Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Programa de Pós-Graduação em Engenharia de Teleinformática

Luiz Soares de Andrade Filho

Projeto de Classificadores de Padrões Baseados em Protótipos Usando Evolução Diferencial

Fortaleza – Ceará
Dezembro 2014

Luiz Soares de Andrade Filho

Projeto de Classificadores de Padrões Baseados em Protótipos Usando Evolução Diferencial

Dissertação de Mestrado apresentada à Coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de ***Mestre em Engenharia de Teleinformática****.*

Área de Concentração: Sinais e Sistemas

Orientador : Prof. Dr. Guilherme de Alencar Barreto

Fortaleza – Ceará
Dezembro 2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Pós-Graduação em Engenharia - BPGE

A567p Andrade Filho, Luiz Soares de.
Projeto de classificadores de padrões baseados em protótipos usando evolução diferencial / Luiz Soares de Andrade Filho. – 2014.
132 f. : il. color. , enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Departamento de Engenharia de Teleinformática, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2014.
Área de concentração: Sinais e Sistemas.
Orientação: Prof. Dr. Guilherme de Alencar Barreto.

1. Teleinformática. 2. Redes neurais. I. Título.

CDD 621.38

UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

CAMPUS DO PICI, CAIXA POSTAL 6007 CEP 60.738-640
FORTALEZA – CEARÁ - BRASIL
FONE (+55) 85 3366-9467 – FAX (+55) 85 3366-9468

LUIZ SOARES DE ANDRADE FILHO

PROJETO DE CLASSIFICADORES DE PADRÕES BASEADOS EM PROTÓTIPOS USANDO EVOLUÇÃO DIFERENCIAL

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Teleinformática.
Área de concentração: Sinais e Sistemas.

Aprovada em: 28/11/2014.

BANCA EXAMINADORA

Guilherme de Alencar Barreto
Prof. Dr. Guilherme de Alencar Barreto (Orientador)
Universidade Federal do Ceará

Ricardo Coelho Silva
Prof. Dr. Ricardo Coelho Silva
Universidade Federal do Ceará

Adriano L. I. Oliveira
Prof. Dr. Adriano Lorena Inacio de Oliveira
Universidade Federal de Pernambuco

Dedico este trabalho àqueles cujo comprometimento se encontra na busca pela verdade e na compreensão da natureza.

Agradecimentos

À minha esposa Marília pelo companheirismo e apoio imprecindíveis,

Aos meus pais Luiz e Lucélia, pelo carinho e apoio ao longo desta atrajetória,

Ao Professor Guilherme de Alencar Barreto, pela confiança, incentivo e dedicação constante durante todo o período de orientação, sendo apoio fundamental para a realização deste projeto,

Aos demais professores do Departamento de Engenharia de Teleinformática, pelas importantes discussões durante o meu curso de mestrado,

À Universidade Federal do Ceará, por permitir este importante passo na minha carreira pessoal e profissional.

"Educar uma pessoa apenas no intelecto, mas não na moral, é criar uma ameaça à sociedade."

Theodore Roosevelt

Resumo

Nesta dissertação é apresentada uma abordagem evolucionária para o projeto eficiente de classificadores baseados em protótipos utilizando Evolução Diferencial. Para esta finalidade foram reunidos conceitos presentes na família de redes neurais LVQ (*Learning Vector Quantization*, introduzida por Kohonen (KOHONEN, 2001) para classificação supervisionada, juntamente com conceitos extraídos da técnica de clusterização automática proposta por Das et al. (DAS; ABRAHAM; KONAR, 2008) baseada na metaheurística Evolução Diferencial. A abordagem proposta visa determinar tanto o número ótimo de protótipos por classe, quanto as posições correspondentes de cada protótipo no espaço de cobertura do problema. Através de simulações computacionais abrangentes realizadas sobre vários conjuntos de dados comumente utilizados em estudos de comparação de desempenho, foi demonstrado que o classificador resultante, denominado LVQ-DE, alcança resultados equivalentes (ou muitas vezes até melhores) que o estado da arte em classificadores baseados em protótipos, com um número muito menor de protótipos.

Palavras-chaves: Classificação baseada em Protótipos, Redes Neurais Competitivas, *Learning Vector Quantization*, Seleção de Modelos, Evolução Diferencial.

Abstract

In this Master's dissertation we introduce an evolutionary approach for the efficient design of prototype-based classifiers using differential evolution (DE). For this purpose we amalgamate ideas from the Learning Vector Quantization (LVQ) framework for supervised classification by Kohonen (KOHONEN, 2001), with the DE-based automatic clustering approach by Das et al. (DAS; ABRAHAM; KONAR, 2008) in order to evolve supervised classifiers. The proposed approach is able to determine both the optimal number of prototypes per class and the corresponding positions of these prototypes in the data space. By means of comprehensive computer simulations on benchmarking datasets, we show that the resulting classifier, named LVQ-DE, consistently outperforms state-of-the-art prototype-based classifiers.

Keywords: Prototype-based Classification, Competitive Neural Networks, Learning Vector Quantization, Model Selection, Differential Evolution.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos

$\mathbf{X}$	*Matriz de vetores de entrada, onde* $\mathbf{x}_n$ *é o n-ésimo vetor*
$\mathbf{y}$	*Vetor de rótulos relacionados ao conjunto de entrada, onde* y_n *é a n-ésima componente*
$\mathbb{R}$	*Conjunto dos números reais*
t	*Épocas de treinamento*
$\mathbf{W}$	*Matriz de pesos neuronais, onde* $\mathbf{w}_n$ *é o n-ésimo vetor*
n_w	*Número de neurônios da rede*
k	*Número de classes do conjunto de dados*
d	*Distância euclidiana quadrática*
v_0	*Primeiro valor para a taxa de treinamento*
v_f	*Último valor para a taxa de treinamento*
η_t	*Taxa de treinamento para a t-ésima época*
e_p	*Total de épocas*
$\mathbf{M}$	*Matriz de protótipos, onde* $\mathbf{m}_n$ *é o n-ésimo Vetor*
n_m	*Número de protótipod da rede LVQ*
w	*Parâmetro largura da "janela"*
$\alpha_{\mathbf{n}}(t)$	*Taxa de aprendizagem individual na rede OLVQ*
Ω	*Conjunto de dados completo para a rede OLVQ*
δ_{ci}	*Delta de Kronecker*
$\mathbf{c}$	*Vetor dos padrões de classes na rede LVQTC*
$\mathbf{p}$	*Vetor dos contadores das classes na rede LVQTC*
$\mathbf{g}$	*Vetor das probabilidades globais na rede LVQTC*
$\mathbf{n}$	*Vetor do número total de padrões de treinamento da classe* $\boldsymbol{c}_i$ *na rede LVQTC*
$\mathbf{S}$	*Conjunto de treinamento de vetores rotulados na rede LVQTC*
p_{tot}	*Soma corrente sobre todo os contadores de treinamento do neurônio* $\boldsymbol{m}_c$ *na rede LVQTC*
f_r	*Fator de redução das taxas de aprendizado na rede LVQTC*
p_{prn}	*Parâmetro de corte na rede LVQTC*
f_{cvg}	*Fator limitante de parada na rede LVQTC*
r_{neu}	*Raiz quadrada da média dos quadrados da distância do neurônio para todos os padrões de treinamento mais próximos e que pertençam ao mesmo rótulo na rede LVQTC*
f_{cont}	*Contaminação do neurônio por uma classe errada na rede LVQTC*

f_{cmax}	*Máximo valor permitido de contaminação por neurônio na rede LVQTC*
p_{min}	*Contador mínimo do número detreinamentos do neurônio na rede LVQTC*
d_{max}	*Distância máxima de S para o neurônio, medida em unidades de R_{neu} na rede LVQTC*
$\mathbf{a}_x$	*A soma de todos $\boldsymbol{a}_i$, exceto $\boldsymbol{a}_k$, onde k indica a classe $\boldsymbol{c}_k$, na rede LVQTC*
$\mathbf{a}_i$	*Indica $(\boldsymbol{g}_i * \boldsymbol{p}_i)/\boldsymbol{n}_i$ na rede LVQTC*
a_{tot}	*A soma de todos os $\boldsymbol{a}_i$ na rede LVQTC*
$\mathbf{n}_c$	*Vetor do número de neurônios por classe da rede LVQ*
$f(\cdot)$	*Função de Aptidão*
$\mathbf{x}_{min}$	*Vetor dos menores valores possíveis para as variáveis de uma solução*
$\mathbf{x}_{max}$	*Vetor dos os maiores valores possíveis para as variáveis de uma solução*
$\mathcal{T}$	*Conjunto dos protótipos na rede SLVQ*
$\mathbf{I}_p$	*Matriz identidade de dimensões $p \times p$*
σ^2	*Parâmetro de dispersão na rede SLVQ*
$\mathbf{X}_i(t)$	*Vetor pai na DE*
$\mathbf{U}_i(t)$	*Vetor de ensaio na DE*
$\mathbf{X}_{i1}(t)$	*Vetor alvo na DE*
β	*Fator escalar na DE*
$\mathcal{J}$	*Conjunto de índices dos elementos que serão submetidos à perturbação na DE*
n_x	*Dimensão do problema na DE*
p_r	*Taxa de recombinação (crossover rate)*
ζ	*População de indivíduos na DE*
$\mathbf{Z}_{i,d}(t)$	*O i-ésimo vetor (cromossomo) da população no passo de tempo (geração) t possui d componentes (dimensões) na rede LVQDE*
n_s	*Tamanho da população na DE*
$\mathbf{V}_c(t)$	*Representação vetorial do cromossomo na rede LVQDE*
$\mathbf{T}_i^{(\omega_k)}$	*Limiar de ativação do referente à classe k e gene i do cromossomo na rede LVQDE*
$L_{max}^{\omega_k}$	*Quantidade máxima de genes que o cromossomo pode representar para a classe k*
$\mathbf{y}_k$	*Valor do rótulo referente a uma k-ésima classe na rede LVQDE*
$\mathbf{c}$	*Vetor que representa a centróide calculada para a classe $\boldsymbol{y}_k$ na rede LVQDE*
$\mathbf{c}_d$	*A componente da dimensão d do vetor centróide na rede LVQDE*
λ	*Fator de penalização na rede LVQDE*
$\mathbf{x}_d(i)$	*A componente d do vetor X(i) na rede LVQDE*
$\mathbf{d}_{max}$	*Vetor de amplitude máxima entre as componentes da centróide, $\boldsymbol{c}_d$, e as componentes mais distantes, $\boldsymbol{x}_d(i^*)$, das amostras de treinamento, na rede LVQDE*
$\mathbf{m}_g$	*Representação de um protótipo gerado (sem o rótulo), onde $\boldsymbol{m}_d$ são suas componentes, na rede LVQDE*

Lista de Siglas

LVQ (*Learning Vector Quantization*)

LVQDE (*Learning Vector Quantization based on Differential Evolution*)

NPC (*Nearest Prototype CLassification*)

MAP (Probabilidade Máximo a Posteriori)

PCA (*Principal Component Analysis*)

DE (*Differential Evolution*)

OLVQ (*Optimized Learning Vector Quantization*)

AS (Aprendizagem Supervisionada)

SOM (*Self-Organizing Maps*)

LVQTC (*Learning Vector Quantization with Training Count*)

SLVQ (*Soft Learning Vector Quantization*)

API (*Application Programming Interface*)

Jsci (*A Science API for Java*)

DH (Hérnia de Disco)

SL (Espondilolistese)

3D (Tridimensional)

2D (Bidimenssional)

HIPS (Sistema de Processamento de Imagem Hierárquica)

PSO (*Particles Swarm Optimization*)

Capítulo 1

Introdução

O *Learn Vector Quantization* (LVQ), ou Aprendizado por Quantização Vetorial, (KOHONEN, 1990a), (KOHONEN, 2001), (KOHONEN, 1995a) pertence a uma classe de algoritmos de aprendizado denominada *Nearest Prototype Classification* (NPC), ou Classificaçã por Proximidade baseada em Protótipos. O LVQ foi introduzido por Kohonen (KOHONEN, 1986) há aproximadamente vinte anos e, desde então, vem sendo amplamente utilizado (HELSINKI, 2002). Tal como o método dos k vizinhos mais próximos (DUDA; HART, 2000), o NPC é um método de classificação local em que as fronteiras de classificação são aproximações locais. Em vez de utilizar todos os pontos do conjunto de treinamento, o NPC conta com um conjunto de vetores protótipos apropriadamente selecionados. Isto torna o método com a complexidade computacional mais baixa, devido ao menor número de ítens que precisam ser armazenados, o que torna também o número de comparações no processo de classificação significativamente menor.

A utilização dos NPCs é fomentada pela literatura com base em duas características principais. A primeira motivação vem da Teoria de Decisão Bayesiana, que é uma abordagem fundamental para problemas de classificação. A construção do classificador envolve dois passos: 1) a construção do modelo das densidades de probabilidade para cada classe distinta, e 2) a construção das fronteiras de classificação utilizando o critério da probabilidade do máximo a posteriori (MAP). Se, por exemplo, as densidades de probabilidade forem bem aproximadas por meio da mistura gaussiana (KAMBHATLA; LEEN, 1995) e se assume-se que todos os componentes possuam intensidade e variância iguais, o classificador MAP se reduz a um NPC baseado em distância euclidiana entre a

amostra de dados e os centros dos seus componentes. Abordagens deste tipo podem funcionar bem na maioria dos casos (KAMBHATLA; LEEN, 1995), mas precisam lidar com a desvantagem que possuem para modelar problemas de maior complexidade. Nestes casos, a estimação das densidade das probabilidades e das fronteiras de classificação necessitam de um conjunto de treinamento grande para gerar um resultado satisfatório.

A segunda motivação parte da idéia da estimação direta das funções discriminantes para problemas de classificação do tipo multiclasse. Nos NPCs, as funções discriminantes são parametrizadas utilizando o conjunto de vetores de protótipos para cada classe. A classificação é baseada na distância entre a amostra de dados e a classe, a qual pertence ao protótipo mais próximo. Frequentemente é utilizada a distância euclidiana para medição de distância. Um dos métodos mais comuns para a construção de protótipos (baseado em funções de discriminantes) é o *Learning Vector Quantization* (LVQ) (KOHONEN, 1990a), (KOHONEN, 2001), para o qual muitas variantes foram desenvolvidas no passado. Os classificadores trabalhados nesta dissertação possuem um bom desempenho em muitas tarefas de classificação (HELSINKI, 2002), porém as regras de seleção (aprendizado) possuem a desvantagem de serem baseadas em heurísticas, o que pode acarretar em resultados não ótimos.

1.1 Motivação

As Redes LVQ possuem inúmeras aplicações práticas, como sistema de reconhecimento de padrões aplicado à detecção de novidade na busca de *outliers* em conjuntos de dados (CHO, October 2006), reconhecimento de gestos, movimento da marcha humana (KORDJAZI, 27-28 February 2012,Penang), sistema de avaliação da satisfação, qualidade de vida em escolas primárias (YUAN-YUAN, 2012) e sistema de controle para próteses virtuais para membros superiores (CAETANO, 2010).

Outras vertentes de pesquisa buscam desenvolver modelos de soluções híbridas, como reconhecimento automático de cinco tipos de glóbulos brancos (LVQ com PCA-*Principal Component Analysis*) (TABRIZI, August 31 - September 4, 2010), reconhecimento de padrões na classificação de fundo acústico em gravações (QIU-HUA, 2007), desenvolvimento de interfaces cérebro-computador utilizando potenciais visualmente evocados (KUGLER, Fevereiro de 2003) (ambos LVQ com Algoritimos Genéticos) e detecção de problema na interrupção de corrente elétrica

(LVQ com Transformada *Wavelet*) (G.MOKRYANI, 2009).

Uma importante etapa no projeto de uma rede LVQ é a quantização do número adequado de protótipos para cada classe. Este procedimento é atualmente realizado empiricamente, por análise do desempenho e curva de aprendizado ou baseado em um valor percentual sugerido (por exemplo: 5% do número de amostras do conjunto de dados) em trabalhos da área, normalmente com valores iguais para todas as classes (CHUNHONG, 2012),(JIANG, 2010). Quantidades pequenas de protótipos podem não ser suficientes para representar a distribuição dos dados de um problema, enquanto um valor demasiado alto pode causar sobretreino (*overfitting*) (ENGELBRECHT, 2007). Desta forma, a correta atribuição do número de protótipos por classe, além de ser uma atividade exaustiva, requer conhecimento especializado do usuário de operação.

1.2 Objetivos

O objetivo geral desta dissertação, assim como seus objetivos específicos, são apresentados nesta seção.

1.2.1 Objetivo Geral

Propor uma estratérgia baseada em computação evolucionária que permita um projeto eficiente de classificadores baseados em protótipos.

Conforme relatado, fica evidente a necessidade de um algoritmo do tipo LVQ que resolva o problema da quantização do númeto de protótipos para o projeto da rede, visando principalmente abranger a utilização prática deste método, permitindo o acesso a usuários leigos e facilitando o uso para os especializados. Além disso, o algoritmo deve estabelecer o número adequado de protótipos para cada classe, permitindo melhor modelagem do problema e desempenho na classificação. Também será buscada uma abordagem que apresente um desempenho similar (ou, de preferência, melhor) aos algoritmos clássicos da família LVQ e a alguns métodos mais robustos derivados do LVQ. Logo, esta dissertação propõe a análise e o desenvolvimento de uma nova versão do algoritmo LVQ que projete automaticamente sua arquitetura, de forma reduzida, visando performance e possibilitando resultados similares aos apresentados pelos métodos clássicos e modernos baseados em LVQ.

1.2.2 Objetivos Específicos

Os objetivos específicos desta dissertação estão listados a seguir:

1. Entender as limitações dos classificadores de padrões baseados em protótipos, principalmente os da família LVQ, através da implementação e testes em conjuntos de dados usados em *benchmarking*.

2. Propor um mecanismo de codificação do indivíduo na metaheurística Evolução Diferencial para definição automática do número de protótipos por classe, e para o posicionamento adequado destes protótipos.

3. Comparar o desempenho do método proposto com o estado da arte em classificadores baseados em protótipos.

1.2.3 Etapas do Processo

1. A primeira etapa será a realização do estudo das diversas variantes dos métodos de classificação clássicos e dos mais mais sofisticadas (mais recentes) da família LVQ. Serão também estudados métodos mais simples para fins de comparação, baseados em centroide e clusterização. Os algoritmos clássicos são os LVQ 1, LVQ 2, LVQ 2.1, LVQ 3, LVQ *Batch* e OLVQ; enquanto os mais modernos são o LVQTC e o *Soft* LVQ. Este estudo visa analisar o comportamento e as características de cada método com o objetivo de adquirir um compreensão mais detalhada dos métodos desta família. Também será realizado um levantamento bibliográfico das publicações de impacto dos últimos três anos relacionadas com o tema, buscando investigar os meios de criação do método a ser proposto.

2. Em seguida será realizado um estudo sobre algoritmos evolucionários, mais especificamente o *Differential Evolution*, assim como suas variantes DE/*best*/1/z, DE/x/nv/z, DE/*rand-to-best*/n_v/z e a DE/*current-to-best*/1+n_v/z. Serão estudadas as influência de parâmetros como comutação, recombinação (cruzamento binomial e cruzamento exponencial) e formas de seleção no comportamento do algoritmo. Tal estudo visa a compreensão do funcionamento do método em detrimento das características da aplicação adotada.

3. Na etapa seguinte será proposta uma nova abordagem de classificador híbrido baseado na fusão de uma Rede LVQ e no algoritmo evolucionário *Differential Evolution*. Serão estudadas as possibilidades de mudanças nas estruturas destes métodos com o objetivo de possibilitar a criação de um novo método híbrido. Também será desenvolvida uma função de aptidão para o *Differential Evolution* pretendendo alcançar as características requeridas para o novo método de classificação.

4. O último objetivo é a implementação de todos os métodos (e suas variações) e, em seguida, aplicá-los em uma seleção de dez *benchmarks* (conjuntos de dados conhecidos), com o objetivo de realizar uma avaliação de desempenho do LVQDE mais acurada em relação às demais abordagens de classificação. A avaliação será baseada em dados estatísticos e gráficos extraídos a partir de um número expressivo de experimentos executados para cada caso. A partir desta avaliação será possível determinar se o método proposto satisfaz os requisitos abordados na seção de objetivos principais.

1.3 Publicação

Ao longo do desenvolvimento desta dissertação o seguinte artigo foi publicado: "*On the Efficient Design of a Prototype-Based Classifier Using Diferential Evolution*", proceedings of the 2014 *IEEE Symposium Series on Computational Intelligence* (SSCI'2014), Orlando, Florida, USA.

1.4 Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma:

- O Capítulo 2 expõe a fundamentação teórica, discorrendo sobre o comportamento e as características de uma rede LVQ, a descrição dos classificadores clássicos da família LVQ, como os LVQ 1, LVQ 2, LVQ 2.1, LVQ 3, LVQ *Batch*, OLVQ e sobre alguns classificadores mais modernos e robustos, tal como os *Soft* LVQ e LVQTC.

- O Capítulo 3 apresenta a fundamentação teórica do algoritmo evolucionário *Differential Evolution*, descrevendo, também, todas as variantes clássicas presentes na literatura.

- Já o Capítulo 4 é dedicado à exposição do novo método de classificação elaborado, LVQ-DE (*Learning Vector Quantization based on Differential Evolution*), descrevendo seu algoritmo e detalhando seu comportamento com relação às influências dos seus parâmetros.

- O Capítulo 5 é dedicado à metodologia empregada nos experimentos utilizados para fins comparatórios entre os diversos métodos implementados, assim como informações sobre os dados utilizados (e preparação destes), além da apresentação das tabelas de resultados das simulações e das considerações a respeito dos experimentos.

- No Capítulo 6 as conclusões obtidas e as sugestões para trabalhos futuros são apresentadas.

- O documento termina com a apresentação das Referências Bibliográficas e Anexos.

Este capítulo apresentou uma breve introdução sobre os classificadores da família *Nearest Prototype Classification*, tal como características do algoritmo, aplicações práticas e motivações do seu uso. Também foi definido o objetivo geral e os objetivos específicos deste trabalho, publicações e como esta dissertação está organizada.

l

Capítulo 2

Classificação de Padrões Baseada em Protótipos

A Classificação baseada em Protótipos (CBP) abrange uma grande família de métodos e algoritmos de classificação de padrões supervisionado. Tal como o método dos K-vizinhos mais próximos ou *K-nearest neighbor* (KNN) (DUDA; HART; STORK, 2006), a PBC é um método de classificação local, no sentido que os limites de classificação são aproximados localmente. Em vez de utilizar todas as amostras de dados de treinamento, a CBP se baseia em um conjunto de vetores protótipos (centróide ou *codebook vectors*) adequadamente escolhidos. Assim, a CBP necessita de um número muito menor de itens para ser armazenado (modelagem do problema), na qual uma nova amostra de dados deve ser comparada para a classificação, tal como o KNN.

Além disso, a CBP possui duas propriedades desejáveis, que são difíceis de se encontrar em classificadores como os MLP padrão e SVM. Em primeiro lugar, devido á inerência da disposição local na construção das fronteiras de classificação, a interpretação das decisões em termos de regras explicativas locais associadas a cada protótipo é facilitada. Em segundo lugar, classificadores baseados em protótipo são facilmente dotados de estratégias adaptativas de adicão e exclusão de protótipos para ajuste da distribuição de dados corrente, uma propriedade valiosa, especialmente na evolução, ou seja, em ambientes não estacionários.

Classificadores baseados em protótipo têm sido desenvolvidos na literatura, basicamente, em três vertentes diferentes. Uma das abordagens provém da Teoria

de Decisão Bayesiana. Neste caso, o projeto do classificador envolve dois passos: 1) a construção de modelos para as densidades de probabilidade das diferentes classes e 2) a construção de fronteiras de classificação, utilizando um critério probabilístico de máximo *a posteriori* (MAP). Se, por exemplo, o modelo assume que (i) a densidades de probabilidade específicas da classe são bem aproximadas por densidades de gaussianas, (ii) as classes são equiprováveis, e (iii) as características de entrada são não correlacionadas e possuem variâncias iguais, o classificador MAP é reduzido a uma classe de algoritmos do tipo CBP muito popular, conhecidos como classificadores de distância dos mínimos (DM) (DUDA; HART; STORK, 2006).

A segunda abordagem origina-se da idéia da estimação direta das funções discriminantes aplicada a problemas de classificação multiclasse. Em CBP, isto é realizado utilizando-se um conjunto de vetores de protótipos para cada classe, e a classificação baseia-se na distância entre uma amostra de dados e a classe à qual pertence o protótipo mais próximo, da mesma maneira como é realizado pelos classificadores DM. Muitas vezes, a medida de distância Euclidiana é utilizada, mas, em princípio, qualquer medida de distância pode ser utilizada. Dois dos métodos mais comuns para a construção de funções discriminantes baseadas em protótipos são, respectivamente, a Aprendizagem por Quantização Vetorial (LVQ) (KOHONEN, 1990b, 2003) e as redes ARTMAP (CARPENTER *et al.*, 1992).

Finalmente, uma terceira abordagem para o projeto de um CBP pode ser usada para a elaboração de classificadores baseados em protótipos (supervisionados) a partir de métodos de *clustering* (ou seja, sem supervisão), como o *Self-Organizing Map* (SOM) (KOHONEN, 2013). Em primeiro lugar, as amostras de dados de treinamento são apresentadas à rede da SOM com um certo número de protótipos. Então, uma vez que o treinamento é concluído, os protótipos da rede SOM são rotulados de acordo com um esquema de votação por maioria, ou seja, um determinado protótipo recebe o rótulo da classe mais frequente entre as amostras de treinamento que estão mais próximos a ele. Quanto à DM e aos classificadores LVQ, a classificação baseia-se na distância entre uma amostra de dados e a classe a qual pertence o protótipo mais próximo.

Geralmente, a possibilidade de utilizar um conjunto de vetores de protótipos para cada classe dá mais flexibilidade para os segundo e terceiro métodos de CBP descritos há pouco. No entanto, o número de protótipos por classe tem que ser definido *a priori* para o segundo método ou pode ser determinada *a posteriori* (isto é, após a fase de

etiquetagem), para o terceiro método. Muitas vezes, estes números estão longe de serem ótimos, exigindo ao usuário uma grande quantidade de experimentação com os dados.

Neste capítulo as abordagens supracitadas para projeto de classificadores baseados em protótipos (CBP) serão apresentadas e discutidas com mais profundidade. Ênfase particular será dada à descrição dos classificadores membros da família LVQ, uma vez que a proposta principal da tese tem por inspiração esta classe de algoritmos de redes neurais artificiais.

2.1 Preliminares

Vamos considerar um conjunto de padrões de entrada e saída de treinamento $\{(\mathbf{x}_l, y_l)\}_{l=1}^{N}$, onde $\mathbf{x}_l \in \mathbb{R}^p$ caracteriza o l-ésimo padrão de entrada e $y_l \in \mathcal{C}$ denota o rótulo de sua classe correspondente. Note que y_l é uma variável discreta (de qualquer natureza numérica ou nominal) que pode assumir apenas um dos K valores no conjunto finito $\mathcal{C} = \{\omega_1, \omega_2, \ldots, \omega_K\}$.

Dado um conjunto de vetores protótipos rotulados $\mathbf{m}_i \in \mathbb{R}^p$, $i = 1, \ldots, n_m$, para todos os classificadores baseados em protótipos descritos nesta seção, onde a atribuição de classe para um novo padrão de entrada $\mathbf{x}(t)$ é baseada no seguinte critério de decisão:

$$\text{Classe de } \mathbf{x}(t) = \text{Classe de } \mathbf{m}_c(t), \tag{2.1}$$

onde

$$c = \arg \min_{i=1,\ldots,n_m} \{\|\mathbf{x}(t) - \mathbf{m}_i\|\}, \tag{2.2}$$

em que $\|\cdot\|$ expressa a distância euclidiana mensurada e c é o índice do protótipo mais próximo entre os n_m disponíveis. Nos parágrafos seguintes, serão descritas resumidamente as regras de aprendizagem para encontrar as posições dos protótipos $\mathbf{m}_i$, $i = 1, \ldots, n_m$ no espaço dos dados (de busca).

2.2 Abordagem Bayesiana

2.2.1 Classificador Distância Mínima ao Centróide

Nesta seção apresenta-se uma derivação com motivação estatística do classificador *Distância Mínima ao Centróide* (DMC), que é um dos mais simples exemplos de classificador baseado em protótipos.

Para começar, assume-se que se está de posse de um conjunto de N pares $\{\mathbf{x}_n, \omega_n\}_{n=1}^{N}$, em que $\mathbf{x}_n \in \mathbb{R}^p$ representa o n-ésimo padrão de entrada e ω_n é o rótulo da classe à qual pertence $\mathbf{x}_n$. Assume-se ainda que se tem um número finito e pré-definido de K classes ($K \ll N$), i.e. $\omega_n \in \{C_1, C_2, \ldots, C_K\}$. Por fim, seja n_i o número de exemplos da i-ésima classe (i.e. C_i). Assim, $N = n_1 + n_2 + \cdots + n_K = \sum_{i=1}^{K} n_i$.

Primeiramente, seja $p(C_i)$ a probabilidade *a priori* da i-ésima classe. Esta é probabilidade de a classe C_i ser selecionada *antes* do experimento ser realizado, sendo o experimento o ato de classificar um certo padrão. Perceba que este é um experimento aleatório, visto que não sabemos de antemão a que classe o padrão será atribuído. Logo, uma modelagem probabilística é plenamente justificável.

O modelo probabilístico mais simples para $p(C_i)$ é a densidade de probabilidade uniforme, ou seja, assume-se que todos os padrões da i-ésima classe são equiprováveis, i.e. tem a mesma probabilidade de ser selecionado aleatoriamente. Assim, pode-se estimar $p(C_i)$ como

$$p(C_i) = \frac{1}{n_i}, \tag{2.3}$$

em que n_i o número de exemplos da i-ésima classe, conforme definido no parágrafo anterior.

Agora vamos olhar apenas para os dados da classe C_i, ou seja, ao subconjunto de padrões $\mathbf{x}_n$ cujos rótulos são iguais a $\omega_n = C_i$. Um modelo probabilístico comum para estes dados é a densidade normal multivariada, denotada por $p(\mathbf{x}_n|C_i)$, de vetor-médio $\mathbf{m}_i$ e matriz de covariância $\mathbf{\Sigma}_i$. Matematicamente, este modelo é dado pela seguinte expressão:

$$p(\mathbf{x}_n|C_i) = \frac{1}{(2\pi)^{\frac{p}{2}}|\mathbf{\Sigma}_i|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}_n - \mathbf{m}_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x}_n - \mathbf{m}_i)\right\}, \tag{2.4}$$

em que $|\mathbf{\Sigma}_i|$ denota o determinante da matriz de covariância $\mathbf{\Sigma}_i$ e $\mathbf{\Sigma}_i^{-1}$ denota a inversa desta matriz.

A densidade $p(\mathbf{x}_n|C_i)$, no contexto de classificação de padrões, também é chamada de *função de verossimilhança* da classe C_i. A função de verossimilhança da classe C_i pode ser entendida como o modelo probabilístico que "explica" como os dados estão organizados (i.e. distribuídos) nesta classe.

Supõe-se agora que um novo padrão $\mathbf{x}_n$ é observado. Pergunta-se então qual é a probabilidade de que este padrão pertença à classe C_i? Em outras palavras, dado $\mathbf{x}_n$, qual a probabilidade de ocorrer C_i? Esta informação pode ser modelada através da função densidade *a posteriori* da classe, $p(C_i|\mathbf{x}_n)$.

Através do Teorema da Probabilidade de Bayes, a densidade a posteriori $p(\mathbf{x}_n|C_i)$ pode ser relacionada com a densidade a priori $p(C_i)$ e a função de verossimilhança $p(\mathbf{x}_n|C_i)$ por meio da seguinte expressão:

$$p(C_i|\mathbf{x}_n) = \frac{p(C_i)p(\mathbf{x}_n|C_i)}{p(\mathbf{x}_n)}. \tag{2.5}$$

Um critério comumente usado para tomada de decisão em classificação de padrões é o critério do *máximo a posteriori* (MAP). Ou seja, um determinado padrão $\mathbf{x}_n$ é atribuído à classe C_j se a moda da densidade a posteriori $p(C_j|\mathbf{x}_n)$ for a maior dentre todas. Em outras palavras, tem-se a seguinte regra de decisão:

$$\text{Atribuir } \mathbf{x}_n \text{ à classe } C_j, \text{ se } p(C_j|\mathbf{x}_n) > p(C_i|\mathbf{x}_n),\ \forall i \neq j.$$

O critério MAP também é comumente escrito como

$$C_j = \arg \max_{i=1,\ldots,K} \{p(C_i|\mathbf{x}_n)\}, \tag{2.6}$$

em que o operador "arg max" retorna o "argumento do máximo", ou seja, o conjunto de pontos para os quais a função de interesse atinge seu valor máximo.

Ao substituir a Eq. (2.5) na regra de decisão do critério MAP, obtém-se uma nova regra de decisão, dada por

$$\text{Atribuir } \mathbf{x}_n \text{ à classe } C_j, \text{ se } p(C_j)p(\mathbf{x}_n|C_j) > p(C_i)p(\mathbf{x}_n|C_i),\ \forall i \neq j,$$

em que o termo $p(\mathbf{x}_n)$ é eliminado por estar presente em ambos os lados da inequação. Em outras palavras, o termo $p(\mathbf{x}_n)$ não influencia na tomada de decisão feita por meio do critério MAP. Além disso, uma suposição comumente feita na prática é a de que as densidades a priori das classes são iguais, ou seja

$$p(C_1) = P(C_2) = \cdots = P(C_K), \tag{2.7}$$

o que equivale a supor que as classes são equiprováveis. Com isto, é possível simplificar ainda mais o critério MAP:

$$\text{Atribuir } \mathbf{x}_n \text{ à classe } C_j, \text{ se } p(\mathbf{x}_n|C_j) > p(\mathbf{x}_n|C_i),\ \forall i \neq j,$$

de tal forma que a regra de decisão passa a depender somente das funções de verossimilhança das classes. Neste caso, o critério MAP é comumente chamado de critério da máxima verossimilhança (*maximum likelihood criterion*, ML).

Conforme mencionado anteriormente, para a função de verossimilhança costuma-se usar a densidade normal multivariada mostrada na Eq. (2.4). Neste caso, o critério ML é escrito tomando-se o logaritmo natural das funções de verossimilhança envolvidas. Assim, o critério ML pode ser escrito como

$$\text{Atribuir } \mathbf{x}_n \text{ à classe } C_j, \text{ se } \ln p(\mathbf{x}_n|C_j) > \ln p(\mathbf{x}_n|C_i),\ \forall i \neq j,$$

ou como

$$C_j = \arg \max_{i=1,\ldots,K} \{\ln p(\mathbf{x}_n|C_i)\}. \tag{2.8}$$

Considerando a Eq. (2.4), tem-se que a função de log-verossimilhança $\ln p(\mathbf{x}_n|C_i)$ pode ser escrita como

$$\ln p(\mathbf{x}_n|C_i) = -\frac{1}{2}(\mathbf{x}_n - \mathbf{m}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_n - \mathbf{m}_i) - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i|, \tag{2.9}$$

em que nota-se que o termo $-\frac{p}{2}\ln(2\pi)$ não influencia na tomada de decisão uma vez que independe de i. Assim, este termo pode ser desconsiderado sem prejuízo ao resultado da classificação.

Para finalizar, duas suposições simplificadoras podem ainda ser feitas a fim de simplificar a Eq. (2.9). São elas:

1. As estruturas de covariâncias das K classes são iguais, ou seja, suas matrizes de covariância são iguais. Em outras palavras,

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \cdots = \boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}. \tag{2.10}$$

2. Além disso, os atributos de $\mathbf{x}_n$ são descorrelacionados entre si e possuem mesma variância (que pode ser feita igual a 1). Neste caso, tem-se que a matriz de covariância de todas as classes é dada por

$$\boldsymbol{\Sigma} = \mathbf{I}_p, \tag{2.11}$$

em que $\mathbf{I}_p$ é a matriz identidade de ordem p. Logo, tem-se que $\mathbf{\Sigma}^{-1} = \mathbf{I}_p$.

