Pós-Graduação em Ciência da Computação

"UMA METODOLOGIA PARA OTIMIZAÇÃO DE ARQUITETURAS E PESOS DE REDES NEURAIS"

Por

AKIO YAMAZAKI

Tese de Doutorado

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

RECIFE, MARÇO/2004

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

AKIO YAMAZAKI

"UMA METODOLOGIA PARA OTIMIZAÇÃO DE ARQUITETURAS E PESOS DE REDES NEURAIS"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR(A): TERESA BERNARDA LUDERMIR

RECIFE, MARÇO/2004

Sumário

Este trabalho propõe uma metodologia para a otimização global de redes neurais. O objetivo é a otimização simultânea de arquiteturas e pesos de redes *Multi-Layer Perceptron* (MLP), com o intuito de gerar topologias com poucas conexões e alto desempenho de classificação para qualquer conjunto de dados.

A otimização simultânea de arquiteturas e pesos de redes neurais é uma abordagem interessante para a geração de redes eficientes com topologias pequenas. Tal aplicação já originou alguns trabalhos com algoritmos genéticos, entretanto existem outras técnicas, como simulated annealing e tabu search, que ainda não foram exploradas para esta finalidade até o presente momento.

Métodos de otimização global podem ser combinados com uma técnica baseada em gradiente (por exemplo, o algoritmo *backpropagation*) em uma abordagem de treinamento híbrido, que procura unir, no mesmo sistema, a eficiência global dos métodos de otimização com o ajuste fino das técnicas baseadas em gradiente. Tal combinação não tem sido estudada para simulated annealing e tabu search, e isto gerou outra motivação para o presente trabalho.

Os resultados mostram que a combinação cuidadosa de técnicas tradicionais de otimização global, como simulated annealing e tabu search, com redes neurais artificiais e métodos baseados em gradiente é capaz de produzir sistemas híbridos bastante eficientes. Por este motivo, uma metodologia foi desenvolvida, combinando as vantagens de simulated annealing, de tabu search e do treinamento híbrido, a fim de gerar um processo automático para obter redes MLP com topologias pequenas e alto desempenho de generalização.

Esta metodologia representa um grande avanço na área de sistemas neurais híbridos e fornece resultados importantes para diversas aplicações práticas. Este trabalho apresenta resultados da aplicação da metodologia proposta em dois domínios práticos: reconhecimento de odores em um nariz artificial e diagnóstico de diabetes. Em ambos os casos, a metodologia obteve resultados satisfatórios e gerou redes com baixo erro de generalização e baixa complexidade. Tais resultados são extremamente importantes para mostrar que a combinação de técnicas de otimização é capaz de produzir sistemas híbridos superiores.

Abstract

This work introduces a methodology for neural network global optimization. The aim is the simultaneous optimization of *Multi-Layer Perceptron* (MLP) network weights and architectures, in order to generate topologies with few connections and high classification performance for any data sets.

Simultaneous optimization of neural network architectures and weights is an interesting approach for the generation of efficient networks with small topologies. Such application has already generated some works with genetic algorithms, but there are other techniques, such as simulated annealing and tabu search, which have not been explored for this purpose until this moment.

Global optimization methods can be combined with a gradient based technique (for example, the backpropagation algorithm) in a hybrid training approach, which tries to join, in the same system, the global efficiency of the optimization methods with the fine tuning of the gradient based techniques. Such combination has not been studied for simulated annealing and tabu search, and this provided another motivation for this work.

The results show that the careful combination of traditional global optimization techniques, like simulated annealing and tabu search, with artificial neural networks and gradient based methods is able to produce very efficient hybrid systems. For this reason, a methodology was developed, combining the advantages of simulated annealing, tabu search and hybrid training, in order to generate an automatic process for obtaining MLP networks with small topologies and high generalization performance.

This methodology represents a great advance in the field of neural hybrid systems and provides important results for several practical applications. This work presents results of the application of the proposed methodology in two practical domains: odor recognition in an artificial nose and diagnostics of diabetes. In both cases, the methodology has obtained satisfactory results and has generated networks with low generalization error and low complexity. Such results are extremely important to show that the combination of optimization techniques is able to produce superior hybrid systems.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Motivação

Em muitas aplicações de redes neurais, o modelo mais comumente utilizado é o *Multi-Layer Perceptron* (MLP) treinado com o algoritmo backpropagation [Rumelhart et al., 1986]. A definição da arquitetura é um tema crucial na aplicação de redes MLP, uma vez que a escolha da topologia tem um impacto significativo na capacidade de processamento da rede a ser utilizada. Dependendo do problema abordado, uma rede neural com poucas conexões pode não ser capaz de resolver a tarefa, devido à quantidade insuficiente de parâmetros ajustáveis. Por outro lado, se a rede possuir conexões demais, pode haver um ajuste excessivo aos dados de treinamento, em um

fenômeno conhecido como *overfitting*, prejudicando a capacidade de generalização da rede. Uma introdução às redes MLP pode ser encontrada no Apêndice A.

Em geral, experimentos com redes MLP são realizados através de repetidas tentativas com diferentes topologias (por exemplo, aumentando progressivamente a quantidade de nodos escondidos) até serem obtidos resultados satisfatórios. Além de consumir bastante tempo, este processo pode obter redes com conexões e nodos desnecessários, pois a escolha da topologia pode não ter sido suficientemente cuidadosa para encontrar arquiteturas as mais compactas possíveis.

Dessa forma, torna-se essencial o desenvolvimento de técnicas automáticas para definição de topologias para uma rede MLP. Técnicas de otimização global, como algoritmos genéticos [Holland, 1975], simulated annealing [Kirkpatrick et al., 1983] e tabu search [Glover, 1986][Hansen, 1986], têm-se apresentado como boas opções para abordar este problema.

A escolha da arquitetura ótima para um dado problema pode ser formulada como um problema de otimização no espaço de possíveis topologias. Considerando alguma função de custo, que pode levar em conta, por exemplo, o erro nos dados de treinamento e o tamanho da arquitetura, o custo de todas as topologias forma uma superfície discreta no espaço, de modo que a escolha da arquitetura ótima passa a ser equivalente à busca do ponto de mínimo global desta superfície.

Um grave problema que surge na otimização de arquiteturas, quando os pesos das conexões não são codificados nas soluções, é que a avaliação das redes passa a apresentar um ruído, pois uma rede treinada com um conjunto completo de pesos é utilizada para avaliar o custo de uma solução, que não contém nenhuma informação sobre os pesos. Dessa forma, diferenças nos parâmetros de treinamento e nas inicializações de pesos podem gerar resultados distintos para a mesma topologia MLP.

Uma alternativa interessante para resolver este problema é a otimização simultânea de arquiteturas e pesos. Segundo esta abordagem, cada ponto do espaço de busca codifica tanto a estrutura topológica da rede como os valores de seus pesos, tornando a avaliação do custo bastante precisa. Para cumprir este objetivo, existem várias técnicas, sendo que algoritmos genéticos [Holland, 1975] têm sido mais

freqüentemente aplicados nos trabalhos existentes na literatura. Entretanto, técnicas como simulated annealing [Kirkpatrick et al., 1983] e tabu search [Glover, 1986][Hansen, 1986] ainda não foram suficientemente exploradas para este objetivo, pois a maioria dos trabalhos que as utilizam têm como finalidade apenas treinar os pesos de topologias fixas. Até o presente momento, ambas as técnicas não foram abordadas para otimizar simultaneamente arquiteturas e pesos de redes neurais, e esta é uma das contribuições deste trabalho.

Além disso, sabe-se que técnicas de otimização global, como simulated annealing e tabu search, são relativamente ineficientes para ajuste fino em buscas locais. Dessa forma, é importante investigar se o desempenho de generalização das redes ainda pode ser melhorado quando as topologias geradas por estas técnicas são treinadas com uma abordagem de busca local, como, por exemplo, o conhecido algoritmo backpropagation. Esta combinação de abordagens de otimização global com técnicas locais, que é freqüentemente chamada de *treinamento híbrido*, tem sido utilizada em trabalhos com algoritmos genéticos [Yao, 1999]. Entretanto, esta abordagem não tem sido comum nos trabalhos com simulated annealing e tabu search, e esta é outra contribuição do presente trabalho.

Neste texto, o algoritmo backpropagation é chamado de técnica *local*, enquanto simulated annealing, tabu search e algoritmos genéticos são considerados técnicas *globais*. Por este motivo, é importante explicar melhor o uso destes termos no presente trabalho.

Técnicas de gradiente, como backpropagation, utilizam informações sobre a derivada da superfície de erro no espaço de busca. Tais informações são consideradas *locais* porque indicam apenas a inclinação da superfície do erro *em torno do ponto onde foram calculadas*. A solução é ajustada de acordo com estas informações locais, de modo a caminhar sempre para o ponto de mínimo da região onde se encontra a solução atual. É por este motivo que estas abordagens são chamadas de *locais*.

Por outro lado, técnicas como simulated annealing, tabu search e algoritmos genéticos exploram o espaço de busca sem fazer uso de informações sobre a inclinação da superfície de erro em torno da solução atual. Os ajustes, que geralmente são

caracterizados por pequenas perturbações aleatórias, são feitos de modo a gerar soluções novas a serem avaliadas, e esta geração não é feita de modo a caminhar sempre para o ponto de mínimo da região onde está a solução atual. Dessa forma, a exploração de novos pontos do espaço de busca não é feita localmente, e, por este motivo, simulated annealing, tabu search e algoritmos genéticos são considerados técnicas *globais* de otimização.

1.2 Objetivos

Este trabalho tem o objetivo de propor uma metodologia para a otimização simultânea de arquiteturas e pesos de redes MLP. A metodologia foi cuidadosamente planejada para desenvolver uma abordagem automática capaz de gerar redes com poucas conexões e alto desempenho de generalização para uma base de dados qualquer. O processo de otimização combina características de simulated annealing e de tabu search, com o intuito de aproveitar as características favoráveis destas técnicas, evitando suas limitações. A metodologia inclui, ainda, a aplicação de um algoritmo local de treinamento, que pode ser, por exemplo, o conhecido backpropagation, para o ajuste fino dos pesos, em uma abordagem de treinamento híbrido.

Até o presente momento, não se encontra, na literatura, uma combinação entre estas técnicas da maneira como foi proposta neste trabalho, procurando explorar ao máximo as potencialidades de cada método, de modo que uma parte possa compensar as deficiências da outra. Dessa forma, o sistema híbrido resultante se torna muito mais eficiente do que suas técnicas constituintes funcionando isoladamente.

Para desenvolver esta metodologia, foi necessário definir alguns aspectos importantes na aplicação de técnicas de otimização global, como a representação das soluções no espaço de busca, a função de custo e o mecanismo de geração de soluções novas (operador). Dessa forma, tornou-se importante aplicar estas etapas no uso de simulated annealing, tabu search e algoritmos genéticos, a fim de verificar se estas escolhas realmente são favoráveis à aplicação das técnicas já conhecidas de otimização global. A análise destes resultados, que são mostrados nos Capítulos 4 e 5, teve o objetivo de justificar a escolha da representação das soluções, da função de custo e do operador no desenvolvimento da metodologia proposta.

No Capítulo 4, o problema utilizado como estudo de caso foi a classificação de odores provenientes de safras distintas de um mesmo vinho em um nariz artificial. O mesmo conjunto de dados já foi utilizado pelo autor deste trabalho em abordagens propostas anteriormente para melhorar o desempenho de classificação [Yamazaki, 2001][Yamazaki and Ludermir, 2001][Yamazaki et al., 2001]. A necessidade de instrumentos portáteis para o reconhecimento de odores torna necessário estudar a viabilidade de implementar o sistema de classificação em hardware. Assim, é muito importante implementar em hardware as redes neurais que apresentem os melhores desempenhos no reconhecimento dos odores de interesse. Por este motivo, torna-se essencial que as redes tenham o menor número possível de conexões, e esta é a principal motivação para a utilização deste conjunto de dados a fim de testar a metodologia proposta.

Entretanto, é importante ressaltar que a metodologia proposta foi desenvolvida para otimizar redes MLP em quaisquer domínios, com o objetivo de apresentar um processo automático que gera, a partir de um conjunto de dados, redes MLP com poucas conexões e alto desempenho de classificação. Desta forma, torna-se importante aplicar a metodologia proposta em um outro domínio, a fim de verificar o desempenho para um novo conjunto de dados. Este é o objetivo do Capítulo 5, que apresenta os resultados obtidos na aplicação da metodologia proposta para o problema do diagnóstico de diabetes em índias Pima, conjunto de dados extraído do conhecido *UCI Machine Learning Repository* [Blake and Merz, 1998]. Estes dados são bastante conhecidos na área de aprendizado de máquina e já foram abordados por diversos trabalhos propostos na literatura [Yao and Liu, 1997][Islam and Murase, 2001].

Além disso, o presente trabalho traz um resumo das abordagens de otimização global de redes neurais usando as técnicas mais tradicionalmente conhecidas (algoritmos genéticos, simulated annealing e tabu search). Tal estudo teve a finalidade de contribuir com um resumo sobre este tópico, pois não é comum encontrar, até o presente momento, um resumo das abordagens de simulated annealing e tabu search para otimizar redes neurais, embora existam bons trabalhos sobre algoritmos genéticos para a mesma finalidade, entre os quais podem ser citados [Branke, 1995], [Yao, 1999], [Lacerda, 2003] e [Lacerda et al., 2002]. Este estudo pode servir como material

introdutório para diversas linhas de pesquisa relacionadas com o uso destas técnicas para otimizar quaisquer aspectos das redes neurais artificiais.

1.3 Organização da Tese

Neste capítulo introdutório, a motivação e os objetivos deste trabalho foram apresentados.

O Capítulo 2 traz um resumo do que existe na literatura sobre métodos de otimização global para redes neurais. Para as três técnicas de otimização mais tradicionais (algoritmos genéticos, simulated annealing e tabu search), são apresentadas noções gerais sobre o funcionamento dos métodos, sendo estudadas, em seguida, as principais aplicações na otimização de diversos aspectos das redes neurais, como o ajuste de pesos e a escolha das arquiteturas.

O Capítulo 3 trata da metodologia proposta, apresentando os detalhes de funcionamento do processo desenvolvido para otimizar simultaneamente arquiteturas e pesos de redes MLP.

O Capítulo 4 apresenta os resultados obtidos para o reconhecimento de odores em um nariz artificial. Inicialmente, apresentam-se explicações sobre a base de dados utilizada, tornando mais clara a importância do domínio prático abordado. Em seguida, são resumidos os trabalhos anteriores do autor desta tese [Yamazaki, 2001][Yamazaki and Ludermir, 2001][Yamazaki et al., 2001], que utilizaram a mesma base de dados em uma abordagem de redes neurais, mas sem a preocupação com a otimização das arquiteturas. Depois, apresentam-se os resultados para simulated annealing, tabu search e algoritmo genético, aplicando a representação de soluções, a função de custo e o operador definidos no Capítulo 3, mostrando que estas escolhas são adequadas para produzir redes MLP com poucas conexões e alta capacidade de generalização. Além disso, são apresentados os resultados obtidos na aplicação da metodologia proposta para este conjunto de dados. Todos estes resultados são comparados através de um teste de diferenças entre médias, para fornecer um respaldo estatístico às conclusões observadas. Por fim, é feita uma análise na seleção de atributos realizada nos experimentos, com o intuito de verificar a importância da contribuição de cada atributo na tarefa de classificação.

O Capítulo 5 mostra os resultados obtidos na aplicação da metodologia proposta ao problema do diagnóstico de diabetes, além dos resultados gerados por simulated annealing, tabu search e algoritmo genético, sendo feita uma comparação com outras técnicas já propostas na literatura e aplicadas ao mesmo conjunto de dados.

No Capítulo 6, apresentam-se as conclusões obtidas com o trabalho desenvolvido e as possibilidades de atividades futuras.

No Apêndice A, apresenta-se uma breve introdução sobre redes neurais artificiais em geral, sendo comentadas as abordagens MLP e TDNN [Lang e Hinton, 1988], que são citadas neste trabalho.

No Apêndice B, são discutidos os narizes artificiais. Apresenta-se um breve histórico sobre o desenvolvimento destes dispositivos, e, em seguida, são analisadas suas partes constituintes, bem como as diversas abordagens existentes para cada componente, com o objetivo de contextualizar o domínio abordado na obtenção dos resultados do Capítulo 4.

Capítulo 2
Otimização Global de Redes Neurais

2.1 Introdução

Atualmente, tem merecido crescente atenção, na área de Inteligência Artificial, o desenvolvimento de *sistemas híbridos*, que resultam da combinação de duas ou mais técnicas distintas para resolver um dado problema. A motivação para tais sistemas está no fato de que as diversas técnicas existentes de Inteligência Artificial podem ser adequadas para determinados casos, mas podem apresentar deficiências significativas para resolver outros tipos de problemas. Estas limitações estimulam o estudo dos sistemas híbridos, os quais procuram combinar as características favoráveis de duas ou mais técnicas, com o intuito de superar as limitações que cada uma apresenta individualmente na resolução do problema de interesse.

Entre estes sistemas híbridos, especial ênfase tem sido dedicada à combinação de técnicas de otimização global e redes neurais artificiais. Métodos de otimização global, como *algoritmos genéticos* [Holland, 1975], *simulated annealing* [Kirkpatrick et al., 1983] e *tabu search* [Glover, 1986][Hansen, 1986], têm sido aplicados em diversas abordagens usando redes neurais, servindo para as mais variadas finalidades, como treinamento dos pesos em uma topologia fixa e busca de arquiteturas ótimas.

Neste capítulo, apresentam-se comentários sobre o uso de técnicas de otimização global combinadas com redes neurais. São analisadas aplicações com algoritmos genéticos, simulated annealing e tabu search, que são as técnicas de otimização global mais tradicionais e conhecidas atualmente. Vale ressaltar que o objetivo não é descrever detalhadamente cada método de otimização, pois tais explicações já existem em grande quantidade na literatura. A intenção é apresentar uma visão geral de como estas técnicas já foram empregadas para otimizar redes neurais, a fim de contextualizar o presente trabalho. Vale comentar, também, que, nos dias de hoje, a aplicação de algoritmos genéticos para otimização de redes neurais está muito mais avançada do que o uso dos demais métodos para a mesma finalidade.

2.2 Algoritmos Genéticos

O primeiro aspecto a ser considerado na implementação de um algoritmo genético é a representação dos parâmetros do problema, ou seja, a codificação das possíveis soluções do problema em estruturas que podem ser manipuladas pelos algoritmos genéticos. Uma solução possível do problema, antes da codificação, recebe o nome de *fenótipo.* Cada fenótipo é codificado em uma estrutura, que recebe o nome de *indivíduo*, *cromossomo* ou *genótipo*. Algoritmos genéticos trabalham com um conjunto de indivíduos simultaneamente, e este conjunto recebe o nome de *população*. Cada indivíduo da população é associado a uma *aptidão*, que representa a capacidade da solução candidata de resolver o problema de interesse [Holland, 1975].

O funcionamento de um algoritmo genético envolve uma seqüência de iterações, que também são chamadas de *gerações*. A cada geração, a população passa pelos processos de *seleção* (escolha dos indivíduos a serem reproduzidos) e *reprodução* (combinação e/ou modificação dos indivíduos selecionados, produzindo os indivíduos

da próxima geração). Um dos métodos de seleção mais utilizados é o da *roleta*, em que cada indivíduo ocupa, em uma roleta, uma área proporcional a sua aptidão. A roleta é girada *N* vezes, sendo escolhidos os *N* indivíduos que participarão da fase de reprodução. Dessa forma, indivíduos com maiores aptidões ficam com maiores probabilidades de serem escolhidos e, portanto, de transferirem suas características para a população seguinte. A reprodução é feita por meio de *operadores genéticos*, que procuram manter as características dos indivíduos selecionados nos novos indivíduos a serem gerados. Os operadores genéticos principais são o *cruzamento* e a *mutação*. O cruzamento é responsável pela combinação de características dos *pais* (indivíduos originais), a fim de gerar os *filhos* (indivíduos criados pela reprodução), sendo aplicado com uma determinada taxa, chamada de *taxa de cruzamento.* A mutação procura manter a diversidade genética na população, fazendo modificações arbitrárias em uma ou mais partes de indivíduos escolhidos aleatoriamente. A taxa com que este operador é aplicado recebe o nome de *taxa de mutação.*

O fato de existirem mais trabalhos que se utilizam de algoritmos genéticos para otimizar redes neurais certamente está associado às inspirações biológicas de ambos os métodos [Duch and Korczak, 1998][Montana, 1995], já que algoritmos genéticos são baseados na evolução natural, enquanto redes neurais artificiais têm motivação no funcionamento do cérebro. Em [Murray, 1994], por exemplo, afirma-se que os algoritmos genéticos proporcionam a abordagem mais natural para a solução do problema de otimização de arquiteturas de redes neurais, porque o cérebro humano também é resultado de evolução biológica. Entretanto, é importante ressaltar que, em termos de otimização de redes neurais, a inspiração biológica não garante que algoritmos genéticos sempre geram os melhores resultados.

Outro aspecto importante que deve ser comentado é que a combinação de algoritmos genéticos com redes neurais já está bastante desenvolvida nos dias de hoje, diferentemente do que ocorre com outras técnicas de otimização global, como simulated annealing e tabu search. Já existem na literatura bons resumos dos trabalhos realizados com algoritmos genéticos para otimizar redes neurais, entre os quais podem ser citados [Branke, 1995] e [Yao, 1999]. Por este motivo, a intenção deste capítulo não é detalhar numerosas abordagens com algoritmos genéticos. O objetivo é comentar as estratégias

de otimização que foram adotadas, fornecendo uma idéia de como está avançada a combinação de algoritmos genéticos com redes neurais.

2.2.1 Otimização de Pesos

O treinamento de uma rede neural geralmente é formulado como uma minimização de uma função de erro, como, por exemplo, o erro médio quadrático entre saídas da rede e saídas desejadas de todos os padrões de treinamento, através de um ajuste iterativo de pesos. Para o treinamento das conhecidas redes Multi-Layer Perceptron (MLP), o algoritmo de aprendizado mais conhecido é o backpropagation [Rumelhart et al., 1986]. Tal algoritmo está enquadrado entre os métodos de *gradiente descendente*, que se utilizam de informações sobre a derivada da função de erro durante o processo de treinamento. Apesar de existirem muitas aplicações eficientes do *backpropagation* para o treinamento de redes MLP, tal algoritmo apresenta, em muitos casos, o grave problema da *convergência local,* ou seja, o estacionamento em mínimos locais da função de erro. Várias abordagens já foram propostas para contornar este problema, como, por exemplo, o uso do conhecido termo de *momentum* [Rumelhart and McClelland, 1986].

Algoritmos de gradiente descendente, como o *backpropagation*, são geralmente considerados como métodos *locais*, pois são concebidos para se aproximar iterativamente do ponto de mínimo a partir de um ponto inicial, fazendo uso de informações sobre o gradiente da função de erro, que são informações locais. Tais informações servem para determinar a direção e a magnitude do ajuste de pesos mais adequado para caminhar em direção ao mínimo.

Em contraste com os métodos de gradiente, as técnicas tratadas neste capítulo são chamadas de *globais*, pois foram originalmente concebidas para realizar uma busca mais geral no espaço, procurando o ponto de mínimo de acordo com um processo que leva em conta aspectos globais da superfície de erro. Estas técnicas, principalmente os algoritmos genéticos, têm sido amplamente utilizadas para melhorar o treinamento de redes neurais, procurando contornar o problema da convergência em mínimos locais. Diferentemente dos algoritmos de treinamento baseados em gradiente descendente, no treinamento com métodos de otimização global, não é necessário que a função de erro

seja contínua, nem diferenciável, pois não existe a necessidade da informação sobre o gradiente [Yao, 1999]. Por esta razão, o treinamento não se restringe a redes com funções de ativação diferenciáveis, podendo ser treinadas, também, redes cujos nodos tenham funções de ativação que apresentam descontinuidades (como, por exemplo, a função sinal ou a função degrau unitário).

Evidentemente, não se pode afirmar que o treinamento com técnicas de otimização global tem desempenho superior aos treinamentos baseados em gradiente descendente, pois esta comparação depende de vários aspectos, como os parâmetros escolhidos para cada técnica e o problema tratado.

Em se tratando dos algoritmos genéticos, as abordagens existentes para a otimização de pesos em redes neurais podem ser classificadas em dois grupos principais: as que utilizam *representação binária* e as que fazem uso da *representação real*. Dependendo do tipo de representação escolhido, devem ser definidos os operadores de reprodução para cruzamento e mutação.

Na representação binária, cada peso da rede neural é codificado como uma seqüência de bits com um comprimento que pode ser fixo ou não. Uma rede neural é representada como uma concatenação de todos os seus pesos no cromossomo. Tal representação tem como vantagem a simplicidade, principalmente no que se refere à aplicação de operadores de reprodução do algoritmo genético (cruzamento e mutação), não havendo a necessidade de definir operadores muito elaborados para lidar com este tipo de cromossomo.

A escolha da representação binária deve ser feita cuidadosamente, pois existe um compromisso entre a precisão da representação e o tamanho dos cromossomos. Se a codificação utiliza poucos bits para representar os pesos, o treinamento pode ser prejudicado, pois os valores reais dos pesos podem não ser representados com precisão suficiente pelos valores discretizados. Por outro lado, se forem utilizados muitos bits para representar os pesos, o tamanho excessivo dos cromossomos pode tornar a evolução bastante ineficiente [Yao, 1999].

Na representação real, cada peso é representado diretamente pelo seu valor real, de modo que cada cromossomo é formado por um vetor de números reais. Dessa forma,

os operadores de cruzamento e mutação binários já não podem ser usados neste tipo de representação. Em [Montana and Davis, 1989], são desenvolvidos vários operadores que incorporam heurísticas a respeito do treinamento de redes neurais. A idéia é preservar o comportamento de extração de características que os nodos de uma rede apresentam ao longo do treinamento.

Diversas aplicações com ambos os tipos de representação são comentadas em [Branke, 1995] e [Yao, 1999].

Independentemente da representação utilizada, é aconselhável que os pesos codificados de um mesmo nodo fiquem juntos em um cromossomo, porque, como os nodos de uma rede neural geralmente se comportam como extratores e detectores de características, a separação de seus pesos de entrada pode levar à destruição deste comportamento quando da aplicação dos operadores de cruzamento [Yao, 1999].

Uma vez definida a representação a ser adotada, é necessário escolher uma função de aptidão, que, geralmente, leva em consideração uma medida de erro para o conjunto de treinamento utilizado. Uma das características favoráveis de usar métodos de otimização global, como algoritmos genéticos, para treinar redes neurais é que funções não-diferenciáveis, como a porcentagem de padrões classificados corretamente (para problemas de classificação), podem ser utilizadas como medidas de aptidão, já que não é necessário computar informações de gradiente [Branke, 1995].

Nos trabalhos de otimização de pesos, uma abordagem que merece destaque é o *treinamento híbrido*, que procura melhorar o desempenho dos algoritmos genéticos da seguinte forma: a rede é treinada com um algoritmo genético e, em seguida, é submetida a um treinamento com um método de gradiente descendente, que realiza um ajuste mais fino dos pesos. O algoritmo genético passa a ter o papel de buscar as melhores regiões do espaço enquanto o método de gradiente fica com a função de identificar o ponto de mínimo destas regiões. Esta abordagem tem sido encontrada nos trabalhos que utilizam algoritmos genéticos combinados com backpropagation [Branke, 1995][Yao, 1999].

2.2.2 Otimização de Arquiteturas

Quando uma rede neural é aplicada para a resolução de um dado problema, a escolha da topologia a ser utilizada é muito importante, pois influencia significativamente o desempenho. Se uma topologia tiver uma quantidade pequena de nodos e conexões, a rede pode não ser capaz de representar e aprender os padrões apresentados, devido à quantidade insuficiente de parâmetros ajustáveis. Por outro lado, se tiver uma quantidade grande de nodos e conexões, a rede pode conter excesso de parâmetros, apresentando dificuldades para generalização quando forem apresentados padrões ainda não vistos.

Em muitos casos, o processo de escolha da arquitetura é feito através de uma seqüência de tentativas com diversas topologias. Portanto, torna-se necessário um método automático para a definição da topologia da rede neural, a fim de evitar este processo demorado e cansativo de tentativas e erros. É esta necessidade que motiva o estudo de métodos de otimização de arquiteturas.

Como já foi explicado, a definição da arquitetura de uma rede neural pode ser formulada como um problema de otimização, em que cada ponto no espaço de busca representa uma arquitetura [Yao, 1999]. Considerando alguma função de custo, que pode levar em conta, por exemplo, o erro de treinamento e o tamanho da rede, os custos de todas as arquiteturas formam uma superfície discreta de busca, de modo que a definição da arquitetura ótima pode ser realizada pela localização do ponto de mínimo global desta superfície.

Assim, o uso de técnicas de otimização global é bastante adequado à resolução deste problema, que tem sido freqüentemente tratado com o uso de algoritmos genéticos [Branke, 1995][Yao, 1999].

