UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PATRICK MARQUES CIARELLI

MODELO DE APRENDIZADO INCREMENTAL BASEADO EM UMA REDE NEURAL COM ARQUITETURA ADAPTATIVA

VITÓRIA
2012

PATRICK MARQUES CIARELLI

MODELO DE APRENDIZADO INCREMENTAL BASEADO EM UMA REDE NEURAL COM ARQUITETURA ADAPTATIVA

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Elias Silva de Oliveira.

Orientador: Prof. Dr. Evandro Ottoni Teatini Salles.

VITÓRIA
2012

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Ciarelli, Patrick Marques, 1982 -
C566m Modelo de aprendizado incremental baseado em uma rede neural com arquitetura adaptativa / Patrick Marques Ciarelli. - 2012.
204 f.: il.

Orientador: Elias Silva de Oliveira.
Coorientador: Evandro Ottoni Teatini Salles.
Tese (Doutorado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Redes neurais (Computação). 2. Algoritmos de expectativa de maximização. 3. Aprendizado do computador. I. Oliveira, Elias Silva de, 1963-. II. Salles, Evandro Ottoni Teatini. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

PATRICK MARQUES CIARELLI

MODELO DE APRENDIZADO INCREMENTAL BASEADO EM UMA REDE NEURAL COM ARQUITETURA ADAPTATIVA

Tese submetida ao programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do Grau de Doutor em Engenharia Elétrica.

Aprovada em 12 de Dezembro de 2012.

COMISSÃO EXAMINADORA

Prof. Dr. Elias Silva de Oliveira
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Evandro Ottoni Teatini Salles
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo

Prof. Dr. Alberto Ferreira de Souza
Universidade Federal do Espírito Santo

Profa. Dra. Claudine Badue
Universidade Federal do Espírito Santo

Dr. Fábio Daros de Freitas
Receita Federal do Brasil

Prof. Dr. Wagner Meira Jr.
Universidade Federal de Minas Gerais

Dedico esta Tese à minha família,
pelo apoio, compreensão e incentivo
que foram indispensáveis para a conclusão
de mais esta etapa da minha vida.

Agradecimentos

Dedico meus sinceros agradecimentos

– à Receita Federal do Brasil e ao Fundação de Amparo a Pesquisa do Espírito Santo pelo apoio dado através da bolsa de estudos durante o período parcial de realização desta Tese;

– ao Programa de Pós Graduação da Engenharia Elétrica e da Informática da UFES, e seus professores, principalmente aos meus orientadores Dr. Elias Silva de Oliveira e Dr. Evandro Ottoni Teatini Salles, que sempre foram prestativos e dispostos a ajudar;

– a todos os amigos da UFES que eu tive a felicidade de conhecer, em especial dos laboratórios CISNE e LCAD, que me incentivaram e ajudaram a realizar mais este passo importante da minha vida.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Lista de Siglas

ARIMA *Autoregressive Integrated Moving Average*.

CBO Classificação Brasileira de Ocupações.

CNAE Classificação Nacional de Atividades Econômicas.

DARN *Deterministic Adaptive RAM Networks*.

DC Distância crítica.

DENFIS *Dynamic Evolving Neural-Fuzzy Inference Systems*.

ECF *Evolving Classification Function*.

ECoS *Evolving Connectionist System*.

ECM *Evolving Clustering Method*.

EFuNN *Evolving Fuzzy Neural Network*.

EM *Expectation Maximization*.

eMLP *evolving Multi Layer Perceptron*.

ePNN *evolving Probabilistic Neural Network*.

eSOM *evolving Self-Organize Maps*.

IA Inteligência Artificial.

IBGE Instituto Brasileiro de Geografia e Estatística.

LVQ *Linear Vector Quantization*.

kNN *k Nearest Neighbor*.

MLP *Multi-Layer Perceptron*.

MMG Modelo de Mistura de Gaussianas.

MST *Minimum Spanning Tree*.

RAM *Random Access Memory*.

RGB *Red Green Blue*.

RNP Rede Neural Probabilística.

RNPe Rede Neural Probabilística evolutiva.

RNPI Rede Neural Probabilística Incremental.

RNPI-EM Rede Neural Probabilística Incremental Baseada em *Expectation Maximization*.

SCE *Sistemas Conexionistas Evolutivos*.

SIE *Sistemas Inteligentes Evolutivos*.

SOM *Self-Organizing Maps*.

SVM *Support Vector Machine*.

Tfi-df *Term frequency - inverse document frequency*.

UCI *University of California Irvine*.

VG-RAM WNN *Virtual Generalizing Random Access Memory Weightless Neural Networks*.

WebKB *World Wide Knowledge Base*.

ZISC *Zero Instruction Set Computer*.

Resumo

Técnicas de aprendizado incremental são aquelas cujo aprendizado acontece continuamente ao longo do tempo e não termina uma vez que os dados disponíveis tenham sido exauridos. Tais técnicas apresentam características úteis em situações em que os dados de um problema são adquiridos em pequenas quantidades ao longo do tempo, e também quando as características de uma tarefa estejam sujeitas a mudanças ao longo do tempo. No entanto, uma desvantagem dessas técnicas é que elas tendem a produzir modelos com estrutura crescente em relação à quantidade de dados usada para treinamento e, com o passar do tempo, podem produzir modelos com estrutura grande o suficiente para tornar inviável sua aplicação para determinadas tarefas. Neste contexto, esta Tese de Doutorado propõe um modelo de aprendizado incremental supervisionado para problemas de classificação com estrutura reduzida, mas mantendo qualidade de resposta comparável às melhores técnicas incrementais.

O modelo proposto, denominado de Rede Neural Probabilística evolutiva (RNPe), é baseado na Rede Neural Probabilística, no Modelo de Mistura de Gaussianas e no algoritmo de *Expectation Maximization*. As principais características do modelo proposto são a capacidade de aprender continuamente durante toda sua existência e, além disso, requerer que cada amostra seja usada somente uma vez durante o treinamento. Acrescenta-se a estas duas características mencionadas o fato de que a RNPe proposta possuir a capacidade de alterar dinamicamente sua arquitetura para adaptar-se melhor a um determinado tipo de problema e também possuir uma estrutura flexível, sendo facilitada a tarefa de inserir ou remover classes.

Experimentos realizados, utilizando-se várias bases de dados de domínio público e usadas na literatura como *benchmark*, mostraram que, de forma geral, o modelo proposto obteve qualidade de resposta semelhante à das melhores técnicas avaliadas e tamanho de estrutura e tempo de classificação tão reduzidos quanto aos das técnicas de menor complexidade. Esses resultados indicam que o modelo proposto consegue obter um compromisso satisfatório entre eficiência e eficácia.

Abstract

Incremental learning techniques are those which learning takes place continuously over time and does not finish once available data have been exhausted. Such techniques have useful characteristics in situations where problem data are acquired in small quantities over time and when the characteristics of a task are also subject to change over time. However, a disadvantage of these techniques is that they tend to produce models with growing structure related to the amount of data used for training, and with time it may produce models with structure large enough to make unfeasible the application of these techniques to accomplish certain tasks. In this context, this PhD thesis proposes a model for incremental supervised learning for classification problems, which has a reduced structure while maintaining a quality comparable to the best incremental techniques.

The model proposed in this paper, called evolving Probabilistic Neural Network (ePNN), is based on Probabilistic Neural Network, the Gaussian Mixture Model and Expectation Maximization algorithm. The main characteristics of the proposed model are the ability to learn continually throughout its existence and the possibility to require that each sample is used only once in the training phase without reprocessing. It is added to the two characteristics mentioned above the fact that the proposed ePNN has the ability to dynamically change its architecture to adapt better to a particular type of problem, and it also has a flexible structure, which facilitates the task to insert or remove classes.

Experiments conducted using various benchmark data sets available in the public domain showed that, overall, the proposed model obtained a similar quality response compared with the best evaluated techniques and structure size and classification time as small as the less complex techniques. These results indicate that the proposed model achieves a satisfactory compromise between efficiency and efficacy.

Capítulo 1

Introdução

Em muitas tarefas do cotidiano, é necessário um ou mais especialistas para analisar um bloco de informações e tomar decisões a partir dos resultados obtidos. Contudo, existem tarefas cuja imensa quantidade de dados a ser estudada requer grande demanda de mão de obra, aumentando o custo do processo e o tempo de análise. Como consequência destes problemas, é cada vez mais frequente o uso de técnicas de aprendizado de máquina na esperança de automatizar alguns processos e torná-los menos demorados e onerosos. Técnicas de aprendizado de máquina permitem a um computador extrair regras e padrões de um conjunto de dados, transformando isto num sistema capaz de realizar tarefas sem a necessidade de, explicitamente, programá-lo para as mesmas.

Historicamente, tais técnicas têm focado sobre tarefas nas quais, frequentemente, está implícita a suposição de haver um conjunto de dados representando bem um problema[1] e a fase de treinamento cessa uma vez que esse conjunto tenha sido processado (Giraud-Carrier, 2000). Exemplos de tais técnicas são os clássicos *multi-layer perceptrons* (MLPs), *support vector machines* (SVMs) e redes neurais de função de base radial (Shiotani et al., 1995).

Como resultado, técnicas tradicionais de aprendizado e construção de modelos têm a vantagem de conhecer suas estruturas antecipadamente, porque estas podem ser otimizadas usando um conjunto de dados conhecidos. Portanto, essas técnicas podem apresentar um bom desempenho em ambientes estáveis, onde há pouca ou nenhuma mudança nas características. Entretanto, o desempenho delas pode cair bruscamente quando uma ou mais características do ambiente mudam. Além disso, precisam ser reprojetadas sempre que nova circunstância ocorrer durante o processo (Leite et al., 2009), tais como nova classe ou mu-

[1] Um problema de classificação é assumido neste trabalho como sendo um conjunto de amostras de uma determinada tarefa que são descritas como um vetor de características (atributos), e cada vetor é rotulado com uma classe (rótulo) de um conjunto definido de classes.

dança drástica em uma ou mais características do ambiente, podendo isto estar associado a um alto custo computacional.

Outra desvantagem é o desempenho de qualquer modelo estar diretamente relacionado à qualidade da base de dados disponível para treinamento. No entanto, o processo de aprendizado dos modelos clássicos considera todo o conjunto de dados já disponível durante a etapa de treinamento e, portanto, sem a possibilidade de acomodar novos conhecimentos quando novos dados são adquiridos. Contudo, a aquisição de dados suficientemente representativos do problema é onerosa, consome tempo e, além disso, nem sempre é possível obter os dados em tempo hábil, pois, normalmente, são disponibilizados em pequenas quantidades ao longo do tempo (Bhattacharyya et al., 2008). Então, apesar dessa abordagem de aprendizado ser aplicável para uma larga escala de problemas, ela não é completamente abrangente. Além disso, existem situações onde o aprendizado deve ocorrer ao longo do tempo de forma contínua, ao invés de uma única vez (Giraud-Carrier, 2000). Alguns exemplos são construções de modelos de perfis de usuários, robôs que necessitam aprender, de modo incremental, características do ambiente ao seu redor, e aplicações da área financeira.

Em tais situações, é desejável ter um sistema flexível e capaz de ser atualizado, sem afetar o desempenho do mesmo sobre os dados antigos. O processo de treinamento deve ser tal que ele aprenda novos conhecimentos sem o esquecimento do conhecimento adquirido. Tal sistema deve ter um compromisso entre duas propriedades (Polikar et al., 2001):

- **Estabilidade**: capacidade de reter o conhecimento sem acontecer um esquecimento catastrófico, porém não é garantido ser capaz de acomodar novo conhecimento;
- **Plasticidade**: capacidade de aprender continuamente novo conhecimento sem, no entanto, ter qualquer garantia de preservar o conhecimento anterior.

Um algoritmo de aprendizado incremental é o mais apropriado para acomodar ambos os conflitantes requisitos de estabilidade e plasticidade (Bhattacharyya et al., 2008), e é desejado ter as seguintes características (Leite et al., 2009; Bhattacharyya et al., 2008; Watts, 2009):

- aprender informação adicional usando novos dados;
- não requerer acesso aos dados antigos usados para treinar o sistema existente, ou seja, não requerer que toda a base de dados seja reapresentada para continuar com o processo de aprendizado e nem manter todos os dados embutidos em sua estrutura;
- não esquecer drasticamente o conhecimento adquirido anteriormente;

- ser capaz de acomodar novas classes que podem ser introduzidas com novos dados;
- não necessitar de conhecimento anterior sobre a estrutura do sistema (adaptação *on-line* da estrutura);
- não precisar de conhecimento prévio sobre as propriedades estatísticas dos dados (isto é, o modelo deve ser não paramétrico);
- não requerer inicialização de protótipos;
- ter habilidade para separar classes cuja superfície de separação é não linear.

Além disso, três suposições são frequentemente implícitas nas pesquisas de aprendizado incremental. Primeiro, o algoritmo deve ser capaz de usar o conhecimento adquirido para realizar uma tarefa em qualquer estágio do aprendizado. Segundo, a incorporação de conhecimento durante o aprendizado deve ser computacionalmente eficiente. Finalmente, o processo de aprendizado não deve fazer uma demanda de espaço exorbitante, tal que os requisitos de memória aumentem em função da quantidade de amostras de treinamento (Langley, 1995). Partindo dessas suposições, Langley (1995) define aprendizado incremental como sendo aquele em que é apresentada somente uma amostra de treinamento por vez, não reprocessa qualquer amostra usada anteriormente e retém somente uma estrutura de conhecimento, tal que não é possível retornar diretamente a estados anteriores de conhecimento, pois estes não estão retidos na memória. Com essa definição, técnicas como MLP, no qual as amostras são reprocessadas várias vezes, e *k Nearest Neighbor* (kNN), que armazena todas as estruturas de conhecimento na memória, não podem ser consideradas como técnicas de aprendizado incremental.

A dificuldade do aprendizado incremental é o aprendizado não ser polarizado *a priori* por qualquer estrutura ou conhecimento estatístico sobre os dados que virão no futuro. Em ambientes não estacionários, o desafio pode ser crucial, pois o sistema pode mudar drasticamente ao longo do tempo devido a mudanças que podem induzir alterações nas características de um problema (Elwell e Polikar, 2011; Widmer e Kubat, 1996).

Na literatura foram feitas algumas propostas de redes neurais com aprendizado incremental. Uma breve revisão dessas técnicas é realizada na seção seguinte.

1.1 Breve Revisão de Literatura

Diversas abordagens neurais foram propostas buscando atender um ou mais requisitos mencionados anteriormente. Muitos dos métodos propostos são baseados na adição de novos neurônios, e, ou, ajuste de pesos nas conexões dos neurônios existentes quando novos

dados de treinamento são adicionados. Entretanto, algumas dessas abordagens não possuem controle eficiente do tamanho de suas arquiteturas, podendo permitir a existência de neurônios similares ou redundantes dentro delas. Consequentemente, o número de neurônios tende a se elevar com o aumento na quantidade de dados de treinamento.

Algumas técnicas na literatura adicionam novos neurônios somente quando certas condições são satisfeitas (Shiotani et al., 1995; Gomm e Williams, 1995; Coghill et al., 2003; Heinen e Engel, 2010). No entanto, essas técnicas não possuem qualquer procedimento para eliminar redundâncias ou reduzir suas arquiteturas. Outros métodos, tais como o proposto em (Bhattacharyya et al., 2008), são mais radicais: além de não ter procedimentos para reduzir o tamanho da arquitetura, o método adiciona um novo neurônio para cada amostra de treinamento. Uma arquitetura crescente tem as desvantagens de necessitar grande demanda de memória e custo computacional cada vez mais elevado. De fato, é importante manter um compromisso entre número de neurônios, custo computacional e armazenamento de pesos das redes neurais. Uma rede neural com habilidade de aprendizado incremental não é muito útil se sua arquitetura for muito grande, pois o tempo para processar um novo dado pode ser proibitivo, principalmente para aplicações em tempo real.

Algumas abordagens foram avaliadas apenas sobre bases de dados com pouca quantidade de atributos (menos que 10) (Vlassis et al., 1999; Heinen e Engel, 2010), deixando em dúvida o desempenho e o custo computacional dessas técnicas para bases de dados com dimensões elevadas (mais de 100 atributos). O tamanho da arquitetura da rede neural apresentada em (Vlassis et al., 1999) pode aumentar repentinamente, porque cada neurônio pode ser dividido em muitos outros neurônios durante uma única iteração de treinamento, onde o número de divisões é proporcional ao número de atributos. Os neurônios da rede neural proposta por Heinen e Engel (2010) usam uma função de transferência Gaussiana com matriz de variância/covariância completa. No entanto, para uma grande quantidade de atributos, essas matrizes se tornam imensas, aumentando o custo computacional para calcular os determinantes e as inversas dessas matrizes. Além disso, podem ocorrer problemas de falta de memória, caso a arquitetura seja grande e seja implementada em ambiente com restrições de memória. Em aplicações do mundo real, há muitos problemas envolvendo dados com elevada quantidade de atributos, tais como processamento de imagens e textos. Portanto, é necessário desenvolver algoritmos melhores para tratar esses tipos de problemas.

Ainda neste contexto, existem modelos com capacidade de aprendizado que se deteriora com o tempo, tendendo a zero. Estudos anteriores têm proposto exemplos dessas abordagens (Heinen e Engel, 2010; Bhattacharyya et al., 2008), em que cada amostra de treinamento tem a mesma contribuição, e assim a influência de cada novo dado de treinamento é reduzida. A habilidade para adicionar novos neurônios na rede ao longo do tempo pode também ser reduzida a zero (Platt, 1991), e, assim, a habilidade para aprender novas informações é degradada.

Tais modelos podem funcionar bem para casos nos quais o ambiente é estacionário. Por outro lado, eles podem não ser muito úteis em aplicações cujo ambiente não seja estacionário, isto é, onde as características mudam ao longo do tempo.

Finalmente, o procedimento de aprendizado de algumas redes neurais pode requerer que cada amostra de treinamento seja processada várias vezes (Shiotani et al., 1995; Coghill et al., 2003; Alpaydin, 1991), em contraste com técnicas que precisam processar os dados somente uma vez. A estrutura inteira de alguns modelos também pode necessitar ser alterada para adicionar novas informações. Esse tipo de procedimento pode tornar o processo de aprendizado lento.

Recentemente, uma família de técnicas de aprendizado incremental tem atraído a atenção de pesquisadores, sendo aplicada em diversas áreas. Essas técnicas são baseadas na construção de modelos através de algoritmos incrementais e fluxo de dados (Watts, 2009). Esses modelos, ditos evolutivos, não somente podem minimizar o problema do armazenamento de grandes quantidades de dados, mas também oferecem características importantes para a modelagem de processos não lineares adaptativos. As principais características dos modelos evolutivos são o aprendizado contínuo, a auto-organização e a adaptação a ambientes desconhecidos. Assim, um processo evolutivo pode ser definido como um processo que se desenvolve e se modifica de forma contínua ao longo do tempo (Leite et al., 2009). Mais detalhes sobre esses modelos são mencionados na próxima seção.

1.2 Sistemas Inteligentes Evolutivos (SIE) e Sistemas Conexionistas Evolutivos (SCE)

Em um sentido geral, sistemas de informação devem ajudar a entender a dinâmica dos processos modelados, evoluir regras automaticamente, obter “conhecimento” que capture a essência desses processos e aperfeiçoe o desempenho durante todo o tempo. Esses requisitos definem um subconjunto da Inteligência Artificial (IA) chamado de Sistemas Inteligentes Evolutivos (SIE) (Kasabov, 2007).

O termo “evolutivo” leva a pensar em algoritmos evolucionários, tais como Algoritmo Genético ou Estratégias Evolucionárias, e todos os seus mecanismos de evolução, como seleção baseada em desempenho, reprodução e mutação. No entanto, “evolutivo”, aqui, tem sentido mais amplo, para mudanças ao longo do tempo e, embora possa ser incorporado aos algoritmos evolucionários, isto não é necessariamente um procedimento obrigatório (Watts, 2009).

Um SIE é um sistema de informação capaz de realizar tarefas inteligentes típicas de seres humanos (como reconhecimento de padrões, aprendizado de linguagem e controle inteligente). Tal sistema deve desenvolver sua própria estrutura, funcionalidade e conhecimento, de forma contínua, auto-organizada, adaptativa e de maneira interativa com as informações de entrada, assim melhorando seu desempenho (Kasabov, 2007).

Uma subdivisão do campo de SIE são os Sistemas Conexionistas Evolutivos (SCE). Um SCE (em inglês *Evolving Connectionist System* - ECoS) é um sistema de representação de conhecimento adaptativo com aprendizado incremental que evolui a sua estrutura e funcionalidade, onde, no núcleo do sistema, está uma arquitetura conexionista que consiste de neurônios artificiais e conexões entre eles.

Em outras palavras, um SCE é uma rede neural ou um conjunto de redes neurais operando continuamente no tempo e adaptando sua estrutura e funcionalidade através de interações contínuas com o ambiente e com outros sistemas (Leite et al., 2009; Kasabov, 2007; Watts, 2009). Esse termo foi cunhado por Kasabov que, em 1998, propôs o primeiro algoritmo de rede neural dessa família (Kasabov, 1998).

Descrevendo mais formalmente, os SCEs são redes neurais com múltiplas camadas de neurônios, tendo ao menos uma camada, chamada de camada evolutiva, que é a camada construtiva, ou seja, aquela a crescer ou diminuir de forma a se adaptar aos dados de entrada, e na qual o algoritmo de aprendizado irá se concentrar. O algoritmo de aprendizado dos SCEs é baseado na acomodação dos novos exemplos de treinamento dentro da camada evolutiva, através da modificação dos valores dos pesos de conexões dos neurônios dessa camada, ou na adição de novos neurônios nessa camada (Watts, 2009).

Vários métodos de SCE podem ser utilizados em aplicações nas quais técnicas tradicionais de inteligência computacional costumam ser usadas. Além disso, eles têm características que os fazem aplicáveis em alguns problemas mais específicos (Kasabov, 2007), tais como, aprendizado incremental, que pode continuar por toda a sua existência, e aprendizado construtivo (isto é, suas arquiteturas podem se alterar para melhor adaptação à tarefa). Os SCEs também dividem o problema em espaços locais, tornando a adaptação mais rápida, não sendo necessário mudar toda a arquitetura para aquisição de novo conhecimento.

Basicamente, o processo de aprendizado de um SIE e, em especial, de um SCE, consiste nas seguintes etapas: aquisição de dados, pré-processamento e avaliação das características, modelagem das conexões e aquisição de conhecimento.

As vantagens dos SCE são possuir arquitetura flexível, diferentemente de técnicas tradicionais (como MLP e SVM), e aprender rapidamente, pois os algoritmos de aprendizado necessitam somente uma única apresentação da base de dados e, por isso, são mais difíceis

de apresentarem *overtraining* (Watts, 2009). Eles podem ser mais resistentes a esquecimento catastrófico (isto é, esquecer informações antigas) em relação a alguns modelos, pois seus dados de treinamento podem ser representados através da adição de novos neurônios, mudando a arquitetura da rede, ao invés da acomodação dos dados adicionais nos neurônios existentes. Eles também podem aprender continuamente, com treinamento que não é restrito a um único conjunto de dados (Watts, 2009).

No entanto, algumas dificuldades enfrentadas por esses sistemas são alguns de seus parâmetros não serem conhecidos previamente, e distúrbios ou mudanças inesperadas que podem acontecer durante o período de aprendizado. Outro problema é que o aprendizado dessas redes, que utilizam um exemplo de treinamento por vez, é afetado pela ordem em que os exemplos são apresentados (Watts, 2009).

1.3 Proposta desta Tese

Muitas das características almejadas para os SCEs são interessantes para tratar tarefas do mundo real. No entanto, tem sido observado na prática que, na tentativa de reter o máximo de informação possível, as redes dessa família tendem a possuir tamanho grande de arquitetura, e isto reduz a velocidade de classificação na parte de teste e aumenta o consumo de memória (Watts, 2009). Isso é um efeito indesejável para uma técnica de aprendizado de máquina.

Sendo assim, essa tese tem como hipótese o desenvolvimento de um modelo de aprendizado incremental com características semelhantes aos SCEs, mas com uma estrutura mais reduzida, necessitando então de menos recursos computacionais e capaz de obter uma relação satisfatória entre eficiência e eficácia. Satisfatória no sentido de possuir eficácia num nível semelhante aos de técnicas mais complexas, mas tão eficiente quanto às das técnicas que possuem pequenas estruturas.

Logo, o objetivo dessa tese é propor mecanismos para obter, de forma incremental, um modelo supervisionado para problemas de classificação. Esse modelo deverá, portanto, incorporar várias das características dos SCEs, mas necessitando menos recursos computacionais que outros modelos da mesma família dos SCEs. Para alcançar tal modelo proposto, essa tese parte do princípio de, para obter uma estrutura reduzida, ser mais conveniente modelar a distribuição da fonte de dados (Vlassis e Likas, 1999; Yang et al., 2012) do que simplesmente estocar amostras de um determinado problema na estrutura (Bhatia e Vandana, 2010; Specht, 1990; Ludermir et al., 1999).

Além disso, esse trabalho se baseia na suposição de muitas distribuições na natureza possuírem uma forma aproximadamente semelhante a uma distribuição Gaussiana (Duda et al.,

2001). Logo, tais distribuições poderiam ser representadas por uma distribuição Gaussiana sem perda significante de informação. Ainda assim, mesmo se a distribuição das amostras de uma classe de um problema supervisionado não se assemelhar a uma Gaussiana, a separação de classes pode ser realizada sem a necessidade de modelar por completo a sua distribuição, mas somente certas regiões de borda das classes.

Com isso, é possível obter um modelo de tamanho compacto apresentando desempenho satisfatório. Um tamanho reduzido é possível, desde que o contorno de separação entre classes não seja geometricamente ou topologicamente muito complexo, o qual necessitaria de uma longa descrição dos contornos, possivelmente incluindo todos os dados de treinamento de cada classe (Ho, 2000).

Contudo, algumas distribuições de dados podem ser mais complexas e não serem representadas devidamente por uma simples Gaussiana. Nesses casos, tais distribuições podem ser modeladas através não de uma, mas com um conjunto de distribuições Gaussianas, conhecido na literatura como Modelo de Mistura de Gaussianas (MMG) (Bishop, 2006). Com um número adequado de Gaussianas, o MMG pode modelar diferentes tipos de distribuições de dados (Bishop, 2006).

Todavia, o MMG apresenta um grande problema, que é a determinação do número de Gaussianas para representar cada distribuição, e os métodos mais clássicos determinam uma quantidade fixa que não se altera ao longo do refinamento do modelo (McLachlan, 1987; Furman e Lindsay, 1994). Outra dificuldade existente diz respeito ao aprendizado incremental, pois os parâmetros de um MMG são inicialmente imprecisos quando são estimados de forma incremental, e essa imprecisão prejudica a qualidade do modelo obtido.

Para tratar esses problemas, o modelo proposto nesse trabalho possui mecanismos para determinar dinamicamente a quantidade necessária de componentes do MMG para uma determinada tarefa e, também, uma forma de reduzir os efeitos negativos das imprecisões dos parâmetros do MMG causados pelo aprendizado incremental.

Este modelo se chama Rede Neural Probabilística evolutiva (RNPe), e o seu nome é em homenagem à Rede Neural Probabilística (RNP) (Specht, 1990), cuja arquitetura e função de transferência dos neurônios são semelhantes à da rede neural proposta. Algumas características da RNPe são semelhantes aos dos SCEs, tais como novas informações são adicionadas ao modelo de forma incremental através de uma amostra por vez, ao invés de ser por bloco de dados. O modelo pode aprender continuamente durante toda a sua existência, pois sua habilidade para aprender novas informações não tende a ser reduzida a zero. O modelo tem capacidade de alterar sua estrutura para se adaptar melhor a um determinado tipo de problema, e sua estrutura é flexível, sendo fácil inserir ou remover classes.

Assim, em contrapartida aos métodos mostrados na revisão de literatura (Seção 1.1), o sistema proposto apresenta as características de possuir controle eficiente do tamanho da arquitetura, permitindo uma dimensão reduzida e com baixo custo computacional; resultado satisfatório sobre bases de dados com grande quantidade de atributos; processo de aprendizado não se deteriorando em demasia com o tempo; capacidade de aprender com somente uma única apresentação dos dados; e, para aprender novas informações, não é necessário alterar toda a sua arquitetura.

Resultados experimentais indicaram que a RNPe obtém desempenho semelhante a algumas redes neurais incrementais e alguns modelos clássicos de aprendizado de máquina, com a vantagem de possuir uma estrutura mais reduzida e, assim, necessitando de menos recursos computacionais.

Além da RNPe, outras contribuições deste trabalho são:

- comparação da RNPe com outros algoritmos do ponto de vista de complexidade e desempenho de classificação, usando várias bases de dados de *benchmark* de domínio público e de diferentes naturezas (Ciarelli et al., 2012; Oliveira et al., 2012);
- avaliação empírica da capacidade do modelo proposto de tratar o dilema da plasticidade e estabilidade (Ciarelli et al., 2012; Ciarelli et al., 2013);
- uma metodologia e métricas para medir o grau de estabilidade e plasticidade de um algoritmo com aprendizado incremental (Ciarelli et al., 2013);
- análise sobre quais características das bases de dados facilitam ou dificultam a tarefa de aprendizado incremental (Ciarelli et al., 2013);
- análise do uso de comitês de RNPes para o problema de classificação e seu uso em tarefas de aprendizado semi-supervisionado.

1.4 Estrutura do Trabalho

Este trabalho está organizado na seguinte estrutura:

- no Capítulo 2 são apresentadas algumas informações sobre processos evolutivos e abordagens de aprendizado semi-supervisionado;
- a descrição de algumas redes neurais incrementais e suas aplicações na literatura é realizada no Capítulo 3;

- a técnica proposta e sua comparação com outras técnicas são apresentadas no Capítulo 4;
- no Capítulo 5 são descritas algumas características das bases de dados e a metodologia usada no trabalho;
- experimentos, resultados e discussões são apresentados no Capítulo 6;
- por fim, no Capítulo 7, a conclusão, indicando os caminhos futuros dessa pesquisa.

Capítulo 2

Processos Evolutivos e Aprendizado Semi-Supervisionado

Este capítulo tem como foco os processos evolutivos e algumas características de aprendizado semi-supervisionado. Enquanto a primeira seção apresenta uma breve caracterização dos processos evolutivos e sua dinâmica, a segunda seção deste capítulo indica algumas informações sobre aprendizado semi-supervisionado e alguns trabalhos na área.

2.1 Processos Evolutivos

A busca por sistemas que consigam modelar perfeitamente ou, pelo menos, de uma forma adequada, a dinâmica dos processos que permeiam a existência humana tem motivado a realização de diversos trabalhos na literatura. Quando são conhecidas muitas informações a respeito de tais processos, podem-se criar modelos artificiais que se comportam de forma similar à natureza de tais processos e, assim, conseguir fazer predições corretas e até mesmo entender melhor os processos reais. No entanto, nem sempre se possui conhecimentos suficientes desses processos, e as construções dos modelos devem ser feitas através de dados coletados dos mesmos. Entretanto, em muitas situações do mundo real, a aquisição de dados pode ser difícil e consumir tempo, além de normalmente serem disponíveis em pequenas quantidades ao longo do tempo. Nesses casos, um sistema que possa ser modelado com os dados inicialmente disponíveis, e que possui um procedimento simples e rápido para adicionar novas informações, à medida que são coletadas, torna-se interessante.

Segundo o dicionário Houaiss da língua portuguesa (Hoauiss, 2010), evolução significa "processo de desenvolvimento ... em que toda a natureza, com seus seres vivos ou inanima-

dos, se aperfeiçoa progressivamente, realizando novas capacidades, manifestações e potencialidades", além disso, é um "processo gradativo, progressivo de transformação, de mudança de estado ou condição; progresso". No sentido usado neste trabalho, pode-se definir um processo evolutivo como um processo que está em constante desenvolvimento, no sentido de gradualmente se aperfeiçoar, mudando ao longo do tempo conforme haja tal necessidade, podendo, até mesmo, permanecer em um estado de contínuo aprendizado ao longo de toda a vida. Neste contexto, o significado de "evolutivo" está num sentido mais amplo do que o termo "evolucionário", que é comumente definido em métodos heurísticos, tais como em algoritmos evolucionários.

Processos evolutivos são difíceis de modelar porque nem todas as informações necessárias são conhecidas inicialmente, e elas podem mudar dinamicamente devido a perturbações inesperadas. Portanto, modelar tais processos é uma tarefa desafiadora. Modelar tais processos é uma tarefa valiosa para muitas aplicações práticas, pois há um grande número de tarefas nas ciências, engenharias e até mesmo nas tarefas do dia-a-dia que estão em constante alteração ao longo do tempo, tais como a dinâmica de bolsa de valores, sistemas de reconhecimento de fala, de face, de outras evidências biométricas, a mudança de organização em tabelas de classificações, tais como a Classificação Nacional de Atividades Econômicas (CNAE) (CNAE, 2012),a de produtos, e a Classificação Brasileira de Ocupações (CBO) (CBO, 2007), entre outros (Kasabov, 2007).

A própria vida pode ser considerada como um processo evolutivo, sendo esse um tempo no qual os organismos existentes evoluem do nascimento até a morte (Hoauiss, 2010). Mudanças contínuas, que ocorrem com certa estabilidade, caracterizam a vida (Kasabov, 2007). Outro exemplo é o sistema imunológico que precisa estar em constante adaptação para defender o organismo de ataques constantes de micro-organismos.

Os processos evolutivos podem apresentar dinâmicas que apresentam diferentes comportamentos (Kasabov, 2007) (a dinâmica de alguns processos evolutivos é semelhante à dos processos estocásticos, pois ambas modificam-se com o tempo):

- Aleatório: não há regra que governa o processo no tempo e o processo não é previsível;
- Caótico: o processo é previsível, mas somente em um curto período de tempo a frente. Isto acontece, pois o processo no momento presente depende do processo em um tempo anterior através de uma função não linear;
- Quase periódico: o processo é previsível, mas sujeito a erro. As regras que o governam modificam-se ligeiramente ao longo do tempo;
- Periódico: o processo repete os mesmos padrões de comportamento ao longo do tempo e é completamente previsível.

Muitos processos complexos em engenharia, ciências sociais, física, matemática, economia, e outras ciências estão evoluindo por natureza. Algumas séries temporais manifestam comportamento caótico, isto é, há padrões vagos de repetição ao longo do tempo, e tais séries temporais podem ser previsíveis aproximadamente somente em um futuro próximo, mas não em um futuro distante (Gleick, 1987; Barndorff-Nielsen et al., 1993). Processos caóticos costumam ser descritos por equações matemáticas que utilizam estados passados para gerar o próximo estado. Fórmulas simples podem descrever comportamentos muito complicados no tempo. Exemplo disso é a Equação 2.1[1], que descreve a população crescente de peixes F_{t+1} baseada na população corrente F_t e um parâmetro g. Para $g > 0,89$ a função se torna caótica (Gleick, 1987).

$$F_{t+1} = 4gF_t(1 - F_t). \tag{2.1}$$

Um processo caótico é definido por regras evolutivas que se encontram entre a desordem dos processos aleatórios e nos quase previsíveis processos quase periódicos. A modelagem de processos caóticos, especialmente se as regras do processo mudam ao longo do tempo, é uma tarefa para um sistema adaptativo que captura as mudanças no processo ao longo do tempo. De uma forma geral, todos os problemas da engenharia, economia, e ciências sociais que são caracterizados por processos evolutivos requerem modelos que estão em contínua adaptação (Kasabov, 2007). Um algoritmo que possua a capacidade de aprendizado incremental é o mais apropriado para realizar tal tarefa, desde que ele tenha a capacidade de tratar o dilema de reter nova informação sem haver esquecimento catastrófico (Bhattacharyya et al., 2008).

Um sistema com aprendizado incremental pode ter a capacidade de aprendizado por toda a sua existência através de um fluxo contínuo de iterações com o ambiente. Além disso, suas características de rápido treinamento e estrutura flexível podem ser desejadas em diversas aplicações, como por exemplo, aplicações em tempo real. Na Figura 2.1, é mostrado um fluxo contínuo de dados entre o ambiente e um sistema de aprendizado, quando este está se adaptando aos dados obtidos do ambiente. Este sistema é composto pelo sistema com aprendizado incremental, além de outros processos, como o de pré-processamento. O sistema de aprendizado se adapta à medida que novos dados são disponibilizados na entrada e analisados pelo sistema. A interferência humana está presente neste ciclo pela necessidade de: rotulação dos dados usados para treinamento, modelagem do sistema de aprendizado (pelo menos inicialmente), verificação se o sistema está funcionando com desempenho satisfatório e por outros motivos que dependem de caso para caso. Entretanto, dependendo do caso, a interferência humana pode acarretar num processo lento e custoso, além de comprometer

[1]Para simplificação das próximas equações, os índices temporais $t+1$ e t serão considerados implícitos nas equações, sendo indicados somente quando houver necessidade.

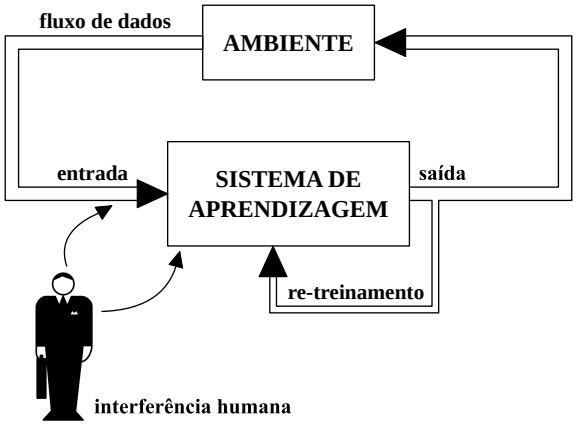


Figura 2.1: Interação entre o ambiente e um sistema de aprendizagem na tarefa de (re)treinamento do mesmo. Os dados são obtidos do ambiente, analisados e/ou utilizados para treinamento do sistema, e uma saída é retornada para o ambiente. A interferência humana existe devido à necessidade de adicionar rótulos para os dados usados como treinamento, entre outras possíveis razões.

as vantagens de tais sistemas, tais como poder aprender através de um fluxo ininterrupto de dados, e aprendizado em tempo real.

Um método possível de contornar esta deficiência é usar métodos de aprendizado semi-supervisionado. Nestes métodos é utilizado um pequeno conjunto de dados rotulados para treinamento do sistema e, através da informação previamente obtida, o próprio sistema se encarrega de rotular os dados e usá-los para treinamento, eliminando (ou reduzindo) a interferência humana no ciclo de aprendizado de máquina ilustrado na Figura 2.1.

O aprendizado semi-supervisionado, suas características e os seus métodos mais comuns são discutidos na próxima seção.

2.2 Aprendizado Semi-Supervisionado

As técnicas clássicas de aprendizado de máquina costumam usar somente um conjunto de dados já previamente rotulados para o treinamento. No entanto, amostras rotuladas são frequentemente difíceis de serem obtidas, sua aquisição é cara, sua obtenção consome tempo e requer esforço da experiência humana para identificá-las. Ainda assim, mesmo obtendo uma base de dados devidamente rotulada, é difícil obter uma que seja altamente representativa do modelo que a gerou. Por outro lado, dados não rotulados podem ser relativamente fáceis de serem coletados, além de serem abundantes, embora tenham sido elaboradas poucas alternativas para poder aproveitá-los. O aprendizado semi-supervisionado trata este problema

usando um grande número de amostras não rotuladas em conjunto com as amostras rotuladas, e assim obtém um modelo melhor da fonte que originou tais amostras. Devido ao aprendizado semi-supervisionado requerer um menor esforço humano e produzir um desempenho aceitável, ele é de grande interesse, ambos na teoria e na prática (Zhu, 2008).

Dados não rotulados sozinhos são geralmente insuficientes para alcançar melhor classificação do que um classificador aleatório, porque não há informação sobre o rótulo das classes (Castelli e Cover, 1995). Contudo, dados não rotulados contém informação sobre a distribuição conjunta das características no espaço. Por isso, eles podem às vezes ser usados, juntos com dados rotulados, para aumentar a precisão de classificação em certos problemas (Nigam et al., 1999).

Para um melhor entendimento desse processo, será enunciado um exemplo dado por Nigam et al. (1999). Considere um simples problema de classificação, no qual as amostras são geradas usando um modelo de mistura de Gaussianas, onde os dados são gerados a partir de duas distribuições Gaussianas, uma por classe, cujos parâmetros são desconhecidos. A Figura 2.2 ilustra o contorno de decisão ótima de Bayes representada pela reta que passa por d, que classifica amostras dentro de duas classes (cada uma modelada por uma Gaussiana), identificadas pelas áreas claras e sombreadas. Agora considere que seja disponível uma quantidade infinita de dados não rotulados. É conhecido que dados não rotulados sozinhos, quando gerados de uma mistura de duas Gaussianas, são suficientes para recuperar os componentes originais da mistura, isto é, identificar corretamente os seus parâmetros: médias, variâncias e parâmetro de mistura (McLachlan e Krishnan, 1997). No entanto, embora seja possível obter corretamente a distribuição das amostras, é impossível associar rótulos das classes a cada uma das Gaussianas sem qualquer dado rotulado. Desta forma, dados rotulados devem ser usados para determinar a classe associada a cada uma das Gaussianas. Tal problema é conhecido por convergir exponencialmente rápido com o número de amostras rotuladas (Castelli e Cover, 1995). Com isto em mente, se houver uma quantidade de dados rotulados para ao menos determinar a classe de cada componente, os parâmetros de cada componente do modelo podem ser estimados com o uso dos dados não rotulados.

Entretanto, não é garantido obter melhores resultados com a abordagem de aprendizado semi-supervisionado. Embora um esforço menor seja utilizado para rotular os dados de treinamento, é necessário despender de uma quantidade de esforço razoável para projetar bons modelos, características, *kernels* e funções de similaridade para aprendizado semi-supervisionado. Tal tarefa pode ser mais crítica de ser realizada do que para abordagem supervisionada, devido a falta de dados de treinamento (Zhu, 2008). Uma má correspondência entre a estrutura do problema e o modelo suposto pode guiar para uma degradação no desempenho do classificador. De fato, vários pesquisadores têm observado que dados não rotulados nem sempre ajudam no aprendizado semi-supervisionado e que podem até reduzir

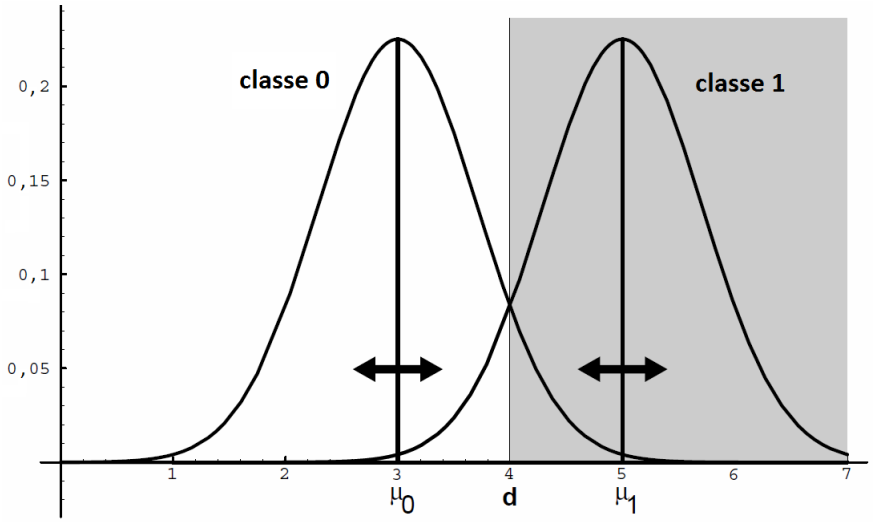


Figura 2.2: Classificação por uma mistura de Gaussianas (Nigam et al., 1999). Se uma quantidade ilimitada de dados não rotulados for disponível, então é possível recuperar os componentes da mistura original. No entanto, dados rotulados são necessários para identificar a qual classe cada componente está associada. A reta que passa por *d* representa o contorno de decisão ótima de Bayes entre as duas classes.

o desempenho sobre certas circunstâncias. No entanto, identificar uma má correspondência entre a estrutura do problema e o modelo suposto é difícil e continua sendo uma questão em aberto na literatura (Zhu, 2008).

2.2.1 Geração de Modelos

Geração de modelos é provavelmente o método de aprendizagem semi-supervisionado mais antigo (Zhu, 2008). Geração de modelos é um método de se conseguir realizar uma estimativa da fonte original geradora dos dados. Entretanto, muitas vezes a fonte que governa os dados pode apresentar uma complexidade elevada para ser modelada de forma direta, então se costuma representá-la por uma mistura de modelos mais simples de serem parametrizados e tratados.

Em misturas de modelos, tais como os obtidos com *Expectation Maximization*, uma grande quantidade de dados não rotulados pode ser usada para identificar a mistura de componentes, sendo que, idealmente, é necessário apenas um exemplo rotulado para cada componente para determinar a distribuição da mistura. Alguns pesquisadores têm mostrado ganho no desempenho utilizando o conjunto de dados não rotulados junto com os dados rotulados (Nigam et al., 1999; Baluja, 1998; Fujino et al., 2005; Wu et al., 2000). Porém, alguns pontos devem ser notados para se obter bom desempenho com esta abordagem (Zhu, 2008).

Identificabilidade do Modelo

O modelo de mistura deve ser idealmente identificável, isto é, para uma família de distribuições p_θ determinadas pelo conjunto de parâmetros θ, θ é identificável se para $\theta_i \neq \theta_j$ $\forall j \neq i$, $p_{\theta_i} \neq p_{\theta_j}$, desde que θ_j não seja uma permutação dos parâmetros de θ_i. Em outras palavras, um modelo é identificável se para diferentes conjuntos de parâmetros θ são obtidas distintas distribuições p_θ. Se um modelo é identificável, em teoria, com um conjunto infinito de dados não rotulados pode ser obtido o conjunto de parâmetros dos componentes da mistura.

Um exemplo de modelo não identificável é dado a seguir baseado em um exemplo apresentado em (Zhu, 2008). Nesse exemplo é utilizada a teoria de decisão Bayesiana, onde a probabilidade de x pertencer a uma classe y é obtida por $p(y|x) = p(x|y)p(y)/p(x)$, onde $p(x)$ e $p(y)$ são a probabilidade de ocorrer x e a classe y, respectivamente, e $p(x|y)$ é a probabilidade de ocorrer x dada a classe y. Então, imagine que o modelo $p(x|y)$ possui uma distribuição uniforme ($unif$) para $y \in \{+1,-1\}$, e com uma grande quantidade de dados não rotulados é observado que a distribuição de $p(x)$ é uniforme no intervalo $[0,1]$. Agora, imagine que entre os dados disponíveis existem apenas dois dados rotulados, o ponto 0,1 rotulado como $+1$ e o ponto 0,9 rotulado como -1. Dessa forma surge a seguinte pergunta: com esses dados é possível determinar o rótulo para $x = 0,5$? De fato a resposta é não, pois os modelos das distribuições de $p(x|y)$ não estão bem identificados, e eles podem assumir uma grande gama de valores. Por exemplo, na Equação 2.2 é apresentado um modelo no qual o ponto é rotulado como -1, já que $p(y = +1|x = 0,5) = 0$ e $p(y = -1|x = 0,5) = 1$. Por outro lado, o modelo apresentado na Equação 2.3 rotula o ponto como $+1$, uma vez que $p(y = +1|x = 0,5) = 1$ e $p(y = -1|x = 0,5) = 0$. A Figura 2.3 ilustra esses dois modelos.

$$p(y = +1) = 0,2;\ p(x|y = +1) = unif(0;0,2);\ p(x|y = -1) = unif(0,2;1) \tag{2.2}$$

$$p(y = +1) = 0,6;\ p(x|y = +1) = unif(0;0,6);\ p(x|y = -1) = unif(0,6;1) \tag{2.3}$$

Mesmo conhecendo que $p(x)$ é uma mistura de duas distribuições uniformes, não se pode identificar unicamente as duas componentes (pode haver mais de uma combinação de mistura das distribuições). A mistura multivariada de Bernoulli é um exemplo de mistura não identificável (McCallum e Nigam, 1998). Por outro lado, é conhecido que mistura de Gaussianas é identificável (Zhu, 2008).

Exatidão do Modelo

Se a suposição do modelo é correta, então é garantido que dados não rotulados aperfeiçoem o desempenho do mesmo (Castelli e Cover, 1995; Castelli e Cover, 1996; Ratsaby e

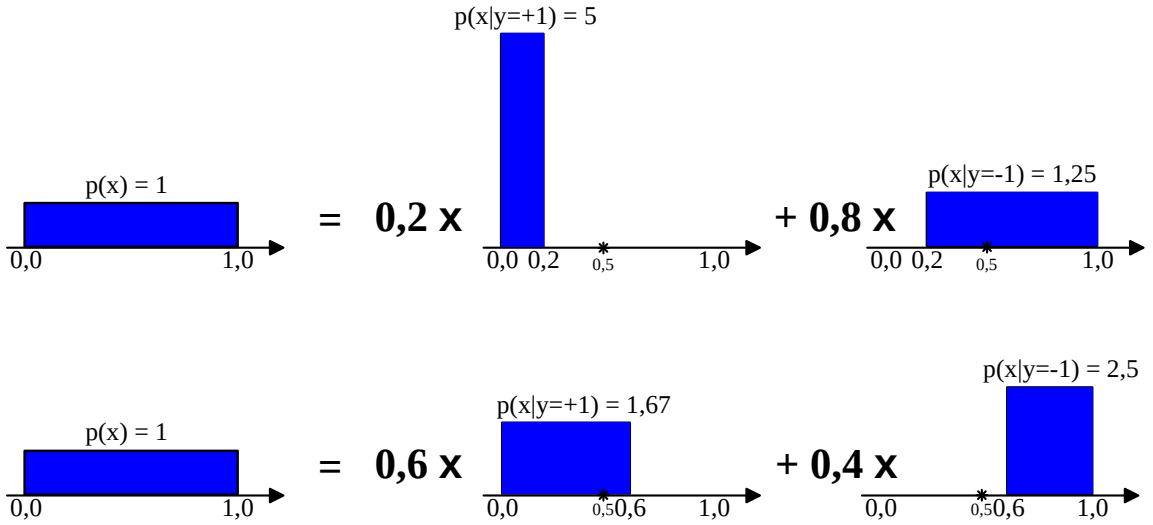


Figura 2.3: Exemplo de modelos não identificados. Nesse exemplo dois modelos são apresentados, onde o ponto $0,5$ pode ser identificado em diferentes componentes dependendo do modelo usado.

Venkatesh, 1995). No entanto, se o modelo estiver errado, dados não rotulados podem prejudicar o desempenho. Algumas abordagens, como as propostas em (Nigam et al., 1999; Corduneanu e Jaakkola, 2001) fornecem um peso menor aos dados não rotulados com o intuito de evitar uma grande interferência dos mesmos sobre um modelo já obtido.

Ótimo Local da *Expectation Maximization*

Mesmo se a suposição do modelo for correta, outro problema que pode ocorrer é a retenção do procedimento de otimização dos parâmetros em um ótimo local (Zhu, 2008). Na prática, modelos de misturas são identificadas pelo emprego de *Expectation Maximization*, que é um algoritmo propenso a estagnar em ótimos locais. Se o ótimo local estiver longe do ótimo global é possível que dados não rotulados prejudiquem o desempenho do aprendizado. Algumas abordagens buscam minimizar essa inconveniência aplicando algoritmos evolucionários ou técnicas para selecionar as melhores condições iniciais para o algoritmo, como realizado em (Nigam, 2001).

2.2.2 Métodos de Aprendizado Semi-supervisionado

Dois métodos são tipicamente usados no aprendizado semi-supervisionado: o auto-treinamento e o co-treinamento.

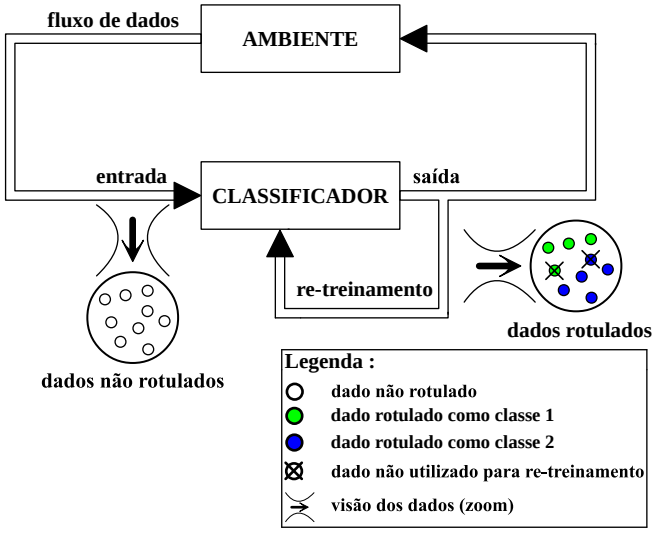


Figura 2.4: Procedimento de auto-treinamento do classificador. Um classificador recebe um conjunto de dados não rotulados e classifica-os. Os dados que receberam os rótulos com maior confiança são usados para o re-treinamento do classificador. Nos círculos são indicados os dados não rotulados, os rotulados e os não usados para re-treinamento.

Auto-treinamento

O auto-treinamento é uma técnica comumente usada para aprendizado semi-supervisionado. No auto-treinamento um classificador é inicialmente treinado com uma quantidade de dados rotulados, normalmente uma quantidade pequena, e então ele é usado para classificar dados não rotulados. Os dados não rotulados identificados com maior confiança pelo classificador, juntamente com possíveis dados rotulados obtidos em um momento posterior à modelagem do classificador, são tipicamente adicionados ao conjunto de treinamento nesse esquema de aprendizado. Dessa forma, o classificador é re-treinado e o processo torna a se repetir. Esse processo é ilustrado na Figura 2.4. Como pode ser notado, neste esquema de aprendizado o classificador usa as suas próprias predições para ensinar a si mesmo. Por essa razão, esse método é também chamado de auto-aprendizado. A aplicação de *Expectation Maximization* para a obtenção dos parâmetros de um modelo de mistura de distribuições pode ser visto como um caso especial de auto-treinamento suave, uma vez que ele escolhe automaticamente, a partir de suas predições, como cada componente deve ser ajustado (Zhu, 2008).

Com esta forma de aprendizado é esperado que um sistema com pouca informação inicial sobre um determinado problema possa obter de forma automática cada vez mais informação a partir dos dados não rotulados e de suas predições. No entanto, pode-se imaginar um efeito oposto: uma classificação errada pode ser realimentada provocando a deterioração do modelo. Para tentar evitar isso, alguns algoritmos se utilizam de alguns artefatos, como usar

somente as amostras identificadas com maior confiança para o treino, reduzir o peso das amostras não rotuladas para o treinamento e procedimentos de esquecimento de dados não rotulados caso a confiança na predição caia abaixo de um limiar.

Co-treinamento

Basicamente, co-treinamento é um procedimento no qual são utilizados 2 ou mais classificadores, que podem ou não ser da mesma natureza, para classificar um conjunto de dados não rotulados. Os dados considerados como corretamente classificados por um classificador são utilizados como treinamento para os demais classificadores e, também, podem ou não serem usados para treiná-lo.

Uma versão do co-treinamento proposta em (Blum e Mitchell, 1998) assume que: 1) as características que compõem um vetor de características podem ser divididas em dois conjuntos, 2) cada conjunto de sub-características é suficiente para treinar um bom classificador, 3) os dois conjuntos são condicionalmente independentes, dado a classe. Inicialmente, dois classificadores separados são treinados com os dados rotulados com os dois sub-conjuntos de características, respectivamente. Cada classificador então classifica os dados não rotulados e ensina o outro classificador com os dados (e suas classes preditas) que foram rotulados com maior confiança. Cada classificador é re-treinado com o conjunto adicional de dados que o outro classificador enviou, e o processo é repetido. Essa versão de co-treinamento é ilustrada na Figura 2.5.

Uma versão proposta em (Goldman e Zhou, 2000) não realiza a divisão do conjunto de características. No método proposto são usados dois classificadores de tipos diferentes nos quais é utilizado todo o conjunto de características para seus treinamentos. Os dados rotulados com maior confiança por um classificador, confiança essa medida através de um conjunto de testes estatísticos, são usados para o treinamento do outro classificador e vice-versa.

Um conjunto de classificadores treinados separadamente com diferentes calibrações sobre o espaço completo de características foi proposto em (Zhou e Goldman, 2004). Esse conjunto de classificadores realiza a classificação dos dados não rotulados. Se a maioria dos classificadores concorda com confiança da classe associada a um dado não rotulado, então esse dado é usado como treinamento, e assim, todos os classificadores são re-treinados com o conjunto atualizado de treinamento. A predição final é feita através de uma ponderação dos votos dos classificadores. Uma versão ligeiramente modificada foi proposta em (Zhou e Li, 2005), na qual são propostos três classificadores. Se dois deles concordam com a classe associada a um dado não rotulado, então esse dado é usado para treinar o terceiro classificador.

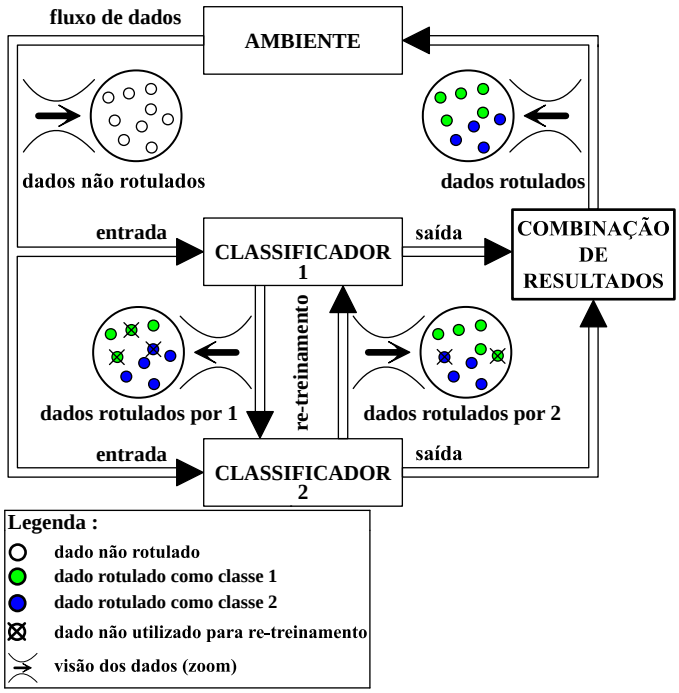


Figura 2.5: Procedimento de co-treinamento de dois classificadores. Dois classificadores são usados para rotular um conjunto de dados. O resultado final da classificação é obtido através de uma combinação dos resultados. Os dados classificados com maior confiança pelo classificador 1 são utilizados para re-treinar o classificador 2 e vice-versa. Nos círculos são indicados os dados não rotulados, os rotulados e os não usados para re-treinamento.

2.3 Conclusão

Um procedimento de aprendizado incremental torna mais prática a tarefa do aprendizado semi-supervisionado, e também mais rápida. Uma técnica pode complementar a outra, pois, embora o aprendizado incremental supervisionado possa aprender à medida que novos dados são obtidos, ele precisa da interferência humana para rotular os dados. Por outro lado, os métodos de aprendizado semi-supervisionado são capazes de rotular as amostras para depois usá-las para treinamento. No entanto, os métodos mais comuns são baseados nas técnicas clássicas de aprendizado. Tais técnicas sofrem da necessidade de um re-treinamento *off-line*, o qual, para muitas técnicas, é um processo lento e custoso, além de muitas vezes precisar de uma quantidade razoável de novos dados antes do re-treinamento. A combinação das duas técnicas permitiria um sistema com um aprendizado rápido, auto-adaptativo, capaz de fornecer rótulos às amostras, ao invés de somente agrupar os dados (como acontece nos métodos não supervisionados), com a vantagem adicional de não precisar da interferência humana para realizar a tarefa de rotulação das amostras para o re-treinamento.