Assim, levando em consideração as simplificações acima, o critério ML passa ser escrito como

$$\begin{aligned} C_j &= \arg \max_{i=1,\ldots,K} \{-\frac{1}{2}(\mathbf{x}_n - \mathbf{m}_i)^T \mathbf{I}_p (\mathbf{x}_n - \mathbf{m}_i)\}, \\ &= \arg \max_{i=1,\ldots,K} \{-\frac{1}{2}(\mathbf{x}_n - \mathbf{m}_i)^T (\mathbf{x}_n - \mathbf{m}_i)\}, \\ &= \arg \max_{i=1,\ldots,K} \{-\frac{1}{2}\|\mathbf{x}_n - \mathbf{m}_i\|^2\}, \end{aligned} \tag{2.12}$$

em que foi considerado o fato de que a norma quadrática de um vetor $\mathbf{v}$ qualquer é igual ao produto interno deste vetor com ele mesmo: $\|\mathbf{v}\|^2 = \mathbf{v}^T\mathbf{v}$.

A partir do desenvolvimento acima exposto, pode-se concluir que o critério ML em última instância depende apenas da distância euclidiana do padrão $\mathbf{x}_n$ ao vetor-médio (ou centróide) da classe C_i. Na Eq. (2.12, o argumento é maximizado, mas desconsiderando o sinal negativo, pode-se buscar pelo argumento que "minimiza" a função distância $\|\mathbf{x}_n - \mathbf{m}_i\|$. Deste modo, pode-se reescrever o critério ML como

$$\text{Atribuir } \mathbf{x}_n \text{ à classe } C_j, \text{ se } \|\mathbf{x}_n - \mathbf{m}_j\| < \|\mathbf{x}_n - \mathbf{m}_i\|, \forall i \neq j,$$

ou como

$$C_j = \arg \min_{i=1,\ldots,K} \{\|\mathbf{x}_n - \mathbf{m}_i\|\}. \tag{2.13}$$

2.3 Algoritmos da Família LVQ

Na rede LVQ cada vetor de entrada é apresentado a todos os neurônios, conforme Figura 2.1. Um procedimento de comparação é realizado em cada neurônio gerando um resultado. Os resultados de todos os neurônios são apresentados em uma camada única destes neurônios, em que cada neurônio representa uma classe (não necessariamente um por classe) ou padrão das informações utilizadas no treinamento (KONO, 2008).

Designa-se a amostra de entrada como $\mathbf{x}(n) \in \mathbb{R}^d$ na instância n, que é recebida por todos os n_w neurônios da rede LVQ, que são definidos pelos vetores de pesos . Inicialmente os n_w vetores de pesos, que representam os neurônios, recebem valores aleatórios, tais como os rótulos. No presente trabalho, os valores iniciais

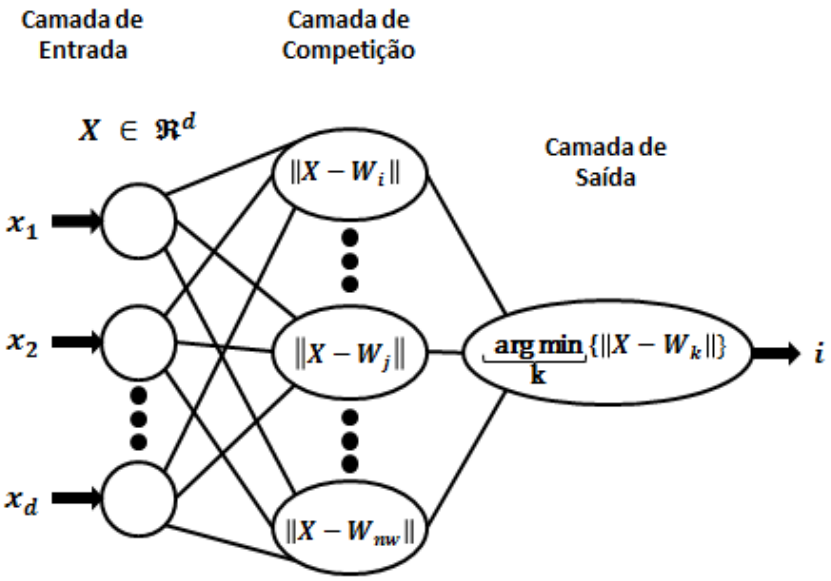


Figura 2.1: Apresenta a arquitetura de uma rede LVQ utilizada para exemplificar seu funcionamento (BEZDEK; PAL, 1995).

dos neurônios são selecionados aleatoriamente do conjunto de treinamento, visto que melhora a performance na convergência. O número de neurônios (ou protótipos) por classe pode ser determinado de forma uniforme ou não, podendo haver múltiplos protótipos por classe, de acordo com as características do problema (NUNES, 2010).

No processo de competição é utilizada uma medida de dissimilaridade, na qual é utilizada a distância euclidiana quadrática, denominada $\mathbf{d}_i(n)$, entre os vetores de entrada e o protótipo $\mathbf{m}_i(n)$. Dessa forma, temos que

$$\mathbf{d}_i(n) = \|\mathbf{x}(n) - \mathbf{m}_i(n)\|^2, \forall i = 1, 2, \cdots, n_w, \tag{2.14}$$

conforme os vetores relativos aos neurônios são atualizados, formam-se regiões no espaço de dados de entrada. Estas regiões são delimitadas por hiperplanos, resultando em um Diagrama de Voronoi, vide a Figura 2.2 (KOHONEN, 1995b).

2.3.1 Algoritmo LVQ-1

Para a n-ésima iteração do algoritmo é dado o par de entrada $\{\mathbf{x}(n), y(n)\}$, onde $\mathbf{X}$ e $\mathbf{y}$ são os conjuntos dos dados (variáveis) de entrada e o conjunto de rótulos, respectivamente. No processo de competição o neurônio campeão i^* é aquele que obtiver maior similaridade (menor distância) do vetor de treinamento $x(n)$, de acordo

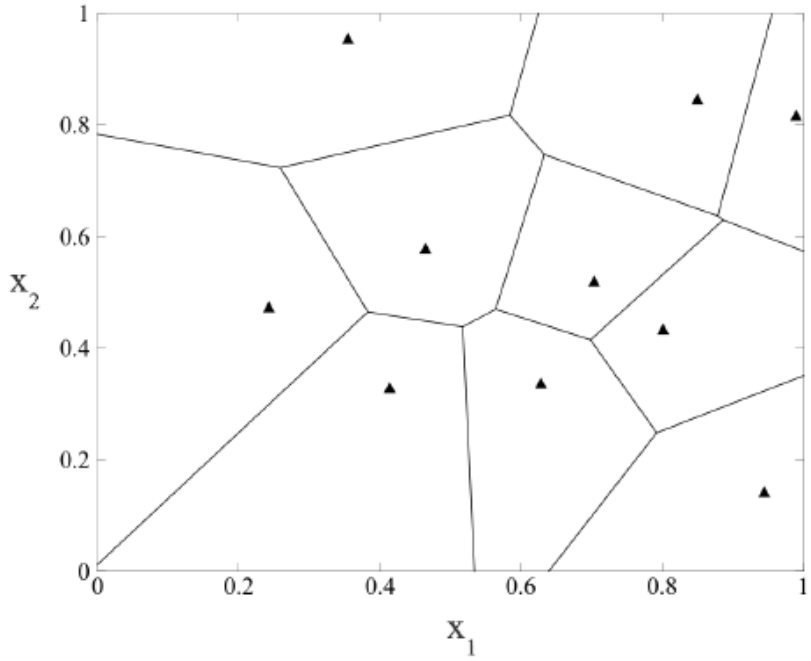


Figura 2.2: Diagrama de Voronoi (XUAN, 2009).

com a regra a seguir:

$$i^* = \arg\min \forall i \|\mathbf{x}(n) - \mathbf{m}_i(n)\|, \text{ em que } \|o\| \text{ é a norma euclidiana.} \tag{2.15}$$

Na etapa de atualização dos pesos, apenas o neurônio vencedor será atualizado com o vetor de entrada, segundo a expressão:

$$\mathbf{m}_i \cdot (n+1) = \mathbf{m}_i \cdot (n) + s(n)\eta \left[\mathbf{x}(n) - \mathbf{m}_i(n)\right], \tag{2.16}$$

onde $s(n) = 1$, quando o rótulo vencedor $\mathbf{m}_i(n)$ for igual ao do vetor de entrada $\mathbf{y}(n)$. Caso contrário, $s(n) = -1$ quando o vetor de entrada não possui a mesma classe do vetor vencedor. O parâmetro η é a taxa de aprendizagem, que deve ter valores preferencialmente entre 0 e 1. No algoritmo LVQ-1 utiliza-se apenas um valor (em comum) para a aprendizagem para todos os neurônios. É importante ressaltar que o momento de parada da execução é determinado por um número predeterminado de épocas fixado anteriormente à execução. Outra possibilidade de parada automática é avaliada, após cada época, se as variações das mudanças de pesos dos neurônios já estão suficientemente pequenas. Neste trabalho, o número de épocas foi predeterminado e a taxa de aprendizado apresenta um decrescimento

exponencial, inversamente proporcional à época de treinamento, conforme a equação a seguir:

$$\eta_t = v_0 \cdot e^{-t\left(\frac{-\ln\frac{v_f}{v_0}}{e_p}\right)}, \tag{2.17}$$

onde v_0 e v_f são, respectivamente, os primeiros e os últimos valores para a taxa de treinamento. η_t é a taxa de treinamento para a t-ésima época e e_p é o total de épocas. O decrescimento monotónico exponencial permite uma convergência mais estável para os vetores dos protótipos na etapa de treinamento, podendo ser também do tipo linear (HAYKIN, 2001a). Decrescendo preferencialmente a partir de 0.1 até próximo de 0, por exemplo 0.0001.

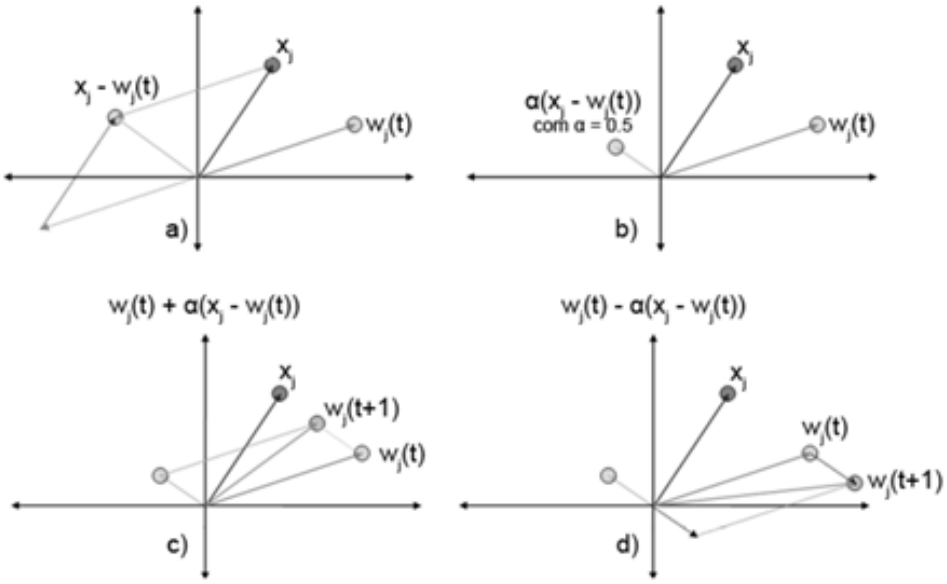


Figura 2.3: Exemplo do comportamento geométrico dos vetores envolvidos no algoritmo LVQ (PEREIRA B. DE B., 2009).

- A Figura 2.3(a) representa a operação de subtração $(\mathbf{x}_j - \mathbf{m}_j)$ na iteração t, que determina o deslocamento máximo que o vetor protótipo pode ter;
- A Figura 2.3(b) representa o vetor resultante da multiplicação do coeficiente de aprendizado pelo vetor da subtração apresentado na Figura 2.3(a). Este vetor determina efetivamente o tamanho do deslocamento que o vetor protótipo sofrerá;
- A Figura 2.3(c) mostra o efeito da adição do vetor resultante da etapa anterior pelo vetor protótipo, em que ocorre uma aproximação entre o registro de

entrada e o vetor protótipo. Esta operação é utilizada quando ocorre a classificação correta do registro de entrada;

- A Figura 2.3(d) mostra o efeito da subtração do vetor resultante da etapa anterior pelo vetor protótipo, em que ocorre um afastamento entre o registro de entrada e o vetor protótipo. Esta operação é utilizada quando ocorre a classificação errada do registro de entrada;

2.3.2 Algoritmo LVQ-2

O processo de decisão na classificação é idêntico ao LVQ-1. Já no aprendizado serão utilizados dois vetores de *codebook* $\mathbf{m}_i$ e $\mathbf{m}_j$, que serão os protótipos mais próximos ao vetor de entrada $\mathbf{x}(n)$, onde serão atualizados simultaneamente. São classificados como vencedor e vice-vencedor. Um deles deve possuir a classe igual a do vetor de entrada e o outro a classe diferente. Desta forma, $\mathbf{x}(n)$ deve cair numa zona de valores chamada janela, definida entre $\mathbf{m}_i$ e $\mathbf{m}_j$. Sejam d_i e d_j as distâncias euclidianas entre $\mathbf{x}(n)$ e $\mathbf{m}_i$ e $\mathbf{m}_j$, respectivamente. Então, $\mathbf{x}(n)$ cairá numa janela de largura w se

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s, \text{ onde } s = \frac{1-w}{1+w} \tag{2.18}$$

Recomenda-se que o parâmetro largura da "janela" w seja entre 0.2 e 0.3.

2.3.3 Algoritmo LVQ-2.1

Utiliza-se a seguinte expressão para atualização dos vetores protótipos:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) - \eta(t)\left[\mathbf{x}(t) - \mathbf{m}_i(t)\right] \tag{2.19}$$

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \eta(t)\left[\mathbf{x}(t) - \mathbf{m}_j(t)\right] \tag{2.20}$$

$\mathbf{m}_i$ e $\mathbf{m}_j$ são os vetores mais próximos de $\mathbf{x}(n)$, onde $\mathbf{m}_j$ e $\mathbf{x}(n)$ são da mesma classe, e $\mathbf{m}_i$ e $\mathbf{x}(n)$ de classes diferentes. A zona de "janela" deve ser aplicada da mesma forma. É importante ressaltar que esta regra deve ser adicionada na fase de treinamento do LVQ-2.

2.3.4 Algoritmo LVQ-3

Uma das motivações do LVQ-3 é que ele considera o que acontece com a localização dos protótipos a longo prazo, o que não acontece com o LVQ-2/LVQ-2.1. Desta forma, foi necessária a inserção de um atributo corretor que possibilite que os protótipos continuem convergindo para as distribuições das classes. A grande diferença do LVQ-3 para o LVQ-2.1 é que o LVQ-3 permite que os neurônios campeão e vice-campeão sejam da mesma classe. Adicionando-se aos procedimentos encontrados no LVQ-2.1, quando os protótipos são da mesma classe, tem-se:

$$\mathbf{m}_k(t+1) = \mathbf{m}_k(t) + \epsilon\eta(t)\left[\mathbf{x}(t) - \mathbf{m}_k(t)\right], \text{ para } k \in i, j \tag{2.21}$$

A partir de uma série de experimentos realizados pelo criador do método (Kohonen), foram encontrados valores aplicáveis a ϵ entre 0.1 e 0.5, relacionados a $w = 0.2$ ou 0.3, respectivamente. Os valores ótimos de ϵ mostram ter uma relação direta com o valor w utilizado na "janela", maior para mais abertas, e vice versa. Dessa forma, com a utilização do atributo ϵ a posição dos protótipos mostram-se estáveis quando estes atingem seus pontos sub-ótimos na cobertura das classes.

2.3.5 Algoritmo OLVQ

Nesta variação do LVQ-1 básico (nesse trabalho chamada de OLVQ), uma taxa de aprendizagem individual $\alpha_n(t)$ é designada para cada protótipo $\mathbf{m}_n$. Assim, o seguinte processo de aprendizagem é obtido:

1 - Inicialização

Seleciona-se aleatoriamente n_m padrões distintos do conjunto de treinamento, produzindo o subconjunto inicial de protótipos $\{\mathbf{m}_1(t), \ldots, \mathbf{m}_{n_m}(t)\}$ no passo $t = 0$.

De $n = 1$ até n_m inicie $\alpha_n(t)$.

2 - Passo de atualização dos protótipos: Escolha um padrão de treinamento $\mathbf{x}_i$ de Ω aleatoriamente com reposição.

Defina o protótipo vencedor $\mathbf{w}_c$ tal que

$$\mathbf{c} = \arg \min_{n=1,\ldots,n_m} d_e(\mathbf{x}_i, \mathbf{m}_n) \tag{2.22}$$

Se classe $(\mathbf{x}_i)$ = classe $(\mathbf{m}_c)$ faça $\mathbf{m}_c(t+1) = \mathbf{m}_c(t) + \alpha_c(t)\,[\mathbf{x}_i - \mathbf{m}_c(t)]$.

Se classe $(\mathbf{x}_i) \neq$ classe $(\mathbf{m}_c)$ faça $\mathbf{m}_c(t+1) = \mathbf{m}_c(t) - \alpha_c(t)\,[\mathbf{x}_i - \mathbf{m}_c(t)]$.

Atualize

$$\alpha_c(t+1) = \frac{\alpha_c(t)}{1 + s(t)\alpha_c(t)} \tag{2.23}$$

onde $s(t) = +1$ se $\mathbf{x}_i$ e $\mathbf{m}_c$ pertencem à mesma classe, e $s(t) = -1$ se $\mathbf{x}_i$ e $\mathbf{m}_c$ pertencem a classes diferentes. Em que d_e denota o operador de distância euclidiana.

3 - Critério de parada: Se todos os $\alpha_n(t) = 0 (n = 1, \ldots, n_m)$ então o algoritmo convergiu. Caso contrário, repetir o passo 2.

É preciso ter uma precaução quanto a forma de atualização das taxas de aprendizagem, pois $\alpha_c(t)$ pode também aumentar. Deve-se evitar que $\alpha_c(t)$ torne-se maior do que 1, o que pode ser imposto no próprio algoritmo. Para valores iniciais dos α_n pode-se escolher algo entre $0,3$ e $0,5$.

2.3.6 Algoritmo LVQ-*Batch*

A versão básica do algoritmo LVQ-1 pode ser escrita de forma reduzida da seguinte maneira

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)s(t)\delta_{ci}\,[\mathbf{x}(t) - \mathbf{m}_i(t)]\ , \tag{2.24}$$

onde $s(t) = +1$ se $\mathbf{x}$ e $\mathbf{m}_c$ pertencerem à mesma classe,
mas $s(t) = -1$ se $\mathbf{x}$ e $\mathbf{m}_c$ pertencerem à classes diferentes.

Desta forma δ_{ci} é denominado Delta de Kronecker ($\delta_{ci} = 1$ para $c = i, \delta_{ci} = 0$ para $c \neq i$).

O algoritmo LVQ-1 pode ser expresso em uma versão *batch*. Assim como uma versão similar ao algoritmo SOM na versão *batch*, a condição de equilíbrio para a

LVQ-1 é expressa como

$$\forall_i,\ E_t\left\{s\delta_{ci}(\mathbf{x}-\mathbf{m}_i^*)=0\right\}. \tag{2.25}$$

Os passos da computaçã do algoritmo *Batch*-LVQ (onde nos passos 2 e 3, os rótulos das classes dos protótipos são redefinidos dinamicamente), poderão então serem expressos, analogicamente ao *Batch Map*, como a seguir:

1. A configuração inicial dos protótipos deve ser iniciada selecionando-se aleatoriamente valores no conjunto de dados de treinamento sem reposição. Nesta etapa os valores originados na classificação de $\mathbf{x}(t)$ ainda não são levados em conta.

2. Inserir o $\mathbf{x}(t)$ novamente, desta vez lendo o $\mathbf{x}(t)$ assim como o valor do rótulo da sua classe sob cada nó vencedor correspondente.

3. Determinar os rótulos dos nós de acordo com a maioria dos rótulos de classes dos exemplos da lista.

4. Multilicar cada lista parcial todos os $\mathbf{x}(t)$ pelos fatores correspondentes $s(t)$ que indicam se $\mathbf{x}(t)$ e $\mathbf{m}_c(t)$ pertencem à mesma classe ou não.

5. Para cada nó i, tomar para o novo valor do vetor de referência a entidade

$$\mathbf{m}_i^* = \frac{\sum_{t'} s(t')\mathbf{x}(t')}{\sum_{t'} s(t')}, \tag{2.26}$$

   onde o somatório assume os índices t' destes exemplos, aqui listado sob o nó i. Os índices t' denotam a época posterior à época corrente (atual) da iteração.

6. Repetir a partir do passo 2 algumas vezes.

Comentário 1. Por questões de estabilidade poderá ser necessário checar o sinal do termo $\sum_{t'} s(t')$. Se ele se tornar negativo, o valor deste nó não deverá ser atualizado.

Comentário 2. Diferentemente do algoritmo LVQ usual, é permitido que a rotulação dos nós mude ao longo das iterações. Esta estratégia normalmente tem rendido melhor exatidão nas classificações do que quando os rótulos dos nó são fixados nos primeiros passos.

2.3.7 Algoritmo LVQ-TC

O LVQTC representa uma modificação do esquema LVQ original, no qual atributos adicionais são anexados a cada neurônio. Estes atributos armazenam informações estatísticas com relação aos treinos a que os neurônios são submetidos. O objetivo do LVQTC é classificar padrões de vetores $\mathbf{x}$ de acordo com os padrões de classes $c_1, c_2, c_3 \ldots c_n$. Estes atributos adicionais serão explorados ao longo dos processos de treinamento e de classificação. Durante o treinamento, estes parâmetros ajudam na substituição de neurônios com baixo desempenho no treinamento e na criação de novos neurônios quando necessário. Durante a classificação, estes atributos fornecem uma estimativa da confiança obtida por cada neurônio.

Neurônios são definidos pelo seguinte conjunto de atributos:

1. um vetor de referência $\mathbf{m}$ no espaço de padrões;
2. um rótulo da classe;
3. um contador para cada classe $p_1, p_2, p_3 \ldots p_n$, armazenando o número de vezes em que cada neurônio foi treinado por tipo de classes; e
4. um vetor $\mathbf{w}$ no espaço de padrões, representando a centroide dos padrões da classe errada (i.e., diferente da classe do neurônio) que treinou o neurônio.

Padrões usados para treinamento devem pertencer aos conjuntos de classes $c_1, c_2, c_3 \ldots c_n$ sem restrições particulares, tal como o número de padrões para cada classe. Na sequência, g_i e n_i denotarão a probabilidade global (ou *a priori*) e o número total de padrões de treinamento da classe c_i, respectivamente.

A inicialização do neurônio faz uso da informação disponível sobre a distribuição dos padrões de mesma classe dentro do conjunto de treinamento. O número inicial total de neurônios é tomado como uma pequena fração (5% ou menos) do total de padrões do conjunto de dados. A fim de limitar a ocorrência de neurônios que serão eventualmente treinados por padrões estatisticamente pobres. Os neurônios serão alocados às classes proporcionalmente ao volume e tamanho das distribuições no espaço de padrões. O volume da classe é estimado como a raiz quadrada do determinante da matriz de correlação correspondente (multiplicado pelo fator 2 por

cada variável do padrão) e o tamanho linear da classe é o dobro da raiz quadrada do determinante. A justificativa para isto é de simples visualização, considerando a matriz de correlação na forma diagonal. A matriz de referência de neurônios $\mathbf{m}$ é inicializada tomando de forma aleatória padrões de mesma classe do conjunto de padrões de treinamento.

O treinamento dos neurônios são organizados em uma sucessão de épocas. Para cada época, os seguintes passos são tomados:

1. zerar os neurônios: inicializar todos os contadores dos neurônios e vetores $\mathbf{w}$ com zero;

2. treinamento dos neurônios: apresentar todo o conjunto de treinamento de vetores rotulados $\mathbf{s}(t), t = 1, 2, 3 \ldots$, escolhendo uma amostra (vetor classe-rótulo), aleatoriamente, de cada vez. Para cada $\mathbf{s}(t)$ encontrar o neurônio $\mathbf{m}_c$ mais próximo ao vetor de referência. Digamos que o vetor de treinamento $\mathbf{s}(t)$ pertença à classe c_s, e supondo que $\mathbf{m}_c$ é rotulado de acordo com a classe c_s. Incrementar o contador de treinamento p_s do neurônio para a classe c_s em 1. Atualizar $\mathbf{m}_c$, a partir de outros vetores de referência de neurônios inalterados, de acordo com:

$$\mathbf{m}_c(t+1) = \mathbf{m}_c(t) + \left(\frac{\alpha_r}{p_{tot}}\right)[\mathbf{s}(t) - \mathbf{m}_c(t)] \text{ se } c_s = c_c \tag{2.27}$$

$$\mathbf{m}_c(t+1) = \mathbf{m}_c(t) - \left(\frac{\alpha_w}{p_{tot}}\right)[\mathbf{s}(t) - \mathbf{m}_c(t)] \text{ se } c_s \neq c_c \tag{2.28}$$

α_r e α_w são dois parâmetros distintos de aprendizagem. Devem-se tomar: $\alpha_r, \alpha_w < 1$ (Kohonen 1984). $p_{tot} = p_1 + p_2 + p_3 \ldots + p_n$ é a soma corrente sobre todo os contadores de treinamento do neurônio $\mathbf{m}_c$. α_r e α_w são monotonicamente decrescidos com o número de épocas: para cada época sucessiva ele será reduzido por um fator $f_r < 1$. Se $c_s \neq c_c$, atualizar o vetor $\mathbf{w}$ do neurônio $\mathbf{m}_c$;

3. poda de neurônios e criação: após a apresentação do conjunto de treinamento e antes do início de uma nova época, poda-se neurônios subtreinados (com treinamento pobre) e cria-se novos neurônios. Esta método tem o objetivo de reduzir a contaminação por padrões treinados de outras classes, conforme as seguintes regras:

4. poda de neurônios: eliminação de neurônios com

$$p_{tot} = p_1 + p_2 + \ldots + p_n < p_{prn} \tag{2.29}$$

onde p_{prn} é um parâmetro de corte modificável pelo usuário; e

5. criação de neurônios: denotaremos por p_w o máximo valor em que o contador de treinamento pode assumir sendo treinado por um padrão de rótulo diferente; se $p_w > p_{prn}$, criar um neurônio de uma classe com o contador de treinamento $p_i = p_w$ e com um vetor $\mathbf{m}$ igual ao vetor do neurônio original.

O treinamento deve parar quando o número de classificações sobre o conjunto de treinamento não apresentar mais uma melhora considerável. Para os cálculos relatados aqui, o treinamento deve parar quando por três épocas consecutivas o número de classificações corretas sobre o conjunto de treinamento não for maior que $(1+f_{cvg})$ vezes o número de classificações obtidas na época anterior, onde $0 < f_{cvg} < 1$.

Ao término da última época, não haverá poda ou criação de neurônios. Contadores de treinamento são recalculados através da leitura de todo o cunjunto de padrões, mantendo os neurônios imutáveis. Ao mesmo tempo, é calculada a contaminação do neurônio através do seu treinamento por padrões de outras classes, f_{cont}, e o raio do neurônio, r_{neu}, como se segue.

f_{cont} é definido por:

$$f_{cont} = \frac{a_x}{a_{tot}} \tag{2.30}$$

onde

$$a_{tot} = a_1 + a_2 + a_3 + \ldots + a_n \tag{2.31}$$

com

$$a_i = g_i p_i n_i \tag{2.32}$$

g_i e n_i sendo a probabilidade global e o número total de padrões de treinamentos da classe c_i, respectivamente; p_i sendo o contador de treinamento para a classe c_i do neurônio; e

$$a_x = \text{ soma sobre todos } a_i \text{ exceto } a_k \tag{2.33}$$

k indicando a classe c_k com os rótulos do neurônio.

r_{neu} é calculado como a raiz quadrada da média dos quadrados (r.m.s.) da distância do neurônio para todos os padrões de treinamento mais próximos e que pertençam ao mesmo rótulo (classe).

Os parâmetros f_{cont} e r_{neu} do neurônio são usados, juntamente com o p_{tot}, no processo de classificação.

A classificação do vetor de padrões $\mathbf{s}$ de uma classe desconhecida é realizada por meio da atribuição do vetor a classe do neurônio mais próximo. Uma estimativa da incerteza da classificação é fornecida por intermédio da contaminação do neurônio f_{cont}, introduzido acima. A classificação também pode ser considerada incerta se o neurônio tiver um valor do p_{tot} muito baixo (i.e., uma estatística de treinamento pobre) ou se estiver muito distante do $\mathbf{s}$. Então, a classificação pode ser considerada incerta se para o neurônio mais próximo se houver:

$$f_{cont} > f_{cmax} \tag{2.34}$$

$$p_{tot} < p_{min} \tag{2.35}$$

$$d > d_{max} \tag{2.36}$$

onde f_{cmax}, p_{min} e d_{max} são parâmetros de corte definidos pelo usuário; d é a distância do neurônio $\mathbf{s}$ medido em unidades de r_{neu}. Os parâmetros de corte f_{cmax}, p_{min} e d_{max} necessitam ser selecionados de acordo com o nível de pureza que se deseja alcançar no processo de classificação. A decisão por uma configuração de alto nível de pureza acarreta em perdas na eficiência da classificação.

2.3.8 Algoritmo *Soft* LVQ

Os classificadores pertencentes à família de métodos LVQ funciona bem em muitas tarefas de classificação (HELSINKI, 2002), mas as regras de seleção (aprendizado) possuem a desvantagem de serem baseadas em uma heurística, podendo não produzir uma solução ótima.

A fim de melhorar o desempenho na classificação, foi proposta uma função de custo para o modelo de seleção. Katagiri *et al.* (JUANG; KATAGIRI, 1992), (KATAGIRI, 1991), (KOMORI; KATAGIRI, 1992), (MCDERMOTT; KATAGIRI, 1994) investigou as funções de custo, as quais derivavam de um processo de duas

fases. Primeiramente uma família de funções discriminantes parametrizadas é construída e, em seguida, uma medida de desempenho é definida. Esta medida de desempenho é relacionada a uma taxa de erro de classificação que conduz a uma perda individual para toda amostra de dado. O modelo de seleção funciona por meio de um gradiente descendente (estocástico) da perda total sobre o conjunto de treinamento.Uma função contínua de perda foi utilizada a fim de aplicar uma otimização baseada em gradiente e diversos hiperparâmetros foram introduzidos com este propósito.

Neste documento, é apresentado o algoritmo do classificador neural SLVQ (SEO, 2003), sendo uma extensão probabilística do classificador LVQ. Em sua concepção básica, a rede SLVQ é uma rede neural competitiva supervisionada que apresenta apenas uma única camada de neurônios e, consequentemente, uma camada de pesos sinápticos a serem ajustados. O vetor de pesos associado a cada neurônio é também chamado de protótipo e cada protótipo também está associado um rótulo (*label*) da classe a qual tal protótipo representa. O algoritmo SLVQ é um classificador do tipo protótipo mais próximo (*nearest prototype*) com desempenho de classificação ótimo segundo uma função objetivo devidamente construída para este fim.

Mais formalmente, considere um conjunto de dados $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \mathcal{I}$, em que n é o número de padrões de treinamento e $\mathcal{I}$ é o conjunto de rótulos das classes existentes. Assim, devemos selecionar um conjunto $\mathcal{T} = \{(\mathbf{m}_j, c_j)\}_{j=1}^{n_m}$ de protótipos devidamente rotulados, $\mathbf{m}_j \in \mathbb{R}^p$, $c_j \in \mathcal{I}$, em que n_m é o número de protótipos.

O conjunto de protótipos $\mathcal{M} = \{\mathbf{m}_j\}_{j=1}^{n_m}$ é determinado minimizando-se a seguinte função custo:

$$J(\mathcal{S}, \mathcal{T}) = \frac{1}{N} \sum_{k=1}^{N} ls((\mathbf{x}_k, y_k), \mathcal{T}), \tag{2.37}$$

tal que

$$ls((\mathbf{x}_t, y_t), \mathcal{T}) = \frac{p(\mathbf{x}_t, \bar{y}_t|\mathcal{T})}{p(\mathbf{x}_t|\mathcal{T})}. \tag{2.38}$$

é importante notar que $p(\mathbf{x}_t, \bar{y}_t|\mathcal{T})$ é a densidade de probabilidade conjunta do padrão $\mathbf{x}_t$ com um rótulo diferente de y_t. Em outras palavras, trata-se de uma maneira de quantificar a chance do padrão $\mathbf{x}_t$ ser gerado (ou ocorrer) com um rótulo

errado. Matematicamente, temos que

$$p(\mathbf{x}_t, \bar{y}_t|\mathcal{T}) = \sum_{\{j:c_j \neq y_t\}} p(\mathbf{x}_t|j)p(j), \tag{2.39}$$

em que $p(\mathbf{x}_t|j)$ é a probabilidade condicional de o padrão $\mathbf{x}_t$ ser gerado pelo j-ésimo protótipo, e $p(j)$ é a probabilidade *a priori* do j-ésimo protótipo (ou seja, a chance de o j-ésimo protótipo ocorrer, independente dos dados). Normalmente, consider-se uma distribuição uniforme para os protótipos; logo, $p(j) = 1/n_m$.

Assumindo que os padrões gerados pelo j-ésimo protótipo seguem uma distribuição normal com centróide $\mathbf{m}_j \in \mathbb{R}^p$ e matriz de covariância $\sigma^2\mathbf{I}_p$, em que $\mathbf{I}_p$ é a matriz identidade de dimensões $p \times p$, temos que

$$p(\mathbf{x}_t|j) = \frac{1}{(2\pi\sigma^2)^{p/2}} \exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{m}_j\|^2}{2\sigma^2}\right\}. \tag{2.40}$$

Com isso, a Equação (2.38) passa a ser escrita como

$$\begin{aligned} ls_t = ls((\mathbf{x}_t, y_t), \mathcal{T}) &= \frac{p(\mathbf{x}_t, \bar{y}_t|\mathcal{T})}{p(\mathbf{x}_t|\mathcal{T})}, \\ &= \frac{\sum_{\{j:c_j \neq y_t\}} \exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{m}_j\|^2}{2\sigma^2}\right\}}{\sum_{k=1}^{n_m} \exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{m}_j\|^2}{2\sigma^2}\right\}} \\ &= \sum_{\{j:c_j \neq y_t\}} P(j|\mathbf{x}_t), \end{aligned} \tag{2.41}$$

em que

$$P(j|\mathbf{x}_t) = \frac{p(\mathbf{x}_t|j)p(j)}{p(\mathbf{x}_t)} = \frac{\exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{m}_j\|^2}{2\sigma^2}\right\}}{\sum_{k=1}^{m} \exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{m}_j\|^2}{2\sigma^2}\right\}}, \tag{2.42}$$

é a probabilidade *a posteriori* de que o padrão $\mathbf{x}_t$ foi gerado pelo j-ésimo protótipo.

A aplicação do método do gradiente descedente estocástico à função custo da Equação (2.41) resulta na seguinte regra de atualização dos protótipos:

$$\mathbf{m}_l(t+1) = \mathbf{m}_l(t) - \alpha^*(t)\frac{\partial ls_t}{\partial \mathbf{m}_l}, \quad l = 1, \ldots, n_m, \tag{2.43}$$

da qual derivamos dois casos:

Caso 1 ($c_l = y_t$): Rótulo do l-ésimo protótipo (c_l) é igual ao rótulo do padrão atual

(y_t).

$$\mathbf{m}_l(t+1) = \mathbf{m}_l(t) - \alpha^*(t)P(l|\mathbf{x}_t)ls_t(\mathbf{x}_t - \mathbf{m}_l) \tag{2.44}$$

Caso 2 ($c_l \neq y_t$): Rótulo do l-ésimo protótipo (c_l) difere do rótulo do padrão atual (y_t).

$$\mathbf{m}_l(t+1) = \mathbf{m}_l(t) + \alpha^*(t)P(l|\mathbf{x}_t)(1 - ls_t)(\mathbf{x}_t - \mathbf{m}_l) \tag{2.45}$$

com $\alpha^*(t) = \alpha(t)/\sigma^2(t')$, em que t e t' são índices temporais distintos cujos papéis ficarão mais claros mais adiante quando apresentarmos o pseudocódigo do algoritmo.

