UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Ranyeri do Lago Rocha

Implementação e Análise de Algoritmos Evolutivos de Classificação em Espaço de Alta Dimensão

Campinas

2017

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Ranyeri do Lago Rocha

Implementação e Análise de Algoritmos Evolutivos de Classificação em Espaço de Alta Dimensão

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Automação.

Orientador: Prof. Dr. Fernando Antonio Campos Gomide

Este exemplar corresponde à versão final da tese defendida pelo aluno Ranyeri do Lago Rocha, e orientada pelo Prof. Dr. Fernando Antonio Campos Gomide

_____

Campinas

2017

Agência(s) de fomento e nº(s) de processo(s): CNPq, 156374/2014-5

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

R582i

Rocha, Ranyeri do Lago, 1990-
Implementação e análise de algoritmos evolutivos de classificação em espaço de alta dimensão / Ranyeri do Lago Rocha. – Campinas, SP : [s.n.], 2017.

Orientador: Fernando Antônio Campos Gomide.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Algoritmos evolutivos. 2. Redes neurais (Computação). 3. Inteligência artificial. 4. Modelos classificadores. 5. Análise dde algoritmos. I. Gomide, Fernando Antônio Campos,1951-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Implementation and analysis of evolving classifier algorithms in high dimensional space
Palavras-chave em inglês:
Evolutionary algorithms
Neural networks (Computing)
Artificial intelligence
Litter models
Analysis of algorithms
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Fernando Antônio Campos Gomide [Orientador]
Sarajane Marques Peres
Rafael Ferrari
Data de defesa: 02-05-2017
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Ranyeri do Lago Rocha - RA: 160928

Data da Defesa: 02 de maio de 2017

Título da Dissertação: "Implementação e Análise de Algoritmos Evolutivos de Classificação em Espaço de Alta Dimensão"

Prof. Dr. Fernando Antonio Campos Gomide (Presidente, FEEC/UNICAMP)
Profa. Dra. Sarajane Marques Peres (EACH/USP)
Prof. Dr. Rafael Ferrari (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Agradecimentos

Agradeço,

a Paula pelos incansáveis incentivos e pela confiança depositada em mim.

a minha família, por toda a ajuda cabível a cada um nesses anos.

ao professor Fernando Gomide, por me aceitar, me inspirar e me recolocar no caminho sempre que preciso.

ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, pelo apoio financeiro.

aos colegas Leandro e Raul pelas contribuições e discussões essenciais. Raul, em especial, pela revisão deste texto.

ao parceiro Amadeu, pelas conversas quase diárias durante todo o tempo dedicado ao mestrado.

aos colegas do Laboratório de Computação e Automação Industrial, LCA, pelas ajudas e companhias.

ao colega Jaime, pela confiança no meu trabalho.

a Faculdade de Engenharia Elétrica e Computação, FEEC, pela oportunidade e suporte em todas as etapas do mestrado.

ao professor Fernando Von Zuben, pela brilhante apresentação dos conteúdos de Rede Neural apresentado em seu curso. Obrigado pela inspiração.

a todos que, por falha de minha memória, possam ser não lembrados nessas linhas. Muito obrigado.

"O espírito humano é mais forte que qualquer remédio.
E é isso que precisa ser alimentado por meio do trabalho,
lazer, da amizade e da família. Isso é o que importa.
Foi disso que nos esquecemos. Das coisas mais simples."
Oliver Sacks

Resumo

Sistemas evolutivos e processamento de dados de alta dimensão são de grande importância prática, atualmente sob intensa investigação. Esta dissertação introduz um neuro classificador evolutivo, avalia seu desempenho usando dados de alta dimensão e compara seu desempenho com classificadores evolutivos e clássicos representativos do estado da arte na área. O neuro classificador processa fluxos de dados continuamente e determina a estrutura de uma rede neural artificial com os respectivos pesos sinápticos. Os resultados de simulação sugerem que o algoritmo proposto é competitivo quando comparado com os modelos evolutivos analisados nesta dissertação. Ele supera, em termos de taxa de classificação, todos os modelos na maioria dos conjuntos de dados considerados. Ainda, o neuro classificador requer um menor tempo de processamento por amostra entre os classificadores evolutivos e os clássicos não evolutivos.

Palavras-chaves: Classificadores evolutivos; redes neurais artificiais; espaço de alta dimensão.

Abstract

Evolving systems and high dimensional stream data processing algorithms are of enormous practical importance, and currently are under intensive investigation. This dissertation introduces an evolving neural classification approach, evaluates its performance using high dimensional data, and compares its performance with evolving and classic classifier algorithms representative of the state of the art. The evolving neural classifier works in one-pass mode to find the neural network structure and its weights using high dimensional stream data. The results achieved by the proposed approach suggests that it is competitive with the evolving models addressed in this dissertation. It outperforms in classification rate all of them in most of the datasets considered. Also, the approach requires the lowest per sample processing time amongst the evolving and classic batch classifiers.

Keywords: Evolving classifier; artificial neural network; high dimensional space.

Lista de ilustrações

Lista de tabelas

Notações

α	parâmetro de proporção da dispersão de grupo em *evolving Neural Network-based Algorithm* (eNNA)
β	limiar de decisão em *evolving Vector Quantization* (eVQ)
$\mathbf{\Theta}$	parâmetros dos consequentes das regras nebulosas
$\mathbf{c}$	centro de grupo
$\mathbf{r}$	dispersão do grupo em *evolving Neural Network-based Algorithm* (eNNA)
$\mathbf{U}$	matriz de pesos da camada de saída do classificador neural
$\mathbf{W}$	matriz de pesos da camada de entrada do classificador neural
$\mathbf{w}$	vetor de pesos (parâmetros da combinação linear)
$\mathbf{\Sigma}$	matriz de dispersão em *evolving Neural Network-based Algorithm* (eNNA)
$\boldsymbol{\sigma}$	desvio padrão do grupo em *evolving Neural Network-based Algorithm* (eNNA)
ν, ϱ, ς	variáveis auxiliares no cálculo da *Recursive Density Estimation* (RDE)
ρ	parâmetro de atualização da dispersão de grupo em *evolving Neural Network-based Algorithm* (eNNA)
ξ	variável de folga em *Support Vector Machine* (SVM)
A, B	matrizes de dispersão em *Semi Random Projection* (SRP)
C	número de classes
D	densidade estimada de dados
L	quantidade de grupos, regras nebulosas ou neurônios ocultos
M	número de dados atribuídos ao grupo em *evolving Neural Network-based Algorithm* (eNNA)
N	número de dados de treinamento

Sumário

1 Introdução

1.1 Motivação

O volume de dados gerados é cada vez maior e mais frequente. Nas indústrias, sensores disponibilizam grande quantidade de informação. Na internet, a crescente oferta de postagens em redes sociais, o compartilhamento de imagens, vídeos e documentos compreendem grande parte do volume de dados disponível para análise atualmente.

Com isso, a busca por técnicas de mineração de dados (*data mining*) eficientes tem aumentado continuamente. A mineração de dados objetiva descobrir padrões e extrair conhecimento presente em massas de dados. Funcionalidades típicas incluem predição, classificação e clusterização, por exemplo (HAN *et al.*, 2011). Classificação é utilizada para atribuir os dados às classes de acordo com os respectivos conteúdos, sendo o foco principal deste trabalho.

Além do volume, a dimensão dos dados em certas aplicações é grande, podendo ser 100-, 1.000- ou 100.000-dimensional. Imagens, vídeos e documentos são exemplos de dados de alta dimensão com alta taxa de geração. No caso de textos da língua portuguesa, que possui aproximadamente 390 mil verbetes (LETRAS, 2009), considerando uma representação sem pré-processamento, um documento poderia ser representado por um vetor com dimensão aproximada de 390 mil. Como outro exemplo, uma imagem definida por uma matriz quadrada com 256 pixels na vertical e 256 pixels na horizontal, pode ser representada por um vetor de dimensão 65 mil, aproximadamente. Mesmo com o pré-processamento para eliminar atributos irrelevantes, a dimensão ainda seria alta.

Usualmente, classificadores processam dados considerando que todos eles estão disponíveis ao mesmo tempo. Quando um grande volume de dados é gerado em um pequeno intervalo de tempo, pode ser inviável ou impraticável armazenar todos os dados para processamento posterior. Alguns casos demandam uma rápida resposta para tomada de decisões, o que inviabiliza a armazenagem de dados para processamento posterior. Nesse cenário, técnicas para mineração de dados sequenciais são mais adequadas, pois elas processam os dados como um fluxo, um a um ou em pequenos blocos. Tais técnicas visam minimizar o armazenamento dos dados, processando-os e analisando-os enquanto disponíveis e descartando-os em seguida.

Nesse contexto, dois tipos de classificadores são importantes: classificadores adaptativos e classificadores evolutivos. O primeiro leva em consideração que sua estrutura é pré-definida e se mantêm fixa durante todo o processamento, adaptando seus parâmetros a cada novo dado apresentado. O segundo evolui sua estrutura ao longo do processo

de aprendizagem juntamente com a adaptação de seus parâmetros. Por exemplo, Liang *et al.* (2006) introduziram uma versão sequencial da Máquina de Aprendizado Extremo (OS-ELM) cuja estrutura é pré fixada durante a aprendizagem e seus parâmetros são atualizados sempre que um dado é apresentado. Lan *et al.* (2009) apresentaram a versão construtiva da Máquina de Aprendizado Extremo *Online* Sequencial (CEOS-ELM) na qual, durante o processo de aprendizagem, novas unidades são adicionadas à estrutura e seus parâmetros atualizados. Estes exemplos apresentam uma visão simplificada de sistemas evolutivos. Segundo Angelov *et al.* (2010), Sistemas Evolutivos Inteligentes (eIS) são aqueles capazes de simultaneamente desenvolver e adaptar sua estrutura e funcionalidades ao longo do tempo. Evoluir pode compreender a compressão ou a expansão da estrutura, sendo essa composta por regras em sistemas nebulosos, neurônios em redes neurais artificias ou ambos em sistemas híbridos. Classificadores evolutivos que atendem essa definição incluem: AnYa-Class (ANGELOV; YAGER, 2011), eClass (ANGELOV; ZHOU, 2008), FLEXFIS-Class (LUGHOFER *et al.*, 2007) e Simpl_eClass (BARUAH *et al.*, 2011). Lughofer (2011b) analisa o comportamento do classificador evolutivo FLEXFIS-Class para processar dados com 17, 52 e 74 atributos. Esta dissertação considera dados de dimensão acima de 10.000 atributos e essa será a ordem de grandeza que aqui define alta dimensão.

1.2 Objetivo

O objetivo desta dissertação é implementar e analisar o desempenho de algoritmos evolutivos de classificação em espaço de alta dimensão. Uma abordagem neuro evolutiva é proposta neste trabalho e seu desempenho é comparado com os classificadores evolutivos baseados em regras nebulosas como AnYa-Class, FLEXFIS_Class e eClass. A abordagem aqui proposta, reúne diferentes conceitos utilizados em diferentes algoritmos evolutivos para definição da estrutura e estimação dos parâmetros.

A análise será baseada no desempenho dos classificadores evolutivos quanto à taxa de classificação, o tempo de execução por amostra e o número de regras nebulosas ou neurônios ocultos. O principal objetivo dessa análise é verificar se esses classificadores são capazes de identificar as classes dos dados em espaços de alta dimensão.

A abordagem neuro evolutiva proposta é também comparada com classificadores não evolutivos clássicos da literatura como a Máquina de Vetores Suporte (SVM), Árvores de Regressão e Classificação (CART), k-*Nearest Neighbor* (kNN) e a Máquina de Aprendizagem Extrema Parcialmente Conectada (PC-ELM). A análise é feita quanto à taxa de classificação.

Ainda, o classificador neuro evolutivo é submetido a um problema de classificação com a distribuição dos padrões variando ao longo do tempo. Essa análise é importante para

algoritmos evolutivos, pois, devido sua capacidade de adaptação, evoluindo sua estrutura e parâmetros, os classificadores evolutivos devem ser capazes de lidar com a mudança da distribuição dos padrões contidos nos dados ao longo do tempo.

1.3 Organização do trabalho

O Capítulo 2 apresenta as definições e características de sistemas evolutivos, enfatizando os classificadores, as etapas de identificação da sua estrutura e a estimação de seus parâmetros.

O Capítulo 3 aborda os conceitos e definições referentes a reconhecimento e classificação de padrões, considerados essenciais em um processo de classificação. Ao longo deste capítulo também são apresentados exemplos de classificadores que processam dados de forma não recursiva.

O Capítulo 4 parte dos conceitos apresentados nos capítulos 2 e 3 para desenvolver o classificador proposto nesta dissertação. O neuro classificador evolutivo (eNNA) é detalhado e aplicado a dados de alta dimensão.

O Capítulo 5 apresenta os resultados computacionais obtidos pelos classificadores evolutivos e não evolutivos aqui tratados. Taxa de classificação, tempo de processamento e complexidade da estrutura do modelo são os valores considerados na comparação. Considera-se dados com e sem mudança de distribuição ao longo do tempo.

Finalmente, o Capítulo 6 conclui a dissertação resumindo suas contribuições e sugerindo itens para investigação futura.

2 Sistemas Evolutivos

Este capítulo apresenta os principais conceitos e características de sistemas evolutivos com ênfase na sua aplicação para classificação. São apresentados métodos para identificação da estrutura de modelos evolutivos, regras para sua evolução ao longo do tempo e algoritmos de estimação de parâmetros usando os dados de entrada.

Segundo Angelov *et al.* (2010), Sistemas Evolutivos Inteligentes (eIS, *evolving Intelligent Systems*) são sistemas capazes de simultaneamente evoluir sua estrutura e respectivas funcionalidades ao longo do tempo. Evoluir neste contexto significa expandir ou comprimir a estrutura e atualizar os parâmetros do sistema frente às mudanças nos dados. Uma característica dos sistemas evolutivos é o processamento dos dados na forma de um fluxo contínuo, um a um ou em blocos, permitindo a aplicação em tempo real. O tratamento em fluxo alivia o armazenamento de dados.

Os sistemas evolutivos utilizam, em sua maioria, uma estrutura composta por regras nebulosas (FRB, *fuzzy rule-based*), redes neurais artificiais (NN, *Neural Network*) (HAYKIN, 1994) ou híbridas neuro-nebulosas. FRB é baseado em conjuntos nebulosos (ZADEH, 1965) nos quais regras SE-ENTÃO compõem o modelo.

Assim como outros métodos de inteligência computacional, eIS são usados em classificação, predição de séries temporais, regressão e agrupamento de dados. Classificadores evolutivos são frequentemente usados em tomada de decisão em sistemas autônomos (e.g. robótica) (ZHOU; ANGELOV, 2007), classificação de textos e imagens (LUGHOFER *et al.*, 2007) e detecção de anomalias em processos industriais (ANGELOV *et al.*, 2006).

Classificadores evolutivos tem a propriedade de desenvolver e auto organizar sua estrutura (ANGELOV *et al.*, 2007). As principais técnicas de identificação da estrutura e estimação dos parâmetros utilizados para modelar sistemas evolutivos são discutidos a seguir.

2.1 Modelagem Evolutiva de Classificadores

No contexto de sistemas evolutivos, mecanismos específicos são utilizados para gradativamente modelar um sistema utilizando os dados de entrada. A capacidade de evoluir um modelo com um mínimo de informação é uma das vantagens dos sistemas evolutivos. Modelos são construídos levando em consideração o histórico de amostras, de forma que somente seja processada a amostra atual a cada instante.

A determinação da estrutura do modelo é feita por meio da partição do espaço de dados, dividindo-o em regiões que contenham dados semelhantes, na maioria dos casos, no

espaço de entrada-saída. Desta forma, a estrutura do modelo reflete a partição do espaço de dados e componentes como regras nebulosas e/ou neurônios são associados às regiões. A Figura 1 mostra a ideia de partição do espaço de dados. Uma técnica utilizada neste contexto é a chamada Estimativa de Densidade Kernel (KDE, *Kernel Density Estimation*) e, em sistemas evolutivos nos quais há um fluxo de dados, utiliza-se uma forma recursiva para estimar a densidade. Um exemplo é a Estimativa de Densidade Recursiva (RDE, *Recursive Density Estimation*) (ANGELOV; FILEV, 2004), detalhada na Seção 2.2.

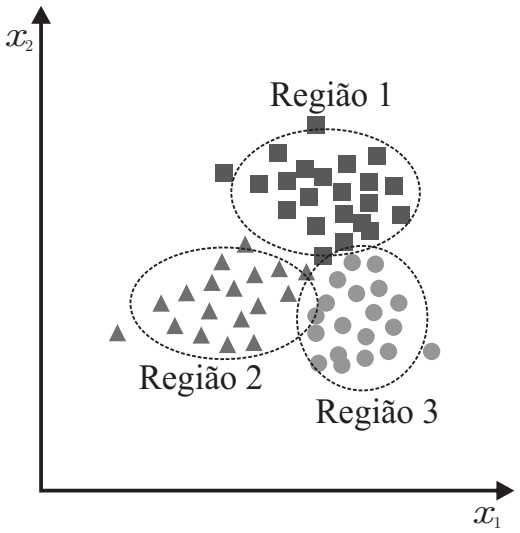


Figura 1 – Partição do espaço dos dados

Uma vez identificada a estrutura do modelo, seus parâmetros devem ser estimados para cada elemento da estrutura do modelo. Em um sentido amplo, estimação de parâmetros é um problema de otimização que visa encontrar modelos melhores a cada novo dado disponível. Um método clássico é o método dos Quadrados Mínimos (LS, *Least Square*) e sua forma recursiva (RLS, *Recursive Least Square*) (YOUNG, 1984) detalhada na Seção 2.3.

2.2 Identificação da Estrutura

A estrutura de um sistema, em geral, é definida por sua representação, seja ela por equações matemáticas, conjunto de regras nebulosas ou uma rede neural (ANGELOV, 2012). O conjunto de regras nebulosas e a rede neural são amplamente utilizados em sistemas evolutivos. A quantidade de regras e o tamanho da rede neural estão diretamente associados à partição do espaço de dados. Conforme mencionado na seção anterior, a utilização de estimativa de densidade recursiva utilizando kernels é uma alternativa para partição do espaço de dados.

A estimativa de densidade kernel é uma forma de estimar a densidade utilizando uma função kernel centrada em cada dado presente e, a estimativa final depende apenas

da relação espacial das amostras. A função kernel é uma função simétrica, unimodal e positiva. Em RDE, diferentes tipos de kernel podem ser utilizados, entre eles Gaussiano, Epanechnikov e Cauchy. Comumente, o kernel Gaussiano é utilizado por KDE. Entretanto, para estimação em modo sequencial, Angelov e Filev (2004) sugerem usar o kernel de Cauchy. A densidade utilizando kernel de Cauchy é definida como:

$$D_k = \frac{1}{1 + \frac{1}{k-1}\sum_{j=1}^{k-1} dist_{jk}^2} \tag{2.1}$$

onde D_k é a densidade estimada no instante k e $dist_{jk}$ é a distância entre o dado no instante k e todos os dados nos instantes passados $j = 1, \ldots, k-1$. O valor $dist_{jk}$ pode ser um distância Euclidiana, como em AutoClass (COSTA *et al.*, 2015) e AnYa-Class (ANGELOV; YAGER, 2011) ou uma medida de similaridade tipo cosseno, como em eClass (ANGELOV; ZHOU, 2008). Outras métricas podem ser usadas, como a Mahalanobis, por exemplo. A expressão (2.1) estima a densidade dos dados no instante k, mas requer que todos os dados de instantes passados estejam disponíveis. Se $\mathbf{x}_k \in \Re^d$ é o dado no instante k, a estimativa de densidade de Cauchy pode ser computada recursivamente utilizando distância Euclidiana da seguinte forma:

$$D_k = \frac{(k-1)}{(k-1)(\mathbf{x}_k^T\mathbf{x}_k + 1) + \varrho_k - 2\nu_k} \tag{2.2}$$

$$\nu_k = \mathbf{x}_k^T\varsigma_k; \quad \varrho_k = \varrho_{k-1} + \mathbf{x}_{k-1}^T\mathbf{x}_{k-1}; \quad \varrho_1 = 0; \quad \varsigma_k = \varsigma_{k-1} + \mathbf{x}_{k-1}; \quad \varsigma_1 = 0;$$

A densidade estimada por (2.2) é uma densidade global pois leva em consideração todos os dados passados. Densidades locais podem ser estimadas para auxiliar na decisão de criação de uma nova região do espaço de dados. Por exemplo, AnYa-Class define uma densidade local, similar à definida por (2.2), mas considerando apenas os dados atribuídos a cada região. Assim, cada uma terá uma densidade local própria. Em outros, como eClass, define-se uma densidade relativa a um dado representativo da região. Esse, denominado ponto focal, tem sua densidade atualizada sempre que cada novo dado é apresentado. A Figura 2 mostra a densidade global do dado $\mathbf{x}_k$ no instante k, densidade essa calculada considerando os dados da região 1 e da região 2. Nos casos em que a densidade local é computada, apenas os dados associados a cada região são considerados no seu cálculo. Os pontos focais podem ser a média dos pontos de cada região, ou o ponto da região com maior densidade global.

Densidades auxiliares, como a local e a do ponto focal, são úteis para decidir quando criar uma nova região no espaço de dados. A regra utilizada em AnYA-Class e eClass segue um padrão: SE (densidade dado atual > densidade dos pontos focais/locais) $ENTÃO$ (Criar nova região). Em AutoClass é utilizada uma média das densidades e as informações da quantidade de *outliers* presentes. Adicionalmente, o nível de ativação de regras nebulosas busca identificar se o novo dado, mesmo com potencial em se tornar uma nova região, pode ser atribuído à região mais próxima.

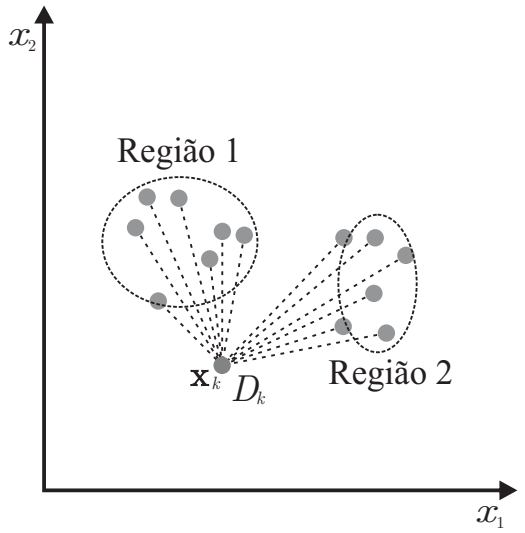


Figura 2 – Cálculo da densidade em um dado

Em outras abordagens, como FLEXFIS-Class (LUGHOFER *et al.*, 2007), a evolução da estrutura é baseada em medidas de distância ou similaridade como as já citadas Euclidiana, Mahalanobis ou cosseno. A Quantização Vetorial (VQ, *Vector Quantization*) agrupa os dados utilizando uma distância para identificar a proximidade entre os dados. Assim como a RDE define um ponto focal para cada região, a VQ define um centro para representar cada grupo. Sistemas evolutivos usam uma versão recursiva da VQ, a Quantização Vetorial Evolutiva eVQ, *evolving Vector Quantization* em (LUGHOFER, 2011a) e originalmente VQ-INC, *Incremental Vector Quantization* em (LUGHOFER, 2008).