Capítulo 3

Aprendizado Incremental em Redes Neurais

Neste capítulo são mencionadas algumas técnicas de aprendizado incremental, as quais utilizam de alguns dos conceitos estabelecidos para os SCEs. Em comum, esses métodos podem aprender com conhecimento inicial zero, isso é, com uma arquitetura vazia, sem qualquer conhecimento inicial sobre o conjunto de dados, incluindo o número de atributos e classes. Além disso, eles são baseados em acomodação de amostras, isto é, amostras são adicionadas em suas arquiteturas de acordo com um conjunto de critérios, ou os pesos dos neurônios são ajustados de forma a adquirir novas informações. Para finalizar, algumas aplicações e comparações presentes na literatura são mencionadas no final deste capítulo.

3.1 Rede Neural Probabilística Incremental (RNPI)

3.1.1 Algoritmo

A Rede Neural Probabilística Incremental (RNPI) é uma rede neural inspirada na flexível habilidade de se adicionar novos neurônios em uma estrutura já em funcionamento da Rede Neural Probabilística, apresentada no Apêndice C. A RNPI possui arquitetura semelhante à Rede Neural Probabilística, mas com uma pequena alteração no modo de treinamento.

Esta rede neural foi formalmente descrita em (Bhattacharyya et al., 2008). No entanto, ideias semelhantes foram apresentadas na literatura (Fan et al., 2004; Ciarelli et al., 2009b). A RNPI é composta por 4 camadas: a camada de entrada, a camada de padrões, a camada de soma e a camada de decisão. Nessa rede neural, as camadas modificáveis, com a chegada

Algoritmo *Algoritmo de aprendizado da Rede Neural Probabilística Incremental*

```
1 para cada novo dado de treinamento faça
2     se o dado é de uma classe que já existe na rede neural então
3         adicionar o dado para a camada de padrões desta classe
4     senão
5         criar um neurônio na camada de soma referente à classe do dado
6         adicionar o dado para a camada de padrões desta classe
```

Algoritmo 1: Algoritmo de treinamento da RNPI.

de novas amostras de treinamento, são a camada de soma e, principalmente, a camada de padrões. À medida que um novo dado de treinamento chega à rede neural, a rede passa pelo processo de aprendizagem. O procedimento de aprendizado apresentado em (Bhattacharyya et al., 2008) é descrito no Algoritmo 1.

3.1.2 Vantagens e Desvantagens

As principais vantagens da RNPI são a fácil implementação, o baixo custo computacional para o seu re-treinamento e a necessidade de calibrar somente um parâmetro, o sigma. No entanto, o valor de sigma afeta fortemente o desempenho da rede, semelhante ao que acontece na Rede Neural Probabilística. Além disso, esse valor não é ajustado durante o procedimento de aprendizado incremental. Uma abordagem para o uso desta rede neural é iniciá-la com uma arquitetura vazia, ou seja, sem nenhum neurônio na camada de padrões e na camada de classes, e, então, usar um valor de sigma que pode estar longe do valor ótimo. Outra abordagem, apresentada em (Bhattacharyya et al., 2008), é utilizar um conjunto de amostras em um procedimento de treino e validação da rede e, assim, determinar um valor ótimo ou sub-ótimo do sigma.

Outra desvantagem deste método é que ele não tem qualquer operação para controlar o tamanho da sua arquitetura, tal que os requisitos de memória e o seu custo computacional para classificar amostras aumentam com a chegada de novos dados de treinamento. Se a quantidade de dados for grande, então a quantidade de memória necessária e o custo computacional serão muito elevados.

3.2 Perceptron Multicamadas Evolutivo (eMLP)

3.2.1 Algoritmo

Outro método neural de aprendizado incremental é o Perceptron Multicamadas Evolutivo (em inglês: *evolving Multi Layer Perceptron* - eMLP), que foi originalmente proposto em (Kasabov, 2003). Uma rede eMLP consiste de três camadas de neurônios: a camada de entrada, uma camada chamada de camada evolutiva e uma camada de saída com uma função de ativação linear saturada. A camada evolutiva é a camada que irá crescer e se adaptar aos novos dados usados para treinamento, além de ser a camada na qual o algoritmo de treinamento mais se concentra (Kasabov, 2007). Para uma melhor compreensão da descrição do algoritmo é considerado que cada neurônio na camada evolutiva possui dois vetores de pesos: um vetor de pesos de entrada We e um vetor de pesos de saída Ws. O número de neurônios na camada de entrada é igual ao número de atributos do problema e o número de neurônios na camada de saída é igual ao número de classes. A Figura 3.1 ilustra a arquitetura de 3 camadas da rede eMLP, onde as linhas pontilhadas indicam quando um novo neurônio é adicionado na camada evolutiva.

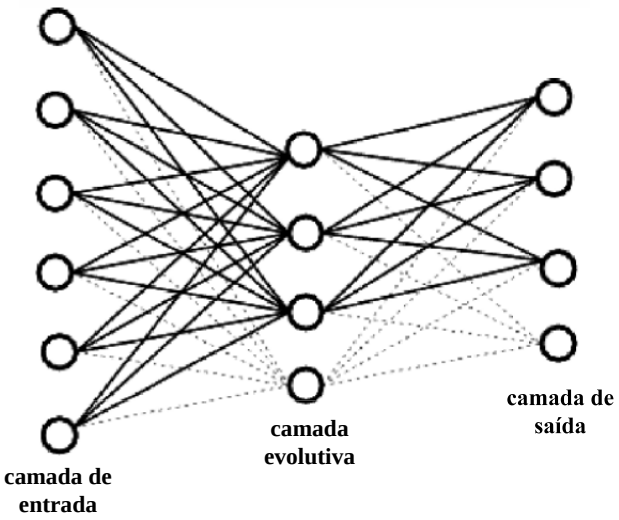


Figura 3.1: Arquitetura da Rede Neural eMLP (Kasabov, 2007).

A ativação A de um neurônio j da camada evolutiva é determinada por:

$$A_j = 1 - D_j, \tag{3.1}$$

onde A_j é a ativação do neurônio j e D_j é a distância normalizada entre o vetor de entrada e o vetor de pesos de entrada do neurônio j.

A medida de distância D_j é preferencialmente calculada com a distância de Hamming normalizada:

$$D_j = \frac{\sum_{i=1}^{d} |x_i - We_{j,i}|}{\sum_{i=1}^{d} |x_i + We_{j,i}|}, \tag{3.2}$$

onde x é o vetor de entrada, We_j é o vetor de pesos de entrada do neurônio j da camada evolutiva e d é a dimensão dos vetores x e We_j e também é o número de neurônios da camada de entrada. Além disso, $We_{j,i}$ é o peso da conexão entre a entrada i e o neurônio j e x_i é o i-ésimo componente do vetor de entrada x.

A saída Oc_p de um neurônio p da camada de saída é calculada pela Equação 3.3 (Watts, 2009):

$$Oc_p = \frac{\sum_{j=1}^{m} A_j Ws_{j,p}}{\sum_{j=1}^{m} A_j}, \tag{3.3}$$

onde Ws_j é o vetor de pesos de saída do neurônio j, p é o neurônio da camada de saída, A_j é a ativação do neurônio j da camada evolutiva e m é o número de neurônios na camada evolutiva. Além disso, $Ws_{j,p}$ é o peso da conexão entre o neurônio j e a saída p e Ws_j é um vetor com dimensão $|C|$, onde $|C|$ é o número de classes do problema de classificação.

A saída é calculada para todos os neurônios da camada de saída, e a amostra é classificada na classe associada ao neurônio da camada de saída com o maior valor de saída.

A forma mais usual do algoritmo de aprendizado é do tipo baseada na acomodação de novos exemplos de treinamento dentro da camada evolutiva: ou através de modificações dos pesos dos neurônios da camada evolutiva ou simplesmente adicionando um novo neurônio à rede. O algoritmo empregado para o treinamento é descrito no Algoritmo 2 (Kasabov, 2007).

Quando um neurônio é adicionado na camada evolutiva, o vetor de pesos das conexões de entrada é igual ao vetor de entrada x, e o vetor de pesos da saída é igual ao vetor de saída desejado Od associado à entrada x.

Os pesos de entrada do neurônio vencedor j da camada evolutiva são modificados de acordo com a Equação 3.4, para $i = 1, \ldots, d$:

$$We_{j,i} = We_{j,i} + \eta_1 \left(x_i - We_{j,i}\right), \tag{3.4}$$

onde η_1 é o primeiro parâmetro de taxa de aprendizado.

Algoritmo *Algoritmo de aprendizado do eMLP*

1 **para cada** novo dado de treinamento **faça**
2 Propagar o vetor de entrada através da rede neural
3 **se** a máxima ativação A_{max} de um neurônio é menor do que um limiar de sensibilidade S_{thr} **então**
4 adiciona um neurônio
5 **senão**
6 calcula o erro entre o vetor de saída calculado *Oc* e o vetor desejado de saída *Od*
7 **se** o erro for maior do que um limiar de erro E_{thr} ou a se saída do neurônio da classe desejada não for a mais altamente ativada **então**
8 adiciona um neurônio
9 **senão**
10 atualiza as conexões do neurônio vencedor na camada evolutiva

Algoritmo 2: Algoritmo de treinamento do eMLP.

Por outro lado, os pesos de saída do neurônio j são modificados de acordo com a Equação 3.5, para $j = 1, \ldots, m$:

$$Ws_{j,p} = Ws_{j,p} + \eta_2 A_j E_p, \tag{3.5}$$

onde η_2 é o segundo parâmetro de taxa de aprendizado. E_p é o erro entre a saída desejada e a saída obtida no neurônio p da camada de saída, dado por:

$$E_p = Od_p - Oc_p, \tag{3.6}$$

onde Od_p é a saída desejada no neurônio p e Oc_p é a saída calculada em p.

No entanto, com este procedimento de treinamento, a rede neural pode crescer muito com a chegada de novos dados para treinamento. Para controlar o tamanho da camada evolutiva durante o processo de aprendizado, e evitar um grande aumento no custo computacional com a chegada de novos dados de treinamento, é aplicada a operação de agregação de neurônios. O princípio da agregação é fundir neurônios que são espacialmente próximos entre si. Esse procedimento pode ser realizado a cada iteração ou depois de certo número de iterações de treinamento. O procedimento de agregação é descrito no Algoritmo 3.

O procedimento de agregação de neurônios é uma regularização importante por trazer melhorias no desempenho de uma rede eMLP em algumas aplicações, tais como sistemas de reconhecimento de fala e imagem.

Uma ligeira modificação do procedimento de treinamento da rede eMLP foi proposta em (Leite et al., 2009). Na modificação proposta, a linha 7 do procedimento de treinamento do Algoritmo 2 é alterada, considerando somente a condição que a saída do neurônio da classe desejada seja a mais altamente ativada, desprezando o valor do erro. Sendo assim, o parâmetro E_{thr} é desnecessário.

Algoritmo *Algoritmo de agregação do eMLP*

1 Encontrar o subconjunto R dos neurônios na camada evolutiva para o qual a distância Euclidiana normalizada $D(We_{rj}, We_{rp})$ e $D(Ws_{rj}, Ws_{rp})$ (Equação 3.7) são menores que um limiar W_{thr}, sendo $\{rj, rp\} \in R$.

$$D(We_{rj}, We_{rp}) = \sqrt{\frac{\sum_{i=1}^{d} \left(We_{rj,i} - We_{rp,i}\right)^2}{d}}. \tag{3.7}$$

2 Mesclar todos os neurônios do conjunto R dentro de um novo neurônio r, e calcular We_r e Ws_r usando as Equações 3.8 e 3.9, onde $|R|$ denota o número de neurônios no subconjunto R.

$$We_r = \frac{\sum_{k \in R} We_k}{|R|}. \tag{3.8}$$

$$Ws_r = \frac{\sum_{k \in R} Ws_k}{|R|}. \tag{3.9}$$

3 Excluir todos os neurônios do subconjunto R.

Algoritmo 3: Algoritmo de agregação do eMLP.

3.2.2 Vantagens e Desvantagens

As principais vantagens da rede eMLP são o ajuste dos seus parâmetros e o procedimento para controlar o número de neurônios na arquitetura. Porém, esse procedimento pode consumir muito tempo para ser usado a cada iteração, principalmente se a natureza do problema exige um número razoável de neurônios. Por outro lado, se o procedimento de agregação de neurônios não for usado por um longo tempo, o tamanho da rede pode crescer muito, aumentando a demanda de memória.

3.3 Rede Neural *Fuzzy* Evolutiva (EFuNN)

3.3.1 Algoritmo

Algumas redes neurais usam alguns princípios da lógica *fuzzy*, tais como conjuntos de regras e inferência *fuzzy*, em uma forma conexionista para armazenar conhecimento. Tais sistemas são baseados nas regras e nos mecanismos de inferência *fuzzy* para o propósito de aprendizado e otimização. A Rede Neural *Fuzzy* Evolutiva (em inglês: *Evolving Fuzzy Neural Network* - EFuNN) é um sistema baseado nesses mecanismos incorporado com a possibilidade de aprendizado incremental (Kasabov, 2007). Essa rede neural foi a primeira SCE descrita por Kasabov em (Kasabov, 1998) e, ao longo do tempo, algumas outras variações dela foram propostas (Kasabov, 2007).

A EFuNN possui uma arquitetura de cinco camadas de neurônios, onde os neurônios

e conexões são criados/conectados conforme os dados são usados para o treinamento. As camadas são respectivamente:

- camada de entrada: é a camada na qual o dado é apresentado à rede;
- camada de *fuzzification* ou quantização *fuzzy* dos dados de entrada: é a camada que realiza a *fuzzification* do dado de entrada. Essa camada não é completamente conectada à camada de entrada. Na verdade, cada neurônio na camada de entrada é conectado a um subconjunto de neurônios dessa camada, onde o número de neurônios nesse subconjunto é determinado pelo número de funções de pertinência (Watts, 2009). Por exemplo, dois neurônios para cada neurônio de entrada pode representar valores *PEQUENOS* e *GRANDES* da variável de entrada. Diferentes tipos de funções de pertinência podem ser usados (triangular, Gaussiana, etc). O número e o tipo de função de pertinência podem ser modificados dinamicamente. De forma geral, essa camada retorna o grau de pertinência de cada variável para cada uma das funções de pertinência através da *fuzzification* (mais detalhes ver Apêndice A);
- camada de regras: é a camada onde se encontram as regras *fuzzy* que evoluem através de aprendizado supervisionado ou não supervisionado. As regras dos neurônios representam protótipos (exemplos, centroides) dos dados de entrada-saída. Uma função de ativação linear ou Gaussiana é usada nos neurônios dessa camada. Cada neurônio j nessa camada é definido por dois vetores: We_j e Ws_j, onde We_j e Ws_j representam as coordenadas de um centro de esfera no espaço de entrada e saída *fuzzy*, respectivamente;
- camada de *fuzzification* dos dados de saída: essa camada realiza a *fuzzification* dos dados de saída (como as classes associadas a um dado, por exemplo), de forma similar à segunda camada. Aqui, uma soma ponderada das entradas e uma função linear saturada são usadas nos neurônios para calcular o grau de pertinência que um vetor de saída, que está associado a um dado de entrada, pertencer a cada uma das funções de pertinência de saída.
- camada de saída: essa camada fornece o valor de saída da rede. Uma função de ativação linear é usada para calcular os valores de *defuzzification* para os dados de saída.

A Figura 3.2 ilustra uma arquitetura da EFuNN em que as linhas pontilhadas (linhas mais claras) indicam a inclusão de um neurônio na camada de regras.

Além dessas camadas, pode ser adicionada uma camada opcional de memória de curto prazo através de uma conexão de retroalimentação com a camada de regras e, por essa razão,

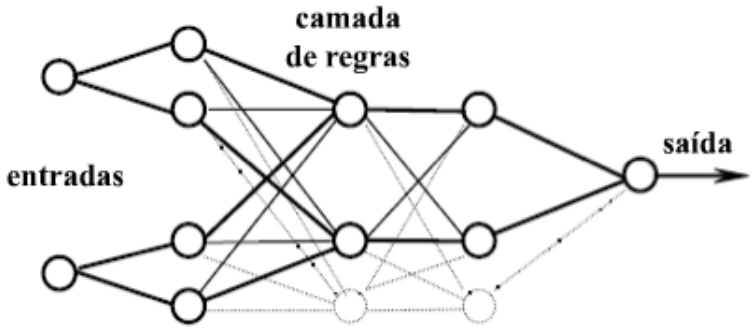


Figura 3.2: Arquitetura da Rede Neural *Fuzzy* Evolutiva (Kasabov, 2007).

ela é chamada de camada de retroalimentação. As conexões dessa camada podem ser usadas para memorizar o relacionamento temporal dos dados de entrada (Kasabov, 2007).

Uma versão reduzida da EFuNN poderia ser uma rede de três camadas, cujas camadas de entrada e saída *fuzzy* seriam eliminadas. Nesse caso, o algoritmo de treino seria ligeiramente modificado, se assemelhando ao do eMLP (Kasabov, 2007).

Cada neurônio na camada de regras possui um próprio conjunto de parâmetros locais, que são a idade, ativação média e a densidade da vizinhança dele. Esses parâmetros são utilizados para o procedimento de poda da rede, que utiliza uma regra *fuzzy* e requer que conceitos relacionados aos parâmetros sejam pré-definidos. Por exemplo, a idade pode ser definida como $NOVOS$ ou $VELHOS$, sendo que um neurônio pode ser considerado $VELHO$ após p amostras de treino. A regra de poda e os valores para os parâmetros são definidos de acordo com a aplicação da tarefa (Kasabov, 2007).

Inicialmente, pode-se imaginar que essa rede neural possui uma estrutura com um número máximo de neurônios, porém os neurônios da camada de regras não possuem nenhuma conexão com as outras camadas. Assim sendo, quando é apresentada uma nova amostra de treinamento x, e não houver nenhum neurônio conectado na camada de regras ou houver necessidade de adicionar um novo neurônio j nessa camada, os pesos das conexões We_j e Ws_j são obtidos por:

$$We_j = x_{fe}, \quad Ws_j = x_{fs}, \tag{3.10}$$

onde x_{fe} e x_{fs} são os dados de entrada (vetor de atributos) e saída (vetor de classes) no espaço *fuzzy* associados à amostra x, respectivamente.

Em alguns momentos um procedimento de atualização dos pesos dos neurônios é necessário. Para atualizar as conexões do neurônio j da camada de regras são usadas as Equações 3.11 e 3.12, onde A_j é a ativação do neurônio j para a entrada x_{fe} e η_1 e η_2 são as taxas de

aprendizado para os vetores de pesos We_j e Ws_j, respectivamente. A dimensão dos vetores We_j e x_{fe} é d, e do Ws_j é n, tal que $i = 1,\ldots,d$ e $p = 1,\ldots,n$.

$$We_{j,i} = We_{j,i} + \eta_1(x_{fe,i} - We_{j,i}), \tag{3.11}$$

$$Ws_{j,p} = Ws_{j,p} + \eta_2 A_j E_p. \tag{3.12}$$

O vetor E tem dimensão n e ele é a diferença entre a saída desejada x_{fs} e o vetor de saídas calculadas Oc (Equação 3.13). Os vetores x_{fs} e Oc também possuem dimensão n.

$$E_p = x_{fs,p} - Oc_p. \tag{3.13}$$

A ativação A_j e o vetor de saída Oc são calculados através das funções $A_j = f_1(We_j, x_{fe})$ e $Oc = f_2(Ws, A)$, onde A é o vetor de ativação dos neurônios na camada de regras para a entrada x_{fe}. Diferentes funções podem ser usadas para f_1 e f_2, incluindo a Equação 3.1.

Existem algumas variantes do procedimento de aprendizado desta rede neural (Kasabov, 2007). No entanto, o procedimento padrão de aprendizado é dado no Algoritmo 4 (Kasabov, 1998; Woodford, 2001).

3.3.2 Vantagens e Desvantagens

A EFuNN já foi utilizada em um grande número de aplicações e, para alguns casos, apresentou alguns dos melhores resultados quando comparada com outras técnicas. Além disso, é possível extrair regras *fuzzy* de sua arquitetura.

No entanto, as desvantagens são que esta rede neural apresenta uma complexidade maior do que as redes neurais anteriormente mencionadas e apresenta um número maior de neurônios, devido ao processo de *fuzzification*, o que também influencia no custo computacional da mesma. Além disso, para algumas situações, a *fuzzification* da entrada não somente é desnecessário (caso de base de dados binários), mas também podem prejudicar o desempenho. Para a maioria das aplicações, técnicas como a rede eMLP são capazes de modelar o conjunto de treinamento com um número menor de neurônios na camada evolutiva do que uma equivalente EFuNN (Watts, 2009; Watts, 2004).

Algoritmo *Algoritmo de aprendizado da EFuNN*

inicializar uma estrutura EFuNN com um número máximo de neurônios e zero conexões
para cada novo dado de treinamento **faça**
    Realiza o processo de *fuzzification* do dado de entrada-saída
    se não há nenhum neurônio na camada de regras com conexões diferentes de zero **então**
        conectar o primeiro neurônio às camadas de entrada e saída *fuzzy*, conforme as Equações em 3.10
    senão
        se há atributos novos que aparecem nesse dado e não têm sido usados em dados anteriores **então**
            criar novas conexões de entrada e/ou saída para os neurônios na camada de regras, de acordo com as correspondentes funções de pertinência
        encontrar a similaridade *fuzzy* normalizada (Equação 3.2) entre o dado com quantização *fuzzy* e os padrões já armazenados nos neurônios da camada de regras
        a partir da similaridade obter a ativação dos neurônios da camada de regras, conforme a função de ativação
        atualizar os parâmetros locais definidos para os neurônios, tais como: a camada de memória de curto prazo (se houver), idade, ativação média, etc.
        encontrar todos os neurônios com um valor de ativação acima de um limiar de sensibilidade S_{thr}
        se não há nenhum neurônio com um valor de ativação maior do que o limiar S_{thr} **então**
            conectar um neurônio às camadas de entrada e saída *fuzzy*, conforme as Equações em 3.10
        senão
            encontrar o neurônio $inda1$ com o maior valor de ativação A
            se modo de treinamento for "1 de m" **então**
                propagar a ativação do neurônio mais ativo para a camada de saída *fuzzy*
            se modo de treinamento for "muitos de m" **então**
                somente os valores de ativação acima do limiar S_{th} são propagados para a a camada de saída *fuzzy*
            encontrar o neurônio $inda2$ vencedor na camada de saída *fuzzy* e a ativação dele
            encontrar o neurônio $indt2$ vencedor desejado na camada de saída *fuzzy* e a ativação dele
            calcular o erro de saída *fuzzy* $E = x_{fs} - Oc$, onde x_{fs} é a saída *fuzzy* desejada e Oc é a saída calculada
            se $inda2$ é diferente de $indt2$ ou $abs(E_{inda2}) > E_{thr}$ **então**
                conectar um neurônio às camadas de entrada e saída *fuzzy*, conforme as Equações em 3.10
            atualizar as conexões de entrada e saída do neurônio $inda1$ aplicando as Equações 3.11 e 3.12, respectivamente
            se se houver a camada de memória de curto prazo **então**
                atualizar as conexões de retro-propagação dos neurônios dela
            para cada neurônio na camada de regras **faça**
                se o neurônio é $VELHO$ e ativação média dele é $BAIXA$ e a densidade dos neurônios nas áreas vizinhas é $ALTA$ ou $MODERADA$ e a soma dos pesos das conexões de entrada ou saída é $BAIXA$ e o neurônio não está associado com a correspondente classe de saída atual (para tarefas de classificação somente) **então**
                    a probabilidade de poda do neurônio é $ALTA$
            agregar neurônios na camada de regras

Algoritmo 4: O algoritmo de treinamento da EFuNN.

3.4 Rede Neural Probabilística Incremental Baseada em *Expectation Maximization* (RNPI-EM)

3.4.1 Algoritmo

O problema de estimar a função densidade de probabilidade baseada em um conjunto de amostras obtidas sequencialmente, uma por vez, ao invés de obtê-las em lote, foi estudado em (Vlassis et al., 1999). Como resultado da pesquisa, os autores também propuseram um classificador baseado na Rede Neural Probabilística, que aprende de forma incremental à

medida que uma nova amostra de treinamento é obtida.

Nesta versão de aprendizado incremental da Rede Neural Probabilística, a camada de padrões de cada classe da rede neural é vista como um modelo de mistura de Gaussianas e os parâmetros do modelo são ajustados de forma incremental através de *Expectation Maximization*. Sendo assim, a rede neural é composta por $|C|$ modelos de mistura de Gaussianas, onde $|C|$ é o número total de classes. Além disso, o número de Gaussianas, ou seja, o número de neurônios, que compõe cada modelo é estimado à medida que novos dados são usados para treinamento. Para tanto, o algoritmo de treinamento da rede neural incorpora procedimentos de divisão, união e remoção de neurônios da rede.

O modelo de mistura de Gaussianas de uma classe c_m, $m = 1, \ldots, |C|$, da rede neural é representado por:

$$p(x) = \sum_{j=1}^{K_m} \omega_j f(x, \mu_j, \Sigma_j), \tag{3.14}$$

onde K_m é o número de *kernels*, ou neurônios, da camada de padrões da classe c_m, ω_j é o peso do j-ésimo neurônio da camada de padrões da classe c_m, tal que $\sum_{j=1}^{K_m} \omega_j = 1$. A função $f(x, \mu_j, \Sigma_j)$ é a função de transferência do neurônio j, onde x é um vetor de entrada com d dimensões, o vetor média μ_j possui d dimensões e a matriz de covariância Σ_j tem tamanho $d \times d$.

Essa função de transferência, também conhecida como função densidade de probabilidade da Gaussiana, é escrita como na Equação 3.15:

$$f(x, \mu_j, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} exp\left[\frac{-(x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j)}{2}\right], \tag{3.15}$$

onde $|\Sigma_j|$ e Σ_j^{-1} são o determinante e a inversa de Σ_j, respectivamente. Por simplicidade e devido ao grande esforço computacional para calcular $|\Sigma_j|$ e Σ_j^{-1}, principalmente para altos valores de d, os autores de (Vlassis et al., 1999) trabalharam com a suposição que as características são estatisticamente independentes. Assim, os valores de covariância entre características distintas são iguais a zero. Nesse caso, Σ_j é uma matriz diagonal (Duda et al., 2001), como é mostrado na Equação 3.16:

$$\Sigma_j = \begin{bmatrix} \sigma_{j,1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{j,2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{j,d}^2 \end{bmatrix}. \tag{3.16}$$

Cada neurônio j da camada de padrões é composto por um vetor de média μ_j, uma matriz diagonal Σ_j e um peso de conexão ω_j, entre j e o neurônio da camada de soma da classe c_m. Portanto, é necessário estimar esses parâmetros para ajustar a rede neural.

Em (Vlassis et al., 1999) são deduzidas e apresentadas equações iterativas para atualizar esses parâmetros para cada neurônio, sempre que um novo dado x é usado para o treinamento. As Equações 3.17, 3.18 e 3.19 as apresentam para $i = 1,\ldots,d$ e $j = 1,\ldots,K_m$. Os índices t e $t+1$ se referem aos valores atuais e futuros das grandezas.

$$\mu_{j,i,t+1} = \mu_{j,i,t} + \frac{P(j \mid x)(x_i - \mu_{j,i,t})}{n\omega_{j,t}}, \tag{3.17}$$

$$\sigma^2_{j,i,t+1} = \sigma^2_{j,i,t} + \frac{P(j \mid x)\left[(x_i - \mu_{j,i,t+1})^2 - \sigma^2_{j,i,t}\right]}{n\omega_{j,t}}, \tag{3.18}$$

$$\omega_{j,t+1} = \omega_{j,t} + \frac{P(j \mid x) - \omega_{j,t}}{n}, \tag{3.19}$$

onde:

$$P(j \mid x) = \frac{\omega_{j,t} f(x, \mu_{j,t}, \Sigma_{j,t})}{p(x)}. \tag{3.20}$$

Em seguida é atualizado o vetor de d dimensões do valor da curtose de cada neurônio usando a Equação 3.21. A curtose é o quarto momento estatístico e ele é usado para determinar se um neurônio deve ser dividido ou não. Uma explicação mais detalhada sobre a curtose é dada no Apêndice B.

$$k_{j,i,t+1} = k_{j,i,t} + \frac{P(j \mid x)\left[\left(\frac{x_i - \mu_{j,i,t+1}}{\sigma_{j,i,t+1}}\right)^4 - k_{j,i,t} - 3\right]}{n\omega_{j,t+1}}. \tag{3.21}$$

A constante n das equações anteriores pode ser vista como um procedimento de média móvel sobre as entradas anteriores. Quanto menor o valor de n, mais rápidas são as alterações dos valores dos parâmetros. Além disso, n influencia os testes estatísticos que estimam o número de neurônios, conforme mencionado a seguir.

Os procedimentos para estimar a quantidade de neurônios são apresentados nos próximos passos. Inicialmente é utilizado um teste estatístico para verificar a hipótese de que os dados de entrada seguem uma distribuição Gaussiana ou não. Assim, baseado no valor da curtose de cada dimensão do neurônio j, é aplicada a Equação 3.22 ($i = 1,\ldots,d$).

$$q_{j,i} = k_{j,i}\sqrt{\frac{n\omega_j}{96}}. \tag{3.22}$$

Para as dimensões onde $|q_{j,i}| < z_{1-\alpha/2}$, a distribuição é considerada normal e nada é feito nessas dimensões, sendo u o percentil da normal z_u, e α o nível de significância do teste. No entanto, para cada dimensão em que o teste indicar que a distribuição não é normal, o neurônio é dividido em dois, e novos parâmetros devem ser calculados para tal dimensão. Quando

um neurônio r é dividido em s e t para a dimensão i, os parâmetros dos novos neurônios para a dimensão i são calculados empregando as Equações 3.23 a 3.27. As demais dimensões são mantidas sem alterações e repetidas nos novos neurônios. Após o procedimento passar por todas as dimensões, os neurônios que sofreram divisão são removidos e os pesos são normalizados, tal que $\sum_{j=1}^{K_m} \omega_j = 1$.

$$\mu_{s,i} = \mu_{r,i} + \sigma_{r,i}, \tag{3.23}$$

$$\mu_{t,i} = \mu_{r,i} - \sigma_{r,i}, \tag{3.24}$$

$$\sigma_{s,i} = \sigma_{t,i} = \sigma_{r,i}, \tag{3.25}$$

$$k_{s,i} = k_{t,i} = 0, \tag{3.26}$$

$$\omega_{s,i} = \omega_{t,i} = \omega_{r,i}. \tag{3.27}$$

O próximo passo é verificar se dois neurônios podem ser unidos ou não. Um critério razoável para unir dois neurônios em um é examinar a proximidade da variância e a média deles (Vlassis et al., 1999). Para verificar isso, inicialmente o teste estatístico F de *Snedecor* (Box, 1953; Markowski e Markowski, 1990) é aplicado para analisar a hipótese de variâncias iguais e, se for obtido sucesso, então o teste estatístico t de *Student* (Zimmerman, 1997) é aplicado para médias iguais, supondo variâncias iguais. Se ambos os testes forem bem sucedidos, então os neurônios são unidos. Assim, para o teste F de *Snedecor* ser aplicado, primeiramente são selecionados os neurônios com a maior (p) e a segunda maior (q) probabilidade *a posteriori*. Depois é formada a razão da maior para a menor variância desses dois neurônios, como é mostrada na Equação 3.28. Esse procedimento deve ser realizado para todas as dimensões ($i = 1, ..., d$).

$$F_i = \frac{\sigma_{p,i}^2}{\sigma_{q,i}^2} \tag{3.28}$$

Desde que $F_i \geq 1$, a hipótese de variâncias iguais é aceita se F_i satisfaz $F_i < F_{n,n,1-\alpha/2}$, onde n é a mesma constante usada para atualizar os parâmetros da rede. Se esse teste for bem sucedido para todas as dimensões, o teste estatístico t de *Student* é usado aplicando a Equação 3.29, na qual é assumida a variância comum $\sigma_i^2 = \sigma_{p,i}\sigma_{q,i}$. Se $|t_i| < z_{1-\alpha/2}$ para todas as dimensões, então a hipótese de médias iguais é aceita e os neurônios são unidos.

$$t_i = \sqrt{n\omega_p}\left(\frac{\mu_{p,i} - \mu_{q,i}}{\sigma_i\sqrt{2}}\right). \tag{3.29}$$

De forma similar para o caso de divisão dos neurônios, nesse procedimento é necessário também obter os novos parâmetros do novo neurônio. Sejam p e q os neurônios que serão unidos e r o novo neurônio, os parâmetros são obtidos empregando as Equações 3.30, 3.31 e

Algoritmo *Rede Neural Probabilística baseada em Expectation Maximization*

1 **para cada** dado de entrada $x = [x_1, \ldots, x_d]$ pertencente a classe c_m **faça**
2 **se** classe c_m não tem recebido qualquer entrada ainda **então**
3 criar um novo neurônio j para a classe c_m, onde $\omega_j = 1$, $\mu_j = x$, $\sigma_{j,i} = 1$ e $k_{j,i} = 0$, para $i = 1, \ldots, d$
4 atualizar a probabilidade *a priori* da classe c_m
5 calcular a partir da Equação 3.20 a probabilidade *a posteriori* para todos os neurônios da camada de padrões da classe c_m
6 aplicar os testes 3.28 e 3.29 sobre os neurônios com a maior (p) e a segunda maior (q) probabilidade *a posteriori*
7 **se** teste for bem sucedido em todas as dimensões **então**
8 unir os dois neurônios em um usando 3.30, 3.31 e 3.32 para obter os parâmetros
9 excluir os neurônios p e q
10 normalizar os pesos, tal que $\sum_{j=1}^{K_m} \omega_j = 1$
11 **para cada** neurônio j, $j = 1, \ldots, K_m$, da classe c_m **faça**
12 atualizar ω_j usando 3.19
13 **para cada** dimensão i, $i = 1, \ldots, d$ **faça**
14 atualizar os parâmetros do neurônio j na dimensão i usando 3.17, 3.18 e 3.21
15 aplicar teste 3.22 sobre a dimensão i do neurônio j
16 **se** teste for bem sucedido **então**
17 dividir o neurônio j sobre a dimensão i nos neurônios s e t
18 calcular os novos parâmetros para a dimensão i usando 3.23, 3.24, 3.25, 3.26 e 3.27
19 normalizar os pesos, tal que $\sum_{j=1}^{K_m} \omega_j = 1$
20 remover neurônios com valores de pesos menores do que $1/n$
21 normalizar os pesos, tal que $\sum_{j=1}^{K_m} \omega_j = 1$

Algoritmo 5: Algoritmo de treinamento da Rede Neural Probabilística apresentada em (Vlassis et al., 1999).

3.32. Então os neurônios p e q são eliminados e novamente os pesos são normalizados para obter $\sum_{j=1}^{K_m} \omega_j = 1$.

$$\mu_{r,i} = \frac{\mu_{p,i} + \mu_{q,i}}{2}, \tag{3.30}$$

$$\sigma_{r,i} = \sqrt{\sigma_{p,i}\sigma_{q,i}}, \tag{3.31}$$

$$\omega_r = \omega_p. \tag{3.32}$$

Finalmente um passo de remoção de neurônios é realizado. O método adotado para realizar essa tarefa é usar um limiar sobre os valores dos pesos dos neurônios. Neurônios com pesos abaixo desse limiar são removidos.

O Algoritmo 5 ilustra todos os passos usados para treinar a Rede Neural Probabilística apresentada em (Vlassis et al., 1999).

Como mostrado no Algoritmo 5 linha 4, existe um procedimento de atualização da probabilidade *a priori* da classe que acabou de receber uma nova amostra de treinamento. Essa atualização da probabilidade deve ser feita tal que a probabilidade *a priori* das classes seja proporcional ao número de amostras usadas para treiná-las.

3.4.2 Vantagens e Desvantagens

Embora esta abordagem seja capaz de realizar muitos tipos de procedimentos e ajustar uma gama muito grande de parâmetros, ela apresenta alguns problemas. Primeiro, essa rede neural usa momento de alta ordem para decidir quando um neurônio deve ser dividido. No entanto, momentos de alta ordem são quase sempre menos robustos do que momentos de ordem menor (tais como média e variância), pois a estimativa deles tende a ser mais instável na presença de *outliers* (amostras inconsistentes com a distribuição do conjunto de amostras) (Welling, 2005). Isso agrava quando eles são calculados em modo incremental. Por essa razão, essas medidas devem ser usadas com cautela (Press et al., 2007).

Segundo, um neurônio pode ser dividido em até d neurônios, onde d é a dimensão do dado. Dessa forma, pode ocorrer uma explosão computacional, principalmente se a dimensão dos dados for grande. Terceiro, o procedimento de divisão dos neurônios apresenta um problema devido ao grau de liberdade: existe um número de equações menor do que o número de parâmetros desconhecidos, então é necessário fazer suposições da forma como eles devem ser divididos. Finalmente, a calibração das médias e variâncias é incremental, de forma que os valores desses parâmetros têm uma imprecisão inerente adicionada a eles. Para poucas dimensões, esse problema não afeta significantemente o desempenho. No entanto, para altas dimensões, a propagação da incerteza pode prejudicar o desempenho da rede, principalmente por causa da matriz de covariância, que tem maior efeito sobre esta rede neural.

De forma geral, percebe-se que esta rede neural pode apresentar bons resultados para problemas que envolvem baixas dimensões, por outro lado, o seu desempenho para problemas de altas dimensões pode ser muito aquém do desejado.

3.5 Outras Redes Neurais com Aprendizado Incremental

As redes neurais citadas anteriormente foram mencionadas com mais detalhes por serem mais próximas ao modelo proposto ou por sua relevância na literatura. No entanto, diversos tipos de redes com aprendizado incremental foram propostos na literatura. Algumas outras versões incrementais de redes neurais que se destacam são:

- eSOM (*Evolving Self-Organize Maps*): versão incremental dos mapas auto organizáveis. Basicamente, essas redes neurais possuem aprendizado não supervisionado e podem dinamicamente alterar o número de saídas (centroides) (Deng e Kasabov, 2000).

- DENFIS (*Dynamic Evolving Neural-Fuzzy Inference Systems*): utiliza princípios de inferência *fuzzy*, além de ser baseada numa técnica de atualização de centroides, chamada de ECM (*evolving clustering method*) (Song e Kasabov, 2001). Também possui procedimentos semelhantes à EFuNN (Kasabov e Song, 2002).

- DARN (*Deterministic Adaptive RAM Networks*): é um tipo de rede neural sem peso derivado do trabalho de Aleksander e Kan (Kan e Aleksander, 1989) que apresenta características de aprendizado incremental (Coghill et al., 2003).

- Extensão temporal do SCE: uma extensão temporal do SCE baseado na rede recorrente de Jordan-Elman (Elman, 1990; Jordan, 1986) também foi proposta. Essa extensão adiciona uma segunda camada evolutiva ao SCE, que é somente conectada à camada evolutiva principal da rede (Kasabov e Watts, 1999).

- ZISC (*Zero Instruction Set Computer*): é um sistema de aprendizado supervisionado implementado em um chip, baseado em uma Rede de Função de Base Radial, cuja arquitetura é adaptável e o aprendizado é a partir de amostra de dados. Nesse sistema nem todos os dados de entrada são rotulados com uma classe, pois eles podem também ser rotulados como não identificados (se relacionados a mais de uma classe) ou não reconhecidos (se relacionados a nenhuma classe) (Coghill et al., 2003; Kasabov, 2007).

- ECF (*Evolving Classification Function*): ECF é uma rede neural supervisionada que usa lógica *fuzzy* para extrair informação. Nessa técnica, sempre que houver necessidade, um dado de treinamento é convertido em neurônio, sendo os pesos das conexões do neurônio obtidos através da quantização *fuzzy* do dado. Além disso, cada neurônio possui um parâmetro chamado de campo de influência, que é uma região do espaço em torno do neurônio na qual o neurônio tem efeito (Coghill et al., 2003; Kasabov, 2007).

3.6 Aplicações

Existe um grande número de publicações na literatura que tem utilizado estas redes neurais em uma larga variedade de tipos de aplicações. Aqui será mencionada somente uma parcela dessas publicações.

A RNPI e algumas versões semelhantes foram aplicadas em (Bhattacharyya et al., 2008; Fan et al., 2004; Ciarelli et al., 2009b). Em (Bhattacharyya et al., 2008) os autores a utilizaram como uma ferramenta para reconhecer padrões em um nariz eletrônico para classificar chá preto. Como o odor do chá preto se altera conforme a estação do ano e região em que ele é colhido, além de somente pequenas quantidades serem disponibilizadas para treinamento

ao longo do tempo, um sistema de aprendizado incremental mostra-se ser apropriado para o caso. Outra aplicação desta rede neural com finalidade diferente foi apresentada em (Fan et al., 2004). Neste trabalho foi proposto um sistema de reconhecimento de face para vídeos em que uma Rede Neural Probabilística é usada para identificar a face. Sempre que uma face desconhecida aparece no vídeo, é adicionada uma nova classe à rede neural e a face desconhecida é usada como treinamento. O rápido treinamento dessa rede neural permite que o sistema trabalhe *on-line* o tempo todo. Por fim, o desempenho da RNPI foi comparado ao de outra técnica na tarefa de classificação de textos em (Ciarelli et al., 2009b). Em todos esses trabalhos foi observado que essa rede neural apresenta um desempenho bom, mas que, no entanto, o tamanho da arquitetura e a quantidade de memória requerida por ela é proporcional ao tamanho da base de dados usada para treino, pois ela armazena toda a base de dados na memória, o que pode ser um grande problema para algumas tarefas.

Por certas razões, os sistemas de potência elétrica podem apresentar alguns problemas, como instabilidade da frequência de alimentação ou valores de tensão fora da faixa nominal. Para tratar de tais problemas, os autores de (Gavoyiannis et al., 2005) usaram a abordagem da RNPI-ME para identificar automaticamente tal situação e realizar a mudança das condições de operação. Nos experimentos realizados foi constatado que essa rede neural possui uma rápida adaptação ao problema, processamento extremamente rápido, e um desempenho acima de 90% na correta identificação do estado do sistema. Além disso, como a natureza dos sistemas de potência é relativamente lenta, essa rede neural pode ser facilmente atualizada, sem interferência humana. No entanto, a dimensão dos dados de entrada para essa rede neural era de apenas sete, o que não a tornava susceptível aos problemas quando altas dimensões estão envolvidas.

A RNPI-ME também foi avaliada no artigo no qual foi proposta. Entretanto, os experimentos em (Vlassis et al., 1999) foram realizados somente sobre base de dados sintéticas com poucas dimensões (uma e duas dimensões).

Em (Leite et al., 2009), os algoritmos do eMLP e do eSOM foram aplicados para a detecção de faltas em máquinas elétricas, e os mesmos foram comparados ao MLP (*Multilayer Perceptron*), SOM (*Self-Organizing Maps*) e a rede Elman. Os resultados indicaram que além de possuir uma precisão maior do que as outras técnicas (acima de 95%), elas necessitaram de um tempo menor de treinamento. Outra comparação entre o eMLP e o MLP foi realizada em (Watts e Kasabov, 2000). Nesse trabalho foi usada como avaliação uma base de dados para reconhecimento de fonemas isolados. A comparação mostrou que, embora o eMLP apresentasse um número maior de neurônios, ele era muito mais adaptável, capaz de reter sua capacidade discriminativa mesmo depois de ser treinado com novas amostras.

O eMLP foi também aplicado para realizar o controle por voz de um robô (Ghobakhlou e

Seesink, 2001), e para controlar um sistema de automação de uma casa através de reconhecimento de palavras (Ghobakhlou e Kasabov, 2003). Em ambos os trabalhos foi demonstrada a capacidade desta rede de reter conhecimento antigo e acomodar conhecimento novo. O problema de detecção de comportamento anormal pelos ocupantes de um ambiente inteligente foi abordado em (Illingworth et al., 2005). Embora a precisão do eMLP tenha sido equivalente a outros métodos investigados, o rápido treinamento e adaptabilidade do eMLP tornaram ele mais útil. O eMLP e o MLP foram avaliados em (Watts e Worner, 2007) sobre uma base de dados de predição da abundância de um tipo de animal na Nova Zelândia. As conclusões obtidas nos experimentos foi que o eMLP foi capaz de adaptar melhor aos novos dados do que o MLP, embora o eMLP não tenha sido capaz de generalizar tão bem quanto o MLP.

Um método de identificar pessoas combinando as informações da fala e imagem foi proposta em (Zhang et al., 2004), usando a ECF e ZISC. Ambas as técnicas mostraram ser apropriadas para a criação de modelos para essa tarefa, pois elas mostraram ser adaptáveis, podem acomodar novas classes (pessoas) sem degradar o seu desempenho para as classes existentes. Em (Coghill et al., 2003) foi realizada uma comparação entre a ECF, ZISC e o MLP na tarefa de reconhecimento de fonema. Enquanto a ECF obteve precisão tão boa quanto ao da MLP com um baixo tempo de treinamento; o ZISC obteve uma precisão inferior aos dois. Um possível motivo pode ser o esquema de classificação adotado pelo ZISC.

A EFuNN é uma das redes apresentadas que foram mais aplicadas na literatura. Em (Abraham e Nath, 2000) foi revista uma variedade de algoritmos sobre a base de dados de série temporal caótica Mackey-Glass e foi observado que a EFuNN era uma das mais precisas e rápidas das técnicas avaliadas. Em (Murali et al., 2003; Ng et al., 2006), a EFuNN foi comparada a uma outra rede neural baseada em inferência *fuzzy*, chamada de ANFIS, sobre o problema de reconhecimento de caligrafia, e a EFuNN mostrou ser mais precisa, mais rápida para o treinamento e melhor para adaptação aos novos dados. No entanto, ela também foi mais lenta para a classificação e precisou de mais memória devido ao grande tamanho de sua arquitetura. Uma avaliação rigorosa da EFuNN e de redes similares ao eMLP foi feito em (Watts, 2004). Enquanto a EFuNN foi comparada a uma rede neural *backpropagation* de inferência *fuzzy* treinada; as versões do eMLP foram comparadas à uma rede MLP treinada. Os resultados mostraram que nas três bases de dados usadas para avaliação as versões da eMLP e da EFuNN e foram capazes de aprender e generalizar tão bem quanto o MLP e a outra técnica, não havendo diferenças estatisticamente relevantes entre a precisão deles, fato notado após ser aplicado um teste estatístico com nível de confiança de 99%.

A tarefa de reconhecer pronúncia em um ambiente ruidoso foi feita em (Kasabov e Iliev, 2000), comparando EFuNN com o LVQ (*Linear Vector Quantization*). A EFuNN conseguiu ter uma taxa de reconhecimento maior do que o LVQ nos experimentos realizados. Vários

modelos de redes neurais com inferência *fuzzy* foram investigados em (Liu et al., 2006) na tarefa de fazer o relacionamento entre a configuração do ventilador usado para alimentar a respiração de um paciente num hospital e o nível de oxigênio no sangue. As redes EFuNN e DENFIS apresentaram baixos erros, embora não tenham obtido o menor nível de erro alcançado na pesquisa. No entanto, o número de neurônios produzidos pela EFuNN foi um dois maiores. Outra comparação foi feita entre o MLP e a EFuNN na aplicação de classificação de imagens de texturas no trabalho apresentado em (Sorwar et al., 2001). A EFuNN obteve uma precisão maior e um tempo de treinamento menor do que o MLP. Aplicações em horticultura foi o tema em (Woodford et al., 2004; Woodford, 2008). O problema nesses trabalhos era identificar a origem dos danos feitos nos frutos e folhas das macieiras. Nessa tarefa, a EFuNN foi mais precisa do que o k-*means*, MLP e SVM (*Support Vector Machine*). Além disso, a habilidade de extrair regras *fuzzy* da EFuNN foi considerada ser uma vantagem nessa aplicação.

O trabalho apresentado em (Abraham e Nath, 2001) comparou uma rede neural *back-propagation*, um modelo ARIMA e a EFuNN para predição de demanda de eletricidade em um estado australiano, e foi observado que a EFuNN foi a mais precisa. Num trabalho seguinte (Bhattacharya et al., 2002), ela foi comparada com programação linear genética, mas ainda assim ela continuou sendo superior.

3.7 Conclusão

Este capítulo apontou e discutiu algumas técnicas neurais de aprendizado incremental que apresentam todos ou pelo menos alguns critérios estabelecidos pela família de redes neurais SCE, e que são próximas ao modelo proposto no capítulo seguinte ou que apresentem relevância na literatura. Na Seção 3.6, foram mencionados alguns trabalhos na literatura que apontam que, de forma geral, as técnicas aqui apresentadas apresentam uma adaptação e um treinamento mais rápido e um desempenho na classificação/identificação que pode ser igual ou superior às técnicas tradicionais, especialmente o MLP.

Capítulo 4

Modelo Proposto

Nesta parte, uma versão incremental da Rede Neural Probabilística (RNP), chamada de Rede Neural Probabilística evolutiva (RNPe), é descrita. Diferentemente da RNP, a RNPe tem capacidade de controlar o tamanho de sua arquitetura e, assim, evitar um crescimento descontrolado da mesma, como acontece com a RNP. Além de se basear na RNP, a rede neural proposta também é baseada no Modelo de Mistura de Gaussianas (MMG), no algoritmo de *Expectation Maximization* e possui procedimentos inspirados em algumas abordagens utilizadas nas redes neurais apresentadas no Capítulo 3. Após a descrição desta técnica, é feito um comparativo com as outras técnicas descritas no Capítulo 3. Também aqui é descrita uma abordagem que utiliza a rede neural proposta para aprendizado semi-supervisionado.

4.1 Rede Neural Probabilística Evolutiva (RNPe)

A abordagem proposta aqui, RNPe, é inspirada em algumas técnicas clássicas, tais como a Rede Neural Probabilística (RNP), o Modelo de Mistura de Gaussianas (MMG) e no algoritmo de *Expectation Maximization* (EM). Detalhes sobre MMG e EM são apresentados no Anexo D. O objetivo da rede neural apresentada é obter um modelo com desempenho e flexibilidade semelhantes à RNP, mas com arquitetura mais reduzida, proporcionando menor custo computacional. A seguir, uma breve descrição da RNP, seguida de algumas desvantagens de seu algoritmo e as modificações sugeridas.

A RNP é uma rede neural supervisionada e não linear, baseada na teoria de decisão Bayesiana. Ela é constituída de quatro camadas: camada de entrada, de padrões, de soma e de decisão (Specht, 1990). Na Figura 4.1 é ilustrada a arquitetura da RNP, onde cada amostra x de um problema é apresentada à camada de entrada, e a classe associada à essa

amostra é retornada na camada de saída. Cada neurônio na camada de entrada está conectada a todos os neurônios na camada de padrões. Por sua vez, a camada de padrões é parcialmente conectada à camada de soma, onde cada neurônio na camada de soma corresponde a uma classe (o número de neurônios nessa camada é igual ao número de classes). Em outras palavras, cada neurônio da camada de padrões está associado a uma classe, sendo, portanto, conectado somente ao neurônio da camada de soma que representa essa classe. Finalmente, todos os neurônios na camada de soma estão conectados ao único neurônio da camada de saída.

Na etapa de classificação, cada amostra de entrada x é apresentada para a camada de entrada. Essa camada não realiza cálculo, enviando a amostra x para a camada de padrões. Nesta, cada neurônio calcula uma saída para a amostra x usando a função Gaussiana $f_i(x, w_i, \sigma)$ indicada na Equação 4.1. Na equação, é ilustrado o cálculo realizado no i-ésimo neurônio da camada de padrões: d é o número de dimensões do vetor x, w_i é o vetor peso com d dimensões do i-ésimo neurônio; e σ^2 é a variância da Gaussiana. O símbolo T representa a transposta do vetor $(x - w_i)$.

$$f_i(x, w_i, \sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left[\frac{-(x-w_i)^T(x-w_i)}{2\sigma^2}\right]. \tag{4.1}$$

As saídas dos neurônios da camada de padrões são as entradas dos neurônios da camada de soma. Cada neurônio da camada de soma está associado a uma classe, e elas fornecem para a camada de decisão a soma das suas entradas. Finalmente, a classe com a maior saída é associada à amostra x na camada de decisão.

O treinamento da RNP é fácil e rápido: consiste em associar cada vetor de amostra de treinamento a um vetor de pesos de um neurônio na camada de padrões (Specht, 1990). O valor de σ^2 é um parâmetro que deve ser escolhido antes do treinamento da rede. Mais detalhes sobre a arquitetura, e os procedimentos de treinamento e classificação da RNP são encontrados no Anexo C.

Apesar de sua fácil implementação, rápido treinamento e arquitetura flexível, a RNP possui uma grande desvantagem: o número crescente de neurônios na camada de padrões. O número de neurônios nessa camada é igual à quantidade de amostras usadas para treinamento. Então, se for usada uma base de treinamento muito grande, o tamanho da sua arquitetura se torna enorme, comprometendo o tempo de resposta da rede.

Para evitar que a arquitetura cresça demasiadamente, diversas abordagens foram propostas. Algumas delas, tal como a descrita em (Heinen e Engel, 2010), propõem usar em cada neurônio da camada de padrões uma função Gaussiana com matriz de covariância completa. Nesta, os parâmetros são estimados por técnicas de maximização da verossimilhança. Com

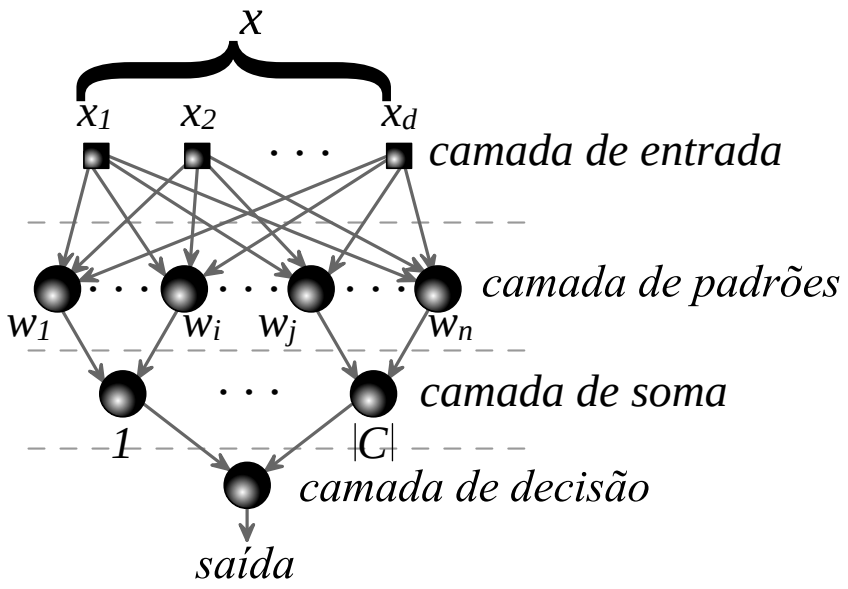


Figura 4.1: Arquitetura da Rede Neural Probabilística, na qual x é a amostra de entrada, d é a dimensão do vetor amostra x, w_i é o vetor peso com d dimensões do i-ésimo neurônio da camada de padrões, n é o total de neurônios na camada de padrões e $|C|$ é o número de classes. A saída retorna a classe associada pela rede à amostra x.

esse procedimento adicional, a RNP pode ser representada usando um pequeno conjunto de funções Gaussianas. A Equação 4.2 indica essa função Gaussiana, onde d é a dimensão da amostra x, μ é o vetor média de tamanho d, Σ é a matriz de covariância de tamanho $d \times d$, e $|\Sigma|$ e Σ^{-1} são o determinante e a inversa de Σ, respectivamente. O valor de d é igual ao número de atributos na amostra x.

$$f(x,\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left[\frac{-(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right]. \tag{4.2}$$

Resolver a Equação 4.2 apresenta desvantagens para o método proposto, pois é necessário calcular a inversa e o determinante da matriz de covariância Σ. O problema de calcular a inversa e o determinante de uma matriz é o custo computacional alto quando a matriz possui grande dimensão (um valor elevado de d). A matriz de covariância pode também ser singular em alguns casos e, assim, não possui inversa.

Para reduzir o grande esforço computacional nos cálculos de Σ^{-1} e $|\Sigma|$ para altos valores de d, a literatura normalmente assume uma hipótese simplificadora das características do problema serem estatisticamente independentes (descorrelacionadas). Assim, o valor da covariância entre diferentes características é igual a zero. Nesse caso, Σ é uma matriz diagonal (Duda et al., 2001), como mostrado na Equação 4.3, onde σ_i^2 é o valor da variância do i-ésimo atributo.

$$\Sigma = \text{diag}(\sigma_1^2\ \sigma_2^2\ \cdots\ \sigma_d^2). \tag{4.3}$$

Como resultado dessa suposição, os cálculos se tornam mais rápidos porque a inversa da matriz agora consiste na inversão das variâncias e o determinante da matriz é o resultado da multiplicação das variâncias. Além disso, os casos de singularidades são mais facilmente tratáveis: basta impor um limite inferior para os valores das variâncias. A desvantagem dessa suposição é a falta de habilidade para tratar corretamente características correlacionadas e, possivelmente, uma quantidade maior de neurônios para modelar uma distribuição com características correlacionadas (Magdon-Ismail e Purnell, 2010; Kühne et al., 2008; Reynolds e Rose, 1995). Em (Magdon-Ismail e Purnell, 2010; Kühne et al., 2008) são mostrados exemplos nos quais um número pequeno de Gaussianas com matriz de covariância completa obtém uma qualidade de resposta mais elevada do que quando os problemas são tratados por um número maior de Gaussianas com matriz diagonal. Entretanto, os mesmos autores afirmam que o alto custo associado à matriz de covariância completa (tanto para armazenamento quanto para atualização) pode ser proibitivo para determinados problemas, e, além disso, ela pode guiar para *overfitting* dos dados.

A forma como os parâmetros da função $f(x,\mu,\Sigma)$ da Equação 4.2 são estimados é outro ponto a ser realçado. Na sua forma clássica, eles são estimados reutilizando várias vezes um conjunto de dados, de modo a se obter estimativa tão precisa quanto possível dos parâmetros. Por outro lado, se esses parâmetros forem estimados de forma incremental, surge uma dificuldade, pois, inicialmente é usada pouca quantidade de dados, e as estimativas serão um tanto imprecisas. Essas imprecisões serão piores quanto maior for a diferença entre o valor real do parâmetro e o valor inicialmente usado.

Essas imprecisões podem prejudicar o desempenho do modelo devido à propagação de erros (incertezas) (Bevington e Robinson, 2003). Para problemas com poucos atributos (valores pequenos de d), tais erros talvez não causem muita interferência, porque a propagação deles no modelo é pequena e pode assim ser desprezada. No entanto, o efeito em problemas com muitos atributos é mais notável e pode ser catastrófico. Além disso, o determinante de uma matriz de covariância tende a zero (ou infinito) à medida que o valor de d aumenta (Lee e Landgrebe, 1993), e representar valores tão baixos (ou altos) pode ser um problema computacional.