Nestas abordagens, a codificação das soluções deve levar em conta alguns aspectos importantes. Primeiramente, o método deve ser capaz de excluir redes inválidas. Além disso, deve ser verificado se os operadores de reprodução, quando aplicados aos indivíduos selecionados, geram redes válidas. Vale ressaltar, ainda, que a representação deve ser capaz de suportar o crescimento em tamanho das redes [Branke, 1995].

Uma dada arquitetura pode ser representada pela especificação de cada nodo e de cada conexão da rede, formando um cromossomo. Esta é a representação *direta* ou *de baixo nível*. Outro tipo de representação, chamada de *indireta* ou *de alto nível*, especifica apenas algumas características da topologia consideradas importantes, como, por exemplo, o número de camadas e de nodos escondidos, de modo que os outros detalhes são determinados na etapa em que a rede é treinada por algum algoritmo [Branke, 1995].

Na representação direta, cada conexão de uma dada arquitetura é especificada individualmente. Dessa forma, uma arquitetura com n unidades pode ser representada por uma matriz n x n, onde cada uma das n linhas simboliza um nodo da topologia, bem como cada uma das n colunas. O elemento na linha i e coluna j representa a conexão que parte do nodo i e entra no nodo j. Por exemplo, pode ser assumido que tal elemento terá valor 1 se a conexão existir, sendo igual a 0 caso a conexão não exista.

Dessa forma, o cromossomo que representa uma arquitetura é formado pela concatenação das linhas da matriz. Para diminuir a quantidade de bits utilizados, podem ser aplicadas restrições na concatenação, dependendo de conhecimento prévio. Por exemplo, em uma arquitetura *feedforward*, sabe-se que os elementos da diagonal principal da matriz, bem como os elementos abaixo da diagonal principal, são todos nulos, pois não existem conexões que partem para nodos de camadas anteriores, nem conexões de um nodo para ele próprio. Então, a concatenação das linhas da matriz pode ser feita usando apenas os elementos acima da diagonal principal, o que traz uma economia na quantidade de bits da representação [Yao, 1999]. Na Figura 2.1, apresenta-se um exemplo desta codificação. Vale notar que a rede possui uma conexão direta entre nodos de camadas não-adjacentes (do nodo 2 para o nodo 5), entretanto este tipo de conexão não traz problemas para a representação, de modo que o algoritmo genético é capaz de lidar bem com redes deste tipo.

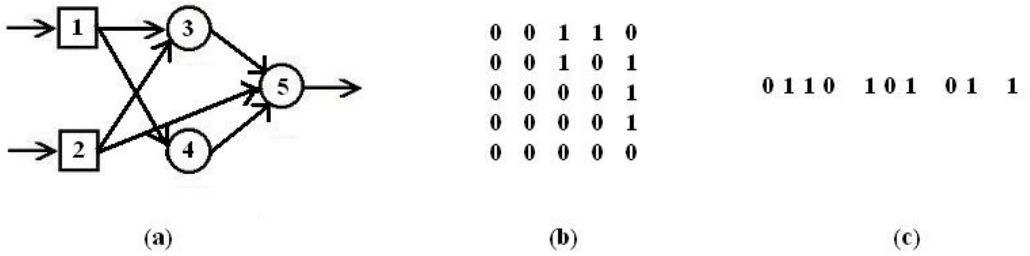


Figura 2.1: (a) Uma rede neural *feedforward*; (b) Matriz das conexões entre as unidades; (c) Representação direta da topologia.

Na Figura 2.2, apresenta-se uma rede neural recorrente. Neste caso, já não pode ser usado o método acima para reduzir a quantidade de bits do cromossomo, o qual deve ser representado pela concatenação das linhas completas da matriz.

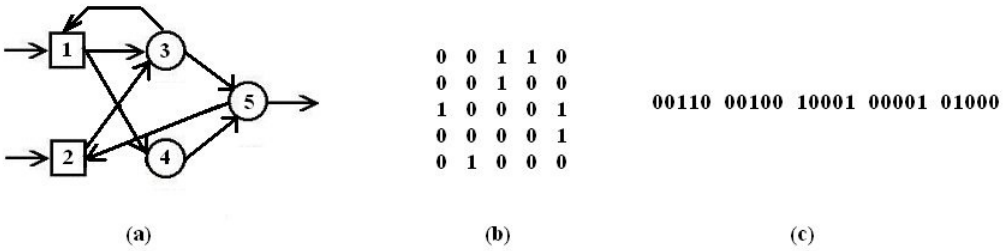


Figura 2.2: (a) Uma rede neural recorrente; (b) Matriz das conexões entre as unidades; (c) Representação direta da topologia.

A representação direta apresenta como vantagem a facilidade de implementação e de conversão entre genótipo e fenótipo. A desvantagem é que esta representação pode tornar o espaço de busca excessivamente amplo, levando à necessidade de um maior número de iterações. Além disso, à medida que as redes crescem em tamanho, os cromossomos podem aumentar explosivamente [Branke, 1995]. Por esse motivo, em geral, a máxima topologia é definida pelo usuário, limitando o crescimento das redes e permitindo uma maior exploração no espaço de busca definido. Por outro lado, esta restrição pode excluir do espaço de busca as melhores redes para o problema abordado.

Uma alternativa para contornar estas limitações é o uso da representação indireta, que codifica apenas algumas características das arquiteturas, utilizando descrições abstratas ou, ainda, gramáticas. Este tipo de representação é capaz de gerar uma especificação mais compacta das topologias de redes neurais, mas, em alguns casos, pode não conseguir especificar uma rede com boa capacidade de generalização [Yao, 1999].

Arquiteturas de redes neurais podem ser especificadas por diversos parâmetros, como o número de camadas escondidas, o número de nodos escondidos em cada camada, o número de conexões entre duas camadas, entre outros. Estes parâmetros podem ser codificados de várias formas em um cromossomo. Por exemplo, em [Harp et al., 1989], um cromossomo contém um ou mais segmentos, cada um deles representando uma camada e seu padrão de conectividade eferente (projeções para outras camadas). Cada segmento é dividido em duas partes: uma contendo informações sobre a camada, como o número de nodos, e outra com especificações das conexões da camada que se projetam para as demais. Além disso, o genótipo também possui informação sobre os parâmetros de aprendizado, que evoluem em conjunto com a topologia, de modo que as interações entre estes parâmetros e as arquiteturas podem ser exploradas ao longo da evolução.

Outra abordagem interessante pode ser encontrada em [Kitano, 1990], na qual as topologias são codificadas através de regras, em uma gramática determinística livre de contexto.

É válido lembrar que a representação indireta pode reduzir bastante o tamanho dos cromossomos, mas a otimização fica restrita a um subconjunto do espaço de possíveis arquiteturas. Por exemplo, se for codificado apenas o número de nodos escondidos, assume-se que a rede é feedforward e tem apenas uma camada escondida. Além disso, admite-se que duas camadas adjacentes são completamente conectadas. Dessa forma, esta representação é mais adequada aos problemas em que o tipo de arquitetura procurada já é conhecido [Yao, 1999].

Independentemente da representação escolhida, quando o método otimiza apenas a arquitetura, a avaliação de uma topologia contém ruído, já que um genótipo sem

nenhuma informação sobre os pesos da rede é aproximado por um fenótipo contendo uma rede treinada [Yao and Liu, 1997]. Dependendo dos pesos iniciais escolhidos e dos parâmetros de treinamento, a avaliação de um mesmo genótipo pode gerar resultados diferentes. Para evitar este problema, a avaliação de cada arquitetura pode ser feita através de várias inicializações de pesos, para que seja computada a média dos resultados obtidos, o que aumenta dramaticamente o tempo de execução.

O cálculo da aptidão de um genótipo que representa uma arquitetura pode levar em conta diversos fatores, tais como o erro obtido para o conjunto de treinamento, o número de épocas utilizadas no treinamento e o tamanho da topologia (quantidade de nodos e conexões). Existem abordagens que incorporam heurísticas na avaliação do desempenho. Por exemplo, em [Whitley and Bogart, 1990], as redes de menor arquitetura contam com maior tempo de treinamento. Dessa forma, as redes menores são favorecidas, mas a recompensa só é significativa quando apresentam desempenho satisfatório, aproveitando bem o maior número de iterações permitidas.

É importante lembrar que as funções de ativação dos nodos também fazem parte da arquitetura, exercendo influência significativa sobre seu desempenho. Na maioria das abordagens que otimizam redes neurais, assume-se que a função de ativação é fixa e igual para todos os nodos da rede, mas já existem abordagens que representam funções de ativação nos pontos do espaço de busca, para que possam ser otimizadas em conjunto com os demais aspectos da topologia da rede. Por exemplo, em [Stork et al., 1990], as funções de transferência são especificadas no genótipo, sendo muito mais complexas do que a função sigmóide. Em [White and Ligomenides, 1993], a representação é mais simples: em vez de codificar as funções de ativação, representam-se a porcentagem dos nodos que utilizam função sigmóide e a porcentagem dos nodos que fazem uso da função gaussiana. Apenas estas duas funções são permitidas, e elas não evoluem durante o processo. O objetivo, então, é determinar a mistura ótima destas duas funções entre o conjunto de nodos da rede.

2.2.3 Otimização Simultânea de Arquitetura e Pesos

Conforme foi comentado anteriormente, quando é otimizada somente a arquitetura, a avaliação de uma dada topologia contém ruído, já que uma rede treinada é usada para

calcular o custo da solução, sendo que a solução representa uma topologia de rede neural sem informação de pesos [Yao and Liu, 1997].

Uma das maneiras mais adequadas para reduzir os efeitos deste ruído é a otimização simultânea de arquiteturas e pesos. Dessa forma, uma rede é especificada não só pela sua topologia, mas também pelo seu conjunto completo de pesos, tornando mais precisa a avaliação de seu desempenho.

As considerações feitas anteriormente sobre a representação das redes também são válidas neste caso. Uma abordagem interessante é encontrada em [Koza and Rice, 1991]. Neste caso, a representação utilizada apresenta duas partes: a primeira contém a representação indireta do padrão de conectividade, e a segunda contém a representação direta dos pesos. A primeira parte influi na ativação da segunda, de modo que, se uma dada conexão é considerada inexistente, seu peso codificado, apesar de permanecer representado no cromossomo, não é utilizado.

Sobre os operadores de reprodução, também são válidos os comentários já apresentados. Os operadores escolhidos podem variar bastante, dependendo do problema abordado e da finalidade da otimização. Por exemplo, em [Braun and Weisbrod, 1993], os operadores de cruzamento agem da seguinte forma: se a conexão existir nos dois pais, é transmitida para os filhos; caso a conexão exista em apenas um dos pais, é passada para a nova geração com uma dada probabilidade especificada pelo usuário. Uma certa fração do peso da conexão é transmitida para a próxima geração, sendo esta taxa definida pelo usuário.

Comentários sobre outros trabalhos de otimização simultânea de arquitetura e pesos usando algoritmos genéticos podem ser encontrados em [Branke, 1995] e [Yao, 1999].

2.2.4 Outras Abordagens

Diversas outras propostas usando algoritmos genéticos para otimizar redes neurais já foram formuladas, mostrando diferentes maneiras de implementar otimização global em variados aspectos das redes neurais [Yao, 1999].

Entre estas propostas, merece destaque o trabalho de E. G. M. de Lacerda [Lacerda, 2003][Lacerda et al., 2002], que estudou a aplicação de algoritmos genéticos à otimização de redes neurais do tipo função de base radial (RBF) [Moody and Darken, 1989]. Entre outras contribuições, é importante ressaltar o estudo das técnicas de *crossvalidation* e *bootstrap* [Kohavi, 1995] quando aplicadas em um algoritmo genético destinado a treinar redes RBF. Sabe-se que, no treinamento destas redes, uma das etapas cruciais é a escolha dos centros das funções de base (mais detalhes sobre este tipo de rede podem ser encontrados em [Moody and Darken, 1989]). Em [Lacerda et al., 2002], esta etapa de treinamento é formulada como um problema de otimização a ser resolvido com um algoritmo genético, de modo que o objetivo é encontrar o subconjunto de padrões de treinamento que, quando utilizado como conjunto de centros da rede RBF, minimiza o erro. As técnicas de *crossvalidation* e *bootstrap*, que são abordagens distintas para definir os conjuntos de treinamento e teste [Kohavi, 1995], são empregadas na função de aptidão dos cromossomos. Experimentos realizados para um problema de aproximação polinomial mostraram que o algoritmo genético consegue explorar melhor o espaço de busca quando se usam *crossvalidation* e *bootstrap* na função de aptidão em relação à abordagem mais tradicional, que é a simples divisão do conjunto de dados em dois subconjuntos (treinamento e teste).

Técnicas de otimização global também podem ser usadas para selecionar atributos de entrada nos padrões da base de dados. Esta aplicação é importante quando o problema tem uma quantidade muito grande de atributos, já que um número muito grande de nodos de entrada pode aumentar dramaticamente o tamanho da rede neural, dificultando o treinamento da mesma. Em geral, o problema é abordado pelo uso de pré-processamento com técnicas que procuram reduzir a dimensionalidade da base de dados, mas também podem ser usadas técnicas de otimização global, como algoritmos genéticos, para atingir este objetivo, como se comenta em [Yao, 1999].

A procura de um conjunto ótimo de atributos para uma rede neural pode ser formulada como um problema de otimização, da seguinte forma: dado o conjunto de possíveis atributos, deseja-se encontrar um subconjunto que contenha o mínimo número de atributos, de modo que o desempenho da rede não seja pior do que o obtido quando se usam todos os atributos [Yao, 1999]. Isto pode ser implementado codificando as

soluções como cromossomos binários com tamanho igual à quantidade de atributos possíveis. Cada bit do cromossomo representa um dos atributos, sendo que o valor 1 indica que o atributo está presente na solução, enquanto o valor 0 indica o contrário. A aptidão de um cromossomo é obtida pelo treinamento da rede neural com o subconjunto de atributos representado pelo indivíduo.

Outra abordagem interessante é a otimização do aprendizado das redes neurais, pois um algoritmo de treinamento pode ter diferentes comportamentos e desempenhos, dependendo da arquitetura à qual é aplicado. Dessa forma, é interessante incluir características do processo de aprendizado na evolução realizada pelo algoritmo genético. Este tipo de abordagem ainda é recente, tendo por finalidade gerar métodos automáticos de otimização do aprendizado em redes neurais [Yao, 1999]. Em geral, as abordagens existentes podem ser divididas em duas categorias: otimização dos parâmetros de um algoritmo de treinamento conhecido e otimização das regras de aprendizado.

Em relação à otimização de parâmetros de treinamento, existem muitas abordagens, principalmente no que se refere aos parâmetros do algoritmo backpropagation, como taxa de aprendizado e termo *momentum*. Por exemplo, em [Harp et al., 1989], os parâmetros do backpropagation são codificados no cromossomo juntamente com a arquitetura da rede, para que as interações entre o aprendizado e a topologia sejam mais exploradas, de modo a ser identificada uma combinação ótima entre ambos.

Outras abordagens procuram otimizar as regras de aprendizado, tendo em vista que o desempenho de uma regra de ajuste de pesos depende da arquitetura aplicada. Daí a necessidade de um processo automático de otimização da regra de aprendizado de acordo com a arquitetura da rede.

Levando em consideração que a regra de aprendizado diz respeito ao comportamento dinâmico de uma rede neural, deve ser escolhido um tipo de representação que consiga especificar bem as regras em um cromossomo. Dada a dificuldade de elaborar uma representação universal para todos os tipos de regras, é necessário assumir certas restrições. Em geral, assume-se que a adaptação dos pesos

depende apenas de informações locais, como as ativações e os pesos atuais do nodo, e que a regra de aprendizado é a mesma para todas as unidades da rede [Yao, 1999]. A regra de aprendizado é descrita como uma função linear das variáveis locais e de seus produtos, que pode ser expressa da seguinte forma [Yao, 1995]:

$$\Delta w(t) = \sum_{k=1}^{n} \sum_{i_i, i_2, \ldots, i_k = 1}^{n} (\theta_{i_1 i_2 \ldots i_k} \prod_{j=1}^{k} x_{i_j}(t-1)), \quad (1)$$

onde t é o tempo, Δw é o ajuste do peso, $x_1, x_2, \ldots, x_n$ são as variáveis locais, e θ são os coeficientes reais que serão determinados pela evolução. Sendo assim, a evolução das regras de aprendizado corresponde à otimização dos vetores reais formados pelos valores de θ. Devido à grande quantidade de termos envolvidos na expressão citada, apenas alguns deles são usados na prática, de acordo com algum conhecimento heurístico prévio.

Por exemplo, em [Chalmers, 1990], a regra de aprendizado é definida usando-se quatro variáveis locais e seus seis produtos dois a dois, de modo que nenhum termo com ordem igual ou superior a três é utilizado. Neste trabalho, a arquitetura é fixa, sendo usada apenas uma camada de saída para a rede neural. O processo evolutivo gerou como resultado a conhecida *regra delta* [Widrow and Hoff, 1960] e algumas de suas variações.

2.3 Simulated Annealing

O método de simulated annealing [Kirkpatrick et al., 1983] é inspirado nos processos de esfriamento de sólidos que alcançam energia mínima, correspondente a uma estrutura cristalina perfeita, se esfriados de forma suficientemente lenta. O processo físico de *annealing* consiste no aquecimento de um sólido e no posterior esfriamento de forma gradual. Os átomos do material possuem alta energia em temperaturas elevadas, possuindo mais liberdade para organizarem arranjos. À medida em que a temperatura é reduzida, as energias atômicas diminuem, até que se obtém um cristal com estrutura regular no estado em que o sistema tem energia mínima. Se o esfriamento for muito rápido, a estrutura cristalina apresenta amplas irregularidades e defeitos, pois o sistema não atinge o estado de energia mínima [Pham and Karaboga, 2000].

Em uma dada temperatura, a distribuição de probabilidade das energias do sistema é determinada pela probabilidade de Boltzmann:

$$P(E) \propto e^{[-E/(kT)]}, \tag{2}$$

onde E é a energia do sistema, k é a constante de Boltzmann, T é a temperatura, e $P(E)$ é a probabilidade de que o sistema esteja em um estado com energia E.

De acordo com (2), em temperaturas altas, $P(E)$ tende a 1 para todos os estados de energia. Pode ser visto, também, que existe uma probabilidade pequena de o sistema estar em um estado de energia alta se a temperatura estiver baixa. Por esta razão, esta distribuição de probabilidades permite que o sistema escape de mínimos locais de energia [Pham and Karaboga, 2000].

Na resolução de um problema de otimização por simulated annealing, os estados do sólido representam as soluções possíveis do problema, as energias dos estados correspondem aos custos das soluções, e o estado de mínima energia corresponde à solução ótima do problema. O método funciona com uma seqüência de iterações, sendo que, a cada iteração, a solução atual é modificada aleatoriamente, para ser criada uma nova solução. Em seguida, é computado o custo da solução gerada, e a variação no custo é utilizada para decidir se a mesma será aceita ou não. Se o custo da solução nova for menor que o custo da solução atual, a solução nova é aceita; caso contrário, a solução nova pode ser aceita ou não, dependendo do *critério de Metropolis* [Metropolis et al., 1953], baseado na probabilidade de Boltzmann. De acordo com este critério, é gerado um número aleatório δ no intervalo [0,1] a partir de uma distribuição uniforme. Se $\delta \leq e^{(-\Delta E/T)}$, onde ΔE é a variação no custo e T é um parâmetro chamado de *temperatura*, então a nova solução gerada é aceita como solução atual. Caso contrário, a solução nova não é aceita, e o processo continua a partir da solução atual.

A escolha de um *esquema de esfriamento*, ou seja, de uma regra para diminuir o parâmetro de temperatura, é muito importante na implementação de simulated annealing, que deve especificar a *temperatura inicial* e uma *regra de atualização* da temperatura. Diversos esquemas já foram propostos na literatura, sendo amplamente utilizados os *esquemas geométricos*, em que a temperatura nova é dada pela temperatura

atual multiplicada por um *fator de redução* (uma constante menor do que 1, sendo próxima de 1) [Pham and Karaboga, 2000].

Considerando um conjunto S de soluções e uma função de custo real f, o algoritmo de simulated annealing procura o mínimo global s, tal que $f(s) \leq f(s')$, $\forall s' \in S$. O processo termina após I iterações, e o esquema de esfriamento atualiza a temperatura T_i da iteração *i*. O parâmetro I deve ser escolhido pelo usuário, de acordo com o problema. A estrutura básica do algoritmo de simulated annealing é apresentada no Algoritmo 2.1 [Boese and Kahng, 1993]:

---

1. $s_0 \leftarrow$ solução inicial em S
2. Para $i = 0$ até $I - 1$
3. Gera solução nova s'
4. Se $f(s') < f(s_i)$
5. $s_{i+1} \leftarrow s'$
6. senão
7. $s_{i+1} \leftarrow s'$ com probabilidade $e^{-[f(s') - f(s_i)]/T_{i+1}}$
8. Atualiza temperatura T_i
9. Retorna s_I.

Algoritmo 2.1: Simulated annealing.

O método de simulated annealing já foi utilizado com sucesso em diversos problemas de otimização global, como se observa em [Corana et al., 1987][Goffe et al., 1994][Sexton et al., 1999]. Entretanto, o uso de simulated annealing para otimizar redes neurais tem sido bem menos freqüente do que a utilização de algoritmos genéticos para o mesmo objetivo, mas alguns trabalhos interessantes podem ser comentados.

Em [Boese and Kahng, 1993][Boese et al., 1995], simulated annealing foi aplicado no treinamento de redes neurais em uma abordagem que retorna, ao final da execução, a melhor solução encontrada durante o processo de otimização, e não a última solução visitada, como ocorre tradicionalmente nas implementações de simulated annealing, de acordo com a estrutura básica do algoritmo apresentada acima. Foi abordado um problema *benchmark* de classificação, que consiste na identificação de objetos no subsolo através de respostas de radar. Os pesos de uma topologia fixa com

quatro nodos escondidos foram codificados como vetores de números reais para serem otimizados pelo algoritmo de simulated annealing.

Em [Porto et al., 1995], simulated annealing e backpropagation foram implementados para o treinamento de uma topologia fixa MLP com duas camadas escondidas, cada uma contendo quatro nodos. O problema abordado foi o reconhecimento de respostas de sonar, com o objetivo de diferenciar as respostas provenientes de objetos metálicos artificiais das respostas oriundas de obstáculos naturais, como rochas e massas de areia. Os resultados mostraram que simulated annealing obteve melhor desempenho do que o algoritmo backpropagation, que freqüentemente ficou preso em mínimos locais.

Em [Stepniewski and Keane, 1997], simulated annealing e algoritmos genéticos foram utilizados para otimizar arquiteturas de redes MLP. Foi usada a representação direta das arquiteturas, ou seja, através de uma seqüência de bits, cujos valores indicam se a conexão está presente ou não na topologia.

O problema abordado foi a identificação de um sistema não-linear, que é descrito por [Su and Sheen, 1992]:

$$y(k) = 2.5\,y(k-1)\sin(\pi \exp(-u^2(k-1) - y^2(k-1))) + u(k-1)[1+u^2(k-1)]. \quad (3)$$

O sistema foi excitado configurando a entrada u como um sinal aleatório de média zero uniformemente distribuído em [-2.0, +2.0].

Os resultados mostraram que simulated annealing e algoritmos genéticos foram capazes de encontrar topologias de redes MLP com boa capacidade de generalização para o problema abordado.

Em [Sexton et al., 1999], simulated annealing e algoritmos genéticos foram empregados para treinar uma topologia fixa com seis nodos escondidos. As soluções no espaço de busca foram representadas por vetores de números reais contendo todos os pesos da rede.

Foram abordados os seis seguintes problemas:

$$y = x_1 + x_2, \tag{4}$$

$$y = x_1 x_2, \tag{5}$$

$$y = \frac{x_1}{|x_2| + 1}, \tag{6}$$

$$y = x_1^2 - x_2^3, \tag{7}$$

$$y = x_1^3 - x_1^2, \tag{8}$$

$$y(t) = y(t-1) + 10.5\left[\frac{0.2\,y(t-5)}{1 + y^{10}(t-5)} - 0.1\,y(t-1)\right]. \tag{9}$$

Para os problemas (4), (5), (6), (7) e (8), foram usadas 50 observações para o conjunto de treinamento. A base de dados foi construída gerando entradas aleatoriamente a partir dos conjuntos $x_1 \in [-100,+100]$ e $x_2 \in [-10,+10]$. Para testar as redes, foram construídos dois conjuntos de dados, cada um contendo 150 observações. O primeiro conjunto foi construído para testar a capacidade de interpolação da rede, sendo gerado a partir dos mesmos conjuntos $x_1 \in [-100,+100]$ e $x_2 \in [-10,+10]$, mas sem incluir observações em comum com o conjunto de treinamento. O segundo conjunto foi construído para testar a capacidade de extrapolação, sendo que a primeira entrada foi gerada a partir dos conjuntos $x_1 \in [-200,-101]$ e $x_1 \in [+101,+200]$, enquanto a segunda entrada foi gerada a partir de $x_2 \in [-20,-11]$ e $x_2 \in [+11,+20]$.

O sexto problema é uma versão discreta da equação de Mackey-Glass, que já foi anteriormente usada na literatura de redes neurais [Goffe et al., 1994]. Esta série caótica é interessante por sua similaridade com as séries temporais encontradas em mercados financeiros. Cinco valores consecutivos da variável dependente foram usados como entradas. O conjunto de treinamento formado por 100 observações foi iniciado pelo ponto (1.6, 0, 0, 0, 0). Foi gerado um conjunto de teste para interpolação a partir de um ponto escolhido aleatoriamente.

Os resultados mostraram que, na maioria dos casos, algoritmos genéticos obtiveram desempenhos melhores do que os obtidos por simulated annealing. Foram abordados mais dois problemas do mundo real. O primeiro foi um problema *benchmark* de classificação (predição de câncer de mama), e o segundo foi uma previsão de série temporal financeira. Novamente, foram treinados os pesos de uma topologia fixa com seis nodos escondidos, e, mais uma vez, os melhores resultados foram obtidos pelos algoritmos genéticos.

Em [Chalup and Maire, 1999], algoritmos do tipo *hill climbing* (incluindo simulated annealing) foram empregados para o treinamento de redes neurais. Os métodos foram usados para treinar uma topologia MLP completamente conectada com dois nodos escondidos. A base de dados foi formada pelos 32 padrões do problema da paridade com 5 bits, que é um problema clássico, em que a rede deve classificar os vetores binários de entrada em duas categorias: vetores com quantidade par de bits iguais a 1 e vetores com quantidade ímpar de bits iguais a 1. Um algoritmo de hill climbing que utiliza busca *in-line* foi proposto e apresentou melhor desempenho do que simulated annealing e hill climbing padrão.

2.4 Tabu Search

Tabu search é um algoritmo de busca iterativa caracterizado pelo uso de uma memória flexível. Neste método, cada iteração consiste em avaliar uma certa quantidade de soluções novas. A melhor solução nova (em termos da função de custo) é aceita, mesmo se seu custo for maior do que o custo da solução atual. Dessa forma, o algoritmo escolhe a solução nova que produz a maior melhoria ou a menor deterioração na função de custo, e isto permite que o método escape de mínimos locais. É utilizada uma *lista tabu* para armazenar uma certa quantidade de soluções mais recentemente visitadas, as quais são classificadas como proibidas em iterações posteriores. Esta estratégia é necessária, pois pode haver um retorno para soluções já visitadas anteriormente, uma vez que o método aceita a melhor solução nova, independentemente de melhorar ou piorar o custo. Por este motivo, devem ser evitados eventuais ciclos na trajetória, o que é possível graças ao uso da lista tabu, que pode proibir um movimento, classificando-o como "tabu".

Idealmente, a lista tabu deveria armazenar todas as soluções previamente visitadas e deveria ser verificada por completo antes de qualquer novo movimento. Esta abordagem, porém, exige muita memória e esforço computacional. Por outro lado, se for armazenada uma única solução anterior, o problema dos ciclos na trajetória não será resolvido. A abordagem mais comumente utilizada é o armazenamento de uma certa quantidade T de soluções mais recentemente visitadas. Neste caso, o parâmetro T, chamado de *comprimento* ou *tamanho da lista tabu* é um parâmetro que deve ser cuidadosamente escolhido, pois, se for muito pequeno, a probabilidade de haver ciclos se torna grande, e, se for muito grande, a busca pode sair de regiões promissoras antes que as mesmas sejam satisfatoriamente exploradas [Pham and Karaboga, 2000].

Dessa forma, a *lista tabu* registra as T últimas soluções visitadas. Quando a lista fica cheia, um novo movimento é registrado em substituição ao movimento mais antigo guardado na lista. Dessa forma, a lista funciona como uma memória *first-in-first-out* (FIFO).

Considerando um conjunto S de soluções e uma função real de custo f, o algoritmo de tabu search procura o mínimo global s, tal que $f(s) \leq f(s')$, $\forall s' \in S$. O processo termina após I iterações e retorna a melhor solução encontrada durante a execução s_{BSF} (*best so far*). O parâmetro I deve ser escolhido pelo usuário, de acordo com o problema. A estrutura básica do algoritmo de tabu search é apresentada no Algoritmo 2.2.