O eVQ utiliza a distância entre $\mathbf{c}_{win}$, o centro mais próximo ao dado $\mathbf{x}_k$, e $\mathbf{x}_k$ para decidir se um novo grupo deve ser criado usando as expressões (2.3) e (2.4):

$$\|\mathbf{x}_k - \mathbf{c}_{win}\| \geq \beta \tag{2.3}$$

$$\beta = fac\frac{\sqrt{d}}{\sqrt{2}} \tag{2.4}$$

onde β é um limiar e d a dimensão dos dados. Se a expressão (2.3) é satisfeita, então um novo grupo é criado com $\mathbf{x}_k$ como centro. O valor de β sugerido por Lughofer (2008) visa acomodar a "maldição da dimensionalidade" em que, quanto maior a dimensão dos dados, maior a distância entre eles. O parâmetro fac foi introduzido por Lughofer (2008) como um fator de influência da constante β e, em muitos testes feitos com diferentes conjuntos de dados, teve seu valor estimado como sendo 0.3. Entretanto, para evitar que muitos grupos sejam criados e não ocorra efeitos fortes de sobreajuste nos dados, outros valores de fac no intervalo $[0, 1]$ podem ser escolhidos. O algoritmo proposto no Capítulo 4 utiliza a expressão (2.4) no processo de definição dos grupos.

A última fase do processo de identificação da estrutura comumente utiliza condições para compressão, eliminando partições ou grupos e as correspondentes regras nebulosas ou neurônios. Esse procedimento segue condições que identificam a qualidade das

partições criadas frente às possíveis mudanças na distribuição dos dados. Dois cenários principais são de interesse: o primeiro diz respeito à eliminação de partições que não representam mais o estado corrente (*concept drift*), conforme ilustra a Figura 3; o segundo diz respeito à eliminação de partições que foram criadas por dados considerados "*outliers*", ou seja, dados cujos valores são inconsistentes em relação aos restantes.

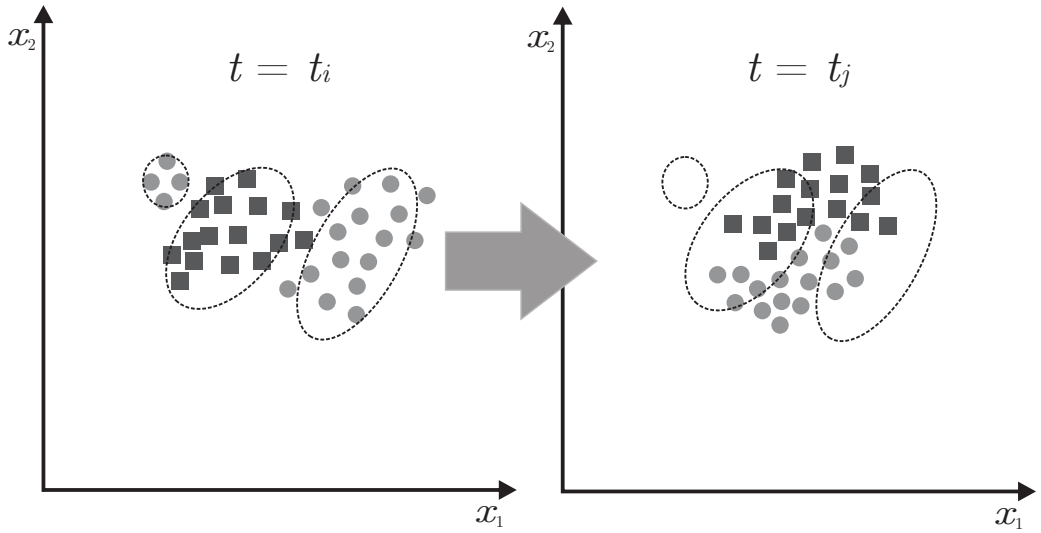


Figura 3 – Variação na distribuição de dados: *concept drift*, $t_i \neq t_j$

A figura 3 ilustra a variação na distribuição dos dados ao longo do tempo. A estrutura evoluída, com três regiões associadas aos dados no instante $t = t_i$, não representa com efetividade os dados com uma nova distribuição no instante $t = t_j$. A estrutura deve, ao longo do tempo, se adaptar às mudanças na distribuição dos dados.

Angelov *et al.* (2010) sugere 3 critérios para monitorar a qualidade dos grupos: 1) quantidade de dados associados a determinado grupo (*Support*); 2) tempo decorrido desde o instante da criação do grupo (*Age*); e 3) acumulado relativo do nível de ativação de determinada regra nebulosa (*Utility*). Esses critérios são responsáveis por identificar, respectivamente, o poder de generalização de um determinado grupo, quão atualizado está o grupo, e o quanto uma regra nebulosa foi utilizada desde a sua geração pelo grupo correspondente.

O método eClass utiliza *Age* para decisão de exclusão de um grupo. O método eTS+ (versão do original eTS (ANGELOV; FILEV, 2005)) utiliza todos os critérios e, se qualquer um deles for satisfeito, então a regra i correspondente é eliminada. Os critérios podem ser:

$$C1 : SE(Utility^i < Utility_{média} - Utility_{desvio}) ENTÃO (L \leftarrow L - 1) \quad (2.5)$$

$$C2 : SE(Age^i > Age_{média} + Age_{desvio}) ENTÃO (L \leftarrow L - 1) \quad (2.6)$$

$$C3 : SE(Support^i < 3) E (k \geq I^i + 10) ENTÃO (L \leftarrow L - 1) \quad (2.7)$$

onde L é o número de grupos em k e I^i é o instante em que o i-ésimo grupo foi criado. $Utility_{média}$ é a média dos valores de *Utility* dos grupos, $Utility_{desvio}$ é o desvio padrão dos valores de *Utility*, $Age_{média}$ é a média das idades dos grupos e Age_{desvio} o desvio padrão das idades dos grupo. Todos os critérios são verificados para cada grupo i. A abordagem proposta nesta dissertação utiliza uma versão do critério 3, conforme (2.7).

Uma vez adaptada a estrutura, sempre que há um novo dado na entrada, prossegue-se com a estimação dos parâmetros da estrutura corrente.

2.3 Estimação de Parâmetros

O modelo evolutivo de um sistema é construído gradualmente e precisa de um mecanismo de estimação de parâmetros que o otimize gradativamente. Na identificação da estrutura, dois tipos são de interesse: regras nebulosas e rede neural. Em particular, uma regra nebulosa tem a forma:

$$Regra^i = SE(Antecedente^i)ENTÃO(Consequente^i) \tag{2.8}$$

Em (2.8), antecedente e consequente possuem formatos específicos. O antecedente de uma regra calcula o grau que um dado de entrada é compatível com um ou mais conjuntos nebulosos. O formato $SE(x_1 \text{ é } A_1)E(x_2 \text{ é } A_2)$, por exemplo, requer a determinação do grau de pertinência de $\mathbf{x} \in \Re^2$ em A_1 e A_2, utilizando as respectivas funções de pertinência. A Figura 4 ilustra como regras nebulosas estão associadas à partição do espaço de dados, onde A_1^1, A_2^1, A_3^1 e A_1^2, A_2^2, A_3^2 são conjuntos nebulosos. Cada regra $SE - ENTÃO$ está associada a uma das regiões.

O consequente de uma regra segue padrões diferentes. Os mais comuns e frequentemente usados são tipo Mamdani (MAMDANI; ASSILIAN, 1975) e Takagi-Sugeno (TAKAGI; SUGENO, 1985). Ambos têm o mesmo padrão de antecedente, mas possuem consequentes distintos. Uma típica regra do tipo Mamdani para um classificador é escrita como:

$$Regra^i = SE(x_1 \text{ é } A_1^i)E(x_2 \text{ é } A_2^i)E \ldots E(x_d \text{ é } A_d^i)ENTÃO(Classe^i) \tag{2.9}$$

Em que $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$ é a amostra, $(x_h \text{ é } A_h^i)$ é o h-ésimo conjunto nebuloso associado à variável x_h na i-ésima regra nebulosa com $i = 1, \ldots, L$ e $h = 1, \ldots, d$. A inferência neste caso é feita por meio da regra de maior ativação τ^i, como:

$$Classe = Classe^{i*}, \qquad i* = \arg\max_{i=1}^{L}(\tau^i) \tag{2.10}$$

com,

$$\tau^i = \prod_{h=1}^{d} A_h^i(x_h), \quad i = [1, L] \tag{2.11}$$

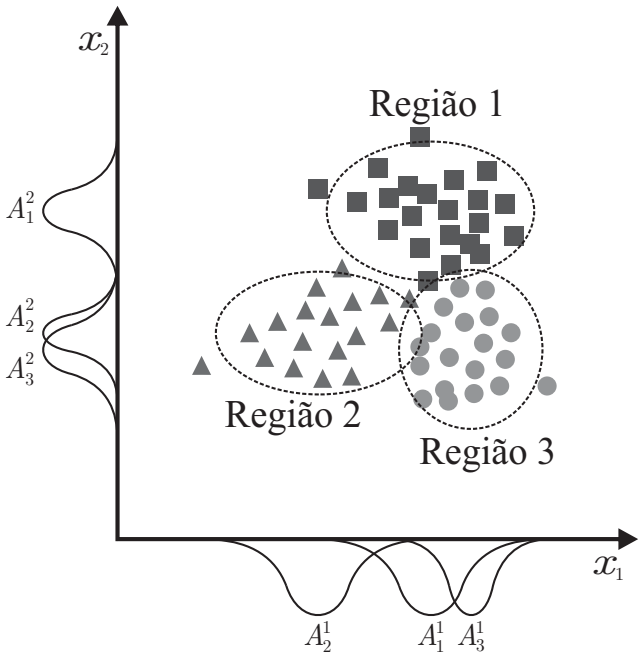


Figura 4 – Definição de conjuntos nebulosos

e,

$$A_h^i(x_h) = \exp\left(-\frac{1}{2}\left(\frac{dist_h^i}{v_h^i}\right)^2\right), \quad i = [1, L], \quad h = [1, d] \tag{2.12}$$

Onde $dist_h^i$ é a distância entre a amostra e o ponto de maior representatividade da i-ésima regra nebulosa $\mathbf{A}^i$ e v_h^i é a dispersão da função de A_h^i. O ponto de maior representatividade é o ponto com maior densidade local, computado utilizando a estimativa de densidade kernel recursiva. A distância $dist$ pode ser euclidiana ou uma medida de similaridade como a cosseno, defina por

$$dist_{cos}(x_k, x_l) = 1 - \frac{\sum_{j=1}^{d} x_{kj} x_{lj}}{\sqrt{\sum_{j=1}^{d} x_{kj}^2 \sum_{j=1}^{d} x_{lj}^2}} \tag{2.13}$$

onde x_k e x_l são dois vetores d-dimensionais.

O modelo Takagi-Sugeno define seu consequente como uma função de ordem zero ou primeira ordem. Ordem zero significa que o consequente é uma função constante $y = cte$, enquanto que a de primeira ordem é tipicamente descrito como:

$$Regra^i = SE(x_1 \text{ é } A_1^i) E (x_2 \text{ é } A_2^i) E \ldots E (x_d \text{ é } A_d^i) ENTÃO (y^i = \hat{\mathbf{x}}^T \Theta^i) \tag{2.14}$$

com $\hat{\mathbf{x}} = [1, x_1, x_2, \ldots, x_d]$ e $\Theta^i = [\theta_0^i, \theta_1^i, \ldots, \theta_d^i]^T$. A saída de uma regra nebulosa i pode ser normalizada, usando:

$$\bar{y}^i = \frac{y^i}{\sum_{i=1}^{L} y^i} \tag{2.15}$$

e a saída, no geral, é computada por:

$$y = \sum_{i=1}^{L} \frac{\tau^i}{\sum_{t=1}^{L} \tau^t} \bar{y}^i \tag{2.16}$$

com τ^i sendo a ativação da i-ésima regra, conforme (2.11). A saída y é usada para definir a qual classe o padrão pertence, entre as classes possíveis, utilizando limiares de decisão.

Dessa forma, os parâmetros $\mathbf{\Theta}$ dos consequentes das i regras podem ser estimados com o método dos Quadrados Mínimos Recursivos ponderado (wRLS, *Recursive Least Square*). Seja k o instante atual, temos:

$$\mathbf{P}_k^i = \mathbf{P}_{k-1}^i - \frac{\lambda^i \mathbf{P}_{k-1}^i \mathbf{x}_k \mathbf{x}_k^T \mathbf{P}_{k-1}^i}{1 + \mathbf{x}_k^T \mathbf{P}_{k-1}^i \mathbf{x}_k} \tag{2.17}$$

$$\mathbf{\Theta}_k^i = \mathbf{\Theta}_{k-1}^i + \mathbf{P}_{k-1}^i \mathbf{x}_k \lambda^i (y_k - \mathbf{x}_k^T \mathbf{\Theta}_{k-1}^i) \tag{2.18}$$

onde $\mathbf{\Theta}_0^i = 0$ e $\mathbf{P}_0^i = \Omega \mathbf{I}$, onde Ω é um escalar positivo com valor elevado. A matriz $\mathbf{P} \in \Re^{(d+1)\times(d+1)}$ e $\mathbf{I}$ é a matriz identidade. O termo λ^i é um fator de ponderação que depende do nível de ativação da i-ésima regra nebulosa. Para $\lambda^i = 1$, a estimação se dá pelo método Quadrados Mínimos Recursivos, com garantia de ótimos globais na solução. Para $\lambda^i = 0$, não há atualização dos parâmetros do modelo. Assim, para $0 < \lambda^i < 1$, a solução garante ótimo locais, garantindo um comportamento adequado dos sub-modelos em cada i-ésima regra (ANGELOV; FILEV, 2004).

O Capítulo 4 apresenta a estrutura de uma rede neural e a atualização dos seus parâmetros utilizando RLS. Embora a estrutura seja composta por neurônios ao invés de regras nebulosas, os parâmetros a serem estimados são também coeficientes de um sistema linear, tal qual o desenvolvido nessa subseção.

2.4 Resumo

Este capítulo resumiu os conceitos de sistemas evolutivos e suas variações como classificadores. Como focos principais, foram abordadas técnicas para identificação da estrutura e estimação dos parâmetros de um classificador. A partição do espaço dos dados utiliza Estimação de Densidade Recursiva (RDE) e Quantização Vetorial evolutiva (eVQ), enquanto que a estimação dos parâmetros é feita através do método Quadrados Mínimos Recursivos (RLS) locais.

3 Classificadores

Este capítulo trata conceitos de reconhecimento com foco em classificação de padrões e nos classificadores que são utilizados como referência nesta dissertação.

Reconhecimento e classificação de padrões, embora pareçam termos semelhantes para um mesmo processo, são na verdade distintos. Para descrever classes, um modelo de classificação precisa extrair e reconhecer os padrões presentes nos dados disponíveis e, a partir deste conhecimento, construir um mecanismo para atribuir outros dados às classes.

O desenvolvimento de um modelo de classificação de dados possui duas fases: fase de treinamento (aprendizagem) e fase de teste (avaliação) do modelo. Na primeira, dados pré selecionados são utilizados para construir um modelo, isto é, aprender os padrões e induzir um mapeamento que discrimine as classes (HAN *et al.*, 2011). Na segunda fase, dados novos são apresentados ao classificador construído. Esses dados são então atribuídos às classes conforme o mapeamento encontrado na primeira fase.

A aprendizagem, ou treinamento, tem o propósito de definir regiões de decisão no espaço de dados. As regiões são delimitadas por fronteiras de decisão caracterizadas por funções. Essas são denominadas funções discriminantes na área de classificação. Se as fronteiras são lineares, isto é, definidas por funções afins, os modelos são chamados de classificadores lineares. Caso contrário, os classificadores são não lineares. Um classificador tem capacidade de generalização se ele classifica corretamente dados não conhecidos à priori. De certa forma, essa visão é intuitiva, mas um modelo com um pequeno erro de aprendizagem pode não ter capacidade de generalização. O quanto o modelo se ajusta aos dados de treinamento na fase de aprendizagem é o que determina sua capacidade de generalização. Tipicamente, tal capacidade de um modelo pode ser monitorada durante a fase de aprendizagem com etapas de validação (PRECHELT, 1998).

Classificadores construídos a partir de dados necessitam assumir fronteiras de decisão definidas por mapeamentos, cujos parâmetros são estimados de forma a minimizar o erro de classificação. Por exemplo, métodos Bayesianos utilizam densidade de probabilidade para determinar a qual classe um dado pertence. Contudo, como essas densidades raramente são conhecidas, métodos de estimação de máxima verosimilhança utilizam os dados de treinamento para estimar parâmetros das densidades de probabilidades.

Classificadores que produzem fronteiras de decisão não lineares, como redes neurais artificias, são o que caracterizam o estado da arte na área. Todos os classificadores citados até aqui necessitam dos dados de treinamento para uma fase de aprendizagem. Presentemente, a demanda é por classificadores cuja fase de aprendizagem é feita sequencialmente para construir e adaptar o classificador gradativamente. Conforme o Capítulo 2,

classificadores evolutivos são classificadores incrementais, com processamento sequencial de dados.

Após uma breve revisão de classificadores lineares, as próximas seções resumem classificadores que processam dados em batelada, como redes neurais, Máquina de Vetores Suporte (SVM, *Support Vector Machine*), Árvore de Regressão e Classificação (CART, *Classification and Regression Trees*), k Vizinhos mais Próximos (kNN, *k Nearest Neighbours*) e uma abordagem recente da máquina de aprendizagem extrema baseada em projeções semi aleatórias (PC-ELM, *Partially Connected Extreme Learning Machine*). Esses são exemplos de classificadores clássicos estado da arte na área.

3.1 Classificadores Lineares

Classificadores lineares são modelos de classificação cujas funções discriminantes são funções afins. Funções discriminantes afins são uma combinação linear dos atributos, que são os componentes do vetor $\mathbf{x}$:

$$g_i(\mathbf{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{id}x_d = \mathbf{w}_i^T\mathbf{x} + w_{i0} \tag{3.1}$$

onde $\mathbf{w}_i = [w_{i1}, \ldots, w_{id}]^T$ é o vetor de pesos (parâmetros da combinação linear) e w_{i0} é um limiar (*bias*) (DUDA *et al.*, 2001). O índice i em (3.1), refere-se à i-ésima função discriminante. Para casos com apenas duas classes, uma função discriminante afim define uma fronteira de decisão que é um hiperplano (hiperplano separador) definido pelo vetor $\mathbf{w}$.

A atribuição de uma classe a um dado de entrada $\mathbf{x} \in \Re^d$, se dá da seguinte maneira:

$$\begin{aligned} \text{classe 1} \quad & \textbf{se} \quad g(\mathbf{x}) > 0, \\ \text{não definida} \quad & \textbf{se} \quad g(\mathbf{x}) = 0, \\ \text{classe 2} \quad & \textbf{se} \quad g(\mathbf{x}) < 0 \end{aligned}$$

Quando $g(\mathbf{x}) = 0$ o dado está na fronteira de decisão. A posição do hiperplano é definida pelo vetor $\mathbf{w}$ e a distância da origem ao hiperplano é dada por $w_0/\|\mathbf{w}\|$. A Figura 5 mostra um hiperplano que caracteriza a fronteira de decisão considerando duas classes e dimensão $d = 3$.

Casos com mais de duas classes exigem tantos hiperplanos quanto o necessário para produzir fronteiras de decisão que particionem o espaço de dados em tantas quantas forem as classes. Por exemplo, emails recebidos por uma pessoa podem ser classificados entre três categorias: *spam*, email pessoal ou email profissional, classe 1, 2 e 3, respectivamente. Assim, uma função $g_i(\mathbf{x})$, $i = 1, 2, 3$, deve ser associada a cada uma das classes, definindo

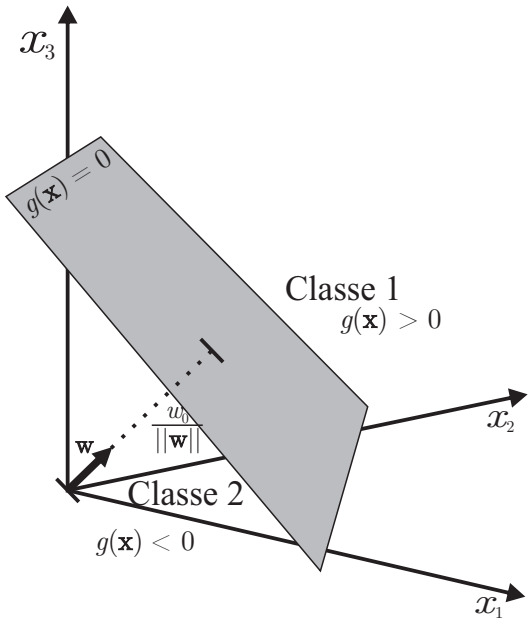


Figura 5 – Hiperplano de um classificador linear e duas classes

fronteiras de decisão entre classes 1-2, 1-3 e 2-3. De forma geral, considerando C classes, a atribuição de um dado $\mathbf{x}$ a uma das C-classes é feita de acordo com a seguinte regra:

$$SE(g_i(\mathbf{x}) > g_j(\mathbf{x}), \quad \forall j \neq i,\, i = 1, ..., C) ENTÃO (\mathbf{x} \rightarrow C_i) \tag{3.2}$$

Em palavras, um dado $\mathbf{x}$ é atribuído à classe C_i se o valor da função discriminante $g_i(\mathbf{x})$ for maior do que todos os valores das funções restantes. A fronteira de decisão entre C_i e C_j é tal que $g_i(\mathbf{x}) = g_j(\mathbf{x})$, conforme ilustra a Figura 6. Em classificação linear o problema central é estimar os vetores $\mathbf{w}_i$ e o valor de w_{i0}, $i = 1, \ldots, C$.

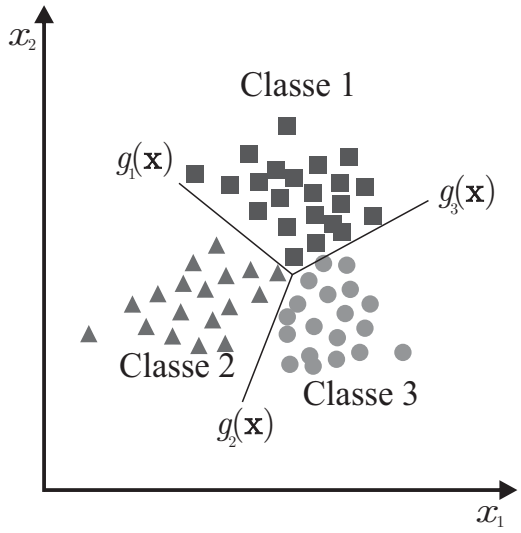


Figura 6 – Hiperplano separador e classificador linear multiclasses

A estimação dos vetores $\mathbf{w}_i$ e dos valores de w_{i0} pode ser feita pelo Método dos Quadrados Mínimos. Usando dados rotulados com as respectivas classes na fase de trei-

namento, temos que:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \tag{3.3}$$

onde,

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \vdots \\ \mathbf{Y}_C \end{bmatrix} \tag{3.4}$$

com $\mathbf{Y} \in \Re^{N \times C}$, em que N é o número de amostras no conjunto de treinamento e C é o número de classes dos dados. As matrizes $\mathbf{Y}_i \in \Re^{N_i \times C}$, $i = 1, \ldots, C$ compreendem as classes de cada uma das $n = 1, \ldots, N$ amostras e N_i o número de amostras na classe i. A formulação considera a classe $\mathbf{y}_n$ como um vetor de dimensão C, ou seja, se classe 1 então $\mathbf{y}_n = [1, 0, 0, .., 0]$, se classe 2 então $\mathbf{y}_n = [0, 1, 0, .., 0]$ e assim por diante, isto é $\mathbf{y}_n \in \{0, 1\}^C$. A matriz $\mathbf{W} \in \Re^{(d+1) \times C}$ compreende os vetores $\mathbf{w}_i$ e os valores w_{i0}, na forma:

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{20} & \ldots & w_{i0} \\ \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_i \end{bmatrix} \tag{3.5}$$

e $\mathbf{X} \in \Re^{N \times (d+1)}$ compreende as N amostras, separadas em i matrizes na forma:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_C \end{bmatrix} \tag{3.6}$$

As matrizes $\mathbf{X}_i$, $i = 1, \ldots, C$ compreendem as amostras de cada classe i. Em cada amostra é inserida uma constante com valor 1, $\mathbf{x}_n = [1, \mathbf{x}]^T$, o que justifica a consideração de $(d+1)$ elementos nas matrizes. A solução que minimiza a soma dos quadrados dos erros é escrita a seguir:

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{X}^\dagger\mathbf{Y} \tag{3.7}$$

onde $\mathbf{X}^\dagger$ é a pseudo inversa da matriz $\mathbf{X}$. A solução (3.7) necessita de um conjunto de treinamento disponível com N amostras armazenadas na matriz $\mathbf{X}$.