Para ilustrar o mencionado acima, suponha uma matriz de covariância Σ diagonal de tamanho $d \times d$, em que as d variâncias estimadas de Σ sejam de igual valor $\sigma^2 = \overline{\sigma^2} \pm \varepsilon$, onde $\overline{\sigma^2}$ é o valor real da variância e ε é o erro associado à estimativa. Se for considerado que o erro da estimativa é em função da variância $\varepsilon = \xi\overline{\sigma^2}$, onde ξ é uma constante, o determinante de Σ é obtido:

$$|\Sigma| = \sigma^{2d} = \left(\overline{\sigma^2} \pm \varepsilon\right)^d = \overline{\sigma^{2d}}\,(1 \pm \xi)^d = |\overline{\Sigma}|\,(1 \pm \xi)^d\,, \tag{4.4}$$

onde $|\overline{\Sigma}|$ é o valor real do determinante. Para um caso hipotético em que $\varepsilon = 0,01\overline{\sigma^2}$ e

$d = 100$, o valor de $|\Sigma|$ pode variar entre $0,366|\overline{\Sigma}|$ a $2,705|\overline{\Sigma}|$. Para $d = 500$, esse valor pode variar entre $0,007|\overline{\Sigma}|$ a $144,773|\overline{\Sigma}|$. Observa-se que, com o aumento do valor d, a escala de valores que o determinante pode assumir cresce muito se os valores das variâncias não forem estimados com precisão.

Para obter um compromisso satisfatório entre eficácia e eficiência do modelo, e reduzir o efeito da imprecisão dos valores estimados, esse trabalho adota algumas estratégias. A primeira é a suposição dos atributos serem estatisticamente independentes; a razão para essa simplificação foi a explicada acima.

Na segunda estratégia, o determinante da matriz de covariância Σ é substituído por um novo parâmetro, chamado tamanho de campo receptivo φ^2, com o objetivo de minimizar a influência das incertezas contidas na matriz de covariância Σ. O tamanho do campo receptivo determina o grau de extensão de cada neurônio. Pequenos valores indicam regiões mais estreitas do efeito do neurônio, porém, com um valor de ativação mais elevado. Para evitar variações bruscas do tamanho do campo receptivo φ^2 durante o treinamento da rede neural, seu valor é alterado com menor frequência e com ajustes menores do que os dos demais parâmetros da rede. A matriz Σ é mantida somente para normalizar a distância entre a amostra de entrada x e o vetor média μ. A Equação 4.5 indica essa mudança:

$$\hat{f}(x,\mu,\Sigma,\varphi) = \frac{1}{\sqrt{2\pi\varphi^2}} \exp\left[\frac{-(x-\mu)^T\Sigma^{-1}(x-\mu)}{2\varphi^2}\right]. \tag{4.5}$$

O termo $2\varphi^2$ foi adicionado na exponencial para garantir a propriedade das funções de distribuição de probabilidade que afirma que a probabilidade de todo o espaço amostral é igual a 1 (integral definida da função no intervalo de $-\infty$ a ∞).

A principal diferença entre a função de transferência da RNP (Equação 4.1) e a função da Equação 4.5 é a primeira considerar a distribuição das amostras uma hiperesfera (mesmo valor de variância para todas as dimensões). A segunda considera a possibilidade das amostras possuírem uma distribuição hiperelipsoidal no espaço de características (diferentes valores de variâncias para cada dimensão). Com essa diferença, a segunda equação tem uma possibilidade maior de modelar melhor a distribuição dos dados.

A rede neural proposta também deve ser capaz de se adaptar ao problema tratado por ela. Para isso, é interessante sua arquitetura poder se modificar de forma a armazenar melhor as informações do problema, podendo crescer ou diminuir conforme o necessário. A terceira estratégia é a utilização de três procedimentos para alterar a estrutura da rede: um de adição, um de união e um de remoção de neurônios. Esses mecanismos são utilizados sempre que uma nova amostra é utilizada para treinamento. Detalhes do modelo neural proposto nesse trabalho são explicados nas próximas subseções.

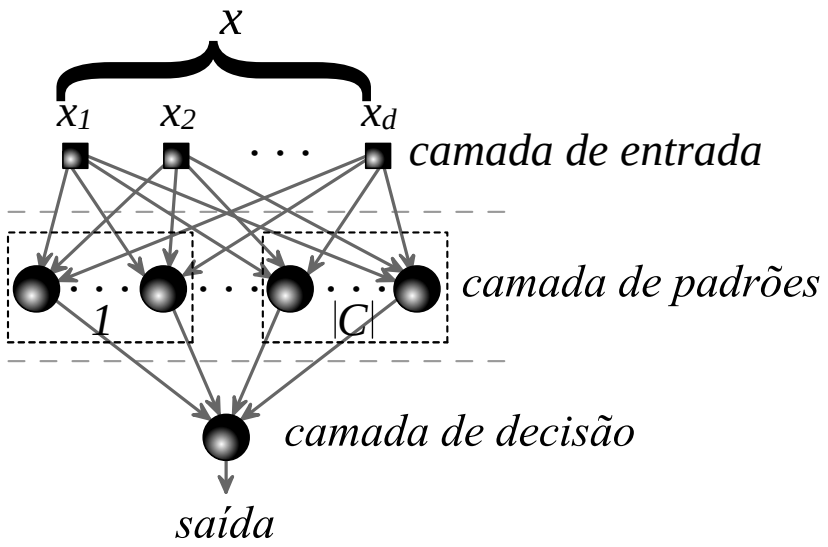


Figura 4.2: Arquitetura da RNPe, onde x é uma amostra com d dimensões a ser classificada, e saída é a classe associada pela rede à amostra x.

4.1.1 Treinamento da RNPe

A arquitetura da RNPe mostrada na Figura 4.2 é similar a arquitetura da RNP, com a diferença da RNPe não possuir a camada de soma. A razão para essa alteração na arquitetura é explicada na Seção 4.1.2. Para entender melhor o método de treinamento da RNPe, a camada de padrões de sua arquitetura é considerada dividida em $|C|$ partes, onde $|C|$ é o número de classes do problema em mãos. Embora a Figura 4.2 ilustre uma arquitetura com $|C|$ classes, esse número pode variar durante o tempo de vida da rede, permitindo acomodar mais classes ou eliminar aquelas sem uso. Esse processo é explicado com mais detalhes no final dessa seção.

Cada parte da camada de padrões é associada a uma única classe e contém um conjunto de neurônios. Cada parte pode ser vista como um MMG, e cada neurônio pode ser visto como um *kernel* no MMG. Um MMG aproxima a função de densidade de probabilidade (fdp) $p(x)$ de uma amostra x usando uma soma ponderada de funções Gaussianas (*kernels*), como indicada na Equação 4.6:

$$p(x) = \sum_{i=1}^{K} \omega_i f(x, \mu_i, \Sigma_i), \tag{4.6}$$

onde K é o número de *kernels* do MMG, ω_i é o peso do i-ésimo *kernel*, tal que $\sum_{i=1}^{K} \omega_i = 1$, $f(x, \mu_i, \Sigma_i)$ é a função Gaussiana da Equação 4.2 com vetor média μ_i e matriz de covariância Σ_i. As dimensões de μ_i e Σ_i são, respectivamente, $d \times 1$ e $d \times d$, onde d é o número de atributos (dimensões) da amostra x. A diferença do MMG usado na RNPe de um MMG

clássico é que, ao invés de usar a função Gaussiana da Equação 4.2, é usada a função da Equação 4.5.

O método para obter o MMG para cada classe envolve um conjunto de procedimentos realizados em sequência. Basicamente, esses procedimentos são (na ordem de ocorrência):

- classificação da amostra de treino: ela é passada pela rede, de tal forma que esta retorna uma classe para a amostra e a saída (ativação) de cada *kernel* para a amostra;
- atualização dos parâmetros dos *kernels*: os parâmetros dos *kernels* (ω, μ, Σ e φ^2) são atualizados para melhor ajuste da rede ao problema;
- adição de *kernel* ao MMG: de acordo com um critério utilizado é verificado se a adição de um *kernel* pode ser útil ao MMG;
- união de *kernels*: *kernels* próximos e apresentando aspectos semelhantes são unidos para reduzir o custo computacional e um tamanho desnecessário da rede;
- remoção de *kernels*: para algumas situações, podem existir *kernels* representando uma porção insignificante de informação, ou até mesmo ruídos embutidos nos dados. Esse procedimento tem como alvo eliminar esses *kernels* para diminuir o tamanho da rede, assim como reduzir a influência de *outliers* (amostras inconsistentes com a distribuição do conjunto de amostras) no desempenho.

Com exceção do procedimento de classificação, explicado numa subseção à parte (Seção 4.1.2), os outros são explicados a seguir. Além desses procedimentos, o processo para adicionar ou remover uma classe na rede também é explicado na Seção 4.1.1.

Atualização dos parâmetros

Para um melhor ajuste da rede é interessante fazer uma atualização dos parâmetros de cada *kernel*. Assim, é necessário fazer estimativas dos valores da média μ e variância σ^2 para todas as dimensões de cada *kernel*, e o peso ω e o tamanho do campo receptivo φ^2 para cada *kernel*.

Sempre que uma nova amostra x é usada para treinamento, os parâmetros dos *kernels* associados à mesma classe de x são atualizados, com exceção do tamanho do campo receptivo φ^2. Os valores da média μ, variância σ^2 e peso ω para cada *kernel* são estimados incrementalmente, usando um conjunto de equações obtido através do algoritmo de *Expectation Maximization*, como apresentado em (Vlassis et al., 1999). As Equações 4.7, 4.8 e 4.9 obtêm

esses valores para $j = 1, \ldots, d$ e $i = 1, \ldots, K$, onde d é o número de atributos (dimensões) do problema e K é o número de *kernels* do MMG associado à mesma classe de x:

$$\mu_{i,j,t+1} = \mu_{i,j,t} + \frac{P(i \mid x)(x_j - \mu_{i,j,t})}{n\omega_{i,t}}, \tag{4.7}$$

$$\sigma^2_{i,j,t+1} = \sigma^2_{i,j,t} + \frac{P(i \mid x)\left[(x_j - \mu_{i,j,t+1})^2 - \sigma^2_{i,j,t}\right]}{n\omega_{i,t}}, \tag{4.8}$$

$$\omega_{i,t+1} = \omega_{i,t} + \frac{P(i \mid x) - \omega_{i,t}}{n}, \tag{4.9}$$

onde

$$P(i \mid x) = \frac{\omega_{i,t}\,\hat{f}(x, \mu_{i,t}, \Sigma_{i,t}, \varphi_{i,t})}{p(x)}, \tag{4.10}$$

sendo os valores de $P(i \mid x)$, $i = 1, \ldots, K$, obtidos no passo de *Expectation*, e os valores da demais equações no passo de *Maximization* do algoritmo de *Expectation Maximization*. Os índices t e $t+1$ indicam os valores atuais e futuros das grandezas. A Equação 4.9 satisfaz a condição $\sum_{i=1}^{K} \omega_i = 1$. Uma característica importante do algoritmo de EM é a sua propriedade de convergência (quando satisfeitas as condições necessárias) para o ótimo global ou local, como provada em (Dempster et al., 1977; Bishop, 2006).

A constante n nas equações anteriores pode ser vista como procedimento de média móvel sobre as amostras de treinamento anteriores. Logo, as equações anteriores não buscam uma convergência estocástica (como o algoritmo tradicional de EM), mas modelar uma distribuição não estacionária dos dados (Vlassis et al., 1999). Como indicado por Vlassis et al. (1999), desde que não existam variações bruscas na dinâmica do problema, é possível obter uma certa convergência com as Equações 4.7 a 4.9. Diferentemente de Vlassis et al. (1999), que usam um valor constante para n, nesse trabalho o valor de n muda sempre que uma nova amostra é utilizada para treinamento. Essa mudança é para evitar que a primeira amostra de treinamento (que é explicitamente representada como um *kernel*) tenha um peso maior que as demais amostras, o que pode ser um problema se a primeira amostra for um *outlier*. Para alterar o valor de n, o número de amostras usadas para treinar cada MMG é contado. Seja n_{atual} o número atual de amostras para treinar uma MMG. Então, $n = \min(n_{atual}, n_{max})$, onde n_{max} é o valor máximo permitido para n e $\min(a, b)$ é uma função que retorna o menor valor entre a e b. Quanto menor o valor para n_{max}, mais rápidas são as mudanças nos parâmetros das Equações 4.7, 4.8 e 4.9. Para certos problemas analisados, foi observado que a atualização das variâncias reduzia o desempenho, pois não era possível estimar valores precisos para elas. Por isso, a atualização ou não das variâncias (no procedimento de atualização) foi adicionado como um parâmetro adicional para o modelo proposto.

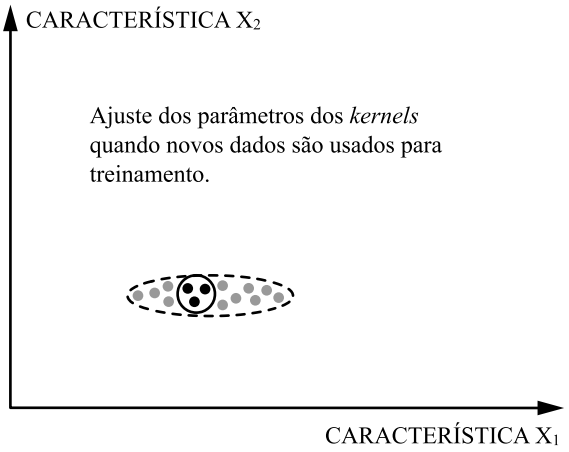


Figura 4.3: Na figura está ilustrado um *kernel* (elipse sólida) modelando a distribuição dos dados usados para treinamento (pontos escuros). No entanto, com o uso de mais dados de treinamento (pontos cinza) o *kernel* é ajustado e ganha novo formato (elipse tracejada).

A Figura 4.3 ilustra um exemplo do resultado da atualização dos parâmetros dos *kernels*. Sejam os pontos escuros as amostras usadas para treinamento e a elipse sólida os contornos de um *kernel*. À medida que novos dados são usados para treinamento (pontos cinza), o *kernel* é ajustado e obtém um novo formato (elipse tracejada).

O tamanho do campo receptivo de um *kernel* é alterado quando uma amostra x passa pela rede e é classificada em uma classe errada. Então, o campo receptivo do *kernel* r, que é o *kernel* com o maior valor de saída associado à classe correta, é ajustado para elevar a sua saída para a amostra x.

Para descobrir o valor de φ_r^2 que maximiza a Equação 4.5 do *kernel* r, a equação é derivada e são seguidos os mesmos procedimentos apresentados em (Vlassis et al., 1999) para obter uma versão incremental da estimativa de φ_r^2. Com isso, a Equação 4.11 é obtida:

$$\varphi_{r,t+1}^2 = \varphi_{r,t}^2 + \frac{P(r \mid x)\left[(x-\mu_{r,t+1})^T \Sigma_{r,t+1}^{-1}(x-\mu_{r,t+1}) - \varphi_{r,t}^2\right]}{n\omega_{r,t+1}}. \tag{4.11}$$

A Equação 4.11 indica que o valor de φ_r aumenta quando as amostras de uma determinada classe estão mais espalhadas, e o valor é reduzido quando as amostras estão mais concentradas em uma determinada região do espaço.

O problema da Equação 4.11 é o valor de φ_r^2 variar bruscamente devido aos valores iniciais de n, indo de 1 até o valor n_{max}. Para evitar isso, é usada uma função linear saturada (cujo valor varia linearmente em um intervalo, e permanece constante fora desse intervalo) para limitar o valor do termo à direita da operação de soma e, assim, limitar o ajuste de φ_r^2 por amostra. Seja esse valor igual a λ, então ele é limitado na faixa $-\eta\varphi_r^2 \leq \lambda \leq \eta\varphi_r^2$, onde

η é uma constante. Quanto maior o valor de η, mais rápidas são as alterações em φ_r^2. Porém, valores elevados (próximos de 1) devem ser evitados para impedir uma hipersensibilidade da rede a amostras de treinamento. Para η igual a zero, φ_r^2 não é alterado. Outra alteração realizada foi a divisão do valor da distância de Mahalanobis ($(x-\mu_r)^T\Sigma_r^{-1}(x-\mu_r)$) pelo número de atributos d, já que a tendência é a escala de valores dessa distância crescer em função da quantidade de atributos. A Equação 4.12 mostra essa alteração.

$$\varphi_{r,t+1}^2 = \varphi_r^2 + \frac{P(r \mid x)\left\{\left[(x-\mu_{r,t+1})^T\Sigma_{r,t+1}^{-1}(x-\mu_{r,t+1})\right]/d - \varphi_r^2\right\}}{n\omega_{r,t+1}}. \tag{4.12}$$

Adição, união e remoção de *kernels*

Três métodos são aplicados para estimar o número de *kernels* no MMG: um, para verificar se é necessário adicionar um novo *kernel* no modelo; outro, para determinar se é conveniente unir dois *kernels*; e, um último, sugerindo a remoção de *kernels* contribuindo com pouca informação à rede.

O primeiro passo é determinar se um novo *kernel* deve ser adicionado ao MMG. Para isso, é utilizado um critério simples: se a amostra de treinamento não for corretamente classificada, ela é inserida como um *kernel* no modelo. Apesar de sua simplicidade, esse esquema permite que a rede não cresça na mesma proporção que a RNP, além de permitir à rede capturar informações diferentes daquelas já embutidas nela mesma. Esse esquema também foi escolhido com o objetivo de reduzir o número de parâmetros a serem calibrados para a rede, pois, em algumas abordagens, é comum utilizar medidas de distâncias ou procedimentos de divisão para crescer a estrutura do modelo e, assim, é necessário calibrar um ou mais parâmetros.

Supondo a amostra x não ter sido corretamente classificada, então é inserido um novo *kernel* u no MMG, cujos parâmetros para cada dimensão j ($j = 1,\ldots,d$) são obtidos como segue:

$$\mu_{u,j} = x_j, \tag{4.13}$$

$$\sigma_{u,j}^2 = \frac{\sum_{i=1}^{K}\omega_i\sigma_{i,j}^2}{\sum_{i=1}^{K}\omega_i}, \tag{4.14}$$

$$\varphi_u^2 = \frac{\sum_{i=1}^{K}\omega_i\varphi_i^2}{\sum_{i=1}^{K}\omega_i}, \tag{4.15}$$

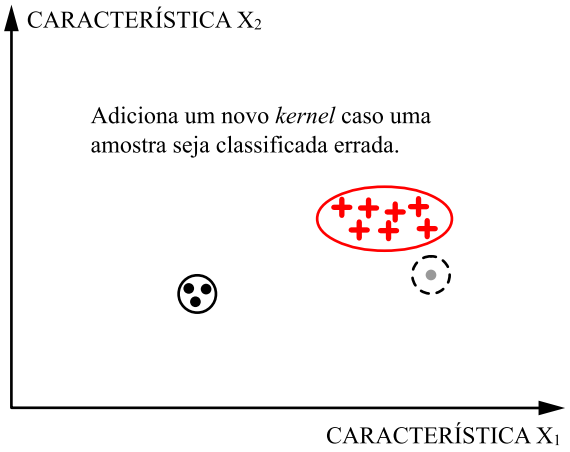


Figura 4.4: Na figura, a nova amostra de treinamento (ponto cinza) é erroneamente classificada como cruz por estar mais próxima ao *kernel* da classe cruz do que da classe ponto. Para corrigir isso, é adicionado um *kernel* (círculo tracejado) com as coordenadas da nova amostra de treinamento.

$$\omega_u = \frac{1}{n}, \tag{4.16}$$

onde K é o número de *kernels* no MMG. Para o cálculo da variância $\sigma^2_{u,j}$ e do campo receptivo φ^2_u é suposto esses valores não variarem muito entre os *kernels* já existentes no MMG, tal que a média ponderada desses valores retorne uma boa estimativa inicial para os parâmetros do *kernel* u. Para garantir $\sum_{i=1}^{K} \omega_i = 1$, o peso ω de cada *kernel* no MMG é recalculado:

$$\omega_i = \frac{(n-1)\omega_i}{n} \quad \text{para } i = 1,\ldots,K \quad i \neq u. \tag{4.17}$$

Uma ilustração do procedimento de adição de neurônios é mostrada na Figura 4.4. Nessa figura são ilustrados os *kernels* (elipses sólidas) de duas classes distintas: uma de pontos escuros e outra de cruzes vermelhas. Quando uma nova amostra da classe dos pontos (ponto cinza) é usada para treinamento, ela é erroneamente classificada como cruz. Para corrigir isso, é adicionado um *kernel* (círculo tracejado) com as coordenadas da nova amostra de treinamento.

O próximo passo é checar se dois *kernels* podem ser unidos e, assim, reduzir a complexidade da rede. Esse é um procedimento muito importante para manter a rede com uma arquitetura reduzida e assim diminuir seu custo computacional e uso de memória.

Ao longo do aprendizado incremental, dois ou mais *kernels* podem se aproximar de tal forma a serem unidos sem perda de informação relevante. Sempre que uma nova amostra de treinamento é usada, a possibilidade de união de dois *kernels* do mesmo MMG é avaliada pela rede. Uma vez que a amostra x tenha passado pela rede, os *kernels* com a maior saída

(p) e com a segunda maior saída (q) (e que estão associados a mesma classe de x) são selecionados para passarem pelos critérios de união de *kernels*. Essa seleção parte do seguinte princípio: se ambos apresentaram uma saída elevada para x, isso indica que eles devem estar próximos entre si. Embora essa suposição não seja a melhor, no sentido de procurar os *kernels* mais próximos, ela apresenta um compromisso razoável entre eficiência e eficácia.

Dois métodos propostos em (Lughofer, 2011) são usados para verificar se os *kernels* p e q podem ser unidos. As condições para unir dois *kernels* parte da consideração deles estarem próximos e seus contornos terem estruturas homogêneas.

Caso os dois *kernels* sejam esféricos, a confirmação sobre uma possível sobreposição, ou tangenciamento entre os dois, é obtida quando a distância entre os centros das esferas é menor ou igual à soma dos seus raios:

$$d(c_p, c_q) \leq r_p + r_q, \tag{4.18}$$

onde $d(c_p, c_q)$ é a distância Euclidiana entre c_p e c_q, c_p e c_q são os centros das esferas, e r_p e r_q são os raios das esferas.

No caso dos *kernels* p e q possuírem um formato elipsoidal, a condição acima pode ser expandida para considerar os diferentes comprimentos dos eixos em cada dimensão, aproximado pelo espalhamento do desvio padrão σ dos *kernels*:

$$d(\mu_p, \mu_q) \leq \frac{\sum_{j=1}^{d} |\mu_{p,j} - \mu_{q,j}|(\sigma_{p,j}\, fac + \sigma_{q,j}\, fac)}{\sum_{j=1}^{d} |\mu_{p,j} - \mu_{q,j}|} + \sqrt{d}\upsilon. \tag{4.19}$$

O fator fac representa o espalhamento do *kernel* que é usado para o critério de sobreposição e de contato. Um $fac = 2$ cobre 96% das amostras que caem dentro do *kernel*, se as mesmas possuírem distribuição Gaussiana (Lughofer, 2011). Para o caso de *kernels* esféricos, $\sigma_{p,1} = \sigma_{p,2} = \ldots = \sigma_{p,d} = r_p/2$, retornando assim para a Equação 4.18. O símbolo υ é uma constante representando o grau de sobreposição necessário para permitir a união de dois *kernels*.

Valores positivos de υ permitem a união de *kernels* que não estão se tocando. Valores negativos de υ exigem os *kernels* mais próximos para ocorrer a fusão. A multiplicação por $\sqrt{d}$ foi adicionada na equação original para ajustar o valor de υ em problemas com diferentes dimensões (número de atributos).

Se a primeira condição for satisfeita, é verificada a segunda. Essa condição verifica se a união dos dois *kernels* forma uma região homogênea. Basicamente, quanto maior o

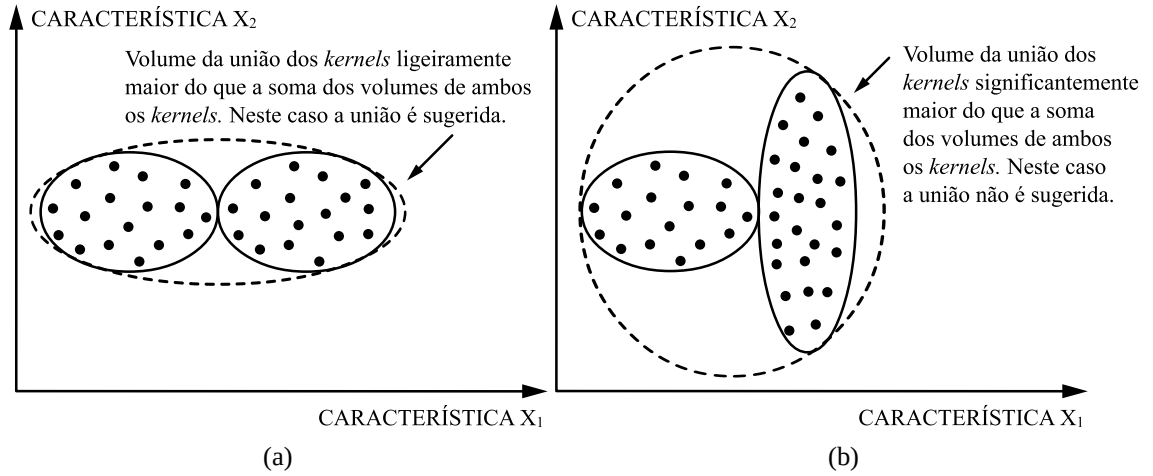


Figura 4.5: Em (a) existem dois *kernels* (elipses sólidas) se tocando e homogêneos, no sentindo do *kernel* resultante da união (elipse tracejada) possuir aspecto semelhante aos dois *kernels* originais. Em (b) os dois *kernels* (elipses sólidas) se tocando não são homogêneos (o *kernel* resultante é muito maior que o volume total dos dois *kernels*)(Lughofer, 2011).

volume do *kernel* resultante em relação à soma dos volumes dos *kernels* originais, menor a homogeneidade. Exemplos de níveis de homogeneidade são mostrados na Figura 4.5. Enquanto, na Figura 4.5(a), o *kernel* maior tem um aspecto semelhante à união dos dois *kernels*, na Figura 4.5(b) é mostrado um *kernel* resultante significativamente maior que a soma dos *kernels* originais.

Portanto, a condição adicional para verificar se o volume do *kernel* resultante t tem volume similar à soma dos volumes dos *kernels* p e q é obtida por:

$$V_t \leq d(V_p + V_q), \tag{4.20}$$

na qual V é o volume de uma elipsoide (Jimenez e Landgrebe, 1998):

$$V = \frac{2\prod_{j=1}^{d}\sigma_j \pi^{d/2}}{d\Gamma(d/2)}, \tag{4.21}$$

com a função Γ definida por:

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt. \tag{4.22}$$

A multiplicação por d na Equação 4.20 é devido ao efeito da maldição da dimensionalidade no cálculo do volume (Lughofer, 2011).

Se as duas condições forem satisfeitas, os *kernels* p e q podem ser unidos para formar o *kernel* t. Os parâmetros do novo *kernel* t são obtidos como segue ($j = 1, \ldots, d$) (Lughofer, 2011):

$$\mu_{t,j} = \frac{\omega_p \mu_{p,j} + \omega_q \mu_{q,j}}{\omega_p + \omega_q}, \tag{4.23}$$

$$\sigma^2_{t,j} = \frac{\omega_p \sigma^2_{p,j}}{\omega_p + \omega_q} + (\mu_{p,j} - \mu_{t,j})^2 + \frac{(\mu_{q,j} - \mu_{t,j})^2}{\omega_p + \omega_q} + \frac{\omega_q \sigma^2_{q,j}}{\omega_p + \omega_q}, \tag{4.24}$$

$$\varphi^2_t = \frac{\omega_p \varphi^2_p + \omega_q \varphi^2_q}{\omega_p + \omega_q}, \tag{4.25}$$

$$\omega_t = \omega_p + \omega_q, \tag{4.26}$$

Depois de criar o *kernel* t, p e q são removidos da rede.

O último passo de alteração na arquitetura da rede é a remoção de *kernels*. Dois procedimentos simples podem ser realizados nessa etapa. Um é definir o número máximo de *kernels* que cada MMG poderá ter. Assim, se o número de *kernels* ultrapassar o valor definido, os *kernels* com os menores valores de ω são removidos. Isto evita o crescimento da rede acima do necessário, eliminando *kernels* que contribuem com pouca informação ao modelo.

O outro procedimento é uma forma simples para tentar remover *outliers* e ruídos da rede, sem eliminar informação recentemente adicionada à rede. Para tanto, ele parte da suposição que um *outlier* possui distribuição aleatória diferente da distribuição dos exemplos de uma determinada classe, e as chances de outra amostra ou *outlier* cair em regiões próximas são baixas. Então, mesmo se um *outilier* for convertido em um *kernel*, esse *kernel* terá seu peso ω reduzido na proporção de novos dados de treinamento serem utilizados. Por outro lado, uma amostra que indica uma tendência de deslocamento espacial da distribuição da classe tenderá a ter o seu peso ω reforçado, caso ela se converta em um *kernel*. Então, para esse procedimento, é usado um limiar $1/(n+\tau)$ sobre os pesos ω dos *kernels*, onde $1/n$ é o peso inicial de cada novo *kernel* inserido e $\tau \geq 1$. *Kernels* com pesos abaixo desse limiar são removidos. Quanto mais elevado o valor de τ, maior o número de iterações para remover um *kernel*, desde que este tenha o valor do seu peso diminuído. A Figura 4.6 ilustra um caso de *kernels* removidos do MMG.

Quando os *kernels* são removidos do MMG, é necessário recalcular os pesos dos outros *kernels*, tal que a soma dos pesos ω dos *kernels* restantes seja igual a um. Para alcançar isso, os pesos dos *kernels* restantes são normalizados pela norma L_1 (Duda et al., 2001).

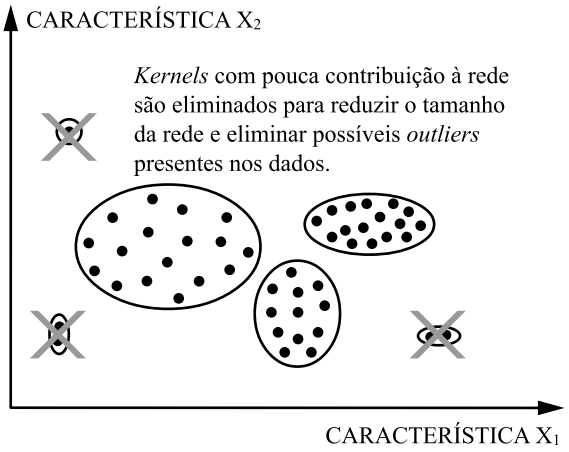


Figura 4.6: Na figura são ilustrados alguns *kernels* modelando a distribuição dos dados de uma classe. Porém, *kernels* representando uma quantidade irrisória dos dados são mais susceptíveis a serem excluídos do modelo.

Adicionando e removendo classes da rede

No início do treinamento, a rede neural está vazia e não tem neurônios em sua arquitetura. Quando uma classe desconhecida aparece durante o treinamento, um novo MMG é adicionado à arquitetura da rede neural. Dessa forma, cada MMG é associado a uma única classe. Toda vez que uma amostra é usada para treinar a RNPe, é verificado se ela é representada por um MMG existente ou requer a construção de um novo MMG para representar uma nova classe. Assim, o número de classes na rede neural varia de zero a $|C|$ durante a fase de treinamento, onde $|C|$ é o número de classes do problema em mãos.

O primeiro *kernel* do primeiro MMG adicionado na rede neural representa a primeira amostra de treinamento x. Esse *kernel* tem os seguintes valores de parâmetros:

$$\mu_{1,j} = x_j, \;\; \sigma^2_{1,j} = 1, \;\; \varphi^2_1 = \varphi_{ini}, \;\; \omega_1 = 1, \tag{4.27}$$

onde φ_{ini} é o valor inicial do campo receptivo φ^2_1. O valor de φ_{ini} deve ser grande o suficiente para evitar singularidades.

Se já existir outros MMGs, e for necessário adicionar um novo (quando há necessidade de se adicionar uma nova classe à rede neural), os parâmetros do primeiro *kernel* nesse MMG são obtidos das Equações 4.13, 4.14 e 4.15, substituindo K por K_T, onde K_T é o número total de *kernels* na rede. O peso desse novo *kernel* é $\omega_1 = 1$. Como mencionado anteriormente, os cálculos das Equações 4.14 e 4.15 assumem as variâncias e o campo receptivo do *kernel* da nova classe como uma média ponderada dos valores existentes na rede. Entretanto, mesmo se essa suposição não for realista, o valor inicial de n para o novo MMG é 1, o que permite uma rápida adaptação desses valores com a chegada de mais dados de treinamento.

Algoritmo *Procedimento de aprendizado da Rede Neural Probabilística evolutiva*

para cada nova amostra de treinamento **faça**
    se a amostra não é de nenhuma classe existente na rede neural **então**
        adiciona à rede um MMG associado à mesma classe da amostra. É adicionado um *kernel* nesse novo MMG.
        se é o primeiro *kernel* da rede **então**
            os parâmetros do *kernel* são obtidos da Equação 4.27.
        senão
            os parâmetros do *kernel* são obtidos das Equações 4.13 a 4.15 e $\omega_1 = 1$.
    senão
        passa a amostra através da rede neural;
        atualiza os *kernels* do MMG associado à mesma classe da amostra usando as Equações 4.7 a 4.9;
        se a rede associou a classe errada para a amostra **então**
            atualiza o tamanho do campo receptivo do *kernel* mais ativado associado à mesma classe da amostra aplicando a Equação 4.12;
            adiciona um *kernel* ao MMG da mesma classe da amostra, cujos parâmetros são obtidos das Equações 4.13 a 4.16;
        checa os dois *kernels* mais ativos do MMG associado à mesma classe da amostra;
        realiza os testes de proximidade (Equação 4.19) e de homogeneidade (Equação 4.20);
        se ambos os testes forem bem sucedidos **então**
            une esses dois *kernels* em um novo, usando as Equações 4.23 a 4.26 e remove os dois *kernels* originais;
        remove os *kernels* com peso ω abaixo do limiar $1/(n+\tau)$ ou aqueles excedentes ao número máximo de *kernels* permitidos para o MMG.

Algoritmo 6: Procedimento de aprendizado da Rede Neural Probabilística evolutiva.

O processo de eliminação de classes é simples: basta retirar da rede o MMG associado à classe que se deseja remover. O Algoritmo 6 ilustra os passos utilizados para treinar a RNPe apresentada nesse trabalho.

4.1.2 Procedimento de classificação

Classificadores baseados em MMG costumam realizar uma soma ponderada das contribuições de cada *kernel*, e a classe do MMG com a maior saída é associada à amostra. A desvantagem dessa abordagem é os *kernels* com os maiores pesos serem os mais decisivos na classificação da amostra, sobrando para os demais *kernels* um poder menor para influenciar nos resultados.

A Figura 4.7 ilustra um exemplo de uma amostra desconhecida pela rede, representada por um ponto cinza (e rotulada como ponto), devendo ser classificada. Embora esteja mais próxima de um *kernel* da classe ponto, existe probabilidade maior da amostra ser associada à classe cruzes, pois está muito distante do *kernel* da classe ponto com maior peso (representada pela elipse mais à esquerda com o maior número de pontos internos). Mesmo se calculada uma média simples, há grande probabilidade de ser associada à classe representada pelas cruzes, pois dois dos três *kernels* estão distantes da amostra.

Então, para evitar essa situação, e inspirado em algumas redes neurais incrementais, o

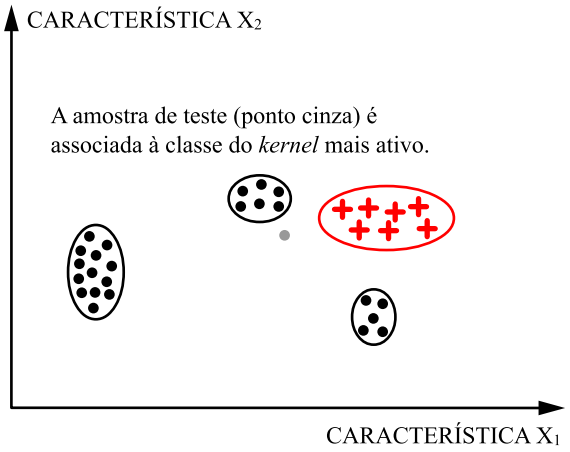


Figura 4.7: Na figura são ilustrados alguns *kernels* modelando a distribuição dos dados de duas classes e uma amostra (ponto cinza) da classe ponto a ser classificada. Embora esteja próxima de um *kernel* da classe ponto, há uma maior probabilidade da amostra ser associada à classe cruz se for usada uma soma ponderada das ativações dos *kernels* para classificação. Para evitar tal situação, é utilizado o esquema de associar a classe do *kernel* mais ativo à amostra.

esquema de classificação adotado é o de associar a cada amostra a classe do *kernel* mais ativo. Outra razão para esta mudança é os *kernels* adicionados recentemente possuírem peso inicial pequeno, enquanto os mais antigos, pesos maiores. Assim, uma soma ponderada dos *kernels* pode privilegiar os mais antigos, tornando mais lenta a adaptação da rede aos novos dados.

Com essa mudança, a saída da rede $O(x)$ para a amostra x é a classe associada ao neurônio mais ativo da rede:

$$O(x) = \arg\max_{i=1}^{|C|} P(c_i|x), \tag{4.28}$$

onde:

$$P(c_i|x) = \max_{j=1}^{K_i} \hat{f}(x, \mu_{i,j}, \Sigma_{i,j}, \varphi_{i,j}), \tag{4.29}$$

sendo $|C|$ o número de classes e K_i o número de neurônios associados à classe c_i ($c_i \in C$, onde C é o conjunto de classes do problema). Os parâmetros $\mu i, j$, $\Sigma_{i,j}$ e $\varphi_{i,j}$ são o vetor média, a matriz de covariância e o efeito de campo do j-ésimo neurônio da classe c_i, respectivamente. Com essa modificação, não há necessidade da camada de soma existente na arquitetura da RNP, sendo removida da rede neural proposta.

4.1.3 Efeito dos Parâmetros de Calibração da Rede

O modelo proposto possui alguns parâmetros cujos valores precisam ser ajustados. Cada parâmetro afeta de forma diferente o comportamento da rede neural proposta, e essa seção tem como objetivo descrever o efeito que cada parâmetro tem sobre a rede. Os parâmetros que necessitam ser calibrados são:

- n_{max}: o valor de n_max afeta a velocidade em que os parâmetros dos neurônios são ajustados na etapa de atualização. A etapa de atualização atua como um procedimento de média móvel, e o valor de n_{max} determina o tamanho da janela móvel sobre as amostras. Quanto menor o valor de n_{max}, mais rápido os parâmetros são alterados. Valores baixos de n_{max} ($n_{max} < 100$) devem ser evitados com o objetivo de obter melhor convergência na atualização (Vlassis et al., 1999). O valor de n_{max} está relacionado com a dinâmica do problema a ser tratado. Problemas com características estacionárias podem ser melhores tratados com o valor de $n_{max} \rightarrow \infty$, já que todas as amostras contribuem com a mesma quantidade de informação. O valor de n_{max} pode assumir qualquer valor real positivo, mas, de preferência, que não seja menor do que 100.

- φ_{init}: o valor φ_{init} determina a abrangência inicial do efeito de cada neurônio. Quanto maior o valor de φ_{init}, mais abrangente a região de efeito do neurônio. No entanto, quanto maior o valor de φ_{init}, menor o valor de máxima ativação (saída) do neurônio. O valor desse parâmetro tem maior efeito sobre a rede quando o campo receptivo é ajustado (efeito determinado pelo parâmetro η). Caso contrário, o seu efeito é reduzido, pois os campos receptivos de todos os neurônios serão iguais ao longo do treinamento do modelo. O valor de φ_{init} pode assumir qualquer valor real maior que zero, mas valores extremamente pequenos devem ser evitados para evitar singularidades.

- η: esse parâmetro determina quanto o valor do campo receptivo φ^2 é ajustado. Quanto maior o valor de η, mais rápido o valor do campo receptivo é ajustado. No entanto, devem ser usados valores baixos de η ($\eta << 1$) para evitar instabilidade no aprendizado. Para $\eta = 0$, não ocorre atualização do campo receptivo φ^2. O valor de η pode assumir qualquer valor real maior ou igual a zero.

- υ: o valor do parâmetro υ determina com que frequência ocorrerá a união de neurônios, além de determinar indiretamente o tamanho da rede. O valor de υ pode assumir qualquer valor real (tanto positivo quanto negativo). Valores positivos elevados de υ fazem com que o procedimento de união de neurônios ocorra com menos frequência, e a rede tende a crescer o tamanho de sua arquitetura. Valores negativos permitem uma maior estabilidade no tamanho da rede, pois o procedimento de união de neurônios ocorre com mais frequência. Para problemas onde a separação entre as classes

é simples, um valor baixo de υ pode obter uma estrutura reduzida com desempenho aceitável.

- τ: o valor de τ determina a eliminação de neurônios da rede. Valores baixos realizam o procedimento de poda de neurônios com mais frequência, o que permite uma estrutura da rede neural mais reduzida. Entretanto, a eliminação de neurônios pode acarretar em uma queda no desempenho. O valor de τ pode assumir qualquer valor real positivo. Para $\tau \rightarrow \infty$ não ocorre eliminação de neurônios.

Além desses parâmetros, foi verificado que o ajuste das variâncias no procedimento de atualização pode prejudicar o desempenho do modelo, principalmente em problemas com grande de atributos (valor de d elevado), pois pode elevar o efeito das estimativas imprecisas dos parâmetros. Para evitar esse tipo de problema, o ajuste das variâncias no procedimento de atualização pode ser ignorado.

4.1.4 Treinamento com uma Base de Dados Inicial

A versão incremental proposta da RNP apresenta a vantagem de ter um treinamento rápido, poder adquirir novo conhecimento conforme o fluxo de dados e não é necessário definir o tamanho de sua arquitetura. No entanto, frequentemente, existe disponível algum conjunto inicial de dados que, embora possa não ser grande, pode servir para obter estimativas mais confiáveis dos parâmetros da rede neural. Alguns modelos, inclusive algumas redes neurais incrementais, podem utilizar várias vezes um conjunto de dados como treinamento com o objetivo de maximizar a qualidade do mesmo. Embora procedimento semelhante também possa ser efetuado na técnica proposta, existe outro esquema aplicável para obter uma melhor estimativa dos parâmetros do modelo.

O outro procedimento consiste em estimar os valores iniciais das variâncias e médias de cada classe e inserir essas informações na forma de *kernels* dentro da rede neural. Uma forma de estimar esses valores é através de um procedimento incremental, com a vantagem de não precisar carregar todos os dados de uma vez na memória de um computador, e continuar a estimativa da onde parou sempre que novos dados forem disponíveis para treinamento.

O algoritmo 7 indica o procedimento a ser seguido, utilizando as Equações 4.30 e 4.31, onde t e $t+1$ são os valores atuais e futuros das grandezas, respectivamente, n_{atual} indica a quantidade atual de amostras usadas na estimativa e d é o número de atributos das amostras, sendo $j = 1, \ldots, d$.

$$\mu_{j,t+1} = \mu_{j,t} + \frac{x_{n,j} - \mu_{j,t}}{n_{atual}}, \tag{4.30}$$

Algoritmo *Algoritmo para calcular incrementalmente a média e variância de uma classe c_i*

1 Inicializar os parâmetros: $n_{atual} = 1$, $t = 1$, $\mu_{j,1} = x_{1,j}$ e $\sigma^2_{j,1} = 1$
2 **faça**
3 $n_{atual} = n_{atual} + 1$
4 Calcula a Equação 4.30
5 Calcula a Equação 4.31
6 $t = t + 1$
7 **enquanto** houver amostras da classe c_i

Algoritmo 7: Algoritmo para calcular incrementalmente média e variância.

$$\sigma^2_{j,t+1} = \sigma^2_{j,t} + (\mu_{j,t+1} - \mu_{j,t})^2 + \frac{(\mu_{j,t+1} - x_{n,j})^2 - \sigma^2_{j,t}}{n_{atual} - 1}, \tag{4.31}$$

Com os vetores de média e de variância de cada classe calculados, eles são inseridos na rede neural na forma de *kernel*, sendo o peso ω de cada *kernel* igual a 1, e o valor n de cada MMG é $n = \min(n_{atual}, n_{max})$. Assim, a rede neural não começa com a arquitetura vazia e o Algoritmo 6 pode ser aplicado para continuar o aprendizado da RNPe. A desvantagem desse procedimento é que, se a distribuição das amostras das classes não se aproximar de uma Gaussiana, os valores estimados podem não coincidir com a realidade e o desempenho da rede pode ser menor do que o de uma rede treinada com uma única passada dos dados.

4.2 Comparação com os SCEs e Algumas Outras Técnicas de Aprendizado Incremental

O algoritmo da RNPe apresenta muitos aspectos similares ao conjunto de critérios propostos por Kasabov para um SCE (Watts, 2009). Alguns deles são: a adição de novos neurônios na rede ser baseada em exemplos de treinamento com certa quantidade de informação nova à rede, e os neurônios adicionados representando explicitamente os próprios exemplos. Se um neurônio não é adicionado, então os parâmetros dos neurônios existentes são ajustados de forma a aperfeiçoar o desempenho da rede para o atual exemplo de treinamento. É utilizada uma função baseada em distância para calcular a ativação dos neurônios de saída da camada evolutiva. No caso da RNPe, a camada de padrões.

Ambos os procedimentos de tais algoritmos são baseados na ideia de um único passo, isto é, a amostra só é apresentada uma vez à rede. Inicialmente, as redes neurais começam sem neurônios na camada evolutiva, e o primeiro neurônio representará explicitamente o primeiro exemplo de treinamento usado. Possuem procedimento de poda de neurônios baseado na relevância destes para a rede neural. As redes neurais da família dos SCEs e a RNPe são capazes de tratar problemas de classificação não lineares.

A principal diferença entre as duas abordagens está na rede aqui proposta utilizar uma função baseada na Gaussiana na camada de padrões para representar a região do espaço de entrada, sendo essa região definida pelos parâmetros da Gaussiana. No SCE, cada neurônio define um ponto no espaço, sendo que esse ponto é representado pelos pesos das conexões dos neurônios. Em adicional, no SCE é atualizado somente o neurônio mais ativo, enquanto que na rede proposta são atualizados os neurônios associados à classe da amostra de treinamento.

As redes eMLP, EFuNN e ECF são exemplos de redes neurais consideradas por Kasabov como da família dos SCEs, e apresentam as características mencionadas acima. eMLP, EFuNN e ECF possuem a característica de armazenar em suas arquiteturas amostras do problema, cuja quantidade varia de acordo com a complexidade da tarefa. Isto ajuda tais técnicas a preservar conhecimento antigo, com a desvantagem de uma arquitetura maior. Embora a RNPe também apresente uma característica semelhante de adicionar as amostras, o tamanho de sua arquitetura tende a ser menor, já que a RNPe busca encontrar um modelo para a fonte dos dados, e não simplesmente armazenar amostras do problema. Contudo, essa característica apresenta uma desvantagem: para problemas em que os contornos de classe são muito complexos, as redes eMLP e EFuNN podem apresentar melhor desempenho, especialmente com respeito a capacidade de reter informação.

Por outro lado, pelo fato da RNPe obter um modelo da distribuição dos dados, seu desempenho sobre os dados utilizados para treinamento é semelhante ao obtido sobre os dados não vistos durante o treino. Já as técnicas eMLP e EFuNN obtêm um desempenho um tanto quanto inferior para dados nunca vistos, em relação ao desempenho alcançado sobre os dados de treinamento. Além disso, os experimentos realizados sugerem que os desempenhos das técnicas RNPe e eMLP são mais influenciados pelo poder discriminativo dos atributos, pois eles tendem a serem maiores para tarefas cujos atributos possuem um alto poder discriminativo. Por outro, o desempenho do EFuNN aparentou ser mais influenciado pela distribuição espacial das amostras e contornos de classe, o que pode ser ocasionado pela lógica *fuzzy*, sendo essa a principal diferença entre EFuNN e eMLP.

As redes RNPe e RNPI são baseadas na Rede Neural Probabilística (Specht, 1990), e cada neurônio adicionado às suas arquiteturas representa a própria amostra responsável por sua adição. Além disso, ambas usam funções de transferências baseadas na Gaussiana. Por outro lado, existem grandes diferenças entre essas técnicas: além dos neurônios da RNPe possuírem mais parâmetros do que os da RNPI, a RNPe também ajusta esses parâmetros durante o aprendizado incremental, diferentemente da RNPI.

A RNPI só possui o procedimento de adicionar neurônios, e nenhum mecanismo para controlar o tamanho da sua arquitetura. Além de contar com o procedimento de adicionar

neurônios, a RNPe pode também fundir ou eliminá-los graças a mecanismos que acontecem junto com a tarefa de aprendizado incremental, evitando assim um crescimento exagerado da sua arquitetura.

Como a RNPI armazena todas as amostras de treinamento em sua arquitetura, e não é influenciada pela ordem de apresentação das amostras, consegue reter mais informações antigas que a RNPe, ao preço de uma arquitetura significativamente maior. Entretanto, a RNPI apresenta um nível menor de adaptação a novos dados de treinamento. Uma característica observada nos experimentos é que a RNPI é mais influenciada pelos contornos de separação entre classes do que pelo poder discriminativo dos atributos, sendo a RNPe mais influenciada por esse último, como mencionado acima.

As redes RNPe e a RNPI-EM apresentam muitas semelhanças. Ambas são baseadas na Rede Neural Probabilística e possuem procedimentos de adicionar, remover e fundir neurônios. O procedimento de atualizar os neurônios é semelhante entre elas. Os neurônios de suas arquiteturas possuem grande gama de parâmetros, e são atualizados de modo incremental. De forma geral, essas duas técnicas tentam modelar a fonte de dados de uma tarefa, sendo o desempenho sobre os dados de treinamento semelhante ao obtido sobre os dados nunca vistos no treino. Muitos dos procedimentos utilizados na RNPe foram inspirados na RNPI-EM, embora não sejam executados da mesma forma.

Entretanto, existe uma série de diferenças entre esses dois métodos. Na RNPI-EM, a adição de neurônios é baseada na divisão dos neurônios existentes, usando como critério informação de alta ordem estatística. Assim, um novo neurônio não representa explicitamente uma amostra de treinamento, como no caso da RNPe.

A RNPI-EM tende a possuir uma arquitetura reduzida, mas com possibilidade de crescer repentinamente, pois cada neurônio nessa rede pode se dividir em vários outros para cada amostra de treinamento usada. Sendo assim, a RNPI-EM apresenta uma arquitetura que pode ser mais instável do que a da RNPe.

Os critérios utilizados para unir os neurônios na RNPI-EM são baseados em testes estatísticos que verificam se dois neurônios possuem médias e variâncias semelhantes. Embora esse método possa reduzir o número de neurônios redundantes, pode falhar em unir neurônios substancialmente sobrepostos, tais como aqueles com mesma variância, mas diferentes médias. O método usado na RNPe consegue tratar esses casos.

Além disso, a RNPI-EM utiliza como função de transferência a função Gaussiana e, assim, ela é mais sujeita a imprecisões do que a rede neural proposta, pelo motivo mencionado na Seção 4.1. Nos experimentos, foi observado que a RNPI-EM apresenta um desempenho inferior à RNPe para a maioria das bases de dados, embora obtenha uma arquitetura

bem compactada. Também foi notado que, assim como a RNPI, ela tem capacidade menor para se adaptar a novos dados do que a RNPe, e seu desempenho é mais influenciado pela distribuição espacial das amostras do que o desempenho da RNPe.

Em comum com a rede ZISC (Coghill et al., 2003), pode ser apontado ambas serem baseadas em funções de base radial, adicionam neurônios representando o próprio exemplo que o originou e podem eliminar neurônios. No entanto, ZISC não possui mecanismo de fundir neurônios que estejam próximos entre si, como no caso da RNPe, além de ser possível nenhum neurônio de saída ser ativado para uma amostra de teste.

Outras características do modelo apresentado são não diminuir a contribuição de novas amostras de treinamento, como acontece em abordagens como a de (Heinen e Engel, 2010), e nem reduzir sua habilidade de adicionar novos neurônios a zero, a exemplo da rede proposta em (Platt, 1991) e, assim, sua habilidade para aprender novas informações não é degradada com o tempo. Além disso, a RNPe não precisa reapresentar várias vezes uma mesma amostra de treinamento para adquirir conhecimento, como é feito nos métodos apresentados em (Shiotani et al., 1995; Coghill et al., 2003; Alpaydin, 1991).

4.3 Método de Aprendizado Semi-Supervisionado

Uma das propostas desse trabalho é o uso da RNPe para aprendizado semi-supervisionado. Desta maneira, seria possível ter um modelo cujos parâmetros possam ser ajustados com os dados inicialmente obtidos e capaz de aprender de forma incremental com a mínima interferência humana, graças ao procedimento de aprendizado semi-supervisionado. Ou, em outro caso, que possa ser remodelado sempre que uma quantidade relevante de dados for obtida.

O método a ser avaliado para aprendizado semi-supervisionado é baseado no co-treinamento, apresentado na Seção 2.2.2. Assim, ao invés de treinar uma única rede neural, será treinado um conjunto de redes neurais, formando um comitê de máquinas, no qual o método para a combinação das respostas individuais é o voto majoritário simples, e a saída do comitê é a classe mais votada pelas máquinas. Se a classe mais votada para uma amostra receber uma quantidade de votos acima de um limiar, essa amostra é usada para treinar as redes neurais que discordaram da saída do comitê. A classe associada a essa amostra é a determinada pelo comitê.

Por ser baseado em um comitê de máquinas, o método de co-treinamento apresenta maior robustez do que o de auto-treinamento. Um comitê de máquinas tem um potencial maior de apresentar resultados melhores em termos de desempenho do que uma única máquina. Os autores de (Hansen e Salamon, 1990) apresentaram justificativas teóricas para a combinação

de diferentes modelos neurais para o problema de classificação binária. Os autores apontaram que, se as taxas de erro de N modelos disponíveis forem independentemente distribuídas e menores que 50%, então a probabilidade da saída combinada por voto majoritário estar errada será menor do que a menor das taxas de erro dos classificadores atuando isoladamente. Experimentos realizados em (Hansen e Salamon, 1990) atestaram essa hipótese.

Uma questão fundamental relacionada a um comitê de classificadores é a diversidade. Para obter bons resultados, cada componente do comitê de máquinas deve apresentar um bom desempenho isoladamente. Contudo, o grupo deve apresentar alta dissimilaridade entre eles com relação aos padrões de erro individuais, de forma a ocorrer uma diversidade de respostas produzidas que possa contribuir na síntese de uma melhor hipótese. Caso não aconteça essa diversidade, a contribuição de cada componente será praticamente a mesma, e não irá compensar o custo computacional de se ter um conjunto de classificadores comportando-se de forma homogênea entre eles (Nascimento e Coelho, 2009; Kuncheva e Whitaker, 2000).

Existem abordagens para garantir a diversidade em comitês de máquinas, sendo uma delas obtida através da manipulação do conjunto de dados de treinamento, quando é apresentado somente parte do conjunto para cada classificador através de procedimento de reamostragem, garantindo assim a diversidade no conhecimento inserido nas máquinas (Nascimento e Coelho, 2009; Kuncheva e Whitaker, 2000; Dietterich, 2000). No procedimento de reamostragem é permitido uma mesma amostra ser usada para treinar diferentes classificadores. No entanto, não é garantido todas as amostras do conjunto de treinamento serem usadas.

O Algoritmo 8 descreve de forma sucinta os procedimentos usados para o treinamento semi-supervisionado da RNPe. Inicialmente, todas as redes do comitê são treinadas com subconjunto de amostras selecionadas por reamostragem da base de treinamento. Em seguida, para reduzir a influência das amostras não rotuladas, os valores dos parâmetros n de cada rede são maximizados para o valor n_{max}, para evitar que a contribuição de uma amostra não rotulada seja muito alta.

Com o treinamento de todas as redes, o processo de classificação é realizado. Para cada nova amostra classificada é associada a ela a classe mais votada pelo comitê de redes. Se a classe mais votada recebeu uma quantidade de votos maior ou igual a um limiar β, essa amostra é usada para treinar uma das redes discordantes do comitê. Essa rede é selecionada aleatoriamente. Caso a quantidade de votos seja menor, nenhum procedimento adicional é realizado.

Muitos dos métodos de aprendizado semi-supervisionados apresentados na literatura são baseados em, primeiro, rotular todas as amostras não rotuladas ou rotular blocos de amostras para depois usá-las para treinamento (Wu et al., 2000; Zhang e Rudnicky, 2006; Blum e Mitchell, 1998; Li e Sethi, 2006; Muslea et al., 2000). Já o método proposto realiza a

Algoritmo *Algoritmo de aprendizado semi-supervisionado para a RNPe*

1 **para** uma base de treinamento inicial **faça**
2 treinar N classificadores com subconjuntos independentes selecionados por reamostragem da base de treinamento.
3 fazer com que todos os parâmetros n de todas as redes sejam iguais a n_{max}.
4 **para cada** nova amostra de entrada **faça**
5 classificar a amostra usando os N classificadores.
6 combinar os resultados retornados pelos classificadores usando voto majoritário e associando para a amostra a classe mais votada.
7 **se** a classe mais votada recebeu um número de votos maior ou igual a um limiar β **então**
8 **se** houver classificadores que discordaram do resultado **então**
9 selecionar aleatoriamente um dos classificadores que discordaram do resultado.
10 retreinar esse classificador utilizando o procedimento ilustrado no Algoritmo 6, sendo usada amostra e a classe rotulada pelo comitê de classificadores.
11 **senão**
12 nenhum retreinamento é realizado.

Algoritmo 8: Algoritmo de aprendizado semi-supervisionado para a RNPe.

tarefa de aprendizado semi-supervisionado de forma incremental, amostra por amostra, não sendo, portanto, necessário acumular uma quantidade de amostras para serem utilizadas para treinamento.

4.4 Conclusão

Neste capítulo foi apresentada a Rede Neural Probabilística evolutiva (RNPe), uma rede neural supervisionada com aprendizado incremental baseada na RNP, no MMG, e em EM. Algumas das suas principais características são a capacidade de adaptar sua arquitetura conforme a necessidade do problema, usar uma única vez as amostras para treinar essa rede e continuar aprendendo durante toda a sua existência. Alguns dos procedimentos utilizados nesta rede neural foram inspirados em outras redes neurais incrementais, sendo que ela apresenta características relacionadas aos SCEs, e suas principais diferenças com relação às técnicas apresentadas no Capítulo 3 foram mencionadas na Seção 4.2.

Ao final, foi descrito um método para aprendizado semi-supervisionado com a RNPe. O modelo proposto é baseado em um comitê de redes neurais, por este ser mais robusto e apresentar maior desempenho que um modelo usando um único classificador.

No capítulo seguinte, serão apresentadas as bases de dados a serem usadas na avaliação empírica da rede proposta e a metodologia empregada nos experimentos.

Capítulo 5

Bases de dados e metodologia

Neste capítulo, são apresentadas as bases de dados utilizadas nos experimentos, as técnicas aplicadas sobre as bases e as medidas escolhidas para quantificar o desempenho de cada técnica. Também são mostrados os testes estatísticos aplicados sobre os resultados.