É importante notar que, em contraste com os algoritmos da família LVQ, em que apenas os dois protótipos vencedores (o da classe correta e o da classe incorreta) são atualizados, no algoritmo SLVQ todos os protótipos com rótulos corretos são atraídos para o padrão atual $\mathbf{x}_t$ proporcionalmente às distâncias deste e ponderados pelo fator $P(l|\mathbf{x}_t)ls_t$, enquanto todos os protótipos com rótulos incorretos são afastados do padrão atual $\mathbf{x}_t$ também proporcionalmente às distâncias deste mas ponderados pelo fator $P(l|\mathbf{x}_t)(1 - ls_t)$.

O parâmetro σ^2 é um hiperparâmetro das regras de aprendizagem mostradas nas Equações (2.44) e (2.45). Este parâmetro assume um valor alto inicialmente e é gradualmente reduzido com o passar das iterações até um valor final σ_f. Um resumo do algoritmo SLVQ é apresentado na próxima subseção. A Figura 2.4 ilustra o tamanho da região ativa bidimensional representando um problema simples de teste com quatro protótipos (símbolos brancos) e quatro classes, com valores diferentes do parâmetro de dispersão σ^2. Valores cinzas indicam a intensidade do fator $ls(1 - ls)$ para as amostras de dados que possuem rótulos iguais aos seus protótipos mais próximos.

A Figura 2.4 mostra que a largura de uma área ativa cresce quando aumenta-se σ^2 e o valor de $ls(1-ls)$ diminui com o aumento da distância a partir dos limitantes correntes da classe. Para que se acelere o processo de aprendizado, primeiro precisa-se definir o valor de corte $0 < \eta \ll 0.25$ para que o protótipo seja atualizado e mudado apenas se $ls(1 - ls) > \eta$. O surgimento de uma zona de janela para a atualização de protótipos também pode ser explorado por estratégias ativas de aprendizagem. Apenas as amostras de dados que se enquadram na região da janela devem ser rotuladas.

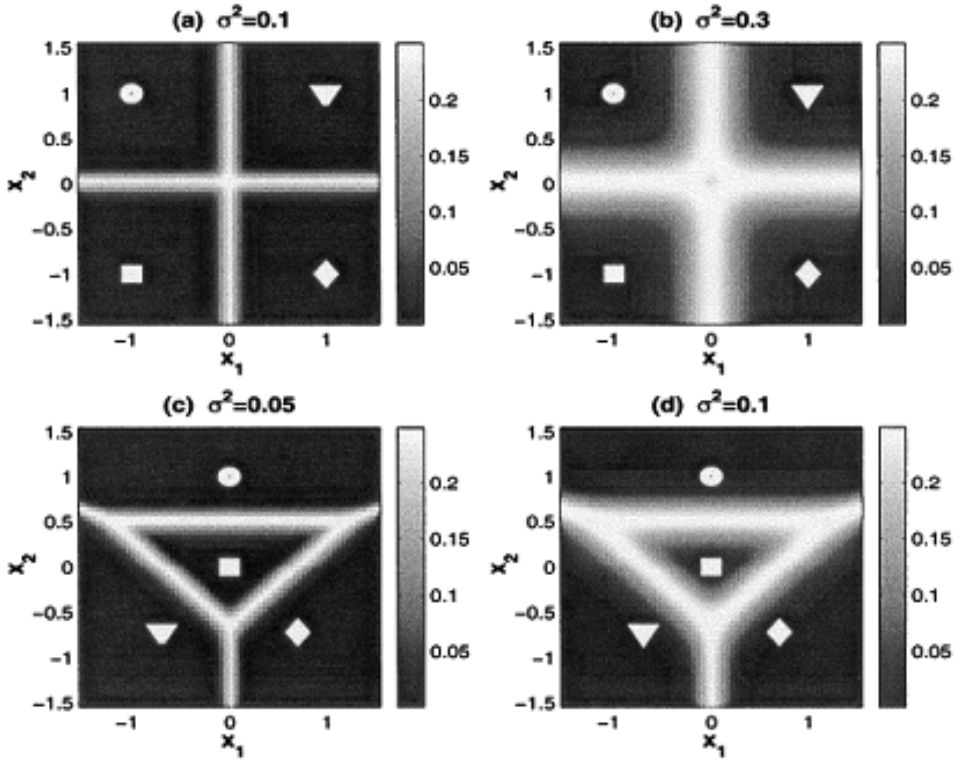


Figura 2.4: Ilustra as áreas ativas para os diferentes valores do parâmetro de dispersão σ^2 para duas configurações de protótipos.

Algoritmo resumido do SLVQ

1. Inicialização - Especificar valores para os seguintes parâmetros de treinamento:

(a) Número de protótipos por classe.

(b) Selecionar os protótipos iniciais e respectivos rótulos.

(c) $\sigma^2(0)$: valor inicial do parâmetro de dispersão (e.g. 1,0).

(d) σ_f^2: valor final do parâmetro de dispersão (e.g. 0,001).

(e) $\rho(0)$: valor inicial do termo usado no decaimento de $\sigma^2(t)$ (e.g. 0,99).

(f) γ: expoente da equação de atualização do parâmetro ρ (e.g. 1,1).

2. Enquanto $(\sigma^2(t) > \sigma_f^2)$ **Faça**

(2.1) Defina a taxa de aprendizagem inicial $0 < \alpha_0 < 1$.

(2.2) Faça $t = t' = 0$.

(2.3) **Repetir os passos abaixo até que um critério de parada seja satisfeito.**

(i) Selecione aleatoriamente um par $(\mathbf{x}_t, y_t)$.

(ii) Calcule as probabilidades de atribuição:

$$P(j|\mathbf{x}_t) = \frac{\exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{w}_j\|^2}{2\sigma^2}\right\}}{\sum_{k=1}^{n_m} \exp\left\{-\frac{\|\mathbf{x}_t - \mathbf{w}_j\|^2}{2\sigma^2}\right\}}, \quad j = 1, \ldots, n_m. \tag{2.46}$$

(iii) Calcule o valor de ls_t correspondente:

$$ls_t = \sum_{\{j: c_j \neq y_t\}} P(j|\mathbf{x}_t) \tag{2.47}$$

(iv) Atualize os protótipos usando as Equações (2.44) e (2.45).

(v) Atualize a taxa de aprendizagem usando a seguinte equação:

$$\alpha(t+1) = \frac{\alpha_0}{1+t} \tag{2.48}$$

(vi) Faça $t = t + 1$.

(2.4) Atualizar o parâmetro $\sigma^2(t')$:

$$\sigma^2(t'+1) = \rho(t')\sigma^2(t') \tag{2.49}$$

(2.5) Atualizar o parâmetro $\rho(t')$:

$$\rho(t'+1) = \rho(t')^\gamma \tag{2.50}$$

(2.6) Faça $t' = t' + 1$.

2.4 Classificadores Baseados na Rede SOM

Introduzida por Kohonen (1982), o conceito de mapas auto-organizáveis foi proposto a partir da observação que mapas corticais se formam de maneira adaptativa e automática (KOHONEN, 1997). Por mapas corticais entende-se o

mapeamento topográfico de regiões no cérebro responsáveis por recepções sensoriais específicas.

A rede SOM tem então como objetivo principal o mapeamento (ou projeção) de um espaço contínuo de dimensão possivelmente elevada em um espaço discreto de dimensão reduzida. A projeção resultante consiste de N_w neurônios dispostos em um arranjo geométrico fixo, de dimensão S. Comumente usa-se o valor $S = 2$, correspondendo a um mapa bidimensional.

Matematicamente, seja um espaço de entrada contínuo $\mathcal{X} \subset \mathbb{R}^D$ e um espaço discreto $\mathcal{Y} \subset \mathbb{R}^S$ formado por N_w vetores. Um dado vetor $\mathbf{x} \in \mathcal{X}$ será representado na rede por um vetor $\mathbf{y}_{i^*} \in \mathcal{Y}$ através do mapeamento $i^*(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$.

A preservação aproximada da topologia dos dados de entrada da rede SOM garante que os vetores de entrada próximos entre si sejam mapeados em vetores próximos no espaço discreto da rede. Essa propriedade é especialmente interessante em aplicações envolvendo visualização de dados de alta dimensão.

A rede SOM, assim como a rede Fuzzy ART, é uma rede neural competitiva que pode ser usada tanto em tarefas de agrupamento de dados quanto de quantização vetorial. A primeira consiste em separar ou encontrar grupos de vetores similares segundo um critério de similaridade. A segunda é a tarefa de substituir um conjunto de N vetores por um conjunto de N_w protótipos, em que $N_w \ll N$. Note que nem todo algoritmo de agrupamento de dados realiza quantização vetorial, mas todo algoritmo de quantização vetorial faz (ou pode fazer) análise de agrupamento.

2.4.1 Arquitetura Geral

A arquitetura de uma rede SOM encontra-se ilustrada na Figura 2.5. Pode-se perceber que todos os N_w neurônios recebem o padrão de entrada $\mathbf{x}(n) \in \mathbb{R}^D$ simultaneamente. Os atributos contidos em $\mathbf{x}(n)$ são ponderados pelo i-ésimo neurônio por um vetor de pesos $\mathbf{w}_i(n) \in \mathbb{R}^D$.

Reunindo os N exemplos disponíveis para o processo de aprendizagem da rede, obtém-se a matriz $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_N]^T$, formando o conjunto de dados de treinamento. De maneira semelhante, a reunião dos vetores de pesos em colunas resulta na matriz $\mathbf{W}(n) = [\mathbf{w}_1(n) \ \mathbf{w}_2(n) \ \cdots \ \mathbf{w}_{N_w}(n)]^T$, que representa a rede SOM.

O mapeamento realizado pela rede SOM pode ser visto como um processo de codificação da matriz de vetores de entrada $\mathbf{X}$ em que o dicionário (*codebook*) é

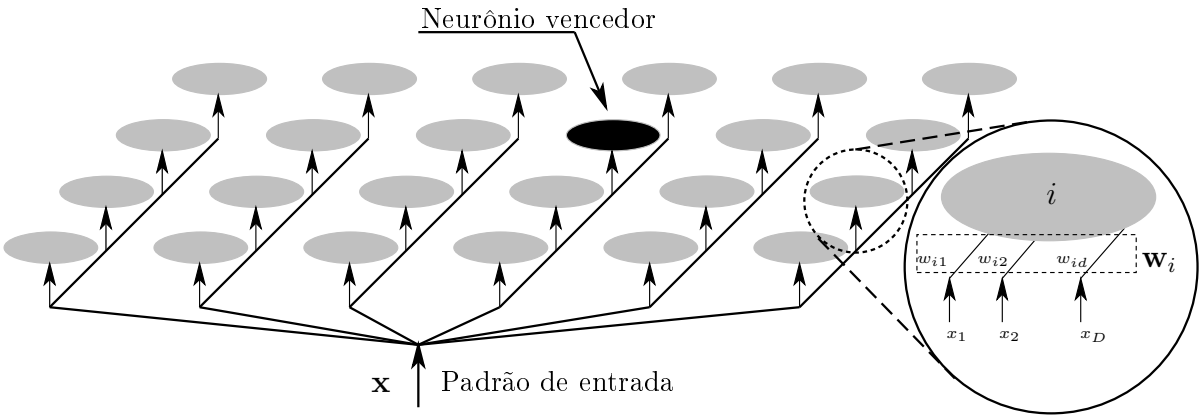


Figura 2.5: Exemplo de rede SOM bidimensional. Os vetores de entrada e de pesos são D-dimensionais. Os N_w neurônios estão uniformemente dispostos em uma grade retangular. Modificado de Aguayo (2008).

formado pelas colunas da matriz de pesos $\mathbf{W}$. A Figura 2.6 reforça essa interpretação ao ilustrar o mapeamento entre os espaço de entrada e saída.

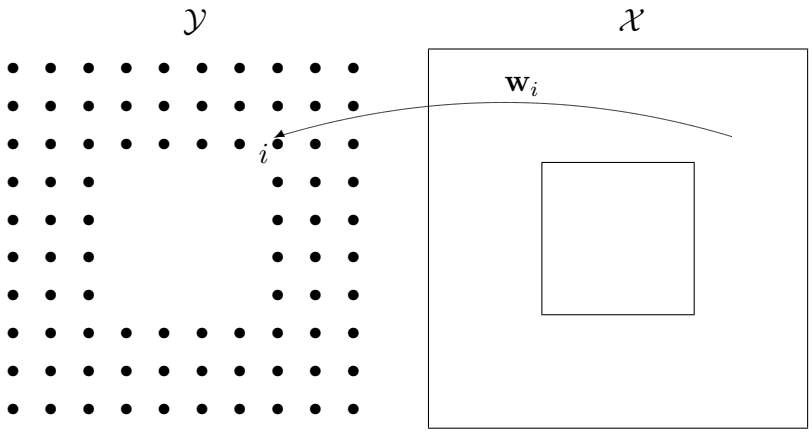


Figura 2.6: Mapeamento entre os espaços $\mathcal{X}$ e $\mathcal{Y}$ realizado pela rede SOM. Modificado de Aguayo (2008).

2.4.2 Treinamento da rede SOM

No início do treinamento da rede SOM, seus N_w neurônios são dispostos de forma regular em uma malha de dimensão $P_1 \times P_2$, considerando um mapa bidimensional. Nesse momento os vetores de pesos possuem valores aleatórios, normalmente pequenos. Para cada vetor apresentado à rede na iteração n do algoritmo de treinamento, são realizadas as etapas de processamento a seguir.

Processo de competição. Inicialmente verifica-se qual o neurônio i^* é o mais próximo da entrada $\mathbf{x}(n)$ pela expressão

$$i^*(n) = \arg\min_{\forall i} \|\mathbf{x}(n) - \mathbf{w}_i(n)\|, \tag{2.51}$$

em que $\|\cdot\|$ denota o cálculo da distância euclidiana. A métrica de dissimilaridade pode ser outra, mas a distância euclidiana é a escolha mais comum.

Processo de cooperação. O vetor de pesos associado ao neurônio vencedor $i^*(n)$, assim como aos seus neurônios vizinhos, são simultaneamente atualizados pela seguinte regra de aprendizagem:

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \eta(n)h(i^*, i; n)[\mathbf{x}(n) - \mathbf{w}_i(n)], \tag{2.52}$$

em que $0 < \eta(n) \leq 1$ corresponde ao valor do parâmetro de aprendizagem na iteração n do algoritmo e a função $h(i^*, i; n)$ é chamada *função de vizinhança*. Esta função define a vizinhança de influência do neurônio vencedor ao determinar quais neurônios terão seus pesos atualizados de modo mais intenso. Uma escolha comum para $h(i^*, i; n)$ é a função gaussiana:

$$h(i^*, i; n) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{i^*}\|^2}{2\sigma^2(n)}\right), \tag{2.53}$$

em que $\mathbf{r}_i$ e $\mathbf{r}_{i^*}$ são, respectivamente, as coordenadas dos neurônios i e i^* na grade de saída. O parâmetro $\sigma(n) > 0$ refere-se à largura da vizinhança considerada: quanto maior seu valor, maior o número de neurônios atualizados em torno do neurônio vencedor.

A Equação (2.52) pode ser reescrita ao substituir $\eta^*(n) = \eta(n)h(i^*, i; n)$, desta forma:

$$\mathbf{w}_i(n+1) = (1 - \eta^*(n))\mathbf{w}_i(n) + \eta^*(n)\mathbf{x}(n). \tag{2.54}$$

Como a equação de atualização dos pesos, Equação (2.52) ou Equação (2.54), depende da proximidade dos neurônios em relação ao neurônio vencedor, existe a tendência do surgimento de regiões específicas na rede SOM sensíveis a determinadas variações nos padrões de entrada. Essa característica constitui a já citada capacidade da rede SOM de preservar, de forma aproximada, a topologia dos dados após o treinamento.

O processo de treinamento da rede SOM encontra-se no Algoritmo 2.1.

2.4.3 Sobre a convergência da rede SOM

Para garantir a convergência dos pesos da rede SOM a valores estáveis durante o algoritmo de treinamento, é preciso reduzir o passo de aprendizado e o parâmetro de espalhamento ao longo das iterações do método (RITTER; SCHULTEN, 1988). Esse precedimento tem como objetivo reduzir gradualmente a influência dos pesos iniciais.

Nesta dissertação, optou-se pelo decaimento exponencial dos parâmetros $\eta(n)$ e $\sigma(n)$:

$$\eta(n) = \eta_0 \left(\frac{\eta_f}{\eta_0}\right)^{(n/n_{\text{MáX}})}, \tag{2.55}$$

$$\sigma(n) = \sigma_0 \left(\frac{\sigma_f}{\sigma_0}\right)^{(n/n_{\text{MáX}})}, \tag{2.56}$$

em que $n_{\text{MáX}}$ é o total de iterações de treinamento, $\eta(1) = \eta_0$, $\eta(n_{\text{MáX}}) = \eta_f$, $\sigma(1) = \sigma_0$ e $\sigma(n_{\text{MáX}}) = \sigma_f$. Os valores iniciais η_0 e σ_0, assim como os valores finais η_f e σ_f, constituem parâmetros a serem especificados para cada problema estudado.

A Figura 2.7 apresenta exemplos de decaimento do parâmetro η para diferentes valores de η_f, considerando-se um valor fixo $\eta_0 = 0{,}9$. Note que as curvas para o parâmetro σ seriam semelhantes.

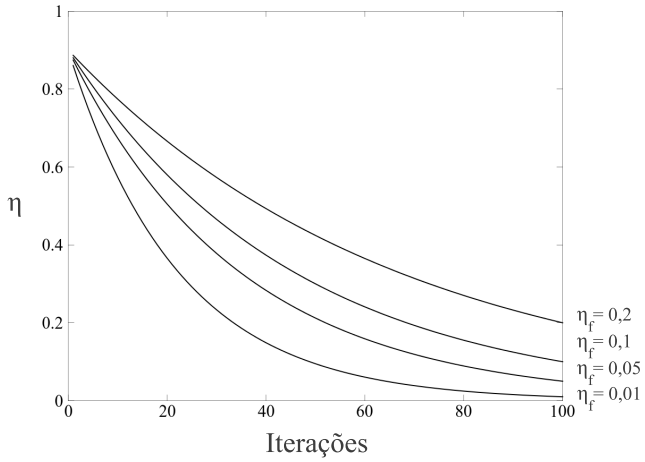


Figura 2.7: Exemplos de decaimento do parâmetro η da rede SOM.

A Figura 2.8 ilustra um exemplo de uma rede SOM durante a etapa de

Algoritmo 2.1 Algoritmo de treinamento da rede SOM.

Constantes

N_w: número de neurônios da rede

η_0: valor inicial do parâmetro de aprendizado η

σ_0: valor inicial do parâmetro de espalhamento σ

D: dimensão de entrada

$D_1 e D_2$: dimensões do mapa

$n_{\text{MÁX}}$: número de iterações de treinamento

Entradas

$\mathbf{x}(n)$: vetor de entrada, dimensão D

Algoritmo

1. Inicialização

Iniciar os pesos $\mathbf{w}_i(0)$ com valores pequenos aleatórios $(i = 1, 2, \ldots, N_w)$

Fazer $\eta(1) = \eta_0$ e $\sigma(1) = \sigma_0$

2. Laço temporal $(n = 1, 2, \ldots, n_{\text{MÁX}})$

2.1 Selecionar $\mathbf{x}(n)$ do conjunto de vetores de entrada

2.2 Armazenar o índice do neurônio vencedor

$$i^*(n) = \arg\min_{\forall i} \|\mathbf{x}(n) - \mathbf{w}_i(n)\|$$

2.3 Atualizar os pesos do vencedor e da vizinhança

Para $i = 1, 2, \ldots, N_w$, calcular

$$h(i^*, i; n) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{i^*}\|^2}{2\sigma^2(n)}\right)$$

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \eta(n) h(i^*, i; n)[\mathbf{x}(n) - \mathbf{w}_i(n)]$$

Decair os parâmetros $\eta(n)$ e $\sigma(n)$

Saídas

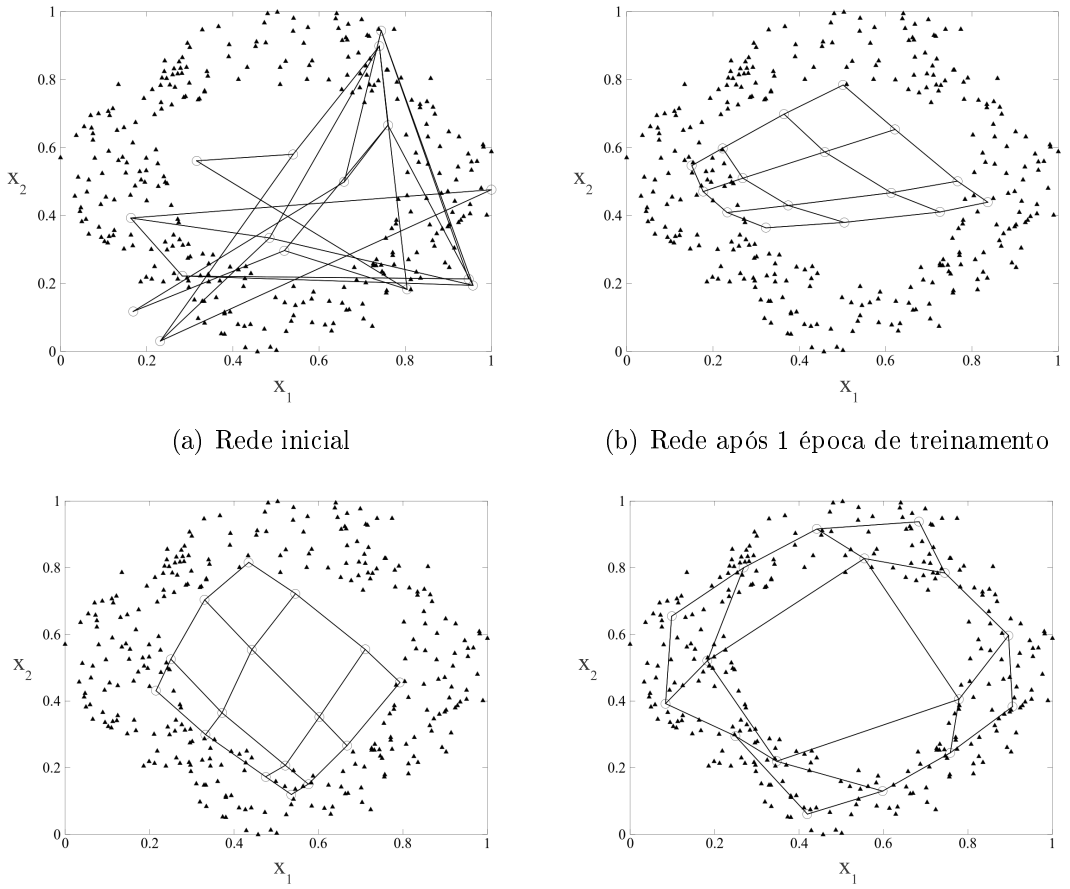
Saída a cada iteração: coordenada $\mathbf{r}_{i*(n)}$ do neurônio vencedor e seu vetor de pesos $\mathbf{w}_{i*(n)}$

Resultado do treinamento: $\mathbf{W}(n_{\text{MÁX}})$: matriz dos pesos dos neurônios (dimensão $D \times N_w$)

Observações

Durante o uso da rede SOM, pós-treinamento, o Passo 2.3 não é necessário

treinamento. Nesse exemplo, tem-se um conjunto de dados bidimensionais, em que estes dados devem ser mapeados pela rede. Percebe-se que, apesar da quantidade de neurônios ser menor que a quantidade de amostras, ao longo das épocas[1] a rede SOM é capaz de obter uma representação condensada dos dados treinados.



(a) Rede inicial

(b) Rede após 1 época de treinamento

(c) Rede após 10 épocas de treinamento

(d) Rede após 100 épocas de treinamento

Figura 2.8: Efeito do treinamento da rede SOM nos pesos dos neurônios.

A Figura 2.9 apresenta a evolução do erro de quantização médio, calculado para a k-ésima época pela Equação (2.57).

$$eqm(k) = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}(n) - \mathbf{w}_{i^*}(n)\|^2, \tag{2.57}$$

em que N é o número de vetores de treinamento. A convergência do erro de quantização médio, conforme ilustrado na Figura 2.9, indica que o mapeamento final é de fato satisfatório.

[1]Uma época consiste de uma apresentação de todos os vetores de treinamento à rede.

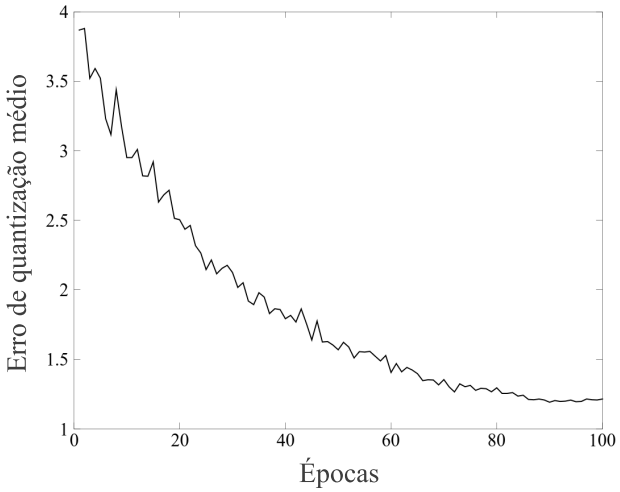


Figura 2.9: Exemplo de convergência do erro médio de quantização durante o treinamento da rede SOM.

2.5 Rede Fuzzy ARTMAP

A rede Fuzzy ARTMAP foi proposta em Carpenter *et al.* (1992) como uma variante da rede ARTMAP (CARPENTER; GROSSBERG; REYNOLDS, 1991) que utiliza os operadores fuzzy da rede Fuzzy ART. Desde então o algoritmo Fuzzy ARTMAP tem sido o mais popular representante da família ART para problemas de aprendizagem supervisionada.

O arquitetura original da rede Fuzzy ARTMAP envolve o treinamento simultâneo de dois módulos Fuzzy ART, sendo cada um deles responsável por associar dois espaços vetoriais distintos, porém relacionados. Em problemas de classificação de padrões um dos espaços é o espaço dos rótulos, enquanto o outro é o espaço de entrada (RAJASEKARAN; PAI, 2000; PALANIAPPAN; ESWARAN, 2009). Em Kasuba (1993) foi realizada uma simplificação na notação da rede Fuzzy ARTMAP, através da redução de redundâncias na arquitetura original. Essa versão é normalmente chamada de *Simplified Fuzzy ARTMAP* (SFAM) e será a utilizada nesta dissertação. Mesmo assim, a denominação Fuzzy ARTMAP será mantida.

2.5.1 Arquitetura da rede Fuzzy ARTMAP

A rede Fuzzy ARTMAP usa dois módulos Fuzzy ART, denotados ART_a e ART_b, interligados por uma matriz de pesos. Cada módulo Fuzzy ART possui duas camadas principais, F_1 e F_2. Entretanto, para fins de classificação de padrões, o módulo

ART_b da rede Fuzzy ARTMAP pode ser simplificado em uma única camada F_2^b (CARPENTER, 2003).

A Figura 2.10 ilustra a arquitetura geral da rede Fuzzy ARTMAP. A seguir são detalhados os principais componentes da arquitetura.

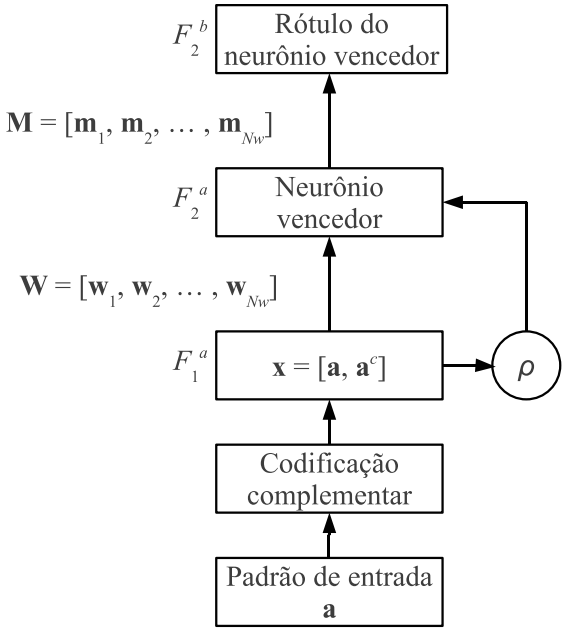


Figura 2.10: Diagrama de blocos da rede Fuzzy ARTMAP. Os índices temporais foram removidos para melhor visualização.

Sinal de entrada. A entrada da rede Fuzzy ARTMAP constitui N vetores $\mathbf{a}(n) \in \mathbb{R}^P$, normalmente transformados via codificação complementar em vetores $\mathbf{x}(n) \in \mathbb{R}^D$, $n = 1, 2, \ldots, N$, em que $D = 2P$. Assim como na rede Fuzzy ART, as D componentes dos vetores de entrada possuem valores limitados entre 0 e 1. Os vetores $\mathbf{x}(n)$ constituem entrada da camada F_1^a do módulo ART_a. Por ser um algoritmo supervisionado, a rede também recebe como entrada os N rótulos das classes associadas aos N padrões de treinamento. Esses rótulos são representados na forma vetorial $\mathbf{y}(n) \in \mathbb{R}^C$, $n = 1, 2, \ldots, N$, em que C é o número de classes possíveis.

Módulo ART_a. O módulo ART_a possui duas camadas, F_1^a e F_2^a. Os vetores de entrada são apresentados à rede pela primeira. A segunda camada é formada pelos protótipos criados ao longo do treinamento. O i-ésimo neurônio corresponde ao vetor de pesos $\mathbf{w}_i(n) \in \mathbb{R}^D$ na iteração n da fase de treino.

Módulo ART_b. O módulo ART_b se resume à camada F_2^b, onde é definida a classe do padrão de entrada atual.

Matriz Inter-MAP. Entre as camadas F_2^a e F_2^b existe uma matriz de pesos $\mathbf{M}$, chamada Inter-MAP, que associa aos protótipos de F_2^a uma classe na camada F_2^b. Seja N_w o número de protótipos em F_2^a na iteração n do treinamento, a matriz Inter-MAP $\mathbf{M}(n) = [\mathbf{m}_1(n)\ \mathbf{m}_2(n)\ \cdots\ \mathbf{m}_{N_w}(n)]$ possui dimensão $C \times N_w$. As colunas da matriz $\mathbf{M}(n)$ determinam na camada F_2^b a classe do padrão de entrada, possuindo construção idêntica aos vetores de rótulo de entrada $\mathbf{y}(n)$, ou seja, apenas uma componente do vetor $\mathbf{m}_i(n)$ é igual a 1, enquanto as outras são iguais a zero.

2.5.2 Treinamento da rede Fuzzy ARTMAP

Para cada vetor apresentado á rede Fuzzy ARTMAP, o seu algoritmo de treinamento deve obedecer às seguintes etapas de processamento.

Codificação da entrada. O padrão de entrada $\mathbf{a}(n) \in \mathbb{R}^P$ deve ser codificado em um vetor $\mathbf{x}(n) \in \mathbb{R}^{2P}$. Isso é feito através do processo de codificação complementar

$$\mathbf{x}(n) = \begin{bmatrix} \mathbf{a}(n) \\ \mathbf{a}^c(n) \end{bmatrix} = \begin{bmatrix} \mathbf{a}(n) \\ \mathbf{1}^P - \mathbf{a}(n) \end{bmatrix}, \tag{2.58}$$

em que $\mathbf{1}^P$ é um vetor de dimensão P contendo somente elementos iguais a 1.

Como até agora denotou-se a dimensão do vetor $\mathbf{x}(n)$ por D, a partir de agora tem-se $D = 2P$.

Processo de competição. Apresenta-se o vetor $\mathbf{x}(n)$ à primeira camada da rede, F_1, e, para cada um dos N_w neurônios, calcula-se a i-ésima ativação, que pode ser entendida como o nível de ressonância do protótipo:

$$t_i(n) = \frac{|\mathbf{x}(n) \wedge \mathbf{w}_i(n)|}{\beta + |\mathbf{w}_i(n)|}, \quad i = 1, 2, \ldots, N_w, \tag{2.59}$$

em que o operador $\wedge$ representa a operação de conjugação *fuzzy*, elemento a elemento, ou seja

$$x_j(n) \wedge w_{ij} \equiv \min\{x_j(n), w_{ij}(n)\}, \tag{2.60}$$

e que $|\mathbf{x}| = \sum_{j=1}^{D} |x_j|$ é a norma L_1 do vetor $\mathbf{x}$. O parâmetro β funciona como um fator de escala positivo para a ativação calculada. Por fim, busca-se pelo índice do neurônio vencedor i^* na iteração n

$$i^*(n) = \arg\max_i \{t_i(n)\}, \quad i = 1, 2, \ldots, N_w. \tag{2.61}$$

Critério de vigilância. Verifica-se se o neurônio vencedor i^* satisfaz o critério de vigilância por meio do seguinte teste:

$$\frac{|\mathbf{x}(n) \wedge \mathbf{w}_{i^*}(n)|}{|\mathbf{x}(n)|} \geq \rho, \tag{2.62}$$

em que $0 < \rho < 1$ é o parâmetro de vigilância. Se o teste de vigilância é satisfeito, segue-se para a etapa seguinte, a de atualização dos pesos. Caso contrário, a ativação do neurônio i^* recebe o valor zero ($t_{i^*}(n) = 0$) e a busca por um novo neurônio é reiniciada usando-se a Equação (2.61). Esse processo de competição em que se verifica o grau de casamento (*matching*) entre o vetor de entrada e os protótipos da rede Fuzzy ART é chamado de *ressonância*.

Critério de predição. Esse novo teste consiste em verificar se o vetor $\mathbf{m}_{i^*}(n)$ prediz exatamente a saída desejada $\mathbf{y}(n)$ para a entrada atual, $\mathbf{x}(n)$. Se o teste falhar, o valor do parâmetro de vigilância é modificado através da adição de uma pequena constante $\epsilon \to 0$, ou seja,

$$\rho = \frac{|\mathbf{x}(n) \wedge \mathbf{w}_{i^*}(n)|}{|\mathbf{x}(n)|} + \epsilon. \tag{2.63}$$

Os testes de vigilância e de predição são repetidos até que um neurônio vencedor passe em ambos os testes ou todos os neurônios da rede tenham sido testados, caso em que um neurônio ainda não usado (*uncommitted*) é escolhido como vencedor.

Atualização dos pesos. Caso um neurônio já usado tenha sido escolhido como vencedor, os seus pesos são atualizados pela seguinte equação:

$$\mathbf{w}_{i*}(n+1) = \eta\left(\mathbf{w}_{i^*}(n) \wedge \mathbf{x}(n)\right) + (1-\eta)\mathbf{w}_{i^*}(n). \tag{2.64}$$

Caso o neurônio vencedor não tenha ainda sido usado (i.e. $\mathbf{w}_{i^*}(n) = \mathbf{1}^D$), o

mesmo recebe o vetor de entrada atual:

$$\mathbf{w}_{i^*}(n+1) = \mathbf{x}(n), \tag{2.65}$$

sua coluna correspondente em $\mathbf{M}(n)$ é atualizada, ou seja,

$$\mathbf{m}_{i^*}(n+1) = \mathbf{y}(n), \tag{2.66}$$

e um novo protótipo é adicionado à rede:

$$N_w = N_w + 1, \tag{2.67}$$
$$\mathbf{w}_{N_w} = \mathbf{1}^D, \tag{2.68}$$
$$\mathbf{m}_{N_w} = \mathbf{0}^C. \tag{2.69}$$

Classificação de um vetor de entrada. Na fase de teste da rede Fuzzy ARTMAP, determina-se um neurônio vencedor em relação a um vetor de entrada desconhecido a partir da Equação (2.61). Após o protótipo $\mathbf{w}_{i^*}$ ser escolhido como vencedor, a classe inferida é aquela determinada por $\mathbf{m}_{i^*}$.

Os passos de treinamento são repetidos para todos os N pares $\{\mathbf{x}(n), \mathbf{y}(n)\}$ disponíveis, sempre retornando o parâmetro de vigilância ρ a um valor base $\overline{\rho}$ ao se apresentar um novo padrão. Um resumo do treinamento da Fuzzy ARTMAP encontra-se no Algoritmo 2.2.

2.5.3 Interpretação geométrica da Rede Fuzzy ARTMAP

Analisando o algoritmo de treinamento da rede Fuzzy ARTMAP, verifica-se que trata-se do algoritmo da rede Fuzzy ART adicionado de um critério de predição que leva em consideração a matriz Inter-MAP e os rótulos conhecidos dos padrões de entrada. Além disso, a rede Fuzzy ARTMAP compartilha das propriedades das redes neurais da família ART, como a capacidade de realizar aprendizado contínuo e a capacidade de lidar com distribuições não-estacionárias.

