vehicle	83,988	**88**,**247**
	(2,246)	(**3**,**072**)
vowel	**94**,**384**	80,152
	(**1**,**424**)	(2,146)
wdbc	91,005	92,075
	(3,283)	(3,134)
wpbc	**63**,**903**	55,957
	(**9**,**489**)	(8,986)

Tabela 6.6: **Taxa de acerto percentual** - médias e desvios padrões entre parênteses; valores em negrito indicam significância estatística ao nível de $\alpha = 0,05$

6.7 Considerações finais

Este Capítulo introduziu um algoritmo de propagação de probabilidades em grafos de protótipos (4Pros). Os métodos similares da literatura tem o objetivo de propagar rótulos. Para transformar a propagação de rótulos em propagação de probabilidades, o 4Pros usa um critério de propagação que incorpora funções de escore apropriadas.

A importância do cálculo dos pesos locais para a construção do grafo de protótipos também foi considerada neste Capítulo. Assim, o 4Pros usa uma distância adaptativa similar à distância usada pelo método IVABC (Seção 5), porém adaptada para dados do tipo ponto e sem

informação de rótulos.

A análise de desempenho contida neste Capítulo usou dez conjuntos de dados do repositório online UCI. O 4Pros, em sua versão 4Pros_{BS}, que otimiza o escore de Brier (BS), foi comparado ao EAGR. Os resultados mostraram desempenho significativamente superior do 4Pros_{BS} em termos de entropia cruzada e escore de Brier em 9 dos 10 conjuntos. Além disso, os dois métodos tiveram desempenhos similares em termos de taxa de acerto, apesar de o EAGR ser treinado especificamente para otimizar essa métrica. Os experimentos mostraram que o 4Pros pode ser considerado um bom método para estimação de probabilidades num cenário semi-supervisionado.

O 4Pros tem custo computacional de treinamento maior do que o EAGR, cujo critério de propagação é minimizado de forma analítica. No entanto, quando o modelo é usado para predizer probabilidades para novas instâncias, os pesos locais dos protótipos do 4Pros são diretamente calculados usando as distâncias, enquanto o EAGR precisa solucionar um problema de otimização para obtê-los.

7 CONCLUSÃO

O conhecimento coroa os esforços com o êxito.

—TEXTOS BUDISTAS

7.1 Considerações finais

Este trabalho foi desenvolvido com foco nas tarefas de classificação e estimação de probabilidades de classe, mais especificamente nos métodos baseados em protótipos, devido a sua versatilidade, facilidade de implementação e interpretabilidade, e nas distâncias adaptativas, capazes de modelar as dispersões das variáveis dos conjuntos de dados.

Alguns pontos importantes de melhoria dos métodos baseados em protótipos existentes na literatura foram expostos e considerados no desenvolvimento deste trabalho, são eles: (i) a suscetibilidade a mínimos locais, devido à má inicialização dos protótipos; (ii) o problema das distâncias adaptativas existentes (como a usada pelo método WFLVQ [63]), que sofrem de perda de informação e de ambiguidade na interpretação dos seus pesos; (iii) a inexistência de algoritmos semi-supervisionados de propagação de probabilidades de classe apropriadas através de grafos de protótipos na literatura.

Dois métodos baseados em protótipos e distâncias adaptativas foram introduzidos neste trabalho para lidar com esses quatro pontos de melhoria. O IVABC foi desenvolvido para realizar classificação multi-classe de dados intervalares e o 4Pros para a tarefa semi-supervisionada de estimação de probabilidades de classe.

O IVABC usa um algoritmo de otimização de enxames [31] de simples implementação para atacar o problema da má inicialização dos protótipos que leva a mínimos locais. Além disso, a distância adaptativa intervalar adotada pelo IVABC é capaz de modelar classes desbalanceadas e compostas de sub-regiões de variados tamanhos e formas, além de resolver a perda de informação no cálculo dos pesos e de ter pesos cujos valores são facilmente interpretáveis. Por fim, o IVABC tem a capacidade de eliminar atributos irrelevantes e protótipos inativos.

O 4Pros, por outro lado, foi desenvolvido para suprir a ausência de algoritmos semi-supervi-

sionados de propagação de probabilidades através de grafos de protótipos. Para isso, o método adota uma versão modificada do critério de propagação, para garantir que a solução encontrada respeite duas condições fundamentais, i.e., as probabilidades de classe das instâncias rotuladas devem ser mantidas ou pouco alteradas e instâncias similares devem ter probabilidades de classe similares.

O 4Pros é flexível quanto ao método de seleção dos protótipos e à construção do grafo, no entanto, neste trabalho, uma distância adaptativa foi escolhida, cujos pesos são calculados sem informação de rótulos de classe (durante a seleção não-supervisionada dos protótipos).

Por fim, este trabalho também apresentou estudos de desempenho para ambos os métodos. Tanto o IVABC quanto o 4Pros foram comparados a algoritmos existentes apropriados, usando conjuntos de dados intervalares sintéticos e reais para o IVABC e conjuntos de dados reais do tipo ponto para o 4Pros.

O IVABC foi comparado ao WFLVQ [63] e a três classificadores baseados em protótipos treinados por diferentes algoritmos de enxames [31, 37, 79], usando a codificação de partículas explicada na Seção 4.3. Os cinco métodos tiveram suas taxas de erro de classificação avaliadas usando quatro conjuntos de dados sintéticos e quatro de dados reais. O IVABC obteve as melhores taxas médias de erro em cinco dos oito conjuntos de dados, sendo estatisticamente equivalente aos melhores algoritmos nos outros três conjuntos. Além disso, o IVABC também apresentou os menores intervalos de confiança para cinco conjuntos de dados, demonstrando robustez. O IVABC também selecionou atributos automaticamente e de forma significativa e eliminou menos protótipos do que o WFLVQ, o que mostra que o IVABC mantém menos protótipos inativos.

Uma versão do 4Pros, 4Pros_{BS}, foi comparada a um algoritmo de propagação de rótulos em grafo de protótipos chamado EAGR, demonstrando cumprir o objetivo de propagar probabilidades de classe apropriadas, apresentando desempenho superior ao EAGR em termos de entropia cruzada e escore de Brier, além de apresentar desempenho similar em termos de taxa de acerto.

7.2 Publicações

No decorrer do desenvolvimento deste trabalho, os seguintes artigos foram publicados em revistas ou conferências internacionais:

- **Artigos diretamente relacionados à tese:**

1. **Silva Filho, T. M., Souza, R. M. C. R., e Prudêncio, R. B. C. (2016), A swarm-trained k-nearest prototypes adaptive classifier with automatic feature selection for interval data, Neural Networks, Volume 80, 19–33:**
   Trata do método IVABC, proposto na Seção 5, que foi desenvolvido com o objetivo de contribuir com uma solução para a tarefa de classificação de dados intervalares, apresentando características que permitem que o método lide com várias das limitações dos métodos baseados em protótipos. O artigo também traz uma análise de desempenho, comparando o método a outros algoritmos existentes na literatura, usando conjuntos de dados intervalares sintéticos e reais.
2. **Silva Filho, T. M., Pimentel, B. A., Souza, R. M. C. R., e Oliveira, A. L. I. (2015), Hybrid methods for fuzzy clustering based on fuzzy c-means and improved particle swarm optimization, Expert Systems with Applications, Volume 42(17–18), 6315–6328:**
   Relacionado ao primeiro artigo, embora seu foco não seja a tarefa de classificação, pois nele foram introduzidos dois métodos de agrupamento difuso que, assim como o IVABC, usam um algoritmo de otimização de enxames para treinar os protótipos (centróides) e evitar mínimos locais.