Antes de apresentar as bases de dados, a Seção 5.1 enumera algumas métricas, chamadas de métricas de complexidade. Elas são utilizadas para mensurar características das bases de dados impactantes na tarefa de aprendizado de máquina. Com o auxílio dessas métricas, é possível verificar detalhes importantes, tais como sobreposição e complexidade do contorno de separação entre classes, ajudando a verificar o grau de dificuldade de cada base de dados.

Na Seção 5.2 são descritas as bases de dados usadas nos experimentos com algumas informações básicas sobre elas, incluindo o objetivo de cada uma e o número de classes e amostras. Uma tabela com as medidas de complexidade de cada base de dados está incluída no final dessa seção.

A metodologia dos experimentos, as métricas de avaliação e as técnicas usadas são apresentadas na Seção 5.3.

5.1 Métricas de Complexidade de Bases de Dados

O comportamento de classificadores tem sido observado, empiricamente, ser fortemente dependente dos dados usados para obter as regras de classificação. Consequentemente, uma análise das características de tais dados torna-se ferramenta essencial para predizer o comportamento de um classificador sobre uma base de dados, sem ser necessário testá-lo, ou para selecionar um melhor classificador para um problema particular (Sotoca et al., 2005; Ho e Basu, 2002).

Tipicamente, as características dos dados são obtidas através de medidas estatísticas como número de classes, número de atributos e entropia das classes, entre outras. Exemplo de tal análise foi realizada no projeto StatLog (Michie et al., 1994), em que um conjunto de medidas estatísticas foi usado para predizer a aplicabilidade de cada classificador avaliado para um problema específico. No entanto, tais medidas foram observadas não serem suficientes para realizar uma boa caracterização de bases de dados, e que, na tarefa de classificação, a distribuição espacial dos padrões é mais importante (Ho e Basu, 2002).

Numa tentativa de melhor caracterizar conjuntos de dados, um grupo de medidas de complexidade das bases de dados tem sido sugerido nos últimos anos (Sotoca et al., 2005; Ho e Basu, 2002; Mansilla e Ho, 2004; Mollineda et al., 2005). Elas incluem informações estatísticas, poder discriminante dos atributos, estimativa do contorno e formato das classes, dentre outras. Algumas dessas medidas foram usadas neste trabalho para uma melhor caracterização das bases de dados.

A seguir, as medidas usadas neste trabalho são informadas baseadas na descrição em (Orriols-Puig et al., 2010), cujos autores disponibilizaram uma implementação em C++ para obtenção dos valores das mesmas.

5.1.1 Medidas de Sobreposição

Estas medidas são focadas na efetividade de um único atributo na separação de classes. Em outras palavras, estima o poder discriminativo dos atributos.

1. **Razão discriminante de Fisher (F1)**: retorna o máximo poder discriminativo de um atributo, isto é:

$$F1 = \max_{i=1}^{d} f_i, \tag{5.1}$$

onde f_i é o poder discriminativo do i-ésimo atributo, d é o número de atributos na base de dados, e a função max retorna o maior poder discriminativo.

Para o caso de uma base de dados com somente duas classes, o valor de f_i é calculado de acordo com a Equação 5.2:

$$f_i = \frac{(\mu_{1,i} - \mu_{2,i})^2}{(\sigma_{1,i})^2 + (\sigma_{2,i})^2}, \tag{5.2}$$

onde $\mu_{k,i}$ e $\left(\sigma_{k,i}\right)^2$ são a média e a variância do i-ésimo atributo da classe c_k.

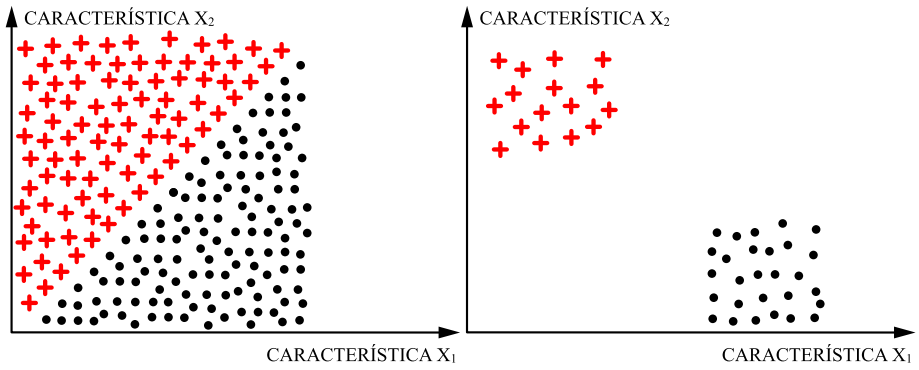


Figura 5.1: Tanto em (a) quanto em (b) a separação entre classes é linear. Em (a) uma única característica não é suficiente para realizar a separação entre as classes, e o valor de F1 é baixo. Em (b) a separação entre as classes pode ser realizada usando qualquer uma das duas características, e o valor de F1 é mais elevado.

Para bases de dados com mais de duas classes, o valor de f_i é obtido pela Equação 5.3:

$$f_i = \frac{\sum_{k=1}^{|C|-1} \sum_{j=k+1}^{|C|} p_k p_j \left(\mu_{k,i} - \mu_{j,i}\right)^2}{\sum_{k=1}^{|C|} p_k \left(\sigma_{k,i}\right)^2}, \tag{5.3}$$

onde $|C|$ é a quantidade de classes; e p_k e p_j são as proporções de amostras das classe c_k e c_j, respectivamente.

Valor alto de F1 indica que ao menos um dos atributos é capaz de separar amostras de classes diferentes em partições paralelas ao eixo do espaço de características. Um valor baixo não significa, necessariamente, que as classes não são linearmente separáveis, mas que não podem ser discriminadas por hiperplanos paralelos a um dos eixos do espaço de características. O limite inferior dessa medida é zero, mas não existe limite superior definido (nas bases de dados analisadas, valores de F1 maiores do que 10 indicaram uma maior facilidade para realizar a correta classificação das amostras das bases de dados).

A Figura 5.1 ilustra duas situações nas quais a separação entre as classes é linear. Na Figura 5.1(a) não é possível realizar uma separação apropriada entre as classes usando somente uma característica, e o valor de F1 é baixo. Na Figura 5.1(b) a separação pode ser realizada usando qualquer uma das duas características, e o valor de F1 é mais elevado.

2. **Volume de região sobreposta (F2)**: calcula a escala de sobreposição das distribuições espaciais de cada classe.

A definição desta medida para bases de dados de duas classes é dada a seguir. Para cada atributo, é calculada a razão da largura do intervalo de sobreposição (intervalo possuidor de amostras de ambas as classes) pela largura de todo o intervalo. Então, a medida retorna o produto das razões calculadas para cada atributo:

$$F2 = \prod_{i=1}^{d} \frac{MIN\_MAX_i - MAX\_MIN_i}{MAX\_MAX_i - MIN\_MIN_i}, \tag{5.4}$$

onde d é o número de atributos, e:

$$MIN\_MAX_i = \min\left(\max\left(a_i, c_1\right), \max\left(a_i, c_2\right)\right), \tag{5.5}$$

$$MAX\_MIN_i = \max\left(\min\left(a_i, c_1\right), \min\left(a_i, c_2\right)\right), \tag{5.6}$$

$$MAX\_MAX_i = \max\left(\max\left(a_i, c_1\right), \max\left(a_i, c_2\right)\right), \tag{5.7}$$

$$MIN\_MIN_i = \min\left(\min\left(a_i, c_1\right), \min\left(a_i, c_2\right)\right), \tag{5.8}$$

onde a_i é o i-ésimo atributo, c_1 e c_2 referem às duas classes, e $\max\left(a_i, c_k\right)$ e $\min\left(a_i, c_k\right)$ são os valores máximo e mínimo do atributo a_i para a classe c_k, respectivamente.

Para o caso de três ou mais classes, o valor absoluto de F2 é calculado para cada par de classes, e a soma desses valores é retornada como saída. A Figura 5.2 ilustra, para cada atributo, o intervalo de sobreposição e a largura total de intervalo de amostras de duas classes hipotéticas. Um baixo valor de F2 significa atributos podendo discriminar bem amostras de classes diferentes. O menor valor de F2 é zero e o maior valor é $|C|(|C|-1)/2$, sendo $|C|$ o número de classes da base de dados. Para $|C| = 2$, o maior valor de F2 é 1.

3. **Eficiência individual de atributo (F3)**: calcula o poder discriminativo de cada atributo individual e retorna o poder discriminativo do atributo capaz de distinguir o maior número de amostras de treinamento dentro da base de dados.

   Desta forma, para cada atributo, é considerada a região de sobreposição (isto é, a região onde existem amostras de ambas as classes) e retorna a razão entre o número de amostras que não estão nessa região de sobreposição e o número total de amostras. O cálculo é realizado para cada par de classes. A maior taxa de discriminação é o valor da medida F3. A Figura 5.3 ilustra, para cada atributo, o intervalo de sobreposição de amostras de duas classes hipotéticas e indica a forma de calculo da medida F3.

   Um alto valor de F3 indica haver pelo menos um atributo com grande capacidade de discriminação. O valor de F3 encontra-se entre 0 e 1.

4. **Eficiência coletiva de atributos (F4)**: segue a mesma ideia da apresentada para F3, mas agora considera o poder discriminativo de todos os atributos.

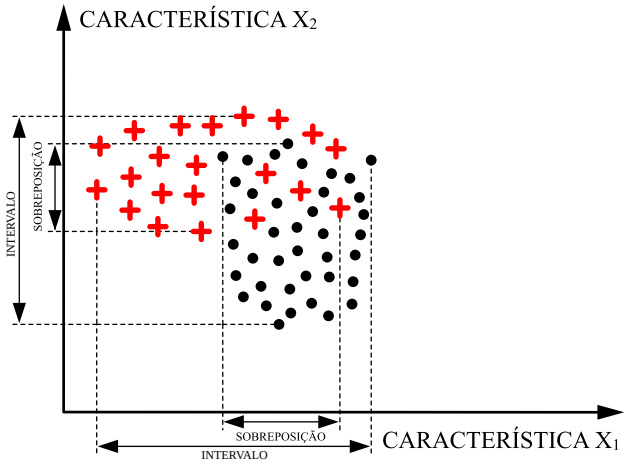


Figura 5.2: A figura mostra a distribuição de duas classes (pontos e cruzes) e o intervalo de sobreposição das amostras de ambas as classes para cada atributo (característica). Também é mostrada a largura total do intervalo de amostras para cada atributo. Quanto maior a razão entre a sobreposição e a largura do intervalo, maior o valor de F2.

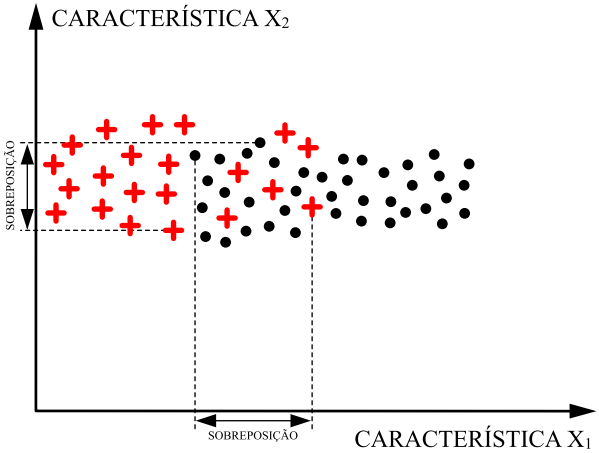


Figura 5.3: A figura mostra a distribuição de duas classes (pontos e cruzes) e o intervalo de sobreposição das amostras de ambas as classes para cada atributo (característica). Para calcular a medida F3, é calculado para cada característica o número de amostras fora do intervalo de sobreposição dividido pelo número total de amostras. No caso dessa figura, a maior porcentagem de amostras fora do intervalo de sobreposição é obtida pela característica X_1, e essa taxa é o valor de F3.

O poder discriminativo coletivo é calculado da seguinte forma: primeiro, é selecionado o atributo mais discriminativo, isto é, o atributo capaz de distinguir o maior número de amostras de uma classe. Esse atributo é selecionado usando o mesmo procedimento descrito para calcular o valor de F3. Então, todas as amostras discriminadas são re-

movidas, e o segundo atributo mais discriminativo é selecionado (usando as amostras restantes). Esse procedimento continua até todas as amostras serem discriminadas ou todos os atributos serem selecionados. Finalmente, a medida retorna a proporção de amostras que têm sido discriminadas. Esse cálculo é realizado para cada par de classes.

Esta medida é ligeiramente diferente da medida F3. Enquanto F3 somente considera a fração de amostras discriminadas pelo atributo mais discriminativo, a medida F4 considera o efeito de todos os atributos. Portanto, F4 providencia mais informação por considerar o efeito de todos os atributos, já que realça o poder discriminativo coletivo de todos os atributos. O valor mínimo de F4 é zero. Para problemas de duas classes o valor está entre 0 e 1.

5.1.2 Medidas de separabilidade das classes

A seguir, serão descritas medidas capazes de estimar a complexidade de separação de amostras de diferentes classes baseadas na forma e contorno das classes.

1. **Fração de pontos no contorno da classe (N1)**: fornece uma estimativa do comprimento do contorno da classe.

   O método baseia-se em calcular um *minimum spanning tree* (MST) conectando todos os pontos (amostras) aos vizinhos mais próximos deles, independentemente da classe ao qual pertençam. Então, os pontos conectados e pertencentes a classes diferentes são contados. Se um ponto é conectado a mais de uma classe diferente, ele é contado somente uma vez. Esses são pontos próximos ao contorno da classe (Figura 5.4). A fração desses pontos pelo número total de pontos no conjunto de dados é o valor da medida N1.

   Valores altos para esta medida indicam que a maioria dos pontos encontra-se próximo ao contorno da classe, e assim pode ser mais difícil para um classificador definir precisamente o contorno dessa classe. O valor de N1 situa-se entre 0 e 1.

2. **Razão da distância média dos vizinhos mais próximos dentro das classes e entre classes (N2)**: compara o espalhamento das amostras dentro de uma classe com a distância dos vizinhos mais próximos de outras classes.

   Para cada amostra x_i é calculada a distância ao vizinho mais próximo dela dentro da classe (Dist_int(x_i)) e a distância ao vizinho mais próximo de qualquer outra classe (Dist_ext(x_i)). Então, o resultado é a razão da soma das distâncias dentro da classe

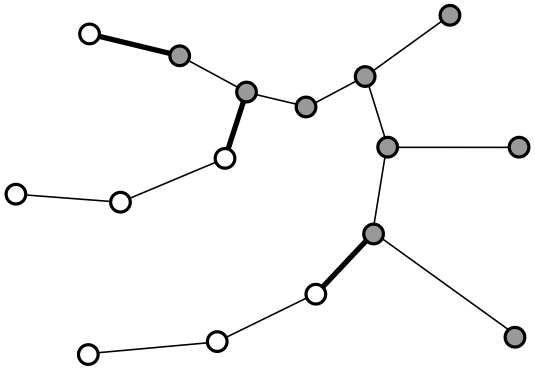


Figura 5.4: Um *minimum spanning tree* conectando pontos de duas classes. Círculos com tons mais escuros pertencem a uma classe diferente à dos círculos com um tom mais claro no interior. As linhas mais espessas conectam pontos de classes diferentes (Ho e Basu, 2002).

pela soma das distâncias fora da classe, isto é:

$$N2 = \frac{\sum_{i=1}^{n} \text{Dist\_int}(x_i)}{\sum_{i=1}^{n} \text{Dist\_ext}(x_i)}, \tag{5.9}$$

onde n é o número de amostras na base de dados.

Valores baixos desta medida sugerem amostras da mesma classe encontrarem-se próximas no espaço de características com relação à distância entre amostras de outras classes. Valores altos indicam amostras da mesma classe dispersas. O menor valor de N2 tende a zero, mas não existe limite superior para esta medida. A Figura 5.5 ilustra duas situações em que são obtidos valores diferentes de N2.

3. **Taxa de erro de *leave-one-out* do classificador vizinho mais próximo (N3)**: informa o quão próximas estão as amostras de diferentes classes. Ela retorna a taxa de erro de *leave-one-out* (ver Subseção 5.3.3) do classificador vizinho mais próximo. Baixos valores dessa medida sugerem haver grande vazio entre os contornos de classe. O valor de N3 pode variar entre 0 a 1.

5.1.3 Medidas de geometria e densidade

Estas medidas têm como finalidade descrever a geometria e a forma da distribuição das classes no espaço de características, numa tentativa de caracterizar indiretamente a separabilidade das classes.

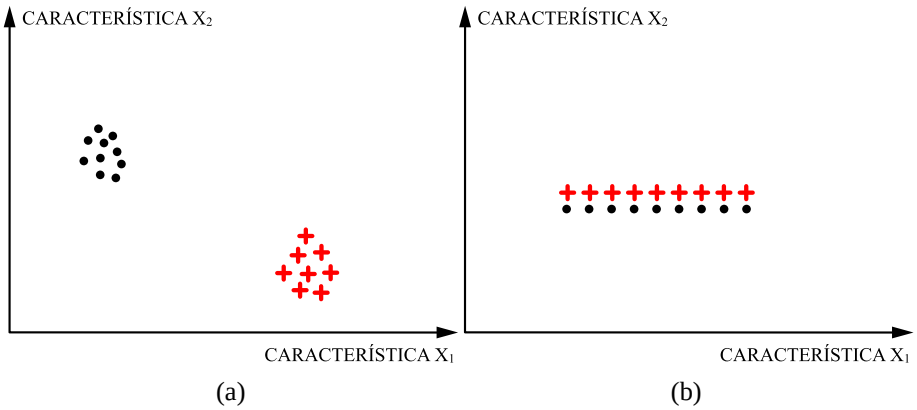


Figura 5.5: Em (a) a distância entre amostras de mesma classe é menor do que a distância para amostras de uma classe diferente, logo o valor de N2 é baixo. Por outro lado, em (b) a distância de amostras de classes diferentes é similar à distância de amostras da mesma classe, assim o valor de N2 é mais elevado.

1. **Não linearidade do classificador vizinho mais próximo (N4)**: dado um conjunto de treinamento, o método cria um conjunto de teste através de interpolação linear entre pares de amostras selecionadas aleatoriamente da mesma classe. Então, a medida retorna a taxa de erro sobre o conjunto de teste usando o classificador vizinho mais próximo. A Figura 5.6 ilustra o procedimento de interpolação linear entre duas amostras de uma classe (pontos escuros) para criar uma nova amostra (ponto mais claro). Para classes linearmente separáveis, o erro esperado é zero. O valor de N4 está na escala entre 0 e 1.

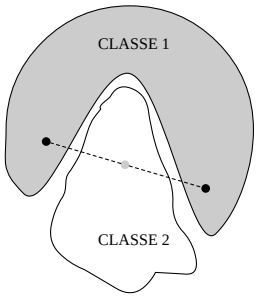


Figura 5.6: A figura mostra a distribuição de amostras de duas classes, a classe 1 (escura) e a classe 2 (clara). Uma interpolação entre duas amostras da classe 1 (pontos escuros) gerou uma nova amostra (ponto mais claro), que está contida dentro da distribuição da classe 2. Quanto mais amostras geradas de uma classe forem classificadas como de uma outra classe, maior o valor de N4.

2. **Fração da esfera de cobertura máxima (T1)**: para calcular essa medida, é realizado

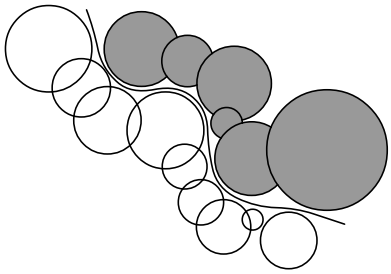


Figura 5.7: Os círculos representam as esferas de cobertura, e as cores claras e escuras indicam as classes das esferas. As esferas crescem tanto quanto o possível antes de tocar em uma amostra de outra classe (representada pela linha de separação de classes). Esferas totalmente contidas em outras são removidas (Ho, 2000).

o seguinte procedimento. Em um ponto (amostra) de um conjunto de dados é centrada uma hiperesfera que cresce tanto quanto o possível antes de tocar qualquer ponto de outra classe. Portanto, um subconjunto de hiperesferas contém um conjunto de pontos da mesma classe sem incluir qualquer ponto de outra classe. A medida considera somente os maiores subconjuntos de hiperesferas, removendo todas aquelas incluídas em outras. Então, a medida retorna a fração do número de hiperesferas pelo número total de pontos. A Figura 5.7 ilustra hiperesferas de duas classes hipotéticas.

O valor de T1 tende a ser mais elevado em problemas onde cada amostra está mais próxima das amostras de outras classes do que das de sua própria classe. Valores menores sugerem as amostras estarem agrupadas em uma hiperesfera, enquanto valores maiores insinuam ser a estrutura de separação entre classes mais estreita (Ho, 2000). O valor de T1 pode variar entre 0 e 1.

5.1.4 Medidas estatísticas e erro assintótico

Além das medidas anteriores, outras informações sobre as bases de dados são mencionadas.

1. **Número de amostras na base de dados (NA)**: indica o total de amostras de cada base de dados.
2. **Número de atributos (A)**: indica o número de atributos do vetor característico das amostras das bases de dados.
3. **Número de classes (C)**: informa a quantidade total de classes de cada base de dados.

4. **Entropia normalizada das classes (EN)**: mede o grau de imprevisibilidade de ser associada a uma amostra a classe correta utilizando somente a probabilidade *a priori* das classes.

   A entropia normalizada EN é calculada usando a Equação 5.10:

$$EN = \frac{\sum_{i=1}^{|C|} p_i \log_2(p_i)}{\log_2(1/|C|)}, \tag{5.10}$$

   onde $|C|$ é o número de classes na base de dados e p_i é a proporção de amostras da classe c_i na base de dados.

   Valores altos de EN indicam ser a proporção de amostras por classes bem equilibrada. Um valor baixo indica a base de dados estar mais polarizada para uma ou mais classes. A escala de valores para EN é entre 0 e 1.

5. **Taxa de erro assintótico do classificador vizinho mais próximo (N6)**: Raudys e Jain sugerem, em (Raudys e Jain, 1991), uma forma de estimar a taxa de erro assintótico de um classificador, no qual a taxa de erro assintótico é a taxa de erro de um classificador quando esse é treinado com um número de amostras tendendo ao infinito[1].

   Seja N3 a taxa de erro *leave-one-out* e N5 a taxa de erro do método de re-substituição (onde todas as amostras são usadas como treino e teste), ambas para o classificador vizinho mais próximo, então, a taxa de erro assintótico N6 pode ser estimada por:

$$N6 = \frac{N3 + N5}{2}. \tag{5.11}$$

   Valores baixos de N6 sugerem bases de dados não apresentando dificuldades elevadas para a correta classificação. O valor de N6 encontra-se entre 0 e 1.

5.2 Bases de Dados

Um total de 20 bases de dados de diferentes origens foram utilizadas nos experimentos. Estão incluídas algumas bases de dados do mundo real e outras artificiais, variando desde aquelas de estudos médicos até as geradas propositalmente para verificar desempenho de classificadores. Esse conjunto de bases foi selecionado com o sentido de proporcionar uma diversidade nas características dos problemas de classificação, não se restringindo a apenas um grupo específico de problemas.

[1] Considerando que as amostras sejam i.i.d. (independentes e identicamente distribuídas).

Também, só foram usadas bases de dados para tarefas de classificação que não possuem nenhum valor faltando em seus dados, isto é, características presentes em algumas amostras, mas ausentes em outras (com exceção das bases *CNAE-9*, *WebKB-4* e *Reuters-8*, tendo sido incluído o valor zero para atributos ausentes nas amostras). A descrição das bases de dados e algumas de suas estatísticas é apresentada a seguir.

- *CNAE-9*: subconjunto de uma base maior de descrições de atividades econômicas de empresas brasileiras, usadas pelo IBGE (Instituto Brasileiro de Geografia e Estatística) para ter uma visão do setor produtivo do País. A base original é formada por amostras multi-rotuladas e centenas de classes. Porém, esse subconjunto contém somente amostras uni-rotuladas das classes mais recorrentes da base de dados original (Ciarelli et al., 2009a). A *CNAE-9* é composta por 1080 amostras igualmente distribuídas em 9 classes (120 por classe) e 856 atributos, sendo cada atributo a frequência de ocorrência de um determinado termo nas descrições. Caso um termo não apareça em uma amostra, sua frequência é zero. A rede neural VG-RAM WNN apresentou o melhor resultado para essa base de dados, com índice de 98,33% de acurácia (porcentagem de amostras corretamente classificadas), quando utilizando 900 amostras para treino e 180 amostras para teste[2].

- *WebKB-4*: os documentos na base de dados WebKB são páginas de Internet colecionadas pela *World Wide Knowledge Base* (WebKB). Essas páginas foram coletadas dos Departamentos de Ciência da Computação de quatro universidades em 1997, e manualmente classificadas dentro de sete diferentes classes, tais como estudante, curso, departamento, etc. A *WebKB-4* é um subconjunto das quatro classes com maior frequência dentro da base de dados da WebKB, totalizando 4199 amostras (Cardoso-Cachopo, 2007). De acordo com o conhecimento do autor, o melhor resultado encontrado está descrito em (Cardoso-Cachopo, 2007), quando utilizado um SVM e alcançado um índice de acurácia de 85,82%, usando 2803 amostras para treinamento e 1396 para teste, além da técnica tf-idf (*term frequency - inverse document frequency*) para realçar os atributos mais relevantes. A base original é composta por milhares de atributos, mas, para reduzir o custo computacional, e tornar viável o seu uso em algumas técnicas, a quantidade de atributos foi reduzida, neste trabalho, para 3000, usando a técnica de Informação Mútua (Church e Hanks, 1990) para seleção dos atributos.

- *Reuters-8*: essa base é um subconjunto com as oito classes mais recorrentes da *Reuters-21578*, atualmente uma das bases de dados mais amplamente utilizadas para pesquisas de categorização de textos. Essa base consiste de artigos coletados ao longo do ano

[2]Resultado não publicado.

de 1987 da agência de notícias *Reuters*, uma das mais importantes do mundo. Os documentos foram reunidos e indexados manualmente em categorias pela *Reuters Ltd.* e *Carnegie Group Inc.* em 1987. Em 1990, os documentos foram disponibilizados pela *Reuters* e *Carnegie Group Inc.* para propósitos de pesquisa. No subconjunto utilizado, todas as amostras são uni-rotuladas, embora na base original existam amostras multi-rotuladas (Cardoso-Cachopo, 2007). Essa base de dados possui um total de 7674 amostras e, em (Cardoso-Cachopo, 2007) é informado um índice de acurácia de 96,98%, quando utilizando o classificador SVM, 5485 amostras para treino e 2189 para teste. Também foi utilizada a técnica tf-idf. Novamente, devido ao grande número de atributos, foi utilizada Informação Mútua (Church e Hanks, 1990) para seleção dos 3000 atributos mais relevantes.

- *Abalone*: base de dados para predição da idade de um tipo de molusco, conhecido como haliote, encontrado na maioria dos mares temperados, como dos Estados Unidos, Austrália e Japão. A idade do haliote pode ser determinada pelo corte da casca, coloração e o número de anéis contados através de um microscópio (Frank e Asuncion, 2010). Outras medidas podem ser usadas e, nesse caso, foram sexo, comprimento e diâmetro da casca e peso de diferentes partes do animal. Esta base de dados é formada por 28 idades de moluscos (classes) e 4177 amostras, cada uma delas contendo oito atributos. Uma das dificuldades dessa base de dados é a pequena quantidade de amostras por classe, sendo para algumas classes disponíveis somente poucas unidades. Uma árvore de decisão proposta em (Tan e Dowe, 2002) alcançou acurácia de 25,7% sobre essa base de dados utilizando o método de 10 *fold cross-validation*.

- *Yeast*: tem como objetivo identificar a localização de proteínas dentro das células. Para isso, são usadas várias informações obtidas através de análises realizadas nas células. Essa base é formada por 1484 amostras divididas em 10 classes, e cada amostra possui oito atributos. Em um experimento relatado em (Allwein et al., 2000), foi usado 10 *fold cross-validation* e SVM para classificação. O resultado obtido foi de 60,3% de acurácia.

- *Blood Transfusion Service Center*: doada pelo Centro de Serviço de Transfusão de Sangue de uma cidade de Taiwan, cada amostra dessa base representa um doador e contém as informações dos meses desde a última doação, o número de doações, total de sangue doado e meses desde a primeira doação. O objetivo é descobrir se o doador doou sangue em março de 2007. Contém 748 amostras com cinco atributos cada uma. Em um experimento usando SVM realizado em (Kobos e Mandziuk, 2012) foi obtida acurácia de 77,71% com 10 *fold cross-validation*.

- *Haberman's Survival*: essa base contém casos de um estudo conduzido entre 1958 e

1970 no hospital da Universidade de Chicago sobre sobrevivência de pacientes submetidos à cirurgia de câncer de mama. O objetivo é prever se os pacientes continuaram vivos cinco anos após a cirurgia. Ela tem 306 amostras, com três atributos cada uma. Os atributos usados foram idade do paciente, ano da operação e número de nódulos detectados. Em (Zhang e Street, 2008) foi obtida acurácia de 74,5% com 10 *fold cross-validation*, quando utilizado um comitê de 100 classificadores SVM.

- *Wine*: tem como objetivo determinar a origem do vinho através de análise química de vinhos na Itália, vindos de três diferentes cultivos. A análise determinou a quantidade de 13 componentes presentes em cada um dos três tipos de bebida, num total de 178 amostras. Alguns desses componentes são os níveis de álcool, de magnésio e de ácido málico, e intensidade e tonalidade da cor. Com um comitê de cinco redes neurais foi obtida acurácia de 97,7% em (Zhou e Goldman, 2004) quando usando 10 *fold cross-validation.*

- *Balance Scale*: conjunto de dados gerado para modelar resultados experimentais psicológicos. O objetivo é identificar, para cada amostra, a partir das informações disponíveis, qual dos dois lados de uma balança está mais baixo (esquerdo ou direito) ou se ambos estão equilibrados. Os atributos das amostras são o peso e a distância do lado esquerdo, e o peso e a distância do lado direito. A maneira de classificar é atribuir a cada amostra o lado de maior momento. Se os momentos forem iguais, então a balança está equilibrada. Essa base é composta por 625 amostras, três classes e quatro atributos. Em (Holmes et al., 2002), a técnica AdaBoost.MH obteve acurácia de 90,82%. O resultado foi obtido de 10 *fold cross-validation*.

- *Iris*: uma das bases de dados mais famosas na literatura de reconhecimento de padrões. Contém três classes de 50 amostras cada uma, sendo cada classe referente a um tipo de planta. Cada amostra possui quatro atributos, sendo os comprimentos e larguras da sépala e pétala de cada planta. Uma classe é linearmente separável das outras e as outras duas não são linearmente separáveis entre si. Um comitê de cinco redes neurais obteve acurácia de 97,8% em (Zhou e Goldman, 2004). O resultado foi obtido de 10 *fold cross-validation*.

- *Car Evaluation*: conjunto de amostras derivadas de um simples modelo de decisão hierárquica para avaliar carros de acordo com os conceitos de aceitabilidade do mesmo. Os atributos considerados são preço, valor de manutenção, número de portas, quantidade de passageiros, espaço interno e segurança. Esses atributos possuem conceitos intermediários como, por exemplo, alto, médio e baixo para a variável preço. Os carros são classificados como inacessíveis, acessíveis, bom e muito bom. Essa base é composta por 1728 amostras classificadas em quatro categorias, sendo cada amostra

composta por seis atributos. A classe mais predominante nessa base está associada à cerca de 70% das amostras. Uma acurácia de 93,3% foi obtida de 10 *fold cross-validation* quando aplicada uma árvore de decisão proposta em (Tan e Dowe, 2002).

- *Heart*: tem como objetivo identificar se um paciente tem ou não uma doença cardíaca a partir de um conjunto de informações. Essa base de dados contém 270 amostras e 13 atributos. Alguns atributos são idade, sexo, pressão sanguínea, taxa de açúcar no sangue e batimento cardíaco. Com um comitê de 100 classificadores SVM foi obtida acurácia de 83,52% com 10 *fold cross-validation* em (Zhang e Street, 2008).

- *Sonar*: a tarefa dessa base de dados é conseguir discriminar os sinais de um sonar que ricochetearam em um cilindro de metal daqueles que ricochetearam numa pedra quase cilíndrica. Na coleta dos dados, o sonar transmitiu um sinal em frequência modulada, aumentando esta frequência. O sinal foi decomposto em 60 bandas de frequência, e a energia de cada banda de frequência é utilizada como atributo. Sendo assim, essa base possui um total de 60 atributos e duas classes (rocha e metal), e um total de 208 amostras. Uma acurácia média de 81,2% foi obtida em (Zhou e Goldman, 2004) com um comitê de cinco redes neurais. O resultado foi obtido para 10 *fold cross-validation*.

- *Image Segmentation*: amostras selecionadas aleatoriamente de uma base de dados maior de sete imagens externas. A tarefa é associar, para cada região de 3×3 *pixels*, o ambiente mais similar, tais como grama, céu e janela. A base de dados é formada por 2310 amostras, sete classes e 19 atributos. Alguns dos atributos usados para classificação são a média de cada camada de cor do espaço RGB (*red*, *green* e *blue*) e o valor da intensidade média. Uma acurácia de 96,74% foi obtida com 10 *fold cross-validation* aplicando uma árvore de decisão para cada par de classes. Esse resultado foi reportado em (Holmes et al., 2002).

- *Zoo*: uma base de dados cujo objetivo é identificar a classe taxonômica dos animais, isto é, identificar se o animal é réptil, anfíbio, mamífero, e assim por diante. Contém 16 atributos com valores Booleanos indicando características do animal, tais como se possui pelo, penas ou é aquático. Há um total de 101 amostras classificadas em sete categorias. Uma acurácia de 95,94% foi obtida com 10 *fold cross-validation* aplicando uma versão de árvore de decisão que utiliza $2^{|C|} - 1$ árvores para classificação, onde $|C|$ é o número de classes. Esse resultado foi reportado em (Holmes et al., 2002).

- *Spambase*: a meta dessa tarefa é classificar um *e-mail* como *spam* ou não *spam*. O conceito de *spam* é diverso, podendo ser desde propagandas de produtos ou *web sites* até pornografia ou mensagens indesejáveis. Essa base é composta por 4601 amostras com 57 atributos cada. Grande parte dos atributos representa a quantidade que determinados termos ocorrem nos *e-mails* e informações sobre palavras maiúsculas nos

mesmos. Uma acurácia de 88,7% foi obtida em (Wang e Witten, 2002) para 10 *fold cross-validation* e um modelo logístico usando *maximum likelihood* para estimar os parâmetros.

- *Diabetes*: conjunto de dados para identificar pacientes com diabetes. Restrições foram impostas na coleta dos dados, tais como todos os pacientes são mulheres com ao menos 21 anos de idade. Algumas informações coletadas das pacientes foram idade, índice de massa corporal, pressão sanguínea e número de gravidezes. A base de dados possui 768 amostras com oito atributos cada. Uma acurácia média de 78,6% foi obtida em (Zhou e Goldman, 2004) com um comitê de cinco redes neurais. O resultado foi obtido para 10 *fold cross-validation*.

- *Texture*: o objetivo dessa base de dados é distinguir as amostras de diferentes texturas de imagens, tais como grama, papel e couro. A base de dados contém 11 classes com 500 amostras cada, sendo cada classe referente a um tipo de textura do álbum de Brodatz (Brodatz, 1966). Cada amostra é formada por 40 atributos obtidos pela estimativa do momento de quarta ordem em quatro orientações. Uma combinação de 25 redes neurais com a técnica de *K-means* obteve acurácia de 99,7% para essa base de dados, conforme informado em (Min e Cho, 2007). O resultado foi obtido com 10 *fold cross-validation*.

- *Gaussian*: base de dados artificial com propósito de verificar o comportamento de classificadores para distribuições fortemente sobrepostas e separação não linear das classes. Essa base de dados é formada por duas classes, ambas com distribuições normais, média igual a 0 e com 2500 amostras cada distribuição. A diferença entre as classes é o desvio padrão, com a primeira classe tendo desvio igual a 1 e a segunda classe igual a 2. Cada amostra possui quatro atributos. Em (Lee, 2000), é informado que um kNN com o método *leave one out* obteve o resultado de 80,6% de acurácia.

- *Concentric*: base artificial, composta por duas distribuições bidimensionais, circulares, concêntricas e com diferentes raios, não havendo qualquer sobreposição das classes. Composta por 2500 amostras. A melhor taxa de classificação teórica é de 100%. Uma acurácia de 98,8% foi obtida em (Min e Cho, 2007) ao aplicar uma combinação de 25 redes neurais com a técnica de *K-means*. O resultado foi obtido de 10 *fold cross-validation.*

As versões originais das bases *WebKB-4* e *Reuters-8* foram obtidas em (Cardoso-Cachopo, 2007), *Texture*, *Gaussian* e *Concentric* em *ELENA database* (Lee, 2000), e as demais em *UCI* (*University of California Irvine*) *database* (Frank e Asuncion, 2010).

Tabela 5.1: Caracterização das bases de dados segundo as medidas de complexidade.

Base de dados	*C*	*A*	*NA*	*F1*	*F2*	*F3*	*F4*	*N1*	*N2*	*N3*	*N4*	*N6*	*T1*	*EN*
Balance	*3*	*4*	*625*	*0,20*	*3,00*	*0,00*	*0,00*	*0,28*	*0,68*	*0,23*	*0,33*	*0,11*	*0,91*	*0,83*
Blood	*2*	*5*	*748*	*0,29*	*0,27*	*0,01*	*0,01*	*0,43*	*0,61*	*0,33*	*0,40*	*0,22*	*0,99*	*0,79*
Iris	*3*	*4*	*150*	*16,04*	*0,05*	*0,57*	*0,57*	*0,10*	*0,21*	*0,05*	*0,01*	*0,02*	*0,89*	*1,00*
Haberman	*2*	*3*	*306*	*0,185*	*0,72*	*0,03*	*0,03*	*0,54*	*0,75*	*0,35*	*0,38*	*0,19*	*0,93*	*0,83*
Car	*4*	*6*	*1728*	*0,29*	*0,39*	*0,67*	*1,70*	*0,16*	*0,65*	*0,07*	*0,20*	*0,04*	*0,99*	*0,60*
CNAE-9	*9*	*856*	*1080*	*2,94*	*0,00*	*0,32*	*1,10*	*0,21*	*0,77*	*0,14*	*0,06*	*0,07*	*1,00*	*1,00*
WebKB-4	*4*	*3000*	*4199*	*0,40*	*0,00*	*0,09*	*7,64*	*0,45*	*0,94*	*0,35*	*0,14*	*0,18*	*1,00*	*0,94*
Wine	*3*	*13*	*178*	*2,67*	*0,00*	*0,81*	*0,81*	*0,12*	*0,58*	*0,05*	*0,00*	*0,03*	*0,99*	*0,99*
Reuters-8	*8*	*3000*	*7674*	*1,03*	*0,00*	*0,39*	*9,84*	*0,23*	*0,82*	*0,15*	*0,11*	*0,08*	*1,00*	*0,64*
Yeast	*10*	*8*	*1484*	*0,85*	*0,20*	*1,00*	*1,00*	*0,65*	*0,95*	*0,47*	*0,40*	*0,24*	*1,00*	*0,75*
Abalone	*28*	*10*	*4177*	*1,28*	*20,16*	*1,00*	*1,00*	*0,91*	*1,50*	*0,80*	*0,67*	*0,40*	*1,00*	*0,75*
Heart	*2*	*13*	*270*	*0,76*	*0,20*	*0,02*	*0,09*	*0,37*	*0,67*	*0,24*	*0,12*	*0,12*	*1,00*	*0,99*
Sonar	*2*	*60*	*208*	*0,47*	*0,00*	*0,05*	*1,00*	*0,29*	*0,74*	*0,13*	*0,12*	*0,06*	*1,00*	*1,00*
Segmentation	*7*	*19*	*2310*	*15,61*	*0,00*	*0,99*	*0,99*	*0,08*	*0,18*	*0,03*	*0,08*	*0,01*	*0,98*	*1,00*
Texture	*11*	*40*	*5500*	*10,29*	*0,00*	*0,95*	*0,95*	*0,03*	*0,35*	*0,01*	*0,02*	*0,00*	*0,99*	*1,00*
Gaussian	*2*	*4*	*5000*	*0,00*	*0,08*	*0,04*	*0,12*	*0,40*	*0,66*	*0,27*	*0,29*	*0,14*	*1,00*	*1,00*
Concentric	*2*	*2*	*2500*	*0,00*	*0,36*	*0,30*	*0,55*	*0,03*	*0,08*	*0,01*	*0,19*	*0,01*	*0,49*	*0,95*
Spambase	*2*	*57*	*4601*	*0,35*	*0,00*	*0,09*	*0,38*	*0,17*	*0,42*	*0,09*	*0,13*	*0,04*	*0,96*	*0,97*
Diabetes	*2*	*8*	*768*	*0,58*	*0,25*	*0,01*	*0,02*	*0,44*	*0,84*	*0,29*	*0,28*	*0,15*	*1,00*	*0,93*
Zoo	*7*	*16*	*101*	*12,61*	*0,00*	*0,57*	*0,57*	*0,15*	*0,22*	*0,03*	*0,02*	*0,00*	*1,00*	*0,85*

5.2.1 Caracterização das bases de dados

As principais informações estatísticas e as medidas de complexidade das bases de dados mencionadas anteriormente estão resumidas na Tabela 5.1. Cada linha da tabela corresponde a uma base de dados e cada coluna é uma das medidas de complexidade descritas na Seção 5.1. As bases de dados *Abalone*, *Car* e *Yeast* continham atributos não numéricos representando categorias. Para tratar esses atributos, eles foram convertidos em valores numéricos.

Para analisar a Tabela 5.1, é bom relembrar: valores altos das métricas F1, F3 e F4 indicam ao menos um atributo na base de dados possuindo alto poder discriminativo, e valor baixo de F2 indica existir pouca sobreposição espacial dos valores de atributos de diferentes classes. Estas características facilitam a tarefa de classificação.

Por outro lado, valores altos de N1, N2 e N4 sugerem contornos das classes não bem definidos, amostras da mesma classe mais dispersas e separação entre classes não linear, respectivamente. Em outras palavras, valores altos de N1, N2 e N4 indicam uma separação de classes mais complexa e, portanto, a tarefa é mais difícil. Por último, valores baixos de N3 e N6, informando o erro do método *leave-one-out* e o erro assintótico da técnica vizinho mais próximo, insinuam ser a correta classificação uma tarefa não muito difícil.

Com isso em mente, e olhando a Tabela 5.1, é possível ver que as bases de dados *Iris*, *Segmentation*, *Texture* e *Zoo* possuem valores elevados para F1 (acima de 10), e baixos para

as medidas N1, N3, N4 e N6. Portanto, devem ser bases de dados mais fáceis para classificação. De fato, os valores de acurácia encontrados na literatura para essas bases de dados foram superiores a 95%.

Em contraparte, as bases de dados *Blood*, *Haberman*, *Yeast* e *Abalone* apresentam maior grau de dificuldade, facilmente verificado olhando-se o valor de N6, que para o caso da base *Abalone*, alcançou um valor de 0,40 (o maior valor entre as bases de dados). Além disso, essas bases de dados possuem grande não linearidade na separação das classes (N4), amostras da mesma classe estão muito espalhadas em relação a amostras de outras classes (N2) e também não apresentam contornos das classes bem definidas (N1). As bases *Yeast* e *Abalone* possuem alto valor para N3, indicando que podem ser problemas complexos de classificação. Os resultados encontrados na literatura para as bases *Blood* e Hamberman, que possuem apenas duas classes, foram inferiores a 80% e, para as bases *Yeast* e *Abalone*, não foram encontradas acurácias superiores a 61%.

A dificuldade de classificação das outras bases de dados encontra-se entre os dois extremos. As bases *CNAE-9*, *WebKB-4* e *Reuters-8* apresentam dificuldade razoável para classificação, sendo a *WebKB-4* apresentando dificuldade mais elevada entre as três. Olhando para essa base de dados, nota-se que ela possui valor significativamente maior para N1 e N3 e menor para F1 e F3, em relação às outras duas bases. Isso indica existirem muitas amostras próximas aos contornos e o poder discriminativo dos atributos é menor do que para as outras bases.

O valor de EN para algumas bases de dados é igual a 1, indicando amostras igualmente distribuídas entre as classes. Os menores valores de EN são das bases *Car*, *Reuters-8*, *Yeast* e *Abalone*, sugerindo possuírem uma quantidade de amostras maior para uma ou mais classes. Por exemplo: 70% das amostras da base *Car* estão associadas a uma classe, e as restantes estão divididas nas outras três classes da base.

O valor de T1 é aproximadamente igual a 1 para a maioria das bases de dados, sendo o menor valor encontrado na base *Concentric*. Como essa base de dados é composta por duas classes com distribuições concêntricas, as amostras de uma classe podem ser representadas por uma única hiperesfera, enquanto cada amostra da outra classe por aproximadamente uma hiperesfera. Visto que a quantidade de amostras por classe não é muito diferente (EN igual a 0,95), o valor de T1 acaba sendo aproximadamente igual a metade.

5.3 Metodologia dos Experimentos

Essa seção apresenta as técnicas usadas nos experimentos e o procedimento para encontrar os parâmetros de cada técnica. Também são informadas as medidas usadas para avaliar os experimentos e os testes estatísticos aplicados sobre os resultados obtidos.

5.3.1 Técnicas usadas para comparação

Além da técnica proposta RNPe, um conjunto de técnicas foi usado para comparação de resultados. Das técnicas descritas no Capítulo 3, foram usadas nos experimentos a RNPI, eMLP, EfuNN e RNPI-EM.

Das técnicas clássicas de aprendizado, foram usadas a RNP (Specht, 1990); Rocchio (Manning et al., 2008); kNN (Duda et al., 2001; Chatterji, 1990); MLP (Haykin, 2005; Bishop, 2006); e, *Virtual Generalizing Random Access Memories Weightless Neural Networks* (VG-RAM WNN) (Komati e Souza, 2002; Souza et al., 2008; Souza et al., 2009). As técnicas clássicas foram usadas nos experimentos para estabelecer limites inferiores em relação ao uso de recursos computacionais e limites superiores em relação a eficiência na classificação. Tais valores são úteis para comparar a RNPe com técnicas mais clássicas de classificação.

A RNP é uma rede neural não linear cujo número de neurônios é igual ao número de amostras usadas para treinamento. Ela possui somente um parâmetro para ajustar, a variância da função Gaussiana (σ^2). Mais detalhes da RNP são apresentados no Apêndice C. O número de protótipos do classificador Rocchio é igual ao número de classes da base de dados usada para o treinamento. Esse classificador baseia-se em encontrar o centroide de cada classe, e não existe parâmetro para ser ajustado. A distância entre cada centroide e amostra de teste é normalmente calculada pela distância Euclidiana.

O kNN é uma técnica sub-ótima de classificação guiando para uma taxa de erro maior do que o mínimo possível, que é a taxa Bayesiana. No entanto, o classificador Bayesiano requer a função de densidade de probabilidade das classes para ser utilizado, de fato, muitas vezes não conhecida. Por não precisar dessa informação, o kNN acaba se tornando uma técnica mais simples e fácil de ser utilizada (Duda et al., 2001). A classificação realizada por essa técnica consiste em calcular a distância entre cada amostra de treinamento e a amostra de teste. A partir destes resultados, são selecionados os k vizinhos mais próximos da amostra de teste, que são as k amostras de treinamento mais próximas da amostra de teste. A classe com mais amostras entre os k vizinhos mais próximos é associada à amostra de teste. A distância frequentemente empregada é a distância Euclidiana. Uma desvantagem do kNN

é a necessidade de armazenar as amostras de treinamento e, dependendo da quantidade de amostras, pode resultar em um alto custo computacional.

MLP é uma das redes neurais mais amplamente utilizadas na literatura, podendo ser aplicada para tarefas de classificação, predição, dentre outras. Ela é uma rede neural direta (ou seja, a saída não é utilizada como entrada), com uma ou mais camadas ocultas, cujos neurônios possuem uma função de transferência não linear, normalmente uma função *sigmoidal* (Bishop, 2006). Seu treinamento consiste em passar várias vezes os dados de treinamento pela rede neural e ajustar os pesos das conexões dos neurônios para obter um melhor ajuste aos dados de treinamento. Cada apresentação da base de treinamento à rede neural é chamada de época.

Nos experimentos, foi utilizado o algoritmo de aprendizado gradiente conjugado escalonado para treinar o MLP. Esse algoritmo foi apresentado em (Møller, 1993), e basicamente combina o método de treinamento de Levenberg-Marquardt (Levenberg, 1944; Marquardt, 1963) com a abordagem de gradiente conjugado. Esse algoritmo foi selecionado para o treinamento, pois, na avaliação realizada em (Demuth e Beale, 2002) com diferentes algoritmos e bases de dados, apresentou bom desempenho sobre diferentes tipos de problemas, principalmente para redes grandes, rápido treinamento, além de exigir modesta quantidade de memória, quando comparada às outras técnicas avaliadas.

VG-RAM WNN é uma rede neural sem peso, cujo treinamento pode ser feito em um único passo. Os neurônios da rede VG-RAM WNN armazenam os pares de entrada-saída apresentados durante o treinamento (Souza et al., 2009). As sinapses indicam as conexões dos neurônios, e elas são representadas por valores binários. Na etapa de teste, a amostra de teste (entrada apresentada) é comparada com todas as entradas dos pares entrada-saída aprendidos pelos neurônios da VG-RAM WNN. A saída de cada neurônio é determinada pela saída do par entrada-saída, cuja entrada é a mais próxima da entrada apresentada, e a métrica usada é a distância de Hamming. Caso exista mais de um par com a menor distância, a saída do neurônio é escolhida aleatoriamente entre esses pares (Komati e Souza, 2002). As saídas dos neurônios são contadas e a classe mais votada é associada à amostra de teste.

Das técnicas apresentadas nessa subseção, as únicas que podem ser consideradas possuírem aprendizado incremental, segundo os critérios de Langley (1995) (apresentados no Capítulo 1), são as redes RNPe, RNPI-EM, eMLP e EFuNN. Embora a RNPI seja apresentada em (Bhattacharyya et al., 2008) como incremental, ela armazena na memória todos os estados anteriores de conhecimento, o que viola uma das condições de Langley. Porém, como ela apresenta características semelhantes a RNP, a rede usada como base para a RNPe, a RNPI foi incluída nos experimentos exclusivos das técnicas com aprendizado incremental.

5.3.2 Medidas de Avaliação

Com a finalidade de avaliar as técnicas em cada parte dos experimentos, são utilizadas as medidas de acurácia, tamanho da estrutura da técnica e o tempo necessário para realizar a classificação. Enquanto a primeira medida quantifica a eficácia dos algoritmos na tarefa de classificação, as duas últimas medidas são aproximações da complexidade dos algoritmos (Platt, 1991). Essas métricas são apresentadas a seguir:

1. **Acurácia**: calcula a porcentagem de amostras de teste que um classificador rotulou corretamente. Esse valor é a razão entre o número de amostras rotuladas corretamente pelo número total de amostras, multiplicada por 100%. Ele é limitado por uma faixa entre 0% a 100%. Quanto maior o valor, melhor a eficácia da técnica para realizar a classificação. A soma da acurácia com a taxa de erro de um algoritmo é igual a 1 (ou 100%). Essa métrica é amplamente utilizada em tarefas de classificação (Demšar, 2006) e, por isso, foi escolhida.

2. **Tamanho da estrutura**: essa métrica é baseada na usada em (Platt, 1991) para medir a complexidade de redes neurais. Em (Platt, 1991), esse valor é calculado contando o número de pesos ou parâmetros que uma técnica necessita para fazer uma tarefa. A medida utilizada nos experimentos desse trabalho também considera a precisão da representação dos valores dos pesos e dos parâmetros. Essa alteração é para tornar mais justa a comparação do número de parâmetros das outras técnicas com o número da VG-RAM WNN, cujo cada parâmetro é representado por somente um *bit*. Sendo assim, o tamanho da estrutura é medido em termos do número de bits necessários para representar a estrutura de cada técnica. Em outras palavras, essa métrica mede a quantidade de memória que a técnica necessita para realizar uma tarefa de classificação. Todas as outras técnicas utilizadas nos experimentos apresentam boa precisão definindo os parâmetros como *float*, que, na linguagem *C*, possui 32 *bits*. Quanto menor o tamanho da estrutura de uma técnica, maior a chance dela possuir uma complexidade menor.

3. **Tempo**: outra medida utilizada para estimar a complexidade de uma técnica é o tempo necessário para realizar a classificação dos dados. Maiores tempos para classificação sugerem uma maior complexidade da técnica. A unidade usada para essa medida foi milissegundos. Todos os algoritmos testados estão implementados em *C*, e os tempos foram obtidos em um computador que foi isolado e possui um processador Intel Dual Core 2,30 GHz e 4 GB de memória RAM.

Além dessas, outras três métricas foram usadas especialmente para verificar a estabilidade e plasticidade de algumas técnicas incrementais. Elas são descritas a seguir.

Métricas de estabilidade e plasticidade

Três métricas são propostas neste trabalho para avaliar os graus de estabilidade e plasticidade de uma técnica incremental. No entanto, antes de apresentá-las, é explicado o procedimento necessário para aplicá-las.

Para calcular essas métricas, é necessário calcular, antes, a quantidade de informação inicial de uma classe que um classificador retém após ser treinado com todas as outras classes. Essa grandeza será indicada a seguir por A. Outra medida realizada é a quantidade de informação de uma classe aprendida por um classificador, após terem sido usadas todas as outras classes para treinamento. Essa grandeza é representada a seguir por B.

Inicialmente, cada base de dados é dividida em $|C|$ subconjuntos, onde $|C|$ é o número de classes da base de dados, e n_i, $i = 1, \ldots, |C|$, é o número de amostras em cada subconjunto, com cada subconjunto contendo somente amostras de uma única classe. Essa divisão das bases de dados evita o aparecimento de padrões similares em diferentes subconjuntos, podendo tornar difícil saber se o classificador aprendeu informação nova ou se simplesmente já era conhecimento obtido em um momento anterior.

Depois de dividir a base de dados como descrito acima, o classificador é treinado com o primeiro subconjunto S_1 e testado com o mesmo subconjunto. O número de amostras corretamente classificadas do subconjunto S_1 é então contado, e o valor é indicado por $A_{1,1}$, no qual o primeiro índice indica o número de subconjuntos usados para treinamento até então, e o segundo índice é o subconjunto usado para teste. A seguir, os subconjuntos S_i, $i = 2, \ldots, |C| - 1$, são também usados para treinamento e o classificador é testado com o subconjunto $S_{|C|}$. O número de amostras corretamente classificadas é referido por $B_{|C|-1,|C|}$, e o significado dos índices é o mesmo mencionado antes. Depois, o subconjunto $S_{|C|}$ é também usado para treinamento, e os subconjuntos S_1 e $S_{|C|}$ são usados para teste. O valor de $A_{|C|,1}$ é o número de amostras corretamente classificadas do subconjunto S_1, e $B_{|C|,|C|}$ é o número de amostras corretamente classificadas de $S_{|C|}$.

Esse procedimento é repetido $|C|$ vezes, até todos os subconjuntos serem usados uma vez como o primeiro e uma vez como o último subconjunto de treinamento. Em outras palavras, esse procedimento de treinamento e teste é repetido até que os valores de $A_{1,i}$, $A_{|C|,i}$, $B_{|C|-1,i}$ e $B_{|C|,i}$, $i = 1, \ldots, |C|$, sejam calculados.

A Figura 5.8 ilustra os cálculos das grandezas A e B para uma base de dados com duas classes ($|C| = 2$). No início, a base de dados é dividida em 2 partes, onde a parte S_1 é a primeira parte usada para treinamento. Após o treinamento com S_1, essa mesma parte é usada para teste e é contado o número de amostras corretamente classificadas, esse valor é referido por $A_{1,1}$. Depois, a parte S_2 é utilizada como teste e é contado o número de amostras

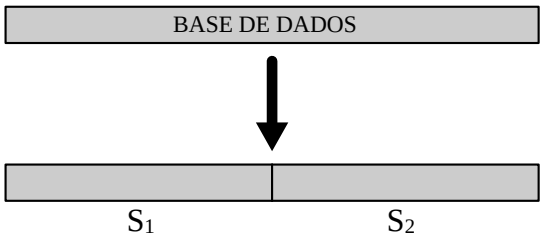


Figura 5.8: Ilustração da divisão de uma base de dados de duas classes em dois subconjuntos (S_1 e S_2) para o cálculo das grandezas A e B. Os dois subconjuntos são usados para treinamento, sendo S_1 o primeiro a ser usado. Os valores de A são as amostras corretamente classificadas de S_1 antes e depois de usar S_2 para treino. Os valores de B são as amostras corretamente classificadas de S_2 antes e depois de usar S_2 para treino.

corretamente classificadas dessa parte. Esse valor é referido por $B_{1,2}$. Em seguida, a parte S_2 também é usada para treinamento (ou seja, o classificador é treinado com S_1 e S_2, nessa ordem) e as partes S_1 e S_2 são usadas para teste. O número de amostras classificadas corretamente de S_1 é identificado pelo valor de $A_{2,1}$, enquanto o número de amostras classificadas corretamente de S_2 é indicado por $B_{2,2}$. Após esses cálculos, a ordem das partes de treinamento é invertida. Agora S_2 é a primeira parte de treinamento e S_1 é a segunda parte. Com isso, os valores de $A_{1,2}$ e $B_{1,1}$ são obtidos quando o classificador é treinado somente com S_2, e $A_{2,2}$ e $B_{2,1}$ são calculados quando são usadas as duas partes para treinamento. Os valores de A são obtidos para o primeiro subconjunto de treinamento, e os valores de B para o último subconjunto.

O valor de $A_{1,i}$ é o número de amostras que o classificador aprende do subconjunto S_i, e $A_{|C|,i}$ é o número de amostras reconhecidas do subconjunto S_i depois dos dados das $|C|$ classes terem sido usados para treinamento. Por outro lado, $B_{|C|-1,i}$ é o número de amostras do subconjunto S_i reconhecidas pelo classificador antes dele ser treinado com o subconjunto S_i. E $B_{|C|,i}$ é o número de amostras aprendidas depois dele ter sido treinado com S_i. Em outras palavras, a grandeza A é usada para medir a quantidade de informação que o classificador reteve da primeira classe usada para treinamento após terem sido usadas todas as classes do problema para treinamento. Enquanto a grandeza B é utilizada para avaliar a quantidade de informação que o classificador aprendeu da última classe usada para treinamento.

As métricas apresentadas abaixo medem as seguintes informações: primeiro, a quantidade de informação média retida para cada classe depois de aprender as demais classes e, segundo, a quantidade de informação média aprendida para cada classe depois que as outras classes foram aprendidas.

- **Retenção (*R*)**: mede o grau de estabilidade de um classificador, isto é, a habilidade de

um classificador reter conhecimento antigo quando uma quantidade nova de informação é apresentada.

O valor da Retenção R de um classificador com relação a uma base de dados é obtido usando a Equação 5.12:

$$R = \frac{1}{|C|} \sum_{i=1}^{|C|} R_i \times 100\%, \tag{5.12}$$

onde

$$R_i = \begin{cases} \frac{A_{|C|,i}}{A_{1,i}}, & \text{se } A_{1,i} > 0, \\ 0, & \text{caso contrário.} \end{cases}$$

A métrica Retenção calcula a média das razões entre o número de amostras reconhecidas de cada classe antes e depois de apresentar todas as classes para o classificador. O valor de R pode variar de 0 a 100%. Quanto maior for o valor de R, melhor a habilidade do classificador para reter conhecimento antigo.