1. $s_0 \leftarrow$ solução inicial em S
2. Atualiza s_{BSF} com s_0 (melhor solução encontrada até o momento)
3. Insere s_0 na lista tabu
4. Para $i = 0$ até $I - 1$
5. Gera um conjunto V de soluções novas
6. Escolhe a melhor solução s' do conjunto V que não está na lista tabu
7. $s_{i+1} \leftarrow s'$
8. Atualiza lista tabu (insere s_{i+1} e, se a lista estiver cheia, apaga a solução mais antiga)
9. Atualiza s_{BSF} (se $f(s_{i+1}) < f(s_{BSF})$)
10. Retorna s_{BSF}

Algoritmo 2.2: Tabu search.

O algoritmo de tabu search já foi empregado em diversos problemas de otimização combinatorial [Knox, 1989][Skorin-Kapov, 1990][Bland and Dawson, 1991][Battiti and Tecchiolli, 1995][Hertz et al., 1995][Sexton et al., 1998], sendo que também existem algumas propostas para problemas contínuos [Bland, 1993][Bland, 1994]. Porém, assim como acontece com simulated annealing, o uso de tabu search para otimizar redes neurais tem sido bem menos freqüente do que a utilização de algoritmos genéticos. Mesmo assim, alguns trabalhos interessantes foram propostos.

Uma versão modificada de tabu search, chamada de *reactive tabu search* (RTS), foi apresentada e testada em [Battiti and Tecchiolli, 1995]. A abordagem foi implementada para treinar topologias MLP fixas para três problemas de classificação. O primeiro foi o problema do "ou-exclusivo" (*XOR*), no qual uma rede MLP com dois nodos escondidos foi treinada para classificar as entradas. A segunda base de dados foi um problema *benchmark* do mundo real relacionado com a discriminação de respostas de sonar. O terceiro problema considerado foi uma tarefa derivada de uma aplicação real na área de Física Experimental, em que uma rede MLP foi utilizada para discriminar padrões derivados de colisões no *Large Electron-Positron* (LEP) *collider* em duas classes: ruído de *background* e evento potencialmente interessante. Para o treinamento de redes neurais recorrentes, foi considerado um problema adicional: uma versão do problema *truck and trailer backup*, que é uma aplicação de controle não-

linear. Uma rede completamente recorrente foi treinada para definir o movimento de um caminhão quando inicializado aleatoriamente em uma dada região do espaço.

Nesta abordagem para o treinamento de redes neurais, cada peso de conexão é descrito por um vetor binário usando o *código de Gray*, que tem a propriedade de que os inteiros $n - 1$ e $n + 1$ são obtidos invertendo um único bit do código de n (os códigos de $n - 1$ e $n + 1$ possuem distância de Hamming igual a 1 em relação ao código de n). A abordagem RTS apresentou desempenho satisfatório para todos os problemas tratados.

Em [Karaboga and Kalinli, 1997], foi proposto um novo modelo de tabu search, que faz uso de paralelismo. Nesta abordagem, um conjunto de algoritmos de tabu search padrão é executado simultaneamente. O método proposto foi usado para treinar uma rede neural recorrente com o objetivo de identificar sistemas dinâmicos. A rede teve topologia fixa e foi treinada pelo algoritmo proposto (tabu search paralelo), pelo tabu search padrão e pelo backpropagation.

Foram abordados dois problemas de identificação de sistemas dinâmicos, sendo um linear e outro não-linear. O sistema linear foi descrito pela equação discreta de terceira ordem:

$$y(k) = 2.627771\ y(k-1) - 2.333261\ y(k-2) + 0.697676\ y(k-3) + 0.017203\ u(k-1) - 0.030862\ u(k-2) + 0.014086\ u(k-3). \tag{10}$$

O sistema não-linear é o que descreve o movimento oscilatório de um pêndulo simples em ângulos pequenos:

$$y(k) = 1.04\ y(k-1) - 0.824\ y(k-2) + 0.130667\ y^3(k-2) - 0.16\ u(k-2). \tag{11}$$

O sinal de entrada para ambos os sistemas, $u(k)$, $k = 0, 1, \ldots, 99$, foi gerado aleatoriamente em [-1.0,+1.0]. Os resultados mostraram que o algoritmo proposto apresentou melhor desempenho do que o tabu search padrão e o backpropagation para a identificação de ambos os sistemas.

Em [Sexton et al., 1998], tabu search foi aplicado para resolver os problemas descritos nas equações (4), (5), (6), (7), (8) e (9), que já foram explicados na seção

anterior. Novamente, foram treinados os pesos de uma topologia fixa com seis nodos escondidos, sendo que as soluções foram representadas por vetores reais contendo todos os pesos da rede. Tabu search gerou melhores resultados do que os obtidos pelo algoritmo backpropagation, tanto para os dados de treinamento, quanto para os dados de teste (interpolação e extrapolação).

Em [Cannas et al., 1999], Tabu Search é usado para otimizar arquiteturas de redes neurais para previsão de séries temporais. Foram utilizadas redes neurais recorrentes que fazem uso de *delays*, ou seja, o processamento temporal é feito através do uso de informações sobre entradas e saídas anteriores no tempo.

A definição da estrutura da rede, ou seja, do número de nodos escondidos e do número de *delays*, foi formulada como um problema de otimização de variáveis inteiras (representação indireta de topologias). Dessa forma, foi possível aplicar tabu search para abordar o problema. Foram otimizadas 5 variáveis, que podiam assumir valores inteiros entre 1 e 20. A função de custo a ser minimizada foi o erro em um conjunto de validação, com o intuito de gerar redes com boa capacidade de generalização.

O problema abordado foi o aprendizado do comportamento autônomo de um circuito caótico não-linear, que é conhecido como *circuito de Chua* [Chua, 1993], sendo considerado um paradigma para o estudo do caos. É um dos sistemas mais simples para os quais a presença do caos foi estabelecida experimentalmente, além de ter sido confirmada numericamente e provada matematicamente. Mais detalhes sobre o problema podem ser encontrados em [Chua, 1993][Cannas et al., 1999].

O uso de tabu search permitiu desenvolver um processo automático para definir as arquiteturas ótimas de redes neurais, permitindo uma convergência mais rápida e estável dos algoritmos usados para treinar as topologias, sem ser necessário muito tempo para o ajuste de parâmetros de treinamento [Cannas et al., 1999]. Resultados similares tinham sido obtidos anteriormente em [Cannas et al., 1998], porém a definição manual das arquiteturas consumia bastante tempo e gerava empiricamente redes com grande dimensionalidade.

2.5 Comentários Finais

Neste capítulo, foram comentadas aplicações envolvendo o uso das mais tradicionais técnicas de busca global (algoritmos genéticos, simulated annealing e tabu search) para a otimização de redes neurais artificiais. O processo de otimização pode ter como objetivo melhorar os mais variados aspectos das redes neurais, como ajuste de pesos, definição de topologias e, até mesmo, otimização do aprendizado. Conforme foi comentado, o uso de algoritmos genéticos está bem mais desenvolvido do que a aplicação de outras técnicas de otimização global. Este fato, em vez de desestimular o estudo de outros métodos globais, como simulated annealing e tabu search, deve servir de motivação para que sejam propostas abordagens cada vez mais interessantes e originais. Como pode ser visto em [Duch and Korczak, 1998], muitas possibilidades estão abertas para o uso de diversas outras técnicas de otimização global que já estão sendo aplicadas em muitas áreas e que nunca foram implementadas para otimizar redes neurais.

Diversas variações das técnicas abordadas neste capítulo já foram propostas [Duch and Korczak, 1998] e não foram descritas neste trabalho, pois, conforme se comentou anteriormente, o objetivo não foi elaborar um resumo para descrever todas as técnicas existentes de otimização global, mas sim fornecer uma noção geral de como está sendo desenvolvida a aplicação destes métodos para a otimização de redes neurais. Resumos desta natureza já existem para algoritmos genéticos, como, por exemplo, [Branke, 1995], [Yao, 1999], [Lacerda, 2003] e [Lacerda et al., 2002], mas não se encontram com facilidade trabalhos resumindo aplicações em otimização de redes neurais usando simulated annealing e tabu search.

Conforme foi visto neste capítulo, as abordagens existentes na literatura para otimizar redes neurais com simulated annealing e tabu search normalmente tratam do ajuste de pesos em topologias fixas ou da otimização apenas das arquiteturas de redes neurais, não existindo, até o presente momento, abordagens que procuram otimizar simultaneamente a arquitetura e os pesos, como já existe na área de algoritmos genéticos.

Além disso, não é comum encontrar abordagens que combinem simulated annealing e tabu search com métodos de gradiente, como o algoritmo backpropagation, em abordagens de treinamento híbrido, que já existem na área de algoritmos genéticos.

Capítulo 3

Metodologia para a Otimização Simultânea de Arquiteturas e Pesos de Redes MLP

3.1 Introdução

Como foi comentado anteriormente, este trabalho propõe uma metodologia para a otimização simultânea de arquiteturas e pesos em redes MLP. Segundo esta abordagem, cada ponto do espaço de busca codifica tanto a estrutura topológica da rede como os valores de seus pesos, tornando a avaliação do custo mais precisa que a obtida representando apenas a arquitetura. O objetivo deste capítulo é detalhar a metodologia proposta, que combina características de simulated annealing e tabu search, além de utilizar um algoritmo local de treinamento, como o conhecido backpropagation, com o

intuito de gerar redes com poucas conexões e boa capacidade de generalização para um conjunto de dados qualquer.

3.2 Metodologia de Otimização Proposta

De acordo com o que foi explicado no Capítulo 2, percebe-se que o método de simulated annealing possui a habilidade de poder escapar de mínimos locais durante o processo de otimização, por conta da escolha entre aceitar ou não uma nova solução que piora o custo. Esta escolha é baseada no cálculo de uma probabilidade, que depende do aumento no custo e de um parâmetro do algoritmo (temperatura). Entretanto, em muitos casos, o método pode apresentar uma certa lentidão na convergência para soluções aceitáveis, dependendo da escolha do esquema de esfriamento (redução da temperatura). Se a temperatura for reduzida de forma muito brusca ao longo das iterações, pode não acontecer uma exploração satisfatória em diversas regiões do espaço de busca, já que, em temperaturas muito baixas, diminui muito a probabilidade de aceitar soluções que pioram o custo. Por outro lado, se a temperatura for reduzida de forma muito suave, a convergência pode se tornar excessivamente lenta, sendo necessária uma quantidade muito grande de iterações.

O método de tabu search, por sua vez, avalia um conjunto de soluções novas a cada iteração (em vez de uma única solução, como acontece em simulated annealing). Este fato torna o algoritmo de tabu search mais rápido, de modo a necessitar, em geral, de menos iterações do que simulated annealing para convergir. A habilidade de tabu search para escapar de mínimos locais está justamente na geração de um conjunto de soluções novas a cada iteração e na posterior escolha da melhor solução gerada, entretanto este processo exige o armazenamento de uma certa quantidade de soluções mais recentemente visitadas em uma lista, para evitar ciclos na trajetória de busca. A implementação desta lista de soluções proibidas (“tabu”) exige bastante memória e tempo de processamento em cada iteração, pois cada solução nova deve ser comparada com as soluções armazenadas na lista, a fim de verificar se está proibida ou não.

Estas observações motivaram a proposta de uma técnica de otimização que combina as principais características favoráveis de simulated annealing e tabu search, procurando evitar suas limitações. Em linhas gerais, o método funciona da seguinte

forma: em cada iteração, é gerado um conjunto de soluções novas a partir da solução atual. Cada uma tem seu custo avaliado, e a melhor solução é escolhida, assim como acontece em tabu search. No entanto, esta solução nem sempre é aceita, diferentemente do que ocorre em tabu search. O critério de aceitação é o mesmo utilizado na técnica de simulated annealing: se esta solução escolhida tiver custo menor do que a solução atual, é aceita; em caso contrário, pode ser aceita ou não, dependendo do cálculo de uma probabilidade, que é dada pela mesma expressão utilizada no método de simulated annealing (controlada pelo parâmetro de temperatura). Durante o processo de otimização, armazena-se apenas a melhor solução encontrada, que é a solução final retornada pelo método.

O armazenamento da melhor solução encontrada durante o processo de otimização é um dos aspectos que contribuem para os bons resultados obtidos por tabu search. Como foi explicado no Capítulo 2, na estrutura básica de simulated annealing, a solução final retornada pelo algoritmo é a última solução visitada, e não a melhor solução encontrada no processo. Entretanto, considerando a aplicação prática, é mais aconselhável guardar sempre a melhor solução encontrada durante a execução. Na literatura, podem ser encontrados alguns trabalhos que implementam simulated annealing com esta estratégia [Boese and Kahng, 1993] [Boese and Kahng, 1994][Boese et al., 1995].

Considerando um conjunto S de soluções e uma função de custo real f, a metodologia proposta procura o mínimo global s, tal que $f(s) \leq f(s')$, $\forall s' \in S$. O processo termina após I_{max} iterações ou se o critério de parada baseado em validação for satisfeito, retornando a melhor solução encontrada durante a execução s_{BSF} (*best so far*). O esquema de esfriamento atualiza a temperatura T_i da iteração i a cada I_T iterações do algoritmo. Em cada iteração, são geradas K soluções novas a partir da atual. Vale lembrar que cada solução contém informações sobre a topologia e os pesos da rede MLP. A estrutura do método é apresentada a no Algoritmo 3.1.

1. $s_0 \leftarrow$ solução inicial
2. $T_0 \leftarrow$ temperatura inicial
3. Atualize s_{BSF} com s_0 (melhor solução encontrada até o momento)
4. Para $i = 0$ até $I_{max} - 1$
5. Se $i + 1$ não for múltiplo de I_T
6. $T_{i+1} \leftarrow T_i$
7. senão
8. $T_{i+1} \leftarrow$ nova temperatura
9. Se critério de parada baseado em validação for satisfeito
10. Interrompa execução do algoritmo
11. Gere um conjunto com K soluções novas a partir de s_i
12. Escolha a melhor solução s' do conjunto
13. Se $f(s') < f(s_i)$
14. $s_{i+1} \leftarrow s'$
15. senão
16. $s_{i+1} \leftarrow s'$ com probabilidade $e^{-[f(s')-f(s_i)]/T_{i+1}}$
17. Atualize s_{BSF} (se $f(s_{i+1}) < f(s_{BSF})$)
18. Fixe a topologia contida em s_{BSF} e use seus pesos como iniciais para treinar com um algoritmo local de treinamento.

Algoritmo 3.1: Algoritmo proposto.

Considerando o que foi explicado, o método procura superar as limitações individuais de simulated annealing e tabu search, procurando combinar as principais características favoráveis de ambos os métodos, com o intuito de tornar mais eficiente o processo de otimização. Além disso, faz uso de treinamento híbrido, combinando um método com características globais com uma técnica local de treinamento de redes neurais, que pode ser implementada, por exemplo, pelo conhecido algoritmo backpropagation.

Para implementar este método na otimização simultânea de arquiteturas e pesos de redes MLP, é necessário definir os seguintes aspectos de implementação: a representação das soluções, a função de custo, o mecanismo de geração de soluções novas (operador), o esquema de esfriamento, os critérios de parada e o algoritmo local de treinamento. Tais aspectos são detalhados a seguir.

3.3 Representação das Soluções

Neste trabalho, as topologias MLP possuem uma única camada escondida, contendo todas as conexões possíveis entre camadas adjacentes, sem haver conexões entre camadas não-adjacentes. Deve ser definida uma topologia máxima, que possui N_1 nodos de entrada, N_2 nodos escondidos e N_3 nodos de saída. Como se pode perceber, os valores de N_1 e N_3 são definidos pelo problema, dependendo do conjunto de dados e do pré-processamento adotado (número de atributos de entrada e número de saídas), enquanto N_2 é um parâmetro que deve ser escolhido na implementação. Uma vez que as topologias possuem todas as conexões entre camadas adjacentes, sem conter conexões entre camadas não-adjacentes, a quantidade máxima de conexões é dada por:

$$N_{\max} = N_1 N_2 + N_2 N_3 . \tag{1}$$

Cada solução s é composta por dois vetores: C, cujos elementos são binários, representando a conectividade da rede, e W, cujos elementos são números reais, representando o conjunto de pesos. Dessa forma:

$$s \equiv (C, W), \tag{2}$$

$$C \equiv (c_1, c_2, ..., c_{N_{\max}}), \quad c_i \in \{0,1\}, \quad i = 1, 2, ..., N_{\max}, \tag{3}$$

$$W \equiv (w_1, w_2, ..., w_{N_{\max}}), \quad w_i \in \Re, \quad i = 1, 2, ..., N_{\max}, \tag{4}$$

sendo que $\Re$ é o conjunto dos números reais.

Assim, a conexão i de uma rede MLP é especificada por dois parâmetros: um *bit de conectividade* (c_i), que é igual a 1 se a conexão existir na rede, sendo igual a 0 em caso contrário, e o *peso da conexão* (w_i), que é um número real. Se o bit de conectividade c_i é igual a 0, seu peso w_i associado não é considerado, já que a conexão i não existe na rede.

A solução inicial s_0 é uma rede MLP com a topologia máxima definida (ou seja, $c_i = 1,\ i = 1, 2, ..., N_{\max}$), sendo que os pesos iniciais são extraídos aleatoriamente de uma distribuição uniforme no intervalo [-1.0, +1.0].

3.4 Função de Custo

Chamando de P_t o conjunto de padrões de treinamento e considerando a existência de N_C classes, define-se a *classe verdadeira do padrão x*:

$$\gamma(x) \in \{1, 2, ..., N_C\}, \quad \forall x \in P_t . \tag{5}$$

Neste trabalho, foi utilizado o critério de classificação *winner-takes-all* ("o vencedor leva tudo"), segundo o qual a unidade de saída que apresentar o maior valor de saída determina a classe do padrão. Por este motivo, a quantidade de nodos de saída (N_3) é igual à quantidade de classes do problema (N_C).

Assim, chamando de $o_k(x)$ o valor de saída do nodo de saída *k* para o padrão *x*, define-se a *classe atribuída ao padrão x*:

$$\phi(x) \equiv \underset{k \in \{1, 2, ..., N_3\}}{\arg\max}\ o_k(x), \quad \forall x \in P_t . \tag{6}$$

Uma vez definida a classe verdadeira e a classe atribuída ao padrão *x*, define-se o *erro da rede para o padrão x* como:

$$\varepsilon(x) \equiv \begin{cases} 1, & se\ \ \phi(x) \neq \gamma(x), \\ 0, & se\ \ \phi(x) = \gamma(x). \end{cases} \tag{7}$$

Dessa forma, o *erro de classificação para o conjunto* P_t, que representa a porcentagem de padrões de treinamento classificados erradamente, pode ser definido como:

$$E(P_t) \equiv \frac{100}{\#P_t} \sum_{x \in P_t} \varepsilon(x), \tag{8}$$

sendo que $\#P_t$ é a quantidade de padrões do conjunto P_t.

Define-se, ainda, a porcentagem de conexões utilizadas pela rede como:

$$\psi(C) \equiv \frac{100}{N_{\max}} \sum_{i=1}^{N_{\max}} c_i . \tag{9}$$

Neste trabalho, o custo $f(s)$ de cada solução *s* é dado pela média aritmética entre o erro de classificação para o conjunto de treinamento e a porcentagem de conexões utilizadas pela rede:

$$f(s) \equiv \frac{1}{2}\,(E(P_t) + \psi(C)). \tag{10}$$

Dessa forma, o algoritmo procura minimizar tanto o erro de treinamento quanto o número de conexões da rede. Vale comentar que apenas redes válidas (ou seja, com pelo menos um nodo na camada escondida) foram consideradas.

3.5 Mecanismo de Geração de Soluções Novas

A partir da solução atual $s = (C, W)$, gera-se uma solução nova $s' = (C', W')$, em que $C' = (c'_1, c'_2, ..., c'_{N_{\max}})$ e $W' = (w'_1, w'_2, ..., w'_{N_{\max}})$, da seguinte forma: extrai-se um número aleatório α a partir de uma distribuição uniforme no intervalo [0, 1]. O valor do novo bit de conectividade c'_i da conexão *i* é dado por:

$$c'_i = \begin{cases} \overline{c}_i, & se\ \alpha \le p, \\ c_i, & se\ \alpha > p, \end{cases} \tag{11}$$

em que $\overline{c}_i$ é o inverso do bit c_i, e p é a probabilidade de inverter cada bit de conectividade, que é um parâmetro de implementação. Além disso, extrai-se um outro número aleatório β a partir de uma distribuição uniforme em [-1.0,+1.0], e o valor do novo peso w'_i correspondente à conexão *i* é dado por:

$$w'_i = w_i + \beta. \tag{12}$$

Percebe-se que o mecanismo de geração de soluções novas atua da seguinte forma: primeiramente, os bits de conectividade da solução atual são invertidos de acordo com uma dada probabilidade (*p*). Esta operação remove algumas conexões da topologia e cria outras. Em seguida, um número aleatório retirado de uma distribuição uniforme em [-1.0,+1.0] é adicionado a cada peso. Estes dois passos alteram tanto a arquitetura quanto os pesos com o intuito de gerar uma nova solução. Vale ressaltar,

ainda, que a quantidade de soluções novas (*K*) geradas em cada iteração é outro parâmetro de implementação da metodologia.

3.6 Esquema de Esfriamento e Critérios de Parada

Utiliza-se o *esquema geométrico de esfriamento* comentado no Capítulo 2, segundo o qual a temperatura nova é igual à atual multiplicada por um fator de redução (*r*), que deve ser menor do que 1, sendo próximo de 1. A temperatura inicial T_0 e o fator de temperatura *r* são parâmetros da implementação, assim como I_T (quantidade de iterações entre duas atualizações sucessivas da temperatura) e I_{max} (máximo número de iterações permitidas). Assim, a temperatura T_i da iteração *i* é dada por:

$$T_i = \begin{cases} r\,T_{i-1}, & se\ i = k\,I_T, \quad k = 1,2,...,\dfrac{I_{\max}}{I_T}, \\ T_i, & caso\ contrário. \end{cases} \tag{13}$$

A execução do algoritmo é finalizada se: (1) o critério GL_5 definido no *Proben1* [Prechelt, 1994] for satisfeito (baseado no erro de classificação para o conjunto de validação); ou (2) a quantidade máxima de iterações for atingida. Para a implementação do critério GL_5, o erro de classificação no conjunto de validação é calculado a cada I_T iterações.

O critério GL_5 é uma boa estratégia para evitar o ajuste excessivo da rede aos exemplos particulares de treinamento (*overfitting*), fenômeno que prejudica o desempenho de generalização da rede. Chamando de P_v o conjunto de padrões de validação, tem-se que o erro de classificação sobre este conjunto é dado por $E(P_v)$, calculado segundo a expressão (8). Dessa forma, denotando por $V(k)$ o valor do erro de classificação $E(P_v)$ calculado na iteração $i = k\,I_T, \quad k = 1,2,...,\dfrac{I_{\max}}{I_T}$, o parâmetro *generalization loss* ($GL(k)$) é igual ao aumento percentual no erro de validação em relação ao mínimo erro de validação já encontrado:

$$GL(k) = 100\left(\frac{V(k)}{\min\limits_{j \le k} V(j)} - 1\right). \tag{14}$$

O critério GL_5 interrompe o treinamento assim que o parâmetro $GL(k)$ ultrapassa 5% [Prechelt, 1994].

3.7 Algoritmo Local de Treinamento

Sabe-se que técnicas de otimização global são relativamente ineficientes para ajuste fino em buscas locais. Dessa forma, é importante melhorar o desempenho de generalização das redes treinando as topologias finais com um método de busca local. Isto foi implementado da seguinte forma: sem alterar as topologias, as redes finais produzidas pela otimização global foram treinadas com um algoritmo local de treinamento. Esta combinação de métodos de otimização global com técnicas locais, que é freqüentemente chamada de *treinamento híbrido*, tem sido utilizada em trabalhos com algoritmos genéticos [Yao, 1999], como se comentou anteriormente.

Em geral, os algoritmos de treinamento mais tradicionais para treinar redes MLP são baseados em gradiente, sendo que o exemplo mais conhecido é o backpropagation. Neste trabalho, o backpropagation foi escolhido pela facilidade de implementação, uma vez que o objetivo é mostrar que a combinação cuidadosa de técnicas simples e tradicionais é capaz de gerar bons resultados. Detalhes sobre o algoritmo backpropagation podem ser encontrados no Apêndice A.

3.8 Comentários Finais

Dessa forma, ficam definidos os aspectos importantes da metodologia proposta para a otimização simultânea de arquiteturas e pesos de redes MLP. É importante ressaltar que os aspectos de representação, função de custo, operador e algoritmo local de treinamento definidos nas seções anteriores servem não apenas para a implementação da metodologia de otimização proposta, podendo, também, ser usados para implementar simulated annealing, tabu search e algoritmos genéticos.

Capítulo 4

Resultados Obtidos para o Reconhecimento de Odores

4.1 Introdução

Uma vez explicados os detalhes da metodologia proposta, este capítulo tem a finalidade de apresentar os resultados obtidos com sua implementação para o problema do reconhecimento de odores em um nariz artificial. Além disso, apresenta resultados de simulated annealing, tabu search e algoritmo genético utilizando a representação de soluções, a função de custo e o operador descritos no Capítulo 3, com o objetivo de mostrar que estas escolhas são adequadas para a metodologia proposta. Os resultados mostram que a metodologia proposta pode ser aplicada de forma eficiente para a

otimização simultânea de arquiteturas e pesos de redes MLP, gerando redes pequenas com alta capacidade de generalização. Além disso, os resultados mostram que a combinação de métodos de otimização global com técnicas locais de gradiente, em uma abordagem de treinamento híbrido, é capaz de melhorar ainda mais o desempenho de generalização das redes MLP.

4.2 Problema e Base de Dados

O problema abordado trata da classificação de odores provenientes de três safras distintas (anos 1995, 1996 e 1997) de um mesmo vinho tinto comercial (Almadén, Brasil) produzido com uvas do tipo *merlot* [De Souza et al., 1999].

A base de dados utilizada foi obtida pela equipe multidisciplinar do Projeto Aroma, formada por membros do Centro de Informática e dos Departamentos de Física e Química Fundamental do Centro de Ciências Exatas e da Natureza da Universidade Federal de Pernambuco [De Souza et al., 1999][Santos, 2000], através do uso de um protótipo de nariz artificial, o qual contém seis sensores distintos de polímero condutor, formados pela deposição eletroquímica de polipirrol com diferentes tipos de dopante.

A aquisição dos dados foi feita de forma automática por um computador, estando os sensores expostos aos odorantes em uma câmara especial de testes. Mais detalhes sobre a construção do protótipo podem ser encontrados em [De Souza et al., 1999].

Foram realizadas três repetições de aquisição de dados para cada safra de vinho. Em cada uma destas repetições, a resistência elétrica de cada sensor foi registrada de meio em meio segundo, sendo que o processo durava cinco minutos. Dessa forma, cada sensor obteve 600 valores registrados para cada safra de vinho.

O conjunto formado pelos seis valores de resistência no mesmo instante de tempo foi considerado como um padrão da base de dados. Assim, cada repetição contém 1.800 padrões (600 de cada safra). Como existem três repetições, tem-se um total de 5.400 padrões na base de dados.

As Tabelas 4.1, 4.2 e 4.3 mostram exemplos de aquisição de dados para as safras de 1995, 1996 e 1997, respectivamente, permitindo uma melhor visualização.

Tempo (s)	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
0.5	938	22553	3303	282	2777	5500
1.0	923	22522	3303	289	2761	5500
1.5	938	22545	3311	289	2761	5500
2.0	930	22530	3303	289	2784	5508
...	...	...	...	...	...	...
300.0	953	22347	3318	282	2769	5539

Tabela 4.1: Exemplo de aquisição de dados para a safra de 1995.

Tempo (s)	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
0.5	846	4737	2716	297	2922	5447
1.0	839	4730	2708	297	2937	5455
1.5	839	4745	2700	289	2922	5455
2.0	823	4730	2700	282	2922	5447
...	...	...	...	...	...	...
300.0	877	5012	2662	297	3097	5500

Tabela 4.2: Exemplo de aquisição de dados para a safra de 1996.

Tempo (s)	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
0.5	968	20523	28931	358	2037	4783
1.0	999	20554	28916	358	2037	4783
1.5	991	20554	28901	358	2029	4776
2.0	984	20554	28923	358	2029	4783
...	...	...	...	...	...	...
300.0	1007	28748	30396	366	2166	4875

Tabela 4.3: Exemplo de aquisição de dados para a safra de 1997.

Vale ressaltar que mais detalhes sobre narizes artificiais podem ser encontrados no Apêndice B do presente trabalho.

4.3 Trabalhos Anteriores com a Base de Dados

A base de dados explicada na seção anterior já foi utilizada em trabalhos anteriores de aplicação de redes neurais, embora sem a preocupação de otimizar arquiteturas e pesos simultaneamente. É importante analisar estes resultados, a fim de identificar a relevância da aplicação da metodologia proposta neste problema.