- **Outros artigos:**

1. **Souza, M. P. S, Silva Filho, T. M., Amaral, G. J. A., e Souza, R. M. C. R. (Aceito para publicação), Investigating Different Fitness Criteria for Swarm-based Clustering, International Journal of Business Intelligence and Data Mining:**
   Avaliou experimentalmente diversos critérios para treinamento de modelos de agrupamento usando métodos baseados em enxames.
2. **Souza, L. C., Souza, R. M. C. R., Amaral, G. J. A., e Silva Filho, T. M. (2017), A Parametrized Approach for Linear Regression of Interval Data, Knowledge-based Systems, Volume 131, 149–159:**
   Propôs um novo método para regressão linear de dados intervalares. Todas as abordagens existentes de regressão linear intervalar usam certos pontos de referência fixos para modelar os intervalos, tais como mínimos e máximos e centros e amplitudes. Isso é uma limitação, pois conjuntos diferentes podem ser modelados por pontos de referência diferentes. O novo método modela intervalos usando a equação da reta e encontra os melhores pontos de referência para as variáveis regressoras

automaticamente. O método constrói duas regressões: uma para os limites inferiores e outra para os limites superiores da variável resposta. Além disso, o novo algoritmo garante a coerência matemática da resposta (todos os mínimos estimados são menores do que os máximos estimados) através de uma transformação Box-Cox da variável resposta.

3. **Kull, M., Silva Filho, T. M., e Flach, P. (2017), Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers, 20th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, Volume 54, 623–631:**
   Introduziu um novo método paramétrico para calibração de classificadores binários, chamado *beta calibration*, que é mais versátil do que o método bastante conhecido de calibração logística, sendo capaz de encontrar curvas de calibração mais variadas, incluindo sigmóides invertidas. Ao contrário da calibração logística, a nova família de curvas de calibração também inclui a identidade, o que significa que *beta calibration* não piora a calibração do classificador original. Além disso, por ser um método paramétrico, *beta calibration* é menos suscetível a sobreajuste em conjuntos de dados pequenos do que regressão isotônica, outro método de calibração bem conhecido.

4. **Perello-Nieto, M., Silva Filho, T. M., Kull, M., e Flach, P. (2016), Background Check: A General Technique to Build More Reliable and Versatile Classifiers, IEEE 16th International Conference on Data Mining (ICDM), 1143–1148, Barcelona:**
   Introduziu uma poderosa técnica para fazer classificadores mais confiáveis e versáteis. *Background check* equipa classificadores com a habilidade de avaliar a diferença entre dados de teste e dados de treinamento. Com isso, o classificador passa a ser capaz de realizar classificação com opção de rejeição, identificar *outliers* e avaliar a confiança em suas predições.

5. **Araújo, M. C., Souza, R. M. C. R., Lima, R. C., e Silva Filho, T. M. (2016), An interval prototype classifier based on a parameterized distance applied to breast thermographic images, Medical & Biological Engineering & Computing, Volume 55 (6), 1–12:**
   Propôs uma nova abordagem para classificar anomalias mamárias (malignas, benignas e cistos) a partir de imagens termográficas, modeladas como variáveis intervalares. O algoritmo mapeia os intervalos para um novo espaço onde as classes

são mais facilmente separadas e usa uma distância Mahalanobis parametrizada para alocar rótulos. O método foi aplicado a uma base brasileira de termografias de mamas de 50 pacientes, obtendo excelentes resultados.

7.3 Trabalhos futuros

Para avançar os estudos sobre métodos baseados em protótipos e distâncias adaptativas, as seguintes atividades descritas a seguir precisam ser desenvolvidas.

1. **Estender o IVABC para outros tipos de dados:** neste trabalho, o IVABC foi aplicado a dados intervalares, mas é possível aplicá-lo a outros tipos de dados, como histogramas, dados modais, dados do tipo ponto, entre outros, desde que a codificação das partículas e a distância sejam adaptadas.

2. **Avaliar o uso de outros métodos populacionais pra treinar os protótipos do ISOPC:** apenas o VABC foi considerado para treinar os protótipos neste trabalho, resultando no método IVABC. Seria interessante verificar o impacto de desempenho resultante do uso de outros métodos populacionais.

3. **Desenvolver a versão multiclasse do 4Pros:** a versão binária do 4Pros foi formulada como uma regressão logística semi-supervisionada. De forma similar, a versão multiclasse do 4Pros pode ser formulada como uma regressão multinomial.

4. **Análise mais aprofundada sobre o desempenho do método 4Pros:** este trabalho apresentou experimentos detalhados apenas para o método IVABC, portanto é necessário analisar melhor o 4Pros, comparando-o a mais algoritmos da literatura e usando mais conjuntos de dados.

5. **Investigar a utilidade do 4Pros para calibragem semi-supervisionada de probabilidades:** quando um certo modelo é construído para estimar probabilidades de classe, é comum que essas probabilidades estimadas não condigam com as reais probabilidades observadas, ou seja, o modelo pode ser superconfiante ou subconfiante. Para corrigir essas distorções, é possível utilizar métodos de calibragem de probabilidades, como Platt scaling [56] ou regressão isotônica [78]. No entanto, esses métodos de calibragem de probabilidades são supervisionados. Portanto, é interessante investigar a possibilidade de usar o método 4Pros se os dados disponíveis para a calibragem forem apenas parcialmente rotulados.