- **Inovação (*I*)**: mede o grau de plasticidade de um classificador, isto é, a habilidade de um classificador aprender novo conhecimento.

  O cálculo do valor da Inovação I é realizado usando a Equação 5.13:

$$I = \frac{1}{|C|} \sum_{i=1}^{|C|} I_i \times 100\%, \tag{5.13}$$

  onde

$$I_i = \begin{cases} \frac{B_{|C|,i} - B_{|C|-1,i}}{n_i - B_{|C|-1,i}}, & \text{se } n_i - B_{|C|-1,i} > 0, \\ 1, & \text{caso contrário.} \end{cases}$$

  A métrica Inovação calcula a média das razões entre o número de amostras reconhecidas de cada classe antes e depois das amostras serem usadas para treinamento. O valor de I pode variar de 0 a 100%. Quanto maior for o valor de I, maior a capacidade do classificador para aprender novas informações.

- **Média harmônica entre Retenção e Inovação (*H*)**: avalia o compromisso entre estabilidade e plasticidade de um classificador para uma base de dados. Esse valor é medido pela Equação 5.14:

$$H = \frac{2RI}{R+I} \times 100\%. \tag{5.14}$$

  Quanto maior o valor de H, melhor a habilidade do classificador para obter o compromisso entre estabilidade e plasticidade. O valor de H pode variar de 0 a 100%.

Um resultado sem diferença entre os valores de Retenção e Inovação para uma base de dados, isto é, $R = I$, sugere o método incremental ser insensível à ordem da apresentação das

classes de uma determinada base de dados (embora isso não implicar ser insensível à ordem das amostras).

5.3.3 Validação Cruzada

Em problemas do mundo real, a quantidade de dados disponível para treinar e avaliar o desempenho de uma técnica é limitada. Com essa limitação, é difícil obter uma estimativa confiável do desempenho de uma técnica. Uma estratégia muito usada na literatura para contornar esse problema é o *k-fold cross-validation* (Arlot e Celisse, 2010).

Em *k-fold cross-validation*, a base de dados é dividida aleatoriamente em k partições (*folds*) mutuamente exclusivas, com aproximadamente o mesmo número de amostras. Em seguida, $k-1$ partições são usadas para treinar uma técnica e uma partição é usada para testar o desempenho da técnica. Esse processo é repetido k vezes, de forma a cada partição ser usada uma vez para testar a técnica. Ao final, o desempenho médio da técnica para cada métrica é calculado. A repetição do processo permite atenuar o efeito de amostras de treinamento não representativas e obter uma medida de desempenho mais confiável.

Em muitos experimentos na literatura é usado o *10-fold cross-validation*, pois, segundo Witten et al. (2011), numerosos testes sobre várias bases de dados, com diferentes técnicas de aprendizado, têm mostrado que 10 é um número apropriado para conseguir a melhor estimativa de desempenho.

Porém, mesmo usando o *10-fold cross-validation*, existe ainda a possibilidade de haver uma ou mais partições com proporções de classes muito diferentes do que na base de dados original. Isso poderia acarretar ter a maioria das amostras de uma classe na partição de teste e poucas amostras nas partições de treino. Para evitar tal situação, é realizada a estratificação das partições. A estratificação é um procedimento para obter a distribuição das classes em cada partição aproximadamente igual à distribuição delas na base de dados de origem. Com isso é obtido o *10-fold cross-validation* estratificado, que foi o utilizado nos experimentos.

Quando o valor de k do *k-fold cross-validation* é igual ao número de amostras da base de dados, é obtido o procedimento do *leave-one-out*. As vantagens desse procedimento é usar a maior quantidade possível de dados para treinamento, o que possivelmente aumenta o desempenho do mesmo, além de não envolver amostragem aleatória. No entanto, as desvantagens são sua alta variância, podendo guiar para uma estimativa não confiável do desempenho, e seu alto custo computacional para grandes quantidades de dados (Witten et al., 2011).

5.3.4 Calibração dos classificadores

A maioria dos classificadores usados nos experimentos possuem parâmetros que necessitam ser ajustados (calibrados) para alcançar um nível de desempenho mais elevado para cada base de dados. Antes de realizar qualquer experimento, é necessário calibrar esses parâmetros usando parte dos dados. O ajuste dos parâmetros é feito utilizando uma parte dos dados para treinar o algoritmo e outra parte para testar o desempenho da técnica com os parâmetros usados no treinamento. Essa segunda parte dos dados é chamada de dados de validação.

Os passos de treino e teste do algoritmo são repetidos para diferentes combinações de parâmetros, e a combinação retornando o melhor resultado de acordo com uma métrica é escolhida. Witten et al. (2011) recomendam dados de treino e validação mutuamente exclusivos: o conjunto de validação deve ser diferente do conjunto de treino para obter bom desempenho na otimização.

Para calibrar os algoritmos utilizados nos experimentos, cada base de dados foi dividida em 10 partes estratificadas, sendo sete usadas para treino e duas para validação. Na calibração, buscou-se maximizar o valor da métrica acurácia. Em caso de duas ou mais calibrações com a mesma acurácia, foi selecionada aquela que obteve o menor tamanho da estrutura do algoritmo.

Diferentes valores de parâmetros foram testados para cada classificador sobre cada base de dados. Para a RNPI e a RNP, foram avaliados diferentes valores para a variância Gaussiana σ^2 (valores: 0,01; 0,05; 0,1; 0,2; 0,25; 0,3; 0,4; 0,5; 0,7; 0,8; 1; 5; 10 e 100). Os parâmetros calibrados em EFuNN foram número de funções de pertinência de entrada (2 e 3), limiar de sensibilidade para adicionar um novo neurônio (0,05; 0,1; 0,3; 0,5 e 0,8), número de neurônios para o procedimento de agregação (100; 500; 1000 e 2000), duas taxas de aprendizado (0,01; 0,05; 0,1; 0,2; 0,5 e 0,8), limiar de erro para atualização (0,3; 0,5 e 0,8) e o número de funções de pertinência de saída (2).

Para eMLP, foram calibrados o limiar de sensibilidade para adicionar um novo neurônio (0,3; 0,5; 1 e $\rightarrow \infty$), número de neurônios para o procedimento de agregação (100; 500; 1000 e 2000), duas taxas de aprendizado (0,01; 0,1; 0,2; 0,5 e 0,8) e limiar de erro para atualização (0,3; 0,5 e 0,8). A classificação de ambos EFuNN e eMLP é realizada usando a ativação do neurônio mais ativo na saída.

Para RNPI-EM, foram calibrados o valor de n (50; 100; 500; 1000 e $\rightarrow \infty$) e o limiar de corte de neurônios (1/50; 1/100; 1/300; 1/500; 1/1000 e 0). Os parâmetros calibrados para a RNPe foram o valor de n_{max} (100 e 1000), o valor inicial do efeito de campo φ_{init} (0,05; 0,1; 0,25; 0,5 e 0,8), o valor de υ (-0,2; 0,0; 0,2; 0,3 e 0,4), o de η (0,000 e 0,001) e o valor de $\tau \rightarrow \infty$. Também foi avaliado se as variâncias deveriam ou não serem atualizadas. O

treinamento dos algoritmos citados acima foi realizado passando somente uma vez os dados de treinamento pelas redes.

Para o número de vizinhos mais próximos (k) do kNN foram avaliados oito valores (1; 3; 5; 7; 9; 11; 13 e 15). Para o MLP, foi utilizada uma arquitetura com uma camada oculta, sendo calibrados o número de neurônios na camada oculta, o número de épocas de treinamento (50; 100; 150; 200; 250 e 300) e a taxa inicial de aprendizado (0,05). Para o número de neurônios na camada oculta, foram avaliados quatro valores proporcionais ao número de atributos d de cada base de dados ($d/2$, d, $3d/2$ e $2d$). Para as bases *CNAE-9*, *WebKB-4* e *Reuters-8*, que possuem número elevado de atributos, foram avaliados os valores de 50, 100, 200 e 300 neurônios na camada oculta para evitar uma arquitetura muito grande. O número de neurônios na camada de entrada e na camada de saída foi igual ao número de atributos e ao número de classes de cada base de dados, respectivamente.

Para o VG-RAM WNN, foram otimizados o número de neurônios (2; 4; 9; 16; 36; 64; 100; 144; 196; 256; 324; 400; 484; 576; 676; 784 e 900) e de sinapses para cada base de dados. O número de sinapses foi proporcional ao número de atributos d de cada base de dados, sendo avaliados o número de sinapses no intervalo $d/10 \leq n\_s \leq 10d$, onde n_s é o número de sinapses pertencente ao conjunto de valores 2^i, para $i = 1, \ldots, 11$. Finalmente, Rocchio não tem parâmetros para serem otimizados.

5.3.5 Testes estatísticos

Comparação de múltiplos algoritmos sobre várias bases de dados

Para comparar estatisticamente múltiplos algoritmos sobre várias bases de dados foi utilizado o teste não paramétrico de Friedman (Friedman, 1937). A vantagem dos testes não paramétricos é não ser necessário fazer uma suposição da distribuição dos valores a serem analisados.

O teste de Friedman ordena os algoritmos para cada base de dados de acordo com o desempenho deles. O algoritmo com o melhor resultado consegue o *rank* 1, o segundo melhor *rank* 2, e assim por diante. Em caso de empate entre dois ou mais algoritmos, é associado o *rank* médio a eles (Demšar, 2006). Por exemplo: caso três algoritmos obtenham o melhor resultado para uma base de dados, então o *rank* deles será $(1+2+3)/3 = 2$.

Considere Ra_j o *rank* médio do j-ésimo algoritmo (de um total de k algoritmos) sobre n bases de dados. O teste de Friedman assume a hipótese nula de que todos os algoritmos são equivalentes e os ranks médios deles devem ser iguais (Demšar, 2006). Essa hipótese é

verificada usando a distribuição χ_F^2:

$$\chi_F^2 = \frac{12n}{k(k+1)} \left[\sum_{j=1}^{k} Ra_j^2 - \frac{k(k+1)^2}{4} \right]. \tag{5.15}$$

No entanto, em (Iman e Davenport, 1980) foi mostrado que a distribuição χ_F^2 é muito conservativa, e Iman e Davenport propuseram a distribuição F de Snedecor com $k-1$ e $(k-1)(n-1)$ graus de liberdade:

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1)-\chi_F^2}. \tag{5.16}$$

Caso a hipótese nula seja rejeitada, um segundo teste é realizado para comparar os algoritmos entre si. O segundo teste realizado pode ser o teste de Nemenyi (Nemenyi, 1963). Esse teste calcula um valor chamado de diferença crítica (DC), em função da quantidade de bases de dados usadas para avaliação e o número de classificadores avaliados, além do nível de significância do teste:

$$DC = q_\alpha \sqrt{\frac{k(k+1)}{6n}}. \tag{5.17}$$

O valor de q_α depende do nível de significância do teste, e uma tabela de valores é disponibilizada em (Demšar, 2006). O desempenho de dois classificadores é significativamente diferente se a diferença entre seus correspondentes *ranks* médios é maior ou igual a DC (Demšar, 2006). Um nível de significância de 5% foi usado para ambos os testes nos experimentos.

Nos testes estatísticos realizados com mais de uma base de dados, os valores das variâncias das medidas de desempenho não são levadas em consideração (Demšar, 2006). As considerações para realizar esse tipo de teste são os resultados medidos serem confiáveis (obtido realizando experimentos suficientes sobre cada base de dados), todos os algoritmos avaliados sobre as mesmas partições e uma quantidade mínima de cinco bases de dados seja usada no teste (Demšar, 2006). O resultado para cada base de dados é considerado como uma amostra no teste estatístico.

Uma representação gráfica dos resultados dos testes, apresentada em (Demšar, 2006), foi utilizada para melhor visualização dos resultados. A Figura 5.9 exibe os resultados de uma suposta análise de quatro métodos hipotéticos, chamados aqui de A1, A2, A3 e A4. A linha superior do diagrama é o eixo no qual é indicado o *rank* médio dos métodos. O eixo é orientado no sentido dos menores *ranks* (melhores) estarem do lado direito.

Os grupos de métodos não significativamente diferentes são conectados por uma linha mais espessa, e a DC entre dois algoritmos para serem considerados diferentes é indicada

em cima do gráfico. Analisando o diagrama da Figura 5.9, verifica-se não haver diferenças estatísticas entre o trio A1, A4 e A2 e entre o par A2 e A3. Também pode ser notado A1 e A4 serem estatisticamente superiores a A3.

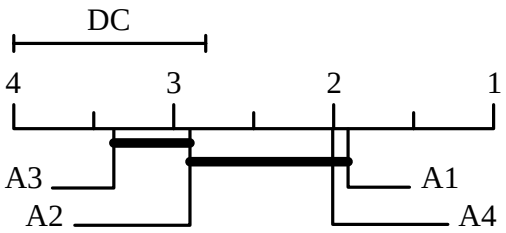


Figura 5.9: Comparação de quatro métodos entre si através dos seus ranks médios. DC significa a diferença crítica entre dois classificadores para serem considerados estatisticamente diferentes. Grupos de classificadores não significativamente diferentes são conectados por uma linha espessa (Demšar, 2006).

Testes de dependência estatística entre variáveis

Alguns testes de dependência estatística foram realizados com intuito de verificar se existe alguma relação entre os resultados obtidos nos experimentos e as características das bases de dados. Na estatística, dependência é referente a qualquer relacionamento estatístico entre duas variáveis aleatórias ou dois conjuntos de dados.

A medida mais popular de dependência entre duas variáveis é o coeficiente de correlação de Pearson. Esse valor é obtido dividindo-se a covariância entre duas variáveis pelo produto de seus desvios padrões. O valor desse coeficiente pode variar de $+1$ (quando há um perfeito relacionamento linear positivo entre as variáveis, ou seja, elas se movem na mesma direção) a -1 (quando há um perfeito relacionamento linear negativo, isto é, quando as variáveis se movem em direções contrárias). Variáveis independentes possuem um coeficiente de correlação igual a 0, entretanto, a recíproca nem sempre é verdadeira (Leon-Garcia, 1993).

No entanto, para tornar válido o cálculo do coeficiente de Pearson, é necessário satisfazer algumas condições, tais como as distribuições das variáveis serem aproximadamente iguais à Gaussiana e relacionamento linear entre variáveis, as quais nem sempre podem ser garantidas.

Para contornar essa desvantagem, pode-se usar o coeficiente de correlação de Kendall (Kendall, 1938). Para o cálculo desse coeficiente, não é necessário conhecer as distribuições

das variáveis e nem supor a existência de um relacionamento linear entre elas. Isso torna essa abordagem mais robusta do que o coeficiente de Pearson.

Na estatística, a força de relacionamento normalmente está associada à tendência de duas variáveis se moverem na mesma direção ou em direções opostas. O coeficiente de Kendall mede essa tendência de forma direta e de fácil entendimento (Noether, 2008).

Sejam (x_1,y_1), (x_2,y_2), ..., (x_n,y_n) n observações conjuntas das duas variáveis X e Y. Então, são contados todos os pares de observações concordantes e todos os pares discordantes, de um total de $n(n-1)/2$ pares. Um par é concordante se $(x_j - x_i)$ e $(y_j - y_i)$ possuem o mesmo sinal, para $i \neq j$, e é discordante se os sinais são opostos. Se houver empates, isto é, $x_j = x_i$ ou $y_j = y_i$, o par não é considerado concordante nem discordante.

Uma forma simples de medir a força de relacionamento entre duas variáveis é calcular a diferença entre a quantidade de pares concordantes e o número de pares discordantes. Um valor positivo elevado indica que as variáveis X e Y possuem uma tendência elevada para se moverem na mesma direção, enquanto um grande valor negativo sugere alta possibilidade das variáveis se moverem em direções opostas. Para normalizar o valor da diferença para a faixa entre -1 e $+1$, essa diferença é dividida por $n(n-1)/2$, e assim é obtido o valor τ do coeficiente de Kendall para o caso em que não ocorrem empates nos dados:

$$\tau = \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} \text{sgn}(x_j - x_i)\text{sgn}(y_j - y_i)}{n(n-1)/2}, \tag{5.18}$$

onde sgn(a) retorna o sinal de a (+1 ou -1).

Para casos em que existam empates nos dados, uma variação na forma de calcular o coeficiente é proposta para manter a faixa de valores entre -1 e $+1$ (Yau, 2012):

$$\tau = \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} \text{sgn}(x_j - x_i)\text{sgn}(y_j - y_i)}{\sqrt{n_x}\sqrt{n_y}}, \tag{5.19}$$

onde n_x e n_y são os números de pares de dados sem empates das variáveis X e Y, respectivamente.

O coeficiente de Kendall é uma medida natural do relacionamento entre X e Y, tal que a razão entre a probabilidade de ocorrer concordância $P(C)$ e de ocorrer discordância $P(D)$ é obtida por (Kvam e Vidakovic, 2007):

$$\frac{P(C)}{P(D)} = \frac{1+\tau}{1-\tau}. \tag{5.20}$$

O coeficiente de correlação de Kendall é aplicado para testar a hipótese de independência de variáveis, sendo a hipótese nula a não existência de correlação entre as variáveis. Para pequenas amostras ($4 \leq n \leq 10$), o teste estatístico é realizado por meio de tabelas especiais. Para amostras maiores ($n > 10$), é usada uma aproximação Gaussiana para a distribuição τ, e, caso:

$$|\tau| > z_{1-\alpha/2}\sqrt{\frac{2(2n+5)}{9n(n-1)}}, \tag{5.21}$$

a hipótese de independência é rejeitada (Kvam e Vidakovic, 2007), sendo u o percentil da distribuição normal z_u, e α o nível de significância do teste.

Testes de dependência estatística foram realizados entre os resultados de acurácia de cada uma das técnicas e cada uma das medidas de complexidade das bases de dados (informadas na Tabela 5.1). Esses testes podem indicar alguma relação entre os resultados obtidos e as características das bases de dados. Um nível de significância de 5% foi usado nos testes realizados nos experimentos.

5.4 Conclusão

Este capítulo apresentou as bases de dados, as técnicas, as métricas e a metodologia a serem empregados nos experimentos. Também foram descritos dois testes estatísticos para verificação de hipóteses nos experimentos. No próximo capítulo, são realizados os experimentos e análise dos resultados.

Capítulo 6

Resultados Experimentais

Neste capítulo, o modelo de aprendizado incremental proposto, a RNPe, é avaliado empiricamente. Na Seção 6.1, são avaliados alguns dos procedimentos propostos para o modelo conseguir uma estrutura reduzida mantendo uma qualidade satisfatória de resposta.

Dois métodos para analisar a estabilidade e plasticidade da RNPe, e de algumas redes neurais incrementais, são aplicados na Seção 6.2. Nessa, é também realizado um estudo sobre as bases de dados para identificar quais são as características que interferem no compromisso entre estabilidade e plasticidade.

A RNPe é comparada com outras técnicas de classificação em relação ao desempenho e complexidade na Seção 6.3. Experimentos com comitê de redes neurais e aprendizado semi-supervisionado são realizados na Seção 6.4. Por fim, uma conclusão desse capítulo é feita na Seção 6.5.

6.1 Avaliação dos procedimentos do modelo proposto

A fim de verificar se os procedimentos propostos para a RNPe atuam de forma satisfatória, vários experimentos são realizados nessa seção. A proposta dessa seção é avaliar modelos que possuem diferentes combinações dos procedimentos apresentados na Seção 4.1, partindo de um modelo mais simples que, de acordo com os critérios de Langley (1995), não é considerado incremental, até o modelo incremental proposto neste trabalho.

A meta do modelo proposto é produzir resultados tão bons quanto aqueles do modelo não incremental, mas com a vantagem de possuir uma estrutura menor, tal que consiga obter um compromisso razoável entre desempenho e tamanho da estrutura do modelo. Também é

esperado que ele satisfaça a todos os critérios propostos por Langley (1995), além de possuir diversas características desejadas para um algoritmo de aprendizado incremental. Será também verificado nos experimentos se é válida a hipótese inicial de poder representar distribuição dos dados através de uma distribuição semelhante a Gaussiana sem haver perda significante de informação.

Os experimentos, resultados e análises são detalhados na Subseção 6.1.1. As conclusões dessa seção de experimentos encontram-se na Subseção 6.1.2.

6.1.1 Experimentos e resultados

Nos experimentos foram utilizadas duas medidas para avaliar os métodos analisados: uma medida de desempenho e outra do tamanho da estrutura de cada modelo, ou seja, o espaço ocupado em memória pelo modelo. O desempenho foi medido nos experimentos pela métrica acurácia, e o tamanho do modelo, pelo número de parâmetros contidos na estrutura do modelo. O tempo de classificação das amostras não foi utilizado como uma medida de complexidade do modelo, por entender que os cálculos realizados na maioria dos modelos avaliados possuírem a mesma natureza, tal que a diferença entre os tempos de classificação de diferentes modelos está relacionada principalmente ao número de parâmetros.

Os procedimentos avaliados foram adição de neurônios na rede, união de neurônios, atualização dos parâmetros média (μ), variância (σ^2) e peso (ω) dos neurônios, ajuste do efeito de campo (φ^2) dos neurônios e remoção de neurônios. A classificação é realizada associando para cada amostra a classe do neurônio mais ativo. E, a não ser quando informado do contrário, os procedimentos de treinamento e classificação são realizados conforme descritos na Seção 4.1.

Os parâmetros de cada modelo foram calibrados usando 90% de cada base de dados, onde 70% desse montante foram usados para treinar os modelos, e os 20% restantes para validar os parâmetros. Uma vez calibrado cada modelo, os experimentos foram realizados aplicando o método 10 *fold cross-validation* sobre as bases de dados, e calculados a média da acurácia e do número de parâmetros obtidos por cada modelo para cada base de dados.

Os parâmetros calibrados foram o valor de n_{max} (100 e 1000) quando usados os procedimentos de adição, atualização (μ, σ^2 e ω) e remoção de neurônios, e o valor inicial do efeito de campo φ_{init} (0,05; 0,1; 0,25; 0,5 e 0,8) quando usados os procedimentos de atualização de μ, σ^2 e ω e ajuste do efeito de campo. Além disso, foram calibrados o valor de η (0 e 0,001) para o procedimento de ajuste do efeito de campo, o valor de υ (-0,2; 0,0; 0,2; 0,3 e 0,4) quando incluído o procedimento de união de neurônios, e o parâmetro τ (10) para o procedimento de remoção de neurônios.

No total, 10 combinações de procedimentos foram avaliadas:

- **M1**: o modelo possui somente o procedimento de adicionar neurônios, nesse caso adicionado para cada amostra usada como treinamento. Esse modelo obtém os mesmos resultados da técnica vizinho mais próximo, ou o 1-NN. O parâmetro a alterar nesse modelo é o vetor média (μ) de cada neurônio, e assim eles são os únicos considerados na contagem dos parâmetros. Não houve calibração de parâmetros. Por esse modelo permitir retornar a um estado anterior de conhecimento, através da remoção de uma amostra de treinamento de sua estrutura, ele não é considerado incremental segundo os critérios de (Langley, 1995);

- **M2**: também só possui o procedimento de adicionar neurônios, porém os neurônios só são adicionados quando a amostra de treinamento é classificada em uma classe errada ao passar pela rede proposta. Novamente, só é considerado o vetor média de cada neurônio na contagem dos parâmetros. Também não há calibração de parâmetros;

- **M3**: semelhante ao M2, porém com inclusão do procedimento de união de neurônios, cujos critérios são avaliados para cada amostra de treinamento. Nesse modelo, assim como nos demais, além do vetor média, os outros parâmetros são considerados na contagem de parâmetros;

- **M4**: os parâmetros dos neurônios (μ, σ^2 e ω) são atualizados para cada amostra, os neurônios são adicionados quando uma amostra de treinamento é classificada em uma classe errada, e os critérios para união dos neurônios são avaliados para cada amostra de treinamento. Diferentemente dos outros modelos avaliados, esse utiliza a Gaussiana (Equação 4.2) como função de transferência dos neurônios, em vez da função proposta (Equação 4.5);

- **M5**: Semelhante ao M4, porém usa a função de transferência proposta (Equação 4.5). Também foi verificado se as variâncias deveriam ser atualizadas ou não;

- **M6**: é o M5 acrescido do procedimento de ajuste do efeito de campo do neurônio mais ativo quando uma amostra é classificada em uma classe errada. Para verificar o efeito de tal ajuste, o valor η foi fixo em 0,001, sendo η a taxa de mudança do efeito de campo. Foi verificado se deveria ser feita a atualização das variâncias;

- **M7**: semelhante ao M6, porém o valor de η é ajustado (0 e 0,001). M7 busca unificar os modelos M5 e M6;

- **M8**: consiste no modelo M7, porém, diferentemente dos outros em que a classificação é realizada pelo neurônio mais ativo, aqui é realizada através da soma ponderada da

saída dos neurônios. Seja $O(x)$ a classe associada à amostra x, $|C|$ o número de classes e c_i a i-ésima classe ($c_i \in C$, onde C é o conjunto de classes). Então, a classificação através da soma ponderada é realizada pela Equação 6.1:

$$O(x) = \arg\max_{i=1}^{|C|} P(c_i|x), \tag{6.1}$$

onde:

$$P(c_i|x) = \sum_{j=1}^{K_i} \omega_{i,j}\, \hat{f}(x, \mu_{i,j}, \Sigma_{i,j}, \varphi_{i,j}), \tag{6.2}$$

sendo K_i o número de neurônios associados a classe c_i, $\hat{f}(x, \mu_{i,j}, \Sigma_{i,j}, \varphi_{i,j})$ a função da Equação 4.5, e $\omega_{i,j}$, $\mu_{i,j}$, $\Sigma_{i,j}$ e $\varphi_{i,j}$ peso, vetor média, matriz de covariância e efeito de campo do j-ésimo neurônio da classe c_i, respectivamente;

- **M9**: o mesmo que M7, com procedimento adicional de remoção de neurônios;
- **M10**: vetor média e variância de cada classe são calculados antes de treinar o modelo M7. O procedimento para calcular esses parâmetros é o descrito na Seção 4.1.4. A atualização ou não das variâncias também foi testada aqui.

Os resultados obtidos em cada modelo para 20 bases de dados são mostrados nas Tabelas 6.1 e 6.2, sendo os valores das acurácias mostrados na Tabela 6.1 e os números de parâmetros na Tabela 6.2. Cada linha de cada tabela é referente a uma base de dados e cada coluna a um modelo avaliado. As últimas duas linhas das Tabelas 6.1 e 6.2 informam o valor médio e a média dos desvios padrões (DPs) das acurácias e do número de parâmetros, respectivamente, para cada modelo sobre todas as bases de dados. Os valores das acurácias estão em porcentagem e os valores do número de parâmetros estão em $\log_{10}$ do número de *bits*, para permitir uma melhor visualização dos dados. A diferença de uma unidade entre os tamanhos dos modelos indica ser o tamanho de um deles 10 vezes maior que do outro. O valor de acurácia mais elevado e o menor número de parâmetros para cada base de dados estão realçados em negrito. Quanto maior o valor da acurácia, maior a capacidade do modelo de reconhecer uma amostra não vista durante o treinamento e, quanto menor o número de parâmetros, menor o tamanho do modelo. A linha "Média" da Tabela 6.2 informa o $\log_{10}$ da média do número de parâmetros, e não a média dos logaritmos do número de *bits*. O menor valor da média dos desvios padrões para cada métrica está realçado em negrito. A linha "Média dos DPs" da Tabela 6.2 informa o $\log_{10}$ da média dos desvios padrões do número de parâmetros.

O modelo M1 atua como se fosse a técnica de vizinho mais próximo, que é simples e popular na literatura, e apresenta desempenho satisfatório (acurácia) em várias tarefas de classificação. A desvantagem está em armazenar todas as amostras de treinamento em sua estrutura.

Tabela 6.1: Resultados obtidos pelos modelos para a métrica acurácia. Valores em porcentagem. O maior valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Bases de dados	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
Balance	**95,04**	90,08	89,60	7,85	91,51	90,08	91,51	90,40	90,87	92,47
Blood	69,11	63,24	58,17	76,06	62,16	76,73	76,73	76,73	76,73	**78,20**
Iris	96,00	90,67	95,33	66,67	**97,33**	**97,33**	**97,33**	**97,33**	**97,33**	**97,33**
Haberman	66,72	60,81	73,87	**73,88**	**73,88**	73,87	73,87	73,87	73,87	73,86
Car	**89,41**	82,12	83,97	28,27	81,25	81,49	81,25	74,31	79,81	84,38
CNAE-9	85,28	77,78	90,46	15,00	**92,22**	92,13	**92,22**	**92,22**	**92,22**	**92,22**
WebKB-4	64,06	59,09	81,26	57,42	82,12	**82,28**	**82,28**	**82,28**	**82,28**	82,21
Wine	73,63	75,23	75,23	**96,60**	96,05	96,05	96,05	96,05	96,05	95,49
Reuters-8	94,61	90,38	**95,15**	57,88	94,57	94,75	94,75	94,75	94,75	94,89
Yeast	53,10	48,45	52,69	7,29	55,32	54,99	54,99	53,84	55,12	**55,73**
Abalone	19,99	19,39	20,59	9,43	23,37	24,04	24,04	23,58	24,25	**25,16**
Heart	67,41	58,89	70,37	45,19	72,59	72,59	72,59	72,59	72,59	**84,07**
Sonar	**83,62**	80,31	74,88	53,38	80,29	79,81	80,29	80,79	80,76	73,60
Segmentation	**93,51**	89,74	63,77	41,65	90,09	90,09	90,09	91,47	89,65	**93,51**
Texture	**99,56**	97,93	92,16	57,07	97,69	97,62	97,69	97,60	97,58	88,67
Gaussian	72,58	67,14	57,50	61,08	56,78	**81,20**	**81,20**	**81,20**	**81,20**	79,88
Concentric	**98,48**	98,04	65,84	36,84	58,48	96,60	96,60	96,60	96,60	96,64
Spambase	86,46	80,42	83,55	73,92	82,98	85,66	85,66	87,20	85,44	**90,15**
Diabetes	66,93	62,11	62,11	61,22	**76,83**	76,18	**76,83**	**76,83**	**76,83**	75,53
Zoo	97,00	**99,09**	95,09	96,09	95,09	95,09	95,09	95,09	95,09	96,00
Média	78,62	74,54	74,08	51,14	78,03	81,93	82,05	81,74	81,95	**82,50**
Média dos DPs	4,52	4,10	4,78	6,02	3,90	**3,33**	3,42	3,97	3,44	3,77

O M2 se constitui numa modificação do modelo M1, onde só são adicionadas na estrutura do mesmo as amostras de treinamento previamente classificadas numa classe errada. Assim, ele busca reduzir o tamanho da estrutura evitando adicionar amostras redundantes ou similares. Comparando os dois com o auxílio da Tabela 6.2, verifica-se, para todas as bases de dados, o M2 obteve estruturas menores que M1. Em média, M2 obteve estrutura aproximadamente quatro vezes menor que M1, sendo para algumas bases, tais como *Iris*, *Texture* e *Concentric*, foi aproximadamente 10 vezes menor.

O valor da redução da estrutura está diretamente relacionado à acurácia alcançada para a base de dados: quanto maior a acurácia, menor a estrutura do modelo M2 em relação ao modelo M1. Por exemplo: para as bases *Iris*, *Reuters-8*, *Texture* e *Zoo*, M2 obteve valores de acurácia acima de 90% e estruturas aproximadamente 10 vezes menores que M1. Ou seja: para cada 10 amostras de treinamento, uma é adicionada a estrutura do M2 (uma é classificada errada), enquanto o M1 adiciona todas as 10 amostras na estrutura.

Por outro lado, para bases em que a acurácia foi menor, como a base *Abalone*, a redução da estrutura foi menor (para *Abalone* foi aproximadamente 1,20 menor do que a da M1). No entanto, o preço de uma estrutura menor foi a redução no desempenho. Para quase todas as bases de dados houve queda neste aspecto quando comparando os dois modelos. As

Tabela 6.2: Tamanhos das estruturas dos modelos medidos em número de pesos e parâmetros. Valores em $\log_{10}(bits)$. O menor valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Bases de dados	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
Balance	4,86	4,07	**3,05**	**3,05**	3,77	3,80	3,82	4,17	3,76	3,08
Blood	5,03	4,57	**2,97**	3,10	3,56	3,00	3,00	3,00	3,00	3,00
Iris	4,24	3,12	**3,05**	**3,05**	**3,05**	3,08	3,08	3,08	3,08	3,08
Haberman	4,42	4,01	**2,83**	**2,83**	**2,83**	2,87	2,87	2,87	2,87	2,87
Car	5,48	4,79	**3,29**	3,33	**3,29**	3,32	3,32	5,26	4,69	5,09
CNAE-9	7,43	6,96	**5,69**	6,05	**5,69**	**5,69**	**5,69**	**5,69**	**5,69**	**5,69**
WebKB-4	8,56	8,26	**5,89**	7,76	**5,89**	**5,89**	**5,89**	**5,89**	**5,89**	**5,89**
Wine	4,82	4,32	4,64	**3,56**	3,69	3,71	3,71	3,61	3,71	3,87
Reuters-8	8,82	7,89	**6,19**	7,36	**6,19**	**6,19**	**6,19**	**6,19**	**6,19**	**6,19**
Yeast	5,53	5,27	**3,77**	3,79	4,42	4,47	4,47	4,56	4,41	3,80
Abalone	6,08	5,99	**4,29**	5,67	4,36	4,38	4,38	4,31	4,37	4,46
Heart	5,00	4,61	**3,29**	**3,29**	**3,29**	3,30	3,30	3,30	3,30	3,30
Sonar	5,56	5,02	**3,90**	**3,90**	5,33	5,34	5,34	5,41	5,32	**3,90**
Segmentation	6,10	5,30	**3,96**	4,36	5,61	5,61	5,62	5,60	5,61	5,47
Texture	6,80	5,46	**4,46**	6,95	5,83	5,83	5,83	5,86	5,80	5,72
Gaussian	5,76	5,29	**2,90**	**2,90**	**2,90**	2,94	2,94	2,94	2,94	2,94
Concentric	5,16	3,80	**2,74**	**2,74**	**2,74**	2,78	2,78	2,78	2,78	2,78
Spambase	6,88	6,25	**3,88**	6,71	6,51	4,13	4,13	6,43	3,88	6,25
Diabetes	5,25	4,82	5,15	5,17	**3,12**	3,14	3,14	3,14	3,14	3,14
Zoo	4,67	3,76	**3,93**	3,97	3,96	3,97	3,97	3,97	3,97	3,93
Média	7,69	7,09	**5,15**	6,64	5,53	5,28	5,28	5,51	5,28	5,40
Média dos DPs	3,62	5,48	**2,54**	5,84	3,68	3,48	3,44	3,88	3,35	4,40

exceções foram as bases *Wine* e *Zoo*, conforme observado na Tabela 6.1. A melhora dos resultados sobre *Wine* e *Zoo* pode ter ocorrido porque amostras com ruído provavelmente não foram incluídas na estrutura do modelo (Jiang e Zhou, 2004). A acurácia média obtida sobre as bases de dados caiu 4%. Então, embora o procedimento adicional reduza o tamanho da estrutura, também reduz o desempenho.

O M2 obteve estrutura menor que M1, mas continuou com taxa de crescimento alta. Para as bases avaliadas, M2 cresce, em média, numa proporção de um neurônio para cada quatro amostras de treinamento. Para reduzir ainda mais a estrutura, o procedimento de união de neurônios é adicionado, e, assim, obtido o modelo M3. Em média, esse modelo conseguiu reduzir em mais de 80 vezes seu tamanho em relação a M2, e mais de 300 vezes em relação a M1. Para a maioria das bases de dados, reduziu para um neurônio por classe, que é a menor estrutura que a rede proposta pode alcançar.

Além de reduzir a estrutura, o M3 conseguiu manter o desempenho para algumas bases no mesmo patamar do obtido por M2, inclusive melhorando os desempenhos de algumas bases, como *Haberman*, *CNAE-9*, *WebKB-4*, *Reuters-8*, *Yeast* e *Heart*. Entretanto, para algumas bases, como *Segmentation* e *Concentric*, o valor da acurácia caiu. Verificou-se que as bases *Haberman*, *CNAE-9*, *WebKB-4*, *Reuters-8*, *Yeast* e *Heart* possuem em comum valor

elevado de N2 (N2 > 0,65) (ver Tabela 5.1, página 103), enquanto as bases *Segmentation* e *Concentric* possuem valor mais baixo (inferior a 0,3).

Como valores elevados de N2 indicam amostras da mesma classe dispersas em relação às de outras classes, é provável o ajuste nas médias e nas variâncias efetuadas no procedimento de união de neurônios conseguir ajustar essas distâncias, aumentando o desempenho. Para valores pequenos de N2, esse procedimento não surte muito efeito. Aplicando o teste de Kendall sobre a diferença entre os desempenhos de M2 e de M3, foi sugerido realmente existir relação significativa com o valor de N2, e, quanto maior o valor N3 (indicando maior quantidade de amostras próximas ao contorno de classe), o modelo M3 tende a apresentar melhor desempenho. Em suma, o procedimento de união de neurônios conseguiu obter redução significativa na estrutura da rede e melhora no desempenho de algumas bases de dados (um total de 10 bases), embora, na média, tenha reduzido o desempenho.

O M4 adiciona o procedimento de atualização de neurônios ao modelo M3, e utiliza a Gaussiana como função de transferência dos neurônios. Alguns dos resultados obtidos pelo M4 em relação à acurácia foram os menores encontrados dentre os modelos avaliados. A atualização incremental dos parâmetros dos neurônios causou instabilidade no modelo, prejudicando o desempenho do mesmo.

O desempenho de M4 tendeu a ser bem inferior aos dos outros modelos para problemas com número maior de classes, pois, em tais bases de dados, há uma variedade maior de possíveis rótulos para cada amostra, embora isso não seja aplicável a todas as bases de dados (a exemplo da base *Balance*, com três classes, cujo desempenho foi muito inferior). A consequência de muitas amostras serem classificadas erradas foi um número maior de neurônios adicionados ao modelo, e, assim, o tamanho da estrutura foi maior que o do modelo anterior em aproximadamente 30 vezes, como pode ser observado na Tabela 6.2.

O modelo M5 possui os mesmos procedimentos do M4, mas utiliza a função de transferência proposta neste trabalho. Como consequência, os resultados da classificação foram melhores, porque houve menor influência das incertezas dos valores estimados. Com acurácia maior na classificação, também foi obtida estrutura menor (em média, mais de 10 vezes inferior que M4), embora para algumas bases de dados M4 conseguisse estrutura menor, como nas bases *Balance*, *Segmentation* e *Sonar*.

Comparando com o modelo M3, foi observado que, em geral, M5 obteve ganhos nos resultados das acurácias. Olhando para as medidas de complexidade das bases de dados mostradas na Tabela 5.1 da página 103, e realizando o teste de Kendall, observa-se uma relação significativa com o valor de F1, tal que bases de dados com valor de F1 mais elevado (F1 > 2,0), como *CNAE-9*, *Wine*, *Segmentation* e *Texture*, tenderam a ter aumento maior no desempenho em relação a M3.

No entanto, para bases de dados com valor menor de F1 (F1 < 0,4) o ganho de desempenho tendeu a ser menor (em alguns casos, até caiu). Exemplos são as bases *Haberman*, *Car* e *Spambase*. Isto indica ser a atualização dos parâmetros, principalmente da variância, benéfica para tarefas em que existam atributos com poder discriminativo elevado.

Em relação ao tamanho da estrutura, M5 foi, em média, aproximadamente, duas vezes maior que M3. E, todas as vezes em que a arquitetura do M5 foi maior que M3, o desempenho também foi maior, com exceção da base *Spambase*, na qual houve aumento considerável da arquitetura (ver Tabela 6.2) acompanhada de uma ligeira queda no desempenho (Tabela 6.1). Isto mostra que, em geral, o aumento da estrutura foi acompanhado de aumento no desempenho, mas, para algumas bases de dados, como *Iris* e *Car*, o desempenho aumentou sem crescer o tamanho da estrutura. Embora o procedimento de atualização tenha aumentado um pouco o tamanho da estrutura do modelo, também aumentou a acurácia média, alcançando um valor superior ao do modelo M2.

O próximo a ser analisado é o M6, composto do modelo M5 acrescido do procedimento de ajuste do efeito de campo. Esse procedimento não alterou muito o valor das acurácias alcançadas para a maioria das bases de dados, com exceção das bases *Blood*, *Gaussian* e *Concentric*, cujos resultados aumentaram em mais de 10% em relação ao modelo M5. As bases *Gaussian* e *Concentric* têm em comum o fato de serem formadas por duas classes cujos centros são aproximadamente o mesmo, sendo a diferença entre as classes a dispersão das amostras (as amostras de uma classe são mais concentradas do que as da outra classe). Com o ajuste do efeito de campo, o modelo pôde se adaptar melhor ao espalhamento das duas classes, sendo para a classe com espalhamento maior das amostras, foi obtido um tamanho do efeito de campo também maior. Ainda que não se saiba a distribuição das amostras da base *Blood*, é razoável supor a distribuição das amostras das suas duas classes um pouco similar aos das outras duas, devido ao aumento de desempenho.

Embora o procedimento de ajustar o efeito de campo acrescente um novo parâmetro em cada neurônio, ele reduziu ligeiramente o número de neurônios do modelo em 3 bases de dados, e houve redução mais acentuada para a base *Spambase*, isto podendo ter influenciado de forma positiva no desempenho dessa base de dados. O procedimento de ajuste do efeito de campo mostrou ser interessante por apresentar, na média, melhora no desempenho e na redução do tamanho do modelo quando comparado à M5. Esse modelo também obteve acurácia média superior ao de M1, o da técnica do vizinho mais próximo.

Numa tentativa de unir os resultados de M5 com M6, o valor de η foi ajustado para os valores 0 e 0,001, e assim obtido M7. Como visto na Tabela 6.1, o desempenho para a maior parte das bases de dados foi o melhor encontrado entre os modelos M5 e M6. O número médio de parâmetros foi aproximadamente igual ao de M5 e menor ao de M6, e o

desempenho médio de M7 foi ligeiramente superior ao de M5.

Até o momento, todos os modelos classificavam uma amostra utilizando a saída do neurônio mais ativo. Porém, no MMG e na RNP é utilizada a soma ponderada das saídas dos *kernels*/neurônios e a classe com a maior saída é associada à amostra. Para verificar o efeito dessa forma de classificar, a classificação realizada pelo modelo M7 foi alterada para ser realizada através da soma ponderada da saída dos neurônios, e assim foi obtido M8. Para 12 bases de dados, não houve diferenças nos desempenhos entre os modelos M7 e M8. No entanto, para 10 dessas bases de dados, cada classe foi representada por somente um neurônio, não havendo, nesses casos, diferenças entre as formas de classificação.

Com relação às outras oito bases de dados, em cinco delas a classificação da saída mais ativa foi superior, sendo o desempenho médio dessas bases aproximadamente 2% superior ao do modelo M8. A classificação pela média ponderada foi superior em três casos, cujo maior aumento de desempenho chegou a 1,54% (base *Spambase*) em relação à M7. Além disso, M8 apresentou quantidade de parâmetros ligeiramente maior que M7. Dessa forma, o procedimento de classificação pelo neurônio mais ativo mostrou ser mais satisfatório em relação a desempenho e complexidade do modelo.

O procedimento de remoção de neurônios foi adicionado ao modelo M7, e assim foi obtido M9. Com esse procedimento, houve pequena redução no tamanho da arquitetura para sete bases de dados, principalmente para a base *Texture*, como visto na Tabela 6.2. Para outras 12 bases de dados, não houve alteração no tamanho da estrutura do modelo, mas, para 10 desses casos, M7 e M9 estão no tamanho mínimo necessário para representar tais bases de dados.

A base *Car* foi a única base de dados na qual o modelo aumentou o tamanho, e foi observado isto acontecer devido a uma configuração dos parâmetros diferente da do modelo M7. Com relação ao desempenho, houve redução média de 0,08% da acurácia nas bases com redução da estrutura e, para essas mesmas bases, houve redução média na estrutura de 5 neurônios (aproximadamente 10 mil *bits*). Pode-se concluir que o procedimento de redução de neurônios conseguiu ligeira redução no tamanho do modelo, embora redução maior não tenha sido possível, principalmente pelo fato do modelo já se encontrar em seu tamanho mínimo para várias bases de dados.

O último modelo avaliado foi o M10. Nele, primeiro são calculados os valores das médias e variâncias de cada classe para depois começar o aprendizado incremental. Esse modelo obteve os maiores valores de acurácia para oito bases de dados. A média da acurácia sobre as bases de dados foi a maior entre os modelos avaliados. O tamanho da sua estrutura foi, na média, aproximadamente duas vezes maior que o do modelo M3, que foi o mais reduzido.

Com o cálculo dos valores das médias e variâncias antes (comparação entre M10 e M7), foram obtidos ganhos de desempenhos maiores de 3% para as bases *Car*, *Heart*, *Segmentation* e *Spambase*, e redução maior para *Sonar* e *Texture*. Nestas, foi observado que, na maioria dos modelos, não foi obtido tamanho mínimo de estrutura para representá-las, podendo evidenciar que a distribuição das classes dessas bases não se aproxima da distribuição Gaussiana. Com isso, o procedimento do cálculo da média e da variância falhou em obter estimativa aproximada dos valores reais desses parâmetros. Embora o procedimento da estimativa inicial dos valores da média e da variância seja simples e restrita com relação à suposição da distribuição das classes ser aproximadamente uma Gaussiana, obteve algum ganho de desempenho sobre algumas bases de dados.

A última linha das Tabelas 6.1 e 6.2 mostra a média dos desvios padrões obtida por cada algoritmo para cada métrica (acurácia e tamanho). Quanto menor esse valor, mais concentrado em torno da média variam os resultados. Para a maioria dos algoritmos, a média dos desvios padrões para acurácia foi próximo de 4%, sendo o maior valor obtido para o modelo M4, que foi o modelo que apresentou a maior instabilidade para a métrica acurácia. A média dos desvios padrões da Tabela 6.2 foi, para a maioria dos modelos, cerca de 60 vezes menor do que a média dos tamanhos das estruturas. O menor desvio padrão foi obtido para o modelo M3, que obteve um tamanho de estrutura mais constante ao longo dos experimentos. O modelo M1 obteve um pequeno valor de desvio padrão quando comparado ao tamanho médio de sua estrutura (média de 10 mil vezes menor), mas como a sua estrutura foi muito superior ao do modelo M3, mesmo uma pequena variação foi o suficiente para ser superior ao de M3.

O procedimento de atualizar os valores das variâncias foi usado pelos modelos para a maior parte das bases de dados, sendo as poucas exceções as bases de dados com grande número de atributos, como a *CNAE-9*, *WebKB-4* e *Reuters-8*. Devido ao grande número de atributos dessas bases, elas devem ser mais sensíveis a estimativas imprecisas de valores e, com isso, resultados melhores são obtidos quando as variâncias não são atualizadas pelo procedimento de atualização.

Os testes estatísticos de Friedman e de Nemenyi foram aplicados sobre os valores das acurácias e quantidade de parâmetros dos modelos. A hipótese nula foi rejeitada no primeiro teste, e o resultado do segundo é indicado na Figura 6.1, para a acurácia, e na Figura 6.2, para o número de parâmetros. Nos testes não são incluídos os valores médios de acurácia e de número de parâmetros obtidos por cada modelo sobre as bases de dados, conforme recomendação em (Demšar, 2006).

Como pode ser observado na Figura 6.1, em relação ao desempenho, M10 foi superior aos modelos M2, M3 e M4; M7, superior aos M3 e M4; e M4 mostrou ser inferior a todos

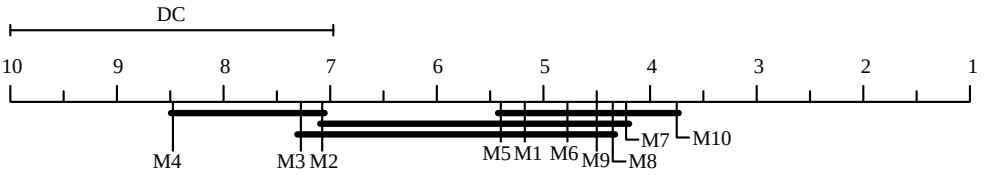


Figura 6.1: Comparação entre os modelos em relação à acurácia usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Modelos não significativamente diferentes entre si são conectados por uma linha espessa. DC é a diferença crítica entre os *ranks* de dois modelos para serem considerados significativamente diferentes.

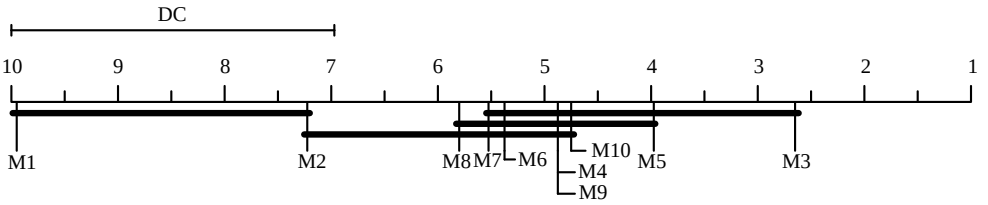


Figura 6.2: Comparação entre os modelos em relação ao número de parâmetros usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Modelos não significativamente diferentes entre si são conectados por uma linha espessa. DC é a diferença crítica entre os *ranks* de dois modelos para serem considerados significativamente diferentes.

os outros, exceto M3 e M2. Por sua vez, a Figura 6.2 mostra que, em termos de número de parâmetros, M3 foi estatisticamente menor que M1, M2 e M8; M5 foi menor que M2 e M1; e, M1, estatisticamente maior que os demais, exceto M2. Não foi possível inferir outras informações do teste de Nemenyi.

Então, como observado, cinco modelos não apresentaram diferenças estatísticas em relação ao de menor tamanho (M3) e de maior desempenho (M10): M5, M6, M7, M9 e M10. Para escolher o modelo a ser usado, o desempate veio usando-se o *rank* feito pelo teste de Friedman para a medida de desempenho. Com isso, o conjunto de procedimentos do modelo M7 foi selecionado para a rede proposta, pois apresenta melhor *rank* entre os que necessitam amostras de treinamento usadas somente uma vez. M10 não foi selecionado por precisar usar os dados de treinamento mais de uma vez.

O modelo M7 também obteve a segunda maior acurácia média sobre as bases, sendo inferior somente a M10, e, para 11 das 20 bases de dados, a menor estrutura possível que podia alcançar. A comparação dos resultados obtidos pelo modelo M7 com os resultados encontrados na literatura é realizada na Seção 6.3 (Tabela 6.13, página 163).

6.1.2 Conclusões

Nos experimentos desta seção foi mostrado que o desempenho do modelo adotado, M7, não possui diferenças estatísticas em relação ao M1, que consiste na técnica de vizinho mais próximo. Em outras palavras, a regularidade de desempenhos de M7 foi semelhante a de M1. Além disso, a estrutura de M7 foi, na média, cerca de 200 vezes menor do que a de M1, havendo diferença estatística entre o tamanho de suas estruturas. Isto indica que o modelo adotado consegue compromisso razoável entre desempenho e complexidade.

Por outro lado, M7 atende a todos os critérios definidos por Langley (1995), por usar uma amostra de treinamento por vez, não reprocessar nenhuma amostra e por estocar somente uma estrutura de conhecimento na memória, tal que não é possível retornar a um estado anterior de conhecimento. Além disso, tal modelo atende a outros requisitos desejados para uma técnica incremental, como ser capaz de acomodar novas classes introduzidas por novos dados, não necessitar de conhecimento prévio sobre a sua estrutura, não necessitar de inicializar sua estrutura para a tarefa de aprendizado, e ser capaz de realizar separação não linear entre classes (a exemplo das bases *Gaussian* e *Concentric*, cuja superfície de separação das classes é não linear).

Para 11 bases de dados, o modelo M7 obteve a menor estrutura possível que podia ser alcançada, o que equivale a representar a distribuição de cada classe por uma função semelhante a uma Gaussiana. E em 10 dessas bases (*Blood*, *Iris*, *Haberman*, *CNAE-9*, *WebKB-4*, *Reuters-8*, *Heart*, *Gaussian*, *Concentric*, *Diabetes*) os valores de desempenho foram tão bons ou melhores que o do modelo M1, que consiste na técnica do vizinho mais próximo. Este resultado comprovou ser viável a hipótese de poder representar as distribuições por uma Gaussiana para realizar a separação entre classes, tendo pouca perda de informação.

6.2 Avaliação da estabilidade e plasticidade

Uma das características desejadas de uma técnica de aprendizado incremental é conseguir manter um compromisso entre as propriedades de estabilidade (não esquecer drasticamente informações passadas) e plasticidade (aprender novas informações com novos dados).

Essa seção de experimentos tem, como meta, avaliar a estabilidade e plasticidade do algoritmo proposto, assim como comparar os resultados obtidos com outras redes neurais incrementais. Para isso, duas metodologias são empregadas.

Na Subseção 6.2.1 são apresentados os resultados do método proposto na Seção 5.3.2 para avaliar esses dois conceitos. Também é realizada, nessa subseção, uma análise sobre as

características das bases de dados que interferem nos graus de estabilidade e plasticidade de cada algoritmo avaliado.

Outra metodologia utilizada foi a apresentada por Polikar et al. (2001) para inferir os níveis de plasticidade e estabilidade dos algoritmos. A metodologia e os resultados obtidos são mostrados na Subseção 6.2.2. Além disso, também é verificado nessa seção o efeito do aumento na quantidade de amostras de treinamento sobre os algoritmos incrementais avaliados.

Para finalizar essa seção, uma comparação entre as duas metodologias e as conclusões gerais obtidas para a RNPe são apresentadas na Subseção 6.2.3.

6.2.1 Avaliação da estabilidade e plasticidade: método proposto

Nessa subseção, o método proposto é aplicado para medir os níveis de estabilidade e plasticidade de algumas técnicas incrementais, e também são verificadas as características em uma base de dados que facilitam ou dificultam a tarefa de aprender novas informações sem esquecer informações antigas.

As técnicas incrementais avaliadas nos experimentos dessa subseção são a RNPe, a RNPI, a RNPI-EM, o eMLP e o EFuNN, e todas as bases de dados da Tabela 5.1 foram utilizadas nos experimentos, com exceção da *WebKB-4* e da *Reuters-8* devido ao tamanho das mesmas. Os níveis de plasticidade e estabilidade de cada método são avaliados pelas métricas Retenção, Inovação e a Média Harmônica deles, todas descritas na Seção 5.3.2. A Retenção mede a quantidade de informação antiga mantida na técnica quando novas informações são inseridas (estabilidade), enquanto Inovação mede a capacidade do algoritmo de aprender novas informações (plasticidade). Quanto maiores os valores de Retenção e Inovação, maior a capacidade de um algoritmo para manter e aprender informação, respectivamente. A Média Harmônica avalia o compromisso entre essas duas medidas.

Para realizar os experimentos, os parâmetros de cada técnica foram ajustados para cada base de dados. Os parâmetros otimizados, assim como o procedimento para ajustá-los, são os descritos na Seção 5.3.4. Após o ajuste dos parâmetros das técnicas, cada base de dados foi dividida em $|C|$ subconjuntos, onde $|C|$ é o número de classes de cada base, e foi empregado o procedimento descrito na Seção 5.3.2 para calcular as medidas de Retenção, Inovação e Média Harmônica. De forma sucinta, nesse procedimento, o classificador é treinado com as amostras de uma classe por vez. Após as amostras de todas as classes terem sido usadas no treinamento, a métrica Retenção estima o quanto de informação o classificador manteve da primeira classe utilizada no treino, enquanto a métrica Inovação verifica a quantidade

Tabela 6.3: Resultados obtidos para a métrica Retenção. Valores em porcentagem. O maior valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Retenção (%)					
Bases de dados	RNPI	RNPI-EM	EFuNN	eMLP	RNPe
Balance	78,94	63,68	**93,02**	34,29	31,43
Blood	**59,23**	49,82	1,45	0,93	21,86
Iris	**97,33**	96,03	42,50	83,60	97,10
Haberman	**63,01**	50,21	5,01	3,82	29,96
Car	**64,24**	37,69	13,69	7,91	5,08
CNAE-9	**100,00**	48,10	13,37	99,99	73,28
Wine	**100,00**	97,53	7,35	31,39	93,93
Yeast	**56,32**	13,64	6,64	15,77	43,25
Abalone	**24,86**	7,39	3,58	19,44	19,36
Heart	**67,00**	50,00	7,64	8,87	48,49
Sonar	79,64	50,00	38,58	**83,52**	82,13
Segmentation	60,91	**63,93**	5,88	57,99	58,00
Texture	82,95	**88,33**	9,27	70,61	81,90
Gaussian	**91,76**	81,83	1,65	3,02	0,09
Concentric	**74,23**	68,25	18,49	3,21	10,66
Spambase	56,27	**74,48**	4,92	25,03	48,91
Diabetes	**100,00**	59,56	0,26	2,84	6,98
Zoo	**100,00**	73,53	96,61	88,27	99,82
Média	**75,49**	59,67	20,55	35,58	47,35
Média dos DPs	**0,00**	2,06	3,60	2,27	3,56

de informação aprendida pelo classificador da última classe a ser usada no treinamento. A média harmônica entre essas duas medidas é a métrica Média Harmônica. Esse procedimento foi realizado 20 vezes com partições aleatórias das bases de dados. Ao final, a média foi calculada para cada uma das métricas.

As Tabelas 6.3, 6.4 e 6.5 mostram os resultados obtidos para as métricas Retenção, Inovação e Média Harmônica, respectivamente. Cada linha corresponde a uma base de dados e cada coluna a um classificador. Todos os valores estão em porcentagem. Os maiores valores para cada base de dados estão realçados em negrito. O valor médio que cada classificador obteve para cada métrica é indicado na penúltima linha de cada tabela. Por sua vez, a última linha de cada tabela indica a média dos desvios padrões (DPs) que cada técnica obteve para as bases de dados. A maior média e a menor média dos desvios padrões de cada tabela estão realçados em negrito.

Como pode ser visto na Tabela 6.5, para algumas bases de dados, tais como *Iris*, *Wine*, *Texture* e *Zoo*, foram obtidos valores altos de Média Harmônica por pelo menos três classificadores. Altos valores da Média Harmônica indicam classificadores obtendo bom compromisso entre estabilidade e plasticidade, e sugere as correspondentes bases de dados apresentando características que as tornam tratáveis para uma tarefa de aprendizado incremental. De fato, essas bases de dados possuem (como visto na Tabela 5.1), de maneira geral, pouca não-

Tabela 6.4: Resultados obtidos para a métrica Inovação. Valores em porcentagem. O maior valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Inovação (%)					
Bases de dados	RNPI	RNPI-EM	EFuNN	eMLP	RNPe
Balance	78,94	63,68	97,95	99,54	**99,68**
Blood	59,23	49,82	**99,96**	99,95	94,89
Iris	97,33	96,03	99,57	**99,73**	97,47
Haberman	63,01	50,21	99,01	**99,45**	88,29
Car	64,24	37,69	86,61	**99,14**	94,24
CNAE-9	**100,00**	48,10	96,65	**100,00**	99,15
Wine	**100,00**	97,53	94,08	99,61	99,43
Yeast	56,32	13,64	79,68	**99,90**	96,33
Abalone	24,86	7,39	80,93	**97,50**	42,93
Heart	67,00	50,00	**99,66**	99,61	81,30
Sonar	79,64	50,00	99,51	**99,90**	99,63
Segmentation	60,91	63,93	82,03	98,36	**99,85**
Texture	82,95	88,33	23,16	99,56	**99,64**
Gaussian	91,76	81,83	99,54	99,31	**99,91**
Concentric	74,23	68,25	99,15	**99,89**	96,30
Spambase	56,27	74,48	**99,90**	99,40	96,51
Diabetes	**100,00**	59,56	99,83	99,82	99,38
Zoo	**100,00**	73,53	59,51	**100,00**	99,77
Média	75,49	59,67	88,71	**99,48**	93,59
Média dos DPs	**0,00**	2,06	2,16	0,29	1,12

linearidade, reduzida sobreposição e atributos com poder discriminativo alto, características que auxiliam na tarefa de classificação.