4.3.1 Multi-Layer Perceptron

Em trabalhos anteriores, redes MLP com uma camada escondida foram utilizadas para classificar a base de dados [Yamazaki, 2001][Yamazaki and Ludermir, 2001a]. As redes contiveram seis nodos de entrada (um para cada sensor) e três nodos de saída (um para cada safra de vinho). Seis topologias distintas foram consideradas (2, 4, 8, 12, 16 e 20 nodos escondidos), com o intuito de investigar como o desempenho de classificação se comporta com o aumento no tamanho da rede. A Figura 4.1 apresenta a topologia de 2 nodos escondidos. Para cada topologia, foram feitas 10 execuções com inicializações de pesos distintas e aleatórias, sendo que os pesos iniciais foram extraídos de uma distribuição uniforme no intervalo [-1.0, +1.0].

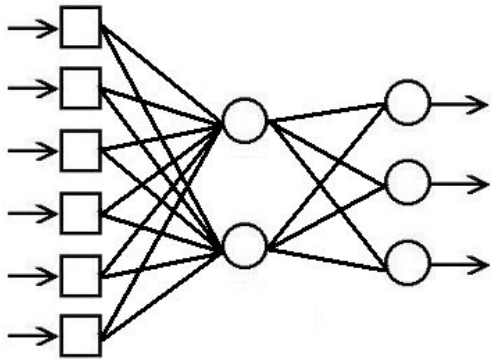


Figura 4.1: Rede MLP com 2 nodos escondidos.

O algoritmo de treinamento utilizado para esta abordagem foi uma versão modificada do método de Levenberg-Marquardt [Fletcher, 1987], por apresentar rápida convergência e robustez [Norgaard, 1997]. Apesar dos requisitos de memória, que

tornam este método impraticável para redes neurais de topologia muito grande, uma vantagem importante desta técnica é que seu único parâmetro ajustável (o valor inicial do parâmetro *lambda*) não é crítico para o desempenho da rede, uma vez que é ajustado de forma adaptativa e apenas influencia a convergência inicial: se seu valor for muito grande, o algoritmo realiza passos iniciais pequenos, e, se for muito pequeno, o algoritmo o incrementa até que sejam realizados passos suficientemente curtos [Norgaard, 1997]. Uma vez que a convergência não sofre grande impacto por conta da escolha deste parâmetro, o método de Levenberg-Marquardt é mais adequado à comparação de topologias distintas aplicadas ao mesmo problema do que o algoritmo backpropagation, que é bastante influenciado pela escolha de seu parâmetro principal (a taxa de aprendizado). Detalhes sobre o método de Levenberg-Marquardt podem ser encontrados no Apêndice A.

Todos os nodos da rede implementaram a função de ativação tangente hiperbólica, e as redes contiveram todas as conexões possíveis entre camadas adjacentes, sem possuir conexões entre camadas não-adjacentes. Foi utilizado o critério de classificação *winner-takes-all* ("o vencedor leva tudo").

O número máximo de épocas permitidas foi 100, e o treinamento era finalizado se: (1) o critério GL_5 definido no *Proben1* [Prechelt, 1994] fosse satisfeito (baseado na soma dos erros quadráticos para o conjunto de validação); ou (2) a quantidade máxima de iterações fosse atingida.

O conjunto de dados foi dividido da seguinte forma: 50% dos padrões de cada classe foram atribuídos aleatoriamente ao conjunto de treinamento, 25% foram atribuídos ao conjunto de validação, e os 25% restantes foram reservados para o conjunto de teste, como sugerido pelo *Proben1* [Prechelt, 1994].

Na Tabela 4.4, apresentam-se, para cada topologia, a média e o desvio-padrão dos erros de classificação para o conjunto de teste obtidos nas 10 execuções realizadas [Yamazaki, 2001][Yamazaki and Ludermir, 2001a].

Quantidade de Nodos Escondidos	Erro de Classificação para o Conjunto de Teste (%)	
	Média	Desvio-padrão
02	35.49	29.58
04	58.88	17.14
08	40.95	09.44
12	36.28	15.06
16	20.96	19.00
20	15.74	12.51

Tabela 4.4: Resultados para as redes MLP.

Pode ser observado que o erro mínimo de classificação para o conjunto de teste foi de 15.74%, o qual foi obtido pela maior topologia aplicada (20 nodos escondidos), que contém 180 conexões (120 conexões dos 6 nodos de entrada para os 20 nodos escondidos mais 60 conexões dos 20 nodos escondidos para os 3 nodos de saída). Além destas execuções, foram feitos experimentos (não sistemáticos) com quantidades maiores de nodos escondidos, mas não foram constatadas diferenças importantes, uma vez que os resultados se estabilizaram nos valores obtidos para a topologia de 20 nodos escondidos.

Com o intuito de melhorar o desempenho, foram implementadas topologias TDNN [Lang and Hinton, 1988] para investigar a influência do processamento temporal nos erros de classificação.

4.3.2 Processamento Temporal com Time Delay Neural Networks

Na base de dados utilizada neste trabalho, os padrões foram obtidos pelos sensores de forma seqüencial a cada meio segundo durante cinco minutos. Entretanto, esta informação de tempo não foi considerada pelos experimentos com redes MLP, uma vez que as redes MLP são projetadas para lidar apenas com padrões estáticos. Com o objetivo de verificar se as características temporais podem melhorar o desempenho de classificação, foram implementadas arquiteturas TDNN. As topologias TDNN utilizadas neste trabalho foram idênticas às arquiteturas MLP, exceto pela camada de entrada, que possui 12 unidades (6 para o padrão atual e outras 6 para o padrão apresentado no instante de tempo anterior). A Figura 4.2 apresenta a arquitetura TDNN de 2 nodos

escondidos. Explicações sobre arquiteturas TDNN podem ser encontradas no Apêndice A do presente trabalho.

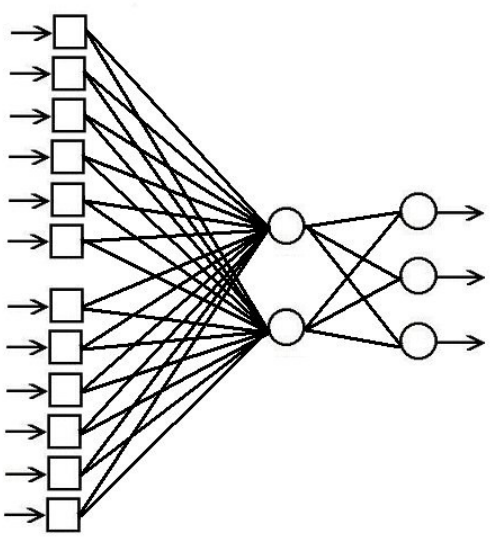


Figura 4.2: TDNN com 2 nodos escondidos. A camada de entrada possui 12 unidades (6 para o padrão atual e 6 para o padrão do instante de tempo anterior).

Dessa forma, a rede considera as relações entre cada padrão e o anterior, e esta informação é utilizada no processo de classificação. Alternativamente, poderiam ser usados mais *delays*, de modo a procurar a quantidade ótima de *delays* a ser adotada. Entretanto, este trabalho não teve a intenção de fazer uma exploração na quantidade de *delays*. O objetivo foi mostrar que a adição de informação temporal melhora o desempenho de classificação.

Novamente, foram treinadas seis topologias distintas (2, 4, 8, 12, 16 e 20 nodos escondidos), e, para cada topologia, foram realizadas 10 execuções com inicializações de pesos distintas e aleatórias, sendo que os pesos iniciais foram extraídos de uma distribuição uniforme no intervalo [-1.0, +1.0]. O algoritmo de treinamento e os critérios de parada foram os mesmos adotados nos experimentos com MLP, assim como o critério de classificação (*winner-takes-all*).

A divisão dos dados em treinamento, validação e teste teve de ser realizada de forma distinta da que foi escolhida para os testes com MLP, a fim de preservar a ordem segundo a qual os padrões foram adquiridos pelos sensores. Como foi dito

anteriormente, foram feitas 3 aquisições de dados para cada safra de vinho. Dessa forma, a primeira aquisição de dados foi atribuída ao conjunto de treinamento, a segunda foi atribuída ao conjunto de validação, e a última foi reservada para testar a rede (neste caso, os três conjuntos de dados possuem a mesma quantidade de padrões). Este particionamento foi escolhido de modo a manter o comportamento temporal dos padrões em cada um dos três conjuntos de dados.

Na Tabela 4.5, apresentam-se, para cada topologia, a média e o desvio-padrão dos erros de classificação para o conjunto de teste obtidos nas 10 execuções realizadas. Tais resultados foram publicados em trabalhos anteriores [Yamazaki, 2001][Yamazaki and Ludermir, 2001a] [Yamazaki and Ludermir, 2001b] [Yamazaki et al., 2001].

Quantidade de Nodos Escondidos	**Erro de Classificação para o Conjunto de Teste (%)**	
	Média	**Desvio-padrão**
02	**03.36**	**10.53**
04	**00.02**	**00.05**
08	**00.00**	**00.00**
12	**00.00**	**00.00**
16	**06.42**	**16.08**
20	**16.11**	**24.95**

Tabela 4.5: Resultados para as topologias TDNN.

As topologias com 8 e 12 nodos escondidos obtiveram erro médio de classificação nulo, e a arquitetura com 4 nodos escondidos conseguiu um erro médio de classificação muito próximo de zero. Percebe-se que, levando em conta as propriedades temporais dos dados, foram obtidos melhores desempenhos de classificação.

Com o intuito de verificar se a melhoria no desempenho foi influenciada pelo novo particionamento dos dados, foram realizados experimentos com redes MLP aplicadas à nova divisão da base de dados. Para cada topologia, as 10 inicializações distintas de pesos foram as mesmas adotadas nos primeiros experimentos com redes MLP, permitindo uma comparação direta entre os diferentes particionamentos dos dados. Os resultados são apresentados na Tabela 4.6 [Yamazaki et al., 2001].

Quantidade de Nodos Escondidos	Erro de Classificação para o Conjunto de Teste (%)	
	Média	Desvio-padrão
02	44.74	19.81
04	59.04	21.79
08	43.33	09.29
12	50.53	17.76
16	32.19	17.50
20	26.92	10.01

Tabela 4.6: Resultados para as redes MLP usando o novo particionamento dos dados.

Pode ser observado que o erro mínimo de classificação obtido para o conjunto de teste foi de 26.92% (para a topologia de 20 nodos escondidos), que é maior do que o mínimo erro encontrado nos primeiros experimentos com MLP (15.74%). Sendo assim, o novo particionamento dos dados não provocou nenhuma melhoria no desempenho de generalização.

De fato, observa-se que o desempenho piorou para todas as topologias, o que pode ser explicado da seguinte forma: no novo particionamento, cada um dos três conjuntos de dados (treinamento, validação e teste) corresponde a uma aquisição diferente. As três aquisições realizadas apresentam algumas diferenças entre si, uma vez que foram feitas em momentos distintos, e o estado dos sensores pode variar devido às condições do ambiente. Neste caso, os dados utilizados para treinamento, validação e teste podem apresentar diferenças significativas, tornando mais difícil a generalização. Nos primeiros experimentos usando redes MLP, os padrões das três aquisições de dados foram atribuídos aleatoriamente aos conjuntos de treinamento, validação e teste, de modo que as irregularidades foram distribuídas nestes três conjuntos de dados, que se tornaram mais similares uns aos outros, tornando a generalização mais fácil.

O uso de informação temporal melhorou os resultados de classificação, porém o número de conexões das redes aumentou significativamente, já que uma topologia TDNN contém mais conexões do que uma arquitetura MLP com a mesma quantidade de nodos escondidos (topologias TDNN possuem 12 nodos de entrada, enquanto redes MLP possuem 6). Este aspecto é muito importante na implementação em hardware das

redes neurais para narizes artificiais, e isto reforça a motivação de otimizar as topologias. Além disso, os melhores resultados foram obtidos através de tentativas com diversas topologias, tornando o processo experimental cansativo e demorado, o que aumenta ainda mais a necessidade de um processo automático para definição de topologias de redes MLP.

4.4 Experimentos com Simulated Annealing

Nestes experimentos, a topologia máxima contém seis nodos de entrada (um para cada sensor), quatro nodos escondidos e três nodos de saída (um para cada safra de vinho), de modo que $N_1 = 6$, $N_2 = 4$ e $N_3 = 3$, de acordo com a metodologia descrita anteriormente. Dessa forma, a quantidade máxima de conexões (N_{max}) é igual a 36.

No mecanismo de geração de soluções novas, a probabilidade de inverter os bits de conectividade da solução atual (p) é igual a 20%. A temperatura inicial (T_0) é igual a 1, e o fator de temperatura (r) é igual a 0.9. A temperatura é reduzida a cada 10 iterações do algoritmo (I_T), e o máximo número de iterações permitidas (I_{max}) é igual a 1.000. Todos estes parâmetros foram escolhidos de forma empírica.

O desempenho do algoritmo de simulated annealing é influenciado pela escolha do esquema de esfriamento e do mecanismo de geração de soluções novas, entretanto não existem regras objetivas para o ajuste da configuração de modo a obter os melhores resultados possíveis, sendo normalmente adotadas configurações variadas dos parâmetros para avaliar o desempenho [Sexton et al., 1999]. Assim sendo, a configuração adotada neste trabalho, que foi escolhida após experimentos preliminares, pode não ser ótima para o problema abordado. Uma exploração de parâmetros mais rigorosa poderia ter gerado melhores resultados, porém este trabalho não tem o intuito de apresentar uma exploração exaustiva dos parâmetros ajustáveis, pois esta possibilidade consumiria um tempo excessivo para a realização dos testes. O objetivo desta abordagem é mostrar que o algoritmo de simulated annealing alcançou bons resultados para o problema de otimização tratado, apesar da dificuldade para o ajuste dos parâmetros.

O particionamento dos dados foi o mesmo utilizado nos primeiros experimentos com MLP: 50% de cada classe para treinamento, 25% para validação e 25% para teste.

Novamente, as unidades escondidas e de saída implementaram funções de ativação tangente hiperbólica, sendo empregado o mesmo critério de classificação (*winner-takes-all*). Tais escolhas serão mantidas em todos os experimentos relatados a seguir.

Foram utilizadas 10 inicializações distintas e aleatórias de pesos, sendo que os pesos iniciais foram extraídos de uma distribuição uniforme em [-1.0,+1.0], de acordo com a metodologia descrita. Para cada inicialização de pesos, foram realizadas 30 execuções de simulated annealing. Foram excluídas as 10 melhores execuções (aquelas com menor erro de classificação para o conjunto de validação), bem como as 10 piores execuções (as de maior erro de validação), de modo que as 10 execuções restantes foram consideradas para os resultados finais. Este procedimento foi adotado com o intuito de considerar apenas as execuções com erros intermediários (ou seja, descartando os melhores e os piores casos). Entretanto, em uma aplicação prática, sabe-se que devem ser escolhidas as melhores redes para resolver o problema de classificação.

Os resultados para essa abordagem são mostrados na Tabela 4.7 [Ludermir and Yamazaki, 2003][Yamazaki et al., 2002a][Yamazaki et al., 2002b][Yamazaki et al., 2002c][Yamazaki and Ludermir, 2003]. Estes resultados mostram que as redes conseguem um erro médio de classificação igual a 6.01% e possuem, em média, 11.51 conexões, quantidade muito menor que o número máximo de conexões permitidas (36). Os valores de desvio-padrão foram baixos, indicando homogeneidade nos resultados ao longo das diferentes inicializações de pesos.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**5.27**	**4.70**	**2.90**	**11.00**
2	**5.81**	**4.70**	**3.40**	**10.80**
3	**3.87**	**4.50**	**3.00**	**11.60**
4	**8.27**	**5.00**	**2.90**	**12.00**
5	**7.24**	**4.30**	**3.20**	**11.50**
6	**3.07**	**4.50**	**3.10**	**11.60**
7	**5.47**	**4.50**	**3.00**	**10.70**
8	**6.50**	**4.40**	**3.10**	**12.00**
9	**6.79**	**4.90**	**3.10**	**12.30**
10	**7.77**	**4.40**	**3.10**	**11.60**
Média	**6.01**	**4.59**	**3.08**	**11.51**
Desvio-padrão	**1.66**	**0.23**	**0.15**	**0.53**

Tabela 4.7: Resultados para simulated annealing.

A fim de verificar se o desempenho das redes finais geradas por simulated annealing poderia ser melhorado, foram treinadas as topologias finais usando o algoritmo backpropagation, de acordo com o que foi comentado no Capítulo 3. A taxa de aprendizado escolhida foi 0.0001, e termo de *momentum* foi igual a 0.8, sendo que a quantidade máxima de iterações permitidas foi igual a 1.000. Novamente, estes valores podem não ter sido ótimos para o problema, porém este trabalho tem a intenção de mostrar que é possível melhorar os resultados de generalização das redes geradas por simulated annealing com a adição de uma fase de treinamento com backpropagation, apesar da dificuldade em ajustar os parâmetros deste algoritmo de gradiente. Os resultados são apresentados na Tabela 4.8 [Ludermir and Yamazaki, 2003][Yamazaki et al., 2002c][Yamazaki and Ludermir, 2003].

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**1.82**	**4.50**	**2.80**	**10.50**
2	**0.99**	**4.90**	**3.20**	**10.80**
3	**1.47**	**4.80**	**2.90**	**10.60**
4	**2.07**	**5.00**	**2.80**	**11.80**
5	**0.67**	**3.90**	**2.90**	**10.20**
6	**3.81**	**4.70**	**3.20**	**12.70**
7	**5.19**	**4.70**	**3.20**	**12.20**
8	**0.38**	**4.30**	**2.80**	**10.40**
9	**1.34**	**4.20**	**3.10**	**10.80**
10	**5.32**	**4.20**	**2.90**	**11.10**
Média	**2.31**	**4.52**	**2.98**	**11.11**
Desvio-padrão	**1.82**	**0.36**	**0.18**	**0.84**

Tabela 4.8: Resultados para simulated annealing combinado com backpropagation.

Os resultados mostram que houve uma redução no erro médio de classificação (de 6.01% para 2.31%), por conta do ajuste fino realizado pelo backpropagation. Tal constatação foi confirmada por um teste de diferença entre médias, conforme será explicado na Seção 4.8, que traz uma comparação mais abrangente entre abordagens diferentes.

Conforme se comentou anteriormente, a abordagem mais básica de simulated annealing, que gerou os resultados das Tabelas 4.7 e 4.8, não guarda a melhor solução encontrada durante o processo de otimização, de modo que a solução final gerada pelo algoritmo é a solução obtida na última iteração. A fim de verificar se este aspecto é capaz de melhorar os desempenhos, foi implementada uma outra versão de simulated annealing, guardando a melhor solução encontrada durante a execução, que obteve os resultados da Tabela 4.9 [Yamazaki and Ludermir, 2003]. Vale lembrar que as soluções iniciais são as mesmas utilizadas nos experimentos anteriores com simulated annealing.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**3.43**	**5.20**	**2.40**	**10.90**
2	**4.03**	**4.90**	**3.20**	**12.30**
3	**2.30**	**4.60**	**3.30**	**12.90**
4	**5.69**	**4.90**	**3.00**	**12.70**
5	**6.99**	**4.40**	**2.80**	**12.30**
6	**4.33**	**5.10**	**3.40**	**12.10**
7	**4.48**	**4.70**	**2.80**	**12.70**
8	**6.33**	**4.40**	**3.50**	**11.70**
9	**6.72**	**4.60**	**2.80**	**11.60**
10	**3.90**	**4.70**	**3.20**	**11.30**
Média	**4.82**	**4.75**	**3.04**	**12.05**
Desvio-padrão	**1.54**	**0.27**	**0.34**	**0.66**

Tabela 4.9: Resultados para a versão de simulated annealing que guarda a melhor solução.

Observa-se que o erro médio de classificação (4.82%) foi menor que o obtido pela versão anterior sem backpropagation (6.01%). Com o intuito de verificar se o desempenho também melhora quando as topologias finais são treinadas com backpropagation, foram obtidos os resultados da Tabela 4.10 [Yamazaki and Ludermir, 2003]. Percebe-se que o erro médio de classificação (1.41%) realmente foi menor que o obtido sem backpropagation (4.82%).

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**0.86**	**4.30**	**2.60**	**10.40**
2	**1.08**	**4.60**	**3.00**	**11.30**
3	**0.73**	**4.60**	**2.80**	**11.60**
4	**1.41**	**5.00**	**2.90**	**12.10**
5	**1.04**	**4.40**	**3.00**	**11.50**
6	**3.08**	**4.80**	**3.10**	**12.40**
7	**0.13**	**4.30**	**2.90**	**10.40**
8	**0.37**	**4.20**	**3.40**	**11.00**
9	**2.70**	**4.50**	**2.90**	**11.50**
10	**2.73**	**4.40**	**2.60**	**10.30**
Média	**1.41**	**4.51**	**2.92**	**11.25**
Desvio-padrão	**1.05**	**0.25**	**0.23**	**0.72**

Tabela 4.10: Resultados para a versão de simulated annealing que guarda a melhor solução combinada com backpropagation.

A mesma metodologia adotada nos experimentos com simulated annealing foi utilizada na implementação de tabu search, como será visto a seguir.

4.5 Experimentos com Tabu Search

Nos experimentos com tabu search, foram adotados os mesmos parâmetros de implementação escolhidos para simulated annealing, exceto os parâmetros que dizem respeito ao esquema de esfriamento, os quais, obviamente, não se aplicam ao método de tabu search. São produzidas 20 soluções novas distintas a cada iteração, e a lista tabu armazena as 10 soluções mais recentemente visitadas.

Uma vez que os pesos das conexões são números reais, é pouco provável encontrar soluções que sejam exatamente idênticas. Por este motivo, foi usado um critério de proximidade para relaxar a comparação direta dos pesos, como se encontra em [Sexton et al., 1988]. De acordo com este critério, uma nova solução é considerada igual a uma das soluções proibidas da lista se cada bit de conectividade da nova solução for idêntico ao bit correspondente na solução proibida e se cada peso da nova solução

estiver entre ± N em relação ao peso correspondente na solução proibida, sendo que o parâmetro N é um número real, o qual, neste trabalho, é igual a 0.01.

Assim como acontece nos experimentos com simulated annealing, não há regras para ajustar os parâmetros de tabu search de modo a obter os melhores resultados, e a configuração usada neste trabalho foi escolhida de forma empírica, podendo não ser ótima para o problema. Novamente, vale lembrar que este trabalho procura mostrar que podem ser alcançados bons resultados usando tabu search para o problema em questão, apesar da dificuldade em ajustar seus parâmetros.

Uma diferença em relação aos experimentos anteriores é que a quantidade máxima de iterações permitidas (I_{max}) é igual a 100. Este número é menor do que a quantidade máxima permitida nos experimentos com simulated annealing (1.000), uma vez que cada iteração de tabu search avalia um conjunto de soluções novas, e não apenas uma, como ocorre na implementação de simulated annealing. Por este motivo, tabu search geralmente necessita de menos iterações do que simulated annealing para convergir.

Nos experimentos com tabu search, foram usadas as mesmas soluções iniciais adotadas nas implementações de simulated annealing. Os resultados são apresentados na Tabela 4.11 [Ludermir and Yamazaki, 2003][Yamazaki et al., 2002b][Yamazaki et al., 2002c][Yamazaki and Ludermir, 2003].

Como pode ser observado, o erro médio de classificação foi de 3.33%, sendo menor que o obtido pela versão de simulated annealing que guarda a melhor solução (4.82%). Além disso, percebe-se que a quantidade média de conexões obtida por tabu search foi semelhante à obtida por simulated annealing.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**5.96**	**5.10**	**3.10**	**11.60**
2	**3.91**	**5.10**	**3.30**	**11.70**
3	**0.93**	**4.30**	**3.20**	**10.80**
4	**3.32**	**5.20**	**2.50**	**11.40**
5	**3.57**	**4.80**	**3.10**	**12.20**
6	**4.72**	**4.60**	**3.10**	**10.90**
7	**3.05**	**4.30**	**3.20**	**11.10**
8	**3.16**	**4.30**	**3.10**	**11.20**
9	**2.18**	**4.80**	**2.70**	**11.20**
10	**2.47**	**4.40**	**3.00**	**11.40**
Média	**3.33**	**4.69**	**3.03**	**11.35**
Desvio-padrão	**1.38**	**0.36**	**0.25**	**0.41**

Tabela 4.11: Resultados para tabu search.

Assim como foi feito nos experimentos com simulated annealing, as topologias finais geradas por tabu search foram treinadas com o algoritmo backpropagation. Os parâmetros de treinamento foram os mesmos adotados na implementação com simulated annealing. Os resultados são mostrados na Tabela 4.12 [Ludermir and Yamazaki, 2003][Yamazaki et al., 2002c][Yamazaki and Ludermir, 2003]. Verifica-se que o treinamento com backpropagation reduziu o erro médio de classificação para 0.72%.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**1.52**	**5.30**	**2.90**	**11.60**
2	**0.20**	**4.60**	**3.20**	**12.10**
3	**0.16**	**4.50**	**3.10**	**11.60**
4	**0.31**	**5.20**	**2.50**	**12.30**
5	**1.10**	**4.60**	**3.00**	**11.60**
6	**3.16**	**5.00**	**3.10**	**11.30**
7	**0.07**	**4.60**	**2.80**	**11.10**
8	**0.20**	**4.50**	**3.20**	**10.70**
9	**0.21**	**4.80**	**3.10**	**11.90**
10	**0.26**	**4.50**	**3.10**	**11.50**
Média	**0.72**	**4.76**	**3.00**	**11.57**
Desvio-padrão	**0.98**	**0.30**	**0.22**	**0.47**

Tabela 4.12: Resultados para tabu search combinado com backpropagation.

A seguir, são explicados os detalhes da implementação do algoritmo genético, bem como os resultados obtidos.

4.6 Experimentos com Algoritmo Genético

Na implementação do algoritmo genético, foi usada uma população com 10 indivíduos, de acordo com a mesma representação de soluções discutida no Capítulo 3 e utilizada nos experimentos com simulated annealing e tabu search. O operador de mutação é o mesmo operador da metodologia proposta (também discutido no Capítulo 3), sendo que a probabilidade de haver mutação em cada peso e em cada bit de conectividade é igual a 0.1. O operador de cruzamento foi inspirado no trabalho de [Braun and Weisbrod, 1993]: para cada conexão, se os bits de conectividade dos pais forem iguais, o filho herda este bit de conectividade, sendo que seu peso é a média dos pesos dos pais; caso contrário, o filho tem o bit de conectividade aleatoriamente escolhido: se for igual a 1, seu peso vem do pai cujo bit é igual a 1, e, se for igual a zero, o peso vem do pai cujo bit é igual a zero.

A geração de uma nova população é feita da seguinte forma: selecionam-se quatro indivíduos da população atual de acordo com o método da roleta (explicado no Capítulo 2). Em seguida, o operador de cruzamento é aplicado a todos os pares possíveis de indivíduos selecionados, gerando seis indivíduos novos (um para cada par de indivíduos selecionados). Para completar a população nova, são mantidos os quatro melhores indivíduos da população original.

A quantidade máxima de iterações permitidas (I_{max}) foi igual à utilizada por tabu search (100), sendo dez vezes menor que a adotada por simulated annealing, já que o algoritmo genético, assim como tabu search, avalia um conjunto de soluções novas em cada iteração, em vez de apenas uma solução nova. Assim como acontece em experimentos com simulated annealing e tabu search, não há regras objetivas para definir os parâmetros de um algoritmo genético, de modo que os valores adotados nesta implementação podem não ser ótimos para o problema. É importante ressaltar que as escolhas foram feitas de modo a manter a maior compatibilidade possível com as implementações de simulated annealing e tabu search. Os resultados para o algoritmo genético são apresentados na Tabela 4.13.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**38.39**	**5.50**	**3.50**	**14.70**
2	**34.78**	**5.20**	**2.90**	**14.40**
3	**34.53**	**5.30**	**3.30**	**13.90**
4	**35.93**	**5.00**	**3.20**	**13.30**
5	**34.87**	**5.40**	**3.00**	**15.30**
6	**33.92**	**5.10**	**2.80**	**14.50**
7	**34.43**	**5.00**	**3.40**	**13.10**
8	**34.99**	**5.20**	**3.50**	**14.70**
9	**34.67**	**5.50**	**3.30**	**14.60**
10	**34.97**	**5.40**	**3.00**	**12.80**
Média	**35.15**	**5.26**	**3.19**	**14.13**
Desvio-padrão	**1.25**	**0.19**	**0.25**	**0.82**

Tabela 4.13: Resultados para o algoritmo genético.

O erro médio de classificação (35.15%) foi mais alto que os obtidos pelas abordagens anteriores, como será explicado na Seção 4.8. Além disso, nota-se que a quantidade média de conexões também foi maior que as obtidas pelos métodos anteriores.