Referências

[1] Mohamed Aly. Survey on Multi-Class Classification Methods, 2005. 21

[2] Annalisa Appice, Claudia D'Amato, Floriana Esposito, e Donato Malerba. Classification of Symbolic Objects: A lazy Learning Approach. *Intelligent Data Analysis*, 10(4):301–324, December 2006. ISSN 1088-467X. 64

[3] Andreas Backhaus e Udo Seiffert. Classification in high-dimensional spectral data: Accuracy vs. interpretability vs. model size. *Neurocomput.*, 131:15–22, May 2014. ISSN 0925-2312. doi: 10.1016/j.neucom.2013.09.048. URL http://dx.doi.org/10.1016/j.neucom.2013.09.048. 21, 28, 29

[4] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713. 27

[5] Avrim Blum e Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, páginas 92–100, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0. doi: 10.1145/279943.279962. URL http://doi.acm.org/10.1145/279943.279962. 23, 30

[6] Thorsten Bojer, Barbara Hammer, Daniel Schunk, e Katharina Tluk von Toschanowitz. Relevance determination in learning vector quantization. In *PROC. OF EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS*, 2001. 38

[7] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950. doi: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2. URL http://dx.doi.org/10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2. 92

[8] Oscar Bustos e Alejandro C. Frery. *Simulação estocástica: teoria e algoritmos*. Monografias de matemática. Conselho Nacional de Desenvolvimento Científico e Tecno-

lógico, Instituto de Matemática Pura e Aplicada, 1992. ISBN 9788524400537. URL https://books.google.com.br/books?id=VupLrgEACAAJ. 67

[9] Deng Cai e Xinlei Chen. Large scale spectral clustering via landmark-based sparse representation. *IEEE Transactions on Cybernetics*, 45(8):1669–1680, Agosto 2015. ISSN 2168-2267. doi: 10.1109/TCYB.2014.2358564. 23, 24, 31

[10] Ricardo J. G. B. Campello, Davoud Moulavi, e Joerg Sander. *Density-Based Clustering Based on Hierarchical Density Estimates*, páginas 160–172. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-37456-2. doi: 10.1007/978-3-642-37456-2_14. URL http://dx.doi.org/10.1007/978-3-642-37456-2_14. 27

[11] Vittorio Castelli e Thomas M. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16(1):105 – 111, 1995. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/0167-8655(94)00074-D. URL http://www.sciencedirect.com/science/article/pii/016786559400074D. 23, 30

[12] Alejandro Cervantes, Inés M. Galvan, e Pedro Isasi. Ampso: A new particle swarm method for nearest neighborhood classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(5):1082–1091, Oct 2009. ISSN 1083-4419. doi: 10.1109/TSMCB.2008.2011816. 46, 47, 51

[13] Corinna Cortes e Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995. ISSN 1573-0565. doi: 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411. 28

[14] Francisco A.T. de Carvalho, Camilo P. Tenório, e Nicomedes L. Cavalcanti Junior. Partitional fuzzy clustering methods based on adaptive quadratic distances. *Fuzzy Sets and Systems*, 157(21):2833 – 2857, 2006. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/j.fss.2006.06.004. URL http://www.sciencedirect.com/science/article/pii/S0165011406002557. 23, 38

[15] Ivanoe De Falco, Antonio Della Cioppa, e Ernesto Tarantino. Facing classification problems with particle swarm optimization. *Applied Soft Computing*, 7(3):652 – 658, 2007. ISSN 1568-4946. doi: http://dx.doi.org/10.1016/j.asoc.2005.09.004. 22, 29, 46, 47, 48, 51, 67, 88, 89

[16] Edwin Diday e Lynne Billard. *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Wiley Interscience, 2006. 55, 58

[17] Edwin Diday e Gerard Govaert. Classification avec Distances Adaptatives. In *Compte redu de l'Académie des Sciences*, volume tome 278 of *série A*, página p. 993. Paris, 1974. 23, 120, 122, 123, 125, 127

[18] Edwin Diday e Gerard Govaert. Classification automatique avec distances adaptatives. *R.A.I.R.O. Informatique Computer Science*, 11(4):329 – 349, 1977. 23, 38

[19] Edwin Diday e Monique Noirhomme-Fraiture. *Symbolic Data Analysis and the SODAS Software*. Wiley Interscience, 2008. 55

[20] Selim Dilmac e Mehmet Korurek. A new ecg arrhythmia clustering method based on modified artificial bee colony algorithm, comparison with ga and pso classifiers. In *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on*, páginas 1–5, June 2013. doi: 10.1109/INISTA.2013.6577616. 46, 47, 51

[21] Russell Eberhart e James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, páginas 39–43, Oct 1995. doi: 10.1109/MHS.1995.494215. 22, 52, 54

[22] Encyclopædia Britannica. Köppen Climate Classification, janeiro 2017. URL `http://goo.gl/G4Uep`. 82

[23] Telmo M. Silva Filho, Bruno A. Pimentel, Renata M.C.R. Souza, e Adriano L.I. Oliveira. Hybrid methods for fuzzy clustering based on fuzzy c-means and improved particle swarm optimization. *Expert Systems with Applications*, 42(17–18):6315 – 6328, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2015.04.032. URL `//www.sciencedirect.com/science/article/pii/S0957417415002687`. 27

[24] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012. ISBN 9781107096394. URL `https://books.google.com.br/books?id=Ofp4h_oXsZ4C`. 20

[25] Pedram Ghamisi, Micael S. Couceiro, e Jon A. Benediktsson. A novel feature selection approach based on fodpso and svm. *Geoscience and Remote Sensing, IEEE Transactions on*, 53(5):2935–2947, May 2015. ISSN 0196-2892. doi: 10.1109/TGRS.2014.2367010. 52

[26] Pedrom Ghamisi e Jon A. Benediktsson. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *Geoscience and Remote Sensing Letters,*

IEEE, 12(2):309–313, Feb 2015. ISSN 1545-598X. doi: 10.1109/LGRS.2014.2337320. 52

[27] Gerard Govaert. *Classification Automatique et Distances Adaptatives*. Thése de 3éme cycle, Université Paris VI, 1975. 23, 120, 122, 123, 125, 127

[28] Matthieu Guillaumin, Jakob Verbeek, e Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, páginas 902–909. IEEE Computer Society, 2010. 23, 30

[29] Sonal Gupta, Joohyun Kim, Kristen Grauman, e Raymond Mooney. *Watch, Listen & Learn: Co-training on Captioned Images and Videos*, páginas 457–472. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-87479-9. doi: 10.1007/978-3-540-87479-9_48. URL http://dx.doi.org/10.1007/978-3-540-87479-9_48. 23, 30

[30] Lei Huang, Xianglong Liu, Binqiang Ma, e Bo Lang. Online semi-supervised annotation via proxy-based local consistency propagation. *Neurocomputing*, 149, Part C: 1573 – 1586, 2015. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2014.08.035. URL http://www.sciencedirect.com/science/article/pii/S0925231214010637. 31

[31] Nafiseh Imanian, Mohammad Ebrahim Shiri, e Parham Moradi. Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems. *Engineering Applications of Artificial Intelligence*, 36(0):148 – 163, 2014. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2014.07.012. 48, 52, 54, 66, 67, 88, 105, 106

[32] Hesam Izakian e Ajith Abraham. Fuzzy c-means and fuzzy swarm for fuzzy clustering problem. *Expert Systems with Applications*, 38(3):1835 – 1838, 2011. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2010.07.112. 53

[33] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, páginas 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-612-2. URL http://dl.acm.org/citation.cfm?id=645528.657646. 23, 30

[34] Marika Kaden, Wieland Hermann, e Thomas Villmann. Optimization of general statistical accuracy measures for classification based on learning vector quantization. In *Proc.*

of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'2014), páginas 47–52, 2014. 21, 28, 29

[35] Marika Kaden, Mandy Lange, David Nebel, Martin Riedel, Tina Geweniger, e Thomas Villmann. Aspects in classification learning-review of recent developments in learning vector quantization. *Foundations of Computing and Decision Sciences*, 39(2):79–105, 2014. 21, 28, 29