Por outro lado, algumas bases de dados são mais complexas e apresentaram maior dificuldade para ser mantido o compromisso entre reter e aprender informação, como pode ser visto na Tabela 6.5. Algumas delas são a *Gaussian* e a *Abalone*. Como informada na descrição das bases (Seção 5.2), as classes da base *Gaussian* possuem grande sobreposição, e as classes de *Abalone* possuem contornos de classe não bem definidos e poucas amostras por classe. Nessas bases, as informações sobre uma classe conflitam com as informações de outras classes, o que torna a tarefa de estabilidade e plasticidade mais difícil.

Quando analisando as técnicas, os maiores valores para a métrica Retenção (Tabela 6.3) foram obtidos pelos classificadores RNPI e RNPI-EM. Por outro lado, eles obtiveram os valores mais baixos para a métrica Inovação (Tabela 6.4), salvo alguns resultados obtidos pela RNPI. Assim, a RNPI e a RNPI-EM mostraram ter maior capacidade para estocar informações antigas e habilidade menor para aprender novas informações.

Já EFuNN, eMLP e RNPe obtiveram valores de Inovação acima de 95% para a maioria das bases de dados, mostrando possuírem grande capacidade para aprenderem novas informações. Como consequência, essas técnicas exibiram menor capacidade para reter informação, como visto na Tabela 6.3.

Tabela 6.5: Resultados da Média Harmônica entre Retenção e Inovação. Valores em porcentagem. O maior valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Média Harmônica (%)					
Bases de dados	RNPI	RNPI-EM	EFuNN	eMLP	RNPe
Balance	78,94	63,68	**95,41**	50,89	47,65
Blood	**59,23**	49,82	2,86	1,83	34,45
Iris	**97,33**	96,03	57,77	90,95	97,28
Haberman	**63,01**	50,21	9,38	7,25	43,75
Car	**64,24**	37,69	23,45	14,56	9,36
CNAE-9	**100,00**	48,10	23,30	99,99	84,27
Wine	**100,00**	97,53	11,80	47,60	96,58
Yeast	56,32	13,64	12,17	27,23	**59,66**
Abalone	24,86	7,39	6,85	**32,42**	26,67
Heart	**67,00**	50,00	14,15	16,22	60,64
Sonar	79,64	50,00	55,32	**90,96**	89,94
Segmentation	60,91	63,93	10,39	72,92	**73,30**
Texture	82,95	88,33	13,21	82,61	**89,88**
Gaussian	**91,76**	81,83	3,24	5,86	0,18
Concentric	**74,23**	68,25	31,00	5,42	19,19
Spambase	56,27	**74,48**	9,35	39,96	64,57
Diabetes	**100,00**	59,56	0,50	5,49	12,70
Zoo	**100,00**	73,53	73,39	93,74	99,79
Media	**75,49**	59,67	25,20	43,66	56,10
Média dos DPs	**0,00**	2,06	4,81	2,82	3,77

Para tarefas onde há mudança contínua nas características, os classificadores EFuNN, eMLP e RNPe têm maior capacidade que a RNPI e a RNPI-EM para se adaptarem a essas mudanças. Em compensação, RNPI e RNPI-EM têm compromisso mais balanceado entre reter e aprender informação. Esses resultados sugerem que, para manter mais informações antigas na estrutura do classificador, é necessário abrir mão de obter conhecimento novo. No outro sentido, para adquirir novos conhecimentos, é necessário não se empenhar tanto em manter as informações antigas. Isto se deve muito a um possível conflito entre a informação atual contida no algoritmo e a nova a ser inserida. A decisão entre privilegiar o conhecimento antigo ou beneficiar o novo altera o comportamento do algoritmo.

Os resultados obtidos pelos algoritmos RNPI e RNPI-EM para as três métricas são iguais entre si. Ou seja, para uma determinada base de dados, o valor de Retenção foi igual ao de Inovação, e, assim, iguais ao da Média Harmônica. Os valores de Retenção e Inovação são iguais porque, tanto a RNPI quanto a RNPI-EM, não são influenciadas pela ordem em que as classes do problema são apresentadas durante o treinamento. Isto acontece porque o aprendizado de cada classe nessas redes é independente do das demais classes. Além de não ser influenciada pela ordem de aprendizado das classes, a RNPI também não é afetada pela ordem de apresentação das amostras.

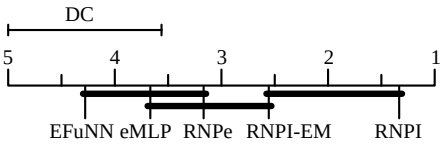


Figura 6.3: Comparação entre os classificadores em relação à métrica Retenção usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

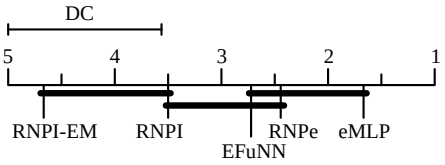


Figura 6.4: Comparação entre classificadores em relação à métrica Inovação usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

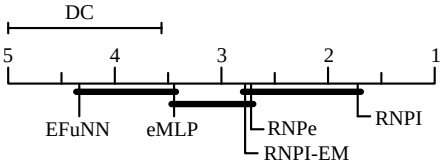


Figura 6.5: Comparação entre os classificadores em relação à métrica Média Harmônica usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

As médias dos desvios padrões apresentadas nas três tabelas apresentam algumas características interessantes. A RNPI obteve desvios padrões iguais a zero nas três tabelas. Isto aconteceu porque o treinamento da RNPI é independente da ordem das amostras, então, como foram sempre usadas todas as amostras para treinamento, sempre foram obtidos os mesmos resultados. Os desvios padrões da RNPI-EM também foram iguais nas três tabelas. Nesse caso, eles foram iguais porque a ordem em que são apresentadas as classes do problema não altera o resultado do treinamento. Por outro lado, elas são diferentes de zero porque a ordem das amostras (dentro de cada classe) influencia o aprendizado. As demais médias dos desvios padrões oscilaram, em sua maioria, em torno de 3% a 4%.

Os testes de Friedman e de Nemenyi foram aplicados sobre as medidas de Retenção, Inovação e Média Harmônica. A hipótese nula do primeiro teste foi rejeitada nas três métricas, e os resultados do teste para as métricas Retenção, Inovação e Média Harmônica estão indicados nas Figuras 6.3, 6.4 e 6.5, respectivamente.

Na Figura 6.3, o algoritmo RNPI é indicado ser estatisticamente superior a RNPe, eMLP e EFuNN em relação à métrica Retenção, e RNPI-EM é superior a EFuNN. A Figura 6.4 indica que, para a métrica Inovação, o algoritmo eMLP foi superior a RNPI e RNPI-EM, e EFuNN e RNPe foram superiores a RNPI-EM. Por fim, a Figura 6.5 mostra que, na Média Harmônica entre Retenção e Inovação, a RNPI foi superior a eMLP e EFuNN; e, RNPe e RNPI-EM foram superiores a EFuNN.

De forma geral, conclui-se ser a técnica RNPI capaz de reter mais informações que as outras. No entanto, a RNPI é uma rede neural baseada em memória, armazenando todas as amostras de treinamento em sua arquitetura sem realizar compressão de dados. Assim, apresenta maior facilidade em recuperar informação do que as demais redes, uma vez que as amostras inseridas como neurônios não sofrem qualquer alteração com o aumento na quantidade de dados de treinamento.

A técnica RNPe obteve um índice elevado para a métrica Inovação e, no *rank* da Média Harmônica, ficou atrás só da RNPI. De acordo com o *rank* obtido pela RNPe nos testes, nota-se tendência a se adaptar a novos dados de forma melhor que as técnicas RNPI e RNPI-EM, que são mais conservadoras, e de reter mais informação do que o EFuNN e eMLP, que mostraram ser mais radicais em relação a descartar mais rápido as informações antigas.

Uma importante tarefa é identificar quais características das bases de dados podem afetar o compromisso entre estabilidade e plasticidade. Reconhecer tais características pode auxiliar na tarefa de determinar o comportamento de uma técnica para certa base de dados. Com esse objetivo, o teste de Kendall foi usado para tentar identificar tais características. Esse teste foi realizado entre cada medida de complexidade da Tabela 5.1 e o resultado da Média Harmônica de cada técnica avaliada. O teste de hipótese de correlação foi realizado nesse experimento, com nível de significância de 5%.

Para cada algoritmo foi observado existir uma ou mais características impactando mais sobre os resultados obtidos. O teste indicou os resultados da RNPI estarem significativamente correlacionados com as medidas F2 e N4, enquanto os resultados da RNPI-EM estão mais correlacionados com as medidas N1, N2, N3 e N4. Quanto maior os valores dessas medidas, menores tendem a ser os resultados dessas técnicas. Valores elevados de N1, N2, N3 e N4 sugerem amostras de diferentes classes muito próximas entre si, contorno das classes não muito bem definido e separação entre classes com muita não linearidade. Valores elevados de F2 sugerem existir grande sobreposição dos valores dos atributos de diferentes classes. De forma rudimentar, pode-se dizer que os resultados dessas técnicas estão mais relacionados à forma do contorno das classes e à sua distribuição espacial.

Além disso, os resultados de RNPI-EM mostraram ser correlacionados com a medida EN, que mede a entropia das bases. Em outras palavras, para bases de dados com maior

desequilíbrio na quantidade de amostras por classe (valor menor de EN), a Média Harmônica de RNPI-EM tende a ser menor. De certa forma, isso é esperado para a RNPI-EM, pois utiliza a probabilidade *a priori* das classes, fazendo com que fique polarizada para classes com número maior de amostras.

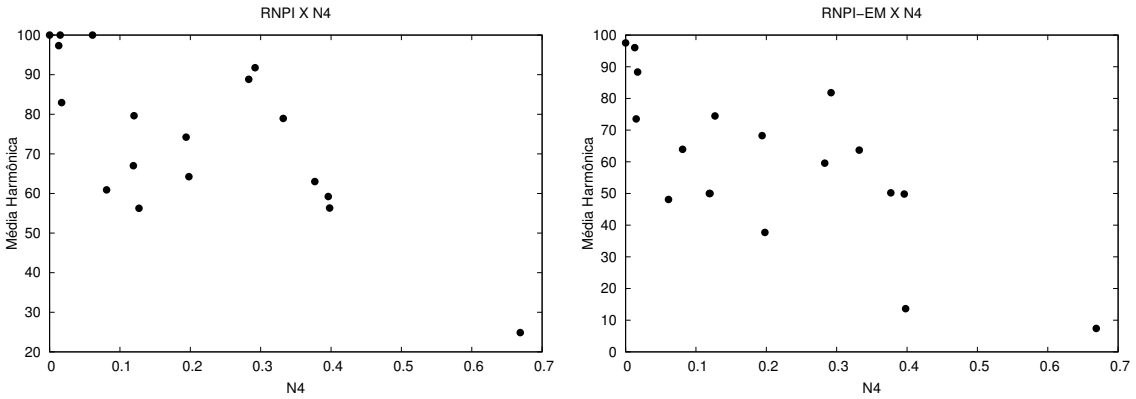
As medidas F1, F2 e N4 mostraram afetar mais os resultados das técnicas eMLP e RNPe. As Médias Harmônicas de eMLP e RNPe tenderam a serem mais elevadas para valores maiores de F1 e menores de F2 e N4. Dessa forma, os resultados do eMLP e da RNPe são mais influenciados pelo poder discriminativo dos atributos e pela complexidade do contorno de separação entre classes.

Com relação ao EFuNN, o teste estatístico sugeriu serem seus resultados influenciados pela medida N3, sendo os melhores resultados do EFuNN obtidos para baixos valores de N3. Altos valores de N3 indicam concentração maior de amostras próximas ao contorno das classes.

Os resultados dessa análise ajudam a explicar os altos valores (em relação a outros obtidos pela própria técnica) para Média Harmônica obtidos por eMLP e RNPe para as bases de dados *Iris*, *CNAE-9*, *Segmentation*, *Texture* e *Zoo*. Essas bases apresentaram em comum altos valores de F1 e baixos valores de F2 e N4 (ver Tabela 5.1, página 103). No entanto, para bases de dados com baixos valores de F1 ou altos valores de F2 e N4, como *Car*, *Gaussian* e *Concentric*, foram alcançados baixos valores de Média Harmônica. Para bases com valores pequenos de N4, como *Iris*, *Wine*, *Texture* e *Zoo*, as técnicas RNPI e RNPI-EM tiveram melhor desempenho, porém obtiveram resultados menores para *Blood*, *Abalone* e *Yeast*, possuidoras de valores elevados de N4. Por fim, EFuNN alcançou resultados de Média Harmônica mais reduzidos para bases com elevado valor de N3, como *Blood*, *Haberman*, *Abalone* e *Diabetes*.

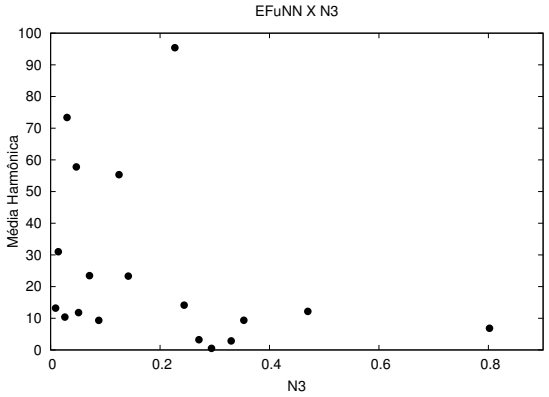
As Figuras 6.6 a 6.9 ilustram o relacionamento entre a Média Harmônica de cada técnica com algumas das medidas de complexidade. A Figura 6.6(a) mostra o gráfico da Média Harmônica da RNPI em função da medida N4. Como pode ser observado, os valores mais elevados da Média Harmônica foram obtidos para baixos valores de N4 (N4 < 0,1) e para valores mais elevados de N4 os resultados tenderam a serem menores. Também notou-se que, das oito bases de dados em que a RNPI obteve Média Harmônica acima de 80%, cinco possuem valor de N4 menor do que 0,07. E, das quatro bases de dados em que foram obtidos resultados menores que 60%, três possuem N4 maior do que 0,39.

A Figura 6.6(b) ilustra o gráfico da relação entre a Média Harmônica da RNPI-EM e a medida N4. Nela, é possível observar resultados mais elevados obtidos para valores reduzidos de N4 (N4 < 0,05), com exceção de um abaixo de 80%. Esse resultado foi da base *Zoo*, e foi menor porque as amostras são mais concentradas em uma classe (valor baixo de



(a) (b)

Figura 6.6: Média Harmônica da RNPI em função da medida N4 (a) e Média Harmônica da RNPI-EM em função da medida N4 (b).



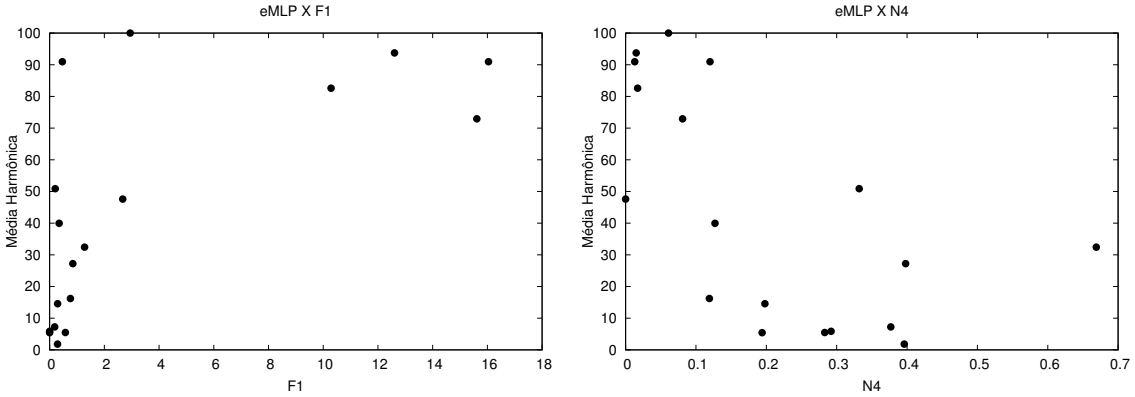
(a)

Figura 6.7: Média Harmônica do EFuNN em função da medida N3.

EN), afetando negativamente a RNPI-EM. Para valores mais elevados de N4 (N4 > 0,2), não foram obtidos resultados maiores que 70%, com exceção de uma base. Para valores de N4 maiores de 0,35 não foram alcançadas Médias Harmônicas acima de 51%.

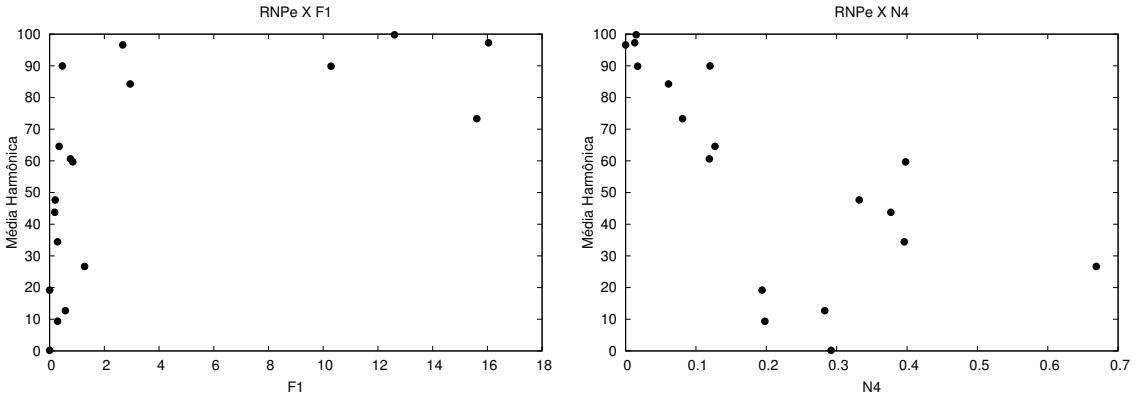
A medida mais correlacionada com os resultados da técnica EFuNN foi N3, e a Figura 6.7 ilustra a Média Harmônica obtida por EFuNN em função de N3. Os valores de Média Harmônica mais elevados de EFuNN foram obtidos para valores de N3 na faixa entre 0 e aproximadamente 0,2. Para valores de N3 mais elevados, não foram obtidos resultados acima de 20%, com exceção da base *Balance*, na qual foi obtido resultado acima de 90%.

Os resultados de ambas as técnicas eMLP e RNPe estão mais correlacionados com as medidas F1 e N4. Como observado nas Figuras 6.8(a) e 6.9(a), valores maiores de Média Harmônica foram obtidos para valores mais elevados de F1. Para F1 maior que 10, não foram



(a) (b)

Figura 6.8: Média Harmônica do eMLP em função das medidas F1 (a) e N4 (b).



(a) (b)

Figura 6.9: Média Harmônica da RNPe em função das medidas F1 (a) e N4 (b).

obtidos valores de Média Harmônica abaixo de 70% em ambas as técnicas. Para F1 menor do que 2, houve maior variância dos resultados, sugerindo que a relevância de outros fatores aumenta para pequenos valores de F1. A maior parte das Médias Harmônicas inferiores a 30% nos dois algoritmos foi obtida para bases possuidoras de baixo valor de F1 e elevado valor de N4.

As Figuras 6.8(b) e 6.9(b) mostram que, para valores baixos de N4 (N4 < 0,1), as Médias Harmônicas dos dois algoritmos foram elevadas, não sendo inferior a 70%, com exceção de um caso para o eMLP. Com valores mais elevados de N4, a tendência é os resultados ficarem menores. Todos os resultados para a RNPe menores do que 50% foram para bases com N4 maiores do que 0,18, enquanto os resultados para o eMLP abaixo de 40% foram para bases de dados com N4 maiores do que 0,1. Isto sugere a RNPe ser menos influenciada pela medida N4 do que o eMLP.

Evidentemente, estas características não atuam isoladas no comportamento da técnica (a base *Balance*, por exemplo, possui um valor um pouco elevado para N3, mas ainda assim EFuNN obteve a sua maior Média Harmônica em cima dessa base), e nem que essas medidas de complexidade sejam as únicas a interferir no comportamento da técnica (segundo Watts (2009), bases de dados com valores binários interferem no desempenho da EFuNN, no entanto, nenhuma medida de complexidade considera tal informação). Também foi observado as medidas de complexidade não serem completamente independentes. Por exemplo, foi encontrada uma correlação entre as medidas F1 e N4. Ainda assim, essas medidas podem auxiliar no entendimento do comportamento de um determinado classificador para certa base de dados.

De forma geral, como pôde ser observado, os algoritmos tendem a obter melhor compromisso entre estabilidade e plasticidade para bases de dados cujos atributos possuem alto poder discriminativo e contornos de classe bem definidos, de preferência linearmente separáveis. A influência de cada uma dessas características varia de algoritmo para algoritmo, embora, de certo modo, todos os algoritmos avaliados foram afetados pelo nível de complexidade da separação entre classes.

6.2.2 Avaliação da estabilidade e plasticidade: método de Polikar et al. (2001)

Em (Polikar et al., 2001) é apresentada uma metodologia para avaliar a estabilidade e plasticidade de algoritmos com aprendizado incremental. Basicamente, o método consiste em dividir cada base de dados em N subconjuntos mutuamente exclusivos, com tamanho aproximadamente igual, e contendo todas as classes da base de dados. Assim, cada algoritmo é treinado com o primeiro subconjunto e seu desempenho é avaliado sobre o primeiro e o último subconjunto. Em seguida, um segundo subconjunto também é usado para treinamento, e é calculado o desempenho sobre os dois primeiros subconjuntos e o último subconjunto. Esse procedimento continua até que $N-1$ subconjuntos sejam usados para treinamento e o desempenho seja calculado sobre os $N-1$ subconjuntos e o último subconjunto.

O último subconjunto não é utilizado para treinamento, e serve para verificar o quanto o desempenho do algoritmo melhorou com o aumento da quantidade de amostras de treinamento, sendo esta uma forma de avaliar a plasticidade do algoritmo. O cálculo do desempenho sobre os subconjuntos usados para treinamento serve para medir o quanto de informação antiga o algoritmo consegue reter quando mais dados são usados para treinamento, sendo esta uma forma de avaliar a estabilidade do algoritmo.

Os experimentos dessa seção foram realizados baseados no procedimento de Polikar et

al.(2001), com a diferença de que os desempenhos são calculados somente no primeiro subconjunto usado para treino, chamado, daqui por diante, de dados antigos, e no último subconjunto, chamado de dados desconhecidos. O desempenho sobre os dados antigos é uma forma de mensurar o nível de estabilidade do algoritmo, enquanto sobre os dados desconhecidos avalia a plasticidade do algoritmo. Os testes sobre as outras partes foram removidos para obter melhor visualização dos dados e por supor que os efeitos são mais realçados no primeiro subconjunto de treino que nos demais. O valor de N usado nos experimentos foi 10.

Os algoritmos avaliados foram RNPI, RNPI-EM, EFuNN, eMLP e RNPe, sendo utilizadas as bases de dados *Haberman*, *Car*, *CNAE-9*, *Yeast*, *Abalone*, *Heart*, *Concentric*, *Spambase* e *Zoo*. O número de bases de dados foi limitado para reduzir o tamanho e quantidade de tabelas nessa seção de experimentos. Antes dos experimentos, os parâmetros de cada algoritmo foram ajustados para cada base de dados através dos procedimentos descritos na Seção 5.3.4. A medida de desempenho usada nos experimentos foi a acurácia, a mesma utilizada em (Polikar et al., 2001).

Nos experimentos, também foi estimado o aumento da complexidade de cada algoritmo quando a quantidade de dados de treinamento aumenta. Para avaliar esse aumento de complexidade, são usadas as medidas do número de parâmetros e tempo de classificação de cada algoritmo. Os experimentos foram realizados aplicando dois 10 *fold cross-validation*, totalizando 20 execuções. Para cada medida de avaliação foi calculado o valor médio das execuções.

A Tabela 6.6 mostra a acurácia média (em porcentagem) sobre o primeiro subconjunto usado para treinamento, que são os dados antigos. Cada coluna da tabela representa uma base de dados e cada linha uma técnica. Para cada base de dados na tabela, há três colunas. As primeiras duas são os resultados obtidos quando os classificadores são treinados só com um subconjunto (*sub 1*) (que são os próprios dados antigos), e com nove subconjuntos (*sub 1-9*), respectivamente. Os resultados obtidos com uma quantidade intermediária de subconjuntos de treinamento encontram-se entre os da primeira e segunda coluna, e foram omitidos para melhor visualização dos resultados. O melhor desempenho médio para cada base de dados e subconjunto de treinamento está realçado em negrito.

A terceira coluna de cada base de dados na tabela é a medida da taxa de aumento (TA) do desempenho. Ela é uma porcentagem da diferença entre as acurácias obtidas para *sub 1-9* e *sub 1*, dividida pela acurácia de *sub 1*. Valores positivos elevados significam grande aumento de desempenho quando a técnica é treinada com *sub 1-9* em relação ao treinamento com *sub 1*. Valores negativos indicam redução do desempenho com o aumento da quantidade de amostras de treinamento.

Tabela 6.6: Acurácia média das técnicas sobre os dados antigos em cada base de dados (valores em porcentagem). TA é a taxa de aumento da acurácia quando o número de amostras de treinamento aumenta de um subconjunto (*sub 1*) para nove subconjuntos (*sub 1-9*). Os maiores valores para cada base de dados estão realçados em negrito.

Técnicas	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA
	Haberman			*Car*			*CNAE-9*		
RNPI	71,88	75,12	4,51	84,17	74,74	-11,20	**100,00**	**100,00**	0,00
RNPI-EM	73,55	73,88	0,45	70,02	77,29	10,37	44,31	55,97	26,33
EFuNN	**86,12**	**78,30**	-9,08	88,34	86,55	-2,03	53,15	65,00	22,30
eMLP	82,67	72,58	-12,21	**94,04**	**89,33**	-5,01	**100,00**	**100,00**	0,00
RNPe	70,94	73,67	3,84	81,40	80,93	-0,57	97,64	95,28	-2,42
	Yeast			*Abalone*			*Heart*		
RNPI	55,99	54,58	-2,52	16,45	12,17	-25,98	73,52	67,41	-8,31
RNPI-EM	32,82	38,37	16,93	18,28	23,88	30,65	57,59	55,56	-3,54
EFuNN	57,11	60,44	5,84	21,35	21,70	1,63	**90,19**	**80,56**	-10,68
eMLP	**89,09**	**82,11**	-7,83	**91,45**	**71,27**	-22,07	**90,19**	75,93	-15,81
RNPe	44,72	53,97	20,68	20,47	24,23	18,36	67,41	70,93	5,22
	Concentric			*Spambase*			*Zoo*		
RNPI	94,76	94,58	-0,19	56,45	53,09	-5,97	**100,00**	**100,00**	0,00
RNPI-EM	62,94	82,60	31,24	60,36	72,46	20,06	93,50	93,50	0,00
EFuNN	**97,82**	97,72	-0,10	90,78	86,83	-4,36	90,09	95,09	5,55
eMLP	97,18	**97,86**	0,70	**92,31**	**88,60**	-4,01	98,00	98,05	0,05
RNPe	81,74	95,76	17,15	76,73	82,31	7,27	**100,00**	99,55	-0,45

De acordo com os valores de TA da Tabela 6.6, dois comportamentos são possíveis de identificar para as técnicas: os classificadores que reduziram o seu desempenho para os dados antigos quando a quantidade de treinamento aumentou, e os classificadores que tiveram o desempenho aumentado.

As técnicas RNPI e eMLP tiveram redução na acurácia para a maioria das bases de dados. Uma explicação para isso é que ambas as técnicas tendem a armazenar as amostras de treinamento em suas arquiteturas (a RNPI armazena todas as amostras na arquitetura). Com o aumento da quantidade de amostras de treinamento, mais dados são armazenados e pode acontecer sobreposição de amostras de classes diferentes, o que faz o desempenho sobre os dados antigos diminuir. Além disso, para o caso do eMLP, as amostras adicionadas como neurônios têm os seus pesos ajustados à medida que novos dados são usados para treinamento, desviando-se assim dos pesos originais. Por exemplo: ambas as técnicas obtiveram acurácias acima de 97% para as bases de dados *CNAE-9* e *Zoo*, que possuem contornos de classe menos complexos (valor baixo de N4) e classes com pouca sobreposição (valor baixo de F2), e a acurácia sobre os dados antigos não caiu com o aumento da quantidade de amostras de treinamento. Para a base *Abalone*, que possui uma complexidade maior nos contornos das classes e um nível maior de sobreposição, a queda de desempenho foi acentuada, sendo acima de 20% para os dois algoritmos.

Como observado na Tabela 6.6, o eMLP obteve as maiores acurácias sobre os dados antigos para a maioria das bases de dados. Como essa técnica tende a adicionar algumas amostras como neurônios na arquitetura, e realiza a classificação de acordo com o neurônio mais ativo, obtém resultados bastante elevados quando classifica amostras já usadas para treinamento. Neste sentido, destaca-se a base *Abalone*, em que ele obteve 91,45% para *sub 1*, enquanto a segunda maior acurácia foi de 21,35%.

As técnicas RNPI-EM e RNPe tiveram um comportamento diferente. Para a maioria das bases de dados, apresentaram ganho de desempenho com o aumento da quantidade de amostras de treinamento. Isto acontece porque, diferentemente das outras técnicas, não são baseadas em simplesmente adicionar amostras na arquitetura, mas modelar a distribuição dos dados. Então, com uma quantidade maior de dados é obtido um modelo melhor ajustado, e com isso aumenta o desempenho sobre os dados antigos.

A RNPI-EM alcançou desempenho relativamente menor do que a RNPe para a base *CNAE-9*. Isso deve ter ocorrido porque essa base possui grande quantidade de atributos (mais de 800) e uma quantidade igual de amostras por classe. Ambas as características afetam negativamente o desempenho da RNPI-EM, pois o número maior de atributos amplia o efeito das imprecisões nos valores dos parâmetros estimados e isto, por sua vez, induz o classificador a classificar as amostras mais em algumas classes do que em outras.

O comportamento da EFuNN não apresentou um padrão muito claro, pois, em quatro bases de dados houve ganho de desempenho, e, em cinco, o desempenho diminuiu. Embora a EFuNN também adicione amostras como neurônios na arquitetura da sua rede, é provável que, por não preservar a informação original contida em cada amostra, a lógica *fuzzy* torne a classificação da EFuNN menos pontual do que o eMLP. Logo, o desempenho sobre os dados antigos (usados para treinamento) tende a ser menor em comparação ao eMLP.

O desempenho do EFuNN foi reduzido em relação aos das técnicas eMLP, RNPe e RNPI para a base *CNAE-9*. Uma explicação plausível é que essa base de dados é constituída por muitos elementos iguais a zero (mais de 99%) a e lógica *fuzzy* não teve efeito sobre estes valores, podendo até mesmo ter prejudicado o desempenho.

A Tabela 6.7 exibe o desempenho médio (em porcentagem) sobre o último subconjunto que nunca é usado para treinamento, chamado aqui de dados desconhecidos. De forma similar à Tabela 6.6, cada coluna representa uma base de dados e cada linha um classificador. As colunas das bases de dados estão divididas em três partes. As primeiras duas são os resultados obtidos quando os classificadores são treinados com um subconjunto (*sub 1*) e com nove subconjuntos (*sub 1-9*), respectivamente. A terceira parte é a medida TA, calculada da mesma forma como apresentado anteriormente. O melhor desempenho médio para cada base de dados e subconjunto de treinamento está realçado em negrito.

Tabela 6.7: Acurácia média das técnicas sobre os dados desconhecidos em cada base de dados (valores em porcentagem). TA é a taxa de aumento da acurácia quando o número de amostras de treinamento aumenta de um subconjunto (*sub 1*) para nove subconjuntos (*sub 1-9*). Os maiores valores para cada base de dados estão realçados em negrito.

Técnicas	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA
	Haberman			*Car*			*CNAE-9*		
RNPI	62,42	**75,30**	20,63	67,25	71,50	6,33	78,84	90,69	15,03
RNPI-EM	**73,55**	73,72	0,23	70,02	77,61	10,83	32,41	50,14	54,71
EFuNN	59,46	59,82	0,61	74,62	**84,00**	12,57	41,71	64,26	54,05
eMLP	59,76	61,64	3,15	75,78	83,40	10,04	76,90	89,44	16,32
RNPe	65,87	72,55	10,14	**78,36**	80,79	3,10	**81,06**	**92,45**	14,05
	Yeast			*Abalone*			*Heart*		
RNPI	40,29	52,49	30,26	9,65	10,93	13,27	**63,89**	65,56	2,61
RNPI-EM	31,60	38,27	21,12	16,03	23,76	48,24	53,15	55,56	4,53
EFuNN	38,65	46,42	20,12	**19,25**	21,31	10,70	58,52	60,93	4,11
eMLP	**43,77**	47,80	9,21	19,09	19,80	3,70	60,00	61,67	2,78
RNPe	35,27	**52,66**	49,30	17,08	**24,42**	42,95	62,04	**72,22**	16,42
	Concentric			*Spambase*			*Zoo*		
RNPI	93,48	94,32	0,90	52,22	51,99	-0,44	68,68	**98,00**	42,69
RNPI-EM	62,94	82,5	31,08	60,10	71,82	19,51	66,68	80,64	20,93
EFuNN	**94,72**	**97,7**	3,15	72,60	81,21	11,85	68,73	88,14	28,24
eMLP	93,32	97,68	4,67	**74,51**	82,43	10,63	**69,68**	95,00	36,33
RNPe	80,76	96,24	19,17	74,30	**82,44**	10,96	67,23	94,55	40,64

Verifica-se, na Tabela 6.7, que, exceto em um único caso, o desempenho das técnicas sobre os dados desconhecidos aumentou com o aumento da quantidade de amostras de treinamento. A rede proposta RNPe obteve alguns dos melhores resultados depois de serem usados os nove subconjuntos de treinamento, e obteve valores de TA acima de 40% em três bases de dados, sugerindo sua capacidade de aprendizado ter sido elevada nessas bases de dados quando as amostras de treinamento aumentaram de um para nove subconjuntos.

Se comparados os resultados das Tabelas 6.6 e 6.7, os da RNPI-EM e RNPe para os dados antigos foram semelhantes aos dos resultados sobre os dados desconhecidos, principalmente na condição em que ambas as técnicas são treinadas com os nove subconjuntos de treino. Isso acontece porque ambos os algoritmos são baseados na construção de modelos que buscam representar a distribuição dos dados. Então, se os dados antigos possuem uma distribuição semelhante aos dados desconhecidos, os desempenhos sobre ambos os conjuntos de dados devem ser semelhantes. Como pode ser visto nas Tabelas 6.6 e 6.7, os resultados obtidos pela RNPe foram, na maior parte das vezes, superiores ao da RNPI-EM, sugerindo ter a rede neural proposta melhor capacidade para modelar os dados.

O desempenho do eMLP sobre os dados desconhecidos foi inferior aos obtidos sobre os dados antigos, pois os dados do último subconjunto nunca foram usados para treinar o eMLP, e, assim, não havia padrões muito semelhantes aos dados desconhecidos armazenados em sua

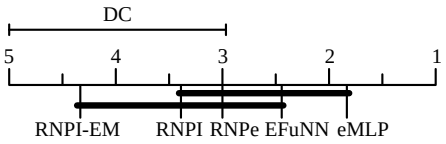


Figura 6.10: Comparação entre as técnicas incrementais em relação à acurácia sobre os dados antigos, usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Técnicas não significativamente diferentes entre si são conectadas por uma linha espessa. DC é a diferença crítica entre os *ranks* de duas técnicas para serem consideradas significativamente diferentes.

arquitetura.

EFuNN apresentou, para as bases *Haberman*, *Yeast* e *Heart*, diferença mais elevada entre os desempenho sobre os dados antigos e os dados desconhecidos. Essas bases possuem contornos de classe complexos e não muito bem definidos, e amostras próximas aos contornos de classe (como sugerido pelos valores elevados de N1, N2, N3 e N4). Então, embora a EFuNN tenha obtido um desempenho alto sobre os dados antigos por ter estocado amostras dessas bases em sua arquitetura, apresentou resultados mais baixos sobre amostras não usadas para treinamento, devido à complexidade da superfície de separação das classes. A base *Abalone* apresentou complexidade ainda mais elevada, pois até a acurácia da EFuNN sobre os dados antigos foi bem reduzida. Os resultados da RNPI foram ligeiramente inferiores ao da tabela anterior, sendo a maior diferença entre os resultados das duas tabelas de aproximadamente 10%.

Outro detalhe da Tabela 6.7 é que as técnicas apresentaram desempenho elevado sobre a base *Zoo* para ambos dados antigos e desconhecidos, exceto para os dados desconhecidos quando os algoritmos foram treinados com somente um subconjunto. Embora essa base de dados seja uma tarefa fácil de classificação (como visto na Tabela 5.1, página 103), a quantidade de amostras em cada subconjunto dessa base de dados é pequena (cerca de 10) e ela é formada por sete classes, resultando em aproximadamente uma amostra por classe em cada subconjunto. Assim, para o primeiro subconjunto de treinamento, a acurácia sobre os dados desconhecidos é baixa, mas, para nove subconjuntos, houve aumento considerável na acurácia obtida por todas as técnicas.

Os testes de Friedman e de Nemenyi foram aplicados sobre os resultados de acurácia para os dados antigos e dados desconhecidos, quando os classificadores foram treinados com os nove subconjuntos de treinamento (*sub 1-9*). A hipótese nula do teste de Friedman foi rejeitada, e as Figuras 6.10 e 6.11 mostram os resultados do teste de Nemenyi para os dados antigos e desconhecidos, respectivamente.

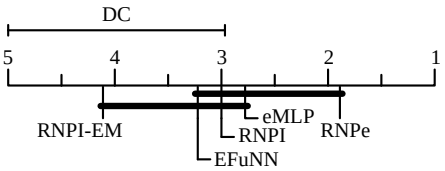


Figura 6.11: Comparação entre as técnicas incrementais em relação à acurácia sobre os dados desconhecidos, usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Técnicas não significativamente diferentes entre si são conectadas por uma linha espessa. DC é a diferença crítica entre os *ranks* de duas técnicas para serem consideradas significativamente diferentes.

O resultado do teste informado pela Figura 6.10 mostra que a RNPI-EM foi estatisticamente inferior ao eMLP, e que as técnicas eMLP e EFuNN tenderam a conseguir uma acurácia mais elevada sobre os dados antigos do que as outras técnicas. A RNPe obteve uma posição no *rank* logo atrás da EFuNN, e na frente da RNPI.

Os resultados desse teste indicam que o eMLP e o EFuNN tendem a reter mais informação (ter maior estabilidade) que as demais técnicas. Como mencionado acima, o procedimento de classificação de ambas as técnicas é baseado no neurônio mais ativo, que, em geral, é representado por uma amostra de treinamento. Isto auxilia na identificação da classe correta para cada amostra já usada para treinamento. A RNPI também armazena amostras em sua arquitetura, mas seu procedimento de classificação considera a contribuição de cada neurônio, sendo uma classificação menos pontual e mais suavizada, o que reduziu a acurácia sobre as amostras que foram usadas para treinamento.

Contudo, a avaliação sobre os dados antigos não dá uma medida exata de quanta informação de fato os algoritmos preservaram do primeiro subconjunto de treinamento, pois padrões semelhantes aos contidos nos dados antigos estavam inseridos nos outros subconjuntos de treinamento. Exemplo desses casos foram a RNPI-EM e a RNPe, em que, para a maioria das bases de dados, a acurácia sobre os dados antigos aumentou quando foram usadas mais amostras de treinamento, sugerindo terem eles inferido informação das outras amostras, úteis para a correta classificação dos dados antigos.

Já na Figura 6.11, a RNPe foi significativamente superior a RNPI-EM, e não houve diferenças estatísticas significantes entre a RNPe e as outras técnicas, embora a RNPe tenha obtida uma posição menor no *rank*. Assim, a RNPe tendeu a aprimorar melhor o seu modelo quando foram disponibilizadas mais amostras de treinamento. Em outras palavras, a RNPe apresentou um grau maior de plasticidade, embora não tenha sido significativamente maior que eMLP, RNPI e EFuNN.

A Tabela 6.8 mostra o número médio de parâmetros e pesos de cada técnica: as colunas

representam as bases de dados e as linhas as técnicas. Assim como nas outras duas tabelas, para cada base de dados existem três colunas, sendo as duas primeiras medidas do número de parâmetros quando as técnicas são treinadas com um (*sub 1-9*) e nove (*sub 1-9*) subconjuntos, respectivamente; a terceira mede a taxa de crescimento (TA) do número de parâmetros. A menor quantidade de parâmetros para cada base de dados está realçada em negrito. A quantidade de parâmetros foi medida em *bits* e, para uma melhor visualização dos dados, são informados os valores de $\log_{10}$ do número de *bits*. Os valores de TA foram calculados sobre o número de *bits*, e não sobre o logaritmo deles.

A Tabela 6.8 informa que a RNPe teve a menor arquitetura para cinco bases e o menor aumento de complexidade (valor de TA) para seis bases de dados. Isto indica que, além de possuir arquitetura bem reduzida, também foi bem estável, pois cresceu mais lentamente que as arquiteturas das outras técnicas e, em cinco bases de dados, sua arquitetura não cresceu com o aumento da quantidade de amostras de treinamento. Somente para a base *Zoo* o eMLP e a RNPI obtiveram arquiteturas ligeiramente menor do que a RNPe. Ainda assim, para um número maior na quantidade de amostras de treinamento, as arquiteturas de ambas as redes tornaram-se maiores.

A RNPI-EM possui também arquitetura bem reduzida para a maioria das bases de dados, com exceção da base *Spambase*, quando, de um conjunto de treinamento para outro, o tamanho de sua arquitetura aumentou em 3700%. Como foi descrito na Seção 3.4, cada neurônio dessa rede pode se dividir em vários outros em somente uma única iteração de treinamento, podendo ocorrer, embora não com frequência, grande aumento no tamanho de sua estrutura. Ainda assim, o tamanho de sua arquitetura foi inferior ao das outras técnicas, com exceção da RNPe.

A RNPI obteve os valores mais elevados de TA para as bases de dados, sendo ligeiramente inferior a 800%, significando que sua arquitetura cresceu quase oito vezes em relação à arquitetura com somente um subconjunto de treinamento. Esse valor elevado se deve ao fato dessa rede sempre adicionar as amostras de treinamento em sua arquitetura. Para as bases *Concentric* e *Spambase*, que possuem número maior de amostras de treinamento, a arquitetura da RNPI foi mais de 100 vezes maior do que da RNPe.

As redes eMLP e EFuNN adicionam neurônios em suas estruturas num ritmo menor do que a RNPI, mas, ainda assim, apresentaram valores de TA acima de 300% para a maioria das bases de dados, e, para algumas delas, suas arquiteturas foram mais de 10 vezes maiores do que da RNPe.

Os tempos necessários para realizar a classificação de cada subconjunto são mostrados na Tabela 6.9: cada coluna representa uma base de dados e cada linha uma técnica, de forma igual às outras tabelas nessa seção. Na coluna *sub 1* estão indicados os tempos obtidos

Tabela 6.8: Número médio de pesos e parâmetros de cada técnica para cada base de dados (valores em $\log_{10}(bits)$). TA é a taxa de aumento do número de parâmetros quando o número de amostras de treinamento aumenta de um subconjunto (*sub 1*) para nove subconjuntos (*sub 1-9*). Os menores números de parâmetros para cada base de dados estão realçados em negrito.

Técnicas	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA
	Haberman			*Car*			*CNAE-9*		
RNPI	3,47	4,42	791,38	4,52	5,48	799,23	6,47	7,43	799,99
RNPI-EM	2,89	2,89	0,00	3,42	3,42	0,00	5,87	5,87	0,00
EFuNN	3,69	4,55	633,88	4,53	5,25	423,83	6,17	6,82	345,89
eMLP	3,34	4,24	702,21	4,23	4,99	467,76	6,47	7,37	700,77
RNPe	**2,87**	**2,87**	0,00	**3,32**	**3,32**	0,00	**5,69**	**5,69**	0,00
	Yeast			*Abalone*			*Heart*		
RNPI	4,58	5,53	799,33	5,13	6,08	799,81	4,05	5,00	797,73
RNPI-EM	**3,88**	**3,93**	9,92	4,33	**4,45**	34,48	3,43	3,43	0,00
EFuNN	4,75	5,65	698,02	4,35	4,56	64,62	4,27	5,14	646,09
eMLP	4,71	5,62	715,95	5,62	6,57	790,60	3,78	4,67	669,47
RNPe	3,93	4,63	409,68	**4,22**	4,46	76,92	**3,30**	**3,30**	0,00
	Concentric			*Spambase*			*Zoo*		
RNPI	4,20	5,16	798,40	5,92	6,88	799,97	3,72	4,67	795,08
RNPI-EM	**2,76**	**2,76**	0,00	4,06	5,65	3795,96	3,88	4,05	47,10
EFuNN	4,06	4,53	197,33	5,76	6,54	511,81	3,98	4,17	55,50
eMLP	3,64	4,38	453,80	5,50	6,28	491,09	**3,64**	4,01	132,24
RNPe	2,78	2,78	0,00	**4,05**	**4,20**	43,18	3,74	**3,93**	56,40

quando as técnicas são treinadas com somente um subconjunto, e na coluna *sub 1-9* quando treinadas com nove subconjuntos. A terceira coluna mede a taxa de crescimento TA do tempo de classificação. O menor tempo de classificação para cada base de dados está realçado em negrito. Os valores de tempo estão em milissegundos.

Nota-se, da tabela, que, para oito bases de dados, a técnica proposta RNPe foi a mais rápida na classificação, tanto quando foi treinada com um subconjunto, como quando foi treinada com nove subconjuntos. A RNPI-EM obteve tempos similares à RNPe em algumas bases de dados. O tempo reduzido dessas duas técnicas está relacionado à baixa complexidade de suas arquiteturas.

As redes eMLP e EFuNN necessitaram de uma quantidade de tempo consideravelmente maior. Em alguns casos, como as bases *Car*, *CNAE-9*, *Abalone* e *Spambase*, o tempo de classificação dessas redes foi mais de 10 vezes maior do que da RNPe. Essa diferença foi maior para as bases com uma maior diferença entre o número de neurônios da RNPe e as redes eMLP e EFuNN.

A RNPI, além de precisar, na média, de um tempo maior que as outras técnicas para classificar as amostras de um subconjunto, a taxa de aumento do tempo é a maior entre as redes avaliadas. Isso acontece porque todas as amostras de treinamento são incorporadas na

Tabela 6.9: Tempo médio necessário para a classificação de um subconjunto de cada base de dados (tempos medidos em milissegundos). TA é a taxa de aumento no tempo de classificação quando o número de amostras de treinamento aumenta de um subconjunto (*sub 1*) para nove subconjuntos (*sub 1-9*). Os menores tempos para cada base de dados estão realçados em negrito.

Técnicas	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA	sub 1	sub 1-9	TA
	Haberman			*Car*			*CNAE-9*		
IPNN	0,56	4,88	778,56	18,20	161,67	788,09	82,95	735,12	786,23
IPNN-EM	0,20	0,22	9,00	2,65	3,48	31,49	111,22	134,79	21,19
EFuNN	1,26	2,67	112,26	15,10	34,30	127,17	194,54	385,19	98,00
eMLP	1,66	4,52	171,99	19,90	60,57	204,39	237,72	974,02	309,74
RNPe	**0,09**	**0,12**	30,11	**0,95**	**1,90**	99,58	**14,81**	**37,74**	154,78
	Yeast			*Abalone*			*Heart*		
IPNN	13,73	122,44	791,51	112,36	1000,51	790,44	0,49	4,32	780,45
IPNN-EM	4,44	**6,16**	38,66	32,00	56,74	77,34	0,27	0,30	10,82
EFuNN	20,98	51,69	146,41	124,22	129,57	4,31	1,81	3,73	106,19
eMLP	31,70	132,35	317,49	285,55	1453,88	409,15	2,22	4,50	103,02
RNPe	**2,92**	10,62	263,77	**15,03**	**35,11**	133,66	**0,13**	**0,14**	6,15
	Concentric			*Spambase*			*Zoo*		
IPNN	36,06	323,69	797,57	210,04	1876,87	793,60	**0,08**	0,82	990,00
IPNN-EM	2,88	2,00	-30,43	17,16	384,90	2142,72	0,23	0,31	33,33
EFuNN	10,40	19,12	83,87	221,36	1044,87	372,03	0,91	0,96	5,45
eMLP	17,78	31,94	79,64	167,95	728,23	333,61	0,99	1,11	11,75
RNPe	**0,50**	**1,81**	262,50	**10,82**	**45,54**	320,75	**0,08**	**0,10**	25,00

arquitetura da rede, o que a torna cada vez mais lenta. Para o caso da base *Concentric*, com sua quantidade maior de amostras, o tempo de classificação da RNPI foi quase 200 vezes maior que o da rede neural proposta. Devido aos baixos valores e algumas imprecisões nas medidas de tempo (Zobel, 2005), houve ligeira diferença nos tempos calculados, e, assim, o valor de TA foi diferente de zero para redes que permaneceram com o mesmo tamanho de arquitetura.

Os testes de Friedman e de Nemenyi foram aplicados sobre o número de pesos e tempo de classificação, quando os classificadores foram treinados com os nove subconjuntos de treinamento (*sub 1-9*). A hipótese nula do teste de Friedman foi rejeitada, e as Figuras 6.10 e 6.11 mostram os resultados do teste de Nemenyi para o número de pesos e tempo de classificação, respectivamente.

Em ambos os testes, a RNPe e a RNPI-EM alcançaram os menores valores de *rank*, e a RNPe obteve o menor *rank* nos dois casos. Isto mostra terem sido estas as técnicas avaliadas com as menores complexidades em suas arquiteturas. Com relação ao teste sobre o número de parâmetros (Figura 6.12), as arquiteturas da RNPe e da RNPI-EM foram estatisticamente menores que as das RNPI e da EFuNN. Enquanto, para o tempo de classificação, ambas as RNPe e RNPI-EM, foram significativamente mais rápidas que as redes RNPI e eMLP.

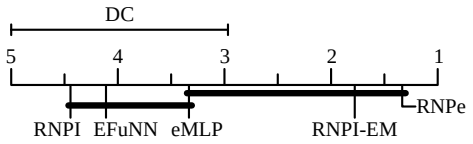


Figura 6.12: Comparação entre as técnicas incrementais em relação ao número de pesos usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Técnicas não significativamente diferentes entre si são conectadas por uma linha espessa. DC é a diferença crítica entre os *ranks* de duas técnicas para serem consideradas significativamente diferentes.

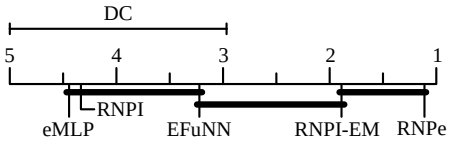
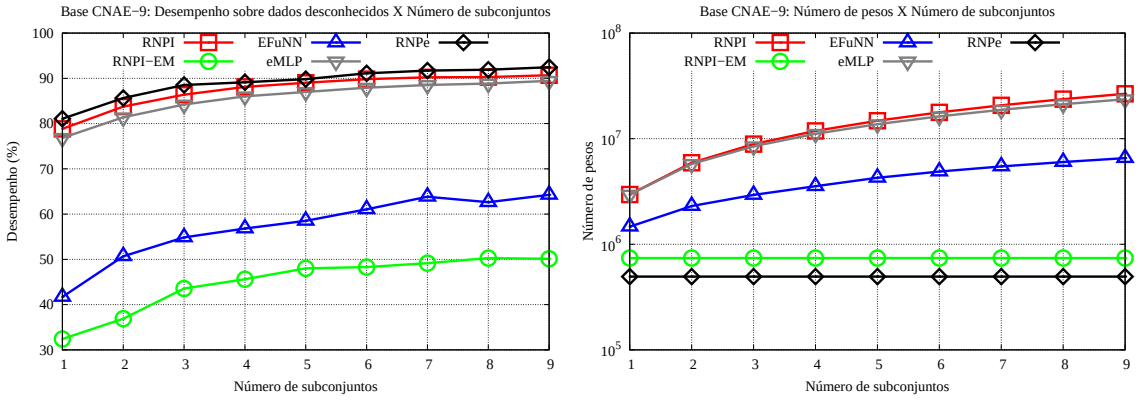


Figura 6.13: Comparação entre as técnicas incrementais em relação ao tempo de classificação usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Técnicas não significativamente diferentes entre si são conectadas por uma linha espessa. DC é a diferença crítica entre os *ranks* de duas técnicas para serem consideradas significativamente diferentes.

Observa-se que, embora o EFuNN tenha *rank* maior que o do eMLP para o número de pesos e parâmetros, em relação ao tempo de classificação, o EFuNN tendeu a necessitar de menos tempo do que o eMLP. Nesta situação, foi notado que a lógica *fuzzy* fez a EFuNN crescer, mas a quantidade de cálculos realizados no EFuNN foi menor do que no eMLP, resultando num tempo menor de classificação.

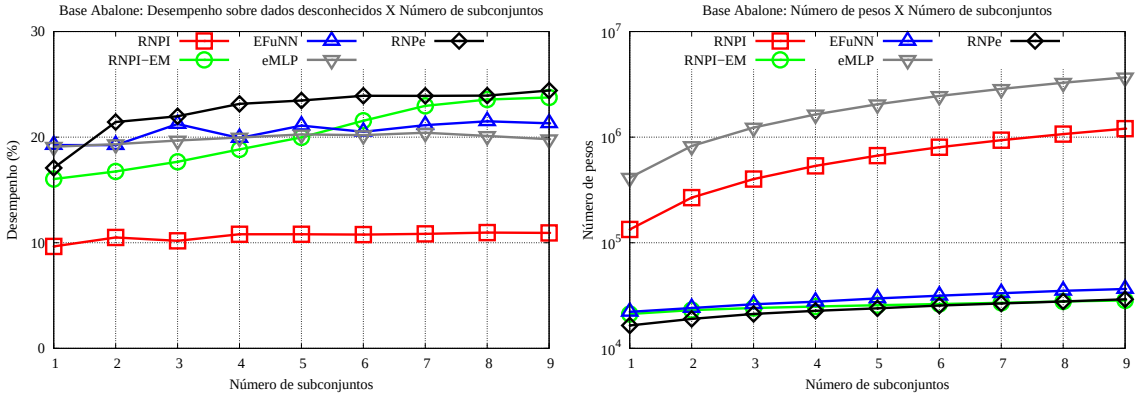
Alguns gráficos são apresentados a seguir para mostrar o aumento da acurácia sobre os dados desconhecidos e do tamanho da arquitetura das redes neurais incrementais à medida que mais amostras são usadas para treinamento. As Figuras 6.14 a 6.16 mostram esses gráficos para as bases *CNAE-9*, *Abalone* e *Spambase*. As medidas foram obtidas para cada subconjunto de treinamento usado. A RNPI é representada por um quadrado; a RNPI-EM por um círculo; o EFuNN por um triângulo; o eMLP por um triângulo invertido; e, a RNPe, por um losango. Os gráficos do lado esquerdo mostram o desempenho em função do número de subconjuntos usados para treinamento. Os gráficos do lado direito mostram o número de pesos numa escala logarítmica em função do número de subconjuntos usados para treinamento.

Os desvios padrões das medidas não foram colocados nos gráficos para obter uma melhor visualização dos resultados. No entanto, foi observado que os desvios padrões da acurácia reduziram ligeiramente à medida que mais dados de treinamento foram utilizados, sendo no-



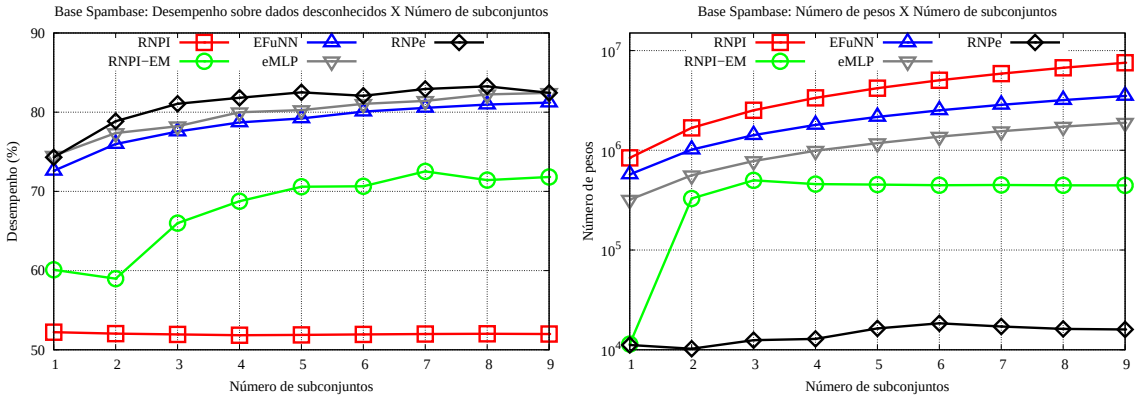
(a) (b)

Figura 6.14: *CNAE-9*: desempenho sobre dados desconhecidos (a) e número de pesos (b).



(a) (b)

Figura 6.15: *Abalone*: desempenho sobre dados desconhecidos (a) e número de pesos (b).



(a) (b)

Figura 6.16: *Spambase*: desempenho sobre dados desconhecidos (a) e número de pesos (b).

tada uma redução maior para a base *CNAE-9*. O desvio padrão médio dos algoritmos após o uso de todas as amostras de treinamento oscilou em torno de 2% para as bases *Abalone* e *Spambase*, e de 4% para a base *CNAE-9*. Por outro lado, à medida que mais dados de treinamento foram usados, os desvios padrões do tamanho da arquitetura das técnicas aumentaram ligeiramente, sendo os maiores desvios obtidos pela rede EFuNN (para a base *CNAE-9*), eMLP (para a base *Abalone*) e RNPI-EM (para a base *Spambase*).

Como as figuras mostram, o tamanho da arquitetura da RNPe foi mantida quase que constante, além de ser manter a menor ao longo do treinamento quando comparada com as das outras redes neurais. Ela teve um pequeno crescimento para a base *Abalone*, devido ao surgimento de novas classes com o aumento do número de amostras, e uma pequena oscilação na base *Spambase*, que surgiu durante a adaptação da rede à base de dados.

A RNPI-EM também obteve um arquitetura reduzida e estável para as bases *CNAE-9* e *Abalone*. Contudo, como explicado anteriormente, para a base *Spambase* (Figura 6.16(b)) ocorreu aumento repentino no tamanho de sua arquitetura, crescendo em média mais de 20 vezes entre o treinamento com o primeiro e o segundo subconjuntos. Um ou mais neurônios da RNPI-EM dividiram-se em outros 57 neurônios durante o treinamento, sendo 57 o número de atributos da base *Spambase*, e isso causou aumento repentino em sua arquitetura.