Outra forma de estimar o vetor de pesos $\mathbf{w}$ é utilizar métodos iterativos, como gradiente descendente ou Newton. Nesses, uma função de erro é considerada e, iterativamente, o algoritmo busca por $\mathbf{w}$ que minimize o erro. O método de Newton, por ser um método de segunda ordem, geralmente fornece um ganho de desempenho por iteração em comparação ao método do gradiente (DUDA *et al.*, 2001).

3.2 Classificadores Não lineares

Como visto na seção anterior, funções discriminantes lineares são hiperplanos que separam classes. Em muitos casos as fronteiras de decisão não são lineares.

3.2.1 Classificador Neural

Uma rede neural multicamadas (MLP, *Multilayer Perceptron* ou *Multilayer Neural Network*) possui uma estrutura com uma camada de entrada cujas unidades representam cada componente do dado de entrada, uma ou mais camadas ocultas com unidades não lineares e uma camada de saída com unidades em que suas saídas representam as classes. A Figura 7 mostra a estrutura de uma rede MLP com: uma camada entrada com d neurônios, uma camada oculta com L neurônios e uma camada de saída com C neurônios. Os dados de entrada são $\mathbf{x} \in \Re^d$ e os rótulos são $\mathbf{y} \in \{0,1\}^C$. Os pesos das conexões sinápticas entre a camada oculta e a camada de entrada são denotados w_{li}, $l = 1, \ldots, L$ e $i = 1, \ldots, d$ e os pesos entre a camada de saída e a camada oculta por u_{jl}, $j = 1, \ldots, C$. As funções f_1 e f_2 são, respectivamente, as funções de ativação dos neurônios da camada oculta e da camada de saída. A não linearidade da fronteira de decisão é produzida por essas funções, pois elas transformam hiperplanos em hipersuperfícies. A inspiração biológica das redes neurais introduzida por McCulloch e Pitts (1943), levou a utilização do termo neurônio para as unidades das camadas da rede, que assim serão denominados a partir deste ponto. Ainda, todos os neurônios da rede são interligados por meio de conexões sinápticas e os respectivos pesos.

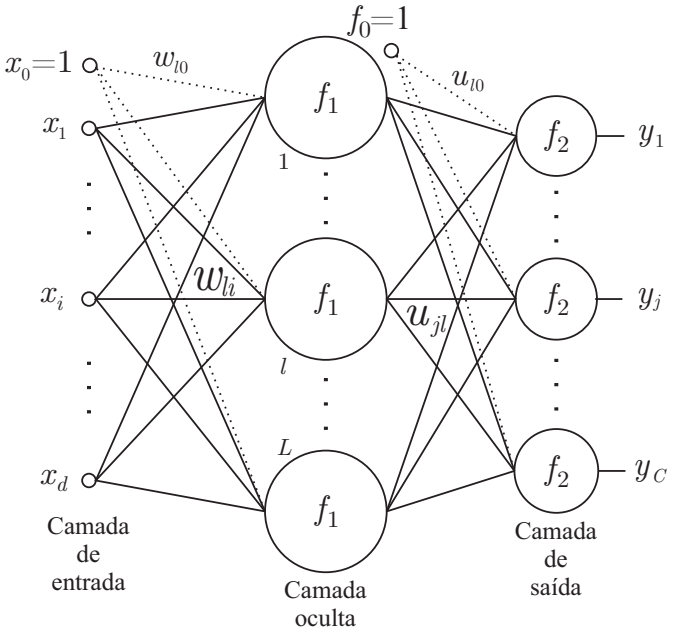


Figura 7 – Rede neural multi camadas

A formulação matemática da rede neural como classificador é a seguinte:

$$y_j = g_j(\mathbf{x}) = f_2(\sum_{l=1}^{L} u_{jl} f_1(\sum_{i=1}^{d} w_{li} x_i + w_{l0}) + u_{j0}), \quad j = 1, .., C \tag{3.8}$$

ou, de forma sucinta, coletando os pesos da camada de entrada na matriz $\mathbf{W} \in \Re^{L \times d}$ e os pesos da camada de saída pela matriz $\mathbf{U} \in \Re^{C \times L}$, temos:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_l \\ \vdots \\ \mathbf{w}_L \end{bmatrix} \qquad \text{e} \qquad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_j \\ \vdots \\ \mathbf{u}_C \end{bmatrix} \tag{3.9}$$

com $\mathbf{w}_1 = [w_{11} \ \ldots \ w_{1d}]$, $\mathbf{w}_l = [w_{l1} \ \ldots \ w_{ld}]$, $\mathbf{w}_L = [w_{L1} \ \ldots \ w_{Ld}]$, $\mathbf{u}_1 = [u_{11} \ \ldots \ u_{1L}]$, $\mathbf{u}_j = [u_{j1} \ \ldots \ u_{jL}]$ e $\mathbf{u}_C = [u_{C1} \ \ldots \ u_{CL}]$. Reescrevendo a equação (3.8):

$$\mathbf{y}(\mathbf{x}) = f_2(\mathbf{U} f_1(\mathbf{W}\mathbf{x} + \mathbf{w}_0) + \mathbf{u}_0), \qquad \mathbf{y} = [y_1, \ldots, y_j, \ldots, y_C]^T \tag{3.10}$$

onde $\mathbf{w}_0 = [w_{10}, \ldots, w_{L0}]^T$ e $\mathbf{u}_0 = [u_{10}, \ldots, u_{C0}]^T$.

Embora seja natural que ambas funções f_1 e f_2 sejam não lineares, é comum escolher $f_1 = f$ e f_2 uma função afim. Esta escolha insere a não linearidade nas fronteiras de decisão e produz uma saída y_j como uma combinação linear das ativações dos neurônios da camada oculta. Assim, a expressão (3.10) torna-se:

$$\mathbf{y} = \mathbf{U} f(\mathbf{W}\mathbf{x} + \mathbf{w}_0) + \mathbf{u}_0, \qquad \mathbf{y} = [y_1, \ldots, y_j, \ldots, y_C]^T \tag{3.11}$$

Em geral, a função de ativação f é a sigmoide ou a tangente hiperbólica:

$$f(\mathbf{h}) = \frac{1}{1 + \exp^{-\mathbf{h}}} \quad \text{ou} \quad f(\mathbf{h}) = \frac{\exp^{\mathbf{h}} - \exp^{-\mathbf{h}}}{\exp^{\mathbf{h}} + \exp^{-\mathbf{h}}} = \tanh(\mathbf{h}) \tag{3.12}$$

onde $\mathbf{h} = \mathbf{h}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{w}_0$. A formulação apresentada em (3.11), mostra aspectos que são de extrema importância no contexto de reconhecimento de padrões. A capacidade de aproximação universal com um conjunto finito de dados é o aspecto essencial. Segundo Hornik (1991), se a quantidade de neurônios disponíveis na camada oculta é suficiente e se suas funções de ativação forem contínuas, não constantes e limitadas, então a rede neural pode aproximar qualquer função contínua em um domínio compacto.

Uma vez definida a estrutura e a função de ativação dos neurônios, o problema central é a estimação dos parâmetros, isto é, os pesos em $\mathbf{W}$ e $\mathbf{U}$, pois esses definem a direção e a posição das hipersuperfícies tal que o erro quadrático médio entre a saída da rede e a saída desejada seja mínimo. Um dos mecanismos para esse propósito é utilizar o método da retropropagação do erro (BP, *Backpropagation*). Tal mecanismo possui duas

etapas: a primeira propaga as entradas mantendo os pesos fixos para calcular a saída da rede e a segunda etapa retropropaga o erro para ajustar todos os pesos da rede ($\mathbf{W}$ e $\mathbf{U}$).

Uma alternativa para ajustar os pesos da camada de saída com neurônios lineares é utilizar o método dos quadrados mínimos. O algoritmo de aprendizagem de redes neurais chamado Máquina de Aprendizado Extremo (ELM, *Extreme Learning Machine*), introduzido por Huang *et al.* (2006), simplifica a necessidade de atualização dos pesos da camada de entrada fixando $\mathbf{W}$ e atualizando os pesos de saída $\mathbf{U}$ com o método dos quadrados mínimos.

Tipicamente, ELM fixa os pesos na camada de entrada aleatoriamente e os mantêm fixos. Como os pesos da camada de saída são calculados por quadrados mínimos, o aprendizado é extremamente rápido pois tem solução analítica (HUANG *et al.*, 2006).

Independentemente do método de estimação dos pesos, as redes neurais artificiais são muito utilizadas devido sua capacidade de formar fronteiras de decisão arbitrárias. As próximas seções apresentam os métodos de classificação representativos do estado da arte na área e que serão utilizados na avaliação de desempenho dos algoritmos tratados nesta dissertação.

3.2.2 Classificador SVM

A Máquina de Vetores Suporte (SVM, *Support Vector Machine*)(VAPNIK, 1995) é um classificador que produz funções discriminantes na forma $g(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})+b$ onde $\phi(\mathbf{x})$ é uma transformação do espaço de atributos e b um limiar (*bias*). A transformação $\phi(\mathbf{x})$ mapeia os dados originais em um espaço de maior dimensão que a original com o objetivo de torná-los linearmente separáveis, o que permite construir nele hiperplanos separadores. A construção de cada hiperplano separador requer determinação dos parâmetros $\mathbf{w}_i$, $i = 1, \ldots, C$ que maximizem uma margem. Por exemplo, se $y_k \in \{1, -1\}$, o classificador é tal que $g(\mathbf{x}_k) \geq 1$ para $y_k = 1$, $g(\mathbf{x}_k) \leq -1$ para $y_k = -1$, o que é equivalente a escrever $y_k g(\mathbf{x}_k) \geq 1$, $k = 1, ..., N$. Quando os dados não são linearmente separáveis no espaço transformado, insere-se uma variável de folga (*slack variable*) que pondere dados que violem a margem. Uma variável de folga ξ_k é definida para cada dado de treinamento, com $\xi_k = 0$ para aqueles que não violam a margem ou estejam exatamente na margem e $\xi_k = |y_k - g(\mathbf{x}_k)|$ para os que violam. Portanto, temos que $y_k g(\mathbf{x}_k) \geq 1 - \xi_k$, $k = 1, ..., N$. A Figura 8 ilustra a ideia da SVM, destacando o hiperplano separador, margem de separação, os vetores de suporte, a variável de folga e o valor da função no hiperplano e nas margens.

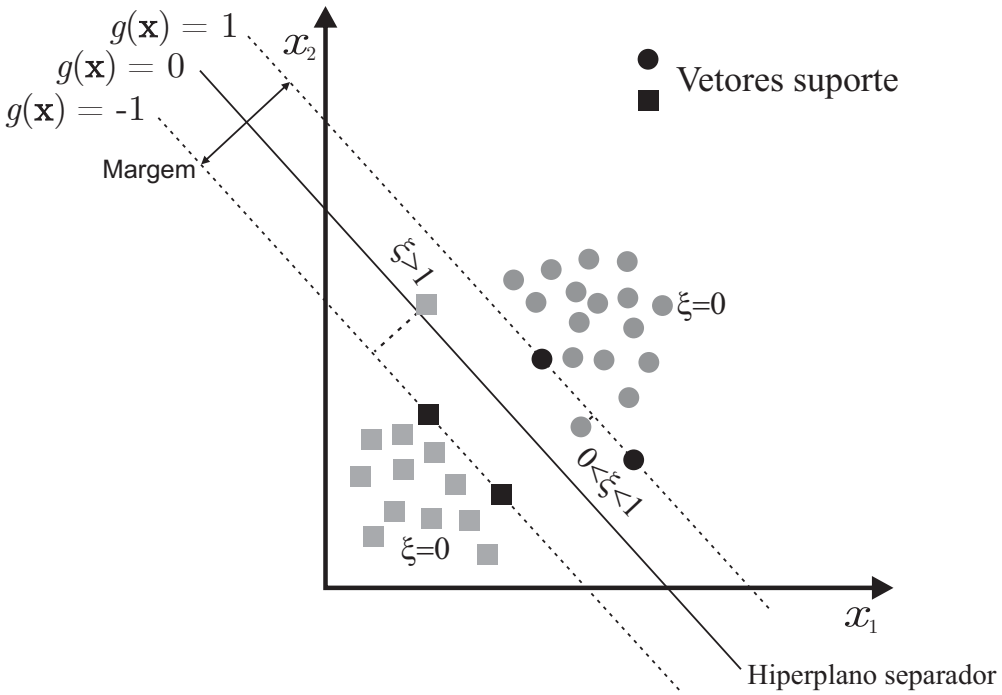


Figura 8 – Hiperplano separador em Máquina de Vetores Suporte - SVM

A SVM pode ser formulada como o seguinte problema de otimização:

$$\begin{aligned} \min_{w,\xi_k} \mathcal{J}(w,\xi_k) =& \frac{1}{2}\|\mathbf{w}\|^2 + R\sum_{k=1}^{N}\xi_k \\ \text{sujeito à:} \quad & y_k g(\mathbf{x}_k) \geq 1-\xi_k,\ k=1,...,N \\ & \xi_k \geq 0 \end{aligned} \tag{3.13}$$

onde R é um parâmetro de regularização que equilibra o erro de treinamento e a complexidade do classificador. A solução de (3.13) decorre das condições de Karush-Kuhn-Tucker e do problema dual equivalente:

$$\begin{aligned} \max \mathcal{L}(\alpha) =& \sum_{k=1}^{N}\alpha_k - \frac{1}{2}\sum_{k=1}^{1}\sum_{l=1}^{1}\alpha_k\alpha_l y_k y_l K(\mathbf{x}_k,\mathbf{x}_l) \\ \text{sujeito à:} \quad & 0 \leq \mathbf{x}_k \leq R \\ & \sum_{k=1}^{N}\mathbf{x}_k y_k = 0 \end{aligned} \tag{3.14}$$

onde $K(\mathbf{x}_k,\mathbf{x}_l)$ é uma função kernel $K(\mathbf{x}_k,\mathbf{x}_l) = \phi(\mathbf{x}_k)^T\phi(\mathbf{x}_l)$. A função kernel tem um papel essencial em SVM. Como citado anteriormente, a função $\phi(\mathbf{x})$ faz o mapeamento dos dados originais para um espaço de maior dimensão onde possam ser separados com um hiperplano. Sem a utilização da função kernel, a equação (3.14) deveria considerar o produto interno $\phi(\mathbf{x}_k)^T\phi(\mathbf{x}_l)$. Entretanto, segundo o teorema de Mercer (MERCER, 1909), se $\phi(\cdot) : \Re^d \rightarrow \Re^q$, com $q \gg d$, então $K(\mathbf{x}_k,\mathbf{x}_l) = \phi(\mathbf{x}_k)^T\phi(\mathbf{x}_l)$ para qualquer função $K(\cdot,\cdot)$ simétrica e semi definida positiva. O resultado deste teorema leva ao conhecido truque

de kernel no qual, ao invés de realizar o produto interno no espaço de alta dimensão, utiliza-se da função kernel no espaço original dos dados, mantendo as considerações de separabilidade dos dados no espaço de alta dimensão. A Figura 9 resume a ideia do truque de kernel, mostrando a transformação dos dados para um espaço de maior dimensão através da função $\phi(\mathbf{x})$.

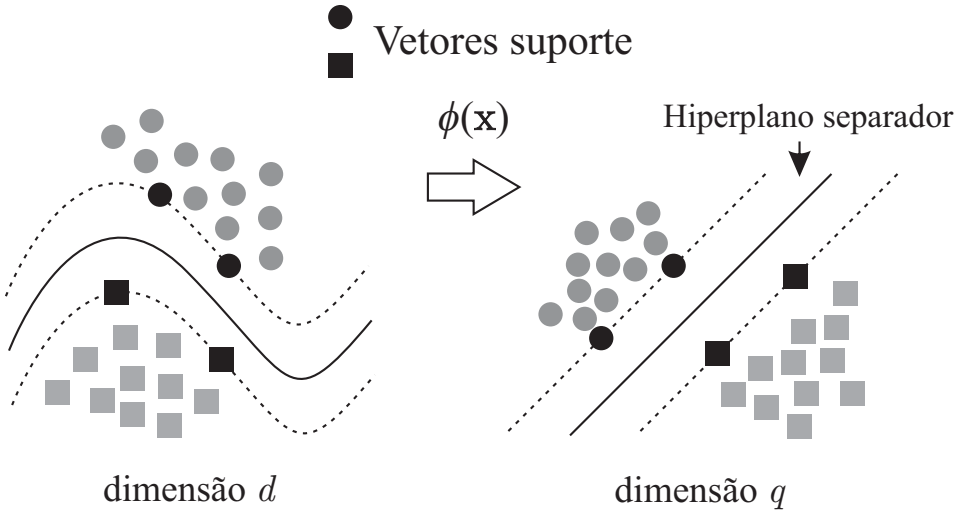


Figura 9 – O truque de kernel

O treinamento é realizado visando encontrar uma fronteira de decisão, tal que a margem seja maximizada. Ao fim, alguns dados do conjunto de treinamento estarão exatamente nas margens, sendo chamados de vetores suporte. Esses representam os dados mais informativos para o processo de classificação, pois apenas eles são utilizados para classificar novos dados.

3.2.3 Classificador k-vizinhos mais próximos

Com menor formalismo matemático e considerando apenas medidas de distância entre os dados, o método k Vizinhos mais Próximos (kNN, *k Nearest Neighbor*) foi proposto por Cover e Hart (1967) como uma extensão do trabalho de Fix e Hodges (1951). O kNN é um exemplo de classificador que, apesar de simples, tem alto custo computacional quando o valor de k e/ou a dimensão dos dados é grande.

Considerando um conjunto de dados de treinamento, o algoritmo kNN determina a distância entre o dado a ser classificado e seus k vizinhos mais próximos para decidir a classe a que pertence. Considerando a distância Euclidiana

$$dist(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{i=1}^{d}(x_{1i} - x_{2i})^2}, \quad \mathbf{x}_1, \mathbf{x}_2 \in \Re^d \tag{3.15}$$

e que N dados compõem o conjunto de treinamento, dados esses devidamente rotulados com as classes, calcula-se para um novo dado $\mathbf{x}$, $dist(\mathbf{x}, \mathbf{x}_i)$, $i = 1, ..., N$. Entre essas

distâncias considera-se as k menores. Se $k = 1$, a classe do dado que define a menor distância é atribuída à classe de $\mathbf{x}$. Para $k > 1$, a classe dos dados com o maior número de instâncias entre os k mais próximos define a classe de $\mathbf{x}$. A Figura 10 ilustra a classificação com kNN considerando $k = 3$.

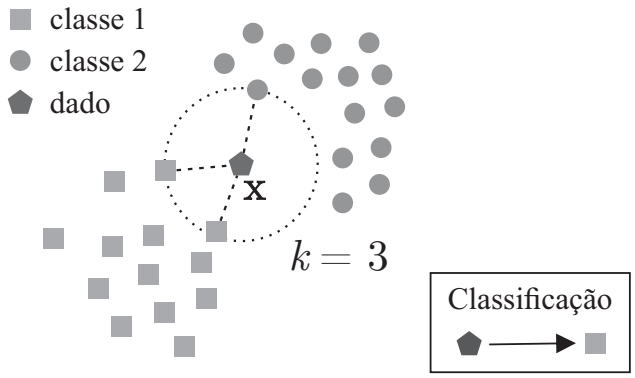


Figura 10 – Classificação com kNN

O calculo da distância entre um e todos os outros dados é um processo custoso e lento. O erro de classificação é diretamente influenciado pelo número de vizinhos k. Para escolher um valor ótimo de k, uma solução é iniciar $k = 1$ e repetir todo o procedimento aumentando progressivamente o valor de k. Considera-se então o valor de k que produz o menor erro (HAN *et al.*, 2011).

3.2.4 Árvore de Regressão e Classificação

Muitos métodos importantes de classificação baseiam-se em árvore de decisão (*Decision Trees*). Dado $\mathbf{x} \in \Re^d$, regras de decisão são construídas utilizando as informações contidas no conjunto de treinamento. Os dados de treinamento são particionados e organizados em uma árvore, tal que o erro em cada folha da árvore seja minimizado, ou seja, cada folha da árvore represente uma classe.

Os elementos que constituem uma árvore são nós, referenciando a pontos de decisão; ramos, definindo os caminhos e folhas representando pontos finais de classificação. O processo de construção de uma árvore de decisão inicia-se com a criação de um nó contendo todos os dados do conjunto de treinamento e uma regra de decisão baseada no atributo mais informativo ("atributo mais informativo" será definido mais adiante). O conjunto de treinamento é dividido de acordo com os ramos e caso o subconjunto de dados contenha apenas dados de uma mesma classe, este torna-se uma folha. Caso contrário, o ramo define um novo nó, com uma nova regra de decisão e os dados novamente particionados. O processo é repetido até que uma condição de parada seja satisfeita.

A Árvore de Regressão e Classificação (CART, *Classification and Regression Tree*) introduzida por Breiman *et al.* (1984) é um exemplo de árvore de classificação binária.

CART é utilizada nas comparações feitas nesta dissertação.

A consideração mais importante quanto à CART é a forma como são feitas as decisões em cada um dos nós na árvore. Esse classificador utiliza o índice de Gini, introduzido em (GINI, 1912). Gini indica o nível de impureza nos nós. Se C é número de classes, tem-se:

$$Gini(N) = 1 - \sum_{c=1}^{C} p_c^2 \tag{3.16}$$

onde o valor p_c corresponde à probabilidade dos dados presentes no nó pertencer à classe c, estimada por $|Classe_{c,N}|/|N|$, ou seja, a proporção de dados da $Classe_c$ quanto ao conjunto de amostras N no nó. A partição com maior $Gini(N)$ é tida como a partição mais pura e novos nós são formados neste ponto.

Diferentes tipos de atributos, contínuos ou discretos, utilizam formas diferentes de decisão. Para o caso discreto, testa-se todas as possibilidades de formação dos conjuntos de decisão. Por exemplo, se o conjunto de atributos é $\{amarelo, redondo\}$, os conjuntos de decisão possíveis são $\{amarelo, redondo\}$, $\{redondo\}$, $\{amarelo\}$ e $\{\}$. Para o caso contínuo, uma forma é testar pontos entre os valores dos atributos em questão. Para um atributo $x = \{x_i, \ldots, x_z\}$, testa-se valores entre x_i e x_z tal que as partições sejam as mais puras (dados da mesma classe) possíveis. Duas partições são formadas em cada teste, $x_1 = \{x_i, \ldots, x_m\}$ e $x_2 = \{x_{m+1}, \ldots, x_z\}$ e após testar todas as partições, escolhe-se aquelas com maior índice de pureza. O ponto de decisão pode ser calculado como $x_d = \frac{x_m + x_{m+1}}{2}$.

O ponto de decisão a ser utilizado é aquele respectivo ao conjunto de decisão, para casos discretos, ou um dos pontos de decisão intermediários, para casos contínuos. É importante lembrar que, em cada nó da árvore, o processo de decisão segue os passos aqui apresentados. A Figura 11 ilustra a ideia de árvore de decisão.

Uma vez definida a árvore, dados não utilizados para a sua construção são usados para avaliar o desempenho do classificador. Em alguns casos, ainda são utilizados métodos de poda de ramos e nós de uma árvore já treinada para reduzir sua complexidade.