[36] Dervis Karaboga e Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007. ISSN 0925-5001. doi: 10.1007/s10898-007-9149-x. 22, 52, 54

[37] James Kennedy e Russell Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, páginas 1942–1948 vol.4, Nov 1995. doi: 10.1109/ICNN.1995.488968. 22, 45, 52, 54, 106

[38] Teuvo Kohonen. *Learning Vector Quantization for Pattern Recognition*. Report TKK-F-A. Helsinki University of Technology, 1986. ISBN 9789517539500. URL `https://books.google.com.br/books?id=PwEkAAAACAAJ`. 21, 28

[39] Teuvo Kohonen. *Self-Organizing Maps*. Springer, third edition, 2001. 21, 22, 27, 28

[40] Meelis Kull, Telmo Silva Filho, e Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In Aarti Singh e Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, páginas 623–631, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR. URL `http://proceedings.mlr.press/v54/kull17a.html`. 34

[41] Moshe Lichman. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 34, 42, 48, 96

[42] Kuan Cheng Lin, Kai Yuan Zhang, e Jason C. Hung. Feature selection of support vector machine based on harmonious cat swarm optimization. In *Ubi-Media Computing and Workshops (UMEDIA), 2014 7th International Conference on*, páginas 205–208, July 2014. doi: 10.1109/U-MEDIA.2014.38. 52

[43] Wei Liu, Junfeng He, e Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, páginas 679–686, 2010. 23, 24, 31

[44] Wei Liu, Jun Wang, Sanjiv Kumar, e Shih-Fu Chang. Hashing with graphs. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, páginas 1–8, 2011. 31

[45] Xueliang Liu e Benoit Huet. Concept detector refinement using social videos. In *VLS-MCMR 2010, International Workshop on Very-Large-Scale Multimedia Corpus, Mining and Retrieval, 29 October 2010, Florence, Italy*, Firenze, ITALY, 10 2010. doi: http://dx.doi.org/10.1145/1878137.1878142. URL `http://www.eurecom.fr/publication/3285`. 23, 30

[46] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, páginas 281–297, Berkeley, Calif., 1967. University of California Press. URL `http://projecteuclid.org/euclid.bsmsp/1200512992`. 27, 28

[47] Wendy L. Martinez e Angel R. Martinez. *Computational Statistics Handbook with MATLAB, Second Edition (Chapman & Hall/Crc Computer Science & Data Analysis)*. Chapman & Hall/CRC, 2007. ISBN 1584885661. 72

[48] Ryszard S. Michalski, Jaime G. Carbonell, e Tom M. Mitchell. *Machine Learning: An Artificial Intelligence Approach*. Number v. 2 in Machine Learning: A Multistrategy Approach. Morgan Kaufmann, 1986. ISBN 9780934613002. URL `https://books.google.com.br/books?id=f9RylgKpHZsC`. 20

[49] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-13316-4. 22

[50] MykoWeb. Fungi of California Species Index, janeiro 2017. URL `http://www.mykoweb.com/CAF/species_index.html`. 71

[51] Èlizbar A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020. URL `http://dx.doi.org/10.1137/1109020`. 27

[52] Rodrigo Y.M. Nakamura, Luis A.M. Pereira, Kelton A. Costa, Douglas Rodrigues, João P. Papa, e Xin-She Yang. BBA: A binary bat algorithm for feature selection. In *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, páginas 291–297, Aug 2012. doi: 10.1109/SIBGRAPI.2012.47. 52

[53] Emanuel Parzen. On estimation of a probability density function and mode. *Ann. Math. Statist.*, 33(3):1065–1076, 09 1962. doi: 10.1214/aoms/1177704472. URL http://dx.doi.org/10.1214/aoms/1177704472. 27, 28

[54] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, e Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 96

[55] Bruno Almeida Pimentel e Renata M.C.R. de Souza. A weighted multivariate fuzzy c-means method in interval-valued scientific production data. *Expert Systems with Applications*, 41(7):3223 – 3236, 2014. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2013.11.013. URL //www.sciencedirect.com/science/article/pii/S095741741300924X. 27

[56] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, páginas 61–74. MIT Press, 1999. 109

[57] Douglas Reynolds. *Gaussian Mixture Models*, páginas 659–663. Springer US, Boston, MA, 2009. ISBN 978-0-387-73003-5. doi: 10.1007/978-0-387-73003-5_196. URL http://dx.doi.org/10.1007/978-0-387-73003-5_196. 27, 28

[58] Atsushi Sato e Keiji Yamada. Generalized learning vector quantization. In D. S. Touretzky e M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, páginas 423–429. MIT Press, 1996. URL http://papers.nips.cc/paper/1113-generalized-learning-vector-quantization.pdf. 28, 29

[59] Petra Schneider, Michael Biehl, e Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Comput.*, 21(12):3532–3561, December 2009. ISSN 0899-7667. doi: 10.1162/neco.2009.11-08-908. URL http://dx.doi.org/10.1162/neco.2009.11-08-908. 38

[60] Petra Schneider, Kerstin Bunte, Han Stiekema, Barbara Hammer, Thomas Villmann, e Michael Biehl. Regularization in matrix relevance learning. *IEEE Transactions on Neural Networks*, 21(5):831–840, May 2010. ISSN 1045-9227. doi: 10.1109/TNN.2010.2042729. 29, 38

[61] Telmo M. Silva Filho e Renata M.C.R. Souza. Pattern classifiers with adaptive distances. In *The 2011 International Joint Conference on Neural Networks*, páginas 1508–1514, July 2011. doi: 10.1109/IJCNN.2011.6033403. 21, 23, 28, 29, 39

[62] Telmo M. Silva Filho e Renata M.C.R. Souza. A weighted learning vector quantization approach for interval data. In Tingwen Huang, Zhigang Zeng, Chuandong Li, e Chi-Sing Leung, editors, *Neural Information Processing*, volume 7665 of *Lecture Notes in Computer Science*, páginas 504–511. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34486-2. doi: 10.1007/978-3-642-34487-9_61. 21, 23, 28, 29, 39, 51

[63] Telmo M. Silva Filho e Renata M.C.R. Souza. Fuzzy learning vector quantization approaches for interval data. In *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, páginas 1–8, July 2013. doi: 10.1109/FUZZ-IEEE.2013.6622424. 21, 23, 24, 28, 29, 39, 51, 52, 64, 71, 105, 106

[64] Telmo M. Silva Filho, Renata M.C.R. Souza, e Ricardo B.C. Prudêncio. A swarm-trained k-nearest prototypes adaptive classifier with automatic feature selection for interval data. *Neural Networks*, 80:19 – 33, 2016. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2016.04.006. URL `http://www.sciencedirect.com/science/article/pii/S0893608016300363`. 21, 22, 23