As arquiteturas da RNPI e do eMLP tiveram um tamanho bem elevado nas bases das três figuras. O aumento considerável em suas arquiteturas se deve à constante inclusão de neurônios. No caso da RNPI, todas as amostras são adicionadas em sua estrutura. O EFuNN obteve estrutura bem reduzida para a base *Abalone*, similar ao das redes RNPI-EM e RNPe, mas sua arquitetura foi mais elevada para as bases *CNAE-9* e *Spambase*. As redes RNPI, eMLP e EFuNN tiveram arquiteturas que continuaram crescendo com o aumento da quantidade de amostras de treinamento.

Com relação ao desempenho sobre os dados desconhecidos, a RNPe e o eMLP obtiveram resultados constantemente entre os melhores obtidos para cada base de dados, sendo a RNPe ligeiramente melhor. A RNPI-EM obteve resultados inferiores aos das demais técnicas para as bases *CNAE-9* (Figura 6.14(a)) e *Spambase* (Figura 6.16(a)). Essas bases de dados possuem um número mais elevado de atributos, e isto deve ter prejudicado o desempenho da RNPI-EM devido a imprecisões dos valores dos seus parâmetros.

Os desempenhos da EFuNN foram mais elevados para as bases *Abalone* e *Spambase*, porém foram menores para a base *CNAE-9*. Mas, como afirmado antes, essa base de dados é composta por mais de 99% de valores iguais a zero, e isto deve ter afetado severamente o desempenho do EFuNN. A base *Spambase* também é composta por uma quantidade grande de valores zeros, em torno de 77%, mas deve ter afetado menos no desempenho do EFuNN. Por fim, a RNPI obteve resultados mais elevados para a base *CNAE-9*, e menores para as

outras bases. Como essa rede neural não considera a probabilidade *a priori* das classes, ela foi melhor para a base *CNAE-9*, pois a quantidade de amostras por classe é equilibrada (EN = 1). Mas, para as bases *Abalone* e *Spambase*, onde existe desequilíbrio, ela obteve acurácia mais reduzida em relação às outras redes.

Dos experimentos realizados nessa seção, é possível observar que a RNPe não teve capacidade muito elevada para recuperar dados antigos. Entretanto, à medida que se aumenta a quantidade de dados de treinamento, seus parâmetros são melhores ajustados e o desempenho, sobre os dados não vistos e os que foram utilizados para o treinamento, tendem a elevar. Também o desempenho sobre os dados não vistos esteve dentre os melhores obtidos, com a vantagem da RNPe obter estrutura mais compacta, precisando tempo menor de processamento dos dados.

6.2.3 Comparação entre a metodologia proposta e a de Polikar et al. (2001) e conclusões gerais

O método proposto neste trabalho e o de Polikar et al. (2001) se assemelham no sentido do aprendizado ser realizado dividindo a base de dados em conjuntos e usando um conjunto de dados por vez para treinamento. Porém, se diferem no critério usado para dividir os dados e na forma de avaliar os conceitos de estabilidade e plasticidade. No primeiro método, é medida a capacidade de um classificador de aprender e reter informações de uma classe, sendo as amostras de uma classe fornecidas todas de uma só vez em um único conjunto de dados, não havendo padrões semelhantes nos outros conjuntos de treinamento. Por sua vez, no método de Polikar et al. (2001), é avaliada a habilidade do classificador de reter a informação das amostras de treinamento e aprimorar o desempenho de classificação sobre outras amostras não vistas durante o treinamento, à medida que novos conjuntos de dados são usados para treino. Contudo, os conjuntos de treinamento no método de Polikar et al. (2001) contêm amostras de todas as classes.

Assim, no método de Polikar et al. (2001) é possível o algoritmo ser treinado com amostras semelhantes às usadas para treinamento num momento anterior. Dessa forma, existe possibilidade de haver aumento de desempenho sobre os dados mais antigos de treinamento, como percebido nos experimentos. Já o procedimento realizado na Seção 6.2 é mais rigoroso, pois todas as amostras de uma classe estão contidas em um único subconjunto, e, assim, novos dados de treinamento não adicionam mais informações às classes já utilizadas para treino. Com isso, a tendência é a retenção de informação diminuir com o aumento na quantidade de dados.

Além disso, a avaliação de Polikar et al. (2001) é em função da porcentagem de amostras

corretamente identificadas, enquanto no método proposto a avaliação é realizada em função da porcentagem de classes reconhecidas. Com formas diferentes de avaliar, podem surgir conclusões diferentes, e até conflitantes. Por exemplo: a RNPI apresentou valor alto para Retenção no método proposto, pois manteve quantidade razoável de informação sobre todas as classes, onde Retenção mede o grau de estabilidade da técnica. Entretanto, no método de Polikar et al. (2001), o desempenho da RNPI sobre os dados antigos (forma proposta para medir estabilidade) foi menor frente aos outros algoritmos, mostrando que a RNPI não reconheceu apropriadamente as amostras de treinamento.

Com o eMLP, aconteceu também outro caso conflitante. No método proposto, o eMLP alcançou níveis baixos de Retenção, por conseguir reter poucas informações sobre as classes. Já no método de Polikar et al. (2001), aconteceu o oposto: obteve alguns dos melhores resultados sobre as amostras usadas para treinamento, sugerindo nível alto de estabilidade.

O método proposto também consegue verificar o comportamento de um algoritmo num caso crítico em relação à ordem em que as amostras de treinamento são usadas, o que o método de Polikar et al. (2001) não consegue avaliar. Além disso, o método proposto consegue, de certa forma, simular um caso em que o ambiente é não estacionário, ocorrendo mudanças nas características do ambiente, sendo essas mudanças representadas pela adição de uma nova classe a cada porção de dados de treinamento. Ainda assim, mesmo que o procedimento apresentado em (Polikar et al., 2001) não represente uma situação extrema, ela é útil para avaliar uma situação mais comum, quando as amostras das classes estão distribuídas de forma mais aleatória.

Com relação às conclusões obtidas para a RNPe, foi notado que a rede proposta possui uma habilidade para aprender novas informações maior do que a sua capacidade de manter informações antigas. Quando são usadas amostras de treinamento semelhantes às que foram usadas num momento posterior, a RNPe tem a tendência a elevar o desempenho sobre os dados já usados para treinamento, sendo em alguns casos observado que o desempenho sobre os dados antigos é semelhante aos dados não vistos. Isto acontece, pois os dados possuem uma distribuição similar, e com uma quantidade maior de dados, o modelo gerado pela RNPe é refinado. Também foi observado que a RNPe possui uma arquitetura estável, que não cresce muito com o aumento da quantidade de amostras de treinamento, assim como o tempo de processamento mantém bem estável, quando comparada a outras técnicas.

Por outro lado, quando os novos dados de treinamento diferem dos dados antigos, a tendência é diminuir o grau de estabilidade quanto mais dados são usados para treinar. Foi observado que esta situação se agrava para casos em que a superfície de separação entre classes é muito complexa (valores altos de N4) e na qual os atributos possuem baixo poder discriminativo entre as classes (valores baixos de F1).

6.3 Experimentos com Técnicas Clássicas de Aprendizado

Como discutido na Seção 1.2, as técnicas de aprendizado incremental podem ser utilizadas em tarefas nas quais métodos clássicos de aprendizado de máquina costumam ser usados. Assim, essa seção tem como finalidade comparar as técnicas incrementais, principalmente a RNPe, com as clássicas.

Outro objetivo dessa seção, é verificar se a RNPe consegue, de fato, realizar um compromisso razoável entre qualidade de resposta (eficácia) e complexidade da arquitetura (eficiência) quando comparada a outros algoritmos de aprendizado de máquina.

Os experimentos, resultados e análises são informados na Subseção 6.3.1, enquanto as conclusões dos experimentos encontram-se na Subseção 6.3.2.

6.3.1 Experimentos e resultados

Nessa seção, a RNPe é comparada com algumas técnicas clássicas de classificação e algumas redes neurais incrementais apresentadas no Capítulo 3. Nos experimentos realizados nessa seção, são avaliadas as seguintes características dos classificadores: a capacidade de generalização e a complexidade das técnicas.

A generalização pode ser entendida como a capacidade da técnica classificar corretamente padrões (amostras) não vistas pelo mesmo durante o treinamento. A métrica utilizada nos experimentos para medir essa capacidade é a acurácia, apresentada na Seção 5.3.2. A complexidade de cada técnica é avaliada indiretamente através de duas medidas: a primeira, calculando o tamanho da estrutura de cada técnica para realizar a classificação (ou seja, o espaço ocupado em memória por cada técnica); e, a segunda, medindo o tempo que cada técnica necessita para classificar as amostras de cada base de dados. Essas duas medidas retornam uma estimativa aproximada da complexidade de cada método.

Além da RNPe, foram utilizados os seguintes classificadores nos experimentos: RNP, MLP, kNN, Rocchio, VG-RAM WNN, eMLP, EFuNN e RNPI-EM. Uma breve descrição das técnicas MLP, kNN, Rocchio e VG-RAM WNN é apresentada na Seção 5.3.1. A RNP é descrita no Apêndice C. As demais são apresentadas no Capítulo 3. A técnica RNPI não é usada para comparação, pois seu algoritmo é semelhante ao da RNP, com a diferença da RNP considerar a probabilidade *a priori* das classes.

Para ajustar os parâmetros de cada modelo, foi utilizado o procedimento descrito na Seção 5.3.4. Os testes foram realizados sobre as 20 bases de dados da Tabela 5.1 usando

Tabela 6.10: Acurácia média das técnicas incrementais e clássicas. Valores em porcentagem. O maior valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Bases de dados	RNP	kNN	MLP	Rocchio	WNN	RNPI-EM	EFuNN	eMLP	RNPe
Balance	90,24	95,04	90,23	73,28	**97,29**	88,78	90,07	87,36	91,51
Blood	76,19	72,72	76,59	56,02	75,66	75,92	64,31	64,44	**76,73**
Iris	94,00	96,00	86,67	**97,33**	66,67	42,67	**97,33**	91,33	**97,33**
Haberman	72,87	70,28	73,53	71,57	72,89	**73,88**	59,81	59,49	73,87
Car	87,27	**89,41**	78,72	64,93	87,50	79,34	84,55	84,26	81,25
CNAE-9	90,74	85,28	95,09	83,80	**95,56**	54,54	62,50	89,72	92,22
WebKB-4	63,56	64,06	**88,85**	72,64	80,59	74,23	65,40	78,92	82,28
Wine	73,63	73,63	73,59	72,42	74,28	**96,08**	64,02	77,06	96,05
Reuters-8	94,66	94,61	**97,21**	85,52	92,12	77,29	85,98	95,44	94,75
Yeast	50,94	53,10	54,92	47,78	47,84	38,94	50,20	48,78	**54,99**
Abalone	**25,90**	19,99	24,92	3,74	23,85	24,16	21,40	19,61	24,04
Heart	65,93	67,41	**81,85**	65,93	76,30	55,56	65,56	65,19	72,59
Sonar	84,60	83,62	78,31	65,88	**87,55**	53,38	84,12	81,24	80,29
Segmentation	74,63	**93,51**	84,42	56,93	86,71	60,87	82,68	89,87	90,09
Texture	91,22	**99,56**	94,58	88,87	99,31	85,65	20,45	98,35	97,69
Gaussian	72,94	72,58	49,16	49,30	50,72	**82,20**	51,50	68,58	81,20
Concentric	97,08	**98,48**	67,31	48,56	61,84	82,28	97,64	97,48	96,60
Spambase	66,27	86,46	80,74	57,05	**94,91**	75,22	81,00	82,35	85,66
Diabetes	66,93	72,79	70,55	63,15	63,41	68,66	56,65	64,85	**76,83**
Zoo	**97,00**	**97,00**	85,53	85,00	91,18	84,09	92,18	95,09	95,09
Média	76,83	79,28	76,64	65,49	76,31	68,69	68,87	76,97	**82,05**
Média dos DPs	4,22	4,07	6,04	4,86	4,63	5,68	5,14	4,52	**3,42**

o procedimento de 10 *fold cross-validation* sobre cada base, e foram calculados os valores médios da acurácia, do número de parâmetros e do tempo de classificação das amostras.

Inicialmente, a capacidade de generalização das técnicas avaliadas é analisada e, para tanto, a Tabela 6.10 mostra a acurácia média obtida por cada técnica sobre cada base de dados. Cada linha da Tabela 6.10 é uma base de dados e cada coluna corresponde a uma técnica. A penúltima linha da tabela informa a acurácia média alcançada por cada técnica para todas as bases de dados, e a última linha informa a média dos desvios padrões obtidos pelas técnicas. Todos os valores estão em porcentagem e o valor de acurácia mais elevado para cada base de dados está realçado em negrito. Quanto maior o valor da acurácia, maior a capacidade de generalização da técnica. A maior acurácia média e a menor média dos desvios padrões também estão realçadas.

Analisando a Tabela 6.10, nota-se que nenhuma técnica obteve o melhor resultado para todas as bases de dados, mas uma boa parte das técnicas mostrou ser especialista em uma ou mais tarefas, e em algumas outras tenderam a não apresentar um resultado competitivo com as demais técnicas. Um exemplo foi o MLP, obtendo resultados elevados para as bases de texto *WebKB-4* e *Reuters-8*, mas alcançou acurácia reduzida para as bases *Iris* e *Gaussian*.

A RNPe obteve os melhores resultados para as bases *Blood*, *Iris*, *Yeast* e *Diabetes*, e resultados próximos aos melhores para as bases *Haberman*, *Wine*, *Abalone*, *Gaussian*, *Concentric* e *Zoo*. As bases de dados em que houve maior diferença entre o melhor resultado e o obtido pela RNPe foram *Car*, *Heart* e *Spambase*: as diferenças entre as acurácias foram aproximadamente de 10%. As bases *Iris*, *Wine* e *Zoo* possuem valores elevados para a medida F1 e valores baixos para a medida N4, o que, como visto na Seção 6.2.1, afetam de forma positiva no desempenho da RNPe.

As bases *Gaussian* e *Concentric* são formadas por duas classes com médias aproximadamente iguais, só sendo diferente o espalhamento das amostras de cada classe. Logo, cada classe dessas bases de dados pode ser representada por uma função Gaussiana sem perda de muita informação, e, assim, a RNPe pode conseguir um bom desempenho sobre essas bases. O baixo valor de acurácia para a base *Car* deve ter ocorrido porque ela é fortemente polarizada por uma das quatro classes, que detém 70% das amostras da base, além de possuir valor baixo para F1 e ligeiramente alto para N4, e o desempenho dos outros classificadores deve ter sido menos afetado por essas características. A maior parte das amostras da base *Blood* (aproximadamente 75%) também está associada a uma de suas duas classes, mas essa característica deve ter afetado quase todos os classificadores, já que a acurácia de seis entre nove classificadores oscilou em torno de 75%. Já para as bases *Heart* e *Spambase*, a RNPe só obteve acurácia inferior em relação a duas das oito técnicas às quais foi comparada, como pode ser visto na Tabela 6.10, o que pode ter sido mais mérito desses classificadores do que necessariamente deficiência do método proposto.

Para nenhuma base de dados a acurácia obtida pela RNPe foi inferior aos resultados alcançados pela técnica Rocchio. E a RNPe foi inferior a RNPI-EM em quatro bases de dados (*Haberman*, *Wine*, *Abalone* e *Gaussian*), sendo a maior diferença entre a RNPI-EM e a RNPe de 1% para a base *Gaussian*. Para essas mesmas quatro bases de dados, a RNPe também apresentou acurácia alta em relação aos demais classificadores, sugerindo que um MMG consegue modelar bem as classes dessas bases de dados, pois ambas as redes são baseadas em MMG. A diferença média entre o melhor resultado obtido para cada base e o resultado da RNPe foi de 3,20%. Para comparação, a diferença média entre os melhores resultados e os resultados do kNN, que obteve os melhores resultados em cinco bases de dados, foi de 5,98%. A RNPe apresentou maior regularidades nos resultados, e, na média, obteve um maior desempenho.

Com relação às outras técnicas, kNN foi a que obteve a maior quantidade de melhores resultados para as bases de dados usadas na avaliação, totalizando cinco, seguida de perto pela rede neural sem peso VG-RAM WNN e a RNPe, ambas totalizando quatro. O VG-RAM WNN obteve resultados elevados principalmente para as bases *Balance* e *Spambase*. O kNN obteve melhores resultados para algumas bases de dados, principalmente para bases

de dados com valores baixos de N1, N2, N3 e N4, como indicado pelo teste estatístico de Kendall. Também foi observado em alguns casos, como *CNAE-9*, *WebKB-4*, *Wine* e *Heart*, que quando a proporção de amostras pelo número de atributos (às vezes chamada de densidade) é baixa, a acurácia do kNN é relativamente inferior aos dos outros algoritmos. Em (Duda et al., 2001) foi provado que para um número infinito de amostras (densidade elevada), a taxa de erro de classificação (100% - acurácia) do kNN não é maior do que duas vezes a taxa de erro mínima da teoria de decisão Bayesiana. No entanto, para uma grande quantidade de amostras de treinamento a estrutura do kNN é grande (como será mostrado na Tabela 6.11). Para valores mais elevados de N3, foi observado que a RNPe tende a apresentar resultados melhores do que o kNN, embora essa relação não tenha sido estatisticamente comprovada pelo teste de Kendall.

RNPI-EM, Rocchio e EFuNN apresentaram, na média, os valores de acurácia mais baixos entre as técnicas. RNPI-EM tendeu a apresentar uma acurácia menor para bases de dados com distribuição igual de amostras por classe, caso das bases *Iris*, *CNAE-9*, *Sonar* e *Segmentation*. Como mencionado anteriormente, devido às imprecisões nos valores estimados dos parâmetros da RNPI-EM, esse classificador é induzido a polarizar sua classificação mais em algumas classes do que em outras.

EFuNN obteve medidas de acurácia menores para as bases *CNAE-9* e *WebKB-4*, que possuem grande quantidade de elementos iguais a zero e, como tais elementos não causam efeito na lógica *fuzzy*, podem ter prejudicado na capacidade de generalização da rede. *Reuters-8* também possui uma quantidade grande de elementos iguais a zero, mas, como há um maior desequilíbrio no número de amostras por classe, como visto pelo baixo valor de EN (Tabela 5.1, página 103), o valor de acurácia é polarizado pela proporção da classe mais abundante.

Rocchio apresenta resultados melhores para bases de dados com distribuição espacial mais simples, como distribuição unimodal cujas variâncias dos valores dos atributos são semelhantes. Um exemplo desse tipo de base é a *Iris*. No entanto, para bases mais complexas, como *Abalone* e *Concentric*, os resultados são bem inferiores aos das outras técnicas.

Embora eMLP não tenha alcançado o melhor resultado para nenhuma base de dados, a acurácia média dessa técnica sobre as bases de dados foi próxima às das técnicas MLP, VG-RAM WNN e RNP, e superior às das técnicas Rocchio, RNPI-EM e EFuNN. A RNP desempenhou melhor para as bases *Abalone*, *Concentric*, *Haberman* e *Zoo*, sendo que, para essas quatro bases, a RNPe também obteve valores elevados de acurácia, indicando comportamento um pouco semelhante entre essas duas técnicas. O MLP obteve alguns bons resultados, principalmente para as bases *CNAE-9*, *WebKB-4*, *Reuters-8*, *Heart*, *Yeast* e *Abalone*, mas, para certas bases de dados, apresentou acurácia baixa quando comparada às de outras técnicas, principalmente para as bases *Car*, *Concentric*, *Gaussian* e *Zoo*. Além disso,

o MLP foi a única técnica avaliada em que as amostras foram usadas várias vezes durante o treinamento.

Na média, os desvios padrões da acurácia foram aproximadamente entre 4% e 5%, sendo o menor desvio obtido pela RNPe. O maior desvio padrão foi obtido pelo MLP, pois, além de seu desempenho depender dos pesos iniciais dos neurônios, durante o treinamento o algoritmo pode estagnar em um ótimo local, o que prejudica a qualidade de sua resposta e aumenta o desvio padrão de seu desempenho.

Para analisar a complexidade das técnicas, são utilizadas as medidas de tamanho das estruturas e tempo de classificação do conjunto de teste. A Tabela 6.11 apresenta o espaço médio ocupado em memória por cada técnica, medido em $\log_{10}$ do número de *bits*. O logaritmo do valor foi calculado para melhor visualização dos dados, sendo a diferença de uma unidade entre o tamanho dos modelos indicativo do tamanho de um deles ser 10 vezes maior que do outro. Cada linha da Tabela 6.11 é uma base de dados e cada coluna corresponde a uma técnica. O tamanho médio de cada técnica sobre todas as bases de dados é indicado na penúltima linha da tabela. A estrutura que requisitou a menor quantidade de espaço em memória para cada base de dados está realçada em negrito. Uma estrutura de tamanho reduzido sugere que a complexidade da técnica é baixa. A linha "Média" da Tabela 6.11 informa o $\log_{10}$ da média do tamanho das estruturas, e não a média dos logaritmos do número de *bits*. De forma semelhante, a linha "Média dos DPs" informa o $\log_{10}$ da média dos desvios padrões do número de *bits*. O valor $-\infty$ indica zero *bit*.

De acordo com os resultados da Tabela 6.11, Rocchio obteve a menor estrutura para todas as bases de dados. As redes RNPI-EM, RNPe também obtiveram estruturas reduzidas. Ambas as redes alcançaram estruturas reduzidas graças aos procedimentos internos de controle do tamanho de suas arquiteturas. A complexidade da estrutura da técnica Rocchio é independente da quantidade de amostras de treinamento, sendo só em função da quantidade de classes e número de atributos do problema de classificação.

A arquitetura do MLP foi pequena, se comparada com as técnicas RNP, kNN e VG-RAM WNN. O MLP possui arquitetura fixa, não se alterando ao longo do treinamento, permitindo uma compactação de informação, independentemente da quantidade de amostras de treinamento.

Por outro lado, as técnicas kNN, RNP e VG-RAM WNN alcançaram estruturas maiores, pois, além de não possuírem procedimentos para redução da estrutura, o tamanho de suas estruturas cresce linearmente com a quantidade de amostras de treinamento. As redes EFuNN e eMLP também tiveram grande quantidade de neurônios acrescidos em suas arquiteturas.

Os classificadores tiveram suas estruturas maiores para as bases *CNAE-9*, *WebKB-4* e

Tabela 6.11: Tamanho médio das estruturas das técnicas incrementais e clássicas (espaço ocupado em memória pelas técnicas). Valores em $\log_{10}(bits)$. A estrutura de menor tamanho para cada base de dados está realçada em negrito. DPs significa desvios padrões.

Bases de dados	RNP	kNN	MLP	Rocchio	WNN	RNPI-EM	EFuNN	eMLP	RNPe
Balance	4,86	4,86	3,35	**2,58**	5,51	3,15	5,04	4,38	3,82
Blood	5,03	5,03	2,97	**2,51**	4,94	3,06	5,15	4,73	3,00
Iris	4,24	4,24	3,35	**2,58**	3,03	3,15	3,48	3,46	3,08
Haberman	4,42	4,42	3,05	**2,28**	4,25	2,89	4,57	4,26	2,87
Car	5,48	5,48	3,53	**2,89**	6,25	3,42	5,26	4,98	3,32
CNAE-9	7,43	7,43	6,92	**5,39**	8,00	5,87	6,87	7,37	5,69
WebKB-4	8,56	8,56	7,46	**5,58**	8,69	6,06	8,52	8,55	5,89
Wine	4,82	4,82	3,60	**3,10**	5,10	4,54	4,91	4,37	3,71
Reuters-8	8,82	8,82	6,68	**5,89**	9,01	6,62	7,04	8,76	6,19
Yeast	5,53	5,53	3,89	**3,41**	6,84	3,92	5,66	5,62	4,47
Abalone	6,08	6,08	4,41	**3,95**	7,77	4,46	4,75	6,57	4,38
Heart	5,00	5,00	3,57	**2,92**	6,00	3,43	5,09	4,63	3,30
Sonar	5,56	5,56	4,78	**3,58**	5,88	4,07	5,48	5,09	5,34
Segmentation	6,10	6,10	4,52	**3,63**	7,12	4,12	5,53	5,39	5,62
Texture	6,80	6,80	5,13	**4,15**	8,49	5,03	7,09	5,53	5,83
Gaussian	5,76	5,76	3,29	**2,41**	5,76	2,98	5,92	5,54	2,94
Concentric	5,16	5,16	2,68	**2,11**	5,51	2,76	4,53	4,14	2,78
Spambase	6,88	6,88	4,75	**3,56**	7,83	5,34	6,55	6,27	4,13
Diabetes	5,25	5,25	3,47	**2,71**	6,20	3,95	5,31	4,92	3,14
Zoo	4,67	4,67	3,81	**3,55**	3,76	4,05	4,24	4,02	3,94
Média	7,69	7,69	6,28	**4,82**	7,98	5,47	7,22	7,64	5,28
Média dos DPs	3,62	3,62	**−∞**	**−∞**	3,84	5,19	5,89	4,87	3,44

Reuters-8, bases com quantidade razoável de amostras, mas com grande número de atributos. Por sua vez, para bases de dados com poucas amostras e atributos, como *Iris* e *Blood*, as estruturas dos algoritmos foram menores. Como indicado também pela Tabela 6.11, o tamanho das estruturas de diferentes técnicas variaram muito, havendo casos em que, para uma mesma base de dados, a menor estrutura foi aproximadamente 1000 vezes menor do que a maior.

Na média, a RNPe obteve estrutura mais de 200 vezes menor que kNN, RNP, VG-RAM WNN e eMLP, mais de 80 vezes menor que EFuNN, aproximadamente 10 vezes menor que MLP, ligeiramente menor que a RNPI-EM, e, cerca de três vezes maior que Rocchio, a técnica avaliada com a menor estrutura. Com isso, é mostrado que o modelo neural proposto apresenta estrutura bastante compacta.

Os desvios padrões do tamanho das estruturas das técnicas MLP e Rocchio foram iguais a zero ($\log_{10}(0) = -\infty$), pois as mesmas possuem uma estrutura de tamanho fixo para cada base de dados, que não cresce e nem diminui com o uso de mais dados de treinamento. Os valores dos desvios de padrões variaram bastante para cada base de dados e algoritmo, sendo o maior desvio padrão médio obtido pela rede EFuNN, indicando que o tamanho de sua arquitetura pode variar bastante. A RNPI-EM obteve um desvio padrão médio na mesma

Tabela 6.12: Tempo médio necessário para as técnicas incrementais e clássicas classificarem uma amostra de cada base de dados. Valores em milissegundos. O menor tempo para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Bases de dados	RNP	kNN	MLP	Rocchio	WNN	RNPI-EM	EFuNN	eMLP	RNPe
Balance	0,40	0,22	**0,02**	0,03	1,22	**0,02**	0,17	0,16	0,11
Blood	0,42	0,26	0,02	0,02	0,46	**0,01**	0,19	0,27	0,03
Iris	0,11	0,10	0,02	0,02	0,04	**0,01**	0,10	0,08	0,03
Haberman	0,18	0,14	0,02	0,02	0,19	**0,01**	0,11	0,15	0,03
Car	0,97	0,62	**0,03**	0,04	5,82	0,04	0,22	0,36	0,06
CNAE-9	7,46	8,70	4,04	0,99	295,21	1,78	3,88	8,97	**0,88**
WebKB-4	89,09	109,82	14,05	6,07	1488,73	7,04	119,45	113,53	**5,66**
Wine	0,20	0,07	0,06	**0,02**	1,37	0,15	0,14	0,14	0,05
Reuters-8	160,92	197,63	**7,02**	9,01	2947,20	14,02	68,39	181,93	8,68
Yeast	0,86	0,53	0,06	**0,04**	26,84	0,06	0,40	0,90	0,08
Abalone	2,43	1,62	**0,06**	0,09	194,80	0,18	0,35	3,51	0,12
Heart	0,18	0,12	0,05	**0,03**	4,40	0,04	0,16	0,18	**0,03**
Sonar	0,27	0,23	0,23	**0,08**	2,40	0,10	0,34	0,30	0,16
Segmentation	1,50	1,06	0,09	**0,06**	40,41	0,09	0,30	0,53	0,36
Texture	4,33	3,59	**0,19**	0,20	928,00	0,40	5,29	0,54	0,49
Gaussian	3,67	1,62	0,03	**0,02**	1,88	0,04	1,24	1,77	0,04
Concentric	1,32	0,74	**0,01**	**0,01**	3,46	0,04	0,10	0,17	0,04
Spambase	4,13	3,60	0,16	**0,15**	204,09	1,29	1,74	1,58	0,16
Diabetes	0,66	0,27	0,03	**0,01**	4,92	0,08	0,26	0,31	0,04
Zoo	0,11	0,05	0,05	**0,02**	0,05	0,06	0,13	0,13	0,05
Média	13,96	16,55	1,31	**0,85**	307,58	1,27	10,15	15,78	**0,85**
Média dos DPs	0,15	0,15	0,06	0,13	0,44	0,25	0,56	0,13	**0,03**

ordem de grandeza que o tamanho médio de sua arquitetura. Embora tenha sido observado que para certas bases de dados o tamanho de sua arquitetura tenha se mantido constante (desvio padrão igual a zero), para outras bases de dados o tamanho de sua arquitetura variou muito e, assim, nem sempre obtendo uma arquitetura de tamanho reduzido.

O tempo necessário de cada técnica para realizar a classificação de uma amostra de cada base de dados é informado na Tabela 6.12. Cada linha dessa tabela corresponde a uma base de dados, e cada coluna a um algoritmo. Um baixo tempo de classificação sugere que a técnica possui pequena complexidade. Os tempos na tabela estão em milissegundos, e o menor tempo para cada base de dados está realçado em negrito. As duas últimas linhas da tabela indicam o tempo médio e a média dos desvios padrões, respectivamente, de cada algoritmo.

Os algoritmos Rocchio, RNPI-EM, RNPe e MLP necessitaram de quantidade menor de tempo para classificar as amostras de cada base de dados, como pode ser visto na Tabela 6.12. Esses algoritmos gastaram um tempo médio de classificação de cada amostra menor do que 1,5 milissegundos. Esses são também os algoritmos com o menor número de parâmetros e pesos, indicando serem possuidores de uma complexidade menor que as outras técnicas.

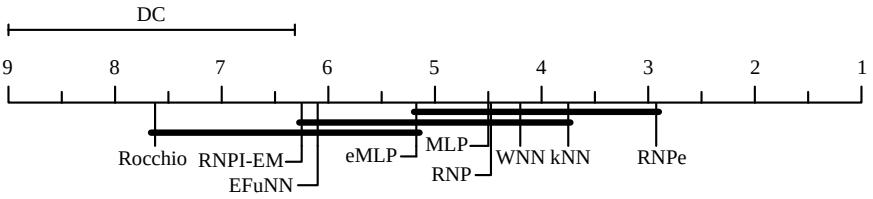


Figura 6.17: Comparação entre os classificadores em relação à acurácia usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

Por outro lado, as técnicas VG-RAM WNN, RNP, kNN, eMLP e EFuNN realizaram a classificação de cada amostra em um tempo maior, gastando um tempo médio maior do que 10 milissegundos, e, para o VG-RAM WNN, foi necessário tempo médio consideravelmente maior. O tempo maior de classificação ocorreu porque essas técnicas possuem estruturas maiores que das outras quatro técnicas mais rápidas (Rocchio, RNPI-EM, RNPe e MLP) e, ou, realizam cálculos mais complexos para a classificação.

As bases de dados que consumiram mais tempo para classificação foram *WebKB-4* e *Reuters-8*, pois, para essas bases, as técnicas precisaram de estruturas maiores. Além disso, as amostras dessas bases de dados possuem quantidade elevada de atributos. Outra característica da Tabela 6.12 é a discrepância no tempo gasto por cada algoritmo para classificação. Houve casos em que, para uma mesma base de dados, o tempo gasto para classificar uma amostra variou desde algumas centenas de microssegundos para até centenas de milissegundos.

As média dos desvios padrões dos tempos de classificação foram baixas, sendo todos menores do que 1 milissegundo, e a maioria menor do que 0,2 milissegundos. O menor desvio padrão médio foi obtido pela RNPe, graças ao tamanho reduzido e a estabilidade de sua arquitetura, que não variou muito. A rede EFuNN obteve o maior desvio médio que, em parte, deve ter sido provocado pela grande variação no tamanho de sua arquitetura, como visto nos desvios padrões da Tabela 6.11.

Os testes estatísticos de Friedman e de Nemenyi foram realizados sobre os resultados informados nas Tabelas 6.10, 6.11 e 6.12 (com exceção das linhas com os valores médios de cada algoritmo sobre as bases de dados). A hipótese nula de que os resultados dos algoritmos são equivalentes foi descartada. Os resultados do teste de Nemenyi são apresentados na Figura 6.17, para a medida de acurácia; na Figura 6.18, para a medida de tamanho das estruturas; e, na Figura 6.19 para a medida de tempo de classificação.

A Figura 6.17 mostra que, em termos da métrica acurácia, a RNPe foi estatisticamente superior às técnicas Rocchio, RNPI-EM e EFuNN; as técnicas kNN, VG-RAM WNN, RNP,

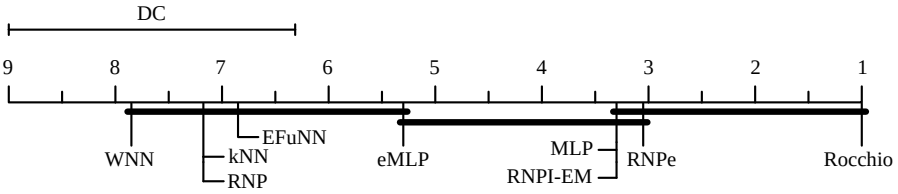


Figura 6.18: Comparação entre os classificadores com relação ao tamanho de suas estruturas usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

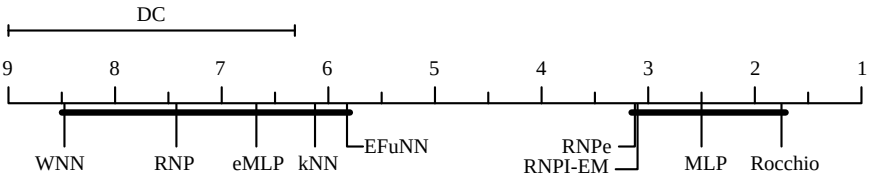


Figura 6.19: Comparação entre os classificadores com relação ao tempo de classificação usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

MLP foram superiores a Rocchio. Com relação ao tamanho das estruturas, a Figura 6.18 indica que as técnicas Rocchio, RNPI-EM, RNPe e MLP foram significativamente menores que VG-RAM WNN, kNN, RNP e EFuNN. Além disso, Rocchio também foi menor que eMLP.

Os resultados indicados na Figura 6.19 sugerem a existência de dois grupos de algoritmos estatisticamente diferentes entre si: algoritmos rápidos, formado pela RNPe, Rocchio, MLP e RPNI-EM, e algoritmos mais lentos, formado pelos algoritmos restantes. Esses quatro algoritmos são os mesmos que obtiveram os menores *ranks* nos resultados da Figura 6.18, podendo-se concluir terem sido os avaliados que possuem a menor complexidade.

Com esses resultados, pode-se observar que a rede neural proposta, a RNPe, foi superior em desempenho as técnicas Rocchio e RNPI-EM, que foram as técnicas avaliadas com algumas das estruturas e tempo de classificação mais reduzidos. A RNPe apresentou estrutura e tempo de classificação significativamente menor que VG-RAM WNN e kNN, que estão entre as melhores técnicas avaliadas. Além disso, os testes não apontaram diferenças estatisticamente significantes entre os desempenhos da RNPe com as técnicas VG-RAM WNN e kNN, e nem diferenças estatisticamente relevantes entre os tamanhos das estruturas e tempos de Rocchio e da RNPI-EM. Assim, é possível concluir que a RNPe alcançou compromisso satisfatório entre desempenho e complexidade da arquitetura.

A Tabela 6.13 compara os resultados obtidos pela RNPe com alguns resultados encon-

Tabela 6.13: Resultados encontrados na literatura em comparação com os resultados alcançados pela RNPe. Valores obtidos para a métrica acurácia (valores em porcentagem).

Base de dados	RNPe	Literatura	Técnica	10 *fold*
Balance	91,51	90,82 (Holmes et al., 2002)	AdaBoost.MH	x
Blood	76,73	77,71 (Kobos e Mandziuk, 2012)	SVM	x
Iris	97,33	97,80 (Zhou e Goldman, 2004)	Comitê de redes neurais	x
Haberman	73,87	74,50 (Zhang e Street, 2008)	Comitê de SVMs	x
Car	81,25	93,30 (Tan e Dowe, 2002)	Árvore de decisão	x
CNAE-9	92,22	98,33[1]	VG-RAM WNN	
WebKB-4	82,28	85,82 (Cardoso-Cachopo, 2007)	SVM com tf-idf	
Wine	96,05	97,70 (Zhou e Goldman, 2004)	Comitê de redes neurais	x
Reuters-8	94,75	96,98 (Cardoso-Cachopo, 2007)	SVM com tf-itf	
Yeast	54,99	60,30 (Allwein et al., 2000)	SVM	x
Abalone	24,04	25,70 (Tan e Dowe, 2002)	Árvore de decisão	x
Heart	72,59	83,52 (Zhang e Street, 2008)	Comitê de SVMs	x
Sonar	80,29	81,20 (Zhou e Goldman, 2004)	Comitê de redes neurais	x
Segmentation	90,09	96,74 (Holmes et al., 2002)	Várias árvore de decisão	x
Texture	97,69	99,70 (Min e Cho, 2007)	Comitê de redes neurais com K-means	x
Gaussian	81,20	80,60 (Lee, 2000)	kNN	
Concentric	96,60	98,80 (Min e Cho, 2007)	Comitê de redes neurais com K-means	x
Spambase	85,66	88,70 (Wang e Witten, 2002)	Modelo logístico	x
Diabetes	76,83	78,60 (Zhou e Goldman, 2004)	Comitê de redes neurais	x
Zoo	95,09	95,94 (Holmes et al., 2002)	Várias árvores de decisão	x
Média	82,05	85,14		

trados na literatura. Cada linha da tabela corresponde a uma base de dados. A primeira coluna são os nomes das bases de dados; a segunda, os resultados alcançados pela RNPe; a terceira, resultados obtidos na literatura com a respectiva fonte; a quarta, a técnica que obteve o resultado na literatura; e, a quinta, indica com um "x" se o resultado na literatura foi obtido pelo procedimento de 10 *fold cross-validation*. Os resultados estão em porcentagem e correspondem a medida de acurácia. A última linha da tabela informa a acurácia média dos resultados da RNPe e da literatura sobre as bases de dados. O melhor resultado encontrado para a *CNAE-9* foi o obtido pela técnica VG-RAM WNN, mas ele não foi publicado. Uma breve descrição sobre cada um dos resultados encontrados na literatura está disponível na Seção 5.2.

Para as bases *CNAE-9*, *WebKB-4*, *Reuters-8* e *Gaussian*, não foram encontrados trabalhos que utilizaram tais bases com 10 *fold cross-validation*. Embora as bases originais da *WebKB-4* e da *Reuters-8* sejam populares, o pré-processamento realizado em (Cardoso-Cachopo, 2007), adicionado com a redução de atributos realizados neste trabalho, torna difícil realizar a comparação com resultados encontrados na literatura. De toda forma, esses resultados dão uma noção da dificuldade da tarefa. Além disso, entre os classificadores testados neste trabalho, o MLP obteve resultados mais elevados que os encontrados na literatura.

Para a base *Gaussian*, foi encontrado o resultado do kNN com a metodologia *leave one*

out. Essa metodologia é um caso especial do *k fold cross-validation*, onde o valor de *k* é igual ao número de amostras da base de dados. Para bases de dados com grande quantidade de amostras, como é o caso da base *Gaussian*, essa metodologia é inviável para alguns algoritmos devido ao seu alto custo computacional. Como esse método permite maximizar a quantidade de amostras de treinamento, é possível ocorrer aumento na acurácia do classificador (Witten et al., 2011), o que, de fato, aconteceu, já que o kNN aplicado para 10 *fold cross-validation* obteve resultado menor. Ainda assim, a RNPe alcançou acurácia semelhante ao resultado encontrado. Com relação à base *CNAE-9*, embora o resultado previamente conhecido não tenha sido alcançado de um 10 *fold cross-validation*, o VG-RAM WNN, algoritmo a obter o resultado apontado na tabela, foi aplicado nesse trabalho sobre a base *CNAE-9* usando 10 *fold cross-validation*.

A RNPe foi, em média, 3,09% inferior aos resultados obtidos na literatura, sendo as maiores diferenças entre os resultados para as bases *Car*, *CNAE-9*, *Yeast*, *Heart* e *Segmentation*. Com relação às bases *Car* e *Yeast*, é provável que a forma como os atributos não numéricos foram codificados tenham afetado no desempenho dos algoritmos sobre os mesmos, pois, ao contrário das outras bases, nenhum algoritmo avaliado obteve resultado próximo aos encontrados na literatura.

Contudo, mais da metade dos resultados indicados na Tabela 6.13 foram obtidos usando métodos mais complexos para classificação, tais como SVMs e comitês de classificadores, que normalmente demandam de quantidade razoável de tempo e de processamento. Em geral, os algoritmos da literatura buscaram somente maximizar a eficácia, sem considerar a eficiência dos algoritmos. Isto fica claro em (Zhang e Street, 2008), ao afirmar que só foram usadas bases de dados de pequenas dimensões nos testes para obter um tempo viável no uso de comitês de SVMs; e em (Holmes et al., 2002), no qual um número exaustivo de árvores de decisão foi utilizado, mas afirmando essa abordagem não ser computacionalmente prática para bases de dados com grande quantidade de classes (Holmes et al. definiram esse valor como 16 no trabalho deles). Por outro lado, o método proposto tenta não só obter uma qualidade razoável de resposta, mas também um algoritmo rápido, o que eventualmente pode resultar em abrir mão de uma quantidade de informação.

6.3.2 Conclusões

Dos resultados dos experimentos, pode-se concluir que a rede neural proposta alcançou desempenho próximo ao das melhores técnicas clássicas avaliadas, e obteve tamanho de estrutura e tempo de classificação tão reduzidos quanto os das técnicas que consomem menos recursos computacionais. Com isso, é mostrado que a RNPe consegue obter um com-

promisso razoável entre qualidade de resposta (desempenho) e complexidade da arquitetura (tamanho da arquitetura e tempo de classificação).

6.4 Experimento com comitê de RNPes e com aprendizado semi-supervisionado incremental

Nesta última etapa dos experimentos, é realizada uma análise do desempenho de um comitê de RNPes e de seu uso para aprendizado semi-supervisionado.

Três técnicas são comparadas nesta seção em relação ao desempenho: uma RNPe, um comitê de nove RNPes e um comitê de nove RNPes com aprendizado semi-supervisionado. Para estes experimentos, não foi realizada calibração das redes neurais, por considerar uma situação em que inicialmente são disponíveis poucos dados de treinamento, não sendo suficiente para obter uma calibração satisfatória das redes. Portanto, nessa condição, foi adotada a configuração dos parâmetros mais frequentemente encontrada durante a calibração da RNPe nos experimentos da Seção 6.3. A configuração usada para os parâmetros foi $n_{max} = 100$, $\varphi_{init} = 0,05$, $\upsilon = 0$, $\eta = 0$ e as variâncias foram atualizadas.

Para os experimentos desta seção, foram usadas as bases de dados da Tabela 5.1, com exceção das bases *WebKB-4* e *Reuters-8* devido ao tamanho das mesmas. Para o treinamento das redes, foram utilizados 10% das bases de dados, e os restantes 90% usados para teste. Para medir o desempenho, foi utilizada a métrica acurácia. Os experimentos foram realizados usando dois 10 *fold cross-validation*, e para cada base de dados e técnica foi calculada a média das 20 execuções.

As redes do comitê de nove RNPes foram treinadas seguindo os mesmos passos usados para treinar uma rede simples. A única diferença é que, para maximizar a quantidade de amostras de treinamento, que nestes experimentos é reduzida, todas as amostras são usadas para treino, porém cada rede é treinada com uma ordem diferente das amostras. Assim, é alcançada uma diversidade entre as redes, pois, a ordem das amostras afeta o aprendizado.

Para o aprendizado semi-supervisionado, foram usados os procedimentos descritos no Algoritmo 8, no qual as amostras rotuladas pelo comitê de redes neurais foram usadas para treinar as redes que discordavam do consenso do comitê de redes neurais. Para o limiar do Algoritmo 8, foi usado o valor 6, o que, para um comitê de nove redes, significa que mais da metade das redes têm de concordar com o rótulo dado a uma amostra. Se não houver um consenso de pelo menos seis redes para um rótulo de uma amostra, essa não é usada para treinamento.

Tabela 6.14: Acurácia média obtida para uma rede, um comitê de nove redes e um comitê de nove redes com aprendizado semi-supervisionado. Valores em porcentagem. O maior valor para cada base de dados está realçado em negrito. DPs significa desvios padrões.

Base de dados	1 RNPe	9 RNPes	9 RNPes-SS
Balance	75,73	**77,21**	73,20
Blood	53,54	**54,03**	53,30
Iris	93,04	**93,70**	**93,70**
Haberman	56,56	**58,25**	54,89
Car	77,77	**79,48**	77,69
CNAE-9	61,41	75,93	**77,39**
Wine	91,11	92,07	**92,20**
Yeast	35,13	**44,36**	42,51
Abalone	16,90	18,30	**18,41**
Heart	62,74	**65,91**	65,06
Sonar	54,59	59,32	**59,34**
Segmentation	63,89	**65,25**	62,77
Texture	89,14	**90,91**	89,85
Gaussian	55,42	**56,88**	50,52
Concentric	63,74	**67,60**	64,14
Spambase	74,71	74,89	**76,78**
Diabetes	70,75	**72,27**	70,78
Zoo	**69,18**	68,31	68,47
Média	64,74	**67,48**	66,17
Média dos DPs	5,84	**4,24**	4,70

Os resultados obtidos por cada técnica são mostrados na Tabela 6.14, onde cada linha são os resultados para uma base de dados. A primeira coluna são os nomes das bases de dados; a segunda, são os resultados referentes a uma rede neural (1 RNPe); a terceira, a um comitê de nove redes (9 RNPes); e, a quarta, a um comitê de nove redes com aprendizado semi-supervisionado (9 RNPes-SS). A penúltima linha da tabela informa a acurácia média obtida por cada técnica sobre as bases de dados, enquanto a última linha indica a média dos desvios padrões. Os valores informados são em porcentagem, e o melhor resultado para cada base de dados está realçado em negrito.

A Tabela 6.14 mostra que, para as 18 bases de dados usadas para avaliação, em 17 o comitê de nove redes alcançou resultados superiores aos de uma única rede. A única exceção aconteceu para a base *Zoo*. O que pode ter prejudicado o desempenho sobre essa base de dados é a pouca quantidade de amostras de treinamento, aproximadamente 10, e o número de sete classes, relativamente alto para a quantidade de amostras de treinamento. Com porção tão limitada de amostras, a diversidade do comitê foi baixa e isso prejudicou o desempenho do mesmo.

Os maiores ganhos obtidos com o comitê foram para as bases *CNAE-9* e *Yeast*, nas quais foram obtidos ganhos de aproximadamente 10%. Além de permitir maior diversidade das respostas, o uso de comitês de RNPes deve ter reduzido o efeito da ordem das amostras sobre

a rede. Foi buscado encontrar alguma relação dos ganhos de desempenho com as medidas de complexidade das bases de dados usando o teste de Kendall, no entanto, não foi encontrado nenhum relacionamento significante entre as variáveis.

O aprendizado semi-supervisionado aplicado ao comitê de redes neurais aumentou ligeiramente o resultado da acurácia para seis bases de dados, onde os maiores aumentos foram para as bases *CNAE-9* e *Spambase*, sendo obtidos ganhos na acurácia de aproximadamente 2%. Porém, reduziu a qualidade dos resultados para 11 bases de dados, sendo a redução maior para quatro bases de dados: *Balance*, *Haberman*, *Concentric* e *Gaussian*.

No teste de Kendall, aplicado sobre a diferença entre os resultados dos comitês com e sem aprendizado semi-supervisionado, foi verificado que o método de aprendizado semi-supervisionado tendeu a apresentar perdas de desempenho maiores para bases de dados com valores baixos de F1 e altos de F2. Ou seja, o método apresentou resultados melhores para bases de dados cujo poder discriminativo é alto e com pequena sobreposição de valores de atributos.

Analisando melhor o comitê de redes neurais, foi observado que a redução do desempenho com o aprendizado semi-supervisionado ocorreu também porque muitas amostras classificadas erradas pelo comitê foram usadas para treinamento, o que prejudicou o desempenho do modelo. Quando foram usadas as classes corretas das amostras no aprendizado semi-supervisionado, a acurácia média foi ligeiramente maior do que o do comitê de nove redes, embora o desempenho tenha sido reduzido para seis bases de dados.

Os desvios padrões da Tabela 6.14 indicam que os dois comitês de nove redes neurais apresentaram resultados mais constantes do que uma única rede neural. Essa observação, aliada à acurácia média dos comitês de redes, sugerem que o uso de comitês de técnicas pode ser mais vantajoso para tratar um problema do que uma única técnica isolada.

A Figura 6.20 mostra os resultados dos testes estatísticos de Friedman e de Nemenyi sobre a Tabela 6.14. A hipótese nula de que os resultados dos algoritmos são equivalentes foi descartada no teste de Friedman. A Figura 6.20 mostra que o comitê de nove redes neurais (9 RNPes) foi superior em relação a uma única rede (1 RNPe), e não houve diferenças estatísticas entre as outras técnicas.

Apesar ter sido obtido poucas bases de dados com ganho quando aplicada a técnica de aprendizado semi-supervisionado, e esses terem sido reduzidos, o nível dos ganhos de desempenho foi semelhante aos apresentados em (Zhang e Rudnicky, 2006), que utilizou somente 10% das amostras para treinamento e, para a maioria das bases de dados, não foram obtidos ganhos de desempenho superiores a 2% com o aprendizado semi-supervisionado. Na abordagem de Zhang e Rudnicky (2006), o conjunto de amostras não rotuladas é dividido em

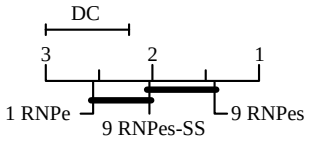


Figura 6.20: Comparação entre as técnicas de classificação usando o teste de Nemenyi. Quanto menor a posição na escala de *rank*, melhor. Classificadores não significativamente diferentes entre si são conectados.

blocos rotulados por um classificador, e, para cada um destes, é usada uma função objetivo para verificar se o desempenho do classificador aumenta com o uso desse bloco. Em caso afirmativo, esse bloco é adicionado ao conjunto de treinamento do classificador, e um novo classificador é treinado.

A abordagem de Zhang e Rudnicky (2006) difere da usada nessa seção, pois, enquanto em (Zhang e Rudnicky, 2006) as amostras são todas rotuladas para depois verificar quais conjuntos de amostras devem ser usados para treinamento, o método usado aqui utiliza as amostras para treinamento à medida que elas são rotuladas e passam no critério que define se devem ser usadas para treinamento ou não.

6.5 Conclusão

Neste capítulo, foi realizado uma série de experimentos para avaliar empiricamente o comportamento da rede neural proposta RNPe. Algumas das conclusões dos experimentos realizados foram que os procedimentos usados para controlar a arquitetura da rede evitaram a mesma de ter uma arquitetura muito grande. E, em mais da metade das bases de dados usadas para avaliação, a arquitetura da RNPe foi a mínima possível, correspondendo a usar uma função similar a Gaussiana para representar cada classe. Como, para maior parte dessas bases de dados, foi obtido desempenho satisfatório, é possível concluir a possibilidade de representar cada classe de um problema por uma Gaussiana para reduzir o custo computacional, ao preço de uma pequena perda de informação. Também foi observada que a modificação proposta na função Gaussiana mostrou ser efetiva, pois, com ela, a rede obteve resultados mais estáveis.

Além disso, o modelo proposto atende a vários requisitos desejados para uma técnica incremental, como ser capaz de acomodar novas classes introduzidas por novos dados, não necessitar de conhecimento prévio sobre a sua estrutura, não necessitar de inicializar sua estrutura para a tarefa de aprendizado, e ser capaz de realizar separação não linear entre classes. A RNPe também atende aos critérios definidos por Langley (1995), por usar uma

amostra de treinamento por vez, não reprocessar nenhuma amostra e por estocar somente uma estrutura de conhecimento na memória, tal que não é possível retornar a um estado anterior de conhecimento.

Observou-se nos experimentos que a capacidade da RNPe para aprender novas informações é maior que sua capacidade para reter informação antiga, e seu desempenho sobre as amostras usadas para treinamento tem a tendência a aumentar quanto mais dados de treinamento são usados. Isto acontece porque os parâmetros do modelo são melhores ajustados. Também foi notado que o desempenho da RNPe está relacionado ao poder discriminativo dos atributos e os contornos de separação entre classes; quanto maior o poder discriminativo dos atributos de um problema e menor a não linearidade da superfície de separação entre classes, maior tende a ser o desempenho da RNPe.

Experimentos realizados com várias outras técnicas mostraram, estatisticamente, que a RNPe obteve desempenho mais elevado do que das técnicas com baixa complexidade (baixo consumo de recursos computacionais), e sua estrutura foi estatisticamente menos complexa que os algoritmos a obter os melhores resultados. Além disso, não foram observadas diferenças estatísticas entre as complexidades de estrutura da RNPe e as técnicas menos complexas, e nem diferenças de desempenho entre a RNPe e as técnicas com os melhores resultados. Essa análise mostra o método proposto alcançando compromisso satisfatório entre desempenho (eficácia) e complexidade de arquitetura (eficiência).

Finalmente, uma breve análise sobre o uso de comitês de RNPes e aprendizado semi-supervisionado foi realizada. O comitê de RNPes alcançou valores de acurácia maiores que uma simples RNPe em quase todas as bases de dados, mesmo usando quantidade reduzida de dados (10%) e sem calibrar devidamente cada rede. O método de aprendizado semi-supervisionado mostrou maiores ganhos para duas bases de dados, porém não atuou muito bem para outros casos. Novas abordagens devem ser exploradas para melhorar a qualidade do método.

Capítulo 7

Conclusão

Técnicas tradicionais de aprendizado de máquina, como *support vector machines* e *multi-layer perceptrons*, têm focado sobre tarefas nas quais é frequente estar implícita a suposição de haver um conjunto de treinamento representativo de uma tarefa, e a fase de treinamento cessa uma vez o conjunto ter sido processado. No entanto, dados suficientemente representativos de uma tarefa nem sempre são disponíveis, e a sua aquisição pode ser onerosa, lenta e obtidos somente em pequenas quantidades por vez. Nesse contexto, surgem os algoritmos de aprendizado incremental. Eles podem continuar aprendendo sempre que novos dados são adquiridos. Entretanto, mesmo que os dados disponíveis tenham sidos exauridos, os novos dados de entrada continuam sendo aprendidos por esses algoritmos.

Nesse trabalho foi proposto um modelo de aprendizado incremental supervisionado para tarefas de classificação capaz de alcançar um compromisso satisfatório entre eficácia e eficiência. Ele é chamado de Rede Neural Probabilística evolutiva (RNPe), é baseado na Rede Neural Probabilística, no Modelo de Mistura de Gaussianas e no algoritmo de *Expectation Maximization*. As principais características do modelo proposto são: novas informações adicionadas através de uma amostra por vez, ao invés de por lote; não ser necessário reprocessar uma amostra de treinamento; e poder aprender continuamente durante toda a sua existência.

A arquitetura da RNPe possui uma estrutura flexível, sendo fácil inserir ou remover classes, e pode crescer ou diminuir para melhor se adaptar a um problema. Para evitar um crescimento exagerado da sua arquitetura, essa é continuamente controlada para permanecer reduzida e igualmente estável, diferentemente de muitos modelos incrementais, cuja estrutura tende a crescer significativamente ao longo do tempo. Dentre os procedimentos propostos, foram incluídos métodos para controlar o tamanho da rede e, também, uma modificação na função de distribuição Gaussiana, pois foi verificado que, para certas condições, essa pode ser muito afetada pela imprecisão dos parâmetros quando esses são estimados incrementalmente.

Para avaliação da RNPe, foi realizada uma série de experimentos para verificar a eficácia dos procedimentos propostos e comparar os resultados obtidos pela RNPe com os de outras técnicas incrementais e populares na literatura. Os experimentos foram realizados sobre bases de dados de domínio público de diferentes características, escolhidas para verificar o comportamento do método proposto sobre diferentes condições.

Na primeira seção dos experimentos, foram avaliados os procedimentos propostos para a RNPe. Algumas das observações sobre os métodos propostos para controlar o tamanho da sua arquitetura e inserir novas informações ao modelo mostraram terem sido estes satisfatórios, pois, de 20 bases usadas na avaliação, 11 foram representadas pela menor arquitetura possível para a rede proposta. Na média, a arquitetura da RNPe foi mais de 200 vezes menor que a estrutura da técnica de vizinho mais próximo, ou 1-NN. Com relação à eficácia da RNPe, esta foi, na média, ligeiramente superior. Também foi verificado o efeito da função Gaussiana no aprendizado incremental, tendo sido observado que, quando a rede neural usou essa função, a acurácia da mesma foi significativamente mais reduzida do que o modelo que utilizava a modificação proposta.

Duas metodologias foram usadas na segunda seção dos experimentos para estimar os graus de estabilidade e plasticidade do modelo proposto e de algumas redes neurais incrementais existentes na literatura e, portanto, comparadas aqui. Na primeira, foram utilizadas as métricas propostas neste trabalho para avaliar a estabilidade e plasticidade dos algoritmos. De forma geral, observou-se a RNPe apresentar uma maior capacidade para aprender (plasticidade) do que sua capacidade de reter informação (estabilidade). Ambas as características foram semelhantes nas redes neurais cujo aprendizado é influenciado pela ordem das amostras.

Os modelos cujo aprendizado é menos influenciado pela ordem das amostras conseguiram reter mais informação, mas tiveram uma habilidade menor para aprender informação nova. Isto sugere que a RNPe tem maior capacidade de se adaptar a mudanças nas características de um problema do que os métodos não influenciados pela ordem das amostras. Tendo como foco apenas o algoritmo RNPe, análises sobre as bases de dados e testes estatísticos indicaram que a RNPe apresenta melhor compromisso entre estabilidade e plasticidade para bases de dados com contornos de classe menos complexos e atributos com poder discriminativo elevado. Testes semelhantes foram aplicados sobre os outros algoritmos individualmente e as conclusões foram ligeiramente semelhantes.

A segunda metodologia aplicada foi a de Polikar et al. (2001). Nesse procedimento, foi observado que, à medida que mais dados de treinamento são usados, a acurácia da RNPe tende a aumentar tanto sobre os dados de treinamento como sobre os de teste. O modelo proposto também alcançou acurácia ligeiramente maior do que as outras técnicas para a mai-

oria das bases de dados. O aumento da complexidade dos algoritmos também foi verificado nesse experimento. As técnicas baseadas em estocar amostras na estrutura tiveram aumento crescente no tamanho da estrutura e no tempo de classificação. Já as técnicas que buscaram modelar a distribuição dos dados tiveram estrutura reduzida e precisaram de um tempo menor para classificação. Neste sentido, a RNPe obteve para a maioria das bases de dados a menor estrutura e tempo de classificação.