3.2.5 Máquina de Aprendizagem Extrema Parcialmente Conectada - PC-ELM

Outra importante abordagem para classificação desenvolvida especificamente para dados de alta dimensão, utiliza conceitos de redução de dimensionalidade e rede neural, processando os dados em batelada. Essa abordagem, chamada de Máquina de Aprendizado Extremo Parcialmente Conectada (PC-ELM, *Partially Connected Extreme Learning Machine*) (ZHAO; MAO, 2015) emprega projeções semi aleatórias (SRP, *Semi Random Projection*) para determinar os pesos da camada de entrada de uma rede ELM, ao invés de aleatoriamente como na ELM.

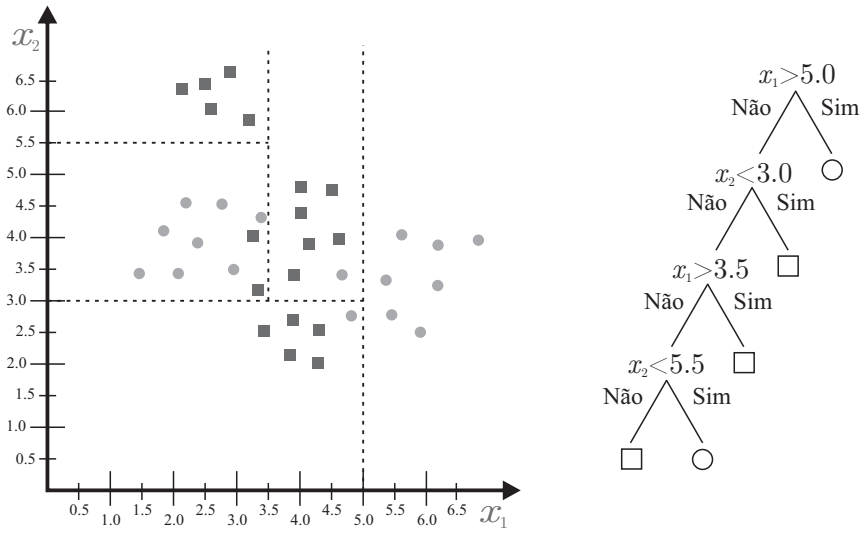


Figura 11 – Árvore de decisão CART

O nome SRP faz referência às fases do algoritmo: primeiro é feita uma amostragem aleatória da dimensão original dos dados e, segundo, uma matriz de pesos da camada de entrada é determinada a partir dos dados de treinamento. A fase aleatória do SRP é baseada no conceito de Projeções Aleatórias (RP, *Ramdon Projections*) cujos fundamentos são apresentados em (JOHNSON; LINDENSTRAUSS, 1984) e mais recentemente em (ACHLIOPTAS, 2001) e (LI *et al.*, 2006). Em RP, define-se uma matriz $\mathbf{W}$ aleatoriamente, seguindo as probabilidades conforme:

$$w_{ij} = \sqrt{s} \begin{cases} 1 & \text{com probabilidade } \frac{1}{2s} \\ 0 & \text{com probabilidade } 1 - \frac{1}{s} \\ -1 & \text{com probabilidade } \frac{1}{2s} \end{cases} \tag{3.17}$$

onde w_{ij} é o elemento da i-ésima linha e j-ésima coluna da matriz $\mathbf{W}$. A constante s é definida no trabalho original com valor 3, assim apenas $\frac{1}{3}$ dos elementos da matriz são não nulos, em média, e consequentemente apenas $\frac{1}{3}$ das entradas são processadas.

Na abordagem de Li *et al.* (2006) demonstra-se que para valores de s iguais a $\sqrt{d}$ ou $\frac{d}{\log d}$, onde d é a dimensão original dos dados, o tempo de processamento é significantemente reduzido com pequena perda em desempenho de classificação. Com esses resultados, foi proposto em (ZHAO; MAO, 2015) que para uma matriz de dados $\mathbf{X} \in \Re^{d \times N}$, apenas $ds = \lfloor \sqrt{d} \rfloor$ dimensões ($ds << d$) são selecionadas aleatoriamente e, então $\hat{\mathbf{X}} \in \Re^{ds \times N}$ é processada utilizando Análise Linear de Discriminantes (LDA, *Linear Discriminant Analysis*) (FISHER, 1936).

Utilizando LDA e as informações das classes de treinamento nos dados, um vetor $\hat{\mathbf{w}}$ de dimensão ds é calculado e expandido para $\mathbf{w}$ com dimensão d. Os valores não selecionados são definidos como nulos após a expansão do vetor $\mathbf{w}$. A matriz de pesos da camada de entrada é definida como: $\mathbf{W} = [\mathbf{w}_1 | \mathbf{w}_2 | \ldots | \mathbf{w}_z]$, onde z é a dimensão do espaço

reduzido. As seguintes expressões são usadas para definir o vetor $\mathbf{w}_i$, $i = 1, \ldots, z$.

$$\text{SRP} = \begin{cases} \mathbf{A} = \sum_{c=1}^{C} n_c (\widehat{\overline{\mathbf{x}}}^c - \widehat{\overline{\mathbf{x}}})(\widehat{\overline{\mathbf{x}}}^c - \widehat{\overline{\mathbf{x}}})^T \\ \mathbf{B} = \sum_{i=1}^{N} (\widehat{\mathbf{x}}^i - \widehat{\overline{\mathbf{x}}}^{c_i})(\widehat{\mathbf{x}}^i - \widehat{\overline{\mathbf{x}}}^{c_i})^T + \eta \mathbf{I}_{d_s} \end{cases} \tag{3.18}$$

$$\mathbf{A}\varphi = \lambda \mathbf{B}\varphi \quad (\mathbf{a}) \qquad \widehat{\mathbf{w}}_i = \sqrt{\lambda_1}\boldsymbol{\varphi}_1 \quad i = 1, \ldots, z \quad (\mathbf{b}) \tag{3.19}$$

$\mathbf{A} \in \Re^{ds \times ds}$ e $\mathbf{B} \in \Re^{ds \times ds}$ são matrizes de dispersão, N é o número de dados de treinamento, C é o número de classes, n_c o número de amostras da classe c, $\hat{\mathbf{X}} \in \Re^{ds \times N}$ é a matriz de dados após a seleção aleatória das entradas, $\widehat{\overline{\mathbf{x}}}$ é o vetor média de todas as amostras, $\widehat{\overline{\mathbf{x}}}^c$ é o vetor de média das amostras da classe c, c_i denota o rótulo da classe da amostra i e $\eta \mathbf{I}_{d_s}$ representa o termo de regularização. A solução se da por meio do problema do autovalor generalizado, representado na equação (3.19) parte (a) e o vetor $\hat{\mathbf{w}}_i$ é então calculado pela equação (3.19) parte (b), onde λ_1 e $\boldsymbol{\varphi}_1$ significam autovalor de maior magnitude e autovetor associado, respectivamente. Em (WELLING, 2005) o autor demonstra a solução do problema de otimização, dados as matrizes $\mathbf{A}$ e $\mathbf{B}$, como um problema do autovalor generalizado. Os passos desde a seleção aleatória da matriz $\hat{\mathbf{X}} \in \Re^{ds \times N}$ até a definição do vetor $\mathbf{w}_i$ são repetidos z vezes, formando a matriz $\mathbf{W}$.

Definidos os pesos da camada de entrada, PC-ELM utiliza o método dos quadrados mínimos para estimar os pesos da camada de saída, similarmente à ELM na seção 3.2.1.

3.3 Resumo

Este capítulo apresentou os classificadores não evolutivos representativos do estado da arte em classificação de padrões. Classificadores lineares utilizando funções discriminantes lineares estimam hiperplanos como fronteira de decisão entre dados linearmente separáveis. Classificadores neurais inserem não linearidade necessária em muitas aplicações práticas para a melhor separação dos dados cujas fronteiras de decisão são não lineares. Métodos alternativos como SVM, kNN, CART e PC-ELM utilizam técnicas para separação do espaço de dados baseados em otimização, medidas de distância, índice de impureza nas partições e redução de dimensionalidade.

4 Neuro Classificador Evolutivo para Espaço de Alta Dimensão

Este capítulo sugere um classificador neural evolutivo para dados de alta dimensão (eNNA, *evolving Neural Network-based Algorithm*). Baseado em sistemas evolutivos e redes neurais artificiais, o eNNA processa os dados como um fluxo contínuo. O eNNA opera em duas etapas: a primeira particiona o espaço de entrada criando grupos elipsoidais associados às classes e a segunda estima os parâmetros dos grupos e da estrutura evoluída.

Modelos evolutivos não têm sido aplicados a problemas com dados de alta dimensão como os tratados nesta dissertação, particularmente quando os dados possuem alto grau de esparsidade. Contudo algumas tentativas já foram consideradas recentemente.

Lughofer (2011b) inclui no método FLEXFIS-Class um processo de seleção de atributos que estima pesos entre 0 e 1 para cada um deles, sugerindo indiretamente um meio para redução de dimensionalidade. Apesar de serem considerados de alta dimensão, os três conjuntos de dados utilizados *CD Imprint*, *Eggs* e *Spam-base* possuem, respectivamente, 74, 17 e 57 atributos. Angelov *et al.* (2007) utiliza o método eClass e quatro diferentes variantes no conjunto *CD Imprint*, mas não menciona o caso de dados de alta dimensão. Wang *et al.* (2013) apresenta um método para evolução da estrutura de modelos fuzzy para problemas de alta dimensão. Entretanto, os autores consideram dados com poucos atributos, como o *MPG* com 7 atributos, *Boston housing data* com 13 atributos e *California Census 1990* com 9 atributos.

4.1 Classificador Evolutivo

O classificador evolutivo (eNNA) processa dados esparsos de alta dimensão, tipicamente com mais de 10 mil atributos e não utiliza técnicas específicas de redução de dimensionalidade como projeções aleatórias, análise de componentes principais, análise de discriminantes lineares, entre outros. O classificar eNNA evolui sua estrutura e estima seus parâmetros de forma recursiva sem a necessidade de armazenar todos os dados. Isto é, primeiro o eNNA define sua estrutura sequencialmente durante um número pré estabelecido de dados. Após a determinação da estrutura, essa é mantida fixa, mas seus parâmetros são adaptados continuamente utilizando o fluxo de dados.

A primeira etapa define a estrutura inicial do classificador, adaptativamente, inserindo e/ou retirando grupos da estrutura de acordo com o fluxo de dados. Essa etapa corresponde à evolução da estrutura do modelo, definida pelos grupos formados. A estrutura pode ser novamente evoluída a partir de um determinado momento, frente à grandes

mudanças nos dados, tal que a estrutura evoluída não mais consiga generalizar a um nível satisfatório. A Figura 12 ilustra a ideia de uma estrutura evoluída pelo classificador evolutivo para dados de dimensão 2. O número de passos na primeira etapa é parâmetro

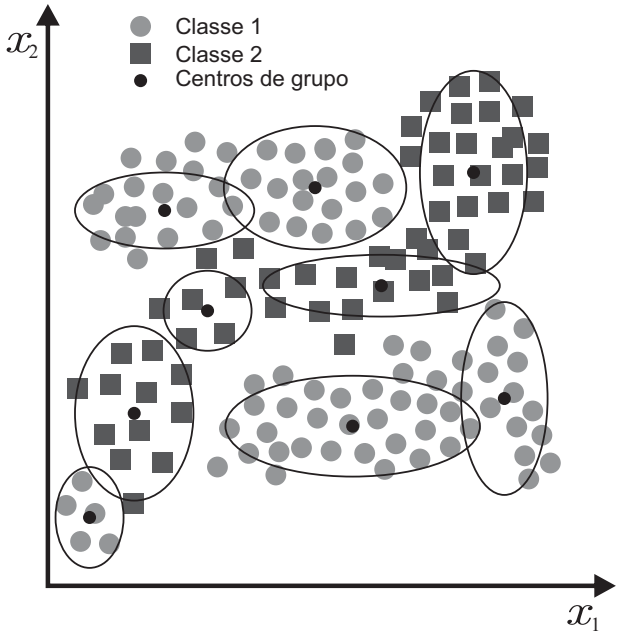


Figura 12 – Estrutura evoluída pelo classificador evolutivo

livre e influencia o desempenho do modelo e a demanda computacional. Uma vez definida a estrutura, os parâmetros dos grupos que compõem o classificador são estimados para adaptá-lo ao estado corrente. Diferente da estrutura do modelo, que pode permanecer a mesma em um período de tempo, os parâmetros são sempre atualizados. A figura 18, seção 4.3, mostra a arquitetura do eNNA.

4.2 Evolução da Estrutura

O processo de construção da estrutura do classificador consiste no particionamento do espaço de dados usando um algoritmo de agrupamento de dados. A maioria dos modelos evolutivos considera partição no espaço de entrada e saída. Para uma entrada $\mathbf{x}$ e saída correspondente $\mathbf{y}$, a partição é feita considerando dados na forma $\mathbf{z} = [\mathbf{x}^T; \mathbf{y}^T]^T$. O eNNA considera somente a entrada $\mathbf{x}$ para a partição do espaço e utiliza a saída $\mathbf{y}$, se disponível, para verificar se a partição a qual foi associado corresponde ou não à classe correta. A etapa de evolução da estrutura ocorre para um número W_{init} de dados, correspondente aos passos $k = 1, 2, \ldots, k_{init}$.

O algoritmo de agrupamento de dados usado pelo eNNA é baseado no algoritmo utilizado em eMG (LEMOS *et al.*, 2011), modificado para tratar dados de alta dimensão. Enquanto eMG considera grupos não necessariamente paralelos aos eixos coordenados, eNNA considera apenas grupos paralelos aos eixos. Essa modificação implica na redução

do custo computacional para dados de alta dimensão. O algoritmo baseia-se na distância de Mahalanobis e assume grupos elipsoidais. A distância de Mahalanobis entre $\mathbf{x}$ e $\mathbf{c}$ é:

$$M(\mathbf{x}, \mathbf{c}) = (\mathbf{x} - \mathbf{c})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{c}) \tag{4.1}$$

onde $\mathbf{c} \in \Re^d$ é o centro do elipsoide e $\mathbf{\Sigma} \in \Re^{d \times d}$ é a matriz de dispersão. Em espaços de alta dimensão o cálculo de $\mathbf{\Sigma}^{-1}$ é custoso. Em particular, como em sistemas evolutivos o cálculo deve ser feito recursivamente, o calculo da matriz inversa é ainda mais custoso.

Em Duda *et al.* (2001) é apresentada a formulação de um discriminante para classificação de dados considerando que tenham uma distribuição normal. Esse discriminante, um para cada classe, tem a forma

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mathbf{c}_i)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{c}_i) + \ln P(classe_i), \quad i = 1, \ldots, C \tag{4.2}$$

e considera-se, neste caso, uma matriz de covariância $\mathbf{\Sigma}$ arbitrária, mas idêntica para todas as classes. Nessa situação, todas as amostras tendem a formar grupos hiper elipsoidais de mesmo tamanho e forma, centrados em $\mathbf{c}_i$ (DUDA *et al.*, 2001). O termo $\ln P(classe_i)$,em que $P(classe_i)$ é a probabilidade a priori dos dados, se considerado igual para todo $i = 1, \ldots, C$, pode ser ignorado na formulação restando exatamente a distância de Mahalanobis ponderada por um termo constante, $-1/2$. A classificação do dado $\mathbf{x}$ é feita definindo-se a qual centro $\mathbf{c}$ esse dado está mais próximo. Em eNNA, a matriz $\mathbf{\Sigma}$ é diferente para cada grupo formado e é composta apenas de elementos na sua diagonal principal, conforme discutido mais adiante.

Para que a estrutura do classificador eNNA seja evoluída, dados são atribuídos aos grupos já existentes ou formam novos grupos. A utilização de grupos hiper elipsoidais é considerada, porém com definição de uma borda. Essa borda é essencial na etapa de evolução da estrutura. Desta forma, se considerarmos a equação de um elipsoide em sua forma canônica (reduzida), com eixos x_1, x_2 e x_3 e centro em c_1, c_2 e c_3 temos que (STEINBRUCH; WINTERLE, 1987):

$$\frac{(x_1 - c_1)^2}{r_1^2} + \frac{(x_2 - c_2)^2}{r_2^2} + \frac{(x_3 - c_3)^2}{r_3^2} = 1 \tag{4.3}$$

onde r_1, r_2 e r_3 são os parâmetros que definem a dispersão da elipse ao longo dos eixos x_1, x_2 e x_3 respectivamente. A borda do elipsoide é definida para todo dado $\mathbf{x}$ tal que a equação (4.3) seja igual a 1. A Figura 13 ilustra a elipse no plano (Figura 13a) e no espaço 3-dimensional (Figura 13b).

A expressão (4.3) pode ser reescrita como:

$$(\mathbf{x} - \mathbf{c})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{c}) = 1 \tag{4.4}$$

em que,

$$\mathbf{\Sigma} = \begin{bmatrix} r_1^2 & 0 & 0 \\ 0 & r_2^2 & 0 \\ 0 & 0 & r_3^2 \end{bmatrix}. \tag{4.5}$$

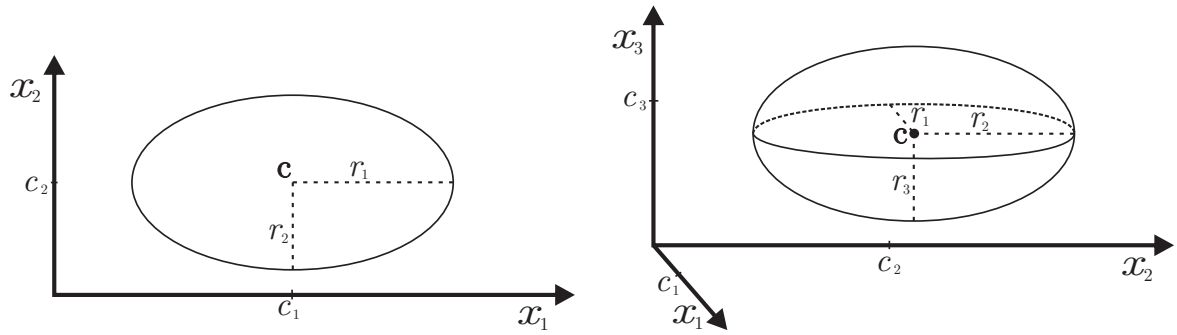


Figura 13 – Ilustração de uma elipse no plano e no espaço 3-dimensional

As expressões (4.1) e (4.4) possuem formas similares. Com essa formulação é possível verificar que, considerando elipsoides com eixos paralelos aos eixos x_1, x_2,..., x_d, $\mathbf{\Sigma}$ é diagonal e a sua inversa contém o inverso das componentes da diagonal principal de $\mathbf{\Sigma}$, o que reduz o custo computacional do cálculo da inversa de $\mathbf{\Sigma}$.

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} \frac{1}{r_1^2} & 0 & \dots & 0 \\ 0 & \frac{1}{r_2^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{r_d^2} \end{bmatrix} \tag{4.6}$$

Em modelos evolutivos baseados em agrupamentos, a decisão se um dado pertence a um grupo depende de um limiar escolhido pelo usuário. Utilizar a equação de um elipsoide é útil para determinar se um dado pertence ao grupo correspondente, pois, calculando-se

$$\Upsilon = (\mathbf{x} - \mathbf{c})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{c}) - 1 \tag{4.7}$$

se $\Upsilon < 0$ o dado pertence ao grupo, se $\Upsilon = 0$ está na borda do grupo e se $\Upsilon > 0$ ele não pertence ao grupo. Portanto, a decisão quanto a pertinência de um dado a um grupo independe da escolha de um limiar arbitrário, mas depende da dispersão.

O primeiro dado é o centro do primeiro grupo e o desvio padrão é zero. Contudo é preciso que a dispersão do grupo seja escolhida para estabelecer a região de influência inicial. Lughofer (2011a) sugere um valor para definir a região de influência considerando a dimensão dos dados, conforme (2.4). Este é o valor inicial para a dispersão do grupo, denotado por $\mathbf{r}$:

$$\mathbf{r} = [r_1, \dots, r_d]^T \quad r_j = fac\frac{\sqrt{d}}{\sqrt{2}}, \quad j = 1, \dots, d \tag{4.8}$$

onde r_j é a dispersão da j-ésima coordenada e fac é uma constante podendo assumir valores entre 0 e 1. A equação (4.8) define inicialmente uma hiperesfera, um caso especial

de uma hiper elipsoide quando todas as dispersões são iguais. A dispersão de um grupo é atualizada sempre que um novo dado é atribuído a este grupo fazendo uma combinação convexa entre o valor da dispersão no passo anterior e o valor atual do desvio padrão $\boldsymbol{\sigma}_k = [\sigma_{1k}, \ldots, \sigma_{dk}]$ com o novo dado incluído, isto é:

$$\mathbf{r}_k = \rho \mathbf{r}_{k-1} + (1 - \rho)\boldsymbol{\sigma}_k \tag{4.9}$$

onde $\rho \in [0, 1]$ (ANGELOV; ZHOU, 2006). A Figura 14 ilustra o efeito da atualização da dispersão de um grupo. O valor ρ determina o quão rápido a dispersão inicial converge para o desvio padrão do grupo (ANGELOV; ZHOU, 2008).

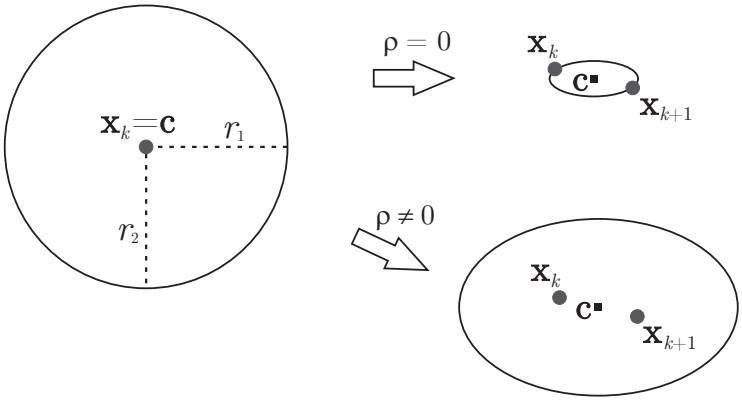


Figura 14 – Efeito de ρ na atualização da dispersão de um grupo

Conforme mostra a Figura 14 quando $\rho = 0$ a dispersão é o desvio padrão e a redução do valor atual da dispersão pode ser significativa. Quando $\rho \neq 0$ a dispersão é a combinação convexa entre a dispersão e o desvio padrão.

A regra para criação de novos grupos durante o processo de evolução da estrutura do modelo assume grupos elípticos, isto é, descritos por:

$$\Upsilon^i = (\mathbf{x} - \mathbf{c}^i)^T (\alpha \boldsymbol{\Sigma}^i)^{-1} (\mathbf{x} - \mathbf{c}^i) - 1, \qquad (\alpha \boldsymbol{\Sigma}^i)^{-1} = \begin{bmatrix} \frac{1}{\alpha(r_1^i)^2} & 0 & \ldots & 0 \\ 0 & \frac{1}{\alpha(r_j^i)^2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\alpha(r_d^i)^2} \end{bmatrix} \tag{4.10}$$

onde i refere-se ao i-ésimo grupo e α é uma constante positiva que define a proporção da dispersão considerada em (4.10). Tendo que a dispersão do grupo i é dada pela combinação convexa entre $\mathbf{r}_{k-1}$ e o desvio padrão $\boldsymbol{\sigma}_k$ e considerando dados com uma distribuição normal, assume-se que aproximadamente 95,45% dos dados estão inseridos em $2\boldsymbol{\sigma}_k$. Esse conceito estatístico suporta a escolha do valor de α que será utilizado no eNNA.

Se L_k é o número de grupos existentes no passo k, então a condição:

$$\Upsilon^i > 0, \quad \forall i \in [1, L_k], \tag{4.11}$$

sugere quando um novo grupo deve ser adicionado na estrutura corrente, conforme ilustra a Figura 15. O efeito da proporção da dispersão α é ilustrado pela linha tracejada na Figura 15. Os parâmetros fac e ρ influenciam o número de grupos que formam a estrutura. Esta influência será analisada no Capítulo 5.