[65] Renata M.C.R. Souza e Telmo M. Silva Filho. *Optimized Learning Vector Quantization Classifier with an Adaptive Euclidean Distance*, páginas 799–806. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-04274-4. doi: 10.1007/978-3-642-04274-4_82. URL `http://dx.doi.org/10.1007/978-3-642-04274-4_82`. 21, 23, 28, 39

[66] Alexandre Szabo e Leandro N. de Castro. The proposal of a constructive particle swarm classifier. In *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on*, páginas 164–168, Dec 2010. doi: 10.1109/NABIC.2010.5716317. 46, 51

[67] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`. 93, 97

[68] Ze Tian e Rui Kuang. *Global Linear Neighborhoods for Efficient Label Propagation*, páginas 863–872. 2012. doi: 10.1137/1.9781611972825.74. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611972825.74. 30

[69] Isaac Triguero, Salvador García, e Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, 2015. ISSN 0219-3116. doi: 10.1007/s10115-013-0706-y. URL http://dx.doi.org/10.1007/s10115-013-0706-y. 23, 30

[70] Ranga R. Vatsavai, Shashi Shekhar, e Thomas E. Burk. A semi-supervised learning method for remote sensing data mining. In *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, páginas 5 pp.–211, Nov 2005. doi: 10.1109/ICTAI.2005.17. 23, 30

[71] De Wang, Feiping Nie, e Heng Huang. Large-scale adaptive semi-supervised learning via unified inductive and transductive model. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, páginas 482–491, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623731. URL http://doi.acm.org/10.1145/2623330.2623731. 23, 30

[72] Fei Wang e Changshui Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2008. 30

[73] Jing Wang, Jingdong Wang, Gang Zeng, Zhuowen Tu, Rui Gan, e Shipeng Li. Scalable k-nn graph construction for visual descriptors. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1106–1113, June 2012. doi: 10.1109/CVPR.2012.6247790. 31

[74] Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, e Xindong Wu. Scalable semi-supervised learning by efficient anchor graph regularization. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1864–1877, July 2016. ISSN 1041-4347. doi: 10.1109/TKDE.2016.2535367. 23, 24, 32, 42, 90, 96

[75] Bin Xu, Jiajun Bu, Chun Chen, Can Wang, Deng Cai, e Xiaofei He. EMR: A scalable graph-based ranking model for content-based image retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):102–114, Jan 2015. ISSN 1041-4347. doi: 10.1109/TKDE.2013.70. 31

[76] Wei Xu. *Symbolic Data Analysis: Interval-valued Data Regression*. Phd thesis, University of Georgia, 2010. 71

[77] Seiji Yamada e Masayuki Okabe. Semisupervised query expansion with minimal feedback. *IEEE Transactions on Knowledge & Data Engineering*, 19(undefined):1585–1589, 2007. ISSN 1041-4347. doi: doi.ieeecomputersociety.org/10.1109/TKDE.2007.190646. 23, 30

[78] Bianca Zadrozny e Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proc. 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD'02)*, páginas 694–699. ACM, 2002. 109

[79] YingChao Zhang, Xiong Xiong, e QiDong Zhang. An improved self-adaptive pso algorithm with detection function for multimodal function optimization problems. *Mathematical Problems in Engineering*, 2013, 2013. 67, 106

[80] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, e Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, páginas 321–328. MIT Press, 2004. 30

[81] Zhi-Hua Zhou e Ming Li. Semisupervised regression with cotraining-style algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1479–1493, 2007. 23, 30

[82] Xiaojin Zhu e Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. 29

[83] Xiaojin Zhu, Zoubin Ghahramani, e John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, páginas 912–919, 2003. 23, 30, 32

Apêndice A – Pesos da Distância do IVABC

A.1 Pesos λ

Os vetores de pesos $\vec{\lambda}_{m,l}$ são calculados com o objetivo de minimizar o critério J_{IVABC}, definido na Equação (5.8), para a l-ésima partícula.

Proposição A.1.1. *Considerando a fórmula da distância $d\pi_{m,l}$ e fixando o m-ésimo protótipo da l-ésima partícula $\vec{w}_{m,l}$ e a j-ésima variável, os pesos $\lambda_{m,l,j}$ que minimizam o critério J_{IVABC} sob as restrições $\lambda_{m,l,j} > 0$ e $\prod_{j\in\Upsilon_l} \lambda_{m,l,j} = \gamma_{m,l,g}$ são calculados de acordo com a Equação* (A.1).

$$\lambda_{m,l,j} = \frac{\{\gamma_{m,l,g}[\prod_{h\in\Upsilon_l} \Delta_{m,l,h}]\}^{\frac{1}{|\Upsilon_l|}}}{\Delta_{m,l,j}}, \tag{A.1}$$

onde $\Delta_{m,l,j}$ é definido pela Equação (A.2).

$$\Delta_{m,l,j} = \sum_{i=1}^{N} \frac{\left[\left(\frac{x_{i,j,L} - w_{m,l,j,L}}{max(j_U) - min(j_L)}\right)^2 + \left(\frac{x_{i,j,U} - w_{m,l,j,U}}{max(j_U) - min(j_L)}\right)^2\right] \delta(m,l,k,i)}{2 \cdot |\Upsilon_l| \cdot ki \cdot N_{acertos,l}} \tag{A.2}$$

Prova A.1.1. Fixando o m-ésimo protótipo e a j-ésima variável, o critério J_{IVABC} pode ser reescrito como dado pela Equação (A.3).

$$J_{IVABC}(l) = \sum_{m=1}^{M} \sum_{j\in\Upsilon_l} \lambda_{m,l,j} \Delta_{m,l,j} \tag{A.3}$$

De acordo com instruções de Diday e Govaert [17] e mais por Govaert [27], a minimização de J_{IVABC} sob as restrições $\lambda_{m,l,j} > 0$ e $\prod_{j\in\Upsilon_l} \lambda_{m,l,j} = \gamma_{m,l,g}$ é feita usando a Equação (A.4).

$$\frac{\delta}{\delta \lambda_{m,l,j}} \left(\sum_{m=1}^{M} \sum_{j\in\Upsilon_l} \lambda_{m,l,j} \Delta_{m,l,j} - \mu \prod_{h\in\Upsilon_l} \lambda_{m,l,h} \right) = 0, \; for \; j \in \Upsilon_l \tag{A.4}$$

Da Equação (A.4) segue a Equação (A.5).

$$\Delta_{m,l,j} - \mu \frac{\prod_{h\in\Upsilon_l} \lambda_{m,l,h}}{\lambda_{m,l,j}} = 0 \Rightarrow \lambda_{m,l,j} = \frac{\mu}{\Delta_{m,l,j}} \left(\prod_{h\in\Upsilon_l} \lambda_{m,l,h} \right) \tag{A.5}$$

Lembrando que $\prod_{h\in\Upsilon_l}\lambda_{m,l,h}=\gamma_{m,l,g}$, o parâmetro na Equação (A.5) é dado pela Equação (A.6).

$$\lambda_{m,l,j}=\frac{\mu\gamma_{m,l,g}}{\Delta_{m,l,j}} \tag{A.6}$$

A restrição $\prod_{h\in\Upsilon_l}\lambda_{m,l,h}=\gamma_{m,l,g}$ pode ser escrita como mostra a Equação (A.7).