A Figura 2.11 ilustra um exemplo bidimensional em que a rede foi treinada de maneira a diferenciar padrões de duas classes. Os parâmetros usados foram $\beta = 0$, $\rho_0 = 0{,}6$ e $\eta = 1$. Observa-se que os protótipos criados são vinculados a rótulos específicos, representados na Figura 2.11 por cores diferentes e no algoritmo de

Algoritmo 2.2 Algoritmo de treinamento da rede Fuzzy ARTMAP.

Constantes

β: parâmetro de escolha, $\beta \geq 0$

ρ_0: valor base do parâmetro de vigilância ρ, $0 < \rho_0 \leq 1$

η: parâmetro de aprendizado, $0 < \eta \leq 1$

$n_{\text{MÁX}}$: número de iterações de treinamento

Entradas

$\mathbf{a}(n)$: vetor de entrada, dimensão P

$\mathbf{x}(n)$: vetor de entrada, dimensão $D = 2P$ (codificação complementar)

$\mathbf{y}(n)$: rótulo do padrão $\mathbf{x}(n)$, dimensão C

Algoritmo

1. Inicialização $(n = 0)$

Criar e inicializar os pesos do neurônio inicial da rede $\mathbf{w}_1(0) = \mathbf{1}^D$

Criar e inicializar os pesos da matriz Inter-MAP, $\mathbf{M}(0) = \mathbf{m}_1(0) = \mathbf{0}^C$

2. Laço temporal $(n = 1, 2, \ldots, n_{\text{MÁX}})$

2.1 Selecionar $\mathbf{x}(n)$ do conjunto de vetores de entrada

2.2 Buscar pelo índice do neurônio vencedor:

$i^* = \arg\max_i\{t_i\}$, em que $t_i(n) = \frac{|\mathbf{x}(n) \wedge \mathbf{w}_i(n)|}{\beta + |\mathbf{w}_i(n)|}, \quad i = 1, 2, \ldots, N_w$

2.3 Teste de ressonância (critério de vigilância)

SE $|\mathbf{x}(n) \wedge \mathbf{w}_{i^*}(n)| > \rho|\mathbf{x}(n)|$, ir para o Passo 2.5

SENÃO, voltar para o Passo 2.2 e buscar um novo neurônio vencedor

2.4 Teste de predição

SE $\mathbf{m}_{i^*}(n) = \mathbf{y}(n)$, ir para o Passo 2.4

SENÃO, fazer $\rho = \frac{|\mathbf{x}(n) \wedge \mathbf{w}_{i^*}(n)|}{|\mathbf{x}(n)|} + \epsilon$ e voltar para o Passo 2.2

2.5 Atualização dos pesos

SE $\mathbf{w}_{i^*}(n) = \mathbf{1}^D$ (i.e., o vencedor nunca foi ativado antes), FAZER

$\mathbf{w}_{i^*}(n+1) = \mathbf{x}(n)$ (i.e., armazena o novo padrão)

$\mathbf{m}_{i^*}(n+1) = \mathbf{y}(n)$ (i.e., armazena a classe do novo padrão)

$N_w = N_w + 1$, $\mathbf{w}_{N_w} = \mathbf{1}^D$ e $\mathbf{m}_{N_w} = \mathbf{0}^C$

SENÃO FAÇA

$$\mathbf{w}_{i*}(n+1) = \eta\left(\mathbf{w}_{i^*}(n) \wedge \mathbf{x}(n)\right) + (1-\eta)\mathbf{w}_{i^*}(n)$$

Saídas

$\mathbf{m}_{i^*}(n)$: vetor de pesos que codifica o rótulo da classe do vetor de entrada na iteração n

Observações

Tipicamente, usa-se a técnica de codificação complementar para pré-processar $\mathbf{a}(n)$.

O número de neurônios é iniciado como $N_w = 1$ e incrementado ao longo das iterações.

treinamento pelas colunas da matriz Inter-MAP $\mathbf{M}$.

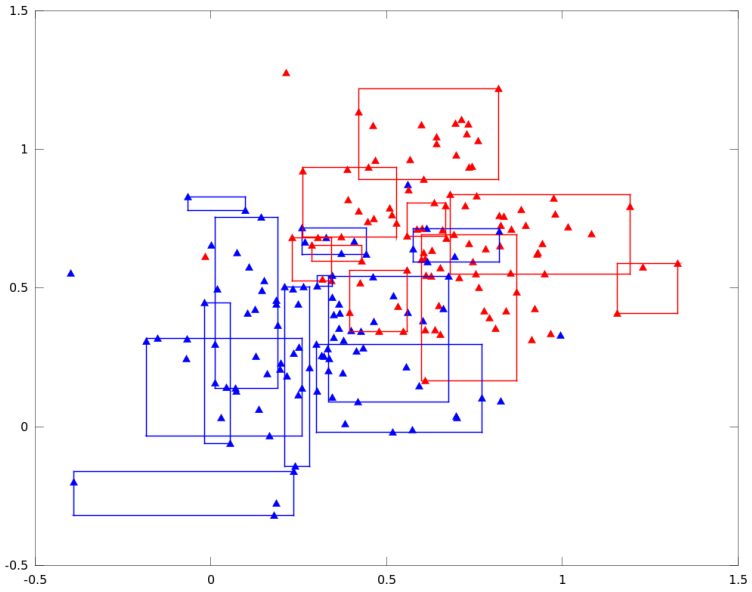


Figura 2.11: Exemplo de operação da rede Fuzzy ARTMAP. As regiões retangulares representam as áreas de influência de cada protótipo da rede. As cores diferentes representam classes diferentes.

É relevante lembrar que o número de protótipos necessários para representar um conjunto de dados é específico do problema considerado. Portanto, os valores dos parâmetros ρ e β devem ser determinados caso a caso para a obtenção de resultados satisfatórios.

O passo de aprendizagem η tem influência direta na parcela de informação que cada vetor de entrada disponível para treinamento transfere aos pesos dos neurônios da rede. Quando seu valor é $\eta = 1$, a rede opera no modo *fast learning* (aprendizado rápido), em que o máximo de informação é extraído de cada padrão. Entretanto, o parâmetro η pode ser ajustado para controlar situações em que existem vetores para treinamento que não são considerados plenamente "confiáveis", como *outliers* e padrões ruidosos. Nesses casos, torna-se interessante a adição de uma parte da informação fornecida pelos padrões anteriormente apresentados. Como essa situação é intrínseca ao problema de interesse, o valor do parâmetro η deve ser escolhido para cada caso.

Este capítulo abordou os classificadores baseada em protótipos, como o DMC, algoritmos da família LVQ (LVQ-1, LVQ-2, LVQ-2.1, LVQ-3, OLVQ e LVQ-Batch)

e algoritmos avançados baseados em LVQ (LVQTC e *Soft* LVQ), classificadores baseados em Rede SOM e em Rede *Fuzzy* ARTMAP. Foi descrito o funcionamento de cada classificador, características, topologias e processos de treinamento.

Capítulo 3

Fundamentos de Evolução Diferencial

Neste capítulo serão apresentados os fundamentos da metaheurística Evolução Diferencial (ED), que será usada na proposta desta dissertação para projetar classificadores baseados em protótipos.

Evolução Diferencial (*Differential Evolution*, DE) é uma metaheurística de busca estocástica baseada em uma população de soluções candidatas, desenvolvido por Storn e Price em 1995 (STORN; PRICE, 1997). Enquanto a DE compartilha similaridades com outros algoritmos evolucionários (AE), ela difere significativamente com relação à informação de distância e de direção da população atual, que é utilizada para guiar o processo de busca. Além disso, as estratégias da DE original foram desenvolvidas para serem aplicadas em problemas que utilizam variáveis contínuas (ENGELBRECHT, 2007).

Para os algoritmos evolucionários canônicos, tal como algoritmos genéticos, a variação de uma geração para a outra é obtida pela aplicação dos operadores de recombinação (*crossover*) e/ ou mutação. Ambos os operadores são usados da seguinte forma: a recombinação é aplicada primeiro aos indivíduos selecionados e, posteriormente, aplica-se a mutação na prole gerada. Para estes algoritmos, os tamanhos dos passos de mutação são medidos a partir de algumas funções de distribuição de probabilidade. A metaheurística ED difere de outros algoritmos evolucionários nas seguintes características:

- a mutação é aplicada primeiro para gerar um vetor de ensaio (*trial vetor*), que será usado posteriormente na operação de recombinação para produção de uma prole, e;

- o tamanho dos passos de mutação não é contabilizado a partir de uma função distribuição de probabilidades conhecida *a priori*.

Na metaheurística DE, os tamanhos dos passos de mutação são influenciados pelas diferenças entre os indivíduos e a população atual.

3.1 Vetor de Diferenças: Explicando a metaheurística DE

A Evolução Diferencial é um algoritmo de otimização global baseado em população, e que utiliza representação através de números reais. O i-ésimo vetor indivíduo (cromossomo) da população da geração t com n_x componentes, definindo suas dimensções, pode ser representado simplificadamente como um vetor de números reais, tal como

$$\mathbf{z}_i(t) = [z_{i,1}(t) \quad z_{i,2}(t) \quad \cdots \quad z_{i,d}(t)]. \tag{3.1}$$

As posições iniciais dos indivíduos fornecem informações valiosas sobre o desempenho que será alcançado. Prover um método de inicialização dos indivíduos de forma aleatória e uniformemente variada é interessante para a construção da população inicial. Desta forma, os indivíduos iniciais terão uma representação mais completa do espaço de busca, com distâncias relativas maiores entre eles. Ao longo do tempo, com o processo de busca, as distâncias entre os indivíduos tornam-se menores, acarretando na convergência dos indivíduos para a mesma solução. Importante ter a ciência de que a magnitude das distâncias iniciais entre indivíduos é influenciada pelo tamanho da população. Quanto mais indivíduos em uma população, menor será a amplitude das distâncias.

As distâncias entre os indivíduos são uma boa indicação da diversidade da população corrente e da ordem de grandeza dos tamanhos dos passos que devem ser tomados para que a população convirja para uma solução adequada. Se os indivíduos estão longe uns dos outros, é mais interessante que se utilize passos de tamanhos maiores, a fim de explorar o espaço de busca da melhor forma possível. Por outro lado, se as distâncias entre os indivíduos são pequenas, os tamanhos dos passos devem ser menores, com o objetivo de explorar melhor áreas locais. É esse procedimento que é obtido por meio da metaheurística DE ao calcular o tamanho do passo de mutação como diferenças ponderadas entre os indivíduos selecionados

aleatoriamente. O primeiro passo da mutação é, portanto, calcular um ou mais vetores de diferença e, então, usar estes vetores de diferença para determinar a grandeza e a direção dos tamanhos dos passos.

A utilização do vetor-diferença como uma medida da variação da população traz algumas vantagens. Primeiramente, informações com relação ao desempenho apresentado, como representada pela população corrente, são usadas para direcionar a busca. Em segundo lugar, devido ao teorema do limite central , os tamanhos dos passos utilizados na mutação se aproximam de uma distribuição normal, desde que a população seja suficientemente grande para permitir um bom número de vetores de diferenças. A média da distribuição formada pelos vetores de diferença é sempre zero, desde que os indivíduos usados para o cálculo dos vetores de diferenças sejam selecionados de maneira uniforme a partir da população. Sob a condição de que os indivíduos são uniformemente selecionados, essa característica decorre do fato de que os vetores de diferenças ou simplesmente diferenciais, $(\mathbf{x}_{i1} - \mathbf{x}_{i2})$ e $(\mathbf{x}_{i2} - \mathbf{x}_{i1})$ ocorrem com igual frequência, onde $\mathbf{x}_{i1}$ e $\mathbf{x}_{i2}$ são dois indivíduos escolhidos ao acaso. A média de valor zero resultante dos tamanhos do passos garante que a população não vá sofrer de deriva genética. Deve-se perceber também que o desvio da distribuição é determinado pela magnitude dos vetores de diferenças. Eventualmente, os vetores de diferenças se tornarão infinitesimais, resultando na grande ocorrência de pequenas mutações.

Se n_v é o número de diferenciais usado, e n_s é o tamanho da população, logo o número total de perturbações diferenciais é dado por

$$\binom{n_s}{2n_v} 2n_v! \approx O\left(n_s^{2n_v}\right), \tag{3.2}$$

que expressa o número total de direções que podem ser exploradas por geração. Para amplificar o poder de exploração da Evolução Diferencial, o número de direções pode ser acrescido, aumentando-se o tamanho da população e/ou o número de diferenciais adotados.

3.2 Operador Mutação em DE

O operador de mutação da metaheurística DE produz um vetor de ensaio para cada indivíduo da população corrente, transformando (multiplicando) um vetor alvo por meio do diferencial ponderado calculado. O vetor de teste será, então, utilizado

pelo operador de recombinação para produzir a prole. Para cada um dos pais, $\mathbf{x}_i(t)$, gerar o vetor de ensaio $\mathbf{u}_i(t)$, conforme a seguir: Escolha um vetor alvo $\mathbf{x}_{i1}(t)$ a partir da população, de tal forma que $i \neq i_1$. Em seguida, selecionar aleatoriamente dois indivíduos $\mathbf{x}_{i2}$ e $\mathbf{x}_{i3}$ a partir da população, de tal forma que $i \neq i_1 \neq i_2 \neq i_3$ e $i_2, i_3 \sim U(1, n_s)$. Usando estes indivíduos, o vetor de ensaio é calculado através da perturbação do vetor alvo da seguinte forma

$$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)) \tag{3.3}$$

onde $\beta \in (0, \infty)$ é o fator de escala, que controla a amplificação da variação das diferenças. Esta operação pode ser visualizada graficamente por meio da Figura 3.1. Diferentes abordagens podem ser usadas para selecionar o vetor alvo e para calcular os diferenciais, o que será discutido nas próximas seções.

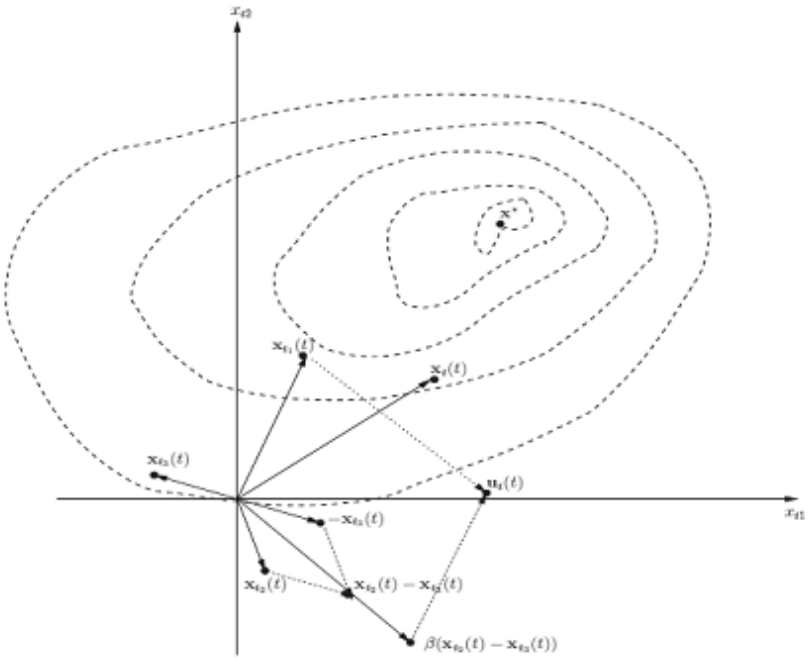


Figura 3.1: Ilustra o operador de mutação da Evolução Diferencial. O ponto ótimo é indicado por $\mathbf{x}*$, onde se assume $\beta = 1,5$. Fonte: Adaptado de Andries P. Engelbrecht (2007 pág.243).

3.3 Operador de Recombinação em DE

O operador de recombinação da metaheurística DE promove uma troca de elementos entre o vetor de ensaio $\mathbf{u}_i(t)$, e o vetor pai (*parent vector*) $\mathbf{x}_i(t)$, para

produção da prole $\mathbf{x'}_i(t)$. Este operador é implementado da seguinte forma

$$\mathbf{x'}_{ij}(t) = \left\{ \begin{array}{l} \mathbf{u}_{ij}(t) \text{ se } j \in \mathcal{J} \\ \mathbf{x}_{ij}(t) \text{ caso contrário} \end{array} \right\} \tag{3.4}$$

onde $\mathbf{x'}_{ij}(t)$ representa o f-ésimo elemento do vetor $\mathbf{x}_i(t)$, e $\mathcal{J}$ é o conjunto de índices dos elementos que serão submetidos à perturbação (em outras palavras, o conjunto de pontos de crossover), conforme a Figura 3.2. Diferentes métodos podem ser utilizados para determinação do conjunto $\mathcal{J}$, sendo as abordagens seguintes as mais frequentemente abordadas:

- **Cruzamento binomial**: Os pontos de cruzamento são selecionados aleatoriamente a partir de um conjunto de possíveis pontos de cruzamento $\{1, 2, \ldots, n_x\}$, em que n_x é a dimensão do problema. O algoritmo 3.1 resume este processo. Nesse algoritmo, p_r é a probabilidade de que o ponto de cruzamento seja incluso e $U(0, 1)$ é um número aleatório uniforme entre 0 e 1. Quanto maior for o valor de p_r, mais pontos de cruzamento serão selecionados, se comparado com a escolha de um valor menor. Isto significa que mais elementos do vetor de ensaio serão utilizados para produzir a prole e menos elementos do vetor pai. Por conta da decisão probabilística ser feita por meio da inclusão de um ponto de cruzamento, pode acontecer de não haver pontos selecionados, caso em que a prole será simplesmente o progenitor (pai) original $\mathbf{x}_i(t)$. Este problema torna-se mais evidente em espaços de busca de baixa dimensão. Para forçar que pelo menos um elemento da prole seja diferente do pai, o conjunto de pontos de cruzamento $\mathcal{J}$ é inicializado para incluir um ponto selecionado aleatoriamente j^*. $\mathbf{p}_i(n)$ determina o n-ésimo valor do vetor gerado no processo de cruzamento.

- **Cruzamento exponencial**: A partir de um índice selecionado aleatoriamente, o operador de cruzamento exponencial elege uma sequência de pontos de cruzamento adjacentes, tratando a lista de potenciais pontos de cruzamento como uma matriz circular. O pseudocódigo no algoritmo 3.2 mostra que pelo menos um ponto de cruzamento é selecionado e, a partir deste índice, seleciona-se o próximo até $\mathbf{U}(0, 1) \geq p_r$ ou $|\mathcal{J}| = n_x$, em que $|\cdot|$

Algoritmo 3.1 *Crossover* Binomial da Evolução Diferencial para seleção de pontos de recombinação

1. Inicialização

$j^* \sim \mathbf{U}(1, n_x)$;

$\mathcal{J} \leftarrow \mathcal{J} \cup \{j^*\}$;

2. Laço temporal

Para cada $j \in \{1, \ldots, n_x\}$ **faça**

Se $\mathbf{U}(0,1) < p_r$ and $j \neq j^*$, **então** $\mathbf{p}_i(n) = \mathbf{x}_i(n)$;

Fim Para

denota a cardinalidade (*i.e.* o número de elementos do seu argumento).

Algoritmo 3.2 *Crossover* Exponencial da Evolução Diferencial para seleção de pontos de recombinação

1. Inicialização

$\mathcal{J} \leftarrow \{\}$; // Inícia o conjunto de í

$j \sim \mathbf{U}(1, n_x - 1)$; // Número aleatóriamente uniforme entre 1 e $(n_x - 1)$

2. Laço temporal

Repetir

$\mathcal{J} \leftarrow \mathcal{J} \cup \{j + 1\}$; // Habilita o índice subsequente ao escolhido

$j = (j + 1) \mod n_x$; // Atribui o índice subsequente ao ponteiro

até $\mathbf{U}(0,1) \geq p_r$ ou $|\Gamma| = n_x$; // Até que uma condição seja aceira

3.4 Operador Seleção em DE

O operador Seleção é aplicado para determinar quais os indivíduos que irão participar da operação de mutação na produção de um vetor de ensaio e para determinar se é o pai ou a prole resultante que irá permanecer (sobreviver) na próxima geração. Com referência ao operador da mutação, um série de operadores podem ser utilizados. Seleção aleatória é normalmente usada para selecionar os indivíduos a partir do vetor alvo. Para a maioria das implementações da metaheurística DE, o vetor alvo pode ser selecionado tanto aleatoriamente, quanto escolhido o melhor indivíduo.

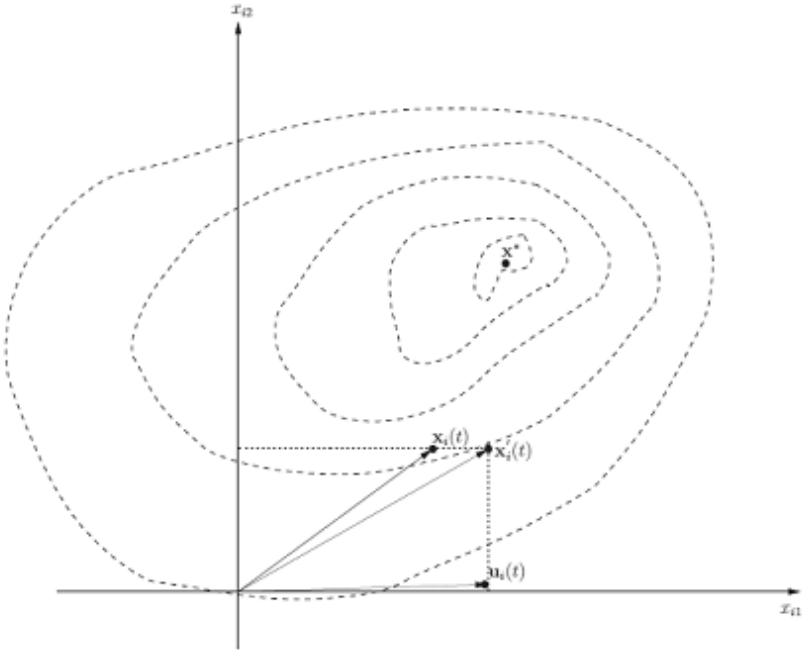


Figura 3.2: Ilustração do funcionamento do operador de cruzamento da metaheurística DE. A figura apresenta que a descendência é constituída pelo primeiro elemento do vetor de ensaio, $\mathbf{u}_i(t)$, e o segundo elemento do pai, $\mathbf{x}_i(t)$. Fonte: Adaptado de Andries P. Engelbrecht (2007 pág.243).

Na geração da população para a próxima geração, normalmente utiliza-se a seleção determinística: na qual a prole substitui o pai se a aptidão da prole é melhor que seu pai. Caso contrário, o pai sobrevive para a próxima geração. Isto assegura que a aptidão média da população não se deteriore, conforme

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{x}'_i(t), & \text{If } f(\mathbf{x}'_i(t)) > f(\mathbf{x}_i(t)) \\ \mathbf{x}_i(t), & \text{If } f(\mathbf{x}'_i(t)) \leq f(\mathbf{x}_i(t)) \end{cases} \tag{3.5}$$

onde $f(\cdot)$ é a função objetivo a ser maximizada.

3.5 Algoritmo Clássico de Evolução Diferencial

O algoritmo abaixo fornece uma implementação genérica das estratégias da Evolução Diferencial básica. A inicialização da população é feita através da seleção de valores aleatórios dos elementos de cada indivíduo a partir dos limites definidos para o problema abordado. Para cada indivíduo $\mathbf{x}_i(t), x_{ij}(t) \sim U(x_{min,j}, x_{max,j})$, onde $(x_{min,j}, x_{max,j})$ definem os limites mínimo e máximo da variável (COELLO, 2007).

Algoritmo 3.3 Algoritmo Clássico da Metaheurística ED

1. Inicialização

Definir o contador de geração, $t = 0$;

Inicializar os parâmetros de controle, β e p_r;

Criar e inicializar a população, $\mathbf{P}(0)$, de n_s indivíduos;

2. Laço temporal

Enquanto condição de de parada não for verdadeira

Para cada indivíduo $\mathbf{x}_i(t) \in \mathbf{P}(t)$

(i) Avaliar a aptidão $f(\mathbf{x}_i(t))$;

(ii) Criar o vetor de ensaio $\mathbf{u}_i(t)$ através da aplicação do operador de mutação;

(iii) Criar a prole $\mathbf{x}_i(t)$, através da aplicação do operador de recombinação;

SE $f(\mathbf{x'}_i(t))$ for melhor que $f(\mathbf{x}_i(t))$

Então Adicionar $\mathbf{x'}_i(t)$ em $\mathbf{P}(t+1)$;

Senão Adicionar $\mathbf{x}_i(t)$ em $\mathbf{P}(t+1)$;

Fim Para

Fim Enquanto

Saídas ou variáveis de interesse

Retornar o indivíduo com melhor aptidão, tal como a solução.

3.6 Parâmetros de controle

Além do tamanho da população, n_s, o desempenho da metaheurística DE é influenciado por dois parâmetros de controle, o fator de escala β e a probabilidade de recombinação p_r. Os efeitos destes parâmetros são discutidos a seguir.

- **Tamanho da população**: como citado anteriormente, o tamanho da população exerce influência direta sobre a capacidade de exploração dos algoritmos da metaheurística DE. Quanto mais indivíduos existem na população, mais vetores diferenciais estarão disponíveis e mais direções podem ser exploradas na busca. No entanto, deve-se ter em mente que a complexidade

computacional por geração aumenta com o tamanho da população. Estudos empíricos fornecem a seguinte diretriz: $n_s \approx 10n_x$, ou seja, a população deve ter aproximadamente dez vezes o número de dimensões do vetor de representação do indivíduo. A natureza do processo de mutação, no entanto, fornece um limite inferior para o número de indivíduos: $n_s > 2n_v + 1$, onde n_v é o número de diferenças utilizado. Para n_v diferenciais, são necessários $2n_v$ indivíduos diferentes, sendo 2 para cada diferencial. O indivíduo adicional representa o vetor alvo.

- **Fator de escala**: o fator de escala, $\beta \in (0, \infty)$, controla a amplitude das variações das diferenças, $(\mathbf{x}_{i2} - \mathbf{x}_{i3})$. Quanto menor o valor de β, menor o tamanho do passo de mutação, tornando mais lento o processo de convergência do algoritmo. Valores maiores para β facilitam a exploração no espaço de busca, mas pode fazer com que o algoritmo ultrapasse um ponto ótimo. O valor de β deve ser pequeno o suficiente para permitir que os indivíduos explorem os ótimos locais e grande o suficiente para manter a diversidade. Como explicado anteriormente, quanto mais indivíduos na população, menor deve ser a magnitude dos vetores de diferença, senão os indivíduos mais próximos terão funções similares. Portanto, os passos de tamanhos menores deves ser usados para explorar áreas locais. Mais indivíduos reduzem a necessidade de passos de mutação grandes. Resultados empíricos sugerem que a utilização de valores altos para n_s e β muitas vezes resultam em convergência prematura e que $\beta = 0,5$ geralmente fornece um bom desempenho.

- **Probabilidade de recombinação**: a probabilidade de recombinação, p_r, tem influência direta sobre a diversidade da população na metaheurística DE. Este parâmetro controla o número de elementos do pai, $\mathbf{x}_i(t)$, que será alterado. Quanto maior for a probabilidade de recombinação, maior será a variação que é introduzida na nova população, aumentando assim a diversidade e a abrangência na exploração no espaço de busca. Utilizar um p_r alto muitas vezes resulta em convergência mais rápida, enquanto diminuir o valor de p_r aumenta a robustez na qualidade da busca.

3.7 Notação DE/x/y/z

Uma série de variações para a metaheurística DE clássica já foi abordada na seção anterior. Tais variações diferem com relação à maneira como o vetor alvo é

selecionado, o número de vetores de diferença utilizado e a forma com que os pontos de cruzamento (*crossover*) serão determinados. Com o objetivo de caracterizar estas variações, uma notação geral é geralmente adotada na literatura sobre a metaheurística DE denotada por DE/x/y/z. Usando esta notação, x refere-se ao método de seleção do vetor alvo, y indica o número de vetores de diferença utilizado e z indica o método de recombinação adotado. Assim, tais variantes da metaheurística DE discutidas na seção anterior são designadas como DE/*rand*/1/*bin* para cruzamento binomial, e DE/*rand*/1/*exp* para cruzamento exponencial. Outras estratégias básicas de Evolução Diferencial incluem as listas a seguir:

- **DE/*best*/1/z**: Para esta estratégia, seleciona-se o melhor indivíduo da população atual, $\widehat{\mathbf{x}}(t)$, como o vetor alvo. Neste caso, o vetor de ensaio é calculado como

$$\mathbf{u}_i(t) = \widehat{\mathbf{x}}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)) \tag{3.6}$$

  Neste caso, qualquer método de recombinação pode ser utilizado.

- **DE/x/n_v/z**: Para esta estratégia, utiliza-se mais de um vetor-diferença. O vetor de ensaio é calculado como

$$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_2,k}(t) - \mathbf{x}_{i_3,k}(t)) \tag{3.7}$$

  onde $\mathbf{x}_{i_2,k}(t) - \mathbf{x}_{i_3,k}(t)$ indica o k-ésimo vetor-diferença, $\mathbf{x}_{i_1}(t)$ pode ser selecionada usando qualquer método adequado para a seleção do vetor-alvo e também pode ser utilizado qualquer método de cruzamento. Com referência à equação que modela o cruzamento binomial, quanto maior for o valor de n_v, mais direções podem ser exploradas por geração.

- **DE/rand-to-best/n_v/z**: Esta estratégia combina os métodos *rand* e *best* para o cálculo do vetor de ensaio (*trial*), da seguinte forma

$$\mathbf{u}_i(t) = \gamma\widehat{\mathbf{x}}(t) + (1-\gamma)\mathbf{x}_{i_1}(t) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_2,k}(t) - \mathbf{x}_{i_3,k}(t)) \tag{3.8}$$

  onde $\mathbf{x}_{i_1}(t)$ é selecionado aleatoriamente, e $\gamma \in [0, 1]$ indica a "cobiça" do operador de mutação. O valor de γ, quanto mais próximo de 1, aumenta a cobiça do processo. Em outras palavras, γ com valor próximo a 1 favorece a

exploração, enquanto um valor próximo de 0 favorece o aprendizado. Uma boa estratégia é de usar um γ adaptativo, com $\gamma(0) = 0$. O valor de $\gamma(t)$ aumenta com cada nova geração até atingir o valor 1.

Note que no caso em que $\gamma = 0$, obtém-se a estratégia DE/*rand*/y/z, enquanto para $\gamma = 1$, tem-se a estratégia DE/*best*/y/z.

- **DE/*current-to-best*/1+n_v/z**: Com esta estratégia, há a mutação do vetor pai utilizando-se, pelo menos, dois tipo de vetores-diferença. Um dos vetor-diferença é calculado entre o vetor *best* e o vetor pai, enquanto o restante das diferenças são calculadas através de vetores selecionados aleatoriamente:

$$\mathbf{u}_i(t) = \mathbf{x}_i(t) + \beta(\widehat{\mathbf{x}}(t) - \mathbf{x}_i(t)) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_1,k}(t) - \mathbf{x}_{i_2,k}(t)) \tag{3.9}$$

Estudos empíricos têm mostrado que a estratégia DE/rand/1/bin mantém uma boa diversidade, enquanto a DE/current-to-best/2/bin apresenta boas características de convergência (QIN; SUGANTHAN, 2005). Devido a esta observação, Qin e Suganthan (QIN; SUGANTHAN, 2005) desenvolveram um algoritmo de metaheurística DE que alterna dinamicamente entre essas duas estratégias.

A cada uma dessas estratégias é atribuída uma probabilidade para ser aplicada, sendo que $p_{s,1}$ é a probabilidade de que DE/*rand*/1/*bin* seja aplicada, em seguida, $p_{s,2} = 1 - p_{s,1}$ é a probabilidade de que DE/*current-to-best*/2/*bin* seja aplicada.

$$p_{s,1} = \frac{n_{s,1}(n_{s,2} + n_{f,2})}{n_{s,2}(n_{s,1} + n_{f,1}) + n_{s,1}(n_{s,2} + n_{f,2})} \tag{3.10}$$

onde $n_{s,1}$ e $n_{s,2}$ são respectivamente o número de descendentes que sobreviverão na próxima geração para o DE/*rand*/1/*bin*, e $n_{f,1}$ e $n_{f,2}$ representam o número de descendentes descartados em cada geração para esta estratégia. Quantos mais descendentes sobreviverem para um estratégia específica, maior será a probabilidade de escolha desta estratégia para a próxima geração.

3.8 Simulações numéricas com a metaheurística DE

Nesta seção é feita uma comparação entre os resultados obtidos na utilização do classificador Evolução Diferencial, a fim de analisar seu comportamento. Desta

forma, é apresentado um estudo cujo objetivo é analisar o efeito provocado pela variação dos principais parâmetros aplicados na Evolução Diferencial. Para o problema de otimização foi escolhida a função de Rastrigin (MENDES, 2004; OLIVEIRA, 2004) bidimensional, pelo fato de ser uma função clássica na literatura para a aferição de desempenho.

3.8.1 Função de Rastrigin

Seja a função conhecida como função de Rastrigin, que possui 10^n mínimos locais, dado pela Equação 3.11. Considerando $n = 2$, tem-se uma função com 100 mínimos locais, conforme a Figura 3.3. Esta função é definida tal que

$$f(\vec{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right] \tag{3.11}$$

onde $A = 10$, $x_i \in [-5,12; 5,12]$ e n é um parâmetro que se refere ao número de dimensões da função. Esta função possui mínimo global $f(\vec{x}) = 0$ quando $\vec{x} = \vec{0}$. A Figura 3.3 apresenta o formato da função quando $n = 2$

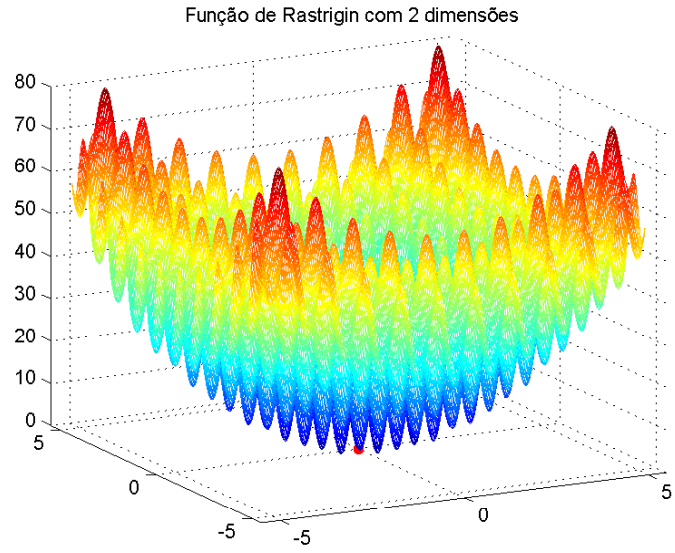


Figura 3.3: Função Rastrigin para n = 2.

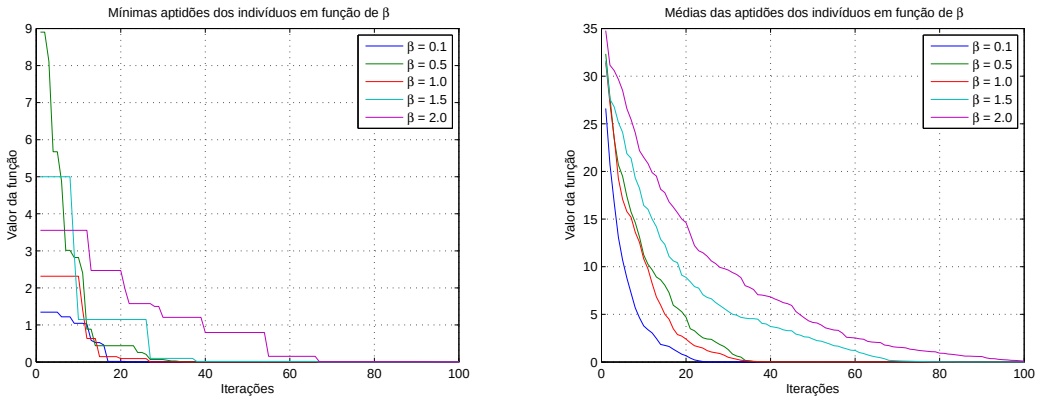
A Função de Rastrigin foi escolhida para o estudo com o intuito de analisar o efeito da utilização de vários parâmetros e obter os melhores valores que possibilitem bons desempenhos nas estratégias da DE. Importante ressaltar que

valores adequados irão depender também das características do problema tratado.