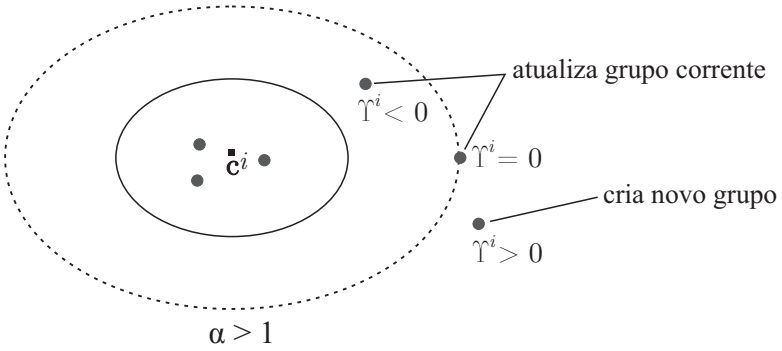


Figura 15 – Criação de novo grupo.

Quando um dado provoca a adição de um novo grupo, o valor para a dispersão inicial atribuído pode ser tal que este grupo sobreponha parcialmente ou completamente a um ou mais grupos existentes. Sobreposição de grupos não é uma condição proibitiva, mas casos extremos de superposição devem ser evitados. Uma forma é calcular a distância entre o centro do novo grupo $\mathbf{c}^{i+1}$ e todos os centros restantes e verificar se:

$$\exists l \neq i+1, l \in [1, L_k] \,|\, \min_l dist^l(\mathbf{c}^{i+1}, \mathbf{c}^l) < fac\frac{\sqrt{d}}{\sqrt{2}} \tag{4.12}$$

Se (4.12) é satisfeita então a dispersão inicial do novo grupo $i+1$ é definida como (2/3) da mínima distância para qual a condição foi satisfeita, reduzindo ou eliminando a superposição. A dispersão inicial do novo grupo será então:

$$\mathbf{r}^{i+1} = [r_1^{i+1}, \ldots, r_d^{i+1}]^T \quad r_j^{i+1} = \frac{2}{3}\min_l dist^l(\mathbf{c}^{i+1}, \mathbf{c}^l),\ j = 1, \ldots, d \quad \text{e} \quad l = 1, \ldots, L_k \tag{4.13}$$

A condição imposta em (4.12) é ilustrada na Figura 16 onde a distância 1 ($dist^1$) é menor que o valor inicial da dispersão para um novo grupo. Assim, o novo grupo criado irá sobrepor boa parte do grupo 1, incluindo seu centro. Portanto, a dispersão inicial do novo grupo, $\mathbf{r}^{i+1}$ (grupo 3), é definido como (2/3) da distância 1, conforme (4.13).

Se um novo grupo não é criado, ou seja, se a condição (4.11) não for satisfeita, então o dado de entrada está próximo a algum grupo já existente. Verifica-se a qual grupo o dado pertence usando

$$i^* = \arg\min_i(\Upsilon^i) \tag{4.14}$$

Contudo, na maioria dos casos reais, como já comentado em momentos anteriores, as fronteiras de decisão entre as classes não são lineares e as classes podem se sobrepor com

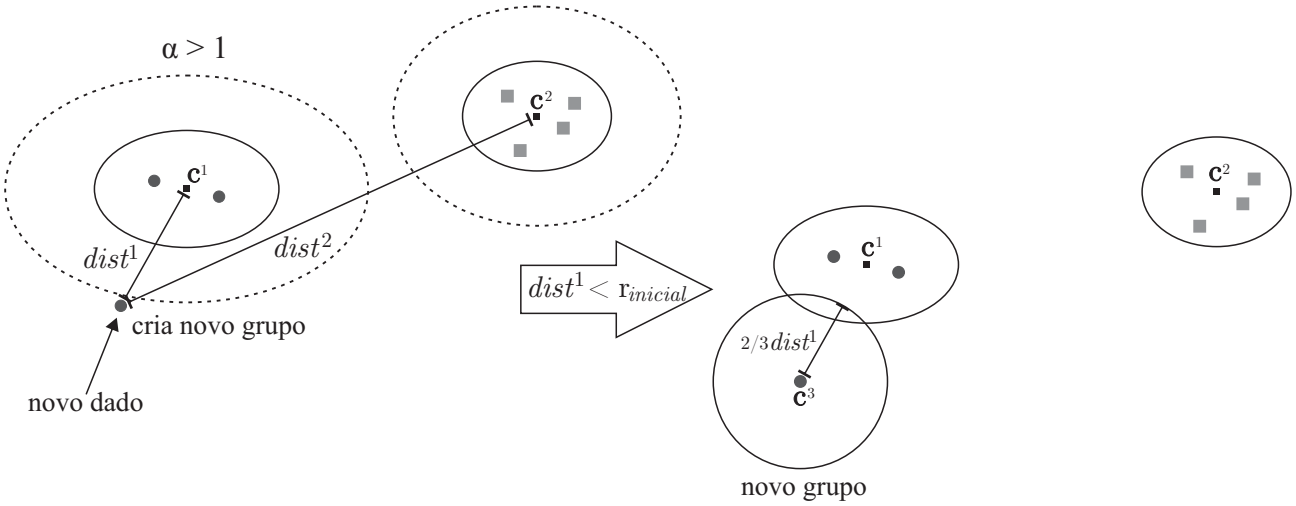


Figura 16 – Condição de sobreposição na criação de um novo grupo.

intensidades diferentes. Um mecanismo é necessário para verificar a compatibilidade entre as classes e o grupo atribuído ao dado. Se o dado pertence a uma classe diferente àquela correspondente ao grupo do centro $\mathbf{c}^{i^*}$, então o novo grupo representa a outra classe. Tal condição cria sobreposição entre estes grupos de classes diferentes, o que também é indesejado. Atualiza-se a dispersão do grupo existente e define-se a dispersão inicial do novo grupo para minimizar a sobreposição entre eles, usando

$$\mathbf{r}^{i^*} \leftarrow 0.5\mathbf{r}^{i^*} \tag{4.15}$$

$$\mathbf{r}^{i+1} = \mathbf{r}^{i^*} \tag{4.16}$$

A intenção é buscar por uma estrutura de grupo que melhor represente as classes. A dispersão do novo grupo (4.16) evita uma possível nova sobreposição se feita segundo (4.8). A Figura 17 ilustra esta situação.

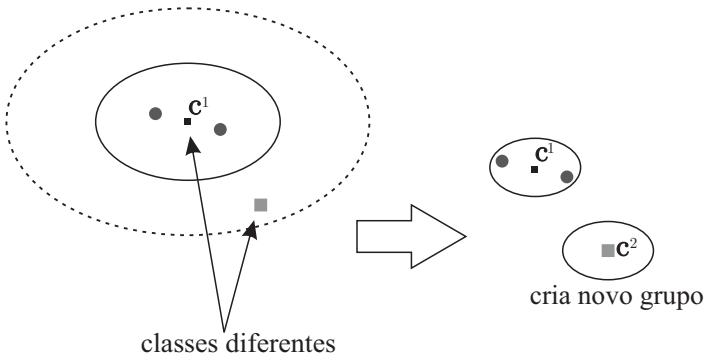


Figura 17 – Criação de novo grupo sob condição de classes distintas.

Quando o grupo i^* é atribuído ao dado de entrada correspondente à classe correta, a correspondente dispersão e o centro do grupo são atualizados recursivamente de acordo

com a média e o desvio padrão, isto é, calcula-se (ANGELOV *et al.*, 2010)

$$\mathbf{c}_k^{i^*} = \frac{M^{i^*}-1}{M^{i^*}}\mathbf{c}_{k-1}^{i^*} + \frac{1}{M^{i^*}}\mathbf{x}_k \tag{4.17}$$

$$(\boldsymbol{\sigma}_k^{i^*})^2 = \frac{M^{i^*}-1}{M^{i^*}}(\boldsymbol{\sigma}_{k-1}^{i^*})^2 + \frac{1}{M^{i^*}-1}(\mathbf{c}_k^{i^*} - \mathbf{c}_{k-1}^{i^*})^2 \tag{4.18}$$

onde M^i é o número de dados atribuídos ao grupo i. Quando um novo grupo é criado $M^{i+1} = 1$. O valor $(\boldsymbol{\sigma}_k^{i^*})^2$ é a variância associada ao grupo i^*.

A etapa de evolução da estrutura apenas mantém ou adiciona novos grupos. Entretanto, é comum que a estrutura do grupo possa, com o tempo, não ser mais representativa. Além disso, pontos fora da curva (*outliers*) podem levar à criação de grupos que não serão atualizados por nenhum dado posterior. Dessa forma, adota-se uma regra para monitorar a estrutura de grupo criada para o classificador.

O Capítulo 2 sugere três formas de monitorar a qualidade da estrutura de um modelo evolutivo. O eNNA utiliza o critério $C3$ descrito em (2.7), com alteração do tempo de monitoramento, isto é:

$$C3 : SE(M_k^i < 3)E(k \geq I^i + 20)ENTÃO(L \leftarrow L - 1) \tag{4.19}$$

em que I^i é o instante em que o i-ésimo grupo foi criado. O monitoramento da estrutura de grupo completa a etapa de evolução da estrutura do classificador. Os valores 3 amostras e 20 iterações utilizados em (4.19) foram definidos em testes preliminares e mantidos fixos. Em outros dados e ambientes com dimensões distintas, pode-se considerar novos testes preliminares. Todos os resultados encontrados e apresentados nesta dissertação utilizam exatamente a equação (4.19), não sendo necessário novos ajustes. A etapa seguinte trata da adaptação dos parâmetros.

4.3 Estimação de Parâmetros

A seção 4.2 mostrou como a estrutura do classificador neural evolutivo é determinada. Uma vez definida, a estrutura se mantém fixa, mas seus parâmetros são atualizados sempre que processa um dado de entrada.

O eNNA é uma rede neural composta por camada de entrada, com cada neurônio associado a um único atributo do dado , uma camada oculta e uma camada de saída com tantos neurônios quanto o número de classes. A figura 18 apresenta a arquitetura do classificador eNNA.

A camada oculta da rede tem papel fundamental no classificador. É responsável pela não linearidade do mesmo. O número de neurônios da camada oculta é definido pelo número de grupos formados na etapa de evolução da estrutura do modelo. Assim como em Rosa *et al.* (2014), a eNNA associa cada neurônio da camada oculta a um grupo.

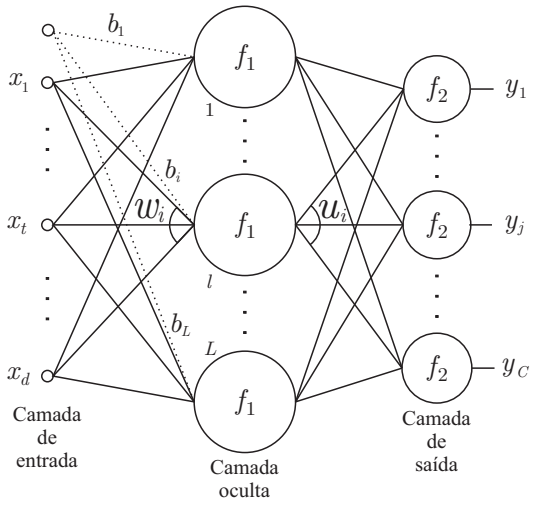


Figura 18 – Arquitetura do classificador eNNA

Da expressão (3.11), a saída do eNNA, similarmente a uma ELM, é

$$\mathbf{y}_k = \sum_{i=1}^{L} \mathbf{u}_i f(\mathbf{w}_i^T \mathbf{x}_k + b_i), \quad k = W_{init} + 1, \ldots, N \tag{4.20}$$

onde $\mathbf{w}_i$ é o vetor de pesos correspondente ao $i-$ésimo neurônio da camada oculta e b_i é o limiar correspondente. O peso da camada de saída é $\mathbf{u}$ e a saída em k, $\mathbf{y}_k$. Os neurônios em conjunto representam as regiões do espaço de dados associados às classes. O eNNA utiliza a função de ativação tangente hiperbólica, $\tanh(\mathbf{x})$.

A expressão (4.20) define a saída $\mathbf{y}_k = [y_k^1, \ldots, y_k^j, \ldots, y_k^C]$, onde C é o número de classes. A classificação é feita decidindo pela classe correspondente à saída mais ativa, isto é

$$classe_k = \arg\max_j (y_k^j),\ j = 1, \ldots, C \tag{4.21}$$

Na etapa de estimação dos parâmetros, os valores $\mathbf{w}$ e $\mathbf{u}$ são, após inicialização, estimados a cada novo dado de entrada. Entretanto, ao contrário da ELM onde os pesos da camada de entrada são escolhidos aleatoriamente e mantidos fixos, no eNNA os pesos $\mathbf{w}$ são vetores centros de grupos definidos na etapa de evolução da estrutura, $\mathbf{w}_i = \mathbf{c}_i$, $i = 1, \ldots, L$. O valor do limiar b_i é definido baseado em discriminantes lineares onde, implicitamente, calcula-se a norma euclidiana entre o dado de entrada $\mathbf{x}_k$ e o centro do grupo, definido agora como $\mathbf{w}_i$. Assim,

$$g(\mathbf{x}_k) = \| \mathbf{x}_k - \mathbf{w}_i \|^2 = (\mathbf{x}_k - \mathbf{w}_i)^T (\mathbf{x}_k - \mathbf{w}_i) \tag{4.22}$$

$$= \mathbf{x}_k^T \mathbf{x}_k - 2\mathbf{w}_i^T \mathbf{x}_k + \mathbf{w}_i^T \mathbf{w}_i \tag{4.23}$$

Como $\mathbf{x}_k^T \mathbf{x}_k$ é constante para cada $\mathbf{x}_k$, esse termo pode ser ignorado e reorganizando a equação (4.23), tem-se

$$g(\mathbf{x}_k) = \mathbf{w}_i^T \mathbf{x}_k - \frac{1}{2} \mathbf{w}_i^T \mathbf{w}_i = \mathbf{w}_i^T \mathbf{x}_k + b_i \tag{4.24}$$

considerando assim $b_i = -0.5\mathbf{w}_i^T\mathbf{w}_i$. A expressão (4.24) é exatamente a mesma expressão utilizada na função $f(\cdot)$, na equação (4.20).

Uma vez definidos os pesos da camada de entrada, resta estimar os valores de $\mathbf{U}$. Este trabalho utiliza os quadrados mínimos recursivo. A cada novo dado, estima-se os valores de $\mathbf{U}$ para continuamente minimizar o erro entre o valor estimado e o esperado. A inicialização é feita com $\mathbf{u}_i = 0$ para não favorecer nenhum neurônio da camada oculta. A atualização se dá segundo as expressões (YOUNG, 1984):

$$\mathbf{g}_k = \frac{1}{\mathbf{h}_k^T\mathbf{P}_{k-1}\mathbf{h}_k + 1}\mathbf{P}_{k-1}\mathbf{h}_k \tag{4.25}$$

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{g}_k\mathbf{h}_k^T\mathbf{P}_{k-1} \tag{4.26}$$

$$\mathbf{U}_k = \mathbf{U}_{k-1} + \mathbf{g}_k[\mathbf{y}_k - \mathbf{h}_k^T\mathbf{U}_{k-1}] \tag{4.27}$$

onde $\mathbf{P}_k \in \Re^{L\times L}$ é inicializada como uma matriz diagonal com valores elevados. Comumente, utiliza-se $\mathbf{P}_k = \delta\mathbf{I}$, $\delta = 1000$. O termo $\mathbf{h}$ é a saída dos neurônios da camada oculta de rede, $\mathbf{h} = \tanh(\mathbf{w}^T\mathbf{x}_k + b_i)$, onde $\mathbf{h}_k \in \Re^L$, $\mathbf{g}_k \in \Re^L$, $\mathbf{U}_k \in \Re^{L\times C}$ composto pelos pesos $\mathbf{u}_i \in \Re^C$ da camada de saída, na forma.

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_i^T \\ \vdots \\ \mathbf{u}_L^T \end{bmatrix} \tag{4.28}$$

O classificador eNNA está resumido pelo pseudocódigo da Figura 21 e a Figura 20 mostra um exemplo de classificação utilizando o eNNA em um dado com dimensão 2, para visualização. A Figura 19 mostra os dados dispostos no plano, com duas classes distintas e com uma fronteira de decisão não linear entre as classes.

A Figura 20a apresenta a estrutura evoluída pelo classificador na primeira etapa de evolução da estrutura, para apenas uma quantidade inicial de dados. É possível observar que os grupos criados são definidos por elipses com eixos paralelos aos eixos x_1 e x_2, conforme proposto. Alguns dados podem não estar associados à algum grupo, pois, na etapa de poda o grupo a ele associado foi removido da estrutura. A Figura 20b mostra os dados classificados após a apresentação de todo o conjunto em fluxo, e mostra os dados não classificados, representando os dados processados na primeira etapa de evolução da estrutura. Neste exemplo, a acurácia de classificação foi de $80,52\%$ e uma estrutura evoluída com 23 grupos.

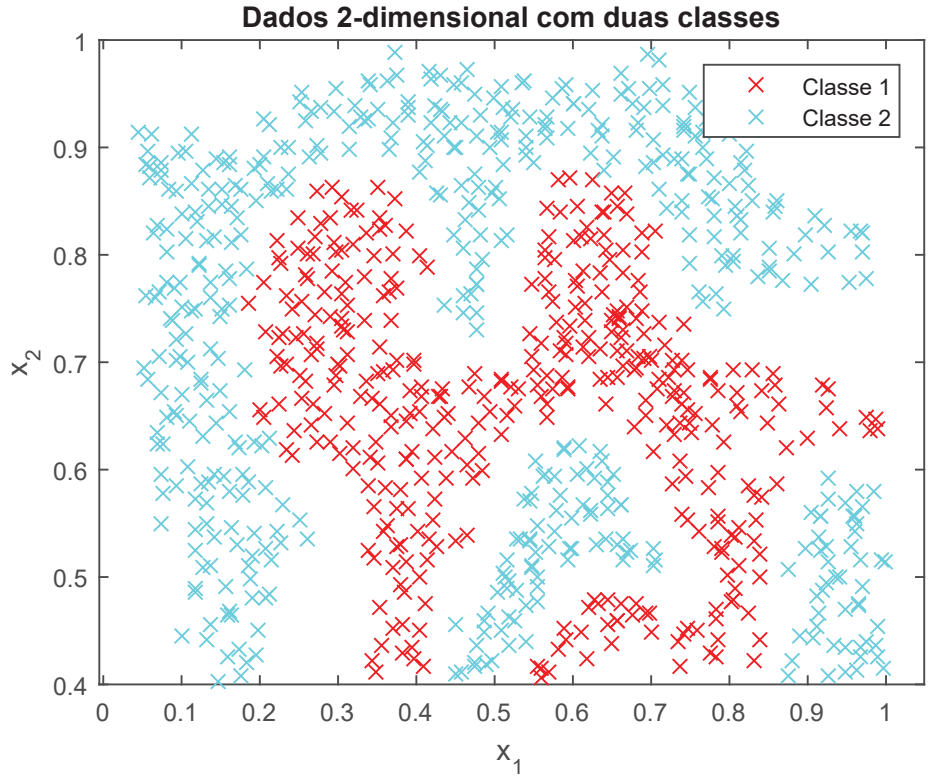
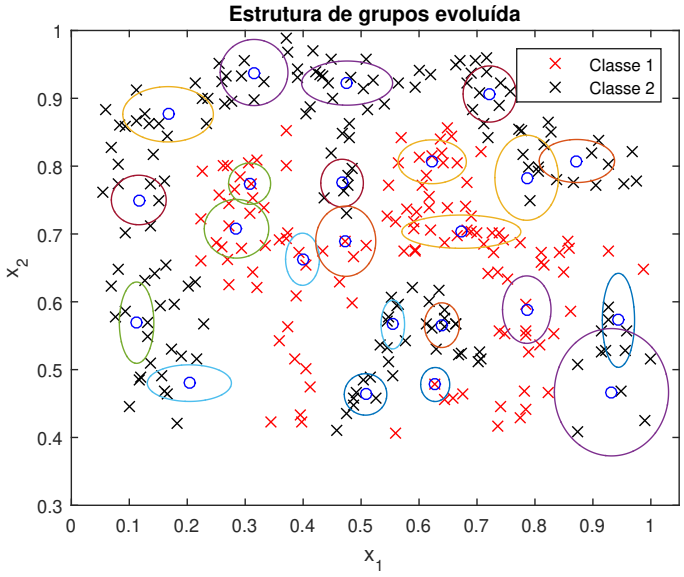
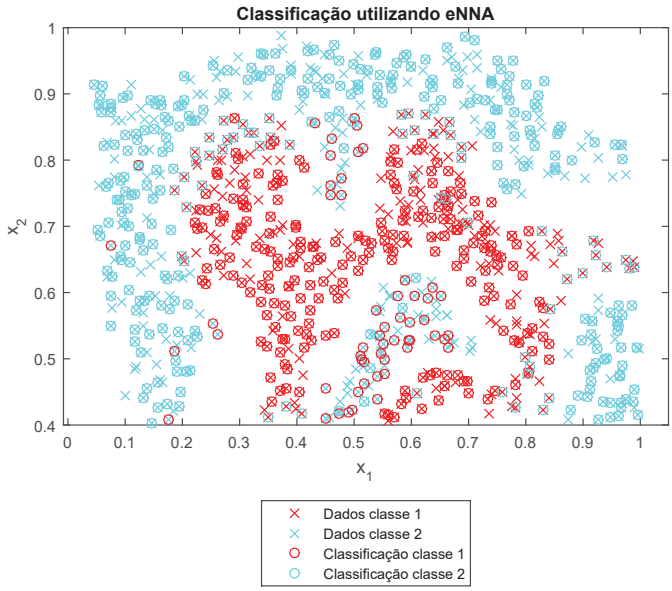


Figura 19 – Dados 2-dimensional com duas classes. Disponível em: <http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html>



(a)



(b)

Figura 20 – Exemplo de classificação utilizando o classificador evolutivo eNNA

Algoritmo eNNA

Inicializa
$k = 1$; $i = 1$; $\mathbf{c}^i \leftarrow \mathbf{x}_k$; $\mathbf{r}^i$ usando (4.8); ρ; W_{init}; $M_k^i = 1$;
para $k = 2$ **até** W_{init} **faça**
  computar Υ^i usando (4.10)
  ler classe C_k correspondente a $\mathbf{x}_k$
  decidir por criar um novo grupo usando (4.11)
  se criar novo grupo **então**
    $i = i + 1$; $\mathbf{c}^{i+1} \leftarrow \mathbf{x}_k$; $M_k^i = 1$; $Cl^{i+1} = C_k$
    checar $dist^i$ usando (4.12)
    se existe algum i tal que (4.12) seja satisfeita **então**
      $\mathbf{r}_k^i = (2/3) * \min dist$ para qual i satisfez;
    senão
      $\mathbf{r}_k^i$ usando (4.8);
    fim se
  senão
    checar a qual grupo $\mathbf{x}_k$ pertence utilizando (4.14)
    se $C_k \neq Cl^i$ **então**
      criar novo grupo
      $i = i + 1$; $\mathbf{c}^{i+1} \leftarrow \mathbf{x}_k$; $M_k^i = 1$; $Cl^{i+1} = C_k$
      atualizar a dispersão do grupo usando (4.15)(4.16)
    senão
      atualizar grupo $i^* \mid M_k^i = M_k^i + 1$;
      atualizar centro $\mathbf{c}^{i^*}$ usando (4.17)
      atualizar dispersão $\mathbf{r}^{i^*}$ usando (4.9)
      atualizar desvio padrão $\boldsymbol{\sigma}^{i^*}$ usando (4.18)
    fim se
  fim se
  verificar se algum grupo pode ser removido usando (4.19)
fim para
$\mathbf{P}_0 = 1000\mathbf{I}$; $\mathbf{u}^i = 0, \quad i = [1, L_k]$
enquanto existir dado disponível **faça**
  computar saída $\mathbf{y}_k$ usando (4.20)
  classificar a amostra $\mathbf{x}_k$ usando (4.21)
  ler classe C_k correspondente a $\mathbf{x}_k$
  atualizar $\mathbf{U}$ usando (4.27)
  computar taxa de classificação (%)
fim enquanto

Figura 21 – Pseudocódigo do classificador eNNA.