Assim como foi feito para as abordagens anteriores, as topologias finais geradas pelo algoritmo genético foram treinadas com backpropagation. Os parâmetros de treinamento foram os mesmos adotados nas implementações de simulated annealing e tabu search. Os resultados são mostrados na Tabela 4.14.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**3.17**	**5.40**	**3.70**	**14.90**
2	**0.62**	**5.30**	**3.10**	**14.60**
3	**0.97**	**5.60**	**3.60**	**15.70**
4	**2.87**	**5.10**	**3.50**	**15.30**
5	**2.04**	**5.00**	**3.70**	**15.50**
6	**4.58**	**5.50**	**3.30**	**15.00**
7	**2.45**	**5.10**	**3.60**	**15.10**
8	**2.13**	**5.10**	**3.50**	**15.80**
9	**3.98**	**5.60**	**3.30**	**14.70**
10	**1.32**	**5.30**	**3.40**	**13.70**
Média	**2.41**	**5.30**	**3.47**	**15.03**
Desvio-padrão	**1.28**	**0.22**	**0.19**	**0.62**

Tabela 4.14: Resultados para o algoritmo genético combinado com backpropagation.

Observa-se que o treinamento com backpropagation conseguiu reduzir substancialmente o erro médio de classificação para 2.41%.

A seguir, são verificados os resultados da aplicação da metodologia proposta no Capítulo 3, que combina características de simulated annealing e tabu search, com o intuito de gerar um processo automático para otimizar simultaneamente arquiteturas e pesos de redes MLP.

4.7 Experimentos com a Metodologia Proposta

A metodologia proposta foi implementada utilizando os mesmos parâmetros adotados nos experimentos com simulated annealing, ou seja, a quantidade máxima de nodos escondidos (N_2) foi igual a 4, a probabilidade de inverter os bits de conectividade da solução atual (p) foi de 20%, a temperatura inicial (T_0) foi igual a 1, o fator de redução da temperatura (r) foi de 0.9, e a temperatura foi reduzida a cada 10 iterações do algoritmo (I_T). O máximo número de iterações permitidas (I_{max}) foi 100, sendo igual ao utilizado nos experimentos com tabu search e o algoritmo genético. A quantidade de soluções novas geradas em cada iteração (K) foi idêntica à usada na implementação de tabu search, sendo igual a 20.

Os resultados obtidos para a aplicação da metodologia proposta são apresentados na Tabela 4.15. Vale lembrar, novamente, que foram utilizadas as mesmas soluções iniciais empregadas nas abordagens de simulated annealing e tabu search.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**0.20**	**4.20**	**2.60**	**9.30**
2	**0.39**	**3.90**	**2.90**	**8.80**
3	**0.10**	**3.90**	**2.60**	**8.70**
4	**0.30**	**4.50**	**3.00**	**9.50**
5	**0.25**	**4.30**	**2.30**	**8.20**
6	**0.08**	**3.90**	**2.90**	**9.40**
7	**0.57**	**3.80**	**2.70**	**8.40**
8	**0.28**	**3.70**	**2.90**	**9.00**
9	**0.26**	**4.20**	**2.70**	**10.00**
10	**0.21**	**4.20**	**2.60**	**8.70**
Média	**0.26**	**4.06**	**2.72**	**9.00**
Desvio-padrão	**0.14**	**0.25**	**0.21**	**0.55**

Tabela 4.15: Resultados para a metodologia proposta.

Percebe-se que o erro médio de classificação (0.26%) foi comparável ao de tabu search combinado com backpropagation (0.72%), que foi o menor erro obtido entre as abordagens anteriores. Além disso, a quantidade média de conexões (9.00) foi menor que as obtidas por todos os métodos anteriores.

4.8 Comparações entre Abordagens

A fim de comparar os resultados obtidos para diferentes abordagens, é importante realizar testes de diferença entre médias, verificando se tais médias podem ser consideradas distintas do ponto de vista estatístico. Para obter os resultados apresentados nesta seção, foi utilizado o teste *t* de diferença entre médias populacionais para dados pareados, que está explicado no Apêndice C.

Considerando os resultados obtidos com o uso de backpropagation com os obtidos sem backpropagation, o teste de diferença entre médias concluiu que o erro médio de classificação foi menor quando o backpropagation foi utilizado. Isto foi observado para ambas as versões de simulated annealing, para tabu search e para o algoritmo genético. Dessa forma, percebe-se que os métodos globais realmente têm dificuldade em fazer ajuste fino dos pesos, apresentando facilidade para identificar as melhores regiões do espaço de busca, mas não para encontrar o ponto de mínimo destas regiões. Este fato mostra que o uso do algoritmo backpropagation para um ajuste fino dos pesos tem a capacidade de melhorar a busca realizada pelos métodos de otimização global.

Na comparação entre ambas as versões de simulated annealing, o teste obteve as seguintes conclusões sobre o erro médio de classificação:

- sem usar backpropagation, a versão que guarda a melhor solução obteve melhor desempenho;

- usando backpropagation, os desempenhos foram equivalentes.

Percebe-se que o uso de backpropagation foi capaz de tornar similares os desempenhos de duas abordagens que obtiveram erros de classificação distintos. Dessa forma, o backpropagation procurou compensar a falta do armazenamento da melhor solução durante as iterações do algoritmo.

Uma vez que a versão de simulated annealing que guarda a melhor solução foi a que apresentou o melhor desempenho, somente esta versão será considerada nas comparações entre diferentes métodos, que serão relatadas a seguir.

Comparando os resultados de simulated annealing, tabu search e o algoritmo genético sem o uso de backpropagation, o teste de diferença entre médias obteve as seguintes conclusões a respeito dos erros de classificação:

- simulated annealing obteve desempenho melhor que o do algoritmo genético;
- tabu search obteve desempenho melhor que o de simulated annealing.

A respeito do alto erro médio de classificação obtido pelo algoritmo genético (35.15%), percebe-se que, mesmo avaliando uma população de 10 soluções em uma mesma iteração, o método obteve um erro de classificação muito acima dos obtidos por simulated annealing (4.82%) e tabu search (3.33%). Tal resultado se deve, certamente, ao operador de cruzamento utilizado, já que uma rede MLP representa o conhecimento de forma distribuída entre todas as conexões. O operador de cruzamento, da forma como foi implementado, tende a destruir o conhecimento distribuído no conjunto de conexões, pois a combinação de pesos de duas redes MLP distintas não combina o conhecimento armazenado nas mesmas. Conclui-se que, para este modelo de rede neural, é importante testar outros operadores de cruzamento, ou até mesmo não utilizá-los. Vale ressaltar que simulated annealing e tabu search, assim como a metodologia proposta, não utilizam nenhum operador que combina informações de soluções distintas.

Vale ressaltar, ainda, que o uso de backpropagation conseguiu diminuir consideravelmente o erro médio de classificação obtido pelo algoritmo genético (de 35.15% para 2.41%), de modo a compensar a ineficiência provocada pelo uso de um operador de cruzamento inadequado.

A respeito da superioridade de tabu search sobre simulated annealing, é importante lembrar que a implementação de tabu search avalia 20 soluções novas a cada iteração, enquanto simulated annealing avalia apenas uma, exigindo uma quantidade maior de iterações. Foram permitidas 1.000 iterações, mas esta quantidade pode não ter

sido suficiente para haver uma exploração satisfatória no espaço de busca. Por este motivo, tabu search teve mais oportunidades de encontrar soluções melhores, avaliando várias soluções novas por iteração.

Considerando as diferentes abordagens que utilizaram backpropagation (simulated annealing, tabu search, algoritmo genético e a metodologia proposta), foram obtidas as mesmas conclusões observadas nos resultados sem backpropagation para o erro de classificação, com as seguintes constatações adicionais:

- a metodologia proposta obteve desempenho melhor que o obtido por simulated annealing e pelo algoritmo genético;
- a metodologia proposta obteve desempenho equivalente ao de tabu search.

A respeito da quantidade de conexões, as conclusões foram as seguintes:

- simulated annealing e tabu search obtiveram uma quantidade de conexões menor que a do algoritmo genético;
- simulated annealing obteve uma quantidade de conexões equivalente à de tabu search;
- a metodologia proposta obteve uma quantidade de conexões menor que a de simulated annealing, de tabu search e do algoritmo genético.

Percebe-se que, para esta base de dados, a avaliação de um conjunto de soluções novas (em vez de uma única solução nova), que é característica de tabu search, permitiu à metodologia proposta obter desempenho de classificação superior ao obtido por simulated annealing. O fato de ter obtido desempenho de classificação equivalente ao de tabu search mostra que o uso de uma probabilidade de aceitação de soluções piores, em substituição à lista tabu adotada por tabu search, não produziu melhoria considerável no que se refere ao erro de classificação. Entretanto, houve uma melhoria significativa no que se refere à quantidade de conexões.

Tal constatação pode ser explicada da seguinte forma: na geração de soluções novas, é pouco provável que seja criado um conjunto de pesos já usado recentemente,

sendo mais provável gerar topologias já utilizadas anteriormente, uma vez que os pesos são contínuos (números reais), enquanto a topologia é discreta (conjunto de bits). Dessa forma, na otimização com tabu search, deve ter havido muitas repetições de topologias, de modo que o espaço de topologias não foi suficientemente explorado. A implementação da lista tabu permitiu este acontecimento, já que uma solução só era considerada proibida se coincidisse totalmente com uma das soluções tabu (em pesos e topologia), de modo que dificilmente uma solução era proibida, mesmo que a topologia fosse repetida. Assim, por explorar melhor o espaço de topologias, a metodologia proposta conseguiu uma quantidade de conexões menor que a obtida por tabu search. Como este aspecto não foi crítico para o ajuste dos pesos, o erro de classificação da metodologia proposta foi equivalente ao de tabu search.

4.9 Seleção de Atributos de Entrada

Uma vez que as técnicas implementadas neste trabalho são capazes de eliminar unidades de entrada das redes MLP, é importante analisar quais atributos de entrada são descartados e quais deles se mostram mais relevantes na tarefa de classificação, facilitando o estudo do problema abordado.

Dessa forma, foram calculados, para cada método implementado, os percentuais de utilização de cada entrada nas topologias otimizadas, como mostra a Tabela 4.16.

Método	**Entradas**					
	1	**2**	**3**	**4**	**5**	**6**
Simulated annealing	**86.67**	**72.00**	**82.00**	**68.67**	**83.67**	**68.33**
Tabu Search	**90.00**	**70.33**	**84.67**	**74.67**	**82.00**	**63.00**
Algoritmo genético	**88.33**	**85.67**	**91.00**	**88.33**	**90.00**	**79.33**
Metodologia proposta	**91.33**	**59.33**	**80.00**	**57.67**	**80.67**	**52.33**

Tabela 4.16: Percentual de utilização das entradas nas topologias otimizadas.

Percebe-se que, para cinco entradas (2, 3, 4, 5 e 6), o método que mais as utilizou foi o algoritmo genético, de modo que as outras abordagens conseguiram obter uma maior quantidade de topologias que não utilizam estas entradas. A metodologia proposta foi a abordagem que menos utilizou estas cinco entradas, conseguindo obter mais topologias que não precisavam destes atributos. Excluindo o algoritmo genético, a entrada 1 foi a mais utilizada pelos demais métodos, sugerindo que este atributo tem a maior contribuição na tarefa de classificação, ou seja, este sensor certamente obteve respostas cujas diferenças entre as safras de vinho foram mais facilmente aprendidas pelas redes MLP. Para todas as abordagens, o menor percentual de utilização foi referente à entrada 6, levando a crer que este atributo é o que menos contribui na distinção entre padrões de classes distintas, pois as respostas deste sensor não variaram substancialmente entre as três safras de vinho.

4.10 Comentários Finais

Os resultados mostram que a implementação de uma metodologia combinando as principais características favoráveis dos algoritmos de simulated annealing e tabu search, além de fazer uso de um algoritmo local de treinamento, é capaz de gerar um sistema híbrido bastante eficiente para otimização de arquiteturas e pesos de redes MLP para o problema abordado.

Considerando o domínio do reconhecimento de odores em um nariz artificial, a quantidade média de conexões foi muito menor do que o máximo permitido. Esta constatação é muito importante em diversas aplicações, como a implementação em hardware de redes neurais para a construção de narizes artificiais. Além disso, os resultados apresentados possibilitam conclusões interessantes sobre a importância de cada atributo de entrada, representando a contribuição de cada sensor na distinção entre as safras de vinho.

No próximo capítulo, são apresentados os resultados obtidos para o problema do diagnóstico de diabetes, com o intuito de avaliar o desempenho da metodologia proposta em um outro domínio.

Capítulo 5

Resultados Obtidos para o Diagnóstico de Diabetes

5.1 Introdução

Este capítulo apresenta resultados da aplicação da metodologia proposta para um outro conjunto de dados, a fim de verificar o desempenho em outro domínio e comparar a presente abordagem com outros trabalhos encontrados na literatura sobre otimização de redes MLP. A base de dados, que trata do diagnóstico de diabetes, foi escolhida de modo a permitir comparações diretas com outros sistemas propostos na literatura. Os dados são bastante conhecidos na área de aprendizado de máquina e já foram abordados por diversos trabalhos com as mais variadas técnicas de classificação. Tal fato é muito

importante para comparar a metodologia proposta com outros trabalhos, tendo em vista a consolidação da abordagem desenvolvida na comunidade científica.

5.2 Problema e Base de Dados

O problema trata da classificação de índias Pima em diabéticas ou não-diabéticas, com base em dados pessoais, como idade e número de filhos, e resultados de exames médicos, como índice de massa corporal e pressão sangüínea. A base de dados utilizada foi extraída do conhecido *UCI Machine Learning Repository* [Blake and Merz, 1998]. Cada padrão é formado por 8 atributos de entrada. O conjunto de dados possui um total de 768 exemplos, sendo que 500 pertencem à classe das não-diabéticas (65.10%). Este problema é considerado difícil por conta da pequena quantidade de exemplos e do alto nível de ruído [Yao and Liu, 1997]. Tal ruído se deve à presença de valores ausentes [Prechelt, 1994]. Dessa forma, a base de dados se mostra bastante interessante para verificar o desempenho da metodologia proposta no presente trabalho.

Um dos métodos que já foram utilizados para abordar este problema é o *EPNet*, proposto em [Yao and Liu, 1997], o qual faz uso de programação evolucionária para a definição de arquiteturas MLP. É utilizada a representação direta das arquiteturas e dos pesos, e cada solução é avaliada por um erro calculado sobre o conjunto de validação.

Na implementação deste sistema para o problema de diabetes, os primeiros 384 exemplos foram usados no conjunto de treinamento, os 192 exemplos seguintes foram empregados no conjunto de validação, e os 192 exemplos finais foram destinados ao conjunto de teste.

A técnica aplica diferentes mutações em uma determinada seqüência. Inicialmente, a arquitetura MLP é treinada com backpropagation. Se o erro não for reduzido significativamente, assume-se que a rede estacionou em um mínimo local, e, por este motivo, a rede é treinada por simulated annealing. Se este segundo treinamento também não conseguir uma redução significativa no erro, inicia-se a seqüência de mutações na arquitetura. Em primeiro lugar, testa-se a eliminação de nodos e conexões. Se esta tentativa não for bem sucedida, o algoritmo passa a adicionar nodos e conexões.

Percebe-se que a seqüência de mutações empregada pelo algoritmo tenta priorizar a manutenção da arquitetura inicial, de modo que o treinamento dos pesos nesta arquitetura fixa é a primeira etapa a ser testada. Apenas se esta tentativa não obtiver sucesso, a arquitetura é alterada. A ordem das mutações na arquitetura dá preferência a arquiteturas pequenas, uma vez que a primeira mutação realizada é a eliminação de nodos e conexões. Se não for possível melhorar o desempenho da rede com esta etapa de eliminação, tenta-se adicionar nodos e conexões. Mais detalhes sobre o método podem ser encontrados em [Yao and Liu, 1997].

Nos experimentos relatados em [Yao and Liu, 1997], o algoritmo *EPNet* foi executado 30 vezes, a partir de diferentes inicializações de pesos. Obteve-se um erro médio de classificação para o conjunto de teste igual a 22.38% (com desvio-padrão de 0.014%), sendo que a quantidade média de conexões nas redes geradas foi de 52.3 (com desvio-padrão de 16.1).

Este conjunto de dados também foi abordado pelo método *CNNDA* (*cascade neural network design algorithm*), proposto em [Islam and Murase, 2001], que foi desenvolvido para definir arquiteturas MLP compactas com duas camadas escondidas.

O algoritmo inicia a definição da arquitetura MLP de forma construtiva, adicionando unidades sucessivamente e verificando a variação no desempenho da rede quando treinada com backpropagation. Quando a rede atinge uma convergência considerada satisfatória, o algoritmo começa a reduzir a arquitetura, eliminando unidades e conexões. A fim de reduzir o tempo de treinamento, usa-se uma nova abordagem, que "congela" temporariamente os pesos de entrada de uma unidade escondida quando sua saída não se altera muito após algumas iterações sucessivas. Maiores detalhes sobre o algoritmo podem ser encontrados em [Islam and Murase, 2001].

Nos experimentos com o problema do diagnóstico de diabetes, foram feitas 30 execuções do algoritmo, partindo de diferentes inicializações de pesos, de acordo com o mesmo procedimento adotado nos experimentos relatados em [Yao and Liu, 1997] para o *EPNet*. O particionamento dos dados nos conjuntos de treinamento, validação e teste também foi mantido. O erro médio de classificação obtido pelo algoritmo *CNNDA* foi

19.91%, e a quantidade média de conexões nas redes geradas foi 40.4. Os valores de desvio-padrão não foram relatados em [Islam and Murase, 2001].

5.3 Experimentos

A fim de possibilitar uma comparação direta, simulated annealing, tabu search, algoritmo genético e a metodologia proposta foram implementados usando o mesmo particionamento de dados adotado nos trabalhos de [Yao and Liu, 1997] e [Islam and Murase, 2001]. Os valores escolhidos para os parâmetros de implementação de cada técnica foram os mesmos utilizados nos experimentos com a base de dados anterior (odores de safras de vinho), exceto no que se refere à topologia máxima, que teve 8 nodos de entrada (N_1), 10 nodos escondidos (N_2) e 2 nodos de saída (N_3), e no tocante ao número máximo de iterações permitidas (I_{max}), que foi igual a 5.000 para simulated annealing e 500 para as demais abordagens. Tais parâmetros foram escolhidos empiricamente. Vale lembrar que, nos experimentos do Capítulo 4, a quantidade máxima de iterações de simulated annealing, que avalia apenas uma solução nova em cada iteração, foi 10 vezes maior que as utilizadas nos outros métodos, os quais avaliam um conjunto de soluções novas em cada iteração.

Vale ressaltar que, para o problema do diagnóstico de diabetes, só foi implementada a versão de simulated annealing que guarda a melhor solução encontrada durante a otimização, já que esta versão foi a que gerou os melhores resultados apresentados no capítulo anterior.

Assim como foi feito nos experimentos de [Yao and Liu, 1997] e [Islam and Murase, 2001], neste trabalho foram realizadas 30 execuções de cada método implementado, partindo de inicializações distintas e aleatórias de pesos. Diferentemente dos resultados do Capítulo 4, não foi eliminada nenhuma execução, de modo que os valores de média e desvio-padrão foram calculados a partir dos resultados das 30 execuções.

A Tabela 5.1 apresenta os resultados obtidos por simulated annealing sem backpropagation. Observa-se que o erro médio de classificação foi de 26.09%. Percebe-se, também que as quantidades médias de nodos escondidos (4.17) e de conexões (27.07) foram substancialmente menores que as quantidades máximas permitidas (10 e

100, respectivamente). Entretanto, o mesmo não pode ser dito a respeito da quantidade média de nodos de entrada (7.60), já que a maioria das redes obtidas utilizou todas as 8 entradas do problema.

Vale ressaltar que, similarmente ao que foi feito no Capítulo 4, os resultados desta seção são brevemente comentados, uma vez que as comparações mais abrangentes serão apresentadas na Seção 5.4, de modo a fornecer um respaldo estatístico às conclusões obtidas através do teste de diferenças entre médias.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	32.29	8	2	25
2	26.04	8	2	26
3	24.48	8	4	31
4	22.92	8	4	24
5	23.44	7	7	27
6	26.56	7	3	24
7	25.52	7	5	25
8	32.81	7	6	27
9	27.08	8	4	25
10	27.08	8	3	34
11	24.48	7	5	24
12	28.65	7	5	24
13	26.56	8	4	27
14	27.08	8	5	24
15	26.04	8	3	30
16	21.88	7	5	26
17	24.48	8	2	30
18	21.88	8	2	25
19	36.46	8	6	25
20	25.00	8	5	28
21	23.96	8	5	30
22	25.00	7	4	28
23	28.13	8	3	26
24	25.52	8	5	27
25	26.56	6	5	26
26	29.69	8	5	33
27	19.79	8	5	23
28	25.00	7	5	28
29	23.96	8	3	30
30	24.48	7	3	30
Média	**26.09**	**7.60**	**4.17**	**27.07**
Desvio-padrão	**3.40**	**0.56**	**1.32**	**2.84**

Tabela 5.1: Resultados para simulated annealing (problema do diagnóstico de diabetes).

Da mesma forma como foi feito no Capítulo 4, as redes finais geradas por simulated annealing foram treinadas com backpropagation para ajuste fino dos pesos. Os resultados estão na Tabela 5.2.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	25.52	8	2	25
2	27.08	8	2	26
3	20.31	8	4	31
4	22.40	8	4	24
5	20.31	7	7	27
6	24.48	7	3	24
7	21.35	7	5	25
8	20.83	7	6	27
9	27.60	8	4	25
10	22.92	8	3	34
11	21.35	7	5	24
12	32.81	7	5	24
13	21.35	8	4	27
14	63.54	8	5	24
15	28.13	8	3	30
16	18.23	7	5	26
17	20.83	8	2	30
18	21.35	8	2	25
19	39.58	8	6	25
20	20.83	8	5	28
21	24.48	8	5	30
22	22.92	7	4	28
23	23.96	8	3	26
24	20.83	8	5	27
25	26.04	6	5	26
26	22.40	8	5	33
27	19.79	8	5	23
28	21.88	7	5	28
29	21.88	8	3	30
30	22.40	7	3	30
Média	**24.91**	**7.60**	**4.17**	**27.07**
Desvio-padrão	**8.45**	**0.56**	**1.32**	**2.84**

Tabela 5.2: Resultados para simulated annealing combinado com backpropagation (problema do diagnóstico de diabetes).

Como pode ser observado, o uso de backpropagation reduziu o erro médio de classificação para 24.91%.

A Tabela 5.3 mostra os resultados obtidos por tabu search sem o uso de backpropagation.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	32.81	7	5	31
2	27.08	7	6	32
3	27.60	8	6	27
4	27.08	8	7	30
5	23.96	8	5	29
6	22.40	8	4	32
7	27.08	8	4	24
8	20.31	7	5	31
9	29.17	8	6	29
10	29.69	8	5	28
11	21.35	8	3	31
12	20.31	8	5	30
13	30.73	8	8	29
14	26.04	8	3	30
15	30.73	6	9	27
16	23.96	8	4	31
17	30.73	8	5	27
18	22.92	7	5	31
19	25.00	7	6	31
20	34.38	8	5	25
21	31.25	8	5	30
22	23.44	8	6	31
23	20.31	8	7	32
24	26.04	7	6	27
25	19.27	7	4	27
26	30.73	8	4	30
27	27.08	8	4	29
28	20.31	8	6	30
29	19.79	8	5	26
30	28.65	8	5	28
Média	**26.01**	**7.70**	**5.27**	**29.17**
Desvio-padrão	**4.33**	**0.53**	**1.34**	**2.13**

Tabela 5.3: Resultados para tabu search (problema do diagnóstico de diabetes).

Observa-se que o erro médio de classificação (26.01%) foi semelhante ao obtido por simulated annealing sem backpropagation (26.09%), constatação que foi confirmada

por um teste de diferença entre médias, como será explicado na Seção 5.4. Sobre as quantidades de entradas, nodos escondidos e conexões, foram feitas as mesmas conclusões observadas nos resultados de simulated annealing: os números de nodos escondidos e conexões foram significativamente menores do que os máximos permitidos, mas isso não foi constatado na quantidade de nodos de entrada.

A Tabela 5.4 mostra os resultados obtidos para tabu search combinado com backpropagation.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	22.92	7	5	31
2	22.92	7	6	32
3	26.56	8	6	27
4	23.96	8	7	30
5	24.48	8	5	29
6	23.44	8	4	32
7	27.08	8	4	24
8	21.35	7	5	31
9	30.21	8	6	29
10	23.44	8	5	28
11	21.35	8	3	31
12	24.48	8	5	30
13	23.96	8	8	29
14	29.17	8	3	30
15	27.60	6	9	27
16	24.48	8	4	31
17	30.73	8	5	27
18	22.92	7	5	31
19	22.40	7	6	31
20	22.40	8	5	25
21	30.21	8	5	30
22	22.40	8	6	31
23	20.83	8	7	32
24	26.04	7	6	27
25	19.27	7	4	27
26	28.13	8	4	30
27	26.04	8	4	29
28	20.83	8	6	30
29	27.08	8	5	26
30	29.69	8	5	28
Média	**24.88**	**7.70**	**5.27**	**29.17**
Desvio-padrão	**3.17**	**0.53**	**1.34**	**2.13**

Tabela 5.4: Resultados para tabu search combinado com backpropagation (problema do diagnóstico de diabetes).

Pode ser observado que o treinamento com backpropagation reduziu o erro médio de classificação para um valor (24.88%) muito próximo do obtido por simulated annealing combinado com backpropagation (24.91%), como será explicado adiante.

A Tabela 5.5 mostra os resultados da aplicação do algoritmo genético ao problema abordado.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	**35.94**	**7**	**5**	**38**
2	**34.90**	**8**	**9**	**38**
3	**30.73**	**8**	**7**	**40**
4	**36.46**	**8**	**4**	**35**
5	**36.46**	**8**	**6**	**42**
6	**31.25**	**8**	**4**	**39**
7	**43.23**	**8**	**7**	**35**
8	**36.46**	**8**	**8**	**39**
9	**36.98**	**8**	**7**	**38**
10	**30.21**	**8**	**5**	**38**
11	**34.38**	**8**	**7**	**41**
12	**36.46**	**8**	**9**	**43**
13	**36.46**	**8**	**6**	**41**
14	**38.02**	**8**	**6**	**37**
15	**32.81**	**8**	**7**	**49**
16	**31.25**	**8**	**7**	**45**
17	**33.85**	**8**	**7**	**40**
18	**36.46**	**8**	**2**	**37**
19	**26.04**	**8**	**7**	**38**
20	**33.85**	**8**	**3**	**36**
21	**39.06**	**8**	**8**	**38**
22	**29.17**	**8**	**8**	**46**
23	**36.46**	**8**	**6**	**38**
24	**32.29**	**8**	**7**	**43**
25	**33.85**	**8**	**6**	**41**
26	**35.94**	**8**	**7**	**38**
27	**23.96**	**8**	**7**	**44**
28	**36.98**	**8**	**6**	**36**
29	**35.42**	**8**	**4**	**37**
30	**35.42**	**8**	**5**	**37**
Média	**34.36**	**7.97**	**6.23**	**39.57**
Desvio-padrão	**3.84**	**0.18**	**1.65**	**3.37**

Tabela 5.5: Resultados para o algoritmo genético (problema do diagnóstico de diabetes).

A abordagem do algoritmo genético obteve erro médio de classificação (34.36%) mais alto que os obtidos pelos métodos anteriores, reforçando a observação feita no Capítulo 4 sobre o uso inadequado do operador de cruzamento. Além disso, observa-se que a quantidade média de conexões (39.57) também foi mais alta do que as obtidas pelas técnicas anteriores.

Com o intuito de verificar se o desempenho de classificação pode ser melhorado com o ajuste dos pesos, as redes finais geradas pelo algoritmo genético foram treinadas com backpropagation, obtendo os resultados da Tabela 5.6.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	27.60	7	5	38
2	21.35	8	9	38
3	21.35	8	7	40
4	23.96	8	4	35
5	21.35	8	6	42
6	22.92	8	4	39
7	26.56	8	7	35
8	20.31	8	8	39
9	22.40	8	7	38
10	27.60	8	5	38
11	20.83	8	7	41
12	19.27	8	9	43
13	23.96	8	6	41
14	20.31	8	6	37
15	18.75	8	7	49
16	20.83	8	7	45
17	20.83	8	7	40
18	43.23	8	2	37
19	20.31	8	7	38
20	19.79	8	3	36
21	22.92	8	8	38
22	20.83	8	8	46
23	21.88	8	6	38
24	20.83	8	7	43
25	20.31	8	6	41
26	25.00	8	7	38
27	21.88	8	7	44
28	35.94	8	6	36
29	25.00	8	4	37
30	23.96	8	5	37
Média	23.40	7.97	6.23	39.57
Desvio-padrão	5.06	0.18	1.65	3.37

Tabela 5.6: Resultados para o algoritmo genético combinado com backpropagation (problema do diagnóstico de diabetes).

Percebe-se que o uso de backpropagation conseguiu reduzir significativamente o erro médio de classificação, tornando-o semelhante aos obtidos por simulated annealing e tabu search, apesar da maior quantidade de conexões.

Aplicando a metodologia proposta, foram obtidos os resultados da Tabela 5.7.