$$\gamma_{m,l,g}=\prod_{h\in\Upsilon_l}\frac{\mu\gamma_{m,l,g}}{\Delta_{m,l,h}}=\frac{\mu^{|\Upsilon_l|}(\gamma_{m,l,g})^{|\Upsilon_l|}}{\prod_{h\in\Upsilon_l}\Delta_{m,l,h}} \text{ então } \mu=\frac{\left(\gamma_{m,l,g}\prod_{h\in\Upsilon_l}\Delta_{m,l,h}\right)^{\frac{1}{|\Upsilon_l|}}}{\gamma_{m,l,g}} \tag{A.7}$$

Finalmente, a solução $\hat{\lambda}_{m,l,j}$ para o peso $\lambda_{m,l,j}$ é dada pela Equação (A.8).

$$\hat{\lambda}_{m,l,j}=\frac{\mu\gamma_{m,l,g}}{\Delta_{m,l,j}}=\frac{\{\gamma_{m,l,g}[\prod_{h\in\Upsilon_l}\Delta_{m,l,h}]\}^{\frac{1}{|\Upsilon_l|}}}{\Delta_{m,l,j}} \tag{A.8}$$

onde $\gamma_{m,l,g}$ modela o volume do m-ésimo protótipo da l-ésima partícula $\vec{w}_{m,l}$ na sua classe $y_{m,l}=g$ e $\Delta_{m,l,j}$ foi definido na Equação (A.2). Usar o peso $\gamma_{m,l,g}$ como uma restrição para os pesos $\lambda_{m,l,j}$ permite que a distância considere a dispersão dos dados representados pelo m-ésimo protótipo da l-ésima partícula quando seus pesos $\lambda_{m,l,j}$ são calculados.

A.2 Pesos γ

Na prática, os pesos γ servem para minimizar outro critério J_{IVABC2}, definido pela Equação (A.9), que é a soma das distâncias médias entre todos os protótipos da l-ésima partícula e os objetos corretamente afetados por eles, dos quais eles são vizinhos mais próximos.

$$J_{IVABC2}(l)=\sum_{g=1}^{G}\sum_{m\in\Omega_g}\frac{\sum_{i=1}^{N}\gamma_{m,l,g}d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)\delta(m,l,k,i)}{N_{m,l,k,g}}, \tag{A.9}$$

onde $\delta(m,l,k,i)=1$ se $\vec{w}_{m,l}$ for um dos k vizinhos mais próximos de $\vec{x}_i$ e $y_i=y_{m,l}=g$, caso contrário, $\delta(m,l,k,i)=0$ e $N_{m,l,k,g}$ é o número de ocorrências de $\vec{w}_{m,l}$ como um dos k vizinhos mais próximos dos N objetos de treinamento que pertencem a classe $y_{m,l}$. Aqui, o interesse não está na soma total das distâncias, mas em quão distante em média cada protótipo está das suas instâncias de treinamento corretamente afetadas.

Proposição A.2.1. *Usando o método dos multiplicadores de Lagrange com* $\gamma_{m,l,g}>0$ *e* $\prod_{m\in\Omega_g}\gamma_{m,l,g}=\beta_{l,g}$, *onde* Ω_g *é o conjunto de protótipos da* g*-ésima classe, os valores dos pesos*

γ que minimizam o critério J_{IVABC2} na Equação (A.9) *são calculados pela Equação* (A.10).

$$\gamma_{m,l,g} = \frac{\left[\beta_{l,g} \prod_{r \in \Omega_g} \left(\sum_{i=1}^{N} d\pi_{r,l}\left(\vec{x}_i, \vec{w}_{r,l}, \vec{\lambda}_{r,l}\right) N^{-1}_{r,l,k,g} \delta(r,l,k,i)\right)\right]^{\frac{1}{M_g}}}{\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right) N^{-1}_{m,l,k,g} \delta(m,l,k,i)}, \tag{A.10}$$

onde M_g é o tamanho de Ω_g.

Prova A.2.1. Fixando a g-ésima classe e o m-ésimo protótipo da l-ésima partícula $\vec{w}_{m,l}$, o critério J_{IVABC2} pode ser reescrito como mostra a Equação (A.11).

$$J_{IVABC2}(l) = \sum_{g=1}^{G} \sum_{m \in \Omega_g} \gamma_{m,l,g} \phi_{g,m,l} \text{ e } \phi_{g,m,l} = \sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l}\right) N^{-1}_{m,l,k,g} \delta(m,l,k,i) \tag{A.11}$$

Mais uma vez de acordo com os trabalhos de Diday e Govaert [17] e Govaert [27], a minimização de J_{IVABC2} é obtida através da Equação (A.12).

$$\frac{\delta}{\delta \gamma_{m,l,g}} \left(\sum_{g=1}^{G} \sum_{m \in \Omega_g} \gamma_{m,l,g} \phi_{g,m,l} - \mu \prod_{r \in \Omega_g} \gamma_{g,r,l} \right) = 0, \text{ para } m \in \Omega_g \tag{A.12}$$

Da Equação (A.12), resulta a Equação (A.13).

$$\phi_{g,m,l} - \mu \frac{\prod_{r \in \Omega_g} \gamma_{g,r,l}}{\gamma_{m,l,g}} = 0 \Rightarrow \gamma_{m,l,g} = \frac{\mu}{\phi_{g,m,l}} \left(\prod_{r \in \Omega_g} \gamma_{g,r,l} \right) \tag{A.13}$$

Lembrando que $\prod_{r \in \Omega_g} \gamma_{g,r,l} = \beta_{l,g}$, o parâmetro da Equação (A.13) é dado pela Equação (A.14).

$$\gamma_{m,l,g} = \frac{\mu \beta_{l,g}}{\phi_{g,m,l}} \tag{A.14}$$

A restrição $\prod_{r \in \Omega_g} \gamma_{g,r,l} = \beta_{l,g}$ pode ser escrita como mostra a Equação (A.15).

$$\beta_{l,g} = \prod_{r \in \Omega_g} \frac{\mu \beta_{l,g}}{\phi_{g,r,l}} = \frac{\mu^{M_g} (\beta_{l,g})^{M_g}}{\prod_{r \in \Omega_g} \phi_{g,r,l}} \text{ então } \mu = \frac{\left(\beta_{l,g} \prod_{r \in \Omega_g} \phi_{g,r,l}\right)^{\frac{1}{M_g}}}{\beta_{l,g}} \tag{A.15}$$

Por fim, a solução $\hat{\gamma}_{m,l,g}$ para o peso $\gamma_{m,l,g}$ é dada pela Equação (A.16).