4.4 Resumo

Este Capítulo apresentou o método Neuro Classificador Evolutivo para Espaço de Alta Dimensão (eNNA). O eNNA possui duas etapas, a primeira particiona o espaço de dados de acordo com as classes. A segunda estima parâmetros do classificador evoluído. O eNNA possui uma estrutura de rede neural com três camadas, no qual cada neurônio da camada oculta é associado a um grupo evoluído na primeira etapa. Assim, os pesos das conexões sinápticas da rede são os centros dos grupos para camada de entrada e os pesos da camada de saída são estimados via quadrados mínimos recursivos sempre que um novo dado é apresentado à rede.

5 Resultados Computacionais

Este capítulo apresenta os resultados computacionais obtidos pelo classificador eNNA quando aplicado a conjuntos de dados de alta dimensão. Os conjuntos de dados são apresentados, enfatizando as características importantes de cada um deles. Também discute-se os métodos utilizados para avaliar e comparar o eNNA com classificadores representativos do estado da arte.

5.1 Descrição dos Conjuntos de Dados

Alta dimensionalidade é comumente referenciada à imagens, vídeos e textos de uma forma geral. Documentos de textos, especificamente, têm a característica adicional de serem esparsos. Um documento pode ser representado por um vetor de dimensão igual a quantidade total de palavras no vocabulário utilizado e apenas poucas palavras são utilizadas no texto entre todas as possíveis.

O conjunto *20 Newsgroup* (MITCHELL, 1997) consiste em uma coleção de documentos de textos com 20 diferentes sub assuntos, categorizados em 7 assuntos com características comuns. São eles: *alt* com documentos relativos ao ateísmo, *comp* com textos relacionados à computação, *misc* com temas diversos, *rec* referente à recreação e entretenimento, *sci* com textos relacionados à ciência, *soc* com discussões sociais e *talk* com documentos relacionados a temas controversos. O conjunto bruto é composto por 20000 documentos e um pré processamento é feito removendo palavras derivadas de um radical comum (*stemming*) e palavras vazias (*stop words*), que são pouco informativas em um contexto geral. Adicionalmente, palavras que apareçam no documentos com frequência menor que 2 são removidas (ZHAO; MAO, 2015). Ao final, um vocabulário de 12000 palavras é selecionado, considerando as palavras mais frequentes. Do total, aproximadamente 18000 documentos são selecionados contendo apenas os assuntos COMP, REC, SCI e TALK. Os quatro assuntos separados são agrupados dois a dois, formando seis conjuntos finais com dois assuntos distintos em cada. O problema aqui é classificar documentos corretamente entre dois assuntos rotulados como classe positiva (valor +1) e classe negativa (valor -1). Os assuntos correspondentes às respectivas classes são apresentados na Tabela 1.

O segundo conjunto *Farm Ads* (LICHMAN, 2013) reúne textos extraídos de sites com tópicos relacionados a animais que vivem em fazendas. De um total de doze sites, o conjunto de dados é formado por textos extraídos de anúncios existentes nas páginas e pelas informações contidas na página onde o anúncio foi publicado. Um pré processamento também foi feito neste caso, removendo *stemming* e *stop words* (MESTERHARM;

PAZZANI, 2011). Os proprietários do conteúdo avaliaram o anúncio segundo critérios e estes foram rotulados em anúncio apropriado (valor +1) e anúncio inapropriado (valor -1). Ao final, o conjunto é composto por aproximadamente 4000 amostras de textos com dimensão 54887 (vocabulário).

A Tabela 1 mostra o número de dados em cada classe considerada, para cada um dos conjuntos de dados descritos.

Tabela 1 – Divisão dos conjuntos de dados - *20 Newsgroup* e *Farm Ads*

Conjunto	# Classe (1)	# Classe (−1)
COMPxREC	3979	3903
COMPxSCI	3952	3903
COMPxTALK	3253	3903
RECxSCI	3952	3979
RECxTALK	3253	3979
SCIxTALK	3253	3952
Farm Ads	2072	2071

Para análise do desempenho do algoritmo com dados não estacionários, utiliza-se o conjunto de dados *Hyperplane*, gerado por um analisador de dados massivos (MOA, *Massive Online Analysis*) (BIFET *et al.*, 2010). O conjunto consiste em 120 mil amostras de dimensão 4. Como característica desses dados, tem-se que as duas classes presentes podem ser separadas por um hiper plano. Ao longo do tempo, a distribuição das duas classes no espaço de dados é alterada gradativamente, mas mantendo-se linearmente separáveis. A mudança de distribuição ocorre após a entrada de 40 mil amostras.

5.2 Método de Avaliação

Para avaliar o desempenho dos classificadores usados na comparação, utiliza-se a porcentagem de acertos, o tempo de processamento por amostra e a complexidade do modelo.

Define-se acurácia como:

$$acurácia = \frac{\text{número de amostras corretamente classificadas}}{\text{número total de amostras classificadas}} \tag{5.1}$$

A acurácia pode fornecer informações distorcidas a respeito do desempenho de um classificador. Por exemplo, Metz (1978) discute o caso em que o conjunto de dados é desbalanceado, isto é, o número de amostras em uma classe é muito maior que da outra. Nesses casos, medidas separadas para cada classe devem ser feitas para analisar o desempenho igualmente. Os conjuntos de dados considerados nesta dissertação possuem quantidades semelhantes de amostras em ambas as classes conforme mostra a Tabela 1. Dessa forma, a medida de acurácia simples (5.1) pode ser usada.

O tempo de processamento dos algoritmos considerados também é computado. O tempo total de execução é comumente utilizado na literatura e seus resultados comparados. Entretanto, para algoritmos evolutivos, o tempo de processamento por amostra é mais representativo, pois essas são executados enquanto houver um fluxo de dados. Assim, nesta dissertação, considera-se o tempo de execução por amostra em milissegundos (ms), computados ao longo de todo o conjunto, incluindo etapas de evolução / treinamento e etapas de adaptação / teste. Para os classificadores não evolutivos, também computa-se o tempo de processamento por amostra para comparação, sendo calculado como tempo total de execução dividido pelo número de amostras.

A complexidade dos modelos evolutivos é avaliada utilizando o número de regras nebulosas evoluídas nos modelos que consideram sistemas nebulosos e o número de neurônios na camada intermediária evoluídos ou pré definidos. Para os métodos alternativos, serão analisados os números de nós (CART) e número de vetores suporte (SVM).

Também serão feitas variações em alguns parâmetros característicos nos algoritmos considerados. Esta variação visa verificar a influência de cada parâmetro na evolução de um modelo e, consequentemente, no desempenho do mesmo. Em eNNA será variado o valor de $fac = [0, 1]$, $\rho = [0, 1]$ e $W_{init} = \{150, 500, 1000\}$. Para eClass0, os parâmetros variados serão $\rho = [0, 1]$ e $r_0 = [0, 1]$. Em FLEXFIS-Class SM o ganho inicial do algoritmo eVQ $init\_gain = [0, 1]$ e $fac = [0, 1]$ também serão alterados. Para os algoritmos não evolutivos, em SVM são verificadas quais funções kernel conduzem a uma melhor aproximação, bem como o método de busca do hiper plano separador. Em kNN é variado o valor de $k = [1, 20]$. Para PC-ELM é verificado o desempenho em função da variação do coeficiente de regularização na faixa $\{2^{-24}, 2^{+25}\}$ (HUANG *et al.*, 2012), considerando uma faixa de valores suficientemente ampla para verificação em aplicações práticas (KULAIF, 2014).

Os classificadores evolutivos utilizam todo o conjunto de dados como um fluxo contínuo, enquanto que, para os classificadores não evolutivos, os dados foram separados em conjunto de treinamento e conjunto de teste na proporção 50/50%.

Todas as simulações foram feitas utilizando MATLAB®2014b, em um desktop equipado com processador Intel®Core i7-2600 2.4GHz com 8Gb de memória RAM.

5.3 Resultados e Análise

O neuro classificador evolutivo proposto nesta dissertação é influenciado por alguns parâmetros e o desempenho depende da escolha desses. O valor dos parâmetro ρ em (4.9), fac em (4.8) e a quantidade de dados iniciais W_{init} são analisados na sequência, comparando com os resultados encontrados para os métodos evolutivos como eClass0, AnYa-Class e FLEXFIS-Class SM; e métodos não evolutivos como SVM, CART, kNN e

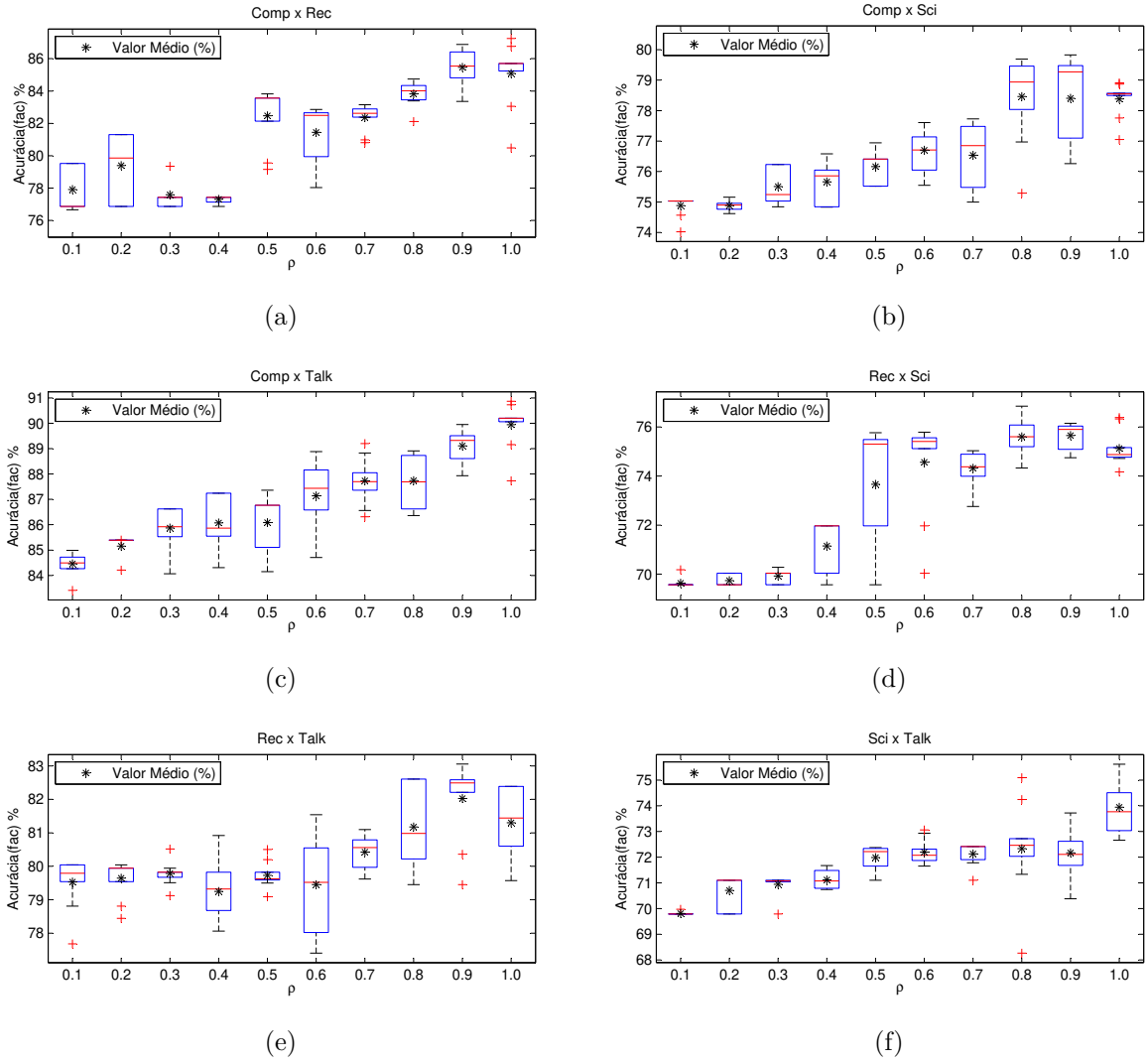


Figura 22 – Sensibilidade do eNNA quanto à variação de ρ: *20 Newsgroup*

PC-ELM.

A primeira análise consiste na verificação da influência dos parâmetros ρ e fac no desempenho do algoritmo eNNA. A Figura 22 mostra o desempenho para os conjuntos *20 Newsgroup* utilizando *boxplots*. Os diagramas *Boxplots* mostram os valores máximo e mínimo de um conjunto de acurácias, a mediana e define uma faixa para indicar a distribuição pertencentes ao segundo e terceiro quartil, ou seja, a faixa onde estão localizados 50% das acurácias consideradas. A mediana mostra quão simétrica é a distribuição dos dados. Valores potencialmente considerados como *outliers* são representados pelo sinal $(+)$. A média dos valores de acurácia é representada pelo símbolo $(*)$.

A Figura 22 mostra como a acurácia é afetada pela escolha de ρ. Conforme (4.9), o valor de ρ influencia diretamente na atualização da dispersão do grupo. Quanto menor o seu valor, maior a velocidade que a dispersão tende ao valor do desvio padrão do grupo. Conforme mostra a Figura 22, tal comportamento foi aferido, no sentido de que quanto

menor o valor de ρ, menor o desempenho médio do classificador eNNA. Pequenos valores fazem com que, praticamente com os primeiros dados, a dispersão inicial do grupo definida em (4.8) tenda rapidamente ao valor do desvio padrão. Dessa forma, um grupo com uma região de abrangência inicial ampla, pode perder sua capacidade de generalização simplesmente por ser atualizado por um dado muito compatível com o grupo, ou seja, muito próximo no espaço de dados. Esse efeito é verificado uma vez que a média das acurácias apresentadas tem seu valor sendo incrementado conforme o valor de ρ aumenta. Apesar de a diferença entre as acurácias mínimas e máximas não ser tão grande, variando em aproximadamente 2%, conforme na Figura 22e e aproximadamente 9%, conforme na Figura 22a, fica claro que a média de desempenho em todos os conjuntos é aumentada proporcionalmente com o aumento do parâmetro ρ.

A segunda análise observa a variação do parâmetro fac entre valores 0 e 1, em cada uma das 10 execuções feitas para o parâmetro ρ. A dispersão inicial de um grupo é definido em função da dimensão do dado e o fator fac fornece uma espécie de ajuste fino deste valor. A interferência deste parâmetro é representada pelos diagramas de caixa (*boxplots*) e, inicialmente, são selecionados apenas os digramas referentes aos parâmetros que geram maiores valores de acurácia média. Para o conjunto Comp x Rec, valores de $\rho = 0.9$ e $\rho = 1$ produziram uma acurácia média semelhante. O diagrama de caixa permite dizer que para $\rho = 0.9$ aproximadamente 50% dos resultados possuem valores maiores que a média, justificando a escolha, mesmo apesar de alguns "outliers" melhores para $\rho = 1$. O conjunto Comp x Sci possui uma avaliação semelhante, visto que três valores produziram acurácias médias próximas, $\rho = 0.8$, $\rho = 0.9$ e $\rho = 1$. A mesma justificativa é válida neste caso e para $\rho = 0.9$, em que mais de 50% dos valores são maiores que a média com 50% destes compreendidos em uma pequena faixa, no topo do diagrama. Os conjuntos Comp x Talk, Rec x Talk e Sci x Talk obtiveram valores bem definidos para acurácia média, utilizando respectivamente $\rho = 1$, $\rho = 0.9$ e $\rho = 1$. Em Rec x Talk entretanto, apesar dos dois valores considerados "outliers", 80% dos resultados estão acima da média encontrada, justificando a escolha. Por último, no conjunto Rec x Sci dois valores produziram acurácias médias bastante próximas, $\rho = 0.8$ e $\rho = 0.9$. Assim, o valor 0.9 é considerado na análise, levando em consideração também que mais de 50% dos resultados estão acima da média encontrada.

Na Figura 23 são apresentadas as curvas de acurácia em função da variação do parâmetro fac, para os valores de ρ definidos anteriormente, um para cada conjunto de dados.

É possível observar que, em geral, a acurácia do classificador eNNA aumenta conforme o valor do parâmetro fac aumenta. A justificativa para o uso deste parâmetro é feita nos capítulos 3 e 4 em função da alta dimensão dos dados, que afeta as distâncias entre pontos adjacentes. Dessa forma, o resultado obtido era esperado e confirma a utili-

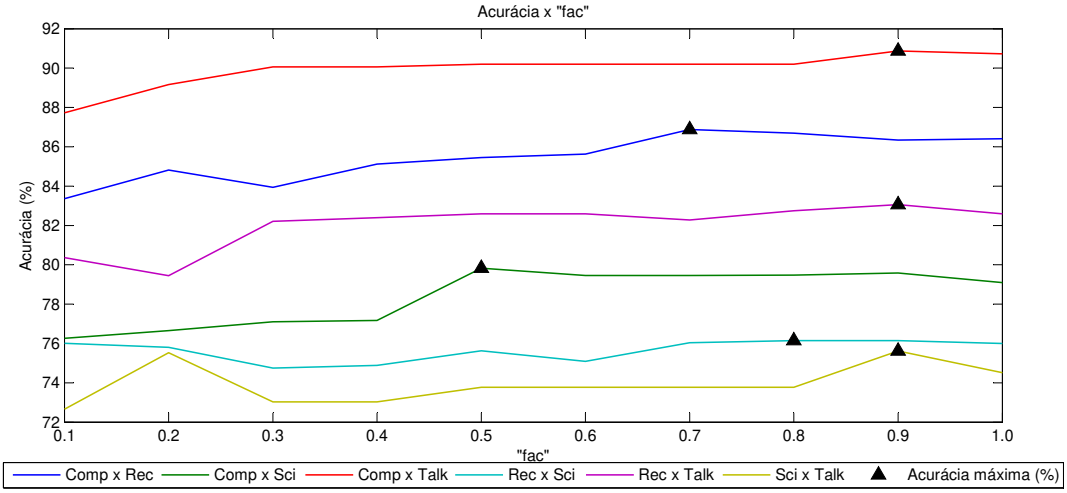


Figura 23 – Sensibilidade do eNNA quanto à variação no valor de fac: *20 Newsgroup*

zação da equação (4.8), seja para definição de limiar de decisão, como em FLEXFIS-Class SM, ou como valor inicial de dispersão, como em eNNA.

Após análise dos parâmetros que influenciam no desempenho do algoritmo eNNA, são comparados na Tabela 2 os resultados de todos os algoritmos, em função do número de grupos/regras/neurônios utilizados na estrutura do modelo, a acurácia em porcentagem e o tempo de processamento por amostra do conjunto de dados em milissegundos. É importante salientar que a acurácia apresentada é o valor máximo encontrado e seus respectivos tempos e quantidade de grupos. Para eNNA, o valor máximo encontrado é o mesmo destacado na Figura 23, para os parâmetros ρ e fac definidos. Para o método PC-ELM é apresentado o valor médio com respectivo desvio padrão de um total de 10 execuções, devido à escolha aleatória dos dados em cada execução. Os métodos evolutivos foram avaliados utilizando todos os dados em um fluxo, incluindo os dados utilizados pelo eNNA na etapa de evolução da estrutura.

Os métodos utilizados na comparação também possuem alguns parâmetros que influenciam no resultado final. Todos os métodos foram executados para cada valor dos parâmetros e apenas os valores que produziram um melhor desempenho em todos os casos são apresentados. Para o método eClass0, dois parâmetros são considerados, sendo $\rho = 0.5$ o valor que conduz ao desvio padrão do grupo e $r_0 = 0.5$ o valor inicial da dispersão do grupo. O método FLEXFIS-Class SM possui dois parâmetros, $ganho\_inicial = 0.5$ referente ao parâmetro que ajuda na definição da taxa de atualização do algoritmo eVQ e $fac = 1$ referente ao fator de multiplicação para o limiar de decisão, conforme utilizado na equação (4.8). Em SVM, foram testadas diferentes funções kernel como linear, quadrática, polinomial com grau 3 e função base radial; bem como testados os métodos para encontrar um hiperplano separador que são: programação quadrática, otimização sequencial mínima e quadrados mínimos. O melhor resultado em todos os casos foi utilizando a

Tabela 2 – Comparação de classificadores com *20 Newsgroup*

Modelo	# Grupo/ Regra/Neurônio	Acurácia (%)	Tempo/ amostra (ms)
Comp X Rec			
eNNA	27	86.88	0.24
eClass0	2	61.77	0.85
FLEXFIS-Class SM	27	49.72	4.40
AnYa-Class	6	44.64	1.10
SVM	-	93.40	1.00
CART	-	85.74	5.40
kNN	-	95.03	368.3
PC-ELM	500	94.40±0.26	1.10
Comp X Sci			
eNNA	26	79.83	0.25
eClass0	2	56.13	0.83
FLEXFIS-Class SM	93	50.32	16.00
AnYa-Class	14	58.68	1.90
SVM	-	86.07	1.50
CART	-	78.34	7.00
kNN	-	88.59	325.20
PC-ELM	500	88.37±0.21	1.20
Comp X Talk			
eNNA	22	90.87	0.24
eClass0	2	63.54	0.83
FLEXFIS-Class SM	30	45.41	5.60
AnYa-Class	8	78.42	1.50
SVM	-	94.38	1.30
CART	-	86.28	5.50
kNN	-	95.85	356.00
PC-ELM	500	94.40±0.17	1.20
Rec X Sci			
eNNA	26	76.15	0.26
eClass0	2	55.48	0.82
FLEXFIS-Class SM	15	50.21	3.00
AnYa-Class	8	52.45	1.80
SVM	-	87.81	1.50
CART	-	79.52	6.20
kNN	-	90.45	414.1
PC-ELM	500	89.99±0.24	1.20
Rec X Talk			
eNNA	27	83.06	0.33
eClass0	2	60.09	0.79
FLEXFIS-Class SM	3	48.40	1.30
AnYa-Class	7	84.08	1.40
SVM	-	89.68	0.87
CART	-	82.19	6.50
kNN	-	91.68	397.3
PC-ELM	500	90.43±0.21	1.20
Sci X Talk			
eNNA	26	75.62	0.26
eClass0	2	58.45	0.80
FLEXFIS-Class SM	4	46.36	1.40
AnYa-Class	9	47.65	1.60
SVM	-	85.28	1.50
CART	-	75.71	6.40
kNN	-	90.43	344.80
PC-ELM	500	88.73±0.23	1.20

função kernel linear e otimização sequencial mínima (SMO, *Sequential Minimal Optimization*). No método kNN, os valores dos parâmetros para os conjuntos são respectivamente $k = \{17, 15, 18, 19, 20, 17\}$. No último método, PC-ELM, é apresentado o resultado verificando a ação dos coeficientes de regularização. Os valores para os coeficientes encontrados são $C = \{256, 512, 512, 256, 128, 1024\}$.