3.8.2 Variação do fator de escala (de perturbação, β)

Nesta primeira simulação fez-se a variação do fator de escala, β, aplicado sobre os indivíduos na população, considerando os valores no intervalo entre $0,1$ e $2,0$; enquanto os parâmetros fixos são: $g_{ger} = 100$, $n_s = 40$ e $p_r = 0,5$.

A Figura 3.4 apresenta as curvas de aptidão mínima e média da população ao longo das gerações, na etapa de treinamento da DE, em função de vários valores para o fator de escala (mais especificamente 0,1; 0,5; 1,0; 1,5 e 2,0).



(a) Mínimas aptidões dos indivíduos em função de β

(b) Médias das aptidões dos indivíduos em função de β

Figura 3.4: Variação da convergência da DE sobre a função de Rastrigin em relação a β.

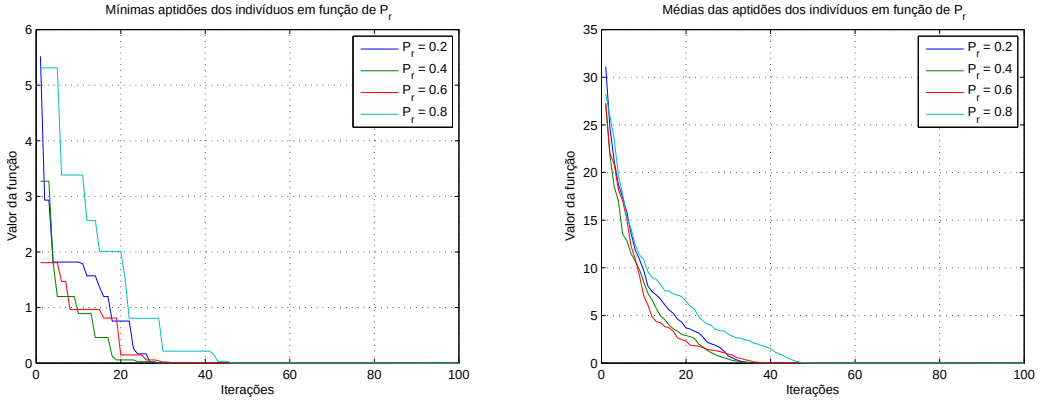
Com base nas Figuras 3.4(a) e 3.4(b) é possível perceber que a utilização de um maior valor para o fator de escala (aumentando o valor de β) acarreta na necessidade de um número maior de avaliações da função objetivo até atingir um ponto sub-ótimo global, exigindo um maior esforço computacional. A escolha de um valor muito baixo, por conseguinte, pode ocasionar perda da diversidade nos indivíduos e fazer com que a solução convirja para um mínimo local.

3.8.3 Variação da probabilidade de recombinação (p_r)

Para a verificação da variação dos valores da probabilidade de recombinação, p_r, aplicados aos indivíduos, foram considerados valores pré-estabelecidos na faixa entre $0,2$ e $0,8$, adotando-se os parâmetros fixos: $g_{ger} = 100$, $n_s = 40$ e $\beta = 0,5$.

Observa-se que a Figura 3.5 apresenta a curva de aptidão mínima e média da população ao longos das gerações, na etapa de treinamento da DE, em função de

vários valores para a probabilidade de recombinação (mais especificamente 0,2; 0,4; 0,6 e 0,8).



(a) Mínimas aptidões dos indivíduos em função de p_r

(b) Médias das aptidões dos indivíduos em função de p_r

Figura 3.5: Variação da convergência da DE sobre a função de Rastrigin em relação a p_r.

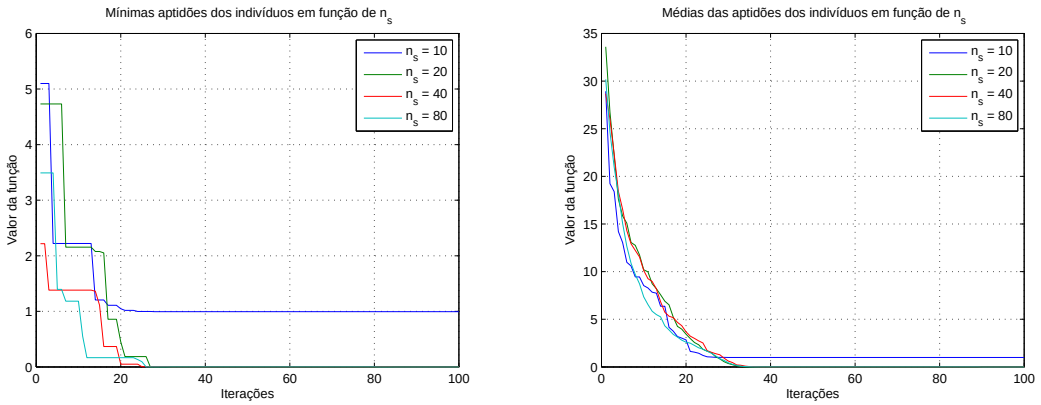
Com relação à Figura 3.4(a) é possível perceber que a utilização um valor para p_r abaixo de $0,3$ (por exemplo, quando $p_r = 0,2$) não é recomendado, visto que diminui muito a diversidade na população, tornando a convergência muito lenta e computacionalmente custosa. Já na Figura 3.4(b), percebe-se que ao utilizar um valor alto, por exemplo $0,8$, a solução apresenta uma convergência rápida, podendo ocasionar parada prematura em detrimento de um mínimo local.

3.8.4 Variação do número de indivíduos da população (n_s)

Este teste faz a variação do número de indivíduos da população, n_s, cujo objetivo é a verificação da influência deste parâmetro no desempenho do algoritmo. Foram estabelecidas populações com 10, 20, 40 e 80 indivíduos. Os parâmetros fixos foram: $g_{ger} = 100$, $p_r = 0,5$ e $\beta = 0,5$.

A Figura 3.6 apresenta a curva de aptidão mínima e média da população ao longos das gerações, na etapa de treinamento da DE, em função de diversos valores para o número de indivíduos da população.

A partir da Figuras 3.6(a) e 3.6(b) é possível perceber que em se utilizando $n_s = 10$ tando a melhor solução quanto os demais indivíduos convergem para um ponto de mínimo local. Isto se dá ao fato de que o pequeno número de indivíduos restringe a capacidade de exploração do algoritmo. As demais simulações



(a) Mínimas aptidões dos indivíduos em função de n_s

(b) Médias das aptidões dos indivíduos em função de n_s

Figura 3.6: Variação da convergência da DE sobre a função de Rastrigin em relação a n_s.

apresentaram desempenho similares, demonstrando que a partir de $n_s = 20$ já se obtém um desempenho aceitável para a função de Rastrigin.

Importante resaltar que o objetivo específico deste teste é estabelecer um número mínimo de indivíduos na população que forneça desempenho aceitável, uma vez que quanto maior a quantidade, maior será o consumo de recursos computacionais e, consequentemente, o tempo de execução.

O presente capítulo expôs os fundamentos de Evolução Diferencial, iniciando com a explicação da metaheurística e seguindo com os operadores de mutação e de seleção. Aborda a versão canônica do algoritmo DE, detalha seus parâmetros de controle e descreve as notações DE/x/y/z presentes na literatura clássica. Por fim, foi apresentada uma série de simulações executadas com o classificador DE, objetivando sua análise comportamental e a seleção da melhor configuração paramétrica.

Capítulo 4

Proposta de um Classificador Baseado em Protótipos Usando DE

Classificadores baseados em protótipos, principalmente os da família LVQ vêm sendo bastante aplicados nos últimos anos, trazendo bons resultados nas mais diversas áreas (PERNER, 2014; ATTIG, 2011; HAMMER, 2011). Porém, para que estes classificadores sejam adequadamente utilizados, o usuário necessita definir um número inicial de protótipos por classe, antes que seja dado início à etapa de treinamento. A quantidade ideal de protótipos é a menor quantidade de protótipos por classe que represente da melhor forma os dados de treinamento. Normalmente a quantidade ótima de protótipos por classe é buscada de forma empírica, após muita experimentação com os dados de treinamento. Isto além de demandar tempo, também requer conhecimento técnico prévio com relação ao comportamento do funcionamento do algoritmo. Visando sanar estas limitações no processo de projeto de classificadores baseados em protótipos do tipo LVQ, foi desenvolvido o algoritmo *Learning Vector Quantization based on Differential Evolution* (LVQ-DE), que possui como objetivo maior automatizar o processo de seleção de um projeto de rede LVQ.

4.1 Um Classificador Baseado em Protótipos Usando DE

O classificador baseado em protótipos que será proposto nesta seção possibilita a otimização de projeto por meio de DE. A abordagem proposta possui grande motivação em aplicações bem sucedidas recentemente do algoritmo DE para agrupamento particional (*cluster*), conforme consta em (DAS; ABRAHAM;

KONAR, 2008; PATERLINI; KRINK, 2006). Mais especificamente, o procedimento de projeto pode ser visto como uma extensão de um classificador de padrões supervisionado da abordagem de agrupamento automático introduzido por Das *et al.* (DAS; ABRAHAM; KONAR, 2008), doravante denotado por método DAK (de Das-Abraham-Konar). Os detalhes do método DAK e sua extensão propostos no documento atual serão abordados nos parágrafos seguintes.

No método DAK, cada cromossomo $\mathbf{z}_i$ na população $\mathbf{P}$, com n_s indivíduos, define uma solução de agrupamento com um certo número de *clusters* posicionados de forma ótima no espaço de dados. Uma das características do método DAK é determinar automaticamente o número ótimo de centróides de *clusters* (ou protótipos) em um cenário não supervisionado.

Com este propósito, eles especificaram um cromossomo cujos primeiros J_{max} componentes, $T_{i,j}$, definem os limiares de ativação associados a exatamente J_{max} centros de *clusters*. Para os padrões de vetores de dimensão d, os $d * J_{max}$ componentes restantes do i-ésimo cromossomo contém os centros de *cluster* para este indivíduo, $\mathbf{m}_j^{(i)}$. Assim, o i-ésimo cromossomo do método DAK é codificado como segue:

$$\mathbf{z}_i(t) = [T_{i,1}\ T_{i,2}\ \cdots\ T_{i,J_{max}} \mid \mathbf{m}_1^{(i)}\ \mathbf{m}_2^{(i)}\ \cdots\ \mathbf{m}_{J_{max}}^{(i)}] \tag{4.1}$$

Por exemplo, seja $T_{i,j}$ o limiar de ativação da j-ésima centróide de cluster do i-ésimo cromossomo. A ideia de ativação de um gene está relacionada à intenção de se obter a menor quantidade de protótipos para representar suficientemente bem o conjunto de dados. Estes parâmetros de ativação são iniciados com valores aleatórios a partir de uma distribuição uniforme no intervalo $[0, 1]$. O parâmetro de ativação do gene será passível tanto de recombinação quanto de mutação, podendo influenciar diretamente na aptidão do indivíduo. O parâmetro de ativação será considerado ativo caso $T_{i,j} > 0,5$, caso contrário, será considerado inativo. Após a criação de uma nova geração, estes parâmetros de ativação são submetidos a uma regra de avaliação, conforme será explanado a seguir.

Ao longo do curso da execução do algoritmo DE, quando um novo cromossomo da prole é criado de acordo com as Equações (3.4) e (3.5), a princípio, os valores dos limites de ativação são usados para selecionar os centróides de *clusters* ativos.

Se devido a uma mutação algum limite $T_{i,j}$ em um descendente excede 1 ou torna-se negativo, ele é forçado a voltar ou a 1 ou 0, respectivamente. Além disso,

Algoritmo 4.1 Validação de genes ativos.

SE $T_{i,j} > 0,5$ **ENTÃO**

O protótipo $\mathbf{m}_{i,j}$ é considerado ATIVO;

SENÃO

$\mathbf{m}_{i,j}$ é INATIVO;

FIM SE

ao se verificar que todos os limiares de ativação são menores do que 0,5, deve-se escolher aleatoriamente dois protótipos e reinicializar seus limiares para um valor aleatório entre $0,5$ e $1,0$, a fim de ter sempre um número mínimo de dois no grupo. Esta foi a abordagem inicialmente utilizada.

Em classificadores supervisionados da família LVQ, os protótipos são rotulados e são permitidos vários protótipos por classe. Assim, com o objetivo de estender o método DAK para permitir o projeto de classificadores baseados em protótipos, precisamos definir o número máximo permitido de protótipos por classe, $L_{max}^{\omega_k}$. Em geral, $1 \leq L_{max}^{\omega_k} < n_k$, onde n_k é o número de exemplos de treinamento da k-ésima classe. Em seguida, devemos definir os valores limiares de ativação do protótipo dentro de uma classe.

No método DAK estendido (EDAK), considerando o i-ésimo cromossomo na população, os primeiros $L_{max}^{\omega_1}$ valores de limiar de ativação correspondem a $L_{max}^{\omega_1}$ vetores protótipos da primeira classe ω_1:

$$\mathbf{t}_i^{(\omega_1)} = [t_{i,1}^{(\omega_1)} \;\; t_{i,2}^{(\omega_1)} \;\; \cdots \;\; t_{i,L_{max}^{\omega_1}}^{(\omega_1)}] \in \mathbb{R}^{L_{max}^{\omega_1}}, \tag{4.2}$$

com o conjunto correspondente de $L_{max}^{\omega_1}$ protótipos dado por

$$\mathbf{W}_i^{(\omega_1)} = [\mathbf{m}_{i,1}^{(\omega_1)} \;\; \mathbf{m}_{i,2}^{(\omega_1)} \;\; \cdots \;\; \mathbf{m}_{i,L_{max}^{\omega_1}}^{(\omega_1)}] \in \mathbb{R}^{d \cdot L_{max}^{\omega_1}}, \tag{4.3}$$

onde d é a dimensão do vetor de padrões. Da mesma forma, os $L_{max}^{\omega_2}$ valores subsequentes de limiar de ativação correspondem aos $L_{max}^{\omega_2}$ vetores de protótipo da segunda classe ω_2, ou seja,

$$\mathbf{t}_i^{(\omega_2)} = [t_{i,1}^{(\omega_2)} \;\; t_{i,2}^{(\omega_2)} \;\; \cdots \; t_{i,L_{max}^{\omega_2}}^{(\omega_2)}] \in \mathbb{R}^{L_{max}^{\omega_2}}, \tag{4.4}$$

Com o conjunto correspondente de $L_{max}^{\omega_2}$ protótipos dados por

$$\mathbf{W}_i^{(\omega_2)} = [\mathbf{m}_{i,1}^{(\omega_2)} \;\; \mathbf{m}_{i,2}^{(\omega_2)} \;\; \cdots \;\; \mathbf{m}_{i,L_{max}^{\omega_2}}^{(\omega_2)}] \in \mathbb{R}^{d \cdot L_{max}^{\omega_2}}. \tag{4.5}$$

Este processo de codificação é repetido até o último componente da classe ω_K:

$$\mathbf{t}_i^{(\omega_K)} = [t_{i,1}^{(\omega_K)} \;\; t_{i,2}^{(\omega_K)} \;\; \cdots \;\; t_{i,L_{max}^{\omega_k}}^{(\omega_K)}] \in \mathbb{R}^{L_{max}^{\omega_k}}, \tag{4.6}$$

com o conjunto correspondente de $L_{max}^{\omega_k}$ protótipos dada por

$$\mathbf{W}_i^{(\omega_K)} = [\mathbf{m}_{i,1}^{(\omega_K)} \;\; \mathbf{m}_{i,2}^{(\omega_K)} \;\; \cdots \;\; \mathbf{m}_{i,L_{max}^{\omega_k}}^{(\omega_K)}] \in \mathbb{R}^{d \cdot L_{max}^{\omega_k}}. \tag{4.7}$$

Agrupando todas as definiçes das Equações (4.2) até (4.7), onde K_{tot} é o total de classes, o i-ésimo cromossomo na geração t codifica um classificador baseado em protótipos representado da seguinte forma:

$$\mathbf{z}_i(t) = [\mathbf{t}_i^{(\omega_1)} \mid \cdots \mid \mathbf{t}_i^{(\omega_{K_{tot}})} \mid \mathbf{W}_i^{(\omega_1)} \mid \cdots \mid \mathbf{W}_i^{(\omega_{K_{tot}})}]. \tag{4.8}$$

A partir do exposto, a dimensionalidade total do cromossomo $\mathbf{z}_i(t)$ é

$$\begin{aligned} d_{tot} &= \dim(\mathbf{z}_i(t)) = \sum_{k=1}^{K_{tot}} L_{max}^{\omega_k} + d \cdot L_{max}^{\omega_k}, \\ &= \sum_{k=1}^{K_{tot}} L_{max}^{\omega_k} \cdot (1+d). \end{aligned} \tag{4.9}$$

Como um exemplo típico, o cromossomo hipotético mostrado na Figura 4.1 corresponde a um classificador baseado em protótipos para um problema bi-dimensional (isto é, d = 2), de duas classes (por exemplo, $K_{tot} = 2$), e $L_{max}^{\omega_1} = L_{max}^{\omega_2} = 3$. O classificador resultante possui três protótipos atribuídos à primeira classe e dois protótipos relacionados à segunda. Note que o segundo protótipo da segunda classe (*i.e.* $\mathbf{m}_{i,2}^{(\omega_2)}$) não se encontra ativo porque seu limiar de ativação está abaixo de 0,5.

Como as redes LVQ caracterizam-se pelo treinamento supervisionado, cada amostra de dado está relacionada (ou rotulada) a uma classe. No método EDAK a disposição destes protótipos no cromossomo é feita separada por regiões, de forma que nunca ocorra um cruzamento de dados entre protótipos pertencentes

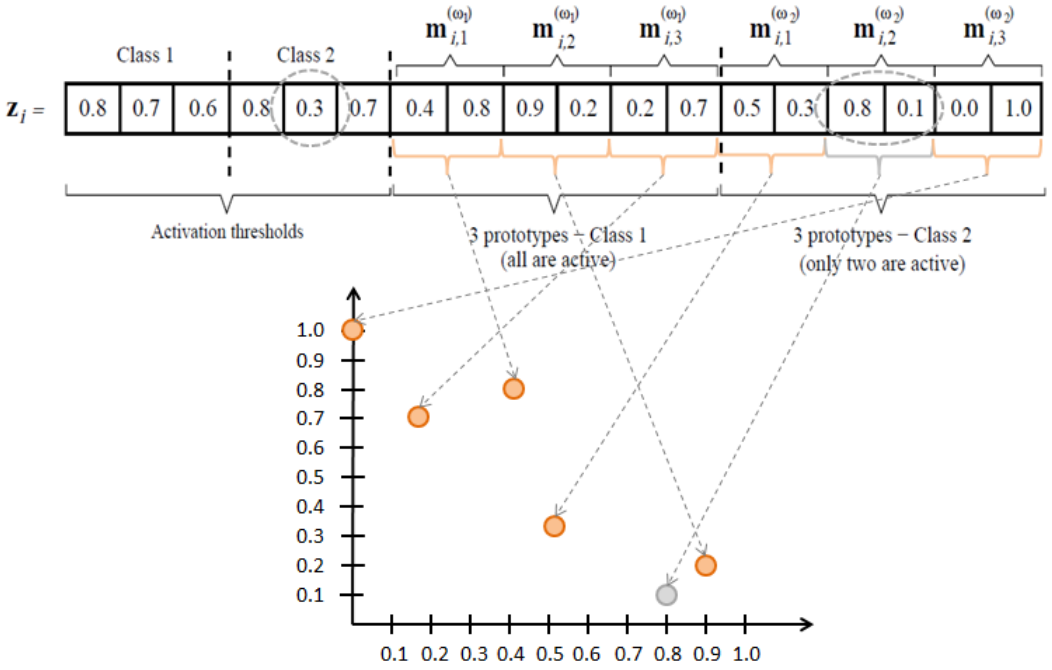


Figura 4.1: Estrutura hipotética do cromossomo para o método EDAK.

a classes distintas. Esta metodologia visa evitar que haja "contaminação genética" entre protótipos de classes diferentes, visto que cada classe possui características de disposição e posicionamento espacial próprios.

4.2 A Função de Aptidão Proposta

Uma vez que estamos interessados em construir classificadores de padrões eficientes, é natural usar uma função de aptidão que leve em consideração uma medida do desempenho da classificação, tal como a taxa de reconhecimento. No entanto, é interessante obter a maior taxa de classificação, sendo isto possível utilizando um menor número de protótipos. Tendo isso em mente, a seguinte função de aptidão é usada pelo método EDAK para a avaliação dos cromossomos na geração t:

$$f(\mathbf{z}_i(t)) = CR(\mathbf{z}_i(t)) - \lambda \cdot Q(\mathbf{z}_i(t)), \tag{4.10}$$

onde $CR(\mathbf{z}_i(t))$ é a taxa de classificação do i-ésimo cromossomo na geração t para o conjunto de validação, e $Q(\mathbf{z}_i(t))$ é o número de protótipos ativos do classificador associado com o i-ésimo cromossomo.

O fator de penalização λ é usado para controlar a influência do número

de protótipos ativos sobre a função de aptidão. Como o objetivo do método EDAK é maximizar $f(\mathbf{z}_i(t))$, grandes (pequenos) valores de λ tendem a produzir classificadores com um baixo (elevado) número de protótipos. A solução final a ser encontrada pelo método EDAK é um compromisso entre uma taxa de classificação boa e um número suficiente de protótipos. A solução final obtida pelo método EDAK será referida a partir de agora como o classificador LVQ-DE.

Vale ressaltar que λ é um hiperparâmetro do método EDAK e seu valor afeta consideravelmente o desempenho da abordagem proposta. Obviamente, problemas de classificação mais complexos (ou seja, com limites de decisão não-linear, disjuntos ou altamente sobrepostos) provavelmente exigirão mais protótipos e, portanto, valores mais baixos para o hiperparâmetro λ.

4.3 Classificador LVQ-DE

O classificador desenvolvido LVQ-DE, como mencionado anteriormente, está fundamentado na estrutura proposta pelo método EDAK. Neste tópico serão apresentadas algumas peculiaridades relacionadas ao funcionamento do classificador LVQ-DE. Um pseudocódigo do algoritmo é descrito no Algoritmo 4.2.

No LVQ-DE, a quantidade de genes, ou L_{max}, que constituirá o cromossomo será baseada na quantidade de amostras de dados apresentadas pelo conjunto de treinamento e nas características de disposição espacial destes dados (quanto mais complexo o conjunto, maior este fator). Este fator, determinado pelo parâmetro p_{gc} (percentagem de geração de cromossomos), representa um percentual que é aplicado no total de amostras do conjunto de dados de treinamento para a determinação do L_{max}, onde normalmente utiliza-se 50%. Após determinado o valor de L_{max}, os cromossomos recebem valores iniciais a partir da obtenção aleatória de dados do conjunto de treinamento. Serão respeitadas as proporções dos números de amostras por classe no conjunto adotado, sendo $L_{max}^{\omega_k}$ o número máximo de genes contidos na classe k para um cromossomo.

A utilização da ativação de um gene será importante no momento da aplicação da função de aptidão a um indivíduo, pois este gene será expresso como uma rede LVQ e apenas os genes (ou protótipos) que estiverem ativos serão utilizados na criação da rede. Este mecanismo permite que protótipos mal treinados ou superpostos sejam excluídos da solução buscada, visto que um dos objetivos é a diminuição da quantidade de protótipos.

Algoritmo 4.2 Algoritmo LVQ-DE.

1. Inicialização

(1.1) Definir a quantidade de gerações, q_{ger} e o contador $t = 0$;

(1.2) Iniciar os parâmetros de controle, p_r, β e λ;

(1.3) Criar e iniciar a população, $\mathbf{P}(0)$, de n_s indivíduos;

(1.4) Aplicar a Regra de Ativação;

(1.5) Aplicar o procedimento de repovoamento em caso de inativação da classe;

2. Laço temporal

PARA (EXT) t menor ou igual a q_{ger} **FAZER**

PARA (INT) cada indivíduo $\mathbf{z}_i(t) \in \mathbf{P}(t)$ **FAZER**

(i) Avaliar a aptidão $f(\mathbf{z}_i(t))$ de cada indivíduo (Rede LVQ);

(ii) Criar o vetor de ensaio (*trial vector*) $\mathbf{u}_i(t)$ através da aplicação do operador de mutação;

(iii) Criar a prole $\mathbf{z'}_i(t)$, através da aplicação do operador de recombinação (*crossover*) por valor dimensional;

SE $f(\mathbf{z'}_i(t))$ for melhor que $f(\mathbf{z}_i(t))$ **ENTÃO**

Adicionar $\mathbf{z'}_i(t)$ para $\mathbf{P}(t+1)$;

SENÃO

Adicionar $\mathbf{z}_i(t)$ para $\mathbf{P}(t+1)$;

FIM SE

(iv) Aplicar a Regra de Ativação, conforme Algoritmo 4.1;

(v) Aplicar o procedimento de repovoamento em caso de inativação da classe;

FIM PARA (INT)

$t = t + 1$;

FIM PARA (EXT)

Saídas ou variáveis de interesse

Retornar o indivíduo com melhor aptidão, como a solução.

Como veremos mais adiante, a função de ativação utilizada na LVQ-DE influencia na diminuição do número total de protótipos, no qual, ao longo das gerações é natural a redução gradual dos genes ativos. Como consequência deste comportamento, é provável que, em algum momento no processo de evolução, possa ocorrer que todos os protótipos de uma mesma classe $\mathbf{y}_k$ estejam inativos. Para resolver este problema, foi criado um mecanismo de repovoamento. Este mecanismo, após verificar que todos os protótipos de uma classe $\mathbf{y}_k$ estão inativos, exclui todo o conteúdo dos protótipos relacionados à classe $\mathbf{y}_k$ em questão e reinicia os valores (repovoa) de todos estes protótipos. Estes valores são gerados a partir do valor calculado da centróide de todos os dados da classe $\mathbf{y}_k$, o qual é acrescido um valor aleatório a cada dimensão do vetor protótipo. Se todos os protótipos de uma classe ω_k estiverem inativos, ou seja:

$$\text{SE } {t_{i,j}}^{(\omega_k)} < 0,5, \quad \forall j \tag{4.11}$$

$$\text{ENTÃO } {\mathbf{m}_{i,j}}^{(\omega_k)} = \mathbf{c}^{(\omega_k)} + \mathbf{p} \tag{4.12}$$

onde $\mathbf{c}^{(\omega_k)}$ é o centróide da classe ω_k e o vetor aleatório $\mathbf{p} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$, em que $\mathbf{I}$ é a matriz de identidade e σ^2 é variância das perturbações aleatrias aplicadas a cada componente do vetor $\mathbf{p}$.

4.4 Outras Considerações sobre o LVQ-DE

O fator de penalização, λ, desempenha a função de hiperparâmetro no modelo de classificador proposto, ou seja, é um parâmetro que influencia a quantidade de protótipos da solução final alcançada. Por meio de experimentação exaustiva, verificou-se que os valores no intervalo $0,001 - 0,01$ geraram bons resultados para o método EDAK sobre os conjuntos de dados avaliados. Quanto maior o valor de λ, maior será a influência da penalização da quantidade de protótipos por rede LVQ, o que força a algoritmo LVQ-DE a diminuir o número de protótipos a cada geração. Quando possui valor menor, ele influenciará menos no decrescimento, sendo indicado para problemas mais complexos, que necessitarão de mais protótipos para uma representação adequada.

No modelo proposto foi utilizada a estratégia de implementação DE/*rand*/1/*bin*, na qual o método apresenta seleção aleatória do vetor alvo, cruzamento

do tipo binomial e um vetor de diferença adotado. Além da estratégia DE/*rand*/1/*bin*, também foi desenvolvido o mesmo algoritmo utilizando as estratégias DE/*rand*/1/*exp*, DE/*best*/1/(*bin* e *exp*), DE/*rand-to-best*/(1 e 2)/(*bin* e *exp*) e DE/*current-to-best*/1+2/*bin*; em que, após experimentos comparativos de desempenho, foi eleita a DE/*rand*/1/*bin* por ter apresentado desempenho superior na maioria dos testes e pela sua simplicidade.

Outras variações desenvolvidas no algoritmo levam em conta a forma de reposição dos genes quando ultrapassam as fronteiras do espaço de busca. Neste caso, além do método de reposição centróide, foram desenvolvidos os métodos de reposição aleatório e reposição na fronteira. A reposição aleatória reconfigura o gene "infrator" com os valores de um gene selecionado aleatoriamente no conjunto de dados, enquanto a reposição na fronteira ajusta os valores das componentes do gene "infrator" que ultrapassarem os limites com o próprio valor fronteiriço. Após a comparação por meio de testes, foi constatado que tanto os métodos de reposição aleatória, quanto o centróide apresentaram desempenho similar, porém o método centróide foi utilizado por se mostrar uma solução mais elegante do ponto de vista teórico.

Na Evolução Diferencial original, o vetor de diferenças $(\mathbf{Z}_i(t) - \mathbf{Z}_j(t))$ é ponderado por um fator constante f_e. Uma escolha usual para este parâmetro de controle fica entre 0.4 e 1.0. Alguns autores citam que a geração deste fator de forma aleatória (em um intervalo especificado, ex. $[0.5, 1]$) possa resultar em uma melhora do desempenho, pois permite uma variação estocástica na amplificação do vetor de diferenças, e também ajuda na retenção da diversidade populacional (DAS, 2005). Porém, no LVQ-DE foi visto que a utilização de um valor constante para f_e resultava em um desempenho melhor, principalmente em decorrência da maior estabilidade no processo de convergência (menor desvio padrão nas simulações).

O p_r (taxa de recombinação) também é utilizado em alguns trabalhos como um número gerado de forma variável, normalmente descrescido linearmente de um valor máximo a um valor mínimo. Por exemplo, quando o $p_r = 1.0$ significa que todas as componentes do vetor pai serão substituídos pelo operador do vetor de diferenças e vice-versa. No LVQ-DE este parâmetro é utilizado como uma constante, visto que ele possui uma característica dualista. Quando é adotado um valor muito alto (ex. acima de 0.5), o algoritmo converge mais rápido, porém, há maior chance de convergir para um máximo local, visto que, em decorrência de uma

maior perturbação (ruidos), também há mais perdas de informações no processo de evolução. Quando se utiliza um valor muito baixo (ex. abaixo de 0.2), há uma convergência mais lenta, demandando maior tempo de execução (maior consumo de processamento), porém a convergência é mais suave, permitindo um resultado mais consistente.

Uma das características importantes do LVQ-DE é que ele fornece as posições finais dos protótipos, sem a necessidade de retreino. Devido ao comportamento de busca otimizada da metaheurística DE, obtém-se como resultado as posições dos protótipos no espaço dos dados. Desta forma, há uma grande diferença de projeto com relação à seleção do modelo quando comparado com o projeto de classificadores baseados em protótipos convencionais. Este comportamento só é possível por conta da estrutura definida para o cromossomo, que permite a incorporação direta das posições dos protótipos, e da influência da função de aptidão na convergência da busca para a rede com melhor taxa de acerto.

Este capítulo abordou as motivações de criação de um classificador baseado em protótipos usando DE, sugeriu uma nova representação de cromossomo e uma função de ativação adaptada. Por fim, apresentou o algoritmo LVQ-DE e detalhou algumas caractarísticas do seu comportamento em detrimento de diversas variantes e parâmetros no tópico Outras Considerações.

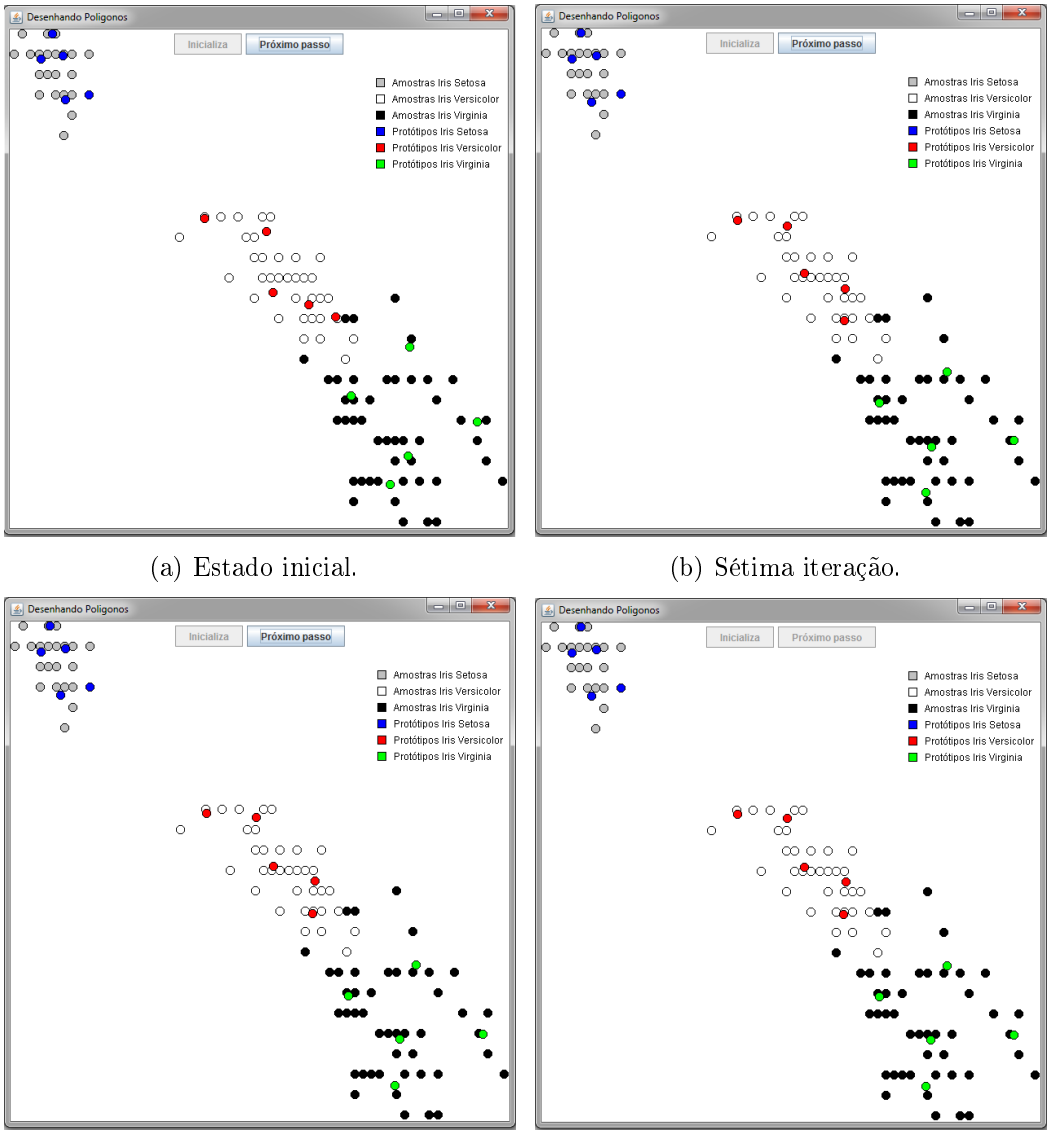
Capítulo 5

Experimentos e Resultados

Como citado nos capítulos introdutórios, o objetivo principal desta dissertação é apresentar um novo método de projeto de classificadores baseados em protótipos, denominado LVQ-DE. Para que seja avaliado o desempenho aprensentado pelo método, outros métodos clássicos como os algoritmos LVQ-2.1, LVQ-3 e OLVQ foram implementados. Também foram implementados variantes mais sofisticados, ou seja, os algoritmos LVQTC e Soft LVQ. As simulações utilizadas para cada método supracitado comportam várias execuções, resultando em uma distribuição das taxas de acerto de todas as execuções em uma simulação de teste. Como insumos gerados pela simulação tem-se métricas estatísticas, que serão utilizadas para comparação entre os classificadores implementados.

Com o objetivo de validar inicialmente o funcionamento dos algoritmos LVQ clássicos, foi elaborada uma interface gráfica para visualização de dados bidimensionais e dos protótipos em tempo de execução. Esta interface permite avaliar o comportamento dos protótipos ao longo das épocas de treinamento por meio de uma exibição dos dados em duas dimensões, conforme a Figura 5.1.