Inicialização de Pesos	Erro de Classificação para o Conjunto de Teste (%)	Quantidade de Nodos de Entrada	Quantidade de Nodos Escondidos	Quantidade de Conexões
1	21.88	8	6	28
2	22.40	8	6	25
3	20.31	7	3	24
4	24.48	6	6	24
5	20.83	8	8	25
6	21.35	8	4	24
7	22.40	8	4	28
8	23.96	8	5	26
9	21.35	6	4	23
10	28.65	7	2	21
11	22.92	7	5	22
12	23.96	7	5	23
13	27.08	7	3	22
14	26.56	8	4	26
15	26.56	8	4	23
16	20.83	8	7	27
17	28.65	8	4	27
18	25.52	8	3	23
19	19.79	7	6	23
20	21.88	7	7	31
21	22.92	7	5	28
22	21.88	8	4	24
23	22.40	8	6	28
24	28.13	7	2	22
25	27.08	7	5	23
26	21.88	8	5	28
27	21.35	8	5	29
28	26.56	8	4	24
29	20.83	8	3	24
30	20.83	8	5	26
Média	**23.51**	**7.53**	**4.67**	**25.03**
Desvio-padrão	**2.72**	**0.63**	**1.45**	**2.50**

Tabela 5.7: Resultados para a metodologia proposta (problema do diagnóstico de diabetes).

Pode ser observado que o erro médio de classificação (23.51%) foi semelhante aos obtidos pelas abordagens anteriores com backpropagation.

Considerando que os resultados da abordagem *EPNet* [Yao and Liu, 1997] não foram obtidos a partir das mesmas inicializações de pesos utilizadas no presente trabalho, foi necessário empregar um teste de diferença entre médias populacionais para dados não-pareados, que está explicado no Apêndice C. Através deste teste, foi possível comparar os resultados obtidos pela metodologia proposta com aqueles obtidos por [Yao and Liu, 1997] utilizando os valores de média e desvio-padrão. Infelizmente, não foi possível realizar o teste para os resultados obtidos pela abordagem *CNNDA* [Islam and Murase, 2001], pois não foram informados os valores de desvio-padrão.

5.4 Comparações entre Abordagens

A fim de comparar os resultados obtidos por diferentes abordagens do presente trabalho, empregou-se o mesmo teste de diferença entre médias para dados pareados utilizado no Capítulo 4 para os dados do nariz artificial. Vale lembrar que todas as conclusões referentes aos resultados dos testes de diferença entre médias se referem ao nível de significância de 5%.

Comparando as abordagens implementadas sem o uso de backpropagation (simulated annealing, tabu search e algoritmo genético), o teste de diferença entre médias obteve as seguintes conclusões a respeito do erro de classificação:

- simulated annealing e tabu search obtiveram desempenhos melhores que o do algoritmo genético;

- simulated annealing e tabu search obtiveram desempenhos equivalentes entre si.

Comparando, agora, as abordagens implementadas com o uso de backpropagation (simulated annealing, tabu search, algoritmo genético e a metodologia proposta), o teste concluiu que todas as abordagens obtiveram erros de classificação equivalentes. Esta conclusão mostra que o uso de backpropagation para ajuste fino dos pesos foi capaz de melhorar o desempenho do algoritmo genético, de modo a torná-lo equivalente aos de simulated annealing e tabu search. Além disso, o desempenho da metodologia proposta também foi equivalente aos de simulated annealing e tabu search.

Entretanto, foram observadas diferenças na quantidade média de conexões obtidas. Sobre este aspecto, o teste concluiu que:

- simulated annealing e tabu search obtiveram quantidades menores que a do algoritmo genético;

- a metodologia proposta obteve quantidade menor que a de tabu search e do algoritmo genético, sendo equivalente à obtida por simulated annealing.

Dessa forma, percebe-se que, embora o uso de backpropagation tenha conseguido tornar equivalentes os erros médios de classificação, as menores topologias (em termos do número de conexões) foram obtidas por simulated annealing e pela metodologia proposta.

As observações anteriores levam à seguinte conclusão: se for levado em consideração apenas o erro de classificação, as quatro abordagens implementadas para esta base de dados são igualmente eficientes, mas, se o objetivo for minimizar a quantidade de conexões, a metodologia proposta e o método de simulated annealing conseguem obter melhores resultados.

Os resultados da metodologia proposta também foram comparados com os obtidos pela abordagem *EPNet* [Yao and Liu, 1997], através do teste de diferença entre médias para dados não-pareados explicado no Apêndice C. Concluiu-se que o erro de classificação e a quantidade de conexões da metodologia proposta foram equivalentes aos do método *EPNet*.

Como já foi explicado, não foi possível realizar um teste de diferença entre médias para a abordagem *CNNDA* [Islam and Murase, 2001]. Avaliando apenas os valores médios obtidos, o erro de classificação da metodologia proposta (23.51%) ficou um pouco acima do erro obtido por *CNNDA* (19.91%), entretanto a quantidade média de conexões utilizadas pela metodologia proposta (25.03) foi bem menor que a obtida por *CNNDA* (40.4), indicando que, em média, a metodologia proposta conseguiu obter redes com menos conexões, sem deteriorar muito o desempenho de classificação. Evidentemente, esta conclusão não possui respaldo estatístico, diferentemente do que foi observado nas constatações anteriores.

5.5 Seleção de Atributos de Entrada

Da mesma forma como foi apresentado no Capítulo 4, o percentual de utilização das entradas foi analisado com o intuito de estudar a seleção de atributos realizada pelas abordagens implementadas, gerando os resultados da Tabela 5.8.

Método	Entradas							
	1	2	3	4	5	6	7	8
Simulated annealing	**100.00**	**96.67**	**96.67**	**86.67**	**96.67**	**96.67**	**96.67**	**90.00**
Tabu search	**93.33**	**100.00**	**96.67**	**96.67**	**93.33**	**93.33**	**100.00**	**96.67**
Algoritmo genético	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**96.67**	**100.00**	**100.00**
Metodologia proposta	**90.00**	**100.00**	**100.00**	**90.00**	**93.33**	**93.33**	**93.33**	**93.33**

Tabela 5.8: Percentual de utilização das entradas nas topologias otimizadas para o problema do diagnóstico de diabetes.

Como pode ser observado, o menor percentual de utilização obtido foi 86.67%, mostrando que cada entrada foi utilizada com muita freqüência entre as topologias geradas por todas as abordagens implementadas. Dessa forma, as diferenças entre as contribuições dos atributos não ficaram muito claras na análise das topologias obtidas, que, em sua maioria, utilizaram todas as oito unidades de entrada possíveis. A maioria das redes MLP necessitou de todas as informações contidas nos oito atributos por conta dos valores ausentes na base de dados, de modo que nenhum atributo foi consideravelmente desprezado na tarefa de classificação. Os valores ausentes tornam o problema difícil, reduzindo a disponibilidade de informação necessária para distinguir as classes. Por este motivo, as topologias não tenderam a descartar nenhum dos atributos, pois, em geral, isto só acontece quanto existe informação irrelevante e desnecessária em alguns atributos. Para esta base de dados, ocorre justamente o contrário: faltam informações nos dados para que a tarefa de classificação seja desempenhada com erros baixos. Esta constatação é reforçada pelos erros de

classificação obtidos por [Yao and Liu, 1997] e [Islam and Murase, 2001], que foram semelhantes aos obtidos pela metodologia proposta no presente trabalho.

5.6 Comentários Finais

Os resultados apresentados mostram que a metodologia proposta consegue obter resultados comparáveis aos encontrados na literatura para o problema do diagnóstico de diabetes. Foi utilizado um conjunto de dados bastante conhecido na comunidade científica, permitindo uma comparação direta com outros trabalhos.

Este problema se mostrou mais difícil do que a classificação de odores apresentada no Capítulo 4, pois os menores erros de classificação já obtidos para o diagnóstico de diabetes na literatura ainda estão muito acima dos erros que podem ser obtidos para os odores das safras de vinho, mostrando que a existência de valores ausentes em uma base de dados com poucos padrões aumenta a complexidade do problema.

Apesar disso, a abordagem proposta conseguiu resultados comparáveis aos obtidos por dois outros sistemas encontrados na literatura para os mesmos dados, tanto no que se refere ao erro de classificação quanto no tocante à quantidade de conexões utilizadas. O próximo capítulo reforça estas conclusões, além de comentar diversas possibilidades de atividades futuras que o presente trabalho origina.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Considerando o que foi apresentado, a principal conclusão que pode ser extraída do presente trabalho é que a metodologia proposta no Capítulo 3 se mostra bastante eficiente, pois combina as principais características favoráveis de simulated annealing e tabu search, evitando suas limitações. A escolha da representação das soluções, da função de custo e do operador para gerar novas soluções apresentou bons resultados, bem como o treinamento das topologias finais usando o conhecido algoritmo backpropagation, em uma abordagem de treinamento híbrido.

A aplicação de simulated annealing e tabu search é capaz de diminuir a quantidade de conexões das redes MLP. Como foi visto no Capítulo 2, que procurou

fornecer uma visão geral da aplicação dos principais métodos de busca global na otimização de redes neurais, o uso de algoritmos genéticos na otimização simultânea de arquiteturas e pesos já rendeu alguns trabalhos interessantes [Branke, 1995][Yao, 1999], entretanto não se encontram abordagens objetivando o emprego de outras técnicas tradicionais, como simulated annealing e tabu search, e esta foi uma das contribuições dos resultados apresentados.

Outra conclusão importante é que simulated annealing e tabu search, assim como algoritmos genéticos, podem ser aplicados com sucesso em abordagens de treinamento híbrido, sendo combinados com técnicas locais de gradiente, que passam a ter o objetivo de proporcionar um ajuste mais fino dos pesos das redes finais. Podem ser encontrados na literatura trabalhos interessantes de treinamento híbrido usando algoritmos genéticos [Branke, 1995][Yao, 1999], mas, até o presente momento, não se encontram propostas que utilizam simulated annealing e tabu search para o mesmo objetivo.

Este trabalho também procurou contribuir na abordagem do reconhecimento de odores em um nariz artificial, que tem merecido crescente atenção da comunidade científica devido ao grande número de aplicações em diversas áreas que necessitam da identificação inteligente de odores em dispositivos automáticos.

Além disso, foi abordado um problema conhecido na comunidade científica (diagnóstico de diabetes), a fim de gerar resultados da aplicação da metodologia proposta em dados já utilizados por outros autores, consolidando a contribuição do presente trabalho.

Uma limitação observada na metodologia proposta foi o fato de a mesma não ser capaz de realizar uma seleção de atributos considerável quando falta informação na base de dados para que a tarefa de classificação seja desempenhada com erros baixos, como foi observado nos resultados para o diagnóstico de diabetes. Isto ocorre porque a eliminação não é feita levando em consideração cada nodo de entrada, de modo que a eliminação de uma unidade de entrada só acontece se forem eliminadas todas as conexões que partem da mesma. Dessa forma, a exclusão de um atributo ocorre como conseqüência do processo de redução de conexões. A metodologia proposta não verifica

a contribuição de cada atributo de entrada durante a otimização, pois a ênfase está na contribuição de cada conexão da rede.

6.2 Trabalhos Futuros

Este trabalho origina diversas perspectivas futuras, entre as quais podem ser destacadas as seguintes possibilidades:

- aplicar a metodologia proposta em um novo conjunto de dados, ampliando o estudo comparativo apresentado neste trabalho;

- aplicar a metodologia proposta para otimizar redes MLP com outras funções de ativação além da tangente hiperbólica (por exemplo, funções não-diferenciáveis);

- desenvolver metodologias semelhantes para otimizar outros modelos de redes neurais além do MLP (por exemplo, redes de função de base radial (RBF) [Moody and Darken, 1989]);

- estudar a viabilidade de implementar em hardware as redes MLP otimizadas pela metodologia proposta para a classificação de odores em um nariz artificial, tendo em vista a miniaturização do dispositivo.

Além destas possibilidades, existem várias outras extensões para o presente trabalho. Por exemplo, é importante investigar o comportamento da metodologia proposta quando o problema a ser resolvido aumenta em complexidade, ou seja, verificar como ocorre o escalonamento para dimensões maiores. Tal aspecto é importante para investigar se a metodologia continua gerando resultados satisfatórios mesmo para problemas de maior complexidade ou se existe um limite de complexidade acima do qual a metodologia não consegue explorar o espaço de busca de modo a apresentar resultados satisfatórios.

Outro trabalho futuro é realizar simulações através de experiências Monte-Carlo, com o intuito de verificar o comportamento da metodologia proposta de forma sistemática e controlada. Tal abordgem permitiria, por exemplo, investigar os resultados

da metodologia em casos extremos controlados, gerando um estudo mais abrangente sobre o funcionamento.

Também é importante estudar os resultados apresentados neste trabalho diferenciando os tipos de erros para cada problema. Sabe-se que, para determinadas aplicações práticas, não é suficiente obter apenas uma taxa de erro de classificação, sendo necessário discriminar os tipos de erros mais freqüentes (por exemplo, diferenciando erros do tipo "falso negativo" e do tipo "falso positivo"). Dessa forma, é interessante ampliar a apresentação dos resultados, de modo a fornecer mais indícios sobre o comportamento da metodologia proposta nos diferentes domínios.

Outra análise importante é o estudo do desempenho da metodologia proposta em termos de tempo de execução. Tal aspecto merece uma atenção especial, sendo importante comparar o desempenho temporal da metodologia proposta com outras técnicas, a fim de verificar se a melhoria nos resultados de erro de classificação e complexidade topológica exigem um custo alto de tempo de execução. Vale ressaltar, ainda, a importância de analisar os tempos de execução separando a fase de otimização global da fase de refinamento local, de modo a permitir conclusões sobre o desempenho temporal em cada uma das fases de execução.

É importante, também, modificar aspectos de implementação da metodologia proposta, como, por exemplo, a representação das soluções. Neste caso, uma possibilidade é codificar apenas os valores dos pesos da rede (eliminando os bits de conectividade), considerando inexistentes as conexões de peso nulo. Dessa forma, o tamanho da representação diminui, podendo gerar um ganho em termos de eficiência, entretanto o valor do peso é perdido quando a conexão desaparece, não podendo ser reaproveitado posteriormente, o que pode deteriorar o aprendizado. Percebe-se que a modificação da representação merece ser mais profundamente estudada, juntamente com a variação de outros aspectos de implementação, como a função de custo, o operador e o esquema de redução da temperatura. Uma possibilidade de alteração na função de custo seria o uso de uma média ponderada entre o erro de classificação e o tamanho da arquitetura, em vez de uma média aritmética simples. Esta abordagem permitiria controlar a importância do erro e da complexidade topológica na atribuição do custo, de forma a guiar o processo de otimização. Por outro lado, esta opção geraria

dois novos parâmetros a serem ajustados na implementação (os pesos da média ponderada). Sobre o operador, poderiam ser testadas outras formas de gerar novas soluções a partir da atual, de modo a explorar o espaço de busca mais eficientemente, e, em relação ao esquema de redução da temperatura, poderiam ser empregadas outras funções de atualização, tendo sempre em vista o compromisso entre uma convergência eficiente e a velocidade de execução.

Uma outra possibilidade é estender os resultados apresentados neste trabalho para valores diferentes dos parâmetros ajustáveis, como a temperatura inicial, o fator de redução da temperatura, a quantidade de soluções geradas em cada iteração, o número máximo de iterações, entre outros, a fim de verificar a sensibilidade dos resultados em relação à escolha destes parâmetros para cada domínio de aplicação. Tal estudo possibilitaria investigar a estabilidade dos resultados gerados e sua dependência em relação ao ajuste paramétrico.

Além de variar os parâmetros ajustáveis para observar os resultados, é importante estudar critérios e métodos automáticos para a definição destes parâmetros, dependendo, por exemplo, de características da base de dados e do domínio prático. Dessa forma, a escolha dos parâmetros ficaria mais voltada aos aspectos particulares do problema a ser resolvido, o que certamente aumentaria a eficiência da metodologia proposta.

Outro exemplo importante de trabalho futuro é o estudo das restrições na otimização, uma vez que o operador, ao gerar novas soluções a partir da atual, está sujeito a restrições relativas à validade da rede neural produzida. Sendo assim, o espaço de busca não é considerado completamente, mas sim com restrições que o limitam ao conjunto de possíveis redes MLP válidas. O estudo destas restrições no processo de otimização é de extrema importância para investigar o impacto destas restrições na exploração do espaço de busca.

Ressalte-se, ainda, a importância de aprofundar o estudo da metodologia proposta no que se refere à seleção de atributos. Tal aspecto, que emerge como uma conseqüência do funcionamento da metodologia, necessita de um estudo mais aprofundado sobre a eficiência da eliminação de atributos na resolução do problema.

Deve ser feito um levantamento bibliográfico sobre seleção de atributos com diferentes métodos, a fim de situar e contextualizar os resultados obtidos, afinal a tarefa de escolher atributos automaticamente tem merecido grande atenção por parte da comunidade científica, gerando muitas abordagens promissoras, notadamente com o emprego de algoritmos genéticos. Além disso, é importante estudar a correlação entre os atributos, com o objetivo de verificar se a seleção realizada pela metodologia proposta realmente consegue eliminar os atributos menos relevantes para a tarefa de classificação.

Outra possibilidade que surge a partir do presente trabalho é a realização de uma análise de complexidade computacional (espacial e temporal) das abordagens apresentadas (simulated annealing, tabu search, algoritmos genéticos e a metodologia proposta).

Finalmente, vale destacar a relevância de investigar quão genérica é a metodologia proposta sob o ponto de vista de otimização em geral, independentemente do domínio de redes neurais artificiais.

Percebe-se, portanto, que existem inúmeras possibilidades de ampliar e continuar o trabalho apresentado. As opções comentadas nesta seção não têm o intuito de esgotar as possibilidades, mas sim de mostrar que é importante definir linhas de pesquisa que possam contribuir para a consolidação da metodologia proposta diante da comunidade científica.

Apêndice A

Introdução às Redes Neurais Artificiais

Redes neurais artificiais são sistemas paralelos distribuídos formados por unidades de processamento simples que calculam determinadas funções, normalmente não-lineares. Estas unidades são distribuídas em camadas, estando interligadas por conexões geralmente unidirecionais, as quais se associam a pesos (nos modelos de redes neurais com pesos). Tais pesos armazenam o conhecimento representado na rede, servindo para ponderar as entradas recebidas por cada unidade constituinte.

A utilização de redes neurais artificiais para a resolução de problemas é caracterizada por um processo de *aprendizado*, durante o qual um conjunto de

exemplos, chamado de *conjunto de treinamento*, é apresentado para a rede, que extrai informações relevantes sobre as características das entradas. O conhecimento representado torna a rede capaz de fazer *generalização*, fornecendo, para dados nunca vistos, respostas coerentes com as características dos exemplos já aprendidos.

As unidades constituintes de uma rede neural podem ser chamadas de *neurônios* ou *nodos*. O modelo de neurônio mais conhecido é o de *McCulloch e Pitts* (MCP) [McCulloch and Pitts, 1943]. Tal modelo representa um neurônio como uma unidade com n entradas ($x_1, x_2, \ldots, x_n$) e uma única saída y. Cada entrada é ponderada por um peso associado, de modo que existem n pesos na unidade ($w_1, w_2, \ldots, w_n$). Na descrição original do modelo MCP, a saída y assume 1 se a soma das entradas ponderadas pelos pesos ($\sum_{i=1}^{n} x_i w_i$) for maior ou igual a um limiar θ, sendo igual a zero em caso contrário.

A partir do modelo MCP original, foram propostos vários outros modelos, considerando a saída y da unidade como uma determinada função, chamada de *função de ativação*, da soma das entradas ponderadas pelos pesos:

$$y = f(\sum_{i=1}^{n} x_i w_i)\,. \qquad (1)$$

Entre as mais utilizadas, merecem destaque as funções de ativação sigmóide logística e tangente hiperbólica [Prechelt, 1994].

Multi-Layer Perceptron (MLP)

Uma das redes neurais mais conhecidas é a rede *Multi-Layer Perceptron* (MLP) [Rumelhart et al., 1986]. Esta rede é composta por unidades descritas pela Equação (1) dispostas em pelo menos uma camada intermediária, além da camada de entrada, que não possui pesos ajustáveis, e da camada de saída, que sofre ajuste de pesos. Outra característica importante é que a rede é do tipo *feedforward*, ou seja, não apresenta conexões que partem de nodos de uma determinada camada para camadas anteriores. A figura A.1 apresenta um exemplo de rede MLP com uma camada intermediária.

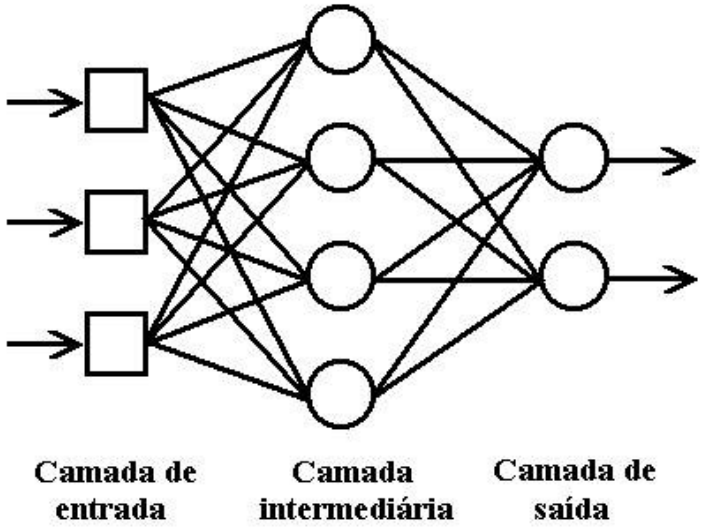


Figura A.1: Exemplo de rede MLP com uma camada intermediária, contendo 3 unidades de entrada, 4 unidades intermediárias e 2 unidades de saída. A rede possui todas as possíveis conexões *feedforward* entre camadas adjacentes, sem conexões entre camadas não-adjacentes.

Redes MLP apresentam um poder computacional muito maior do que aquele apresentado pelas redes sem camadas intermediárias, que resolvem apenas problemas linearmente separáveis. Teoricamente, sabe-se que redes com uma camada intermediária podem implementar qualquer função contínua [Cybenko, 1989], enquanto redes com duas camadas intermediárias podem implementar qualquer função matemática [Cybenko, 1988].

Algoritmo Backpropagation

O algoritmo de aprendizado mais conhecido para redes MLP é o backpropagation [Rumelhart et al., 1986], que utiliza pares de entradas e saídas desejadas, ajustando os pesos da rede por meio de um mecanismo de correção de erros. Este algoritmo é baseado no método do gradiente descendente, de modo que as funções de ativação utilizadas devem ser contínuas e diferenciáveis. As mais utilizadas são a sigmóide logística e a tangente hiperbólica [Prechelt, 1994].

O treinamento envolve duas fases: a *fase forward* e a *fase backward*. Na *fase forward*, a entrada da rede é apresentada à primeira camada de entrada, cujos nodos geram saídas dependendo de seus pesos. Estas saídas servem de entrada para a camada seguinte, que repete o processo para as camadas subseqüentes. As saídas produzidas

pelas unidades da última camada são comparadas com as saídas desejadas para o padrão de entrada. Na *fase backward*, esta comparação é utilizada para ajustar os pesos dos nodos da última camada de modo a reduzir seus erros. Em seguida, são ajustados os pesos da penúltima camada, dependendo dos erros da última camada ponderados pelos pesos das conexões entre a penúltima e a última camadas. O processo se repete até que os pesos da primeira camada intermediária sejam ajustados [Rumelhart et al., 1986].

Sejam E_p a função de erro para o padrão p, t_{pj} a saída desejada para o padrão p na unidade j, o_{pj} a saída atual para o padrão p na unidade j e w_{ij} o peso da conexão da unidade i para a unidade j. Considera-se a função de erro como sendo a soma dos erros quadráticos:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \,. \tag{2}$$

A ativação da unidade j, para o padrão p, pode ser escrita como:

$$net_{pj} = \sum_i w_{ij} o_{pi} \,. \tag{3}$$

A saída da unidade j é a função de ativação f_j aplicada na ativação:

$$o_{pj} = f_j(net_{pj}) \,. \tag{4}$$

Dessa forma, pela regra da cadeia, pode-se escrever:

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \,. \tag{5}$$

Considerando (3), o segundo termo do lado direito de (5) pode ser escrito como:

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_k w_{kj} o_{pk} = o_{pi} \,. \tag{6}$$

Escrevendo

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}, \tag{7}$$

a expressão (5) se torna

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} o_{pi}. \tag{8}$$

Diminuir o valor de E_p significa fazer as mudanças nos pesos proporcionais a $\delta_{pj} o_{pi}$, ou seja,

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi}. \tag{9}$$

Pela regra da cadeia, de acordo com (7), pode-se escrever

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}}. \tag{10}$$

O segundo termo do lado direito de (10) é dado por

$$\frac{\partial o_p}{\partial net_{pj}} = f_j'(net_{pj}). \tag{11}$$

Para as unidades de saída, o primeiro termo do lado direito de (10) é obtido simplesmente derivando (2) em relação a o_{pj}:

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}). \tag{12}$$

Para as unidades intermediárias, o mesmo termo pode ser calculado usando a regra da cadeia:

$$\frac{\partial E_p}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_i w_{ik} o_{pi} = -\sum_k \delta_{pk} w_{jk}. \tag{13}$$

O algoritmo procura minimizar o erro obtido pela rede ajustando pesos e limiares para que eles correspondam às coordenadas dos pontos mais baixos da

superfície de erro. Para este fim, utiliza informações do gradiente, que está na direção e no sentido em que a função de erro tem taxa de variação máxima. Dessa forma, garante-se que a rede caminha sobre a superfície na direção que reduz mais o erro. Vale ressaltar que, apesar de o algoritmo convergir para muitas superfícies simples, pode ficar aprisionado em mínimos locais nas superfícies mais complexas. Existem técnicas que aceleram a convergência do algoritmo e procuram evitar a incidência em mínimos locais, entre as quais se destacam a utilização de uma taxa de aprendizado decrescente e o uso de um termo de *momentum*.

Time Delay Neural Network (TDNN)

No reconhecimento de padrões cuja variação no tempo é um parâmetro importante, o uso de redes MLP treinadas com backpropagation pode não ser satisfatório, pois esta abordagem se destina ao aprendizado de mapeamentos estáticos. Um dos métodos propostos para lidar com este tipo de problema é o uso de atrasos no tempo (*delays*), de modo que a entrada da rede é formada por trechos das seqüências temporais dos dados, e o treinamento pode ser feito para estes trechos como se fossem padrões estáticos.

Uma das abordagens bastantes conhecidas que se utiliza de atrasos no tempo para realizar processamento temporal é a rede *time delay neural network* (TDNN) [Lang and Hinton, 1988], que é uma rede *feedforward* de múltiplas camadas, cujas entradas são trechos das seqüências temporais de interesse. A figura A.2 ilustra um exemplo de rede TDNN.

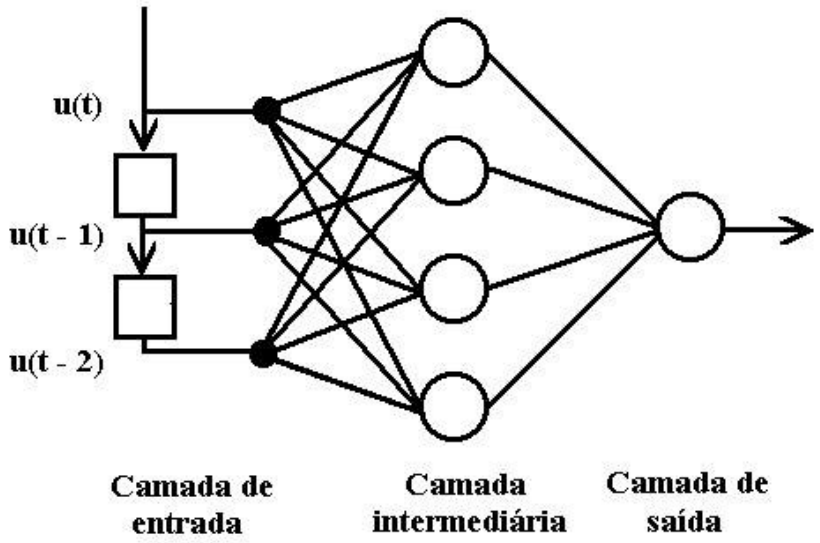


Figura A.2: Exemplo de rede TDNN contendo dois atrasos de tempo, quatro unidades intermediárias e uma unidade de saída. Esta rede também possui todas as possíveis conexões *feedforward* entre camadas adjacentes, sem conexões entre camadas não-adjacentes.

Inicialmente, esta técnica foi idealizada com a finalidade de introduzir explicitamente o conceito de simetria no tempo, como encontrado no problema de reconhecimento de fonemas isolados utilizando um espectrograma. Um espectrograma é uma imagem bidimensional cuja dimensão vertical corresponde à freqüência e cuja dimensão horizontal corresponde ao tempo, de modo que a intensidade da imagem corresponde à energia do sinal. Em [Lang and Hinton, 1988], foi utilizada com sucesso uma rede TDNN para reconhecer quatro fonemas isolados.