$$\hat{\gamma}_{m,l,g} = \frac{\mu\beta_{l,g}}{\phi_{g,m,l}} = \frac{\left[\beta_{l,g}\prod_{r\in\Omega_g}\left(\sum_{i=1}^{N} d\pi_{r,l}\left(\vec{x}_i,\vec{w}_{r,l},\vec{\lambda}_{r,l}\right)N_{r,l,k,g}^{-1}\delta(r,l,k,i)\right)\right]^{\frac{1}{M_g}}}{\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)N_{m,l,k,g}^{-1}\delta(m,l,k,i)} \tag{A.16}$$

A.3 Pesos β

O peso $\beta_{l,g}$ captura a importância da g-ésima classe no banco de dados, considerando os protótipos da g-ésima classe na l-ésima. Na prática, os pesos β servem para minimizar um terceiro e último critério J_{IVABC3}, definido pela A.17.

$$J_{IVABC3}(l) = \sum_{g=1}^{G}\beta_{l,g}\frac{\sum_{m=1}^{M_g}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)\delta(m,l,k,i)}{N_g}, \tag{A.17}$$

onde $\delta(m,l,k,i) = 1$ se $\vec{w}_{m,l}$ for um dos k vizinhos mais próximos de $\vec{x}_i$ e $y_i = y_{m,l} = g$, caso contrário $\delta(m,l,k,i) = 0$ e N_g é o número de ocorrências de protótipos da g-ésima classe entre os k vizinhos mais próximos das instâncias de treinamento que pertencem à g-ésima classe.

Proposição A.3.1. *Usando o método dos multiplicadores de Lagrange com* $\beta_{l,g} > 0$ *e* $\prod_{g=1}^{G}\beta_{g_l} = 1$, *onde G é o número de classes do conjunto de dados, os valores dos pesos* β *que minimizam o critério* J_{IVABC3} *na Equação* (A.17) *são calculados pela Equação* (A.18).

$$\beta_{l,g} = \frac{\left[\prod_{a=1}^{G}\left(\sum_{m=1}^{M_a}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)N_a^{-1}\delta(m,l,k,i)\right)\right]^{\frac{1}{G}}}{\sum_{m=1}^{M_g}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)N_g^{-1}\delta(m,l,k,i)} \tag{A.18}$$

Prova A.3.1. Fixando a g-ésima, o critério J_{IVABC3} pode ser reescrito como mostra a Equação (A.19).

$$J_{IVABC3}(l) = \sum_{g=1}^{G}\beta_{l,g}\zeta_{l,g} \text{ e } \zeta_{l,g} = \sum_{m=1}^{M_g}\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i,\vec{w}_{m,l},\vec{\lambda}_{m,l}\right)N_g^{-1}\delta(m,l,k,i) \tag{A.19}$$

Novamente, de acordo com Diday e Govaert [17] e [27], a minimização de J_{IVABC3} é calculada de acordo com a Equação (A.20).

$$\frac{\delta}{\delta\beta_{l,g}}\left(\sum_{g=1}^{G}\beta_{l,g}\zeta_{l,g} - \mu\prod_{a=1}^{G}\beta_{l,a}\right) = 0, \text{ para } g = 1,\ldots,G \tag{A.20}$$

Da Equação (A.20), resulta a Equação (A.21).

$$\zeta_{l,g} - \mu \frac{\prod_{a=1}^{G} \beta_{l,a}}{\beta_{l,g}} = 0 \Rightarrow \beta_{l,g} = \frac{\mu}{\zeta_{l,g}} \left(\prod_{a=1}^{G} \beta_{l,a} \right) \tag{A.21}$$

Lembrando que $\prod_{a=1}^{G} \beta_{l,a} = 1$, o parâmetro da Equação (A.21) é dado pela Equação (A.22).

$$\beta_{l,g} = \frac{\mu}{\zeta_{l,g}} \tag{A.22}$$

A restrição $\prod_{a=1}^{G} \beta_{l,a} = 1$ pode ser escrita como mostra a Equação (A.23).

$$1 = \prod_{a=1}^{G} \frac{\mu}{\zeta_{l,a}} = \frac{\mu^G}{\prod_{a=1}^{G} \zeta_{l,a}} \text{ então } \mu = \left(\prod_{a=1}^{G} \zeta_{l,a} \right)^{\frac{1}{G}} \tag{A.23}$$

Finalmente, a solução $\hat{\beta}_{l,g}$ para o peso $\beta_{l,g}$ é dada pela Equação (A.24).

$$\hat{\beta}_{l,g} = \frac{\mu}{\zeta_{l,g}} = \frac{\left[\prod_{a=1}^{G} \left(\sum_{m=1}^{M_a} \sum_{i=1}^{N} d\pi_{m,l} \left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l} \right) N_a^{-1} \delta(m,l,k,i) \right) \right]^{\frac{1}{G}}}{\sum_{m=1}^{M_g} \sum_{i=1}^{N} d\pi_{m,l} \left(\vec{x}_i, \vec{w}_{m,l}, \vec{\lambda}_{m,l} \right) N_g^{-1} \delta(m,l,k,i)} \tag{A.24}$$

Apêndice B – Pesos da Distância do 4Pros

B.1 Pesos λ

Os vetores de pesos $\vec{\lambda}_m$ são calculados com o objetivo de minimizar o critério J_{4Pros}, definido na Equação (B.1).

$$J_{4Pros} = \sum_{m=1}^{M} \sum_{i=1}^{N} \mathrm{da}_m \left(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m \right) \delta(m,k,i), \tag{B.1}$$

onde $\delta(m,k,i) = 1$ se o m-ésimo protótipo for um dos k protótipos mais próximos da i-ésima instância, caso contrário $\delta(m,k,i) = 0$.

Proposição B.1.1. *Considerando a fórmula da distância da_m e fixando o m-ésimo protótipo $\vec{w}_m$ e a j-ésima variável, os pesos $\lambda_{m,j}$ que minimizam o critério J_{4pros} sob as restrições $\lambda_{m,j} > 0$ e $\prod_{j=1}^{p} \lambda_{m,j} = \gamma_m$ são calculados de acordo com a Equação* (B.2).

$$\lambda_{m,j} = \frac{\{\gamma_{m,g}[\prod_{h=1}^{p} \Delta_{m,h}]\}^{\frac{1}{p}}}{\Delta_{m,j}}, \tag{B.2}$$

onde $\Delta_{m,j}$ é definido pela Equação (B.3) *e representa a soma das diferenças quadráticas da j-ésima variável do m-ésimo protótipo e das instâncias das quais ele é um dos k vizinhos mais próximos.*

$$\Delta_{m,j} = \sum_{i=1}^{N} \left(x_{i,j} - w_{m,j} \right)^2 \delta(m,k,i) \tag{B.3}$$

Prova B.1.1. Fixando o m-ésimo protótipo e a j-ésima variável, o critério J_{4Pros} pode ser reescrito como dado pela Equação (B.4).

$$J_{4Pros} = \sum_{m=1}^{M} \sum_{j=1}^{p} \lambda_{m,j} \Delta_{m,j} \tag{B.4}$$

De acordo com instruções de Diday e Govaert [17] e mais por Govaert [27], a minimização de J_{4Pros} sob as restrições $\lambda_{m,j} > 0$ e $\prod_{j=1}^{p} \lambda_{m,j} = \gamma_m$ é feita usando a Equação (B.5).