A interface gráfica desenvolvida permite visualizar a localização dos protótipos a cada época na execução do treinamento da rede LVQ-2. Esta ferramenta permite analisar a convergência dos protótipos ao longo das épocas e o ajuste correto dos parâmetros do algoritmo de classificação. A Figura 5.1 mostra um exemplo típico de resultados para o conjunto de dados Íris, utilizando-se apenas dois atributos (comprimento da pétala e largura pétala, ambos mensurados em centímetros), permitindo assim sua visualização em duas dimensões ao longo de 20 gerações.



(a) Estado inicial. (b) Sétima iteração.

(c) Décima quarta iteração. (d) Última iteração.

Figura 5.1: Interface gráfica mostra em duas dimensões o comportamento convergênte do treinamento da Rede LVQ-2 aplicado ao conjunto de dados Iris.

5.1 Metodologia e Considerações

As simulações foram desenvolvidas na linguagem Java, utilizando o ambiente de desenvolvimento Eclipse, onde foram utilizadas algumas APIs (*Application Programming Interface*) tais como a Jsci (*A Science API for Java, link: jsci.org.uk/*), utilizada para permitir operações algébricas, matriciais e estatísticas, e a *JFreeChart* (*link: www.jfree.org/jfreechart/*), que é uma biblioteca *open source* que permite a criação de gráficos ou diagramas estatísticos.

Também foi utilizado o ambiente MATLAB para a geração de gráficos estatísticos de melhor qualidade. Neste caso foram gerados arquivos textos, em java, contendo os dados extraídos da simulação executada e, em seguida, lidos por um procedimento MATLAB para geração dos gráficos neste ambiente. A configuração do hardware utilizado para as simulações foi um *desktop* com processador Intel Core i5 – 2500K 3.3Ghz com duas unidades em paralelo da memória Ram Kingston HyperX Blu 4GB 1600MHz DDR3.

Para o treinamento e avaliação de cada algoritmo foi utilizado um método de validação cruzada. A validação cruzada é uma técnica para avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. Busca-se, então, estimar o quão acurado é este modelo na prática, ou seja, o seu desempenho para um novo conjunto de dados. O método de validação cruzada utilizado nesta dissertação foi o *holdout*, que consiste em dividir o conjunto total de dados em dois ou mais subconjuntos mutuamente exclusivos. Neste Trabalho o conjunto de dados original foi dividido em três partes, nas seguintes proporções: 3/5 dos dados para treinamento, 1/5 para validação e os 1/5 restantes para testes, podendo serem modificadas conforme características do conjunto de dados tratado. A parcela do conjunto separada para validação é usada para determinar os hiperparâmetros do modelo.

Uma simulação consiste em uma série de execuções dos algoritmos. Para cada execução, o conjunto de dados original é embaralhado e utilizado conforme o método de validação cruzada adotado. A partir dos dados estatísticos que sumarizam esta série de execuções é tomado o resultado final da simulação. As métricas estatísticas utilizadas para análise das simulações são os valores mínimo, máximo, média, mediana, desvio padrão e coeficiente de variação da taxa de acerto do classificador.

Menos utilizado, o coeficiente de variação de Pearson é uma medida de dispersão relativa empregada para estimar a precisão de experimentos, e representa o desvio padrão expresso como porcentagem da média (aES, 2008; TAVARES, 2007). Sua principal qualidade é a capacidade de comparação de distribuições diferentes, pois, como duas distribuições podem ter valores médios diferentes, o desvio padrão dessas duas distribuições não é comparável. Deste forma, a solução é usar o coeficiente de variação, conforme a Equação 5.1. O coeficiente de variação (CV) é obtido pela razão entre o desvio padrão, σ, e a média, μ.

$$C_v = \frac{\sigma}{\mu} \tag{5.1}$$

Outro recurso para análise do resultado de uma simulação é o histograma, o qual apresenta o comportamento da distribuição estatística, como na Figura 5.2. O histograma permite uma avaliação qualitativa de distribuição de valores da taxa de acerto, um vez que o estudo trata de uma quantidade muito grande de simulações e comparações.

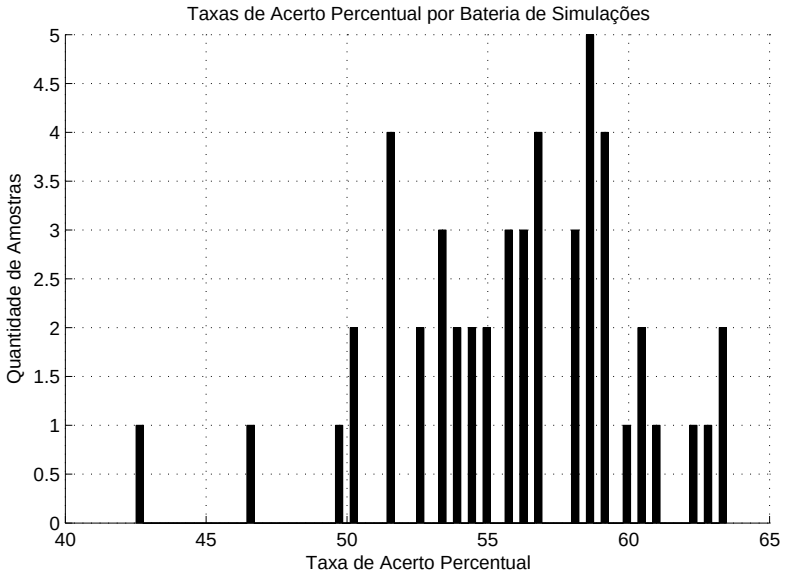


Figura 5.2: Histograma da taxa de acerto sintetizando a distribuição estatística das taxas de acerto percentuais para uma bateria de execuções de uma simulação para a rede LVQ 2.1.

Para todos os algoritmos LVQ avaliados normalmente 200 épocas de treinamento para cada execução, visto que resultou nos melhores desempenhos a partir de testes empíricos. Nestes casos, o número de neurônios precisa ser estipulado antes do treinamento, tendo sido sempre o mesmo número de neurônios para todas as classes. O número adequado de neurônios pode variar de acordo com as características do problema abordado, normalmente em função da precisão necessária requerida para o resultado e do classificador adotado. Com relação aos algoritmos LVQ clássicos (LVQ 2.1, LVQ 3 e OLVQ), foi utilizada a taxa de aprendizado variável monotonicamente decrescente de $0,1$ a $0,001$ ao longo das épocas de treinamento e valor de W (largura da janela) igual a $0,25$ (valores sugeridos na literatura variam entre $0,2$ e $0,3$).

O algoritmo LVQTC teve seus parâmetros configurados da seguinte maneira:

os parâmetros de aprendizagem α_r e α_w receberam o valor $0,01$, sendo linearmente decrescidos por um fator f_r igual a $0,01$ (o valor de f_r pode ser diminuido para refinar a convergência ou aumentado para acelerar a velocidade); o parâmetro de *cutoff* p_{prn} recebe o valor 5, tal que ao aumentar seu valor, aumenta-se a confiabilidade no posicionamento do neurônio estatisticamente. Se for pequeno, tende a aumentar o número de neurônios. O parâmetro de limiar de variação para critério de parada f_{cvg} recebe $0,01$.

O treinamento do LVQTC encerra-se quando o número de classificações corretas para o conjunto de treinamento não apresenta melhora sensível. Desta forma, o critério de parada é utilizado quando durante três épocas sucessivas o número de classificações corretas não ultrapassa $(1 + f_{cvg})$ vezes o número de classificações na época passada, onde $0 < f_{cvg} < 1$. Ao fim da última época nenhum neurônio será removido ou criado. Os três parâmetros utilizados para cálculo da incerteza na etapa de cassificação são o fator de contaminação máximo f_{cmax} com valor $0,976$, número mínimo de treinamentos p_{min} com valor $7,8636$, e a distância máxima de S d_{max} igual a $0,8643$, conforme descritos na seção 2.3.7.

No caso do algoritmo *Soft* LVQ houve certa dificuldade em determinar uma configuração ótima (ou subótima) para seus parâmetros, pois, além de possuir muitos parâmetros, o comportamento do algoritmo pode variar bastante conforme as características do conjunto de dados utilizado. O parâmetro de dispersão varia linearmente a cada época de $1,0$ até $0,001$. O parâmetro beta ρ (utilizado no decaimento do parâmetro de dispersão) inicia com o valor $0,9$, o parâmetro γ (expoente de equação de atualização do parâmetro ρ) inicia com valor $1,1$, e a taxa de aprendizado parte do valor $0,01$, devendo ser atualizada ao longo das épocas.

Com relação ao algoritmo LVQ-DE, uma série de parâmetros pode influenciar no resultado final alcançado e no comportamento de convergência do aprendizado. Foram adotadas, como padrão, populações com 50 indivíduos, uma vez que não foi notado melhora aparente nos resultados com valores reduzidos para os vários conjuntos de dados utilizados. Além disso, uma população muito grande torna o tempo médio de processamento demasiadamente alto.

A determinação do tamanho do cromossomo é de grande importância, podendo variar de acordo com a complexidade do problema. Este parâmetro é denominado percentagem de geração de cromossomos, simbolizado por p_{gc}, em que na maioria dos casos recebeu valor de $0,5$. Logo foi adotado que o cromossomo teria tamanho

inicial de 50% do tamanho de amostras de dados para conjuntos pequenos (abaixo de 400 amostras) e 20% para os demais conjuntos, podendo variar em casos extremos (conjuntos muito grandes). Este procedimento confere uma variabilidade maior aos indivíduos da primeira geração da população.

Outros dois parâmetros que merecem atenção são o fator de escala β e a probabilidade de recombinação p_r. A literatura sugere alguns valores padrões para a utilização destes parâmetros, porém os valores mais adequados destes parâmetros podem variar de acordo com o problema abordado. Os valores dos β e p_r podem ser encontrados a partir da avaliação da curva de aprendizado, gerada pela função de aptidão aplicada ao conjunto de validação. Foi utilizado o valor $\beta = 0.1$ e $p_r = 0.6$ na maior parte dos casos. Em outros casos foram utilizadas algumas variações destes, conforme análise prévia da curva de convergência. O uso de valores inadequados destes parâmetros pode acarretar em problemas no tempo de convergência e na qualidade da busca do algoritmo. A Figura 5.3 ilustra o gráfico da curva de aprendizado utilizada para a seleção correta destes parâmetros:

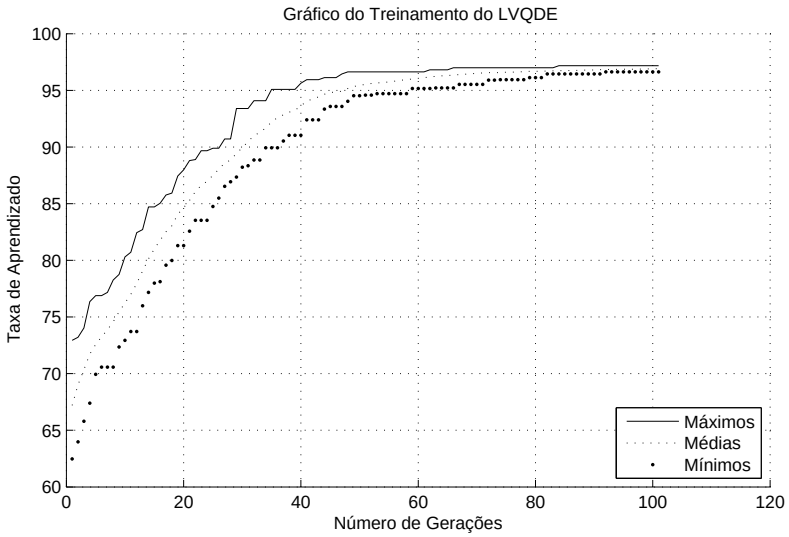


Figura 5.3: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto de validação.

O parâmetro p_r influencia na inserção de variabilidade nos cromossomos ao longo das gerações. Sendo assim, quando o valor de p_r é alto, gera-se muita instabilidade no sistema, tornando difícil a preservação do aprendizado adquirido. Quando o valor de p_r é muito baixo, há uma tendência de acomodação na busca e uma estagnação do processo de aprendizagem. Considerando que o parâmetro β possui função de amplificar o efeito do aprendizado, quando este possui um valor alto o sistema tende

a aprender rápido, porém há uma tendência de convergir prematuramente para um mínimo local. Quando se adota um valor baixo, o sistema converge com menor velocidade, porém requer um número maior de gerações, o que reflete em um maior tempo de processamento. Portanto, é importante ressaltar que a correta utilização dos parâmetros β e p_r permite uma convergência mais adequada com menor tempo de processamento.

Outra função útil da curva de aprendizado está na escolha adequada do número de gerações a ser adotado na etapa de treinamento. Ao longo da análise da curva observa-se que o número ótimo (ou subótimo) de gerações encontra-se a partir do ponto em que a curva se estabiliza. A utilização deste número pode tornar-se um pouco problemática para fins práticos, pois é comum a ocorrência de variações em torno deste ponto de estabilização. Logo, foi adotado como número de gerações o ponto de convergência encontrado acrescido de 20% do número total de gerações, visando suprimir a influência da flutuação no ponto de estabilização. A utilização da adição dos 20% visa garantir a convergência do aprendizado. Importante salientar que o ajuste do número de gerações é de grande importância no ganho do desempenho (em termos de menor tempo de processamento) na etapa de treinamento para uma posterior bateria de testes de simulação. Também deve ser ajustada antes de qualquer treinamento, visando um aprendizado adequado para o conjunto em utilização.

Conforme já mencionado, o grande diferencial do classificador LVQ-DE está na determinação do número de protótipos por classe que compõe o classificador gerado. O parâmetro que exerce maior influência na determinação da quantidade de protótipos é o fator de penalização λ, que é utilizado na função de aptidão, com a função de determinar quanto o número de protótipos penaliza o desempenho da rede LVQ gerada. Desta forma o λ pode ser considerado como o hiperparâmetro do classificador LVQ-DE. A influência deste parâmetro pode ser comprovada ao longo das simulações, que foram executadas utilizando como valores padrões $0,25$, $0,5$, $0,75$ e $1,0$. A inclusão do parâmetro λ na função de aptido possui a intenção de forçar o decremento do número de protótipos. Desta forma, quanto maior, mais intenso será o decremento e vice versa.

5.2 Experimentos Computacionais

Ao longo dos experimentos foram executados os algoritmos LVQ-1, LVQ-2, LVQ-2.1, LVQ-3, LVQ *Batch*, OLVQ, LVQTC, *Soft* LVQ, DMC e o LVQ-DE. Como mencionado anteriormente, o classificador DMC foi utilizado nas simulações como o método mais simples, servindo como ponto de referência para o que se espera de um classificador baseado em protótipo. Os algoritmos LVQ-1, LVQ-2 e LVQ *Batch* não foram utilizados para fins de ranqueamento. Isto porque o algoritmo LVQ *Batch* possui resultado similar ao LVQ-1, por se tratar de uma versão *batch* do mesmo, e os algoritmos LVQ-1, LVQ-2 também não foram utilizados por possuírem desempenho reconhecidamente, segundo a literatura, inferior ou similar às versão 2.1 e 3 dos mesmos. Assim, tanto o LVQ-2.1 quanto o LVQ-3, foram utilizados para as comparações de desempenho. Os demais algoritmos (OLVQ, LVQTC e *Soft* LVQ) também foram utilizados para compor a tabela de resultados comparativos.

Sobre os conjuntos de dados utilizados, é importante ressaltar, de antemão, que os dados são normalizados, por atributo, no intervalo entre $0,1$ e $0,9$ antes de serem utilizados nos algoritmos de classificação. A utilização do intervalo $0,1$ e $0,9$ se deu a fim de dificultar que valores não excedam as barreiras de $0,0$ e $1,0$ estabelecidas como ponto de fronteira no processo de validação dos valores ao longo do treinamento dos algoritmos.

Foram utilizados 10 conjuntos de dados nas simulações para fins comparativos, cada qual com características distintas, objetivando avaliar o desempenho do método proposto diante de circunstâncias diversas de problemas de classificação para comparação com os demais métodos previamente citados neste trabalho.

Em seguida, serão exibidas as tabelas em versão simplificadas dos resultados dos experimentos elaborados. As tabelas são constituidas dos valores da mediana e do desvio padrão das taxas de acerto percentuais para cada bateria de experimentos. Os números de simulações, números de protótipos, dados estatísticos adicionais, assim como parâmetros utilizados, estão presentes nas tabelas completas de resultados no apêndice A. As métricas estatísticas mais utilizadas como medidas de comparação foram, por ordem de prioridade, a mediana, o desvio padrão e, por último, o coeficiente de variação. A mediana possui vantagem em comparação com a média por apresentar seu resultado menos sensível à influência de *outliers*, que implicam, tipicamente, em prejuízos na interpretação dos resultados das simulações. Com

relação à quantidade de execuções em uma simulação (ou quantidade de amostras de taxa de acerto por execução) foram utilizadas na maioria dos casos 100 execuções para cada simulação. Com exceção para algumas simulações mais custosas feitas com o LVQ-DE sobre conjuntos com grande quantidade de exemplos e/ou de variáveis. Nestes casos foram utilizadas de 30 a 50 execuções. Nas tabelas a seguir, as linhas em negrito representam os dois melhores resultados nas simulação referentes a um conjunto de dados.

Tabela 5.1: Quantitativos dos conjuntos.

Conjunto	Atributos	Classes	Dados	d_{tot} ($p_{gc} = 1.0$)
Vert. Column	6	3	310	2170
Heart	13	2	270	3780
Breast Cancer	10	2	699	7689
Glass	10	6	214	2350
Balance	4	3	625	3125
Wine	13	3	178	2492
Vehicle	18	4	946	17974
Dermatology	33	4	366	12444
Wall-Following	4	4	5456	27280
Couro Caprino	80	2	154	12474

5.2.1 Conjunto de dados *Vertebral Column*

Conjunto de dados contendo os valores de seis características biomecânicas, utilizadas na classificação de pacientes ortopédicos em três classes: normal (100 pacientes), hérnia de disco (60 pacientes) e espondilolistese (150 pacientes) (NETO, 2009/8). Cada paciente é representado no conjunto de dados por seis atributos biomecânicos decorrentes da forma e orientação da pelve e coluna lombar (nesta ordem): incidência pélvica, inclinação pélvica, lordose lombar, inclinação sacral, raio pélvico e grau de espondilolistese. A seguinte convenção é usada para os rótulos de classe: hérnia de disco (DH), espondilolistese (SL), Normal (NO). Site de referência: *https://archive.ics.uci.edu/ml/datasets/Vertebral+Column*.

Na simulação referente à Figura 5.4 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}005$; $p_{gc} = 0{,}5$; $q_{ger} = 400$ e $n_s = 50$, que resultou em uma solução onde o total de protótipos foi 6 e a taxa de acerto igual a

Tabela 5.2: Resultados para o conjunto de dados *Vertebral Column*.

Classificador	Mediana	Desv. Pad.	# Protótipos		
			ω_1	ω_2	ω_3
MDC	74.19	4.80	1	1	1
OLVQ	75.80	5.58	5	5	5
LVQ-2.1	**75.80**	**4.75**	**15**	**15**	**15**
LVQ-3	74.19	5.64	15	15	15
Soft LVQ	69.35	4.51	30	30	30
LVQTC	70.96	5.02	15	15	15
LVQ-DE ($\lambda = 0.005$)	**77.42**	**4.03**	**2**	**3**	**2**

77,42%.

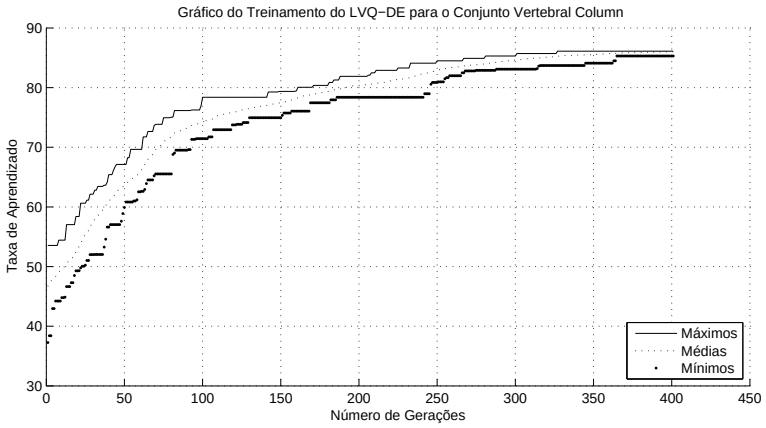
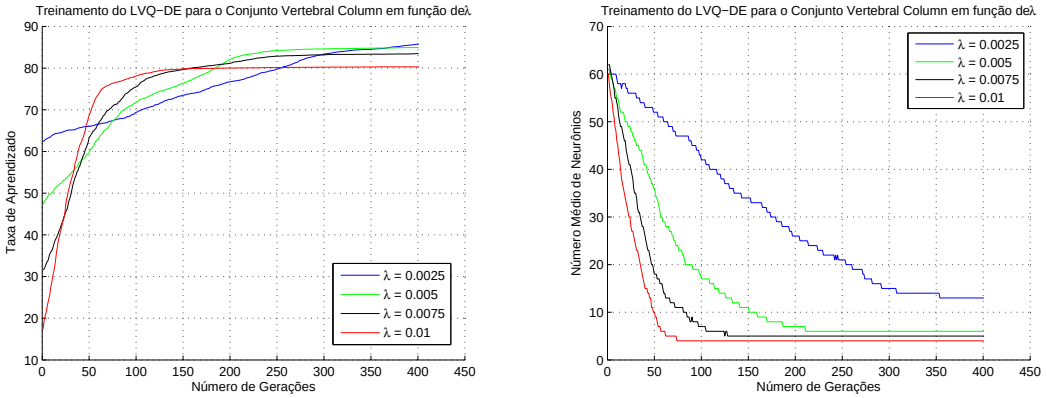


Figura 5.4: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Vertebral Column*.

5.2.2 Conjunto de dados *Heart Disease Cleveland*

Conjunto de dados em que cada instância contém 13 atributos. Possui o objetivo de detectar a presença de doenças do coração em pacientes. São rotulados da seguinte forma: 3 para normal, 6 para doença permanente e 7 para doença reversível. A seguir, lista-se os atributos utilizados: 1. idade, 2. sexo, 3. tipo de dor no peito (4 valores), 4. pressão arterial de repouso, 5. soro de colesterol em mg/dl, 6. glicemia de jejum maior que 120 mg/dl, 7. resultados eletrocardiográficos de descanso (valores $0, 1, 2$), 8. frequência cardíaca máxima atingida, 9. angina



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.5: Efeito da variação paramétrica de λ sobre conjunto de dados *Vertebral Column*.

induzida pelo exercício, 10. pico de idade, 11. pico do exercício do segmento ST, e 12. número dos grandes vazos $(0-3)$ colorido por fluoroscopia. Site de referência: *https://archive.ics.uci.edu/ml/datasets/Statlog+(Heart)*.

Tabela 5.3: Results on the dataset *Heart Disease Cleveland*.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**	
			ω_1	ω_2
MDC	81.48	5.15	1	1
OLVQ	81.48	4.86	5	5
LVQ-2.1	77.77	5.09	15	15
LVQ-3	79.62	4.68	10	10
Soft LVQ	**83.33**	**5.09**	**5**	**5**
LVQTC	79.62	5.29	5	5
LVQ-DE ($\lambda = 0.01$)	**83.33**	**4.48**	**1**	**1**

Na simulação referente à Figura 5.6 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}01$; $p_{gc} = 0{,}5$; $q_{ger} = 200$ e $n_s = 50$, que resultou em uma solução onde o total de protótipos foi 2 e a taxa de acerto igual a 90,74%.

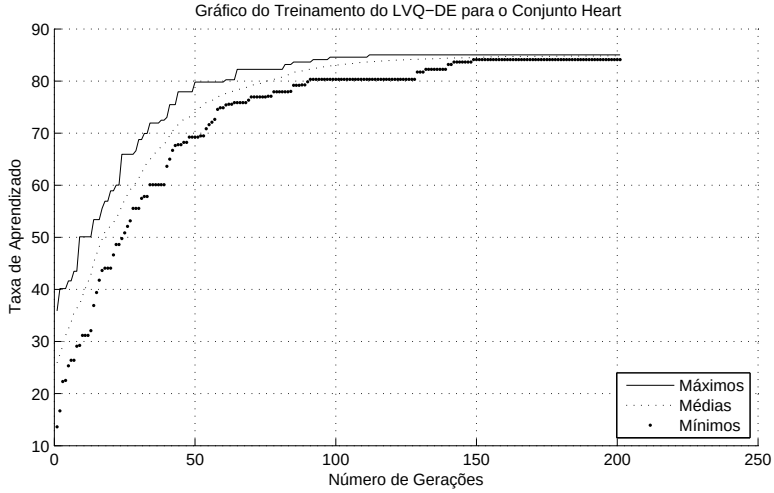
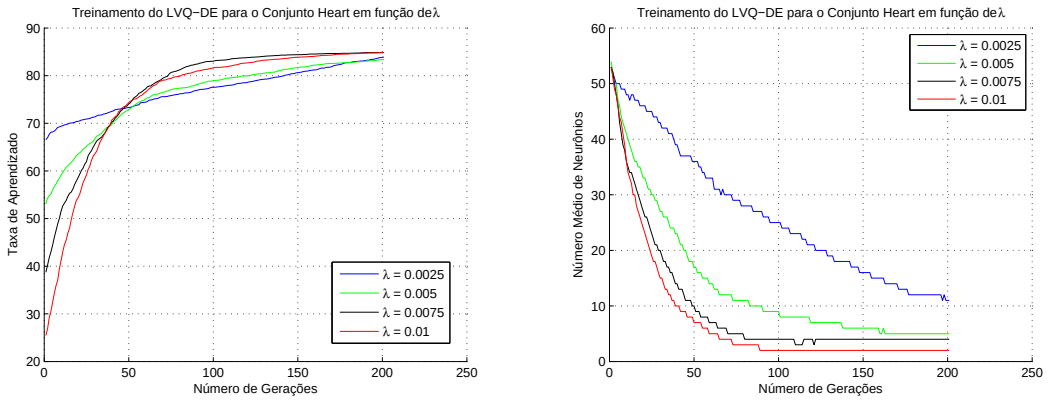


Figura 5.6: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Heart Disease Cleveland*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.7: Efeito da variação paramétrica de λ sobre conjunto de dados *Heart Disease Cleveland*.

5.2.3 Conjunto de dados *Breast Cancer Wisconsin*

Este conjunto de dados é formado por amostras levantadas periodicamente pelos Dr. H. Wolberg, W. Nick Street e Olvi L. Mangasarian na *University of Wisconsin, U.S.A.*, que relata seus casos clínicos no tratamento do câncer de mama. O banco de dados, portanto, reflete este agrupamento cronológico dos dados. Os dados foram gerados no período de 1989 a 1991, congregando 699 amostras no seu total. Importante salientar que ao longo dos anos algumas amostras foram retiradas do conjunto principal, visando melhorar a qualidade da interpretação dos dados. Este conjunto possui 30 atributos e 2 classes (malígno e benígno). Site de referência:

http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original).

Tabela 5.4: Results on the dataset *Breast Cancer Wisconsin.*

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**	
			ω_1	ω_2
MDC	96.29	1.79	1	1
OLVQ	96.29	1.50	10	10
LVQ-2.1	96.66	1.48	10	10
LVQ-3	**97.03**	**1.34**	**10**	**10**
Soft LVQ	96.66	1.28	10	10
LVQTC	96.29	2.41	5	5
LVQ-DE ($\lambda = 0.005$)	**97.03**	**1.14**	**1**	**2**

Na simulação referente à Figura 5.8 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}005$; $p_{gc} = 0{,}3$; $q_{ger} = 150$ e $n_s = 30$, que resultou em uma solução onde o total de protótipos foi 3 e a taxa de acerto igual a 98,52%.

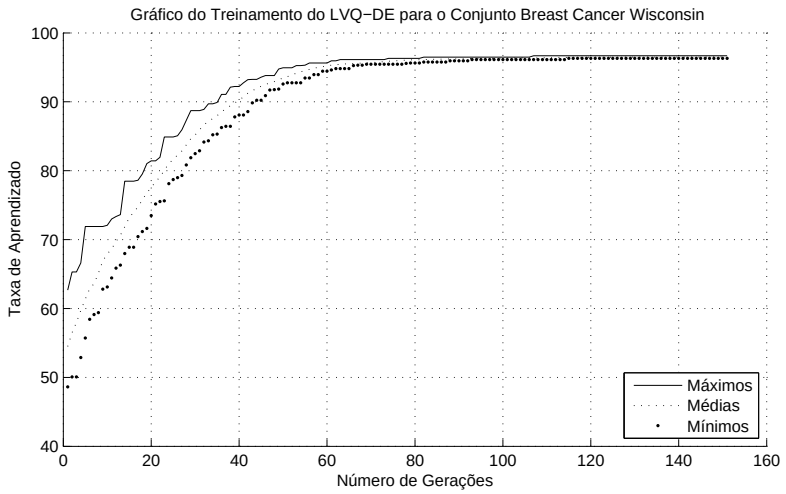
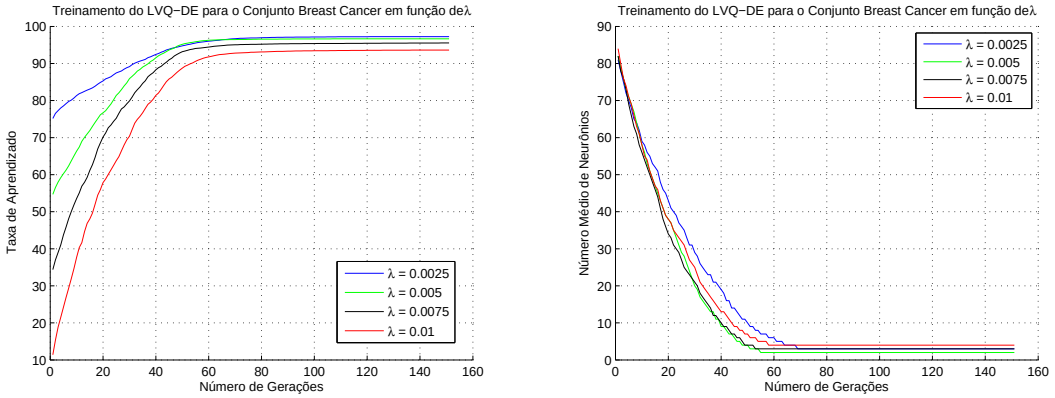


Figura 5.8: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Breast Cancer Wisconsin.*

5.2.4 Conjunto de dados *Glass Identification*

Conjunto de dados dados originário do Serviço de Ciências Forenses dos Estados Unidos, classifica seis tipos de vidros em termos dos conteúdos no seu teor de óxido



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.9: Efeito da variação paramétrica de λ sobre conjunto de dados *Breast Cancer Wisconsin*.

(i.e. Na, Fe, K, etc). O estudo da classificação dos tipos de vidro foi motivada por investigações criminológicas. Na cena do crime, a amostra de vidro pode ser usada como prova, se for identificada corretamente. Este conjunto contém 10 atributos, são eles: número de identificação, índice de refração, sódio, magnésio, alumínio, silício, potácio, cálcio, bário e ferro. O rótulo pode representar 6 classificações de vidros, são eleas: *building windows float processed*, *building windows non float processed*, *vehicle windows float processed*, *containers*, *tableware* e *headlamps*. Site de referência: *http://archive.ics.uci.edu/ml/datasets/Glass+Identification*.

Tabela 5.5: Resultados para o conjunto de dados *Glass Identification*.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**					
			ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
MDC	75.00	5.19	1	1	1	1	1	1
OLVQ	87.5	5.19	12	12	12	12	12	12
LVQ-2.1	87.5	5.40	8	8	8	8	8	8
LVQ-3	**90.0**	**5.03**	**12**	**12**	**12**	**12**	**12**	**12**
Soft LVQ	76.25	5.14	12	12	12	12	12	12
LVQTC	87.5	5.53	12	12	12	12	12	12
LVQ-DE ($\lambda = 0.0075$)	**92.50**	**4.60**	**1**	**3**	**1**	**1**	**1**	**2**

Na simulação referente à Figura 5.10 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}0075$; $p_{gc} = 0{,}5$; $q_{ger} = 300$ e $n_s = 50$, que resultou em uma solução onde o total de protótipos foi 9 e a taxa de acerto igual a 95,00%.

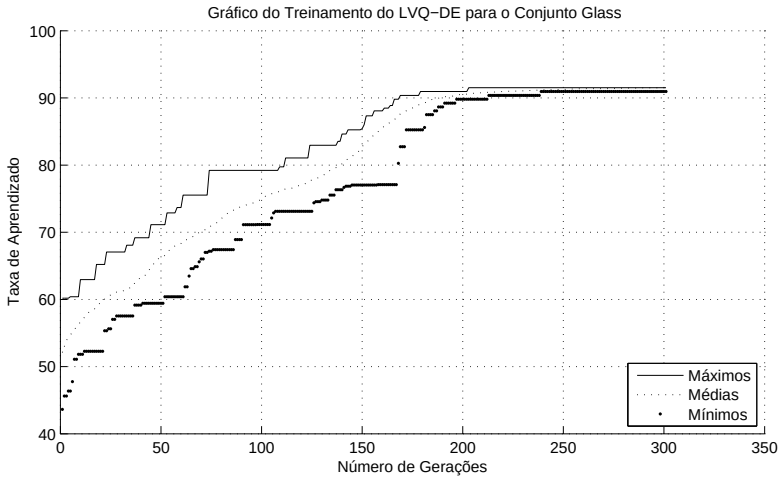
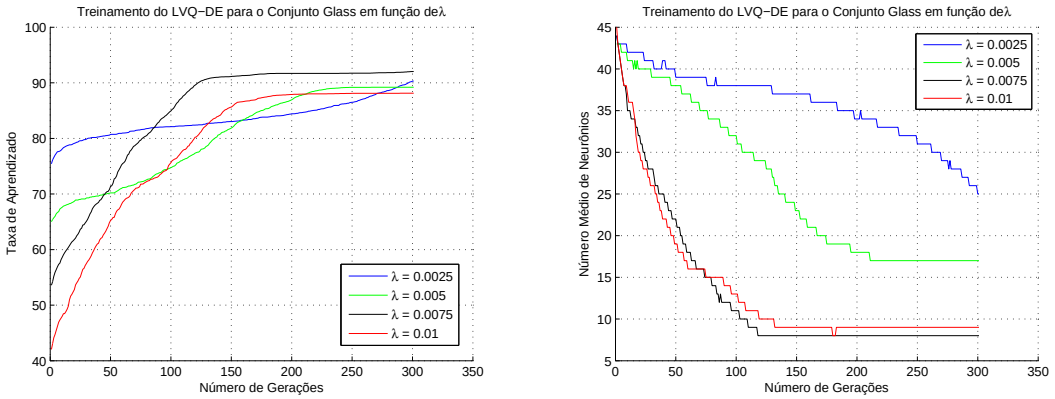


Figura 5.10: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Glass Identification*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.11: Efeito da variação paramétrica de λ sobre conjunto de dados *Glass Identification*.

5.2.5 Conjunto de dados *Balance*

Este conjunto de dados foi gerado para modelar resultados psicológicos experimentais. Cada exemplo é classificado como tendo a escala do movimento do paciente pendendo para a direita, pendendo para a esquerda, ou quando equilibrado (HUME, 2014). Os atributos são o peso da esquerda, a distância da esquerda, o peso certo e a distância certa. A maneira correta para encontrar a classe é o maior entre (distância da esquerda vezes peso da esquerda) e (distância da direita vezes peso da direita). Se eles forem iguais, é equilibrado. Os atributos de uma instância são: 1. nome da classe: 3 (L, B, R), 2. peso da esquerda: 5 (1, 2,

3, 4, 5), 3. distância da esquerda: 5 (1, 2, 3, 4, 5), 4. peso da direita: 5 (1, 2, 3, 4, 5), e 5. distância da direita: 5 (1, 2, 3, 4, 5). Site de referência: *http://archive.ics.uci.edu/ml/datasets/Balance+Scale*.

Tabela 5.6: Resultados para o conjunto de dados *Balance*.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**		
			ω_1	ω_2	ω_3
MDC	73,98	4,00	1	1	1
OLVQ	84,55	2,58	5	5	5
LVQ-2.1	79,67	2,45	15	15	15
LVQ-3	**85,36**	**2,49**	**5**	**5**	**5**
Soft LVQ	**89,02**	**2,54**	**10**	**10**	**10**
LVQTC	79.62	5.29	5	5	5
LVQ-DE ($\lambda = 0.005$)	83.33	4.48	4	1	4

Na simulação referente à Figura 5.12 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}005$; $p_{gc} = 0{,}5$; $q_{ger} = 300$ e $n_s = 50$, que resultou em uma solução onde o total de protótipos foi 13 e a taxa de acerto igual a 88.62%.