Os resultados apresentados na Tabela 2 mostram que o desempenho do classificador eNNA, considerando acurácia e tempo de execução por amostra, é superior ao desempenho dos classificadores evolutivos eClass0, FLEXFIS-Class SM e AnYa-Class, exceto para o conjunto Rec X Talk, em específico, em que o classificador AnYa-Class teve um desempenho superior no quesito acurácia. É relevante comentar que os classificadores evolutivos FLEXFIS-Class SM e AnYa-Class foram capazes de definir uma estrutura, criando grupos ao longo da execução. Entretanto, esses não foram suficientes para uma boa generalização e classificação dos dados. Devido à natureza dos dados e dos algoritmos, os grupos podem ter sido atualizados com dados de classes distintas, contribuindo assim para um desempenho insatisfatório nesses casos. Parte dessa análise é estendida ao classificador eClass0, mas com o agravante de ter apenas definido dois grupos no início do processo de evolução, um para cada classe distinta. Ao longo da apresentação dos dados, o algoritmo não foi capaz criar novos grupos de modo a aumentar a acurácia, ou, melhorar a classificação.

Considerando ainda apenas os classificadores evolutivos, eNNA obteve menor tempo de processamento por amostra em todos os conjuntos de dados, mesmo quando com uma complexidade maior da estrutura. Grande contribuição para este resultado está no fato de evoluir a estrutura do modelo apenas por um determinado período inicial do processo, definido por W_{init}. Considerando que os seis conjuntos de dados em *20 Newsgroup* tem aproximadamente 8000 amostras cada, com $W_{init} = 150$, conforme utilizado neste experimento, a estrutura é evoluída para aproximadamente 2% dos dados totais apresentados. Entretanto, nos outros classificadores, a estrutura é evoluída ao longo da apresentação de todas as amostras. O fato interessante nesse resultado é que, mesmo considerando muito menos amostras para evoluir a estrutura, os resultados foram superiores em quase todos os casos. Adiante uma análise para outros valores de W_{init} é considerada justificando o uso de apenas 150 amostras.

A comparação do eNNA com classificadores não evolutivos como SVM, CART, kNN e PC-ELM tem como principal objetivo avaliar o desempenho considerando apenas a acurácia de classificação. Como processam os dados de forma distinta, a comparação levando em consideração o tempo de processamento, por exemplo, não deve ser feita. Entretanto, os métodos utilizados são considerados estado da arte na classificação de padrões e ajudam na análise com relação à eficácia do algoritmo no geral. De imediato, é considerado que os algoritmos não evolutivos podem atingir valores de acurácia superiores aos evolutivos pelo simples fato de terem acesso a todos os dados disponíveis e assim considerar cada contribuição no treinamento do seu modelo. Esse fato é comprovado na Tabela 2, na qual os resultados mostram que eNNA obteve acurácia inferior aos métodos SVM, kNN e PC-ELM em todos os conjuntos considerados para *20 Newsgroup*. O método CART entretanto obteve resultados superiores apenas nos conjuntos Rec x Sci e Sci x Talk, sendo superado pelo eNNA nos outros quatro conjuntos, mas com valores de acurácia

bastante competitivos em todos os casos.

A complexidade do modelo nessa comparação não é levada em consideração, já que apenas o modelo PC-ELM possui uma estrutura semelhante, utilizando uma rede neural. Entretanto, vale a apresentação dos resultados obtidos para SVM quanto à quantidade de vetores suporte encontrados e à quantidade de nós obtidos no treinamento do método CART, os quais formam a estrutura dos métodos. Os valores encontrados em cada caso, para vetores suporte e nós, foram: Comp x Rec - $\{2362, 471\}$, Comp x Sci - $\{2475, 693\}$, Comp x Talk - $\{1943, 437\}$, Rec x Sci - $\{2583, 693\}$, Rec x Talk - $\{2313, 609\}$ e Sci x Talk - $\{2338, 643\}$. Os valores mostram que dado o conjunto total de dados utilizados para treinamento em SVM, são utilizados mais da metade desses como vetores suporte, ou seja, vetores que serão utilizados na classificação de novas amostras. Para CART, foram utilizados menos nós, mas a estrutura final da árvore de decisão ainda é bastante complexa.

Na Figura 24 são apresentadas evoluções de acurácia e grupos ao longo das etapas de identificação da estrutura e estimação de parâmetros para cada um dos seis conjuntos formados a partir do *20 Newsgroup*. Na etapa indicada em cada um dos seis gráficos como "etapa de aquecimento" (identificação da estrutura), não é calculada a taxa de classificação, mesmo que as classes reais estejam disponíveis. Por esse motivo, o período inicial (0 até W_{init}) aparece com valor zero. Porém, é nesse período que a estrutura é evoluída, adicionando e excluindo grupos, conforme apresentado no gráfico inferior em cada uma das sub Figuras 24a-24f. A evolução da estrutura mostra determinados momentos de crescente adição de grupos e outros de crescente poda de grupos. Essa é uma das vantagens dos algoritmos evolutivos em que apenas o número de grupos necessário é utilizado e caso contrário estes são eliminados. Apesar de parecer uma ação drástica, apenas grupos com baixa qualidade são retirados.

A seguir, os mesmo aspectos anteriores são analisados para um novo conjuntos de dados, o *Farm Ads*. Na primeira análise é verificada a influência do parâmetro ρ no desempenho de classificação dos dados. A Figura 25 mostra os resultados para o conjunto em cada uma das 10 execuções, variando o parâmetro ρ no intervalo de 0 a 1. O padrão encontrado no conjunto de dados *20 Newsgroup* também é observado para *Farm Ads* em relação à variação do desempenho. É observado que o valor médio da acurácia aumenta com o aumento do valor de ρ, mesmo com uma variação atípica para $\rho = 0.8$. O valor médio neste conjunto de dados varia, aproximadamente, em 3% no intervalo considerado para o parâmetro. Entretanto, novamente é verificado o aumento da acurácia em função de ρ. A segunda análise considera também a variação do parâmetro fac, na faixa de valores entre 0 e 1. Dessa forma, na Figura 25 é observado que o maior valor de média de acurácia acontece para $\rho = 0.9$, como na maioria dos casos para o conjunto *20 Newsgroup*, reforçando a influência do parâmetro, regulando a taxa de convergência da dispersão ao

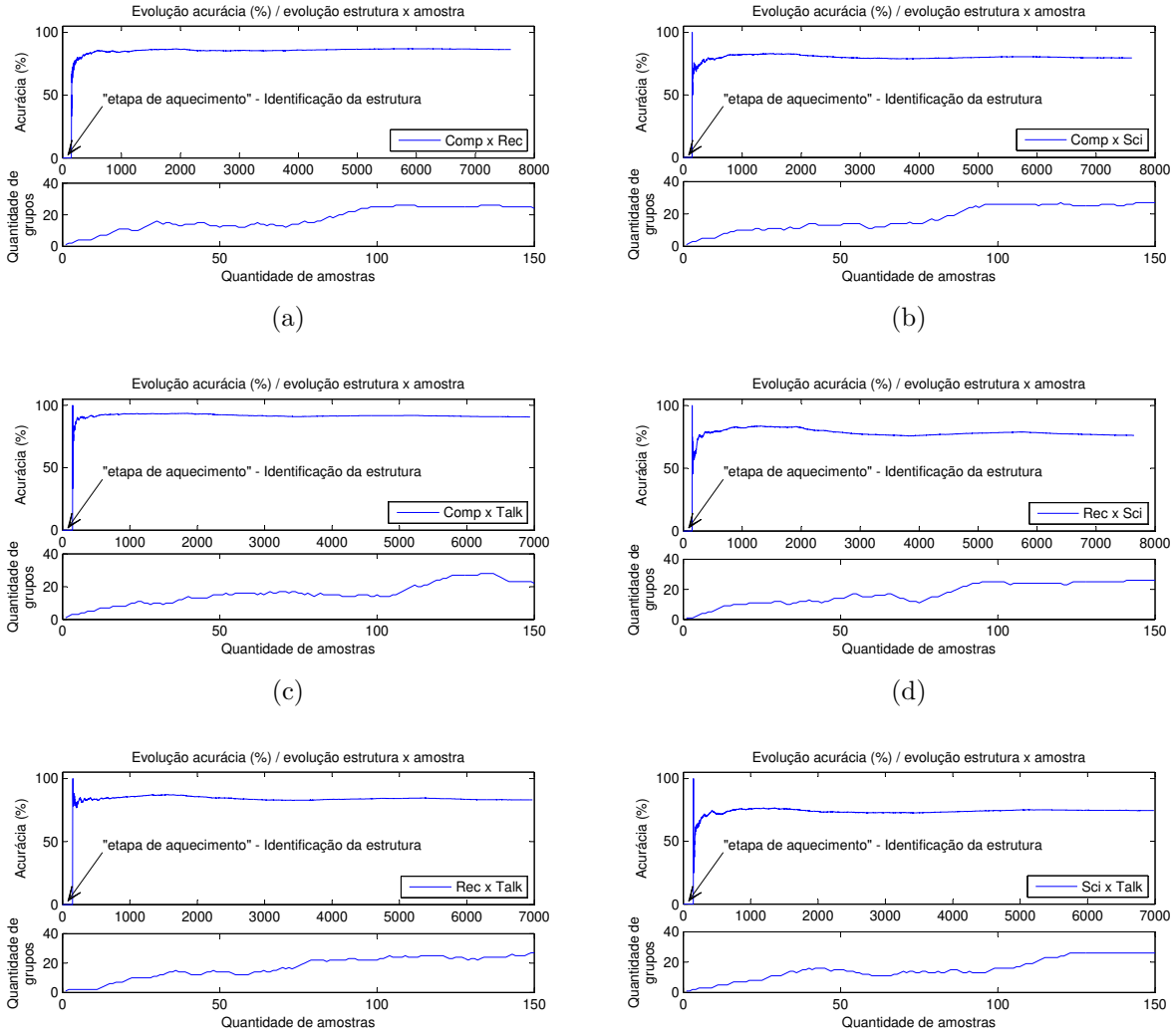


Figura 24 – Evolução da acurácia e da quantidade de grupos para *20 Newsgroup* no eNNA

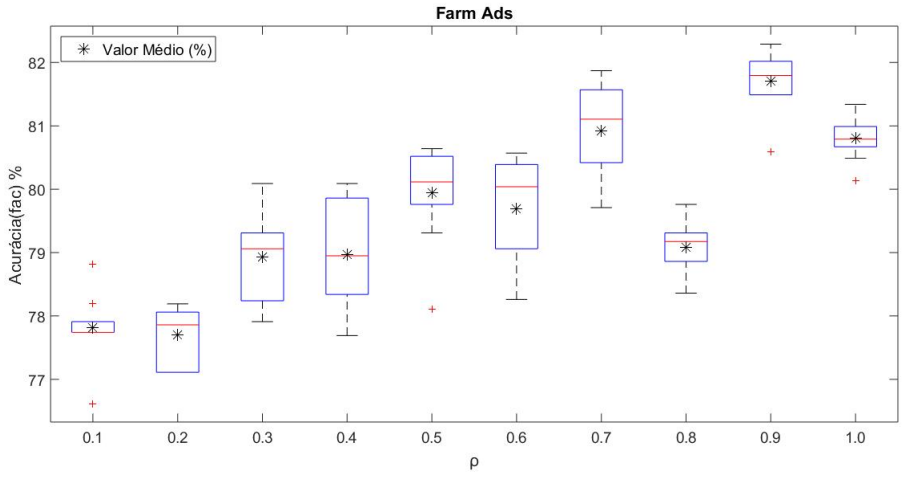


Figura 25 – Sensibilidade do eNNA quanto à variação do ρ: *Farm Ads*

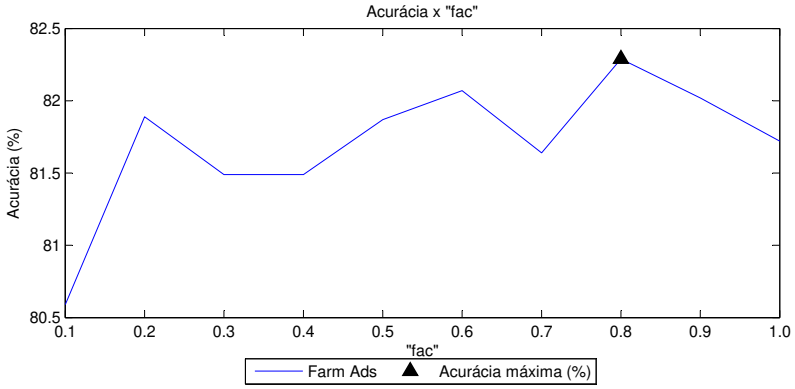


Figura 26 – Sensibilidade do eNNA quanto à variação no valor de fac: *Farm Ads*

desvio padrão do grupo. O diagrama de caixa, neste caso, mostra que mais de 50% dos resultados estão acima da média encontrada e o valor considerado "*outlier*" está no mesmo nível de acurácia das outras melhores médias, $\rho = 0.7$ e $\rho = 1$. Essa análise suporta a escolha dos resultados obtidos para $\rho = 0.9$ nas próximas análises.

Uma vez analisado o efeito de ρ, a Figura 26 mostra o valor de acurácia encontrado pelo algoritmo eNNA em cada uma das 10 execuções quanto à variação do parâmetro fac. A curva mostra que, neste conjunto de dados, o melhor desempenho é atingindo com valor $fac = 0.8$, reforçando a influência da dimensão dos dados na definição dos grupos, conforme expressão (4.8). Entretanto, observa-se que a variação da acurácia em *Farm Ads*, assim como nos outros conjuntos de dados, é pequena apesar de presente. Os resultados encontrados para eNNA e todos os outros classificadores evolutivos comparados para *Farm Ads* são apresentados na Tabela 3, seguindo as mesmas considerações feitas anteriormente, para valores de parâmetros que levaram ao melhor desempenho de acurácia. No método eClass0 é apresentado resultado para $\rho = 0.5$ e $r_0 = 0.5$. Em FLEXFIS-Class SM os parâmetros são $ganho\_inicial = 0.5$ e $fac = 1$. Nos métodos não evolutivos, para SVM é definido função kernel linear e método dos quadrados mínimos (LS, *Least Square*). Em kNN é utilizado o valor $k = 1$, com melhor desempenho entre os valores analisados. Por último, em PC-ELM, o valor do coeficiente de regularização utilizado é $C = 64$. A

Tabela 3 – Comparação de classificadores com *Farm Ads*

Modelo	**# Grupo/ Regra/Neurônio**	**Acurácia (%)**	**Tempo/ amostra (ms)**
eNNA	28	82.29	1.7
eClass0	2	46.54	2.3
FLEXFIS-Class SM	2	55.68	3.7
AnYa-Class	5	49.24	4.4
SVM	-	89.23	3.4
CART	-	84.02	17.0
kNN	-	87.78	54.3
PC-ELM	500	85.80±0.64	2.7

Tabela 3 mostra que o método eNNA é superior, tanto na acurácia quanto no tempo

de processamento, à todos os métodos evolutivos considerados: eClass0, FLEXFIS-Class SM, AnYa-Class. Novamente, são considerados nessa análise inicial, acurácia e tempo de processamento por amostra e, em ambos, os resultados são melhores. Para *Farm Ads* é relevante ressaltar que eClass0 e FLEXFIS-Class SM apenas definiram duas regras, sendo uma para cada classe, no início da execução do algoritmo. Mesmo que AnYa-Class tenha conseguido definir novas nuvens de dados, nenhum dos métodos foi capaz de evoluir uma estrutura que melhor representasse os dados na dimensão considerada.

O tempo de processamento por amostra na Tabela 3 mostra que, mesmo com uma estrutura mais complexa que os outros métodos evolutivos, eNNA processa os dados em um menor tempo por amostra. O valor $W_{init} = 150$ representa aproximadamente 4% dos dados apresentados de forma contínua nos algoritmos. Mesmo assim, a reduzida quantidade de dados utilizados na identificação da estrutura do eNNA foi capaz de atingir um melhor desempenho em comparação aos modelos que utilizam 100% dos dados na identificação da estrutura. Para *Farm Ads* também são considerados casos com outras inicializações do parâmetro W_{init}, analisados mais adiante.

Novamente, são considerados classificadores não evolutivos representativos do estado da arte para comparação com eNNA. Essa análise introduz confiabilidade nos resultados encontrados, tendo em vista que são métodos consagrados da literatura. Dessa forma, apenas acurácia será considerada nesta análise sendo que o tempo de processamento é apenas indicado de modo informativo para estes métodos. Assim, os classificadores não evolutivos possuem um melhor desempenho que eNNA e todos os classificados evolutivos para casos de alta dimensão. Como já apresentado, com aproximadamente 4% dos dados apresentados de forma contínua, dado a dado, eNNA atinge 82.29% de acurácia contra 89.23% no melhor dos classificadores não evolutivos, SVM. Além disso, o classificador CART novamente obteve um desempenho bastante próximo ao encontrado com eNNA.

Por último, a análise da complexidade da estrutura é feita entre eNNA e os métodos não evolutivos. Em SVM, os vetores suporte são considerados como definidores da estrutura, uma vez que são utilizados na classificação dos dados do conjunto de treinamento. Em CART, a quantidade de nós encontrados na divisão do espaço de dados define a estrutura aqui considerada. Para esses casos, são definidos: 2071 vetores suporte e 219 nós. É importante ressaltar que, *Farm Ads* possui 4143 amostra divididas em conjunto de treinamento e teste, com 2072 e 2071 dados, respectivamente. Dessa forma, apenas 1 dado não foi utilizado como vetor suporte em SVM.

Na Figura 27 é apresentado a evolução da estrutura e estimação dos parâmetros para eNNA ao longo da apresentação dos dados. No período definido como "etapa de aquecimento" (identificação da estrutura), não é calculada acurácia. A parte inferior da Figura 27 apresenta a evolução dos grupos durante a etapa de aquecimento. A presença de momentos de inserção e exclusão de grupos é essencial na evolução da estrutura do

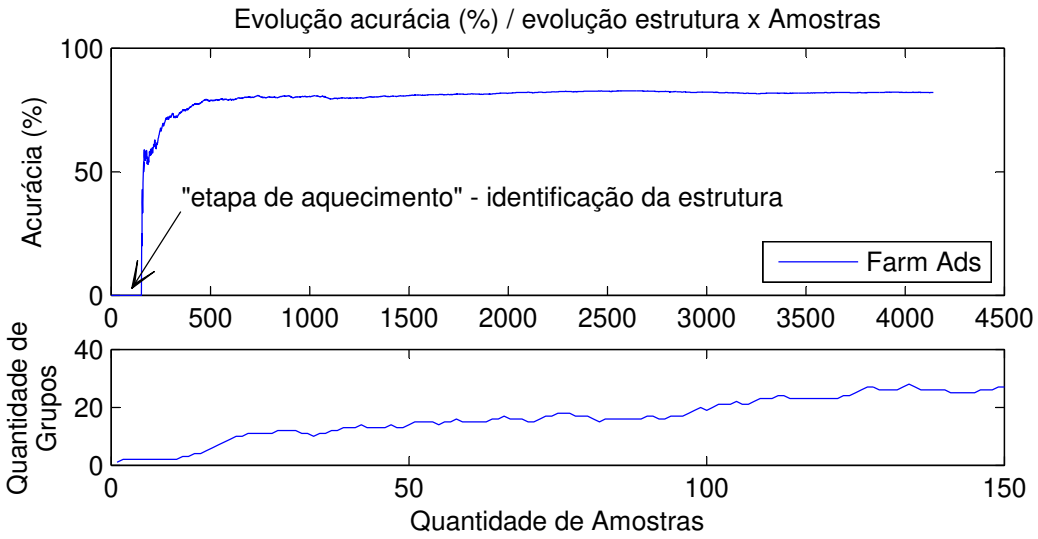


Figura 27 – Evolução da acurácia para *Farm Ads* no eNNA

modelo, considerando apenas grupos com alta qualidade.

A análise da quantidade de dados utilizados na etapa de identificação da estrutura é feita na sequência, onde W_{init} é inicializado com 150, 500 e 1000 amostras e as análises de acurácia (%), tempo de processamento total em segundos e quantidade de grupos são feitas. A Figura 28 apresenta um gráfico de barras para cada um dos conjuntos considerados para *20 Newsgroup*. Acurácia, tempo de processamento e quantidade de grupos são colocados lado a lado e sua análise é direta. O desempenho do algoritmo eNNA é pouco influenciado pelo parâmetro W_{init}, responsável pela duração da etapa de identificação da estrutura, quando considerando apenas acurácia. É possível observar que, em todos os conjuntos, a acurácia sofreu pouca modificação, aproximadamente 3% no melhor dos casos para Sci x Talk. Entretanto, o tempo de processamento e a quantidade de grupos são diretamente afetados por W_{init}. O tempo de processamento total teve um aumento médio de aproximadamente 5 vezes, no intervalo de 150 a 1000 amostras utilizadas. A análise apenas do tempo não nos leva a uma conclusão ruim considerando o aumento de amostras, entretanto, considerando que o aumento médio da acurácia é de 1,5% não é justificável o aumento do número de amostras iniciais na etapa de aquecimento. Ainda, é analisada a complexidade dos modelos frente à quantidade de dados inicializados. O resultado mais uma vez afirma que não há razões para utilizar valores altos para W_{init}. A quantidade de grupos evoluídos ao longo da etapa de identificação da estrutura para $W_{init} = 150$ em média foi de 25 grupos, enquanto que, para 1000 amostras, foi de 56 grupos; um aumento de aproximadamente 30 grupos para um ganho médio de apenas 1,5% na acurácia. Com esses resultados é concluído que a utilização de 150 amostras são suficientes para um bom desempenho do eNNA, sendo o melhor custo benefício entre os três cenários verificados.

Para o conjunto *Farm Ads* foi realizado o mesmo experimento e o resultado é

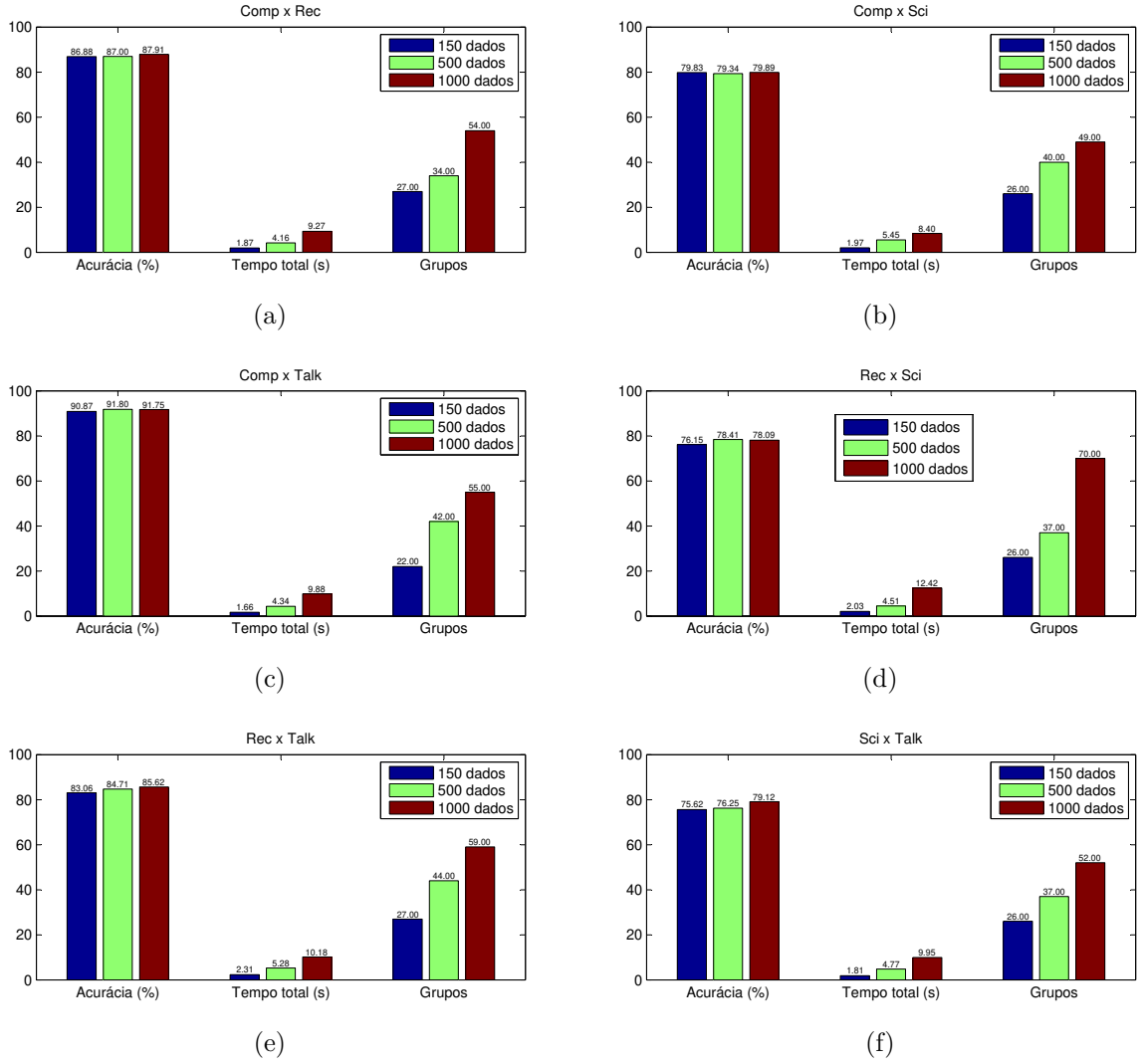


Figura 28 – Desempenho do eNNA em função de W_{init} para *20 Newsgroup*

apresentado na Figura 29, também com análises comparativas entre acurácia, tempo de processamento total em segundos e quantidade de grupos evoluídos para 150, 500 e 1000 amostras na etapa de identificação da estrutura. Novamente, é possível observar que o aumento na acurácia do classificador eNNA não é expressivo o suficiente para justificar o aumento da quantidade W_{init}. O tempo de processamento total dos dados teve um aumento de 36,49 segundos, indo de 6,94 segundos com $W_{init} = 150$ para 43,43 segundos com $W_{init} = 1000$. Além disso, a quantidade de grupos evoluídos para 150 amostras foi de 28 enquanto que para 1000 foi de 58 grupos, significando um aumento de 30 grupos. Nesses casos, o aumento da acurácia foi de apenas 1,64% e, como nos casos anteriores, a utilização de $W_{init} = 150$ representa o melhor cenário dentre os verificados, considerando o conjunto acurácia, tempo total e número de grupos.