Este tipo de rede é apropriado para diversas aplicações, como controle adaptativo em ambientes não-estacionários, identificação de sistemas dinâmicos, eliminação de ruído, equalização adaptativa de canais de comunicação cujas freqüências variam no tempo, modelagem de séries temporais não-estacionárias e classificação temporal de padrões não-estacionários.

O treinamento pode ser feito com o mesmo algoritmo backpropagation utilizado para as redes MLP. Outra abordagem interessante é o uso de métodos de segunda ordem, como o de Levenberg-Marquardt [Fletcher, 1987].

Método de Levenberg-Marquardt

Considerando a soma dos erros quadráticos:

$$E = \frac{1}{2}\sum_{n}(\varepsilon^{n})^{2} = \frac{1}{2}\left\|\varepsilon\right\|^{2}, \tag{14}$$

onde ε^{n} é o erro para o *n*-ésimo padrão, e ε é o vetor formado pelos elementos ε^{n}, e supondo um deslocamento no espaço de pesos a partir do ponto *w* para o ponto *w'*, se o deslocamento *w'* – *w* for pequeno, o vetor de erro ε pode ser expandido em série de Taylor para a primeira ordem:

$$\varepsilon(w') = \varepsilon(w) + Z\,(w'-w), \tag{15}$$

onde *Z* é a matriz de elementos

$$Z_{ni} \equiv \frac{\partial\,\varepsilon^{n}}{\partial\,w_{i}}. \tag{16}$$

A função de erro (14) pode, então, ser aproximada por

$$E = \frac{1}{2}\left\|\varepsilon(w) + Z\,(w'-w)\right\|^{2}. \tag{17}$$

Se este erro for minimizado em relação aos novos pesos *w'*, obtém-se

$$w' = w - (Z^{T}Z)^{-1}Z^{T}\varepsilon(w). \tag{18}$$

Em princípio, a fórmula de atualização (18) poderia ser aplicada iterativamente, a fim de tentar minimizar a função de erro. O problema desta abordagem é que a magnitude da alteração de (18) pode ser relativamente grande, e, neste caso, a aproximação linear dada em (17) deixaria de ser válida. O algoritmo de Levenberg-Marquardt [Marquardt, 1963] resolve este problema procurando minimizar a função de erro, ao mesmo tempo em que procura manter pequena a magnitude da alteração dos pesos, de modo que a linearização permanece válida. Utiliza-se uma função de erro modificada, dada por:

$$\tilde{E} = \frac{1}{2}\left\|\varepsilon(w) + Z\,(w'-w)\right\|^2 + \lambda\left\|w'-w\right\|^2, \tag{19}$$

onde o parâmetro λ controla a magnitude da alteração. Para valores de λ suficientemente grandes, o valor de $\left\|w'-w\right\|^2$ tenderá a ser pequeno. Minimizando (19) em relação a *w'*, obtém-se

$$w' = w - (Z^T Z + \lambda I)^{-1} Z^T \varepsilon(w), \tag{20}$$

onde *I* é a matriz unidade. Nota-se que, para valores muito grandes de λ, a expressão (20) é a mesma da modificação dos pesos no método do gradiente descendente, com a magnitude da alteração controlada por λ^{-1}. Dessa forma, para valores suficientemente grandes de λ, o erro necessariamente diminui, gerando um pequeno passo na direção do gradiente negativo.

Na prática, deve ser escolhido um valor inicial para λ, e este valor deve ser modificado adequadamente durante o processo de minimização. Uma abordagem comum é atribuir um valor arbitrário, como λ=0.1, e monitorar a alteração na função de erro. Se o erro diminuir após a modificação dada por (20), aceita-se o novo vetor de pesos, o valor de λ é diminuído por um fator de 10, e o processo é repetido. Se aumentar, não se aceita o novo vetor de pesos e o valor de λ é aumentado por um fator de 10, para que a alteração seja efetuada novamente, até que o erro diminua [Bishop, 1995]. Sendo assim, o valor inicial de λ não é um parâmetro particularmente crítico, pois é ajustado durante o processo e influencia, apenas, a taxa de convergência inicial: se for muito grande, o algoritmo tomará passos pequenos, e, se for muito pequeno, o algoritmo terá que aumentar o parâmetro até que sejam tomados passos suficientemente pequenos [Norgaard, 1997].

Apêndice B

Narizes Artificiais

Uma grande variedade de gases e vapores precisa ser analisada em diversas aplicações, como diagnóstico médico, monitoramento ambiental e controle de qualidade em indústrias de alimentos, bebidas e cosméticos. Muitas destas análises são feitas pelo olfato de indivíduos especializados, o que apresenta uma série de limitações, como a influência de alergias e doenças, os riscos da inalação de certas substâncias tóxicas, a fadiga e a mudança da sensibilidade olfativa de acordo com o estado mental.

Sendo assim, tornam-se necessários sistemas capazes de detectar e classificar odores, vapores e gases automaticamente, sendo favoráveis em diversas áreas. Por exemplo, em relação ao monitoramento ambiental, existe a necessidade de analisar misturas combustíveis, detectar vazamentos e controlar a qualidade do ar e as emissões

industriais. Os diagnósticos médicos podem ser facilitados, uma vez que os odores da respiração representam sintomas de problemas gastrointestinais, infecções, diabetes e doenças do fígado. Além disso, tecidos infectados emitem odores característicos. Já na indústria de alimentos, os narizes artificiais teriam aplicações no controle da qualidade e no monitoramento de processos de produção.

Histórico de Narizes Artificiais

Provavelmente, o primeiro equipamento criado com a finalidade de detectar odores foi desenvolvido por Moncrieff em 1961 [Moncrieff, 1961]. Tal sistema era mecânico, sendo que os primeiros narizes eletrônicos foram desenvolvidos por Wilkens e Hatman em 1964 [Wilkens and Hatman, 1964], Buck *et al.* em 1965 [Buck et al., 1965] e Dravnieks e Trotter, também em 1965 [Dravnieks and Trotter, 1965]. Entretanto, o conceito de nariz artificial como um sistema inteligente com sensores químicos para a classificação de odores só apareceu em 1982, nos trabalhos de Persaud e Dodd [Persaud and Dodd, 1982], e em 1985, nos trabalhos de Ikegami e Kaneyasu [Ikegami and Kaneyasu, 1985].

A denominação "nariz artificial" surgiu no fim da década de 1980 [Gardner and Bartlett, 1994], sendo usada especificamente em uma conferência em 1987 [Gardner, 1987]. Em 1989, uma sessão do *Advanced Workshop on Chemosensory Information Processing* da OTAN foi dedicada ao olfato artificial [Schild, 1990]. A primeira conferência dedicada ao tópico de narizes eletrônicos ocorreu em 1990 [Gardner and Bartlett, 1992].

Constituição de um Nariz Artificial

Um nariz artificial é formado por duas partes principais: um *sistema sensor* e um *sistema de reconhecimento de padrões*. O sistema sensor pode ser composto por um conjunto de sensores distintos, em que cada elemento mede uma propriedade diferente do composto odorante, ou por um único dispositivo (por exemplo, um espectrômetro), que produz um conjunto de medições para cada odorante. Pode ser constituído, ainda, por uma combinação de ambas as estruturas [Keller et al., 1995].

Cada composto odorante apresentado ao sistema sensor produz um padrão característico. Em seguida, apresenta-se este sinal ao sistema de reconhecimento de padrões. Normalmente, é necessária uma etapa de pré-processamento dos sinais gerados pelo sistema sensor antes da fase do reconhecimento de padrões [Gardner and Hines, 1997]. A estrutura básica do processamento em um nariz artificial pode ser ilustrada pela Figura B.1.

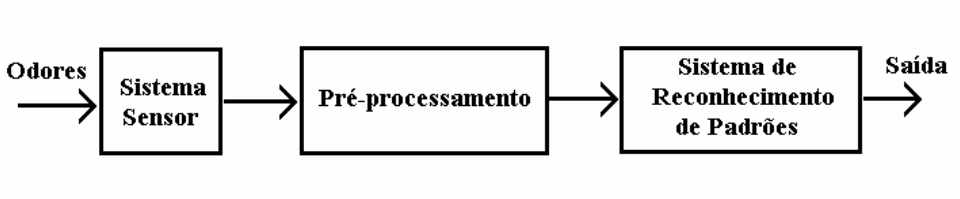


Figura B.1: Estrutura básica de um nariz artificial.

A seguir, são explicadas as partes constituintes de um nariz artificial.

Sistema Sensor

O olfato dos mamíferos é capaz de identificar uma ampla faixa de compostos odorantes com alta sensibilidade e de reconhecer substâncias pelas combinações e proporções relativas dos componentes. Tal capacidade se deve à atuação de elementos sensores no epitélio olfativo (neurônios receptores), que possuem perfis de sensibilidade amplos e com superposições e não são altamente específicos. Por este motivo, o sistema de sensores dos narizes eletrônicos em geral é formado por um conjunto de elementos não-específicos, ou seja, com seletividades amplas e parcialmente superpostas.

Existem duas categorias principais de sensores com esta característica: *sensores de absorção* e *semicondutores quimicamente sensíveis* [Kress-Rogers, 1997].

Sensores de Absorção

Um sensor de absorção [Grate et al., 1997] é formado por uma camada de material absorvente quimicamente seletivo aplicada em um dispositivo de base acústico ou óptico. Quando em contato com os odorantes, esta camada sofre mudanças em certas propriedades ópticas, como índice de refração, ou mecânicas, como espessura,

densidade e elasticidade. Estas mudanças são detectadas pelos dispositivos de base, que podem ser ópticos, como o *SPR* (*surface plasmon resonance*) [Lawrence and Geddes, 1997], ou acústicos, como o *SAW* (*surface acoustic wave*) [D'Amico et al., 1997]. Tais dispositivos de base produzem um sinal relacionado à concentração do odorante em contato com a superfície.

Semicondutores Quimicamente Sensíveis

Em relação a este tipo de sensor, existem duas classes principais: os *sensores de semicondutores de óxido metálico* e *os sensores de polímeros condutores* [Kress-Rogers, 1997].

Semicondutores de Óxido Metálico

No sensor semicondutor de óxido metálico [Kohl, 1997], o composto mais utilizado é o SnO_2, sendo também encontrados ZnO, In_2O_3, WO_3, Fe_2O_3 e Ga_2O_3. Este tipo de sensor funciona com base em mudanças na condutividade induzidas por adsorção de gases e reações superficiais subseqüentes. Em geral, opera em temperaturas entre 100 e 600 ^{0}C, daí a necessidade de um elemento aquecedor, resultando em um consumo de energia relativamente alto.

Este sensor é formado por uma película de cerâmica eletricamente aquecida, sobre a qual se deposita um filme fino poroso de óxido metálico dopado com diversos metais. O óxido metálico dopado se comporta como um semicondutor do tipo *N*, de modo que a adsorção de oxigênio na superfície provoca a retirada de elétrons da banda de condutividade. Os gases interagem com o oxigênio adsorvido na superfície, afetando a condutividade do filme de óxido metálico [Persaud and Travers, 1997].

A sensibilidade do dispositivo pode ser controlada pela temperatura. Assim, o conjunto de sensores deste tipo pode ter todos os elementos idênticos com temperaturas distintas, para apresentarem perfis variados de seletividade, ou pode usar elementos diferentes. Em temperaturas distintas, as sensibilidades são diferentes, mas são amplas e apresentam superposição, o que possibilita o uso em narizes artificiais que procuram funcionar de forma semelhante ao olfato natural. Outra forma de modificar as características de resposta é a variação do agente dopante [Kohl, 1997].

O uso de configurações resistivas para a medição é suficiente na maioria dos casos, não sendo necessários dispositivos mais complexos. A incorporação destes semicondutores de óxido metálico em dispositivos microeletrônicos como transistores de efeito de campo (FET) origina os MOSFETs, que também são utilizados como sensores de gás [Persaud and Travers, 1997].

Exemplos da utilização de sensores semicondutores de óxido metálico em narizes artificiais podem ser encontrados em [Gardner et al., 1990], [Sundgren et al., 1991], [Keller et al., 1994], [Di Natale et al., 1995] e [Di Natale et al., 1996].

Polímeros Condutores

Polímeros condutores foram identificados pela primeira vez na década de 1970 e pertencem a uma classe de materiais orgânicos que podem ser sintetizados eletroquimicamente [Gardner and Bartlett, 1995]. A variabilidade na escolha da estrutura química e a facilidade no controle da deposição do material, entre outros fatores, favorecem a aplicação desta tecnologia em dispositivos microeletrônicos.

Estes materiais orgânicos podem ser gerados a partir de uma grande variedade de monômeros (por exemplo, o pirrol) em diversos solventes (por exemplo, na água), e este processo pode ocorrer na presença de uma variedade de íons. Em uma célula eletroquímica de três eletrodos, o monômero passa por diversas reações, até que o polímero se torna insolúvel, precipitando-se sobre o eletrodo de trabalho na forma de um filme fino. Para o caso do polipirrol, que é um polímero condutor bastante conhecido e utilizado, detalhes sobre o processo de síntese podem ser encontrados em [Gardner and Bartlett, 1991].

A deposição de polímeros condutores é bem controlada, pois estes materiais podem ser eletrodepositados seguramente em áreas variadas (em geral, entre 0.01 mm^2 e 10 cm^2) e com diversas espessuras (por exemplo, de 0.1 a 10 μm). Isto permite que um polímero condutor seja depositado sobre áreas definidas de dispositivos de silício. Além disso, existe a possibilidade de depositar filmes em superfícies convexas, côncavas e reentrantes. Outro aspecto vantajoso é que a deposição ocorre em temperatura ambiente [Gardner and Bartlett, 1995].

Sabe-se que as propriedades eletrônicas de certos polímeros condutores podem ser modificadas pela presença de um gás ou vapor. Por este motivo, estes materiais têm sido utilizados na fabricação de dispositivos microeletrônicos quimicamente sensíveis, como resistores. Uma vantagem de seu uso em relação aos semicondutores de óxido metálico é a operação em temperatura ambiente, de modo que um conjunto de sensores de polímeros condutores não apresenta alto consumo de energia e pode ser usado em um instrumento portátil [Gardner and Bartlett, 1995].

Os sensores de polímeros condutores [Persaud and Travers, 1997] se baseiam na medição das mudanças de resistência em estruturas de filmes finos. Estes sensores respondem a espécies deficientes de elétrons (por exemplo, NO_2) e a espécies ricas em elétrons (por exemplo, NH_3), e esta resposta se deve a uma reação química na superfície, que acrescenta ou remove portadores de carga no filme semicondutor. As medições podem ser feitas com configurações simples resistivas ou com dispositivos microeletrônicos, como transistores de efeito de campo (FET*)*.

Os sensores construídos com polímeros condutores apresentam diversas características favoráveis, entre as quais se destacam [Persaud and Travers, 1997]:

- a rápida cinética de adsorção em temperatura ambiente,
- o pequeno consumo de energia (da ordem de microwatts), pois não é necessário um elemento aquecedor,
- a resistência ao envenenamento por compostos que normalmente tornariam inativos os sensores inorgânicos de semicondutores, como os compostos que contêm enxofre,
- e a possibilidade de o polímero ser construído com especificidade para determinados compostos químicos.

Exemplos da utilização de sensores com polímeros condutores em narizes artificiais podem ser encontrados em [Slater et al., 1993], [Pearce et al., 1993], [Gardner et al., 1994], [De Souza et al., 1999] e [Santos, 2000].

Pré-processamento dos Dados

É importante examinar cuidadosamente os dados gerados pelo conjunto de sensores. Tais dados devem ser preparados da melhor forma possível para o uso do método de reconhecimento de padrões escolhido. Por exemplo, em determinadas situações, os vetores de resposta dependem linearmente da concentração do odor; em outros casos, a dependência é não-linear, tornando-se necessária uma linearização através de alguma função matemática, se a técnica de análise de padrões for linear [Gardner and Hines, 1997].

Considerando que cada sensor i gera um sinal de saída no tempo $x_{ij}(t)$ em resposta a um odor j, a resposta de um conjunto de n sensores pode ser representada por um vetor $x_j(t)$ [Gardner and Hines, 1997]:

$$x_j(t) = (x_{1j}(t), x_{2j}(t), \ldots , x_{nj}(t)). \quad (1)$$

Em geral, a resposta de cada sensor no decorrer do tempo depende dos seguintes fatores [Gardner and Bartlett, 1994]:

- tipo de fluxo do sistema de transporte do odor a partir da origem até o conjunto de sensores,
- natureza do odor (por exemplo, tipo e concentração),
- cinética da reação entre o odor e o material ativo do sensor,
- difusão do odor no material ativo,
- natureza do material sensor (por exemplo, estrutura física, porosidade e constantes térmicas),
- natureza do substrato que serve de suporte ao material ativo (por exemplo, condutividade térmica e impedância acústica),
- e condições do ambiente (por exemplo, temperatura do material ativo, umidade e pressão).

Na maioria dos casos, a resposta é representada por um vetor r_j de parâmetros que independe do tempo [Gardner and Hines, 1997]:

$$r_j = (r_{1j}, r_{2j}, \ldots , r_{nj}). \tag{2}$$

A escolha dos parâmetros de resposta é fundamental para o desempenho do sistema de reconhecimento de padrões. Diversos modelos já foram sugeridos, entre os quais podem ser citados [Gardner and Bartlett, 1994]:

- o *Modelo da Diferença*:

$$r_{ij} = x_{ij}^{\max} - x_{ij}^{\min}, \tag{3}$$

- o *Modelo Relativo*:

$$r_{ij} = \frac{x_{ij}^{\max}}{x_{ij}^{\min}}, \tag{4}$$

- o *Modelo de Diferença Fracional*:

$$r_{ij} = \frac{x_{ij}^{\max} - x_{ij}^{\min}}{x_{ij}^{\min}}, \tag{5}$$

- e o *Modelo Logarítmico*:

$$r_{ij} = \log (x_{ij}^{\max} - x_{ij}^{\min}), \tag{6}$$

onde $x_{ij}^{\max}$ e $x_{ij}^{\min}$ são, respectivamente, os valores máximo e mínimo do sinal $x_{ij}(t)$.

Dessa forma, a resposta dos sensores a um conjunto de m odores pode ser representada por uma matriz R, em que cada coluna j é a resposta do conjunto de sensores ao odor j:

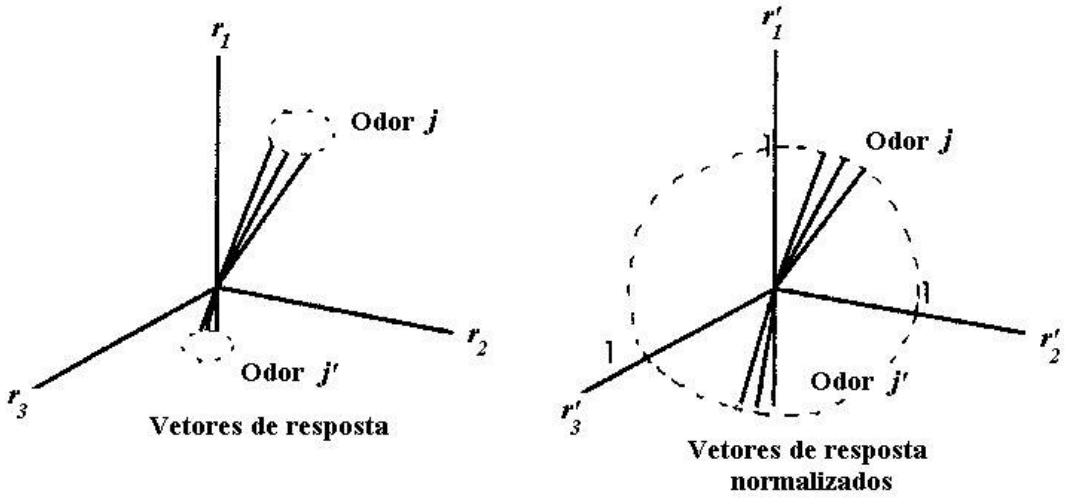
$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{bmatrix}. \tag{7}$$

A complexidade do problema de análise de padrões está relacionada com o número de termos não-nulos da matriz *R* [Gardner and Hines, 1997]. Por exemplo, se os *n* sensores forem independentes, ou seja, se cada sensor responder apenas a um entre os *m* odores, então cada linha só possuirá um elemento não-nulo. Neste caso, a matriz de resposta é uma maneira conveniente de manipular os dados, não sendo necessário utilizar métodos mais sofisticados de análise de padrões. Em geral, os sensores não são independentes, de modo que cada um responde a uma variedade de odores com sensibilidade diferente. Dessa forma, uma técnica de reconhecimento de padrões mais elaborada é necessária para processar os sinais originados pelo conjunto de sensores.

Muitas vezes, é necessária uma normalização dos dados, para compensar as flutuações de concentração. A discriminação entre odores complexos similares pode ser melhorada substancialmente se os comprimentos dos vetores de resposta forem normalizados [Gardner and Hines, 1997]. A Figura B.2(a) mostra uma representação geral de um espaço de 3 sensores, com os vetores de resposta para três amostras dos odores *j* e *j'*. Na Figura B.2(b), os vetores de resposta foram normalizados de acordo com a expressão:

$$r'_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{n} r_{ij}^2}}, \tag{8}$$

onde r'_{ij} é o valor normalizado correspondente ao valor original r_{ij}.



(a) (b)

Figura B.2: Efeito da normalização dos vetores de resposta como método de pré-processamento em um nariz artificial.

Este método é particularmente útil quando as concentrações das amostras não têm relevância para o problema, sendo mais importante a discriminação dos odores. O processo de normalização compensa variações experimentais nas concentrações e tem sido usado no pré-processamento de respostas de sensores de óxido metálico e de polímeros condutores. É necessário cuidado com a aplicação desta técnica, pois pode haver aumento no ruído das respostas mais fracas.

A visualização gráfica dos vetores de resposta pode ajudar no pré-processamento dos dados. Para este fim, utilizam-se gráficos polares, nos quais a resposta de cada sensor é indicada por um parâmetro radial, com o ângulo indicando sua posição no conjunto de sensores. Por exemplo, a Figura B.3 mostra gráficos polares da alteração fracional de condutância em um conjunto de doze sensores de polímeros condutores em um nariz artificial [Gardner, 1993]. A diferença entre os padrões originados pelos dois odores distintos pode ser facilmente visualizada.

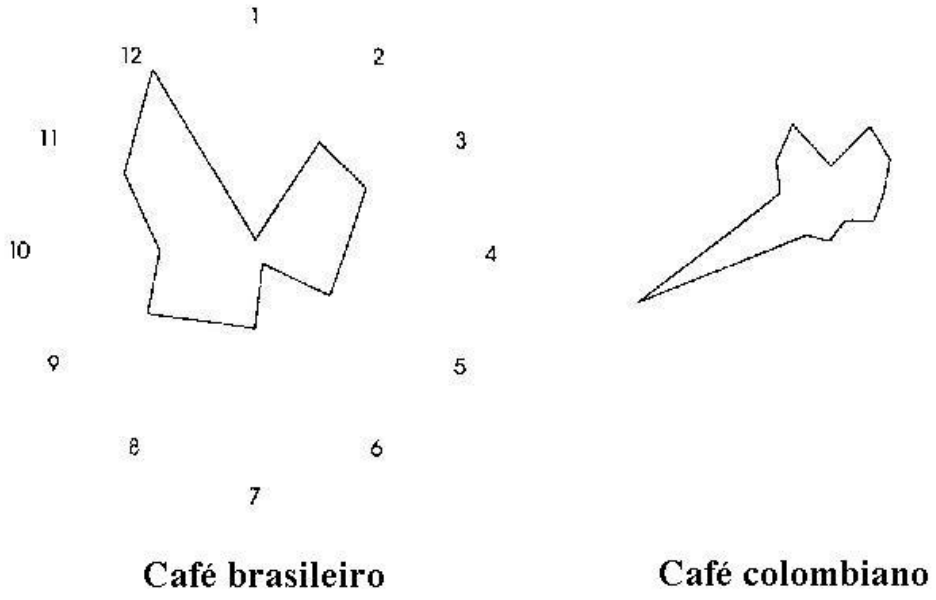


Figura B.3: Gráficos polares dos vetores de resposta para odores de café brasileiro e de café colombiano em um nariz artificial que contém doze sensores de polímeros condutores.

Sistema de Reconhecimento de Padrões

Em geral, os sistemas de reconhecimento de padrões são divididos em *paramétricos* e *não-paramétricos* [Gardner and Hines, 1997]. Uma técnica paramétrica assume que os dados dos sensores podem ser descritos por uma função densidade de probabilidade. Normalmente, admite-se que os dados têm uma distribuição normal, com média e variância conhecidas. Este tipo de técnica exige a construção de uma base de conhecimento, contendo as funções densidade de probabilidade apropriadas. As técnicas de reconhecimento de padrões que não assumem funções densidade de probabilidade são conhecidas como não-paramétricas.

As principais técnicas de reconhecimento de padrões utilizadas em narizes artificiais são brevemente comentadas a seguir [Gardner and Hines, 1997].

Análise de Função Discriminante

A Análise de Função Discriminante (DFA) é um método paramétrico de análise de padrões que tem sido utilizado para processar dados de narizes eletrônicos. Esta técnica

assume que os dados possuem distribuição multi-normal e determina as funções discriminantes, que se relacionam linearmente com os vetores de resposta dos sensores.

Exemplos de utilização da análise de função discriminante em narizes eletrônicos podem ser vistos em [Gardner and Bartlett, 1992] e [Gardner et al., 1992].

Análise de Clusters

É um método não-supervisionado que tem sido utilizado na discriminação de vetores de resposta, identificando agrupamentos (*clusters*) a partir do conjunto de dados disponíveis. As métricas aplicadas normalmente são lineares (euclidianas), embora existam variações que se utilizam de métricas não-lineares.

Alguns exemplos da utilização da análise de clusters para narizes artificiais podem ser vistos em [Gardner, 1991] e [Gardner and Bartlett, 1992].

Análise das Componentes Principais

É uma técnica não-paramétrica, supervisionada e linear que consiste em expressar os vetores de resposta dos sensores em termos de combinações lineares de vetores ortogonais. Cada vetor ortogonal (componente principal) contribui na variância dos dados com um grau decrescente de importância.

Podem ser encontrados exemplos do uso de análise das componentes principais para narizes eletrônicos em [Gardner, 1991], [Gardner and Bartlett, 1992] e [Di Natale et al., 1995].

Redes Neurais Artificiais

São sistemas não-paramétricos e, em geral, não-lineares compostos por unidades de processamento paralelamente interconectadas e, normalmente, adaptativas, cuja organização é baseada em modelos físicos de sistemas biológicos.

Redes neurais artificiais têm apresentado diversas características favoráveis no reconhecimento de padrões de odor [Craven et al., 1996], como a capacidade de lidar com sinais não-lineares provenientes dos sensores, adaptabilidade, tolerância a erros, tolerância a ruído e paralelismo inerente, permitindo rapidez no uso após o treinamento

[Gardner et al., 1990]. O tipo mais popular é o *Multi-Layer Perceptron* (MLP), que é treinado com o algoritmo backpropagation [Rumelhart et al., 1986]. Maiores detalhes sobre redes MLP e algoritmo backpropagation podem ser encontrados no Apêndice A do presente trabalho.

Exemplos de utilização de redes neurais para narizes eletrônicos podem ser vistos em [Gardner et al., 1990], [Sundgren et al., 1991], [Gardner et al., 1994], [Keller et al., 1994], [Di Natale et al., 1996], [De Souza et al., 1999] e [Santos, 2000].

Em [Gardner et al., 1990], foram utilizados doze sensores comerciais de óxido de estanho em contato com cinco compostos químicos (metanol, butan-1-ol, propan-2-ol, 2-metil-1-butanol e etanol). Aplicou-se uma rede MLP com uma camada intermediária e função de ativação sigmoidal, que foi submetida ao treinamento com backpropagation. Apenas oito padrões foram obtidos para os experimentos, sendo que sete deles eram usados para treinamento, enquanto o oitavo servia para testar a rede. Os resultados mostraram que, para uma saída desejada de 1, as saídas geradas foram da ordem de 0.8 ou acima, e, para uma saída desejada de 0, as saídas geradas foram da ordem de 0.2 ou abaixo.

Em [Sundgren et al., 1991], foram utilizados seis sensores MOSFET em contato com dois tipos de misturas gasosas (um tipo contendo quatro gases e outro tipo contendo dois gases). Neste caso, usou-se uma rede MLP com uma camada intermediária treinada com backpropagation com a finalidade de identificar a concentração de cada gás componente na mistura. As concentrações previstas pela rede e as concentrações reais foram comparadas graficamente e apresentaram curvas próximas umas das outras.

Em [Gardner et al., 1994], utilizaram-se 24 sensores de polímeros condutores em contato com aromas de tipos diferentes de cerveja. Foi aplicada uma rede MLP com uma camada intermediária treinada com backpropagation, tendo a finalidade de classificar os odores das soluções apresentadas aos sensores. Foi obtido um percentual de classificações corretas de 93%.