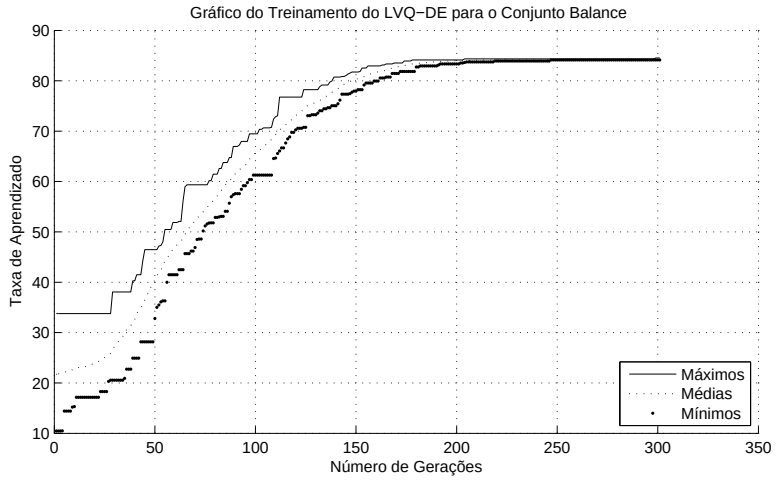
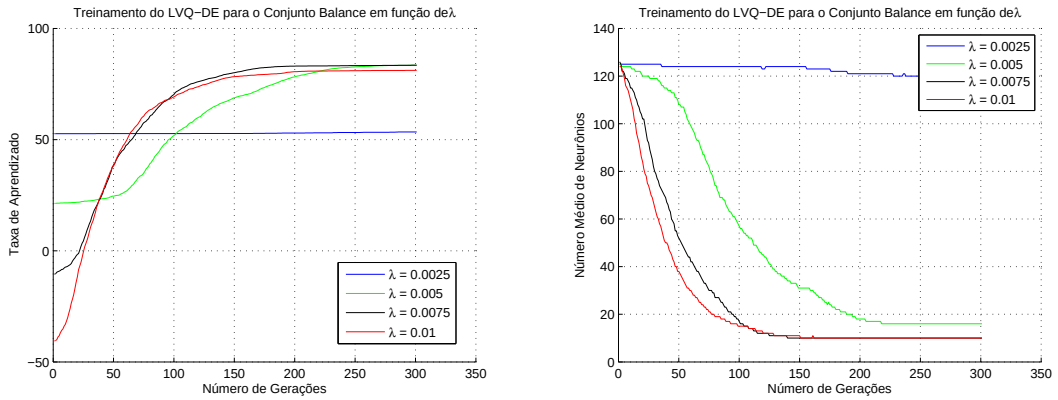


Figura 5.12: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Balance*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.13: Efeito da variação paramétrica de λ sobre conjunto de dados *Balance*.

5.2.6 Conjunto de dados *Wine*

Estes dados são os resultados de análises químicas de vinhos produzidos na mesma região da Itália, mas derivados de três produtores diferentes. A análise determinou as quantidades de 13 componentes encontrados em cada um dos três tipos de vinhos. Os atributos são: 1) álcool, 2) ácido málico, 3) cinza, 4) alcalinidade das cinzas, 5) magnésio, 6) total de fenóis, 7) flavanoids, 8) fenóis não flavonides, 9) proantocianidinas , 10) a intensidade da cor, 11) matiz, 12) OD 280/ OD 315 de vinhos diludos e 13) prolina. Site de referência: *http://archive.ics.uci.edu/ml/datasets/Wine*.

Tabela 5.7: Resultados para o conjunto de dados *Wine*.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**		
			ω_1	ω_2	ω_3
MDC	97,05	3,39	1	1	1
OLVQ	97,05	3,73	4	4	4
LVQ-2.1	94,11	4,08	8	8	8
LVQ-3	97,05	3,57	6	6	6
Soft LVQ	**97,05**	**3,76**	**4**	**4**	**4**
LVQTC	91,17	4,94	8	8	8
LVQ-DE ($\lambda = 0.005$)	**97,05**	**3,76**	**1**	**2**	**1**

Na simulação referente à Figura 5.14 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}005$; $p_{gc} = 0{,}5$; $q_{ger} = 250$ e $n_s = 50$, que resultou em uma solução onde o total de protótipos foi 4 e a taxa de acerto igual a 97,06%.

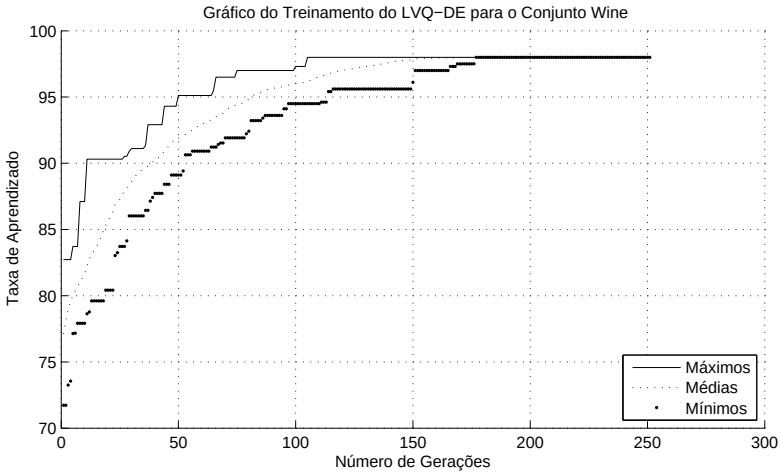
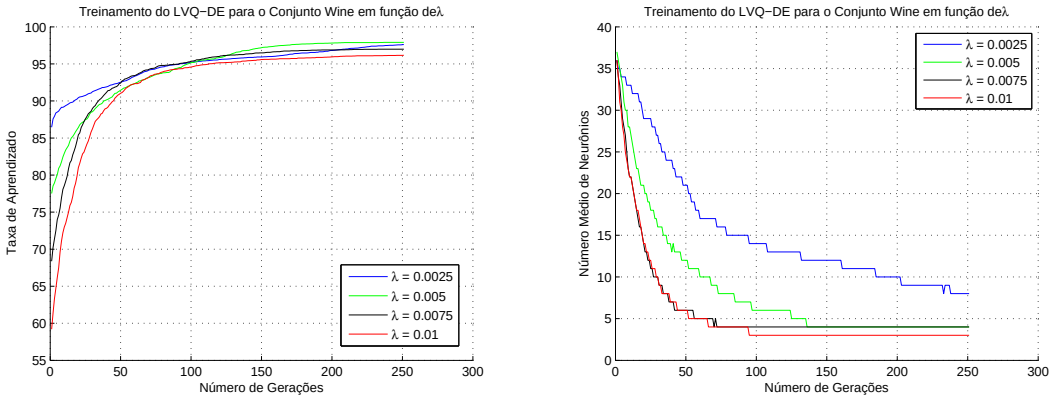


Figura 5.14: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Wine*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.15: Efeito da variação paramétrica de λ sobre conjunto de dados *Wine*.

5.2.7 Conjunto de dados *Vehicle Silhouettes*

Conjunto apresenta objetos $3D$ inseridos em imagens $2D$ por meio da aplicação de um recurso de extração de característica de formas em bordas de objetos $2D$. O objetivo deste conjunto de dados é a classificação a partir do contorno de um dos quatro tipos de veículos utilizados na amostragem, utilizando um conjunto

de características extraídas do contorno. Desta forma, o veículo deverá ser visto de apenas um dos diferentes ângulos possíveis. Os recursos foram extraídos dos contornos que extrai uma combinação de características independentes de escala utilizando dois momentos clássicos de medidas baseadas em variância escalada, assimetria e curtose sobre os maiores e menores eixos, e medidas heurísticas tais como depressões, circularidade, retangularidade e compacidade. Site de referência: *http://archive.ics.uci.edu/ml/datasets/Statlog+(Vehicle+Silhouettes)*.

Tabela 5.8: Resultados para o conjunto de dados *Vehicle Silhouettes*.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**			
			ω_1	ω_2	ω_3	ω_4
MDC	44,31	3,19	1	1	1	1
OLVQ	**67,06**	**2,95**	**20**	**20**	**20**	**20**
LVQ-2.1	59,88	3,85	20	20	20	20
LVQ-3	64,67	2,68	10	10	10	10
Soft LVQ	52,69	4,27	10	10	10	10
LVQTC	56,58	3,04	10	10	10	10
LVQ-DE ($\lambda = 0.0075$)	**65,86**	**2,78**	**2**	**2**	**2**	**2**

Na simulação referente à Figura 5.16 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}0075$; $p_{gc} = 0{,}3$; $q_{ger} = 250$ e $n_s = 30$, que resultou em uma solução onde o total de protótipos foi 17 e a taxa de acerto igual a 68,26%.

5.2.8 Conjunto de dados *Dermatology*

Desenvolvido pelo Dr. H. Altay Guvenir do Department of Computer Engineering and Information Science situado na Bilkent University. O objetivo deste conjunto é determinar o tipo de doenças de pele do tipo eritemato-escamosas. As doenças deste grupo são a psoríase, a dermatite seborreica, líquen plano, pitiríase rósea, dermatite crônica, e pitiríase rubra pilar. Os pacientes foram avaliados clinicamente primeiro com 12 características. Em seguida, amostras de pele foram tomadas para a avaliação de 22 características histopatológicas. Os valores das características histopatológicas são determinadas por uma análise das amostras sob um microscópio. Este conjunto de dados contém 34 atributos, em

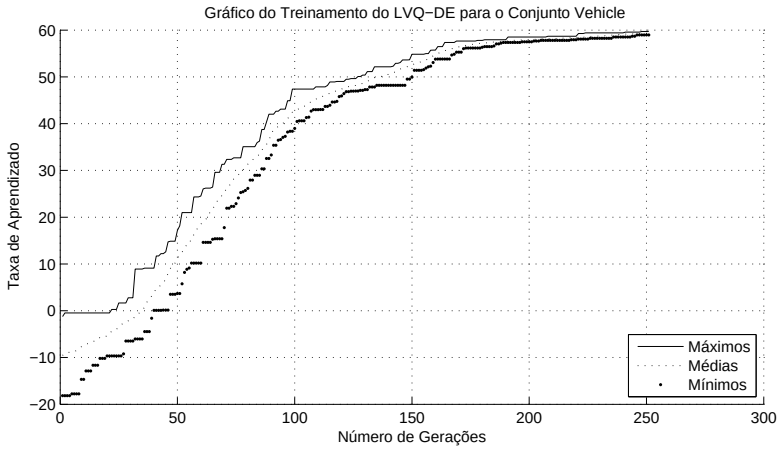
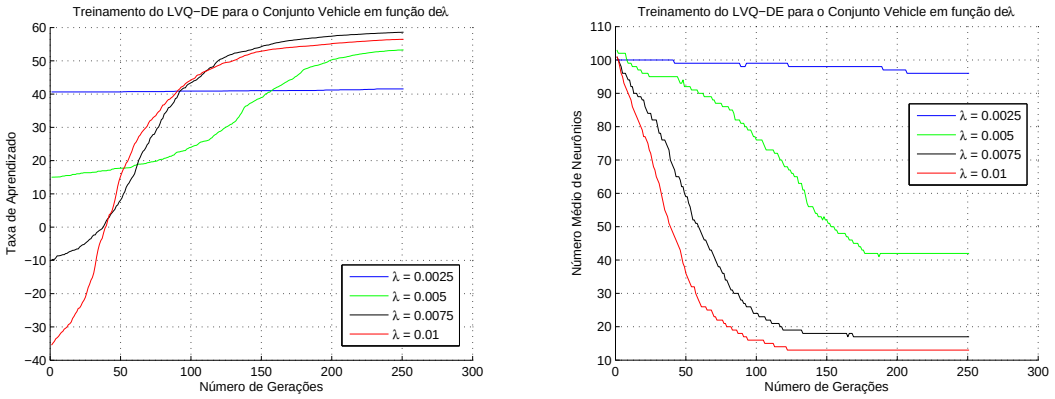


Figura 5.16: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Vehicle Silhouettes*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.17: Efeito da variação paramétrica de λ sobre conjunto de dados *Vehicle Silhouettes*.

que 33 correspondem a valores lineares e um deles é nominal. Site de referência: *http://archive.ics.uci.edu/ml/datasets/Dermatology*.

Na simulação referente à Figura 5.18 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}0075$; $p_{gc} = 0{,}4$; $q_{ger} = 150$ e $n_s = 40$, que resultou em uma solução onde o total de protótipos foi 7 e a taxa de acerto igual a 95,71%.

5.2.9 Conjunto de dados *Wall Following*

Estes dados foram coletados a partir da navegação do robô SCITOS G5 ao longo da parede de uma sala em sentido horário, em quatro etapas, utilizando

Tabela 5.9: Resultados para o conjunto de dados *Dermatology*.

Classificador	Mediana	Desv. Pad.	# Protótipos					
			ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
MDC	95,71	2,03	1	1	1	1	1	1
OLVQ	97,14	2,33	6	6	6	6	6	6
LVQ-2.1	92,85	3,07	8	8	8	8	8	8
LVQ-3	95,71	2,54	6	6	6	6	6	6
Soft LVQ	**98,57**	**1,33**	**6**	**6**	**6**	**6**	**6**	**6**
LVQTC	91,42	3,64	6	6	6	6	6	6
LVQ-DE ($\lambda = 0.0075$)	**97,14**	**2,20**	**1**	**1**	**1**	**1**	**1**	**1**

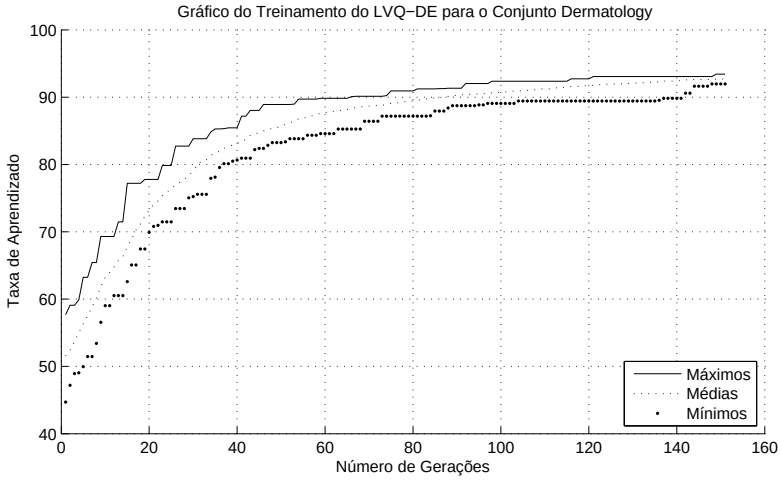
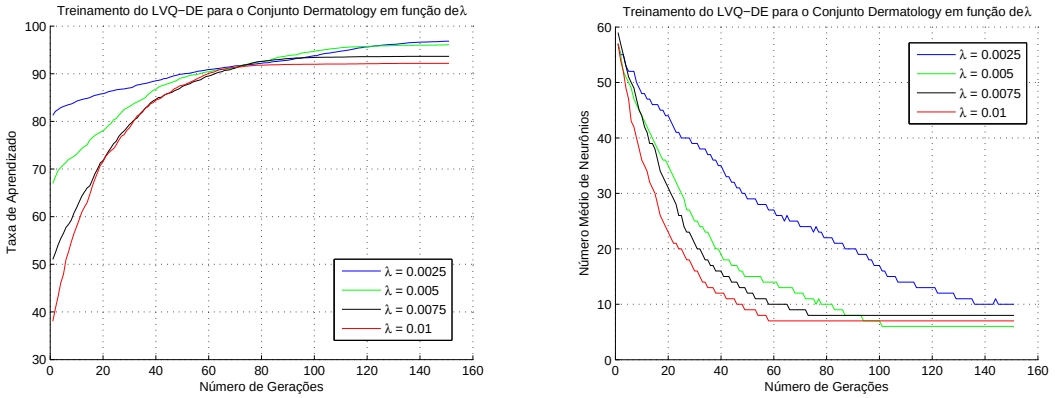


Figura 5.18: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Dermatology*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.19: Efeito da variação paramétrica de λ sobre conjunto de dados *Dermatology*.

24 sensores de ultra-som dispostos em torno do robô. Os valores consistem nas leituras dos valores mínimos dos sensores entre o robô e um obstáculo dentro do arco de graduação de 60 graus em cada sensor, localizados na parte da frente, esquerda, direita e da parte de trás do robô. Neste trabalho, foi utilizado o conjunto de dados reduzido com 4 (paraâmetros) leituras dos sensores como as 'distância frente' , 'distância esquerda' , 'distância direita' e 'distância de traseira'; além do rótulo. O rótulo pode conter 4 diferentes classificações, são elas *Move-Forward*, *Slight-Right-Turn*, *Sharp-Right-Turn* e *Slight-Left-Turn*. Site de referência: *http://archive.ics.uci.edu/ml/datasets/Wall-Following+Robot+Navigation+Data*.

Tabela 5.10: Resultados para o conjunto de dados *Wall Following*.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**			
			ω_1	ω_2	ω_3	ω_4
MDC	60,41	1,26	1	1	1	1
OLVQ	**89,72**	**1,06**	**40**	**40**	**40**	**40**
LVQ-2.1	84,63	4,39	40	40	40	40
LVQ-3	**85,41**	**1,82**	**20**	**20**	**20**	**20**
Soft LVQ	66,00	0,82	20	20	20	20
LVQTC	85,96	1,81	40	40	40	40
LVQ-DE ($\lambda = 0.0025$)	84,22	2,19	24	7	23	4

Na simulação referente à Figura 5.20 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}0025$; $p_{gc} = 0{,}1$; $q_{ger} = 300$ e $n_s = 30$, que resultou em uma solução onde o total de protótipos foi 85 e a taxa de acerto igual a 90,73%.

5.2.10 Conjunto de dados Qualidade do Couro Caprino

Conjunto de dados utilizado para classificação da qualidade de couro de caprinos (QUEIROZ, 2013). Neste trabalho, a partir de cada peça de couro gera-se uma imagem digital em cores, com 3264 linhas e 2448 colunas. Em seguida, cada imagem é convertida para níveis de cinza com 8 bits de resolução, e finalmente reduzida para uma matriz, com tamanho 40×40. O conjunto compreende a extração de características com os valores das variâncias nas colunas e linhas da extrutura 40×40

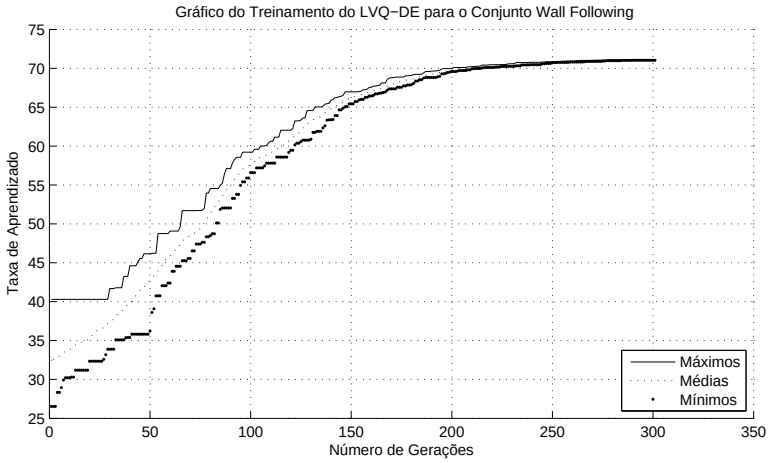
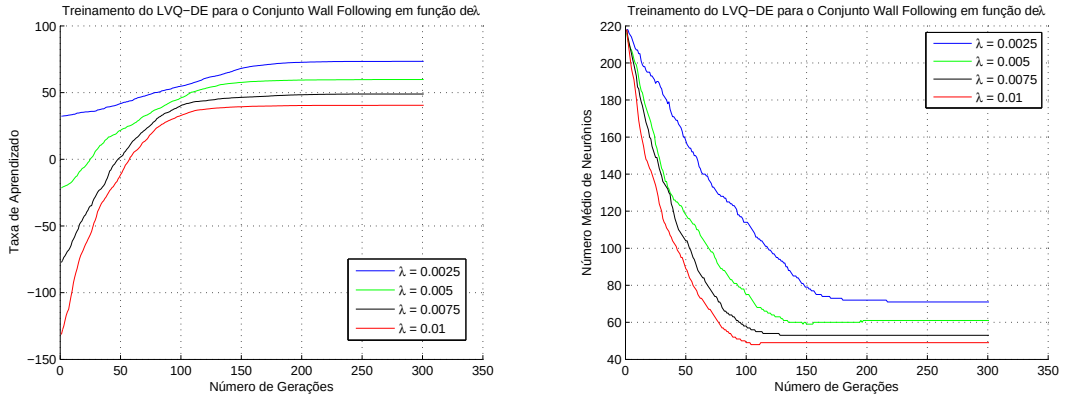


Figura 5.20: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto *Wall Following*.



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.21: Efeito da variação paramétrica de λ sobre conjunto de dados *Wall Following*.

resultante. Desta forma, o conjunto de dados possui 80 atributos e duas classes. As classes estão relacionadas à qualidade do couro, que pode ser superior ou inferior.

Na simulação referente à Figura 5.22 foi utilizada a seguinte configuração paramétrica: $p_r = 0{,}6$; $\beta = 0{,}1$; $\lambda = 0{,}005$; $p_{gc} = 0{,}5$; $q_{ger} = 300$ e $n_s = 50$, que resultou em uma solução onde o total de protótipos foi 3 e a taxa de acerto igual a 83,33%.

5.3 Resultados Obtidos

As Figuras 5.5, 5.7, 5.9, 5.11, 5.13, 5.15, 5.17, 5.19, 5.21 e 5.23 apresentam as simulações sobre o efeito da variação paramétrica de λ, onde os itens (a) mostram

Tabela 5.11: Resultados para o conjunto de dados Qualidade do Couro Caprino.

Classificador	**Mediana**	**Desv. Pad.**	**# Protótipos**	
			ω_1	ω_2
MDC	83,33	4,72	1	1
OLVQ	86,66	5,88	4	4
LVQ-2.1	**86,66**	**4,47**	**4**	**4**
LVQ-3	**86,66**	**4,37**	**4**	**4**
Soft LVQ	86,66	4,98	4	4
LVQTC	81,66	8,56	6	6
LVQ-DE ($\lambda = 0.005$)	83,33	5,69	1	1

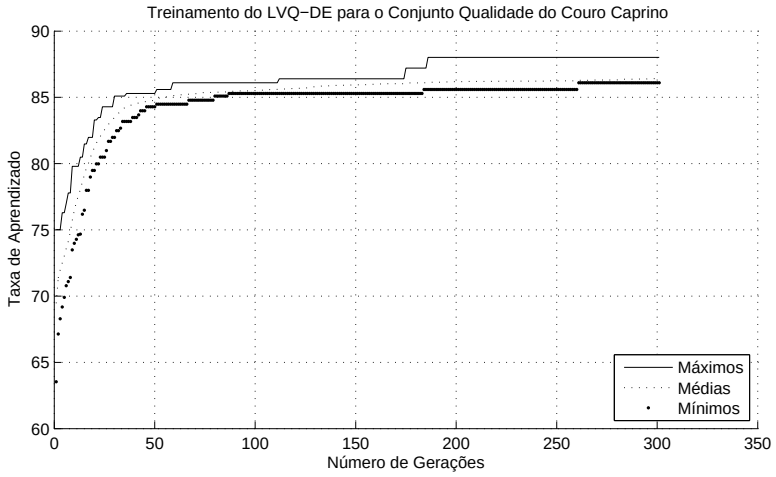
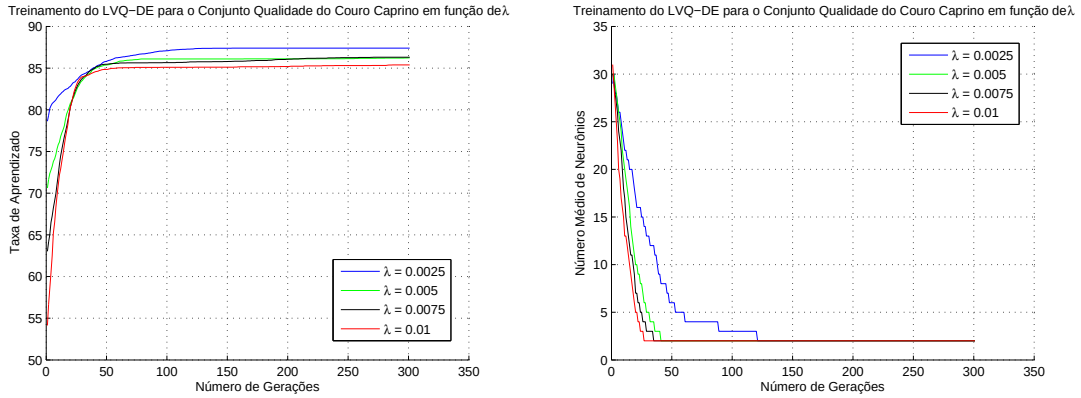


Figura 5.22: Curva de aprendizado típica do algoritmo LVQ-DE, gerada pela avaliação da função de aptidão para o conjunto Qualidade do Couro Caprino.

as curvas de aprendizado ao longo das gerações e os itens (b) exibem o número médio de protótipos dos indivíduos ao longo das gerações. A partir dos resultados apresentados nos itens (b), percebe-se que a utilização de um valor baixo para λ acarreta em soluções com número maior de protótipos. Isto se deve ao fato de estas soluções serem menos penalizadas em decorrência do número de protótipos ativos. Com relação aos itens (a) destas figuras, não houve grandes diferenças nos resultados apresentados, porém, nota-se que ao utilizar um valor baixo para λ há uma melhora sutil na curva de aprendizado. Isto se deve ao uso de um maior número de protótipos, o que acarreta em uma representatividade mais abrangente dos dados de treinamento. Portanto, um valor adequado de λ deve buscar uma



(a) Curva de aprendizado em função de λ.

(b) Quantidade média de protótipos em função de λ.

Figura 5.23: Efeito da variação paramétrica de λ sobre conjunto de dados Qualidade do Couro Caprino.

solução que possua um número reduzido de protótipos, mas que este número seja "grande" o suficiente para garantir um aprendizado adequado, conforme restrições de desempenho estabelecidas pela aplicação.

Com base nos resultados apresentados nas Tabelas 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10 e 5.11, é possível perceber que o método LVQ-DE apresentou desempenho superior para cinco conjuntos de dados, quais sejam: *Vertebral Column*, *Glass*, *Heart*, *Balance* e *Breast Cancer*. Outro resultado muito importante é que o classificador LVQ-DE também apresentou o número protótipos consideravelmente menor que os segundos melhores métodos para todos os conjuntos de dados mencionados, conforme consta na Figura 5.24. Também foi verificado um empate técnico com cinco métodos (incluído o LVQ-DE) para o conjunto *Wine*. Neste caso o algoritmo *Soft* LVQ obteve um desvio padrão um pouco menor que os demais 4 métodos, porém o classificador LVQ-DE (4 protótipos) e o classificador DMC (3 protótipos) obtiveram número de protótipos menores que o algoritmo *Soft* LVQ (12 protótipos). Uma explicação do empate entre os métodos pode ser devido à baixa complexidade do conjunto de dados *Wine*.

Em dois outros casos, para os conjuntos *Vehicle* e *Dermatology*, o método LVQ-DE teve o segundo melhor desempenho entre os outros métodos. Para o conjunto de dados *Vehicle*, apesar de o algoritmo OLVQ ter tido desempenho superior de aproximadamente um ponto percentual, o classificador LVQ-DE modelou o problema com apenas 8 protótipos, enquanto o algoritmo OLVQ necessitou de 80 protótipos. O outro conjunto em que o classificador LVQ-DE ficou em segundo lugar

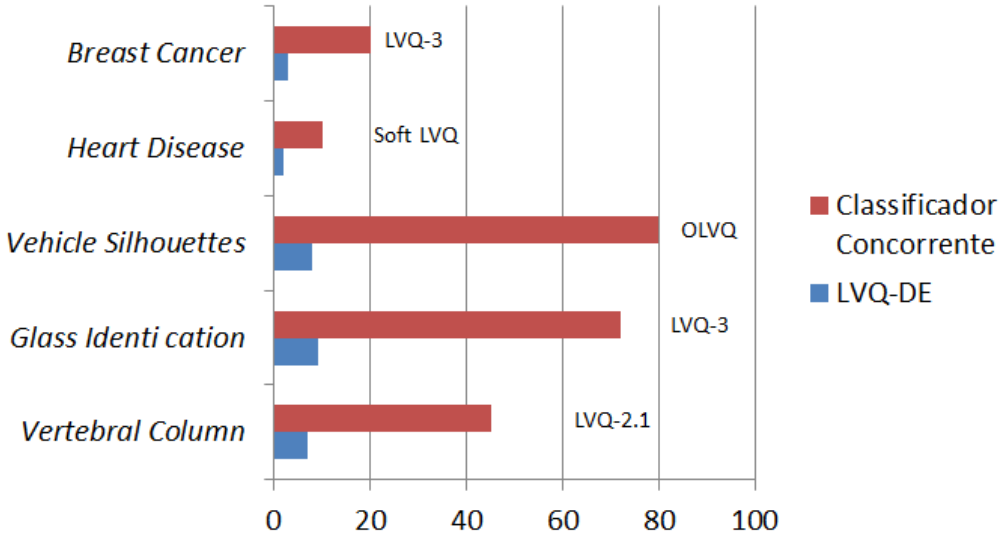


Figura 5.24: Comparativo do quantitativo do número médio de protótipos entre o método LVQ-DE e outros classificadores concorrentes.

foi o *Dermatology*. Neste caso o algoritmo *Soft* LVQ superou o classificador LVQ-DE em $1,4$ pontos percentuais, porém o LVQ-DE modelou o problema com apenas 6 protótipos, enquanto o *Soft* LVQ utilizou 36 protótipos.

Embora o algoritmo LVQ-DE tenha apresentado desempenho inferior nestes dois casos (*Vehicle* e *Dermatology*), ficou evidente que o método proposto atinge desempenhos muito bons com um número muito reduzido de protótipos. Isto pode ser um fator decisivo na escolha deste classificador em aplicações práticas que exijam menor tempo de processamento na etapa de classificação (aplicações em tempo real) e menor alocação de memória (sistemas embarcados com grandes limitações). Esta vantagem fica mais evidente quando é aplicado na modelagem de problemas complexos e/ou com grande quantidade de dados, uma vez que nestes casos os métodos clássicos necessitam de um número alto de protótipos para uma boa modelagem do problema.

Em outros dois conjuntos de dados o classificador LVQ-DE obteve desempenho inferior a outros métodos, foram eles: *Wall-Following* e Qualidade do Couro Caprino. No caso do *Wall-Following*, o algoritmo OLVQ obteve desempenho médio de $89,72\%$, enquanto o do classificador LVQ-DE foi de $84,22\%$. Além de outros três métodos que obtiveram desempenho levemente superior para este conjunto de dados. Para o conjunto Couro Caprino ocorreu resultado semelhante, com o algoritmo LVQ-3 obtendo desempenho de $86,66\%$, enquanto o do classificador LVQ-DE de $83,33\%$.

Da mesma forma, outros três métodos superaram o classificador LVQ-DE para este problema.

O presente capítulo discutiu sobre as metodologias e as escolhas de parâmetros utilizadas nas simulações. Em seguida abordou as técnicas e conjuntos de dados adotados para os experimentos computacionais. Por fim, foram abordados os resultados obtidos por meio de gráficos e tabelas, e explanados os resultados obtidos e suas interpretações.

Capítulo 6

Conclusões e Perspectivas

O método de classificação LVQ-DE foi concebido com o intuito de solucionar uma deficiência encontrada nos métodos LVQ clássicos, que é a dificuldade de configurar o número adequado de protótipos do NPC. Alguns autores aconselham a utilização de 5% do número de amostras do conjunto de dados. Porém, esta abordagem não funciona bem em todos os casos. Assim, os classificadores clássicos LVQ tornam-se de uso complexo e inviável para usuários leigos ou com pouca familiaridade com o método. Por se tratar de um problema de otimização, foi incorporado o algoritmo *Differential Evolution* modificado sobre o método LVQ, sendo inspirado por uma abordagem aplicada em clusterização. Desta forma, foi dada ênfase não só à configuração automática do modelo, mas também ao menor número de protótipos necessário.

Ao longo dos testes, foi comprovado que o método LVQ-DE é capaz de selecionar o modelo de representação com menor número de protótipos automaticamente, sem apresentar perda perceptível no desempenho. A otimização do número de protótipos só foi possível graças à forma como foi desenvolvida a função de aptidão da DE, na qual é beneficiado o modelo com melhor taxa de acerto e o menor número de protótipos em sua representação. Foi criado um hiperparâmetro, chamado de parâmetro de penalização (variando entre $0,01$ até $0,1$, usualmente $0,05$), que exerce influência direta no número final de protótipos do modelo, atuando mais ou menos na penalização da função de aptidão com relação ao número de protótipos. Este parâmetro deve ser selecionado pelo usuário de acordo com as exigências da aplicação e do conjunto de dados que será utilizado. Desta forma, pode-se afirmar que foi solucionado o problema da configuração automática do número de protótipos

por classe do modelo.

Com base nos experimentos elaborados, descritos detalhadamente na seção subsequente, o método LVQ-DE obteve um desempenho superior aos outros nove métodos na maior parte dos experimentos executados. Aplicando os experimentos sobre dez conjuntos de dados (*benckmarks*), o método LVQ-DE obteve o melhor desempenho (ou entre os dois melhores desempenhos) em oito casos (80%). Logo, fica evidente que o LVQ-DE apresentou uma média de desempenho nos testes realizados bastante promissora, se mostrando competitivo tanto com as abordagens clássicas LVQ, quanto com os métodos mais modernos e sofisticados como os LVQTC e *Soft* LVQ. Além disso, em todos os casos apresentados, o método LVQ-DE apresentou o número total de protótipos igual ou inferior aos melhores resultados apresentados pelos outros métodos. Ademais, apresentou, na maior parte dos experimentos o número total de protótipos acentuadamente inferior (menor que 50%) aos demais métodos comparados. Característica importante quando se trata de aplicações com limitações de *hardware* ou de tempo real, que se caracterizam por pouco recursos de memória ou requerem alto desempenho, respectivamente.

Desta forma, o LVQ-DE mostrou-se um método de uso mais prático tanto para usuários especialistas, quanto para leigos. Outro fato importante é que não há a necessidade de retreino após a execução do método LVQ-DE, pois o mesmo já resulta na rede treinada e com seus protótipos ajustados.

6.1 Perspectivas para trabalhos futuros

É fato que o desempenho na etapa de treinamento do LVQ-DE possui um custo computacional mais elevado que os demais métodos clássicos LVQ, por se tratar de um método de otimização baseado em busca estocástica. Mas, por se tratar de um classificador muito recente e pouco investigado, há um vasto campo para melhoria e compreensão do método. Como proposta de trabalhos futuros seria interessante uma abordagem automatizada de seleção de valores para f_e e c_r (adaptativos), na qual estes valores possam ser medidos por meio da análise de convergência da curva de aprendizado no processo de treinamento (calibragem). Esta abordagem visa evitar um *overtraining* em um retreino e também diminuir os tempos de uma provável bateria de experimentos, sem perda da qualidade dos resultados. Outra possibilidade seria adotar esta mesma metodologia da aplicação do *Differential Evolution* modificado em outros algoritmos evolucionários, como por

exemplo o PSO (*Particle Swarm Optimization*), entre outros, bem como realizar testes comparativos.

Este capítulo resgata as motivações que permeiam a criação de novo classificador baseado em protótipos LVQ-DE. Assim como relata a superioridade deste classificador em detrimento de outros modelos, baseando seu desempenho sobre um grupo de conjuntos de dados. Finalmente, aborda as perspectivas de trabalhos com relação ao LVQ-DE e outras possíveis variantes.

Apêndice A

Tabelas de Experimentos em Versão Completa

Esta seção é composta pelas tabelas contendo todos os experimentos realizados com os seguintes algoritmos: LVQ 2.1, LVQ 3, OLVQ, LVQTC, *Soft* LVQ, Centroide NPC e LVQDE. Possui todos os dados estatísticos extraídos nos experimentos, quais sejam: mínimo, máximo, média, mediana, desvio padrão e coeficiente de variação. É constituída por dez tabelas, cada qual contendo os dados dos experimentos relativos a um conjunto de dados.