É importante ressaltar aqui que, o tempo de processamento é diretamente proporcional à dimensão dos dados. Como pode ser observado nas Figuras 28 e 29, para o mesmo

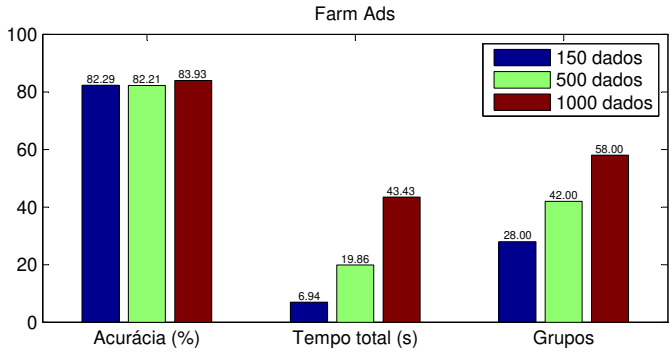


Figura 29 – Desempenho do eNNA em função de W_{init} para *Farm Ads*

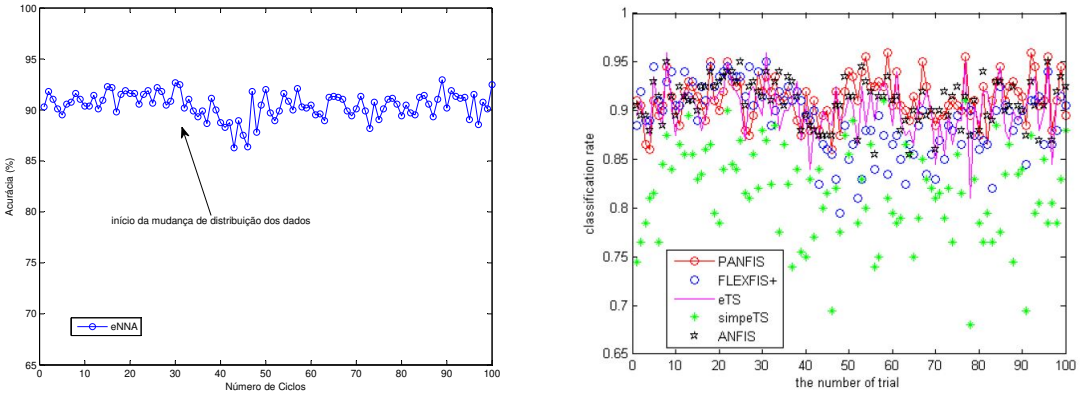
aumento na quantidade de amostras, $W_{init} = 150$ para $W_{init} = 1000$, houve um aumento no tempo de processamento de aproximadamente 5 vezes nos conjuntos *20 Newsgroup* e de aproximadamente 7 vezes no conjunto *Farm Ads*.

Análise em cenário com mudança de contexto

Como já comentado em capítulos anteriores, uma das características dos algoritmos evolutivos é a adaptação da estrutura e dos parâmetros frente a mudanças na distribuição dos dados de entrada. Esta mudança de distribuição, também chamada de mudança de contexto (*concept drift*), é bastante presente em aplicações reais, com dados não controlados. Para simular tais casos, conjuntos de dados sintéticos são gerados de tal forma que a distribuição dos dados muda ao longo do tempo.

Nesta seção é analisado o desempenho do algoritmo eNNA aplicado a um conjunto de dados sintético chamado *Hyperplane*, com dimensão 4 e 120 mil amostras. As duas classes distintas no conjunto podem ser separadas por um hiperplano e sua posição é variada ao longo do tempo. Os procedimentos seguidos são os mesmo descritos em (PRATAMA *et al.*, 2014). Nesse trabalho, são comparados os algoritmos evolutivos PANFIS-ERLS, PANFIS-RLS, ANFIS (JANG, 1993), eTS (ANGELOV; FILEV, 2004), Simp_eTS (ANGELOV; FILEV, 2005) e FLEXFIS+ (LUGHOFER, 2011b). O desempenho dos algoritmos é avaliado apresentando sequencialmente 1200 amostras por um total de 100 vezes, de forma cíclica. A cada ciclo de 1200 amostras são verificadas a acurácia e a quantidade de grupos evoluídos, sendo esses apresentados respectivamente nas Figuras 30 e 31.

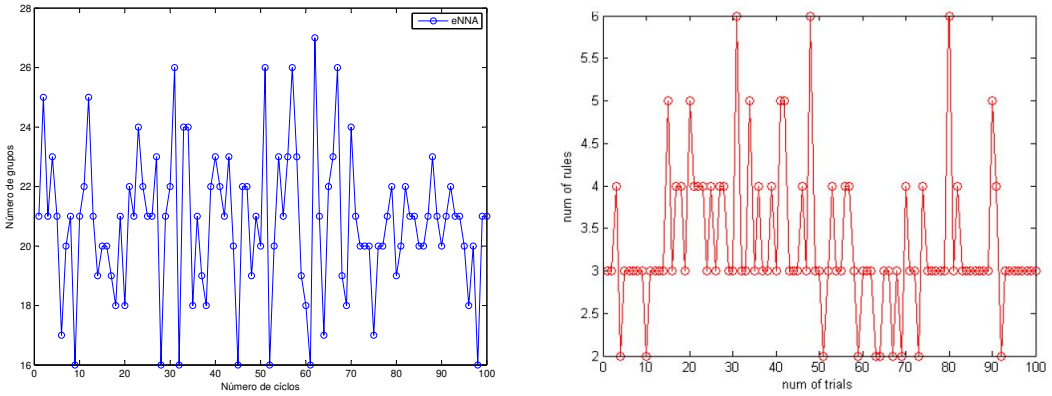
A Figura 30 mostra que eNNA se comporta de forma competitiva quando comparado com os métodos utilizados em Pratama *et al.* (2014). A acurácia em cada um dos ciclos, conforme apresentado, mostra que a partir do ciclo 33 é iniciada a mudança da distribuição dos dados, levando a um ligeiro decrescimento na acurácia em todos os modelos. Entretanto, a cada novo ciclo, nova estrutura é evoluída de forma a se adaptar à nova distribuição dos dados, aumentando o desempenho progressivamente até novamente



(a) (b) (PRATAMA *et al.*, 2014)

Figura 30 – Acurácia (%) em cada ciclo (a) eNNA e (b) PANFIS, ANFIS, eTS, Simp_eTS e FLEXFIS+

estabilizar próximo do ciclo 50.



(a) (b) (PRATAMA *et al.*, 2014)

Figura 31 – Número de grupos evoluídos em cada ciclo (a) eNNA e (b) PANFIS

A Figura 31 por sua vez, apresenta a evolução da estrutura quando utilizando o eNNA e PANFIS. O padrão encontrado é comparativo, mas é clara a utilização de um menor número de grupos em PANFIS quando comparado com eNNA. Em média, são 3.19 contra 20.77 grupos, respectivamente.

Os resultados médios de acurácia em cada um dos algoritmos comparados são apresentado na Tabela 4, incluindo o resultado médio para eNNA. Todos os resultados, que não eNNA, são reproduzidos de Pratama *et al.* (2014).

Os resultados encontrados indicam a qualidade do algoritmo proposto eNNA frente à mudança de contexto em dados, quando utilizado o método cíclico para apresentação desses. Como os dados são apresentados na forma de um fluxo, o algoritmo eNNA é capaz de evoluir sua estrutura com rapidez e eficiência, sempre que submetido a novo ciclo dentro do fluxo contínuo dos dados.

Tabela 4 – Resultado comparativo - Conjunto de dados - *Hyperplane*

Modelo	**# Grupos/Regras**	**Acurácia %**
eNNA	20.77	90.45
PANFIS-ERLS(PRATAMA *et al.*, 2014)	3.19	91.55
PANFIS-RLS(PRATAMA *et al.*, 2014)	3.19	91.26
ANFIS(PRATAMA *et al.*, 2014)	16.00	90.90
eTS(PRATAMA *et al.*, 2014)	14.49	90.16
Simp_eTS(PRATAMA *et al.*, 2014)	5.23	81.71
FLEXFIS+(PRATAMA *et al.*, 2014)	7.22	89.06

5.4 Resumo

Este capítulo tratou da avaliação e comparação de classificador eNNA utilizando os conjuntos de dados *20 Newsgroup*, *Farm Ads* e *Hyperplane*. Foi avaliada a influência dos parâmetros ρ e fac no desempenho do eNNA, assim como foi feita uma análise de acurácia e complexidade da estrutura durante as etapas de identificação da estrutura e estimação dos parâmetros. Por último, avaliou-se o desempenho do eNNA frente à mudança de distribuição dos dados durante a apresentação dos mesmos na forma de um fluxo contínuo, incluindo comparações com algoritmos representativos do estado da arte na área.

6 Conclusão

Sistemas evolutivos e processamento de dados de alta dimensão em um fluxo contínuo são de grande importância prática e atualmente sobre intensa investigação. Tais sistemas são importantes porque são capazes de desenvolver sua estrutura e os parâmetros correspondentes de forma contínua para se adaptar às mudanças nos dados ao longo do tempo. Em particular, classificadores evolutivos têm sido desenvolvidos e aplicados às tarefas do mundo real, mas não em casos com dados de alta dimensão.

Esta dissertação desenvolveu um classificador evolutivo baseado em redes neurais artificiais, denominado Neuro Classificador Evolutivo para Espaço de Alta Dimensão (eNNA). Dividido em duas etapas essenciais, o eNNA identifica sua estrutura primeiramente e adapta seus parâmetros em um segundo momento. A identificação da estrutura utiliza agrupamento evolutivo elipsoidal. Sempre que um novo dado é apresentado ao classificador, esse pode representar a criação de um novo grupo ou ser atribuído a um grupo existente. Na estrutura final, cada neurônio da camada intermediária de uma rede neural é associado a um grupo. Uma classe pode ter vários neurônios da camada intermediária a ela associada e os pesos sinápticos são atualizados sempre que um novo dado é apresentado ao classificador.

O algoritmo eNNA foi avaliado e comparado com classificadores evolutivos considerados estado da arte como eClass0, FLEXFIS-Class SM e AnYa-Class utilizando dados de alta dimensão. Esses classificadores são compostos por regras nebulosas e definem suas estruturas baseados na densidade dos dados e em medidas de distâncias. A análise de desempenho baseada em taxa de classificação em porcentagem, tempo de processamento por amostra e complexidade da estrutura sugere que a abordagem neuro evolutiva eNNA é competitiva em classificação de dados de alta dimensão em fluxo contínuo. Os conjuntos de dados considerados, *20 Newsgroup* e *Farm Ads*, possuem respectivamente 12000 e 54877 atributos, sendo essas as dimensões dos dados. O eNNA foi capaz de identificar no espaço de dados uma estrutura de uma rede neural que obteve taxa de classificação superior aos classificadores evolutivos considerados em praticamente todos os subconjuntos em *20 Newsgroup* e em *Farm Ads*. Além disso, o tempo de processamento por amostra em eNNA é menor que o de todos os classificadores evolutivos em todos os conjuntos de dados, sendo esse um resultado importante se tratando de algoritmos para o processamento contínuo em tempo real.

A consideração feita na etapa de identificação da estrutura, em que apenas grupos com eixos paralelos são criados e/ou atualizados, reflete diretamente no tempo de processamento quando a dimensão dos dados aumenta. Enquanto os algoritmos evolutivos

estado da arte evoluem suas estruturas ao longo de todo o fluxo de dados, o eNNA dedica apenas uma parte do fluxo para essa tarefa. É evidenciado nesta dissertação que a utilização de uma quantidade de dados limitada na identificação da estrutura não afeta significativamente a taxa de classificação do algoritmo eNNA. Esse ponto dá a liberdade necessária para processar os dados de alta dimensão sem abrir mão do desempenho.

O algoritmo eNNA foi também avaliado usando conjunto de dados não estacionários e se mostrou igualmente competitivo quando comparado com algoritmos evolutivos como PANFIS-ERLS, PANFIS-RLS, ANFIS, eTS, Simp_eTS e FLEXFIS+. A taxa de classificação média para as 100 execuções realizadas nesse caso foi tão precisa quanto às encontradas por classificadores alternativos. A complexidade da estrutura é maior para eNNA, mas não causa prejuízo em seu desempenho computacional.

Quando comparado aos classificadores não evolutivos como SVM, CART, kNN e PC-ELM, fica evidente a superioridade dos métodos não evolutivos frente ao evolutivo eNNA. Entretanto, vale a ressalva de que todos os métodos não evolutivos utilizaram 50% dos dados na fase de treinamento, fornecendo informação quanto à distribuição dos dados para os algoritmos em questão. A superioridade dos métodos não evolutivos não indica que o método eNNA é ineficaz no seu propósito. Isso mostra que o método eNNA, assim como os outros evolutivos, possui uma limitação de desempenho ditada pelos métodos semelhantes não evolutivos, como a rede neural utilizada em PC-ELM, por exemplo. Os resultados encontrados evidenciam o quão distante de um possível valor "ótimo" o algoritmo proposto está.

Como perspectivas futuras, fica a necessidade da busca por uma estrutura mais simples, com grupos mais representativos que os utilizados pelo eNNA atual. Mesmo com resultados satisfatórios em taxa de classificação e tempo de processamento, a estrutura evoluída mostra pontos passíveis de melhorias, principalmente em dados de uma mesma classe dispostos mais distantes da fronteira de decisão, por exemplo. Estudos para evoluir uma estrutura que represente melhor as fronteiras das classes são temas para trabalhos futuros, podendo melhorar a complexidade da estrutura identificada, reduzir o tempo de processamento por amostras e ainda obter melhores resultados.

Referências

ACHLIOPTAS, D. Database-friendly random projections. In: *Proceedings of the Twentieth Symposium on Principles of Database Systems*. Santa Barbara, California, USA: ACM, 2001. p. 274–281.

ANGELOV; P.; GIGLIO; V.; GUARDIOLA; C.; LUGHOFER; E.; LUJAN; J.M. An approach to model-based fault detection in industrial measurement systems with application to engine test benches. *Measurement Science and Technology*, Institute of Physics Publishing, p. 1809–1818, 2006.

ANGELOV, P. *Autonomous Learning Systems: From Data Streams to Knowledge in Real-time*. Chichester, UK: John Wiley & Sons, 2012.

ANGELOV, P.; FILEV, D. Simpl_ets: a simplified method for learning evolving takagi-sugeno fuzzy models. In: *Proceedings of the IEEE International Conference on Fuzzy Systems*. Reno, NV, USA: IEEE, 2005. p. 1068–1073.

ANGELOV, P.; FILEV, D. P.; KASABOV, N. *Evolving Intelligent Systems: Methodology and Applications*. Hoboken, NJ, USA: Wiley-IEEE Press, 2010.

ANGELOV, P.; YAGER, R. Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density. In: *Proceedings of the IEEE Workshop on Evolving and Adaptive Intelligent Systems*. Paris, France: IEEE, 2011. p. 62–69.

ANGELOV, P.; ZHOU, X. Evolving fuzzy systems from data streams in real-time. In: *Proceedings of the International Symposium on Evolving Fuzzy Systems*. Ambelside, UK: IEEE, 2006. p. 29–35.

ANGELOV, P.; ZHOU, X.; FILEV, D.; LUGHOFER, E. Architectures for evolving fuzzy rule-based classifiers. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. Montréal, Canada: IEEE, 2007. p. 2050–2055.

ANGELOV, P. P.; FILEV, D. P. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, v. 34, n. 1, p. 484–498, 2004.

ANGELOV, P. P.; ZHOU, X. Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems*, v. 16, n. 6, p. 1462–1475, 2008.

BARUAH, R. D.; ANGELOV, P.; ANDREU, J. Simpl_eclass: simplified potential-free evolving fuzzy rule-based classifiers. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. Anchorage, AK, USA: IEEE, 2011. p. 2249–2254.

BIFET, A.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B. MOA: massive online analysis. *Journal of Machine Learning Research*, v. 11, p. 1601–1604, 2010.

BREIMAN, L.; FRIEDMAN, J.; STONE, C. J.; OLSHEN, R. A. *Classification and Regression Trees*. New York, NY, USA: Chapman & Hall/CRC, 1984.

COSTA, B. S. J.; ANGELOV, P. P.; GUEDES, L. A. Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. *Neurocomputing*, Elsevier, v. 150, p. 289–303, 2015.

COVER, T. M.; HART, P. E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, v. 13, n. 1, p. 21–27, 1967.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2. ed. New York, NY: Wiley-Interscience, 2001.

FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, Wiley Online Library, v. 7, n. 2, p. 179–188, 1936.

FIX, E.; HODGES, J. L. *Discriminatory analysis, nonparametric discrimination: Consistency properties*. Randolph Field, TX, 1951. Technical Report 4, n. 3, 477 p.

GINI, C. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica*, v. 1, 1912.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994. ISBN 0023527617.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, Elsevier, v. 4, n. 2, p. 251–257, 1991.

HUANG, G.-B.; ZHOU, H.; DING, X.; ZHANG, R. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 42, n. 2, p. 513–529, 2012.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, Elsevier, v. 70, n. 1, p. 489–501, 2006.

JANG, J.-S. R. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics*, IEEE, v. 23, n. 3, p. 665–685, 1993.

JOHNSON, W.; LINDENSTRAUSS, J. Extensions of Lipschitz mappings into a Hilbert space. In: *Conference in modern analysis and probability (New Haven, Conn., 1982)*. [S.l.]: American Mathematical Society, 1984. v. 26, p. 189–206.

KULAIF, A. C. P. *Técnicas de regularização para máquinas de aprendizado extremo*. Dissertação (Mestrado) — Universidade de Campinas, Campinas-SP, 2014.

LAN, Y.; SOH, Y. C.; HUANG, G.-B. A constructive enhancement for online sequential extreme learning machine. In: *Proceedings of the International Joint Conference on Neural Networks*. Piscataway, NJ, USA: IEEE Press, 2009. p. 208–213.

LEMOS, A.; CAMINHAS, W.; GOMIDE, F. Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, v. 19, n. 1, p. 91–104, 2011.

LETRAS, A. B. de. *Busca no Vocabulário*. 2009. Disponível em: <http://www.academia.org.br/nossa-lingua/busca-no-vocabulario>. Acesso em: 02 Junho 2016.

LI, P.; HASTIE, T. J.; CHURCH, K. W. Very sparse random projections. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. Philadelphia, PA, USA: ACM, 2006. p. 287–296.

LIANG, N.-Y.; HUANG, G.-B.; SARATCHANDRAN, P.; SUNDARARAJAN, N. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, v. 17, n. 6, p. 1411–1423, 2006.

LICHMAN, M. *UCI Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>. Acesso em: 10 Maio 2016.

LUGHOFER, E. Extensions of vector quantization for incremental clustering. *Pattern Recognition*, Elsevier, v. 41, n. 3, p. 995–1011, 2008.

LUGHOFER, E. *Evolving Fuzzy Systems - Methodologies, Advanced Concepts and Applications*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2011.

LUGHOFER, E. On-line incremental feature weighting in evolving fuzzy classifiers. *Fuzzy Sets and Systems*, Elsevier, v. 163, n. 1, p. 1–23, 2011.

LUGHOFER, E.; ANGELOV, P.; ZHOU, X. Evolving single- and multi-model fuzzy classifiers with FLEXFIS-class. In: *Proceeding of the IEEE International Fuzzy Systems Conference*. London, UK: IEEE, 2007. p. 1–6.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, Elsevier, v. 7, n. 1, p. 1–13, 1975.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

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

MESTERHARM, C.; PAZZANI, M. J. Active learning using on-line algorithms. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. San Diego, California, USA: ACM, 2011. p. 850–858.

METZ, C. E. Basic principles of roc analysis. In: *Seminars in Nuclear Medicine*. [S.l.]: Elsevier, 1978. p. 283–298.

MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

PRATAMA, M.; ANAVATTI, S. G.; ANGELOV, P. P.; LUGHOFER, E. Panfis: A novel incremental learning machine. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 55–68, 2014.

PRECHELT, L. Early stopping-but when? In: *Neural Networks: Tricks of the trade*. New York, NY: Springer, 1998. p. 55–69.

ROSA, R.; GOMIDE, F.; DOVZAN, D.; SKRJANC, I. Evolving neural network with extreme learning for system modeling. In: *Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems*. Linz, Austria: IEEE, 2014. p. 1–7.

STEINBRUCH, A.; WINTERLE, P. *Geometria Analítica*. São Paulo-SP: Makron Books, 1987.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, n. 1, p. 116–132, 1985.

VAPNIK, V. N. *The nature of statistical learning theory*. New York, NY, USA: Springer, 1995.

WANG, D.; ZENG, X.-J.; KEANE, J. A. A simplified structure evolving method for Mamdani fuzzy system identification and its application to high-dimensional problems. *Information Sciences*, Elsevier, v. 220, p. 110–123, 2013.

WELLING, M. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto*, v. 3, 2005.

YOUNG, P. *Recursive Estimation and Time-series Analysis: An Introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1984.

ZADEH, L. A. Fuzzy sets. *Information and Control*, Elsevier, v. 8, n. 3, p. 338–353, 1965.

ZHAO, R.; MAO, K. Semi-random projection for dimensionality reduction and extreme learning machine in high-dimensional space. *IEEE Computational Intelligence Magazine*, v. 10, n. 3, p. 30–41, 2015.

ZHOU, X.; ANGELOV, P. An approach to autonomous self-localization of a mobile robot in completely unknown environment using evolving fuzzy rule-based classifier. In: *Proceedings of the Symposium on Computational Intelligence in Security and Defense Applications*. Honolulu, Hawaii: IEEE, 2007. p. 131–138.