Na terceira seção dos experimentos, a RNPe foi comparada com várias técnicas de aprendizado, e testes estatísticos indicaram que sua acurácia foi superior às técnicas avaliadas que possuíam as menores estruturas e tempos de classificação. Os testes também sugeriram que a RNPe foi estatisticamente menor e mais rápida do que as técnicas que obtiveram os melhores resultados para a medida de acurácia. Não houve diferenças estatísticas em relação ao tamanho e tempo de classificação entre a RNPe e as técnicas de menor complexidade, como também não houve diferenças estatísticas em relação à acurácia entre a RNPe e as técnicas que apresentaram os melhores resultados para essa métrica. A acurácia média da RNPe foi superior às das outras técnicas e o tamanho médio de sua estrutura foi aproximadamente três vezes maior do que a de Rocchio, a técnica com a menor estrutura avaliada. A terceira menor técnica obteve uma estrutura maior que quatro vezes à de Rocchio.

Os resultados obtidos nos experimentos indicam o modelo proposto alcançando um compromisso satisfatório entre eficiência e eficácia, pois, para a maioria das bases de dados, obteve-se uma estrutura tão reduzida quanto às das técnicas de menor complexidade, e obteve-se para as bases de dados acurácia em nível semelhante aos dos melhores algoritmos avaliados. Além disso, das 20 bases de dados avaliadas, a RNPe apresentou a menor representação para 11 delas, que consiste em representar cada classe por uma função semelhante à da Gaussiana. Com isso, a hipótese de poder representar um modelo incremental através de um modelo reduzido se torna plausível.

Possíveis continuações desse trabalho incluem uma simplificação do modelo apresentado, necessitando de menos parâmetros para calibrar e procedimentos mais simples. E também:

- estudar formas para aprimorar a qualidade da rede, sendo alguns caminhos apontados neste trabalho, como o uso de comitês de redes neurais;
- estudar outras formas de realizar aprendizado semi-supervisionado para obter um método com melhor qualidade;
- estudar possíveis formas de aumentar o grau de estabilidade do método proposto sem que ocorra um grande aumento em sua arquitetura; e,

- por fim, paralelizar a arquitetura da rede, pois o mesmo é viável, aumentando a velocidade de classificação da rede.

Referências Bibliográficas

Abraham, A. e Nath, B. (2000). Hybrid intelligent systems: A review of a decade of research. Relatório técnico, School of Computing and Information Technology, Faculty of Information Technology , Monash University, Australia.

Abraham, A. e Nath, B. (2001). A neuro-fuzzy approach for modelling electricity demand in Victoria. *Applied Soft Computing*, 1:127–138.

Allwein, E. L., Schapire, R. E., e Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, páginas 113–141.

Alpaydin, E. (1991). GAL: Networks that grow when they learn and shrink when they forget. *International Journal of Pattern Recognition and Artificial Intelligence*, 8:391–414.

Arlot, S. e Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, páginas 40–79.

Baluja, S. (1998). Probabilistic modeling for face orientation discrimination: learning from labeled and unlabeled data. *Neural Information Processing Systems*, páginas 1–7.

Barndorff-Nielsen, O. E., Jensen, J. L., e Kendall, W. S. (1993). *Networks and Chaos - Statistical and Probabilistic Aspects*. Chapman and Hall, London.

Bevington, P. R. e Robinson, D. K. (2003). *Data Reduction and Error Analysis for the Physical Sciences*. Mc Graw Hill, 3 edição.

Bhatia, N. e Vandana (2010). Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8(2):302–305.

Bhattacharya, M., Abraham, A., e Nath, B. (2002). A linear genetic programming approach for modeling electricity demand prediction in victoria. *Hybrid Information Systems, Advances in Soft Computing*, páginas 379–394.

Bhattacharyya, N., Metla, A., Bandyopadhyay, R., Tudu, B., e Jana, A. (2008). Incremental PNN classifier for a versatile electronic nose. *Third International Conference on Sensing Technology*, páginas 242–247.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blum, A. e Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the Workshop on Computational Learning Theory*, páginas 1–10.

Box, G. E. P. (1953). Non-normality and tests on variances. *Biometrika*, 40(3/4):318–335.

Brodatz, P. (1966). *Textures: A Photographic Album for Artists and Designers*. Dover Publications, New York.

Cardoso-Cachopo, A. (2007). Datasets for single-label text categorization. http://web.ist.utl.pt/ acardoso/datasets/.

Castelli, V. e Cover, T. M. (1995). On the exponential value of labeled samples. *Pattern Recognition*, 16:105–111.

Castelli, V. e Cover, T. M. (1996). The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42:2101–2117.

CBO (2007). Classificação brasileira de ocupações - CBO. Relatório técnico, Ministério do Trabalho e Emprego - MTE, http://www.mtecbo.gov.br/.

Chatterji, G. P. B. N. (1990). A class of new KNN methods for low sample problems. *IEEE – Transactions on Systems Man and Cybernetics*, 20(3):715 – 718.

Church, K. W. e Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22 – 29.

Ciarelli, P. M., Oliveira, E., Badue, C., e Souza, A. F. D. (2009a). Multi-label text categorization using a probabilistic neural network. *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, 1:133–144.

Ciarelli, P. M., Oliveira, E., e Salles, E. O. T. (2009b). Neural network based in expectation maximization for on-line text categorization. *IX Simpósio Brasileiro de Automação Inteligente (SBAI)*, páginas 1–6.

Ciarelli, P. M., Oliveira, E., e Salles, E. O. T. (2012). An incremental neural network with a reduced architecture. *Neural Networks*, 35:70–81.

Ciarelli, P. M., Oliveira, E., e Salles, E. O. T. (2013). Impact of the characteristics of data sets on incremental learning. *Engineering Applications of Artificial Intelligence*. No prelo.

CNAE (2012). Classificação nacional de atividades econômicas - CNAE 2.0. Relatório técnico, Instituto Brasileiro de Geografia e Estatística (IBGE), http://www.ibge.gov.br/concla.

Coghill, G., Zhang, D., Ghobakhlou, A., e Kasabov, N. (2003). Connectionist systems for rapid adaptive learning: A comparative analysis on speech recognition. *Proc. of the 7th Joint Conference on Information Sciences*, páginas 1365–1368.

Corduneanu, A. e Jaakkola, T. (2001). Stable mixing of complete and incomplete information. Relatório técnico, MIT Artificial Intelligence Laboratory.

Dempster, A. P., Laird, N. M., e Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Demuth, H. e Beale, M. (2002). *Neural Network Toolbox for Use with MATLAB: User's Guide*. The MathWorks, 4 edição.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Deng, D. e Kasabov, N. (2000). ESOM: An algorithm to evolve self-organizing maps from on-line data streams. *Int. Joint Conf. Neural Netw.*, 6:3–8.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, páginas 1 – 15.

Duda, R. O., Hart, P. E., e Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience, New York, 2 edição.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Elwell, R. e Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531.

Fabri, J. A. e Rissoli, V. R. V. (2000). Desenvolvimento de um sistema especialista fuzzy aplicado a domínios genéricos do conhecimento. Relatório técnico, Universidade Regional de Blumenau.

Fan, J., Dimitrova, N., e Philomin, V. (2004). Online face recognition system for videos based on modified probabilistic neural networks. *International Conference on Image Processing*, 3:2019 – 2022.

Frank, A. e Asuncion, A. (2010). UCI machine learning repository.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701.

Fujino, A., Ueda, N., e Saito, K. (2005). A hybrid generative/discriminative approach to semi-supervised classifier design. *20th National Conference on Artificial Intelligence*, páginas 764–769.

Fung, C. C., Iyer, V., Brown, W., e Wong, K. W. (2005). Comparing the performance of different neural networks architectures for the prediction of mineral prospectivity. *IEEE Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, páginas 394 – 398.

Furman, W. D. e Lindsay, B. G. (1994). Testing for the number of components in a mixture of normal distributions using moment estimators. *Computational Statistics & Data Analysis*, páginas 473–492.

Gavoyiannis, A. E., Voumvoulakis, E. M., e Hatziargyriou, N. D. (2005). On-line supervised learning for dynamic security classification using probabilistic neural networks. *Power Engineering Society General Meeting*, 3:2669–2675.

Georgiou, V., Pavlidis, N. G., Parsopoulos, K. E., Alevizos, e Vrahatis, M. N. (2004). Optimizing the performance of probabilistic neural networks in a bionformatics task. Relatório técnico, University of Patras, Greece.

Ghobakhlou, A. e Kasabov, N. (2003). A methodology for adaptive speech recognition systems and a development environment. *Proc. Artif. Neural Netw. Neural Inform . Process.*, páginas 316–319.

Ghobakhlou, A. e Seesink, R. (2001). An interactive multi modal system for mobile robotic control. *5th Biannu. Conf. Artf. Neural Netw.*, páginas 93–99.

Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications*, páginas 215–223.

Gleick, J. (1987). *Chaos: making a new science*. Penguin Books, New York.

Goldman, S. e Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proc. 17th International Conf. on Machine Learning*, páginas 327–334.

Gomm, J. B. e Williams, D. (1995). An adaptive neural network for on-line learning and diagnosis of process faults. *IEE Colloquium on Qualitative and Quantitative Modeling Methods for Fault Diagnosis*, páginas 9/1 – 9/5.

Gould, W., Pitblado, J., e Sribney, W. (2006). *Maximum Likelihood Estimation with Stata*. Stata Press Publication, 3 edição.

Hansen, L. e Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

Haykin, S. (2005). *Neural Networks - A Comprehensive Foundation*. Pearson Prentice Hall, 9 edição.

Heinen, M. R. e Engel, P. M. (2010). IPNN: An incremental probabilistic neural network for function approximation and regression tasks. *Brazilian Symposium on Neural Networks*, 11:25–30.

Ho, T. K. (2000). Complexity of classification problems and comparative advantages of combined classifiers. *First International Workshop on Multiple Classifier Systems*, páginas 97–106.

Ho, T. K. e Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:289–300.

Hoauiss (2010). Dicionário Houaiss da língua portuguesa.

Holmes, G., Pfahringer, B., Kirkby, R., Frank, E., e Hall, M. (2002). Multiclass alternating decision trees. *Proceedings of the 13th European Conference on Machine Learning*, páginas 161–172.

Huang, C.-J. e Liao, W.-C. (2003). A comparative study of feature selection methods for probabilistic neural networks in cancer classification. *IEEE International Conference on Tools with Artificial Intelligence*, páginas 451 – 458.

Illingworth, F. R., Callaghan, V., e Hagras, H. (2005). A neural network agent based approach to activity detection in AmI environments. *IEEE International Workshop on Intelligent Environments*, páginas 92–99.

Iman, R. L. e Davenport, J. M. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, páginas 571–595.

Jatmiko, W., Fukuda, T., Sekiyama, K., e Kusumoputro, B. (2005). Optimized probabilistic neural networks in recognizing fragrance mixtures using higher number of sensors. *IEEE Sensors*, página 4.

Jiang, Y. e Zhou, Z.-H. (2004). Editing training data for knn classifiers with neural network ensemble. *Lecture Notes in Computer Science*, 3173:356–361.

Jimenez, L. e Landgrebe, D. (1998). Supervised classification in high dimensional space: Geometrical, statistical and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Reviews and Applications*, 28(1):39–54.

Jordan, M. I. (1986). Serial order: a parallel distributed processing approach. Relatório técnico, University of California, San Diego.

Kalatzis, I., Piliouras, N., Ventouras, E., Papageorgiou, C. C., Rabavilas, A. D., e Cavouras, D. (2003). Comparative evaluation of probabilistic neural network versus support vector machines classifiers in discriminating ERP signals of depressive patients from healthy controls. *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, 2:981 – 985.

Kan, W. e Aleksander, I. (1989). A probabilistic logic neuron network for associative learning. *Neural computing architectures: the design of brain-like machines*, páginas 156–171.

Kasabov, N. (1998). Evolving fuzzy neural networks - algorithms, applications and biological motivation. *Methodologies for the conception, design and application of soft computing*, páginas 271–274.

Kasabov, N. (2003). *Evolving Connectionist Systems*. Springer Verlag, 1 edição.

Kasabov, N. (2007). *Connectionist Systems - The Knowledge Engineering Approach*. Springer-Verlag London, 2 edição.

Kasabov, N. e Iliev, G. (2000). A methodology and a system for adaptive speech recognition in a noisy environment based on adaptive noise cancellation and evolving fuzzy neural networks. *Neuro-Fuzzy Pattern Recognition*, páginas 179–203.

Kasabov, N. e Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference systems. *IEEE Transactions Fuzzy Syst.*, 10(2):144–154.

Kasabov, N. e Watts, M. J. (1999). Spatial-temporal adaptation in evolving fuzzy neural networks for on-line adaptive phoneme recognition. Relatório técnico, University of Otago, Dunekin, Nova Zelândia.

Kemp, G. (2002). Estimation and inference in econometrics - notes on maximum likelihood estimation. Relatório técnico, University of Essex.

Kendall, M. (1938). A new measure of rank correlation. *Biometrika*, 30:81–93.

Kobos, M. e Mandziuk, J. (2012). Bandwidth selection in kernel density estimators for multiple-resolution classification. *Lecture Notes in Artificial Intelligence*, páginas 378–386.

Komati, K. S. e Souza, A. F. D. (2002). Vergence control in a binocular vision system using weightless neural networks. *Proceedings of the 4th International Symposium on Robotics and Automation*, páginas 1–8.

Kühne, M., Pullella, D., Togneri, R., e Nordholm, S. (2008). Towards the use of full covariance models for missing data speaker recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing*, páginas 4537–4540.

Kuncheva, L. I. e Whitaker, C. J. (2000). Measures of diversity in classifier ensembles. *Machine Learning*, 51(2):181 – 207.

Kvam, P. H. e Vidakovic, B. (2007). *Nonparametric Statistics with Applications to Science and Engineering*. Wiley-Interscience.

Langley, P. (1995). Order effects in incremental learning. Em Reimann, P. e Spada, H., editors, *Learning in humans and machines: Towards an interdisciplinary learning science*, páginas 154–167. Elsevier.

Lee, C. e Landgrebe, D. A. (1993). Analyzing high dimensional multispectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 31(4):792–800.

Lee, J. A. (2000). The Elena project. Relatório técnico, Neural Network Group - Université Catholique de Louvain, http://www.dice.ucl.ac.be/neural-nets/Research/Projects/ELENA/elena.htm.

Leite, D. F., Jr., P. C., e Gomide, F. (2009). Sistemas conexionistas evolutivos. *IX Simpósio Brasileiro de Automação Inteligente (SBAI)*, páginas 1–6.

Leon-Garcia, A. (1993). *Probability and Random Processes for Electrical Engineering*. Addison Wesley Longman, 2 edição.

Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168.

Li, M. e Sethi, I. K. (2006). Confidence-based active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1251–1261.

Liu, F., Ng, G. S., Quek, C., e Loh, T. F. (2006). Artificial ventilation modeling using neuro-fuzzy hybrid system. *Int. Joint Conf. Neural Netw.*, páginas 5166–5171.

Ludermir, T. B., de Carvalho, A., Braga, A. P., e Souto, M. C. P. (1999). Weightless neural models: A review of current and past works. *Neural Computing Surveys*, 2:41–61.

Lughofer, E. (2011). Dynamic evolving cluster models using on-line split-and-merge operations. *International Conference on Machine Learning and Applications*, 10:20–26.

Magdon-Ismail, M. e Purnell, J. T. (2010). Approximating the covariance matrix of GMMs with low-rank perturbations. *International Journal of Data Mining and Bioinformatics*, 3(1):1–15.

Manning, C. D., Raghavan, P., e Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Mansilla, E. B. e Ho, T. K. (2004). On classifier domains of competence. *Proceedings of the 17th International Conference on Pattern Recognition*, 1:136–139.

Mao, K. Z., Tan, K. C., e Ser, W. (2000). Probabilistic neural-network structure determination for pattern classification. *IEEE Transactions on Neural Networks*, 11:1009 – 1016.

Markowski, C. A. e Markowski, E. P. (1990). Conditions for the effectiveness of a preliminary test of variance. *The American Statistician*, 44(4):322–326.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441.

McCallum, A. e Nigam, K. (1998). A comparison of event models for naive bayes text classification. *Workshop on Learning for Text Categorization*, páginas 1–8.

McLachlan, G. J. (1987). On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Applied Statistics*, 36(3):318–324.

McLachlan, G. J. e Krishnan, T. (1997). *The EM Algorithm and Extensions*. John Wiley and Sons, New York.

Mendel, J. M. (1995). Fuzzy logic systems for engineering: A tutorial. *IEEE Proc.*, 83:345–377.

Michie, D., Spiegelhalter, D., e Taylor, C. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood.

Min, J.-K. e Cho, S.-B. (2007). Multiple classifier fusion using k-nearest localized templates. *Proceedings of the 8th international conference on Intelligent data engineering and automated learning*, páginas 447–456.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533.

Mollineda, R. A., Sánchez, J. S., e Sotoca, J. M. (2005). Data characterization for effective prototype selection. *Proc. of the 2nd Iberian Conf. on Pattern Recognition and Image Analysis*, páginas 27–34.

Murali, T., Sriskanthan, N., e Ng, G. S. (2003). Comparative analysis of the two fuzzy neural systems ANFIS and EFuNN for the classification of handwritten digits. *7th IEEE Int. Symp. Consumer Electron.*

Muslea, I., Minton, S., e Knoblock, C. A. (2000). Selective sampling with redundant views. *Proceedings of the national conference on artificial intelligence*, páginas 621–626.

Nascimento, D. S. C. e Coelho, A. L. V. (2009). Bagging heterogêneo evolutivo: Caracterização e análise comparativa com ensembles homogêneas de redes neurais RBF. *Anais do IX Simpósio Brasileiro de Automação Inteligente*, página 6.

Nemenyi, P. (1963). *Distribution-free Multiple Comparisons*. Tese de Doutorado, Princeton University.

Ng, G. S., Erdogan, S., Shi, D., e Wahab, A. (2006). Insight of fuzzy neural systems in the application of handwritten digits classification. *Int. J. Image Graph.*, páginas 511–532.

Nigam, K. (2001). *Using unlabeled data to improve text classification*. Tese de Doutorado, Carnegie Mellon University.

Nigam, K., Mccallum, A. K., Thrun, S., e Mitchell, T. (1999). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, páginas 1–34.

Nikolaev, N. Y. (2007). Probabilistic neural networks. Relatório técnico, Goldsmiths College, United Kingdom.

Noether, G. E. (2008). Why kendall tau? Relatório técnico, University of Connecticut.

Oliveira, E., Ciarelli, P. M., e Salles, E. O. T. (2012). *An Evolving System in the Text Classification Problem*, volume 1, chapter 18, páginas 467–486. IGI Global.

Orriols-Puig, A., Macià, N., e Ho, T. K. (2010). Documentation for the data complexity library in c++. Relatório técnico, Universitat Ramon Llull - Barcelona, Spain.

Papoulis, A. (1991). *Probability, random variables, and stochastic processes*. Mc Graw Hill, 3 edição.

Parzen, E. (1962). On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 3:1065 – 1076.

Patra, P. K., Nayak, M., Nayak, S. K., e Gobbak, N. K. (2002). Probabilistic neural network for pattern classification. *IEEE Proceedings of the 2002 International Joint Conference on Neural Networks*, 2:1200 – 1205.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225.

Polikar, R., Udpa, L., Udpa, S. S., e Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural network. *IEEE Transactions on Systems, Man, and Cybernetics. C: Applications and Reviews*, 31(4):497–508.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., e Flannery, B. P. (2007). *Numerical Recipes - The Art of Scientific Computing*. Cambridge University Press, 3 edição.

Ratsaby, J. e Venkatesh, S. S. (1995). Learning from a mixture of labeled and unlabeled examples with parametric side information. *Proceedings of the 8th Annual Conference on Computational Learning Theory*, páginas 412–417.

Raudys, S. J. e Jain, A. K. (1991). Small sample size effects in statistical pattern recognition: recmmendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264.

Reynolds, D. A. e Rose, R. C. (1995). Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83.

Shiotani, S., Fukuda, T., e Shibata, T. (1995). A neural network architecture for incremental learning. *Neurocomputing*, 9:111–130.

Song, Q. e Kasabov, N. (2001). ECM, a novel on-line, evolving clustering method and its applications. *5th Biannu. Conf. Artf. Neural Netw. Expert Syst*, páginas 87–92.

Sorwar, G., Abraham, A., e Dooley, L. (2001). Texture classification based on DCT and soft computing. *10th IEEE Int. Conf. Fuzzy Syst.*, 3:545–548.

Sotoca, J. M., Sánchez, J. S., e Mollineda, R. A. (2005). A review of data complexity measures and their applicability to pattern classifcation problems. *Actas del III Taller Nacional de Minería de Datos y Aprendizaje*, páginas 77–83.

Souza, A. F. D., Badue, C., Pedroni, F., Oliveira, E., Dias, S. S., Oliveira, H., e de Souza, S. F. (2008). Face recognition with VG-RAM weightless neural networks. *Lecture Notes in Computer Science*, 5163:951–960.

Souza, A. F. D., Pedroni, F., Oliveira, E., Ciarelli, P. M., Henrique, W. F., Veronese, L., e Badue, C. (2009). Automated multi-label text categorization with VG-RAM weightless neural networks. *Neurocomputing*, 72:2209–2217.

Specht, D. (1990). Probabilistic neural networks. *Neural Networks*, 3(1):109 – 118.

Specht, D. F. (1988). Probabilistic neural networks for classification, mapping, or associative memory. *IEEE International Conference on Neural Networks*, 1(24):525 – 532.

Tan, P. J. e Dowe, D. L. (2002). Mml inference of decision graphs with multi-way joins and dynamic attributes. *Proceedings of the 15th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, páginas 131–142.

Therrien, C. W. (1992). *Discrete Random Signals and Statistical Signal Processing*. Prentice-Hall.

Vlassis, N. e Likas, A. (1999). A kurtosis-based dynamic approach to gausian mixture modeling. *IEEE Transactions on Systems, Man, and Cybernetics. A: Systems and Humans*, 29(4):393 – 399.

Vlassis, N. A., Papakonstantinou, G., e Tsanakas, P. (1999). Mixture density estimation based on maximum likelihood and sequential test statistics. *Neural Processing Letters*, 9:63 – 76.

Wang, Y. e Witten, I. H. (2002). Modeling for optimal probability prediction. *International Conference on Machine Learning*, páginas 650–657.

Watts, M. J. (2004). *Evolving connectionist systems: Characterisation, simplification, formalisation, explanation and optimisation*. Tese de Doutorado, University of Otago, Dunedin, New Zealand.

Watts, M. J. (2009). A decade of Kasabov's evolving connectionist systems: a review. *IEEE Transactions on Systems, Man, and Cybernetics. C: Applications and Reviews*, 39(3):253–269.

Watts, M. J. e Kasabov, N. (2000). Simple evolving connectionist systems and experiments on isolated phoneme recognition. *first IEEE Conf. Evol. Comput. Neural Netw.*, páginas 232–239.

Watts, M. J. e Worner, S. (2007). Comparison of multi-layer perceptrons and simple evolving connectionist systems over the Lincoln Aphid data set. Relatório técnico, Lincoln University, Nova Zelândia.

Welling, M. (2005). Robust higher order statistics. *International Conference on Artificial Intelligence and Statistics*, página 8.

Widmer, G. e Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101.

Witten, I. H., Frank, E., e Hall, M. A. (2011). *Data Mining. Practical machine learning tools and techniques*. Elsevier, 3 edição.

Woodford, B. J. (2001). Comparative analysis of the EFuNN and the Support Vector Machine models for the classification of horticulture data. *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems*, páginas 70–75.

Woodford, B. J. (2008). Evolving neurocomputing systems for horticulture applications. *Applied Soft Computing*, 8:564–578.

Woodford, B. J., Deng, D., e Benwell, G. L. (2004). A wavelet-based neuro-fuzzy system for data mining small image sets. *Australasian Workshop on Data Mining and Web Intelligence*, páginas 139–144.

Wu, Y., Tian, Q., e Huang, T. S. (2000). Discriminant-EM algorithm with application to image retrieval. *Proceedings of IEEE conference on Computer Vision and Pattern Recognition*, 1:222–227.

Yang, M.-S., Lai, C.-Y., e Lin, C.-Y. (2012). A robust EM clustering algorithm for Gaussian mixture models. *Pattern Recognition*, 45(11):3950–3961.

Yau, C. (2012). *R tutorial with bayesian statitics using openBUGS*.

Zhang, D., Ghobakhlou, A., e Kasabov, N. (2004). An adaptive model of person identification combining speech and image information. *8th International Conference on Control, Automation, Robotics and Vision*, páginas 413–418.

Zhang, R. e Rudnicky, A. I. (2006). A new data selection principle for semi-supervised incremental learning. *Proceedings of the 18th International Conference on Pattern Recognition*, 2:780–783.

Zhang, Y. e Street, W. N. (2008). Bagging with adaptive costs. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):577–588.

Zhou, Y. e Goldman, S. (2004). Democratic co-learning. *Proceedings of the 16 th IEEE International Conference on Tools with Artificial Intelligence*, páginas 594–202.

Zhou, Z.-H. e Li, M. (2005). Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17:1529–1541.

Zhu, X. (2008). Semi-supervised learning literature survey. Relatório técnico, University of Wisconsin - Madison.

Zimmerman, D. W. (1997). A note on interpretation of the paired-samples t test. *Journal of Educational and Behavioral Statistics*, 22(3):349–360.

Zobel, J. (2005). *Writing for computer science*. Springer-Verlag London, 2 edição.

Apêndice A

Fuzzification

A *fuzzification* é uma operação fundamental na lógica *fuzzy*, pois em grande parte das aplicações da lógica *fuzzy* os dados são numéricos, sendo necessário, então, quantizar estes dados em conjuntos *fuzzy*, isto é, converter uma entrada numérica em conjuntos *fuzzy*. O processo de *fuzzification* estabelece uma interface entre os dados descritos numericamente e os descritos simbolicamente (como as palavras, por exemplo).

Os conjuntos *fuzzy* são uma generalização da teoria dos conjuntos tradicionais para resolver os paradoxos gerados a partir da classificação "verdadeira ou falsa" da lógica clássica. Tradicionalmente, uma proposição lógica aceita somente dois extremos: ou é "completamente verdadeira" ou "completamente falsa". Entretanto, nos conjuntos *fuzzy* são aceitáveis valores intermediários entre estes dois extremos, podendo uma proposição ser parcialmente verdadeira ou parcialmente falsa (Fabri e Rissoli, 2000; Mendel, 1995).

Um conjunto *fuzzy* F definido sobre um conjunto de valores X permitido para uma variável é caracterizado por uma função de pertinência (também chamada de função característica, função de descriminação ou função de indicador) $\mu_F(x)$ que assume valores no intervalo [0; 1], onde $x \in X$. A função de pertinência providencia uma medida de grau de similaridade ou de pertinência de um elemento de X para o conjunto *fuzzy* F (Mendel, 1995). Dessa forma, o conceito de "grau de pertinência" expande a teoria dos conjuntos tradicionais, e os grupos são rotulados qualitativamente (usando termos linguísticos pode-se ter: alto, baixo, quente, frio, novo, velho, etc) e os elementos destes conjuntos são caracterizados pelo grau de pertinência que indica o grau que cada elemento pertence a um conjunto. As funções de pertinência podem ser de diferentes tipos, sendo as formas mais comuns a triangular, a trapezoidal, a linear segmentada e a Gaussiana (Mendel, 1995).

A Figura A.1 mostra quatro pontos no espaço $\Re^2$, no intervalo entre [0; 100] para o eixo x e [0; 1] para o eixo y. A *fuzzification* dos pontos é realizada em três funções de pertinência:

pequena (indicada por *S*), média (indicada por *M*) e alta (indicada por *H*). Assim sendo, os pontos após o processo de *fuzzification* são aproximadamente:

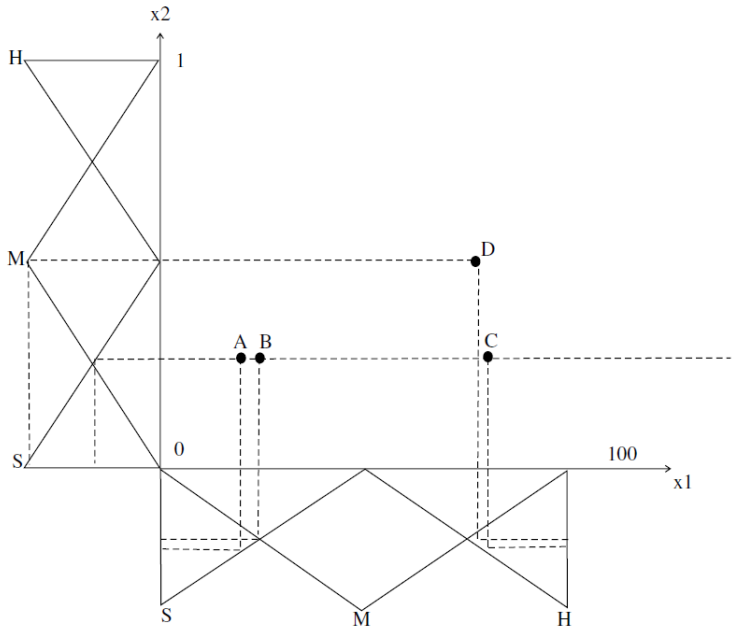


Figura A.1: *Fuzzification* de pontos no espaço $\Re^2$ (Kasabov, 2007).

- ponto *A*:
  - para o valor no eixo *x*1: $S : 0,60$, $M : 0,40$ e $H : 0,00$;
  - para o valor no eixo *x*2: $S : 0,45$, $M : 0,55$ e $H : 0,00$.
- ponto *B*:
  - para o valor no eixo *x*1: $S : 0,50$, $M : 0,50$ e $H : 0,00$;
  - para o valor no eixo *x*2: $S : 0,45$, $M : 0,55$ e $H : 0,00$.
- ponto *C*:
  - para o valor no eixo *x*1: $S : 0,00$, $M : 0,40$ e $H : 0,60$;
  - para o valor no eixo *x*2: $S : 0,45$, $M : 0,55$ e $H : 0,00$.
- ponto *D*:
  - para o valor no eixo *x*1: $S : 0,00$, $M : 0,45$ e $H : 0,55$;
  - para o valor no eixo *x*2: $S : 0,00$, $M : 1,00$ e $H : 0,00$.

Nota-se, do exemplo anterior, que os pontos após a quantização *fuzzy* saíram de um espaço de 2 dimensões para um outro de 6 dimensões.

O número de funções de pertinência é pré-definido. Resoluções maiores são obtidas usando mais funções de pertinência ao preço de uma complexidade computacional maior. As funções de pertinência não precisam se sobrepor, mas uma grande força da lógica *fuzzy* é que as funções de pertinência podem ser feitas para se sobreporem. Isto expressa o fato que um copo pode estar parcialmente *CHEIO* e parcialmente *VAZIO* ao mesmo tempo (Mendel, 1995).

Apêndice B

Curtose

Embora medidas como média e variância possam ser suficientes para caracterizar uma distribuição, como uma Gaussiana, pode haver situações onde tais medidas não fornecem informações suficientes. Então, há necessidade de se recorrer a momentos de alta ordem (média e variância são conhecidos também como momentos de baixa ordem) (Therrien, 1992). Algumas das informações obtidas com momentos de alta ordem são a simetria e o grau de curvatura da distribuição.

A curtose é um momento de alta ordem, mas especificamente, ele é o quarto momento estatístico. Tal medida informa o grau de achatamento de uma distribuição. O índice do momento da curtose para amostras contínuas é dado pela Equação B.1 (Vlassis e Likas, 1999), e a versão discreta é dada pela Equação B.2 (Press et al., 2007).

$$k = \int_{-\infty}^{+\infty} \left(\frac{x-\mu}{\sigma} \right)^4 p(x), \tag{B.1}$$

$$k = \frac{m_4}{std^4}, \tag{B.2}$$

onde m_4 é o momento de quarta ordem centrado na média aritmética e std^4 é o desvio padrão do conjunto, elevado à quarta potência. O quarto momento centrado na média e a quarta potência do desvio padrão são dados, respectivamente, pelas Equações B.3 e B.4:

$$m_4 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^4 f_i}{n}, \tag{B.3}$$

$$std^4 = \left[\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2 f_i}{n} \right]^2, \tag{B.4}$$

onde x_i é um dado de um conjunto de amostras, $\overline{x}$ é a média amostral, n é o número total de amostras e f_i é a ponderação de cada amostra x_i. Para o caso onde as amostras possuem pesos iguais, $f_i = 1 \; \forall \; i$. Pela Equação B.4 pode ser visto que a quarta potência do desvio padrão é a mesma que o quadrado da variância. O erro padrão da estimação da curtose de uma distribuição normal é $\sqrt{96/n}$ quando o desvio padrão é o valor real; e $\sqrt{24/n}$ quando o desvio é estimado das amostras (Press et al., 2007).

O índice do momento da curtose para uma distribuição normal ou Gaussiana é igual a 3, embora existam outras distribuições que possuam curtose igual a 3 (Papoulis, 1991). Sendo assim, a Gaussiana é uma distribuição de alta ordem, no entanto, ela é bem definida pelas informações dos momentos de baixa ordem. Entretanto, como a Gaussiana costuma ser usada como referência para as demais distribuições, é comum encontrar equações para calcular a curtose com um termo -3, para tornar o quarto momento da Gaussiana igual a zero (Press et al., 2007).

Uma distribuição cuja curtose possua o valor 3 é chamada de Mesocúrtica. Se o valor da curtose for menor, a distribuição é chamada de Platicúrtica e a curva da distribuição tem uma aparência mais achatada. Por fim, se a curtose for maior, ela é chamada de Leptocúrtica e a aparência da curva é mais afilada (Press et al., 2007). Esses três tipos diferentes de distribuição são mostrados na Figura B.1. Sendo assim, é possível averiguar se uma distribuição é normal ou não observando o valor de sua curtose (Vlassis e Likas, 1999). No entanto, momentos de alta ordem são quase sempre menos robustos do que momentos de baixa ordem (principalmente na presença de *outliers* (Welling, 2005)), e por isso devem ser usados com cautela (Press et al., 2007). É importante realçar que não existe uma relação entre as

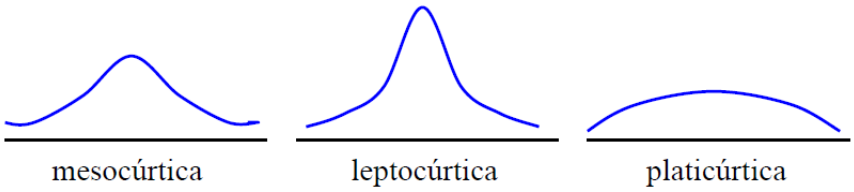


Figura B.1: Tipos de distribuições de acordo com o valor da curtose.

situações de assimetria e as situações de curtose de um mesmo conjunto. Ou seja, assimetria e curtose são medidas independentes e que não se influenciam mutuamente. A medida responsável pela quantificação da assimetria é o momento estatístico de terceira ordem.

Apêndice C

Rede Neural Probabilística

A Rede Neural Probabilística (RNP) foi inicialmente proposta por Specht em 1990 (Specht, 1990). Ela é uma rede neural supervisionada de arquitetura direta, multicamadas com mapeamento não linear da entrada para a saída. Essa rede neural é baseada na teoria de decisão Bayesiana para realizar a classificação de padrões, e para essa tarefa ela estima o modelo da fonte dos dados. Esse modelo é obtido através da estimação da função de densidade de probabilidade de cada padrão, a partir da utilização do método não paramétrico de Parzen (Specht, 1988; Parzen, 1962).

A RNP é composta por quatro camadas: a camada de entrada, a camada de padrões, a camada de soma e a camada de decisão. Na Figura C.1 é mostrada a sua arquitetura, onde cada neurônio na camada de soma corresponde a uma classe e a camada de padrões é parcialmente conectada à camada de soma. Em outras palavras, cada neurônio da camada de padrões está associado a uma classe, sendo, portanto, conectado somente ao neurônio da camada de soma que representa essa classe. Na Figura C.1 x representa o vetor de entrada (amostra à qual é desejada associar uma classe ou rótulo) e $w_{i,j}$ é o vetor peso do neurônio j associado à classe c_i, sendo $i = 1,\ldots,|C|$ e $j = 1,\ldots,n_i$, onde $|C|$ é o número de classes e n_i é o número de neurônios na camada de padrões associados à classe c_i. Ambos os vetores x e $w_{i,j}$ são de dimensão d. O valor de $P(c_i|x)$ representa a probabilidade da amostra x pertencer à classe c_i e $O(x)$ é a saída da rede que indica que classe deve ser associada à amostra x.

A RNP possui um treinamento de apenas um passo para todo o conjunto de treinamento, sendo por isso o treinamento muito rápido em comparação a outros tipos de redes neurais (Georgiou et al., 2004; Duda et al., 2001; Specht, 1988). O treinamento da RNP é simples: para cada amostra de treinamento x da classe c_i é adicionado na camada de padrões um neurônio j associado à classe c_i, cujo vetor de pesos é $w_{i,j} = x$. Além disso, é necessário determinar um valor para o desvio padrão das Gaussianas usadas nas funções de transferência

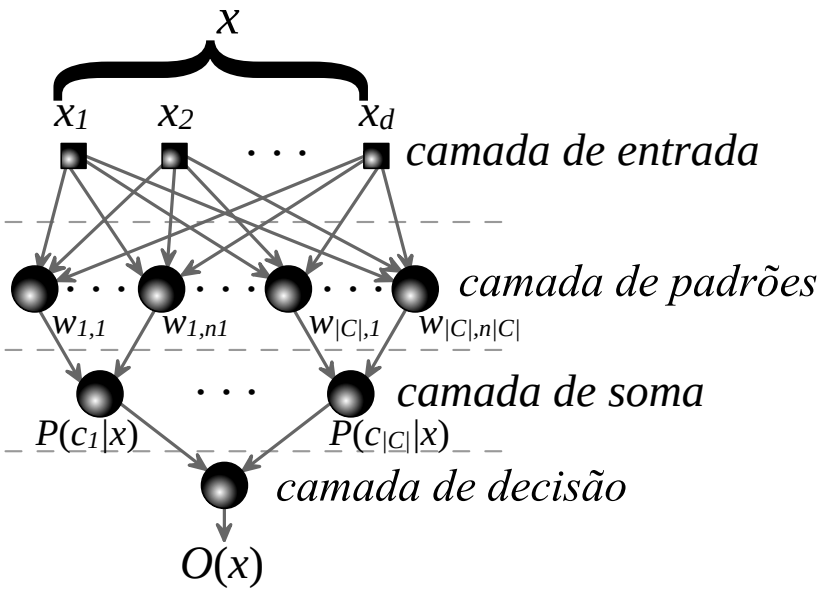


Figura C.1: Arquitetura da Rede Neural Probabilística.

Algoritmo *Algoritmo de aprendizado da Rede Neural Probabilística*

```
1 para cada dado de treinamento faça
2     se o dado é de uma classe que já existe na rede neural então
3         adicionar um neurônio na camada de padrões dessa classe, sendo o vetor de pesos desse neurônio igual ao vetor do dado de treinamento
4     senão
5         criar um neurônio na camada de soma referente à classe do dado
6         adicionar um neurônio na camada de padrões dessa classe, sendo o vetor de pesos desse neurônio igual ao vetor do dado de treinamento
```

Algoritmo 9: Algoritmo de treinamento da Rede Neural Probabilística.

dos neurônios na camada de padrões. O procedimento de aprendizado dessa rede neural é resumido no Algoritmo 9.

Uma vez treinada, a rede é capaz de realizar a tarefa de classificação. Essa tarefa é descrita usando a Figura C.1 para facilitar o entendimento. Inicialmente, o vetor de entrada x é apresentado à camada de entrada da rede. Nessa camada não é realizado nenhum cálculo, e ela simplesmente transmite o vetor de entrada x para a próxima camada: a camada de padrões. A entrada x passa por uma função de transferência em cada neurônio da camada de padrões:

$$ft(x, w_{i,j}, \sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left[\frac{-(x-w_{i,j})^T(x-w_{i,j})}{2\sigma^2}\right], \tag{C.1}$$

para todo $i = 1, \ldots, |C|$ e $j = 1, \ldots, n_i$, sendo n_i o número de neurônios na camada de padrões associados à classe c_i. O símbolo T representa a transposta do vetor $(x - w_{i,j})$. Se tanto x quanto $w_{i,j}$ forem normalizados de forma que $x^T x = w_{i,j}^T w_{i,j} = 1$, a seguinte função de

transferência é obtida:

$$ft(x, w_{i,j}, \sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left[\frac{x^T w_{i,j} - 1}{\sigma^2}\right], \tag{C.2}$$

onde σ (sigma) representa o desvio padrão da Gaussiana e σ^2 a variância. Este é o único parâmetro a ser configurado nas funções de transferência. Sigma muito pequeno causa uma aproximação muito ruidosa e pode não generalizar bem, enquanto que para sigma muito grande a Gaussiana é mais suave e causa perda de detalhes (Nikolaev, 2007).

Em seguida, o resultado obtido na camada de padrões passa para a camada de soma. Nessa camada é calculada a probabilidade da entrada x pertencer à classe c_i, sendo que cada neurônio dessa camada corresponde a uma única classe. Assim, a saída do i-ésimo neurônio dessa camada é dada por:

$$P(c_i|x) = \frac{\frac{h_i}{n_i} \sum_{j=1}^{n_i} ft(x, w_{i,j}, \sigma)}{p(x)}, \tag{C.3}$$

onde:

$$p(x) = \sum_{i=1}^{|C|} \frac{h_i}{n_i} \sum_{j=1}^{n_i} ft(x, w_{i,j}, \sigma), \tag{C.4}$$

onde h_i é a probabilidade *a priori* da classe c_i e $p(x)$ é o fator evidência de x. A divisão por $p(x)$ é meramente um fator de escala para garantir que a soma das probabilidades *posteriori* $P(c_i|x)$ seja igual a um (Duda et al., 2001), podendo ser removida sem alteração no resultado da classificação. Se for considerada a probabilidade *a priori* amostral de cada classe, isto é, $h_i = n_i/n$ (onde n é o número total de neurônios na camada de padrões), o termo h_i/n_i pode ser desconsiderado.

Por último, o resultado da probabilidade de cada classe passa pela camada de decisão, que associa à amostra x a classe que apresentar a maior probabilidade:

$$O(x) = \arg\max_{i=1}^{|C|} P(c_i|x), \tag{C.5}$$

onde arg max retorna a classe com a maior probabilidade.

Os passos para realizar a classificação de uma amostra utilizando essa rede neural são descritos no Algoritmo 10.

Algumas vantagens da Rede Neural Probabilística em relação a outras redes neurais são: fácil implementação, a possibilidade de ser implementada em paralelo e novos padrões e/ou classes poderem ser incorporadas facilmente. Além disso, em experimentos realizados em várias bases de dados, essa rede alcançou resultados comparáveis a outras técnicas mais complexas e custosas (Specht, 1988; Kalatzis et al., 2003; Huang e Liao, 2003; Patra et al., 2002; Fung et al., 2005; Jatmiko et al., 2005).

Algoritmo *Algoritmo de classificação da Rede Neural Probabilística*

1 **para cada** nova amostra **faça**
2 **para cada** classe existente na rede **faça**
3 passar a amostra por cada neurônio na camada de padrões relacionada a essa classe e calcular a saída usando a função de transferência da Equação C.1
4 passar as saídas desses neurônios para o neurônio na camada de soma associado a essa classe, e calcular a saída usando a Equação C.3.
5 Determinar o neurônio com a maior saída na camada de soma e associar a classe relacionada a ela para a amostra

Algoritmo 10: Algoritmo de classificação da Rede Neural Probabilística.

No entanto, essa rede possui algumas desvantagens. Uma delas é a necessidade de armazenar as amostras de cada classe, o que pode ser um problema para tarefas de classificação que envolvem grandes quantidades de amostras com grande dimensões (valores altos de d) (Duda et al., 2001). Outra desvantagem é a possibilidade de amostras redundantes, que pode ocasionar não somente um aumento do esforço computacional como também tornar a rede muito sensível para os dados de treinamento e apresentar baixa capacidade de generalização (Mao et al., 2000).

Apêndice D

Expectation Maximization

Em algumas construções de estimadores e intervalos de confiança, costuma-se usar certas funções de dados, tais como a média e a variância amostral, e se descarta o resto dos dados. Efetivamente, busca-se compactar a quantidade de informação em termos de um número menor de dados. No entanto, surge a questão da quantidade de informação útil que está sendo descartada através dessa compactação e, portanto, se a qualidade desses procedimentos pode ser melhorada se forem realizadas inferências sobre algumas das informações que a princípio seriam descartadas (Kemp, 2002).

O ponto de partida para determinar se estão sendo ou não descartadas informações valiosas através da compactação dos dados é a função de verossimilhança. A função de verossimilhança é uma função dos parâmetros obtidos através dos dados observados e ela busca verificar a verossimilhança entre os dados observados e os parâmetros obtidos. Com o objetivo de encontrar o conjunto de parâmetros que foi responsável por gerar os dados observados, procura-se fazer a estimação da máxima verossimilhança de um conjunto de parâmetros, que é simplesmente estimar o conjunto de valores de parâmetros que consegue maximizar a função de verossimilhança dos dados observados (Kemp, 2002).

O objetivo do algoritmo de *Expectation Maximization* é encontrar a solução de máxima verossimilhança através de modelos usando variáveis latentes (ou variáveis ocultas), que são variáveis que não podem ser observadas diretamente, mas podem ser inferidas indiretamente através de outras variáveis que podem ser observadas ou medidas diretamente (Bishop, 2006). Este é um método elegante, poderoso e amplamente usado para a obtenção da máxima verossimilhança, e que foi apresentado formalmente por Dempster, Laird e Rubin (Dempster et al., 1977).

Seja X o conjunto de todos os dados observados, Z o conjunto de todas as variáveis latentes do modelo e θ o conjunto de todos os parâmetros do modelo. Então, a função do

logaritmo da verossimilhança[1] é obtida por:

$$\ln p(X|\theta) = ln\left\{\sum_{Z} p\,(X,Z|\theta)\right\}. \tag{D.1}$$

Uma observação importante é que o somatório das variáveis latentes aparece dentro do logaritmo. A presença da soma evita que o logaritmo atue diretamente sobre a distribuição conjunta, resultando em expressões complexas para a solução da máxima verossimilhança.

No entanto, se para cada observação em X fosse informado o correspondente valor da variável latente Z (ou seja, qual a contribuição de cada variável na formação da observação), então o conjunto de dados completo estaria disponível, podendo ser chamado de $\{X,Z\}$; sendo o conjunto de dados observados X referenciado como incompleto. Com o conjunto de dados completo, a função de verossimilhança é da forma $\ln p(X,Z|\theta)$, e a maximização da função do logaritmo da verossimilhança dos dados completos é obtida de forma direta.

Na prática, porém, não é disponibilizado o conjunto de dados completo $\{X,Z\}$, mas somente os dados incompletos X. O conhecimento dos valores das variáveis latentes em Z é dado somente pela distribuição a *posteriori* $p(Z|X,\theta)$. Como não se pode usar o logaritmo da verossimilhança dos dados completos, é usado o valor esperado da variável latente obtido a partir da distribuição a *posteriori*, que corresponde ao passo **E** (*Expectation*) do algoritmo de *Expectation Maximization*. No passo seguinte, denominado passo **M** (*Maximization*), é obtido um novo conjunto de valores dos parâmetros que tendem maximizar a função do logaritmo da verossimilhança. Dessa forma, os atuais parâmetros são simbolizados por θ_t e, após o sucessivo emprego dos passos E e M, é obtida uma estimativa aprimorada θ_{t+1} (Bishop, 2006). Inicialmente este algoritmo é inicializado escolhendo alguns valores iniciais para os parâmetros θ_0.

No passo E, os correntes valores dos parâmetros θ_t são usados para encontrar a distribuição a *posteriori* das variáveis latentes dada por $p(Z|X,\theta_t)$. Essa distribuição a *posteriori* é utilizada para encontrar a expectativa do logaritmo da verossimilhança dos dados completos avaliados para algum valor do parâmetro θ. Essa expectativa, denotada por $\wp(\theta,\theta_t)$, é dada por (Bishop, 2006):

$$\wp(\theta,\theta_t) = \sum_{Z} p\,(Z|X,\theta_t)\ln p\,(X,Z|\theta). \tag{D.2}$$

No passo M, é obtida uma estimativa aprimorada θ_{t+1} maximizando a função:

$$\theta_{t+1} = \arg\max_{\theta} \wp(\theta,\theta_t), \tag{D.3}$$

[1] O uso do logaritmo da função ao invés do simples uso da função é justificado por duas razões (Gould et al., 2006). Do ponto de vista estatístico, é mais fácil obter os valores dos parâmetros e do ponto de vista numérico, os valores obtidos são mais facilmente representados na faixa usual de representação computacional dos dados.

Algoritmo *Algoritmo de Expectation Maximization*

1 escolher um conjunto inicial de parâmetros para θ_0 e definir $t = 0$
2 calcular o valor inicial do logaritmo de verossimilhança
3 **faça**
4 passo E: calcular $p(Z|X,\theta_t)$
5 passo M: calcular θ_{t+1} usando Equação D.3
6 $\theta_t = \theta_{t+1}$, $t = t+1$
7 calcular o valor do logaritmo de verossimilhança para θ_t
8 **enquanto** a convergência do logaritmo da verossimilhança ou dos valores dos parâmetros não for alcançada

Algoritmo 11: Algoritmo de *Expectation Maximization*.

onde arg max retorna o conjunto de parâmetros que maximizam $\wp(\theta,\theta_t)$.

Com o procedimento apresentado anteriormente, o logaritmo pode atuar diretamente sobre a distribuição conjunta $p(X,Z|\theta)$ e o passo de maximização M será tratável. O algoritmo geral de *Expectation Maximization* é mostrado no Algoritmo 11 (Bishop, 2006).

Cada ciclo de *Expectation Maximization* irá incrementar o logaritmo da verossimilhança dos dados incompletos, a menos que ele já tenha alcançado um ótimo local ou global.

O algoritmo de *Expectation Maximization* pode também ser aplicado quando as variáveis não observáveis correspondem a valores faltantes no conjunto de dados. No entanto, ele só será válido se os valores dos dados estiverem faltando aleatoriamente (Bishop, 2006).

D.1 Modelo de Mistura de Gaussianas

Um caso específico da aplicação de *Expectation Maximization* é estimar os parâmetros do Modelo de Mistura de Gaussianas (MMG) que melhor podem se ajustar a um conjunto de dados observados (Bishop, 2006). Novamente, o objetivo é maximizar o logaritmo da função de verossimilhança do modelo de misturas através de um conjunto de dados observados. Sabendo que uma distribuição de misturas Gaussianas pode ser escrita como uma superposição linear de Gaussianas, tem-se, para um valor observado x, representado por um vetor com d dimensões, a seguinte expressão:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k,\Sigma_k), \tag{D.4}$$

onde $\mathcal{N}(x|\mu_k,\Sigma_k)$ é uma distribuição Gaussiana, e é dada por:

$$\mathcal{N}(x|\mu_k,\Sigma_k) = \frac{1}{\sqrt{(2\pi)^d|\Sigma_k|}} exp\left[\frac{-(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}{2}\right], \tag{D.5}$$

sendo K o número de componentes Gaussianas da mistura, μ_k e Σ_k o vetor média e a matriz de covariância da componente k de d e $d \times d$ dimensões, respectivamente, e π_k é o coeficiente de mistura da componente k (a contribuição de k na mistura). Além disso, $|\Sigma_k|$ e Σ_k^{-1} são o determinante e a inversa da matriz de covariância Σ_k, respectivamente, e T representa a transposta do vetor $(x - \mu_k)$.

Suponha agora não um, mas um conjunto X de N dados observados i.i.d. (independentes e identicamente distribuídos) e, aplicando o logaritmo sobre a Equação D.4, conforme a Equação D.1, e após algumas manipulações matemáticas, é obtido:

$$\ln p(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k) \right\}, \tag{D.6}$$

onde x_n é o n-ésimo dado do conjunto X.

Para descobrir o valor de μ_k que maximiza a Equação D.6, a equação é derivada em relação a μ_k e a derivada é igualada a zero, obtendo assim a Equação D.7:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{n,k} x_n, \tag{D.7}$$

sendo:

$$N_k = \sum_{n=1}^{N} \gamma_{n,k}, \tag{D.8}$$

e:

$$\gamma_{n,k} = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}. \tag{D.9}$$

O valor de n_k pode ser interpretado como o número efetivo de pontos associados ao componente k da mistura. Pode ser notado que a média μ_k para a k-ésima componente Gaussiana é obtida usando uma média ponderada de todos os pontos no conjunto de dados, e que o fator de ponderação para o ponto x_n é dado pela probabilidade *a posteriori* $\gamma_{n,k}$ do componente k ter sido responsável por gerar x_n.

Derivando a Equação D.6 em relação Σ_k e igualando o resultado igual a zero, é obtido o valor de Σ_k que maximiza a Equação D.6:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{n,k} (x_n - \mu_k)(x_n - \mu_k)^T. \tag{D.10}$$

Finalmente, $\ln p(X|\pi,\mu,\Sigma)$ é maximizado em relação ao coeficiente de mistura π_k. Levando em conta que a soma dos coeficientes de mistura deve ser igual a um, deve ser usado

Algoritmo *Algoritmo de Expectation Maximization aplicado ao Modelo de Misturas Gaussianas*

1 determinar o número de componentes da mistura (K) e escolher os valores iniciais para as médias μ_k, covariâncias Σ_k e coeficientes de mistura π_k
2 calcular o valor inicial do logaritmo de verossimilhança (Equação D.13)
3 **faça**
4 passo E: calcular as responsabilidades aplicando a Equação D.9
5 passo M: re-estimar os parâmetros usando as responsabilidades correntes e aplicando as Equações D.7, D.10 e D.12
6 calcular o valor do logaritmo de verossimilhança para os novos parâmetros estimados usando a Equação D.13
7 **enquanto** a convergência do logaritmo da verossimilhança ou dos valores dos parâmetros não for alcançada

Algoritmo 12: Algoritmo de *Expectation Maximization* aplicado ao MMG.

o multiplicador da Lagrange e maximizado a seguinte equação:

$$\ln p(X|\pi,\mu,\Sigma) + \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right). \tag{D.11}$$

Realizando os cálculos necessários é obtido:

$$\pi_k = \frac{N_k}{N}. \tag{D.12}$$

Assim, o coeficiente de mistura para o k-ésimo componente pode ser visto como a responsabilidade média que aquele componente tem para explicar os pontos de dados.

É importante realçar que as Equações D.7, D.10 e D.12 não são uma solução de forma fechada para a obtenção dos parâmetros do modelo de mistura, porque a responsabilidade $\gamma_{n,k}$ depende desses parâmetros de uma maneira complexa através de D.9. Por outro lado, se estivesse disponível o conjunto de dados completo $\{X,Z\}$, as soluções dadas pelas Equações D.7, D.10 e D.12 seriam a forma fechada para a obtenção dos parâmetros (Bishop, 2006).

Assim, como no caso mais geral, também será usado o esquema iterativo dos passos E e M para obter os parâmetros do modelo. Enquanto que, no passo E, são usados os valores correntes dos parâmetros para calcular as probabilidades posteriori, ou responsabilidades $\gamma_{n,k}$; o passo M é usado para re-estimar as médias, as covariâncias e os coeficientes de mistura usando as Equações D.7, D.10 e D.12. O algoritmo de *Expectation Maximization* aplicado ao modelo de mistura de Gaussianas é mostrado no Algoritmo 12 (Bishop, 2006).

$$\ln p(X|\mu,\Sigma,\pi) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)\right\}. \tag{D.13}$$

Apêndice E

Publicações

Como parte desse trabalho foram desenvolvidos e publicados os seguintes trabalhos no período do doutorado que possuem, em maior ou menor grau, relação com o tema proposto:

Publicações em revista:

1. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. An incremental neural network with a reduced architecture. Neural Networks, v. 35, p. 70-81, 2012.

2. Souza, A. F.; Pedroni, F.; Oliveira, E.; Ciarelli, P. M.; Henrique, W. F.; Veronesse, L.; Badue, C.. Automated multi-label text categorization with VG-RAM weightless neural networks. Neurocomputing, v. 72, p. 2209-2217, 2009.

3. Ciarelli, P. M.; Oliveira, E.; Badue, C.; Souza, A. F.. Multi-label text categorization using a Probabilistic Neural Network. International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), v. 1, p. 133-144, 2009.

4. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. Multi-label incremental learning applied to web page categorization. Neural Computing and Applications, 2013.

5. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. Impact of the characteristics of data sets on incremental learning. Engineering Applications of Artificial Intelligence. (A ser publicado).

Publicações em capítulo de livro:

1. Ciarelli, P. M.; Krohling, R. A.; Oliveira, E.. Particle Swarm Optimization Applied to Parameters Learning of Probabilistic Neural Networks for Classification of Economic Activities. In: Aleksandar Lazinica. (Org.). Particle Swarm Optimization. Viena, Austria: I-Tech Education and Publishing, p. 313-328, 2009.

2. Oliveira, E.; Ciarelli, P. M.; Salles, E. O. T.. An Evolving System in the Text Classification Problem. In: IGI Global. (Org.). Handbook of Research on Computational Intelligence for Engineering, Science, and Business. Estados Unidos: IGI Global, v. 1, p. 467-486, 2012.

Trabalhos completos publicados em anais de congressos:

1. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. An Evolving System Based on Probabilistic Neural Network. Brazilian Symposium on Artificial Neural Network (SBRN), São Bernardo do Campo, 2010.

2. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. Neural Network based in expectation maximization for on-line text categorization. IX Simpósio Brasileiro de Automação Inteligente (SBAI), Brasília, 2009.

3. Ciarelli, P. M.; Oliveira, E.. An Enhanced Probabilistic Neural Network Approach Applied to Text Classification. 14th Iberoamerican Congress on Pattern Recognition (CIARP), Guadalajara, 2009.

4. Ciarelli, P. M.; Oliveira, E.. Agglomeration and Elimination of Terms for Dimensionality Reduction. 9th International Conference on Intelligent Systems Design and Applications (ISDA), Roma, 2009.

5. Freitas, F. D. De; Ciarelli, P. M.; Souza, A. F.. Previsão da arrecadação federal com redes neurais. IX Congresso Brasileiro de Redes Neurais, Ouro Preto, 2009.

6. Oliveira, E.; Ciarelli, P. M.; Badue, C.; Souza, A. F.. Using a Probabilistic Neural Network for a Large Multi-label Problem. Brazilian Symposium on Artificial Neural Networks (SBRN), Salvador, 2008.

7. Ciarelli, P. M.; Oliveira, E.. A Comparison Between a kNN based Aproach and a PNN Algorithm for a Multi-Label Classification Problem. VIII International Conference on Intelligent Systems Design and Applications (ISDA), Taiwan, 2008.

Resumos expandidos publicados em anais de congressos:

1. Ciarelli, P. M.; Oliveira, E.. Avaliando o Desempenho de uma Rede Neural Probabilística Baseada em Centróides. 7th Brazilian Symposium in Information and Human Language Technology, São Carlos, 2009.

Resumos publicados em anais de congressos:

1. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. Uma Rede Neural Probabilística Baseada em Modelos Evolutivos. 42º Simpósio Brasileiro de Pesquisa Operacional (SOBRAPO), Bento Gonçalves, 2010.

2. Ciarelli, P. M.; Oliveira, E.; Salles, E. O. T.. Rede Neural com Aprendizado Incremental Aplicada à Predição da Arrecadação Federal. 42º Simpósio Brasileiro de Pesquisa Operacional (SOBRAPO), Bento Gonçalves, 2010.