Tabela 1: Resultados das simulações dos métodos propostos com o conjunto de dados *Vertebral Column*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**58.06**	**85.48**	**73.03**	**74.19**	**4.80**	**0.0658**	**cl1 = cl2 = cl3 = 1**
LVQ-2.1	51.61	83.87	69.87	69.35	5.40	0.0770	cl1 = cl2 = cl3 = 5
	61.29	**85.48**	**74.75**	**75.80**	**4.75**	**0.0636**	**cl1 = cl2 = cl3 = 15**
	56.45	85.48	72.48	72.58	5.81	0.0801	cl1 = cl2 = cl3 = 30
LVQ-3	61.29	87.09	73.74	74.19	5.58	0.0757	cl1 = cl2 = cl3 = 5
	59.67	**88.70**	**74.11**	**74.19**	**5.64**	**0.0761**	**cl1 = cl2 = cl3 = 15**
	58.06	87.09	74.42	75.80	5.96	0.0802	cl1 = cl2 = cl3 = 30
OLVQ	**61.29**	**90.32**	**75.01**	**75.80**	**5.58**	**0.0744**	**cl1 = cl2 = cl3 = 5**
	56.45	85.48	71.32	70.96	6.04	0.0847	cl1 = cl2 = cl3 = 15
	59.67	82.25	71.87	72.58	5.20	0.0724	cl1 = cl2 = cl3 = 30
LVQTC	54.83	79.03	70.32	70.96	6.01	0.0855	cl1 = cl2 = cl3 = 5
	58.06	**82.25**	**71.67**	**70.96**	**5.02**	**0.0701**	**cl1 = cl2 = cl3 = 15**
	58.06	85.48	71.0	70.96	6.17	0.0870	cl1 = cl2 = cl3 = 30
Soft LVQ	48.38	48.38	48.38	48.38	5.74	11866	cl1 = cl2 = cl3 = 5
	58.06	72.58	66.39	66.12	3.62	0.0545	cl1 = cl2 = cl3 = 15
	58.06	**79.03**	**68.65**	**69.35**	**4.51**	**0.0657**	**cl1 = cl2 = cl3 = 30**
LVQDE	67.74	87.09	77.93	77.41	4.46	0.0572	Pen = 0.001; cl1 = 1, cl2 = 2, cl3 = 1
	66.12	87.09	77.32	77.41	5.11	0.0661	Pen = 0.0075; cl1 = 2, cl2 = 2, cl3 = 1
	67.74	**85.48**	**77.22**	**77.42**	**4.03**	**0.0523**	Pen = **0.005**; **cl1 = 2, cl2 = 3, cl3 = 2**
	62.90	85.48	76.80	77.41	4.80	0.0624	Pen = 0.0025; cl1 = 4, cl2 =10, cl3 = 5

Fonte do conjunto de dados: Vertebral Column Data Set, UCI Machine Learning Repository, 2013.

Tabela 2: Resultados das simulações dos métodos propostos com o conjunto de dados *Wall Following*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**57.52**	**63.94**	**60.45**	**60.41**	**1.26**	**0.0208**	**cl1 = cl2 = cl3 = 1**
LVQ-2.1	31.00	44.22	43.03	43.57	2.30	0.0534	cl1 = cl2 = cl3 = cl4 = 10
	37.43	66.69	49.23	49.67	3.93	0.0799	cl1 = cl2 = cl3 = cl4 = 20
	69.81	**90.55**	**83.57**	**84.63**	**4.39**	**0.0525**	**cl1 = cl2 = cl3 = cl4 = 40**
LVQ-3	76.05	86.05	81.85	82.11	2.19	0.0267	cl1 = cl2 = cl3 = cl4 = 10
	79.72	**89.17**	**85.23**	**85.41**	**1.82**	**0.0214**	**cl1 = cl2 = cl3 = cl4 = 20**
	84.77	92.01	88.98	89.03	1.51	0.0170	cl1 = cl2 = cl3 = cl4 = 40
OLVQ	80.27	86.60	83.51	83.57	1.60	0.0191	cl1 = cl2 = cl3 = cl4 = 10
	82.56	90.27	86.64	86.60	1.42	0.0164	cl1 = cl2 = cl3 = cl4 = 20
	87.06	**92.38**	**89.68**	**89.72**	**1.06**	**0.0118**	**cl1 = cl2 = cl3 = cl4 = 40**
LVQTC	67.33	81.28	76.55	76.42	3.14	0.0410	cl1 = cl2 = cl3 = cl4 = 10
	76.33	85.87	81.72	81.97	2.09	0.0255	cl1 = cl2 = cl3 = cl4 = 20
	81.92	**90.0**	**85.99**	**85.96**	**1.81**	**0.0211**	**cl1 = cl2 = cl3 = cl4 = 40**
Soft LVQ	61.00	69.17	66.00	66.00	1.95	0.0296	cl1 = cl2 = cl3 = cl4 = 10
	64.58	**68.16**	**65.95**	**66.00**	**0.82**	**0.0125**	**cl1 = cl2 = cl3 = cl4 = 20**
	49.72	53.48	51.77	52.24	1.32	0.0255	cl1 = cl2 = cl3 = cl4 = 30
LVQDE	82.01	89.90	85.63	85.64	1.64	0.0192	Pen = 0.0075; cl1 =3 , cl2 =9 , = cl3 =2
	82.75	90.55	86.51	86.60	1.76	0.0204	Pen = 0.005; cl1 = 12, cl2 = 4, cl3 = 10, cl4 = 2
	77.43	**88.44**	**84.24**	**84.22**	**2.19**	**0.0260**	**Pen = 0.0025; cl1 = 24, cl2 = 7, cl3 = 23, cl4 = 4**

Fonte do conjunto de dados: Wall-Following Robot Navigation Data Set, UCI Machine Learning Repository, 2013.

Tabela 3: Resultados das simulações dos métodos propostos com o conjunto de dados *Glass Identification*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**65.0**	**90.0**	**76.1**	**75.0**	**5.19**	**0.0682**	**cl1 = ...= cl6 = 1**
LVQ-2.1	50.0	95.0	77.5	77.5	9.46	0.1220	cl1 = ...= cl6 = 4
	67.5	**100.0**	**86.7**	**87.5**	**5.40**	**0.0622**	**cl1 = ...= cl6 = 8**
	70.0	97.5	86.9	87.5	5.55	0.0638	cl1 = ...= cl6 = 12
LVQ-3	72.5	97.5	84.92	85.0	6.23	0.0733	cl1 = ...= cl6 = 4
	72.5	100.0	86.92	87.5	5.24	0.0603	cl1 = ...= cl6 = 8
	70.0	**97.5**	**88.45**	**90.0**	**5.03**	**0.0569**	**cl1 = ...= cl6 = 12**
OLVQ	70.0	97.5	83.82	83.75	5.91	0.0706	cl1 = ...= cl6 = 4
	70.0	100.0	85.62	86.25	5.38	0.0628	cl1 = ...= cl6 = 8
	72.5	**97.5**	**87.75**	**87.5**	**5.19**	**0.0591**	**cl1 = ...= cl6 = 12**
LVQTC	70.0	95.0	81.15	80.0	6.45	0.0795	cl1 = ...= cl6 = 4
	70.0	95.0	84.45	85.0	6.00	0.0710	cl1 = ...= cl6 = 8
	75.0	**95.0**	**87.25**	**87.5**	**5.53**	**0.0634**	**cl1 = ...= cl6 = 12**
Soft LVQ	47.5	65.0	56.58	56.25	4.32	0.0765	cl1 = ...= cl6 = 4
	62.5	80.0	71.58	71.25	4.89	0.0683	cl1 = ...= cl6 = 8
	67.5	**87.5**	**76.66**	**76.25**	**5.14**	**0.0670**	**cl1 = ...= cl6 = 12**
LVQDE	75.0	97.5	91.55	92.5	4.84	0.0528	Pen = 0.01; cl1 =1, cl2 = 2, cl3 = 1, cl4 = 1, cl5 = 1, cl6 = 1
	82.5	**100.0**	**92.15**	**92.5**	**4.60**	**0.0499**	**Pen = 0.0075; cl1 =1, cl2 = 3, cl3 = 1, cl4 = 1, cl5 = 1, cl6 = 2**
	77.5	97.5	89.95	90.0	4.91	0.0545	Pen = 0.005; cl1 =2, cl2 = 4, cl3 = 1, cl4 = 2, cl5 = 1, cl6 = 2
	77.5	97.5	87.9	87.5	5.25	0.0598	Pen = 0.0025; cl1 =5, cl2 = 9, cl3 = 2, cl4 = 3, cl5 = 2, cl6 = 3

Fonte do conjunto de dados: Glass Identification Data Set, UCI Machine Learning Repository, 2013.

Tabela 4: Resultados das simulações dos métodos propostos com o conjunto de dados *Heart Disease Cleveland*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**61.11**	**88.88**	**80.74**	**81.48**	**5.15**	**0.0638**	**cl1 = cl2 = 1**
LVQ-2.1	44.44	83.33	56.77	52.77	13.08	0.2304	cl1 = cl2 = 5
	61.11	88.88	75.48	75.92	5.71	0.0757	cl1 = cl2 = 10
	62.96	**88.88**	**77.31**	**77.77**	**5.09**	**0.0658**	**cl1 = cl2 = 15**
LVQ-3	66.66	88.88	79.68	79.62	4.77	0.0598	cl1 = cl2 = 5
	64.81	**92.59**	**79.46**	**79.62**	**4.68**	**0.0589**	**cl1 = cl2 = 10**
	62.96	90.74	79.27	79.62	5.47	0.0690	cl1 = cl2 = 15
OLVQ	**66.66**	**92.59**	**80.99**	**81.48**	**4.86**	**0.0600**	**cl1 = cl2 = 5**
	66.66	92.59	80.18	79.62	4.82	0.0601	cl1 = cl2 = 10
	62.96	90.74	79.31	79.62	5.51	0.0695	cl1 = cl2 = 15
LVQTC	**64.81**	**88.88**	**77.74**	**79.62**	**5.29**	**0.0681**	**cl1 = cl2 = 5**
	62.96	90.74	76.62	77.77	6.21	0.0810	cl1 = cl2 = 10
	61.11	96.29	77.70	77.77	6.19	0.0796	cl1 = cl2 = 15
Soft LVQ	**70.37**	**92.59**	**83.08**	**83.33**	**5.09**	**0.0613**	**cl1 = cl2 = 5**
	72.22	90.74	81.85	83.33	5.57	0.0681	cl1 = cl2 = 10
	72.22	90.74	82.40	82.40	4.75	0.0577	cl1 = cl2 = 15
LVQDE	**74.07**	**90.74**	**82.96**	**83.33**	**4.48**	**0.0541**	**Pen = 0.01; cl1 = 1, cl2 = 1**
	72.22	92.59	82.29	81.48	4.61	0.0560	Pen = 0.0075; cl1 =2 , cl2 =2
	66.66	90.74	81.18	81.48	5.73	0.0706	Pen = 0.005; cl1 =3 , cl2 =2

Fonte do conjunto de dados: Statlog (Heart) Data Set, UCI Machine Learning Repository, 2013.

Tabela 5: Resultados das simulações dos métodos propostos com o conjunto de dados *Balance*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**62.60**	**85.36**	**73.48**	**73.98**	**4.00**	**0.0544**	**cl1 = cl2 = cl3 = 1**
LVQ-2.1	46.34	88.61	48.55	46.34	7.59	0.1563	cl1 = cl2 = cl3 = 5
	66.66	85.36	76.37	76.42	4.24	0.0556	cl1 = cl2 = cl3 = 10
	71.54	**83.73**	**79.59**	**79.67**	**2.45**	**0.0309**	**cl1 = cl2 = cl3 = 15**
LVQ-3	**78.86**	**91.05**	**85.80**	**85.36**	**2.49**	**0.0290**	**cl1 = cl2 = cl3 = 5**
	76.42	87.80	83.02	82.92	2.66	0.0320	cl1 = cl2 = cl3 = 10
	70.73	82.92	77.83	78.04	3.14	0.0404	cl1 = cl2 = cl3 = 15
OLVQ	**79.67**	**89.43**	**84.74**	**84.55**	**2.58**	**0.0304**	**cl1 = cl2 = cl3 = 5**
	72.35	88.61	81.18	81.30	3.53	0.0435	cl1 = cl2 = cl3 = 10
	67.47	84.55	76.69	76.42	4.18	0.0546	cl1 = cl2 = cl3 = 15
LVQTC	**70.73**	**83.73**	**76.89**	**77.23**	**3.29**	**0.0428**	**cl1 = cl2 = cl3 = 5**
	69.10	82.11	76.50	76.42	3.25	0.0425	cl1 = cl2 = cl3 = 10
	68.29	84.55	76.60	76.42	3.94	0.0514	cl1 = cl2 = cl3 = 15
Soft LVQ	86.17	91.05	88.59	88.61	1.50	0.0169	cl1 = cl2 = cl3 = 5
	85.36	**91.05**	**88.83**	**89.02**	**1.46**	**0.0164**	**cl1 = cl2 = cl3 = 10**
	85.36	91.86	88.34	88.61	1.87	0.0212	cl1 = cl2 = cl3 = 15
LVQDE	76.42	91.05	86.65	86.99	3.06	0.0353	Pen = 0.0025; cl1 = 9, cl2 = 1, cl3=10
	82.92	**92.68**	**88.60**	**89.43**	**2.54**	**0.0287**	**Pen = 0.005; cl1 =4 , cl2 =1, cl3 = 4**
	81.30	92.68	88.74	88.61	2.37	0.0267	Pen = 0.0075; cl3 =1 , cl2 =1, cl3 = 3

Fonte do conjunto de dados: Balance Scale Data Set, UCI Machine Learning Repository, 2013.

Tabela 6: Resultados das simulações dos métodos propostos com o conjunto de dados *Breast Cancer*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**91.85**	**100.0**	**96.37**	**96.29**	**1.79**	**0.0186**	**cl1 = cl2 = 1**
LVQ-2.1	65.92	99.25	94.90	95.55	4.84	0.0510	cl1 = cl2 = 5
	93.33	**100.0**	**96.60**	**96.66**	**1.48**	**0.0153**	**cl1 = cl2 = 10**
	94.07	99.25	96.50	96.29	1.17	0.0122	cl1 = cl2 = 15
LVQ-3	94.07	100.0	97.11	97.03	1.30	0.0134	cl1 = cl2 = 5
	94.07	**100.0**	**97.03**	**97.03**	**1.34**	**0.0138**	**cl1 = cl2 = 10**
	94.07	100.0	96.87	97.03	1.22	0.0126	cl1 = cl2 = 15
OLVQ	92.59	100.0	96.31	96.29	1.58	0.0164	cl1 = cl2 = 5
	93.33	**99.25**	**96.41**	**96.29**	**1.50**	**0.0156**	**cl1 = cl2 = 10**
	92.59	100.0	96.50	96.66	1.50	0.0155	cl1 = cl2 = 15
LVQTC	**86.66**	**99.25**	**95.37**	**96.29**	**2.41**	**0.0252**	**cl1 = cl2 = 5**
	86.66	98.51	94.85	95.55	2.71	0.0285	cl1 = cl2 = 10
	88.88	98.51	95.05	95.55	2.22	0.0233	cl1 = cl2 = 15
Soft LVQ	94.07	98.51	96.49	96.29	1.27	0.0132	cl1 = cl2 = 5
	94.07	**98.51**	**96.49**	**96.66**	**1.28**	**0.0133**	**cl1 = cl2 = 10**
	94.81	99.25	96.51	96.29	1.23	0.0127	cl1 = cl2 = 15
LVQDE	94.07	99.25	97.15	97.03	1.21	0.0125	Pen = 0.0025; cl1 = 1, cl2 = 2
	94.07	**99.25**	**97.15**	**97.03**	**1.14**	**0.0117**	**Pen = 0.005; cl1 =1 , cl2 =1**
	94.07	99.25	97.14	97.03	1.10	0.0114	Pen = 0.0075; cl1 =1 , cl2 =1

Fonte do conjunto de dados: Breast Cancer Wisconsin (Original) Data Set , UCI Machine Learning Repository, 2013.

Tabela 7: Resultados das simulações dos métodos propostos com o conjunto de dados *Vehicle*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**37.72**	**52.69**	**44.40**	**44.31**	**3.19**	**0.0719**	**cl1 = cl2 = cl3 = cl4 = 1**
LVQ-2.1	30.53	65.86	42.62	40.41	7.52	0.1766	cl1 = cl2 = cl3 = cl4 = 10
	42.51	62.87	51.74	50.89	4.74	0.0917	cl1 = cl2 = cl3 = cl4 = 15
	49.70	**67.06**	**59.38**	**59.88**	**3.85**	**0.0649**	**cl1 = cl2 = cl3 = cl4 = 20**
LVQ-3	**59.28**	**71.85**	**64.87**	**64.67**	**2.68**	**0.0413**	**cl1 = cl2 = cl3 = cl4 = 10**
	56.28	73.05	65.11	65.26	3.73	0.0573	cl1 = cl2 = cl3 = cl4 = 15
	59.28	72.45	65.84	65.86	3.19	0.0485	cl1 = cl2 = cl3 = cl4 = 20
OLVQ	59.28	73.65	66.01	65.86	3.22	0.0488	cl1 = cl2 = cl3 = cl4 = 10
	59.28	75.44	65.91	65.56	3.36	0.0510	cl1 = cl2 = cl3 = cl4 = 15
	62.27	**73.65**	**67.29**	**67.06**	**2.95**	**0.0438**	**cl1 = cl2 = cl3 = cl4 = 20**
LVQTC	**49.70**	**62.27**	**56.15**	**56.58**	**3.04**	**0.0541**	**cl1 = cl2 = cl3 = cl4 = 10**
	46.70	65.26	56.50	56.28	3.99	0.0707	cl1 = cl2 = cl3 = cl4 = 15
	50.29	64.07	57.25	56.88	3.45	0.0603	cl1 = cl2 = cl3 = cl4 = 20
Soft LVQ	**47.30**	**62.27**	**53.61**	**52.69**	**4.27**	**0.0797**	**cl1 = cl2 = cl3 = cl4 = 10**
	41.31	61.07	51.59	52.09	4.63	0.0898	cl1 = cl2 = cl3 = cl4 = 15
	36.52	55.68	46.72	45.80	4.68	0.1002	cl1 = cl2 = cl3 = cl4 = 20
LVQDE	54.49	70.65	64.37	64.67	3.66	0.0569	Pen = 0.0025; cl1 = 6, cl2 = 6, cl3 = 7, cl4 = 6
	52.69	74.85	65.62	65.26	3.71	0.0565	Pen = 0.005; cl1 = 4, cl2 = 3, cl3 = 4, cl4 = 3
	59.88	**70.65**	**65.34**	**65.86**	**2.78**	**0.0426**	**Pen = 0.0075; cl1 = 2, cl2 = 2, cl3 = 2, cl4 = 2**

Fonte do conjunto de dados: Statlog (Vehicle Silhouettes) Data Set, UCI Machine Learning Repository, 2013.

Tabela 8: Resultados das simulações dos métodos propostos com o conjunto de dados *Wine*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**88.23**	**100.0**	**95.32**	**97.05**	**3.39**	**0.0356**	**cl1 = cl2 = cl3 = 1**
LVQ-2.1	35.29	100.0	78.11	82.35	15.16	0.1941	cl1 = cl2 = cl3 = 4
	76.47	100.0	92.17	91.17	5.46	0.0592	cl1 = cl2 = cl3 = 6
	82.35	**100.0**	**93.64**	**94.11**	**4.08**	**0.0436**	**cl1 = cl2 = cl3 = 8**
LVQ-3	82.35	100.0	93.94	94.11	3.53	0.0376	cl1 = cl2 = cl3 = 4
	85.29	**100.0**	**95.58**	**97.05**	**3.57**	**0.0374**	**cl1 = cl2 = cl3 = 6**
	85.29	100.0	94.94	94.11	3.51	0.0370	cl1 = cl2 = cl3 = 8
OLVQ	**85.29**	**100.0**	**96.05**	**97.05**	**3.73**	**0.0389**	**cl1 = cl2 = cl3 = 4**
	85.29	100.0	95.17	95.58	3.60	0.0378	cl1 = cl2 = cl3 = 6
	85.29	100.0	94.82	94.11	3.39	0.0357	cl1 = cl2 = cl3 = 8
LVQTC	73.52	100.0	90.64	91.17	5.84	0.0644	cl1 = cl2 = cl3 = 4
	73.52	100.0	90.23	91.17	5.16	0.0571	cl1 = cl2 = cl3 = 6
	79.41	**100.0**	**91.47**	**91.17**	**4.94**	**0.0540**	**cl1 = cl2 = cl3 = 8**
Soft LVQ	**91.17**	**100.0**	**97.54**	**97.05**	**2.67**	**0.0263**	**cl1 = cl2 = cl3 = 4**
	91.17	100.0	96.56	97.05	2.89	0.0300	cl1 = cl2 = cl3 = 6
	88.23	100.0	95.68	97.05	3.06	0.0320	cl1 = cl2 = cl3 = 8
LVQDE	88.23	100.0	95.17	94.11	3.18	0.0334	Pen = 0.0025; cl1 = 1, cl2 = 3, cl3 = 1
	79.41	**100.0**	**95.76**	**97.05**	**3.76**	**0.0393**	**Pen = 0.005; cl1 = 1, cl2 = 2, cl3 = 1**
	82.35	100.0	94.94	94.11	4.16	0.0438	Pen = 0.0075; cl1 = 1, cl2 = 1, cl3 = 1

Fonte do conjunto de dados: Wine Data Set, UCI Machine Learning Repository, 2013.

Tabela 9: Resultados das simulações dos métodos propostos com o conjunto de dados *Dermatology*.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**88.57**	**100.0**	**96.07**	**95.71**	**2.03**	**0.0201**	**cl1 = ...= cl6 = 1**
LVQ-2.1	11.42	95.71	48.94	40.71	27.83	0.5688	cl1 = ...= cl6 = 4
	81.42	98.57	90.57	90.0	3.99	0.0441	cl1 = ...= cl6 = 6
	87.14	**100.0**	**92.88**	**92.85**	**3.07**	**0.0331**	**cl1 = ...= cl6 = 8**
LVQ-3	88.57	98.57	94.62	94.28	2.31	0.0245	cl1 = ...= cl6 = 4
	88.57	**100.0**	**95.71**	**95.71**	**2.54**	**0.0266**	**cl1 = ...= cl6 = 6**
	91.42	100.0	95.37	95.71	2.17	0.0227	cl1 = ...= cl6 = 8
OLVQ	91.42	100.0	96.17	95.71	2.14	0.0223	cl1 = ...= cl6 = 4
	90.0	**100.0**	**96.17**	**97.14**	**2.33**	**0.0242**	**cl1 = ...= cl6 = 6**
	88.57	100.0	95.00	94.28	2.29	0.0241	cl1 = ...= cl6 = 8
LVQTC	80.0	97.14	91.60	91.42	3.68	0.0401	cl1 = ...= cl6 = 4
	82.85	**98.57**	**91.34**	**91.42**	**3.64**	**0.0398**	**cl1 = ...= cl6 = 6**
	84.28	97.14	90.00	90.0	3.54	0.0394	cl1 = ...= cl6 = 8
Soft LVQ	92.85	100.0	97.23	97.14	1.79	0.0184	cl1 = ...= cl6 = 4
	95.71	**100.0**	**98.0**	**98.57**	**1.33**	**0.0135**	**cl1 = ...= cl6 = 6**
	92.85	100.0	98.04	98.57	1.77	0.0181	cl1 = ...= cl6 = 8
LVQDE	88.57	100.0	95.57	95.71	2.20	0.0230	Pen = 0.0025; cl1 =1 , cl2 =2, cl3 =1 , cl4 =1, cl5 =1 , cl6 =1
	90.0	100.0	96.02	95.71	2.31	0.0240	Pen = 0.005; cl1 =1 , cl2 =1, cl3 =1 , cl4 =1, cl5 =1 , cl6 =1
	91.42	**100.0**	**96.17**	**97.14**	**2.20**	**0.0229**	**Pen = 0.0075; cl1 =1 , cl2 =1, cl3 =1 , cl4 =1, cl5 =1 , cl6 =1**

Fonte do conjunto de dados: Dermatology Data Set, UCI Machine Learning Repository, 2013.

Tabela 10: Resultados das simulações dos métodos propostos com o conjunto de dados Couro.

Método	Mínimo	Máximo	Média	Mediana	Desv. Padrão	Coef. Variação	Qtd Prot. Classe
Centroid NPC	**70.0**	**93.33**	**83.56**	**83.33**	**4.72**	**0.0564**	**cl1 = cl2 = 1**
LVQ-2.1	73.33	96.66	85.66	86.66	5.22	0.0610	cl1 = cl2 = 2
	76.66	**96.66**	**86.33**	**86.66**	**4.47**	**0.0518**	**cl1 = cl2 = 4**
	76.66	100.0	85.4	85.0	4.51	0.0528	cl1 = cl2 = 6
LVQ-3	73.33	93.33	85.40	86.66	5.34	0.0625	cl1 = cl2 = 2
	73.33	**93.33**	**85.13**	**86.66**	**4.37**	**0.0513**	**cl1 = cl2 = 4**
	73.33	96.66	85.59	86.66	4.96	0.0580	cl1 = cl2 = 6
OLVQ	66.66	96.66	83.60	83.33	5.82	0.0696	cl1 = cl2 = 2
	70.0	**96.66**	**85.06**	**86.66**	**5.88**	**0.0691**	**cl1 = cl2 = 4**
	73.33	96.66	83.80	83.33	5.21	0.0622	cl1 = cl2 = 6
LVQTC	50.0	93.33	77.86	80.0	11.52	0.1479	cl1 = cl2 = 2
	63.33	93.33	79.73	80.0	7.15	0.0897	cl1 = cl2 = 4
	53.33	**93.33**	**80.60**	**81.66**	**8.56**	**0.1062**	**cl1 = cl2 = 6**
Soft LVQ	76.66	96.66	85.22	85.0	5.44	0.0638	cl1 = cl2 = 2
	76.66	**93.33**	**85.44**	**86.66**	**4.98**	**0.0583**	**cl1 = cl2 = 4**
	76.66	93.33	85.77	86.66	4.54	0.0529	cl1 = cl2 = 6
LVQDE	73.33	96.66	82.93	83.33	5.41	0.0652	Pen = 0.0025; cl1 = 2, cl2 = 3
	66.66	**93.33**	**83.26**	**83.33**	**5.69**	**0.0683**	**Pen = 0.005; cl1 =1 , cl2 =1**
	63.33	93.33	83.46	83.33	6.31	0.0756	Pen = 0.0075; cl1 =1 , cl2 =1

Fonte: Classificação da qualidade de peles de caprinos (QUEIROZ, 2013).

Referências Bibliográficas

aES, P. R. B. G. *Métodos Quantitativos Estatísticos*. [S.l.]: IESDE Brasil S.A., 2008.

AGUAYO, L. *Redes Neurais Competitivas para Detecção de Novidades em Séries Temporais*. Tese (Doutorado) — Universidade Federal do Ceará, Brasil, 2008.

ATTIG, A. Using prototype-based classification for automatic knowledge acquisition. *Pattern Recognition, Machine Intelligence and Biometrics - Springer Berlin Heidelberg*, p. 197–212, 2011.

BEZDEK, J. C.; PAL, N. R. Two soft relatives of learning vector quantization. *Neural Networks*, v. 8, n. 5, p. 729–743, 1995.

CAETANO, D. S. D. *Comparação entre as redes LVQ e MLP no controle de próteses virtuais para membros superiores*. Dissertação (Mestrado) — Universidade Federal de Umberlândia, 2010.

CARPENTER, G. Default ARTMAP. *CAS/CNS Technical Report Series*, n. 008, 2003.

CARPENTER, G.; GROSSBERG, S.; REYNOLDS, J. H. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, Elsevier, v. 4, n. 5, p. 565–588, 1991.

CARPENTER, G. A.; GROSSBERG, S.; MARKUZON, N.; REYNOLDS, J. H.; ROSEN, D. B. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, v. 3, n. 5, p. 698–713, 1992.

CHO, H. joo Lee e S. Application of lvq to novelty detection using outlier training data. *Elsevier Science Pattern Recognition Letters*, v. 27(13), p. 1572–1579, October 2006.

CHUNHONG, W. Research on color recognition of urine test paper based on learning vector quantization (lvq). *International Conference on Instrumentation*

& Measurement, Computer, Communication and Control, IEEE, DOI 10.1109/IMCCC.2012.205, p. 978–0–7695–4935–4/12, 2012.

COELLO, C. A. *Evolutionary Algorithms for Solving Mult-ObjectiveProblems*. [S.l.]: Springer, 2007.

DAS. Two improved differential evolution schemesfor faster global search. *in Proc. 7th CEC*, p. 966–973, 2005.

DAS, S.; ABRAHAM, A.; KONAR, A. Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on Systems, Man, and cybernetics, Part A*, v. 38, n. 1, p. 218–237, 2008.

DUDA, R. O.; HART, P. E. *Pattern Classification and Scene Analysis*. [S.l.: s.n.], 2000.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2nd. ed. [S.l.]: John Wiley & Sons, 2006.

ENGELBRECHT, A. P. *Computational intelligence*. [S.l.]: Library of Congress Cataloging-in-Publication Data, 2007.

G.MOKRYANI, M.-R. *Detection of Inrush Current Based On Wavelet Transform and LVQ Neural Network*. [S.l.: s.n.], 2009.

HAMMER, B. Prototype-based classification of dissimilarity data. *Advances in Intelligent Data Analysis X - Springer Berlin Heidelberg*, v. 7014, p. 185–197, 2011.

HAYKIN. *Processos de Aprendizagem. In: Redes Neurais: Princípios e Prática*. [S.l.]: Porto Alegre: Bookman, 2001a.

HELSINKI. *Bibliography on the self-organizing map (SOM) and learning vector quantization (LVQ)*. 2002. Disponível em: <http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html>.

HUME, T. *Balance Scale - UCI - Machine Learning Repository*. 2014. Disponível em: <http://archive.ics.uci.edu/ml/datasets/Balance+Scale>.

JIANG, Z. Feature data optimization with lvq technique in semantic image annotation. *10th International Conference on Intelligent Systems Design and Applications*, 2010.

JUANG, B.-H.; KATAGIRI, S. Discriminative learning for minimum error classification (pattern recognition). *IEEE Trans. Signal Process*, v. 40, p. pp. 3043–3054, 1992.

KAMBHATLA, N.; LEEN, T. K. Classifying with gaussian mixtures and clusters. *Proc. Neural Inform. Processing Syst. Conf.*, v. 7, p. 681–688, 1995.

KASUBA, T. Simplified fuzzy ARTMAP. *AI EXPERT*, v. 8, p. 18–25, 1993.

KATAGIRI, S. New discriminative training algorithms based on the generalized probabilistic descent method. *IEEE Workshop Neural Networks Signal Processing*, p. 299–308, 1991.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, Springer, v. 43, n. 1, p. 59–69, 1982.

KOHONEN, T. *Learning vector quantization.* 1986.

KOHONEN, T. Improved versions of learning vector quantization. *Proc. IJCNN Int. Joint Conf. Neural Networks*, v. 1, p. 545–550, 1990.

KOHONEN, T. Improved versions of learning vector quantization. In: *Proceedings of the 1990 International Joint Conference on Neural Networks (IJCNN'90)*. [S.l.: s.n.], 1990. v. 1, p. 545–550.

KOHONEN, T. *LVQ Pak: The learning vector quantization program package*. 1995.

KOHONEN, T. *Self-Organizing Maps*. [S.l.]: Berlin, Germany, 1995.

KOHONEN, T. *Self-Organizing Maps*. 2nd extended. ed. Berlin, Alemanha: Springer-Verlag, 1997.

KOHONEN, T. *Self-Organizing Maps*. [S.l.]: Third Edition, 2001.

KOHONEN, T. Learning vector quantization. In: ARBIB, M. A. (Ed.). *The Handbook of Brain Theory, Neural Networks*. 2nd. ed. [S.l.]: MIT Press, 2003. p. 631–635.

KOHONEN, T. Essentials of the self-organizing map. *Neural Networks*, v. 37, p. 52–65, 2013.

KOMORI, T.; KATAGIRI, S. Application of a generalized probabilistic descent method to dynamic time warping-based speech recognition. *Conf. Acoust., Speech, Signal Processing*, p. 497–500, 1992.

KONO, Y. Utilização de rede neural lvq para previsão do nível do rio paraguai. *INPE*, São José dos Campos, p. (INPE–15191–TDI/1300), 2008.

KORDJAZI, N. Gait recognition for human identification using ensemble of lvq neural networks. *2012 International Conference on Biomedical Engineering (ICoBE),*, 27–28 February 2012,Penang.

KUGLER, M. *Uma contribuição ao desenvolvimento de interfaces cerebro-computador utilizando potenciais visualmente evocados*. Dissertação (Mestrado) — CEFET-PR, Curitiba, Fevereiro de 2003.

MCDERMOTT, E.; KATAGIRI, S. Prototype-based minimum classification error/generalized probabilistic descent training for various speech units. *Comput. Speech, Language*, v. 8, p. 351–368, 1994.

MENDES, R. *Population Topologies and Their influence in Particle Swarm Performance*. Tese (Doutorado) — University of Minho, Portugal, 2004.

NETO, G. B. A. R. Wci 04 on the application of ensembles of classifiers to the diagnosis of pathologies of the vertebral column: A comparative analysis. *Latin America Transactions, IEEE*, v. 7 (4), p. 487–496, 2009/8.

NUNES, I. *Redes neurais artificiais: para engenharia e ciências aplicadas*. [S.l.]: São Paulo, 2010.

OLIVEIRA, A. C. M. *Algoritmos Evolutivos Híbridos com Detecção de regiões Promissoras em Espaços de Busca Contínuos e Discretos*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais - INPE, 2004.

PALANIAPPAN, R.; ESWARAN, C. Using genetic algorithm to select the presentation order of training patterns that improves simplified fuzzy ARTMAP classification performance. *Applied Soft Computing*, v. 9, n. 1, p. 100–106, 2009.

PATERLINI, S.; KRINK, T. Differential evolution and particle swarm optimisation in partitional clustering. *Computational Statistics & Data Analysis*, v. 50, p. 1220–1247, 2006.

PEREIRA B. DE B., R. C. R. *Data Mining Using Neural Networks: A Guide for Statisticians.* [S.l.]: Pennsylvania, 2009.

PERNER, P. Detecting the transition stage of cells and cell parts by prototype-based classification. *Advances in Data Mining. Applications and Theoretical Aspects - Springer Berlin Heidelberg*, v. 8557, p. 189–199, 2014.

QIN, A.; SUGANTHAN, P. Self-adaptive differential evolution algorithm for numerical optimization. *In Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, p. 1785–1791, 2005.

QIU-HUA, T. Aplication of lvq neural network combined with the genetic algorithm in acoustic seafloor classification. *Chinese Journal of Geophysics*, v. 50, No.1, p. 291–298, 2007.

QUEIROZ, E. Estudo comparativo de métodos de extração de características para clas-sificação da qualidade de peles de caprinos com opção de rejeição. *CBICCongresso Brasileiro de Inteligência Computacional*, 2013.

RAJASEKARAN, S.; PAI, G. Simplified fuzzy ARTMAP as pattern recognizer. *Journal of computing in civil engineering*, v. 14, p. 92, 2000.

RITTER, H.; SCHULTEN, K. Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. *Biological Cybernetics*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 60, n. 1, p. 59–71, 1988.

SEO, S. Soft nearest prototype classification. *IEEE Transactions on Neural Networks*, n. 2, 2003.

STORN, R.; PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.*, v. 11, p. 341–359, 1997.

TABRIZI, P. R. Using pca and lvq neural network for automatic recognition of five types of white blood cells. *32nd Annual International Conference of the IEEE EMBS Buenos Aires, Argentina*, August 31 – September 4, 2010.

TAVARES, M. *Estatística Aplicada à Administração*. [S.l.]: Departamento de Políticas em Educação a Distância DPEAD, 2007.

XUAN. Network voronoi diagram based range search. *In Proceedings of the International Conference on Advanced information Networking and Applications. IEEE Computer Society, Washington, DC*, p. 741–748, 2009.

YUAN-YUAN, G. Evaluation on life satisfaction of left-behind junior high school children based on lvq network. *2012 8th International Conference on Natural Computation (ICNC 2012)*, 2012.