$$\frac{\delta}{\delta \lambda_{m,j}} \left(\sum_{m=1}^{M} \sum_{j=1}^{p} \lambda_{m,j} \Delta_{m,j} - \mu \prod_{h=1}^{p} \lambda_{m,h} \right) = 0, \; for \; j \in \{1, \ldots, p\} \tag{B.5}$$

Da Equação (B.5) segue a Equação (B.6).

$$\Delta_{m,j} - \mu \frac{\prod_{h=1}^{p} \lambda_{m,h}}{\lambda_{m,j}} = 0 \Rightarrow \lambda_{m,j} = \frac{\mu}{\Delta_{m,j}} \left(\prod_{h=1}^{p} \lambda_{m,h} \right) \tag{B.6}$$

Lembrando que $\prod_{h=1}^{p} \lambda_{m,h} = \gamma_m$, o parâmetro na Equação (B.6) é dado pela Equação (B.7).

$$\lambda_{m,j} = \frac{\mu \gamma_m}{\Delta_{m,j}} \tag{B.7}$$

A restrição $\prod_{h=1}^{p} \lambda_{m,h} = \gamma_m$ pode ser escrita como mostra a Equação (B.8).

$$\gamma_m = \prod_{h=1}^{p} \frac{\mu \gamma_m}{\Delta_{m,h}} = \frac{\mu^p (\gamma_m)^p}{\prod_{h=1}^{p} \Delta_{m,h}} \text{ então } \mu = \frac{\left(\gamma_m \prod_{h=1}^{p} \Delta_{m,h} \right)^{\frac{1}{p}}}{\gamma_m} \tag{B.8}$$

Finalmente, a solução $\hat{\lambda}_{m,j}$ para o peso $\lambda_{m,j}$ é dada pela Equação (B.9).

$$\hat{\lambda}_{m,j} = \frac{\mu \gamma_m}{\Delta_{m,j}} = \frac{\{\gamma_m [\prod_{h=1}^{p} \Delta_{m,h}]\}^{\frac{1}{p}}}{\Delta_{m,j}} \tag{B.9}$$

onde γ_m modela o volume do m-ésimo protótipo $\vec{w}_m$ na conjunto de dados e $\Delta_{m,j}$ foi definido na Equação (B.3). Usar o peso γ_m como uma restrição para os pesos $\lambda_{m,j}$ permite que a distância considere a dispersão dos dados representados pelo m-ésimo protótipo quando seus pesos $\lambda_{m,j}$ são calculados.

B.2 Pesos γ

Na prática, os pesos γ servem para minimizar outro critério, J_{4Pros2}, definido pela Equação (B.10), que representa a soma das distâncias médias entre todos os protótipos e os objetos afetados por eles, dos quais eles são vizinhos mais próximos.

$$J_{4Pros2} = \sum_{m=1}^{M} \frac{\sum_{i=1}^{N} \gamma_m \mathrm{da}_m \left(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m \right) \delta(m,k,i)}{N_{m,k}}, \tag{B.10}$$

onde $\delta(m,k,i) = 1$ se $\vec{w}_m$ for um dos k vizinhos mais próximos de $\vec{x}_i$, caso contrário, $\delta(m,k,i) = 0$ e $N_{m,k}$ é o número de ocorrências de $\vec{w}_m$ como um dos k vizinhos mais próximos dos N objetos

de treinamento. Aqui, o interesse não está na soma total das distâncias, mas em quão distante em média cada protótipo está das suas instâncias de treinamento afetadas.

Proposição B.2.1. *Usando o método dos multiplicadores de Lagrange com* $\gamma_m > 0$ *e* $\prod_{m=1}^{M} \gamma_m = 1$, *onde* M *é o número total de protótipos, os valores dos pesos* γ *que minimizam o critério* J_{4Pros2} *na Equação* (B.10) *são calculados pela Equação* (B.11).

$$\gamma_m = \frac{\left[\prod_{r=1}^{M}\left(\sum_{i=1}^{N} da_r\left(\vec{x}_i, \vec{w}_r, \vec{\lambda}_r\right) N_{r,k}^{-1}\delta(r,k,i)\right)\right]^{\frac{1}{M}}}{\sum_{i=1}^{N} da_m\left(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m\right) N_{m,k}^{-1}\delta(m,k,i)}. \tag{B.11}$$

Prova B.2.1. Fixando o m-ésimo protótipo $\vec{w}_m$, o critério J_{4Pros2} pode ser reescrito como mostra a Equação (B.12).

$$J_{4Pros2}(l) = \sum_{m=1}^{M} \gamma_m \phi_m \text{ e } \phi_g = \sum_{i=1}^{N} \mathrm{da}_m\left(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m\right) N_{m,k}^{-1}\delta(m,k,i) \tag{B.12}$$

Mais uma vez de acordo com os trabalhos de Diday e Govaert [17] e Govaert [27], a minimização de J_{4Pros2} é obtida através da Equação (B.13).

$$\frac{\delta}{\delta\gamma_m}\left(\sum_{m=1}^{M}\gamma_m\phi_m - \mu\prod_{r=1}^{M}\gamma_r\right) = 0, \text{ para } m \in \{1,\ldots,M\} \tag{B.13}$$

Da Equação (B.13), resulta a Equação (B.14).

$$\phi_m - \mu\frac{\prod_{r=1}^{M}\gamma_r}{\gamma_m} = 0 \Rightarrow \gamma_m = \frac{\mu}{\phi_m}\left(\prod_{r=1}^{M}\gamma_r\right) \tag{B.14}$$

Lembrando que $\prod_{r=1}^{M}\gamma_r = 1$, o parâmetro da Equação (B.14) é dado pela Equação (B.15).

$$\gamma_m = \frac{\mu}{\phi_m} \tag{B.15}$$

A restrição $\prod_{r=1}^{M}\gamma_r = 1$ pode ser escrita como mostra a Equação (B.16).

$$1 = \prod_{r=1}^{M}\frac{\mu}{\phi_r} = \frac{\mu^M}{\prod_{r=1}^{M}\phi_r}, \text{ então } \mu = \left(\prod_{r=1}^{M}\phi_r\right)^{\frac{1}{M}} \tag{B.16}$$

Por fim, a solução $\hat{\gamma}_m$ para o peso γ_m é dada pela Equação (B.17).

$$\hat{\gamma}_m = \frac{\mu}{\phi_m} = \frac{\left[\prod_{r=1}^{M}\left(\sum_{i=1}^{N}\mathrm{da}_r\left(\vec{x}_i, \vec{w}_r, \vec{\lambda}_r\right) N_{r,k}^{-1}\delta(r,k,i)\right)\right]^{\frac{1}{M}}}{\sum_{i=1}^{N} d\pi_{m,l}\left(\vec{x}_i, \vec{w}_m, \vec{\lambda}_m\right) N_{m,k}^{-1}\delta(m,k,i)} \tag{B.17}$$