Em [Keller et al., 1995], o nariz artificial é composto por 11 sensores de óxido de estanho em contato com substâncias que podem ser adquiridas comercialmente,

como acetona, vinagre e fluidos de limpeza. Foram utilizados 619 padrões para treinamento e 196 para teste. Aplicou-se uma rede MLP com uma camada intermediária treinada com backpropagation, sendo obtido um percentual de 92.9% de classificações corretas para os padrões de teste.

Em [Di Natale et al., 1996], apresenta-se um nariz artificial contendo quatro sensores de semicondutores de óxido metálico com a finalidade de classificar odores de duas diferentes safras de um mesmo vinho. Como técnica de reconhecimento de padrões, foi aplicada uma rede MLP de uma camada intermediária treinada com backpropagation, que conseguiu identificar corretamente todo o conjunto utilizado para teste.

Em [De Souza et al., 1999] e [Santos, 2000], são apresentados três protótipos de narizes artificiais, todos fazendo uso de sensores de polímeros condutores. O primeiro é composto de três sensores distintos, tendo a finalidade de classificar substâncias simples (metanol, tetracloreto de carbono e etanol) através da aplicação de uma rede MLP com uma camada escondida treinada com backpropagation. O sistema classificou corretamente 96.5% dos padrões de etanol, 98.4% dos padrões de metanol e 98.3% dos padrões de tetracloreto de carbono. O segundo apresenta melhorias na construção dos sensores e na aquisição dos dados, sendo testado para reconhecer odores de substâncias mais complexas, como diferentes safras de vinho. Foram aplicadas redes MLP, *Radial Basis Function* (RBF) e de Elman, permitindo uma comparação nos desempenhos de diferentes abordagens de redes neurais. Os resultados obtidos na classificação não apresentaram diferenças significativas entre as três abordagens de redes neurais. O terceiro protótipo apresenta mudanças na tecnologia utilizada para produzir os sensores e ainda está em aperfeiçoamento. Tal protótipo já foi testado com substâncias puras (etanol, benzeno, metanol e tetracloreto de carbono) e com algumas misturas destas substâncias (etanol com tetracloreto de carbono e benzeno com metanol), obtendo 100% de classificações corretas para os padrões de teste.

Em abordagens anteriores do autor deste trabalho [Yamazaki, 2001][Yamazaki and Ludermir, 2001][Yamazaki et al., 2001], o uso de processamento temporal melhorou sensivelmente a classificação de odores de safras distintas de um mesmo

vinho, levando em consideração as variações dos sinais dos sensores ao longo da aquisição dos dados.

Apêndice C

Testes de Diferenças entre Médias

Neste trabalho, foram comparados resultados obtidos para diferentes abordagens. Dessa forma, foi importante realizar testes de diferença entre as médias dos resultados obtidos, a fim de verificar se tais médias podem ser consideradas distintas do ponto de vista estatístico. Aplicando duas técnicas para a mesma base de dados, foi utilizado um teste de diferença entre médias para dados pareados quando os resultados eram provenientes das mesmas inicializações de pesos. Entretanto, houve comparações entre técnicas distintas (para a mesma base de dados) com resultados provenientes de inicializações diferentes de pesos. Neste caso, foi empregado um teste de diferença entre médias para dados não-pareados. Ambos os testes aplicados neste trabalho são explicados a seguir.

Teste de diferença entre médias para dados pareados

Com o intuito de comparar as médias obtidas para duas abordagens distintas, usando as mesmas inicializações de pesos, foi empregado o conhecido teste *t* de diferença entre médias populacionais para dados pareados [Meyer, 1983]. Este teste é bastante utilizado na comparação de resultados de algoritmos de aprendizado.

O teste foi aplicado da seguinte forma: considera-se que os resultados obtidos (por exemplo, valores de erro) pela primeira abordagem são dados por $\{X_1, X_2, \ldots, X_N\}$, enquanto os resultados obtidos pela segunda abordagem são dados por $\{Y_1, Y_2, \ldots, Y_N\}$, de modo que X_i e Y_i foram obtidos a partir da mesma inicialização de pesos (N é a quantidade de inicializações de pesos que foram adotadas nos experimentos). Dessa forma, o objetivo é verificar se a média populacional μ_X é estatisticamente menor que média populacional μ_Y. Vale ressaltar que, em muitos trabalhos com técnicas de aprendizado, tal consideração é feita para N partições do conjunto de dados (por exemplo, em uma abordagem de validação cruzada), de modo que a média é calculada a partir dos resultados obtidos pela mesma técnica quando aplicada a diferentes partições do conjunto de dados. Entretanto, no presente trabalho, a partição dos dados foi mantida constante para todos os experimentos, de modo que a média não é calculada a partir de diferentes partições, mas sim a partir de diferentes inicializações de pesos das redes neurais.

A comparação é feita com base em um teste de hipóteses, ou seja, considera-se uma hipótese inicial (comumente chamada de *hipótese nula*), que é confrontada com uma *hipótese alternativa*. Neste trabalho, a hipótese nula assume que $\mu_X = \mu_Y$, enquanto a hipótese alternativa assume que $\mu_X < \mu_Y$.

Uma vez enunciadas as hipóteses, calculam-se os valores $\{D_1, D_2, \ldots, D_N\}$, sendo que $D_i = X_i - Y_i$. A média destes valores é dada por μ_D, e o desvio-padrão é dado por σ_D. Em seguida, escolhe-se o *nível de significância* (que, neste trabalho, é igual a 5%, sendo um dos níveis mais comumente utilizados na literatura), e, a partir da distribuição *t* de Student, é obtido o *valor crítico* da variável *t* para $N - 1$ graus de liberdade.

A fim de decidir se a hipótese nula será aceita ou rejeitada, calcula-se:

$$T = \frac{\mu_D}{\left(\frac{\sigma_D}{\sqrt{N}}\right)}. \quad (1)$$

Se o valor calculado T for maior que o valor crítico obtido da distribuição *t* de Student, rejeita-se a hipótese nula, de modo que a média μ_X é considerada *estatisticamente menor* que a média μ_Y com um nível de significância de 5%. Caso contrário, não é rejeitada a hipótese nula, e a média μ_X é considerada *estatisticamente igual* à média μ_Y (novamente, considerando um nível de significância de 5%).

Teste de diferença entre médias para dados não-pareados

Para comparar médias oriundas de duas populações distintas, o teste de diferença entre médias para dados não-pareados deve ser aplicado da seguinte forma: considerando que a primeira população originou N_X amostras, cuja média populacional é dada por μ_X e cujo desvio-padrão populacional é dado por σ_X, e que a segunda população originou N_Y amostras, cuja média populacional é dada por μ_Y e cujo desvio-padrão populacional é dado por σ_Y, o número de graus de liberdade da variável *t* da distribuição de Student vai depender do fato de as variâncias σ_X^2 e σ_Y^2 serem iguais ou diferentes. No caso de serem iguais, a variável *t* terá o seguinte número de graus de liberdade:

$$\nu = N_X + N_Y - 2. \quad (2)$$

Caso contrário, a quantidade de graus de liberdade deve ser calculada pela expressão:

$$\nu = \left[\frac{(\omega_X + \omega_Y)^2}{\frac{\omega_X^2}{N_X + 1} + \frac{\omega_Y^2}{N_Y + 1}} \right] - 2, \quad (3)$$

sendo que

$$\omega_X = \frac{\sigma_X^2}{N_X}, \tag{4}$$

$$\omega_Y = \frac{\sigma_Y^2}{N_Y}. \tag{5}$$

Para verificar se as variâncias são iguais ou diferentes, aplica-se o teste *F* de diferença entre variâncias populacionais: a hipótese nula assume que $\sigma_X^2 = \sigma_Y^2$, enquanto a hipótese alternativa assume que $\sigma_X^2 \neq \sigma_Y^2$. Sendo σ_X^2 a maior variância, calcula-se:

$$F = \frac{\sigma_X^2}{\sigma_Y^2}. \tag{6}$$

Em seguida, escolhe-se o *nível de significância* (que, neste trabalho, assume 5%, sendo um dos níveis mais comumente utilizados na literatura), e, a partir da distribuição *f* de Fisher, é obtido o *valor crítico* da variável *f* para $N_X - 1$ graus de liberdade no numerador e $N_Y - 1$ graus de liberdade no denominador. Se o valor calculado *F* for maior que o valor crítico, rejeita-se a hipótese nula, de modo que as variâncias são consideradas distintas, e o número de graus de liberdade da variável *t* de Student é dado por (3). Caso contrário, aceita-se a hipótese nula, e as variâncias são consideradas iguais, de modo que o número de graus de liberdade da variável *t* é dado por (2).

Uma vez definido o número de graus de liberdade da variável *t* de Student, calcula-se:

$$T = \frac{\mu_X - \mu_Y}{\sqrt{\omega_X + \omega_Y}}. \tag{7}$$

O valor de *T* é utilizado da mesma forma como é feito no teste para dados pareados: se o objetivo for verificar se a média μ_X é maior que a média μ_Y, a hipótese nula assume que $\mu_X = \mu_Y$, enquanto a hipótese alternativa assume que $\mu_X < \mu_Y$. A partir da distribuição *t* de Student, é obtido o valor crítico da variável *t* para ν graus de liberdade. Se *T* for maior que este valor crítico, rejeita-se a hipótese nula, de modo que a

média μ_X é considerada menor que a média μ_Y. Caso contrário, não se rejeita a hipótese nula, e a média μ_X é considerada igual à média μ_Y.

Referências

[Battiti and Tecchiolli, 1995] R. Battiti and G. Tecchiolli, "Training neural nets with the reactive tabu search", *IEEE Transactions on Neural Networks*, 6:1185-1200, 1995.

[Bishop, 1995] C. M. Bishop, "Parameter Optimization Algorithms", In C. M. Bishop, *Neural Networks for Pattern Recognition*, pp. 253-294, Oxford University Press, 1995.

[Blake and Merz, 1998] C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases", http://www.ics.uci.edu/~mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, CA, 1998.

[Bland, 1993] J. A. Bland, “Nonlinear optimization of constrained functions using tabu search”, *International Journal of Mathematical Education in Science and Technology*, 24, pp. 741-747, 1993.

[Bland, 1994] J. A. Bland, “A derivative-free exploratory tool for function minimization based on tabu search”, *Advances in Engineering Software*, pp. 91-96, 1994.

[Bland and Dawson, 1991] J. A. Bland and G. P. Dawson, “Tabu search and design optimization”, *Computer Aided Design*, 23, pp. 195-201, 1991.

[Boese and Kahng, 1993] K. D. Boese and A. B. Kahng, “Simulated Annealing of Neural Networks: the “Cooling” Strategy Reconsidered”, *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 2572-2575, 1993.

[Boese and Kahng, 1994] K. D. Boese and A. B. Kahng, "Best-So-Far vs. Where-You-Are: Implications for Optimal Finite-Time Annealing", *Systems and Control Letters*, 22(1), pp. 71-78, Jan. 1994.

[Boese et al., 1995] K. D. Boese, D. E. Franklin and A. B. Kahng, "Training Minimal Artificial Neural Network Architectures for Subsoil Object Detection", *Proc. SPIE Aerosense-95: Detection Technologies for Mines and Minelike Targets*, pp. 900-911, April 1995.

[Branke, 1995] J. Branke, “Evolutionary Algorithms for Neural Network Design and Training”, Technical Report n. 322, Institute AIFB, University of Karlsruhe, January 1995.

[Braun and Weisbrod, 1993] H. Braun and J. Weisbrod, “Evolving neural feedforward networks”, *Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms*, pp. 25-32. Springer Verlag, 1993.

[Buck et al., 1965] T. M. Buck, F. G. Allen and M. Dalton, “Detection of chemical species by surface effects on metals and semiconductors”, In T. Bregman e A. Dravnieks (Eds.), *Surface Effects in Detection,* Spartan Books Inc., USA, 1965.

[Cannas et al., 1998] B. Cannas, S. Cincotti, A. Fanni, M. Marchesi, F. Pilo and M. Usai, “Performance analysis of locally recurrent neural networks”, *Int. J. COMPEL*, Vol. 17, pp. 708-716, 1998.

[Cannas et al., 1999] B. Cannas, A. Fanni, M. Marchesi and F. Pilo "A Tabu Search algorithm for optimal sizing of locally recurrent neural networks" *Proc. of 10th Int. Symp. On Theoretical Electrical Engineering* (ISTET '99), Magdeburg, Germany, Sept. 1999, pp. 267-272, 1999.

[Chalmers, 1990] D. J. Chalmers, “The evolution of learning: an experiment in genetic connectionism”, *Proceedings of the 1990 Connectionist Models Summer School* (D. S. Touretzky, J. L. Elman, and G. E. Hinton, eds.), pp. 81-90, Morgan Kaufmann, San Mateo, CA, 1990.

[Chalup and Maire, 1999] S. Chalup and F. Maire, “A Study on Hill Climbing Algorithms for Neural Network Training”, *Proceedings of the 1999 Congress on Evolutionary*

Computation (CEC'99), Volume 3, pp. 2014-2021, Washington, D.C., USA, July 6-9, 1999.

[Chua, 1993] L. O. Chua, "Global unfolding of Chua's circuit", *IEICE Trans. Foundamentals – I*, Vol. E76-A, pp. 704-734, 1993.

[Corana et al., 1987] A. Corana, M. Marchesi, C. Martini and S. Ridella, "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm", *ACM Transactions on Mathematical Software*, 13, pp. 262-280, 1987.

[Craven et al., 1996] M. A. Craven, J. W. Gardner and P. N. Bartlett, "Electronic noses – development and future prospects", *Trends in Analytical Chemistry*, vol. 5, n. 9, 1996.

[Cybenko, 1988] G. Cybenko, "Continuous valued neural networks with two hidden layers are sufficient", Technical Report, Department of Computer Science, Tufts University, 1988.

[Cybenko, 1989] G. Cybenko, "Approximation by superpositions of a sigmoid function", *Mathematics of Control, Signals and Systems*, 2: 303-314, 1989.

[D'Amico et al., 1997] A. D'Amico, C. Di Natale and E. Verona, "Acoustic Devices", In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment,* pp. 197-223, CRC Press, 1997.

[De Souza et al., 1999] J. E. G. de Souza, B. B. Neto, F. L. dos Santos, C. P. de Melo, M. S. Santos and T. B. Ludermir,

"Polypyrrole Based Aroma Sensor", *Synthetic Metals*, 102, pp. 1296-1299, 1999.

[Di Natale et al., 1995] C. Di Natale, F. A. M. Davide, A. D'Amico, G. Sberveglieri, P. Nelli, G. Faglia and C. Perego, "Complex chemical pattern recognition with sensor array: the discrimination of vintage years of wine", *Sensors and Actuators B 24-25:* 801-804, 1995.

[Di Natale et al., 1996] C. Di Natale, F. A. M. Davide, A. D'Amico, P. Nelli, S. Groppelli and G. Sberveglieri, "An electronic nose for the recognition of the vineyard of a red wine", *Sensors and Actuators B 33:* 83-88, 1996.

[Dravnieks and Trotter, 1965] A. Dravnieks and P. J. Trotter, "Polar vapour detection based on thermal modulation of contact potentials", *J. Sci. Instrum., 42:* 624, 1965.

[Duch and Korczak, 1998] W. Duch and J. Korczak, "Optimization and global minimization methods suitable for neural networks", *Neural Computing Surveys 2*, http://www.icsi.berkeley.edu/~jagota/NCS, 1998.

[Fletcher, 1987] R. Fletcher, "Practical Methods of Optimization", Wiley, 1987.

[Gardner, 1987] J. W. Gardner, "Pattern recognition in the Warwick Electronic Nose", *8th Int. Congress of European Chemoreception Research Organisation,* University of Warwick, UK*,* 1987.

[Gardner, 1991] J. W. Gardner, "Detection of vapours and odours from a multisensor array using pattern recognition.

Part 1: principal components and cluster analysis", *Sensors and Actuators B, 4*: 108-116, 1991.

[Gardner, 1993] J. W. Gardner, "Intelligent ChemSADs for artificial odour-sensing of coffee and lager beers", *11th Int. Symp. Olfaction & Taste*, Sapporo, Japan, 1993.

[Gardner and Bartlett, 1991] J. W. Gardner and P. N. Bartlett, *Nanotechnology,* 2, 19, 1991.

[Gardner and Bartlett, 1992] J. W. Gardner and P. N. Bartlett, "Sensors and Sensory Systems for an Electronic Nose", *NATO ASI Series E: Applied Sciences*, Vol. 212, Kluwer, Dordrecht, 1992.

[Gardner and Bartlett, 1994] J. W. Gardner and P. N. Bartlett, "A brief history of electronic noses", *Sensors and Actuators B,* 18-19*,* pp. 211-220, 1994.

[Gardner and Bartlett, 1995] J. W. Gardner and P. N. Bartlett, "Application of conducting polymer technology in Microsystems", *Sensors and Actuators A*, 51, 57-66, 1995.

[Gardner and Hines, 1997] J. W. Gardner and E. L. Hines, "Pattern Analysis Techniques", In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment,* pp. 633-652, CRC Press, 1997.

[Gardner et al., 1990] J. W. Gardner, E. L. Hines and M. Wilkinson, "Application of artificial neural networks to an electronic olfactory system", *Meas. Sci. Technol.* 1: 446-451, 1990.

[Gardner et al., 1992] J. W. Gardner, H. V. Shurmer and T. T. Tan, "Application of an electronic nose to the discrimination of coffee", *Sensors and Actuators B, 6*: 71-75, 1992.

[Gardner et al., 1994] J. W. Gardner, T. C. Pearce, S. Friel, P. N. Bartlett and N. Blair, "A multisensor system for beer flavour monitoring using na array of conducting polymers and predictive classifiers", *Sensors and Actuators B, 18-19:* 240-243, 1994.

[Glover, 1986] F. Glover, "Future paths for integer programming and links to artificial intelligence", *Computers and Operation Research*, Vol. 13, pp. 533-549, 1986.

[Goffe et al., 1994] W. L. Goffe, G. D. Ferrier and J. Rogers, "Global optimization of statistical functions with simulated annealing", *Journal of Econometrics*, 60, pp. 65-99, 1994.

[Grate et al., 1997] J. W. Grate, M. H. Abraham and R. A. McGill, "Sorbent Polymer Materials for Chemical Sensors and Arrays", In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment,* pp. 593-612, CRC Press, 1997.

[Hansen, 1986] P. Hansen, "The steepest ascent mildest descent heuristic for combinatorial programming", *Conf. on Numerical Methods in Combinatorial Optimisation*, Capri, Italy, 1986.

[Harp et al., 1989] S. A. Harp, T. Samad, and A. Guha, "Towards the genetic synthesis of neural networks", *Proc. of the Third Int'l Conf. on Genetic Algorithms and Their*

Applications (J. D. Schaffer, ed.), pp. 360-369, Morgan Kaufmann, San Mateo, CA, 1989.

[Hertz et al., 1995] A. Hertz, E. Taillard, D. de Werra, "A Tutorial on Tabu Search", *Proc. of Giornate di Lavoro* AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes), pp. 13-24, Italy, 1995.

[Holland, 1975] J. H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, MI, 1975.

[Ikegami and Kaneyasu, 1985] A. Ikegami and M. Kaneyasu, "Olfactory detection using integrated sensors", *Proc. 3rd Int. Conf. Solid-State Sensors and Actuators (Transducers'85),* Philadelphia, PA, USA, pp. 136-139, 1985.

[Islam and Murase, 2001] M. M. Islam and K. Murase, "A new algorithm to design compact two-hidden-layer artificial neural networks", *Neural Networks*, 14, 1265-1278, 2001.

[Karaboga and Kalinli, 1997] D. Karaboga and A. Kalinli, "Training recurrent neural networks for dynamic system identification using parallel tabu search algorithm", *12th IEEE Int. Symp. on Intelligent Control*, Istanbul, Turkey, 1997.

[Keller et al., 1994] P. E. Keller, R. T. Kouzes and L. J. Kangas, "Three neural network based sensor systems for environmental monitoring", *IEEE Electro 94*

International Conference, Boston, MA, pp. 378-382, 1994.

[Keller et al., 1995] P. E. Keller, L. J. Kangas, L. H. Liden, S. Hashem and R. T. Kouzes, "Electronic noses and their applications", *IEEE Northcon Technical Applications Conference (TAC'95)*, Portland, OR, 1995.

[Kitano, 1990] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system", *Complex Systems*, (4):461-476, 1990.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C.D. Gellat Jr. and M.P. Vecchi, "Optimization by simulated annealing", *Science*, 220: 671-680, 1983.

[Knox, 1989] J. E. Knox, "The application of tabu search to the symmetric traveling salesman problems", Ph.D. Thesis, University of Colorado, USA, 1989.

[Kohavi, 1995] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", *International Joint Conference on Artificial Intelligence* (IJCAI), 1995.

[Kohl, 1997] D. Kohl, "Semiconductor and Calorimetric Devices and Arrays", In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment, p*p. 533-561, CRC Press, 1997.

[Koza and Rice, 1991] J. R. Koza and J. P. Rice, "Genetic generation of both the weights and architecture for a neural network", 1991.

[Kress-Rogers, 1997] E. Kress-Rogers, “Biosensors and Electronic Noses for Practical Applications”, In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment,* pp. 3-39, CRC Press, 1997.

[Lacerda, 2003] E. G. M. de Lacerda, “Seleção de Modelos de Redes Neurais RBF via Algoritmos Genéticos”, Tese de Doutorado, Centro de Informática, Universidade Federal de Pernambuco, 2003.

[Lacerda et al., 2002] E. G. M. de Lacerda, A. C. P. L. F. de Carvalho and T. B. Ludermir, “A Study of Crossvalidation and Bootstrap as Objective Functions for Genetic Algorithms”, *VII Brazilian Symposium on Neural Networks*, pp. 118-123, Porto de Galinhas–PE, Brazil, November 11-14, 2002.

[Lang and Hinton, 1988] K.J. Lang and G.E. Hinton, “The development of the time-delay neural network architecture for speech recognition”, Technical Report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, PA, 1988.

[Lawrence and Geddes, 1997] C. R. Lawrence and N. J. Geddes, “Surface Plasmon Resonance (SPR) for Biosensing”, In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment,* pp. 149-168, CRC Press, 1997.

[Ludermir and Yamazaki, 2003] T. B. Ludermir and A. Yamazaki, "Neural Networks for Odor Recognition in Artificial Noses", *2003 International Joint Conference on*

Neural Networks (IJCNN 2003), Portland, Oregon, July 20-24, 2003, to be published.

[Marquardt, 1963] D. W. Marquardt, "An algorithm for least-squares estimation of non-linear parameters", *Journal of the Society of Industrial and Applied Mathematics*, 11 (2), 431-441, 1963.

[McCulloch and Pitts, 1943] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics,* 5:115-133, 1943.

[Metropolis et al., 1953] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of state calculations by fast computing machines", *J. of Chem. Phys.*, Vol. 21, No. 6, pp. 1087-1092, 1953.

[Meyer, 1983] P. L. Meyer, "Testes de hipóteses", *Probabilidade – Aplicações à Estatística*, 2ª. Edição, pp. 370-395, Livros Técnicos e Científicos Editora S.A., Rio de Janeiro – RJ, 1983.

[Moerland and Fiesler, 1996] P. D. Moerland and E. Fiesler, "Hardware-friendly learning algorithms for neural networks: an overview", *Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems* (MicroNeuro 96), pp. 117-124, IEEE Computer Society Press, February 12-14, 1996.

[Moncrieff, 1961] R. W. Moncrieff, "An instrument for measuring and classifying odours", *J. Appl. Physiol., 16:* 742, 1961.

[Montana, 1995] D. J. Montana, “Neural Network Weight Selection Using Genetic Algorithms”, S. Goonatilake and S. Khebbal (eds.), *Intelligent Hybrid Systems*, pp. 85-104, John Wiley & Sons, 1995.

[Montana and Davis, 1989] D. Montana and L. Davis, “Training feedforward neural networks using genetic algorithms”, *Proc. of Eleventh Int'l Joint Conf. on Artificial Intelligence*, pp. 762-767, Morgan Kaufmann, San Mateo, CA, 1989.

[Moody and Darken, 1989] J. Moody and C. J. Darken, “Fast learning in networks of locally-tuned processing units”, *Neural Computation*, 1(2):281-294, 1989.

[Murray, 1994] D. Murray, “Tuning neural networks with genetic algorithms”, *AI Expert*, 9:27-31, 1994.

[Norgaard, 1997] M. Norgaard, “Neural Network Based System Identification Toolbox Version 1.1 For Use with MATLAB”, Technical Report 97-E-851, Department of Automation, Technical University of Denmark, Denmark, 1997.

[Pearce et al., 1993] T. C. Pearce, J. W. Gardner, S. Friel, P. N. Bartlett, P. N. and N. Blair, “Electronic nose for monitoring the flavour of beers”, *Analyst*, 118, 371, 1993.

[Persaud and Dodd, 1982] K. Persaud and G. H. Dodd, “Analysis of discrimination mechanisms of the mammalian olfactory system using a model nose”, *Nature, 299:* 352-355, 1982.

[Persaud and Travers, 1997] K. C. Persaud and P. J. Travers, “Arrays of Broad Specificity Films for Sensing Volatile Chemicals”,

In E. Kress-Rogers (Ed.), *Handbook of Biosensors and Electronic Noses: Medicine, Food and the Environment,* pp. 563-592, CRC Press, 1997.

[Pham and Karaboga, 2000] D.T. Pham and D. Karaboga, "Introduction", *Intelligent Optimisation Techniques* (Edited by D.T. Pham and D. Karaboga), pp. 1-50, Springer-Verlag, 2000.

[Porto et al., 1995] V. W. Porto, D. B. Fogel and L. J. Fogel, "Alternative Neural Network Training Methods", *IEEE Expert*, 10(3): 16-22, 1995.

[Prechelt, 1994] L. Prechelt, "Proben1 – A Set of Neural Network Benchmark Problems and Benchmarking Rules", Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany, September, 1994.

[Rumelhart and McClelland, 1986] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing, Volume 1: Foundations*, The MIT Press, 1986.

[Rumelhart et al., 1986] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation", *Parallel Distributed Processing* (Edited by D.E. Rumelhart and J.L. McClelland), Vol. 1, pp. 318-362, Cambridge, MIT Press, 1986.

[Santos, 2000] M. S. Santos, "Construção de um nariz artificial usando Redes Neurais", Tese de Doutorado, Centro de Informática, Universidade Federal de Pernambuco, 2000.

[Schild, 1990] D. Schild, "Chemosensory Information Processing", *NATO ASI Series H: Cell Biology,* Vol. 39, Springer, Berlin, 1990.

[Sexton et al., 1998] R. S. Sexton, B. Alidaee, R. E. Dorsey and J. D. Johnson, "Global optimization for artificial neural networks: a tabu search application", *European Journal of Operational Research* (106)2-3, pp. 570-584, 1998.

[Sexton et al., 1999] R. S. Sexton, R. E. Dorsey and J. D. Johnson, "Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing", *European Journal of Operational Research* (114), pp. 589-601, 1999.

[Skorin-Kapov, 1990] J. Skorin-Kapov, "Tabu search applied to the quadratic assignment problem", *ORSA Journal on Computing*, 2, pp. 33-45, 1990.

[Slater et al., 1993] J. M. Slater, J. Paynter and E. J. Watt, "Multi-layer conducting polymer gas sensor arrays for olfactory sensing", *Analyst*, 118, 379, 1993.

[Stepniewski and Keane, 1997] S. W. Stepniewski and A. J. Keane, "Pruning Back-propagation Neural Networks Using Modern Stochastic Optimization Techniques", *Neural Computing & Applications*, Vol. 5, pp. 76-98, 1997.

[Stork et al., 1990] D. G. Stork, S. Walker, M. Burns and B. Jackson, "Preadaptation in neural circuits", *Proc. of Int'l Joint Conf. on Neural Networks*, Vol. I, (Washington, DC), pp. 202-205, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.

[Su and Sheen, 1992] Yaw-Terng Su and Yuh-Tay Sheen, "Neural Network for System Identification", *Int. J. Systems Sci*, Vol. 23, No. 12, pp. 2171-2186, 1992.

[Sundgren et al., 1991] H. Sundgren, F. Winquist, I. Lukkari and I. Lundström, "Artificial neural networks and gas sensor arrays: quantification of individual components in a gas mixture", *Meas. Sci. Technol.* 2: 464-469, 1991.

[White and Ligomenides, 1993] D. White and P. Ligomenides, "GANNet: a genetic algorithm for optimizing topology and weights in neural network design", *Proc. of Int'l Workshop on Artificial Neural Networks* (IWANN'93), pp. 322-327, Springer-Verlag, 1993.

[Whitley and Bogart, 1990] D. Whitley and C. Bogart, "The evolution of connectivity: Pruning neural networks using genetic algorithms", *Proc. of Int'l Joint Conf. on Neural Networks*, Vol. I, pp. 134-137, Washington, D.C., Lawrence Erlbaum Associates, Hillsdale, N.J., 1990.

[Widrow and Hoff, 1960] B. Widrow and M. E. Hoff, "Adaptive switching circuits. Institute of Radio Engineers", *Western Electronic Show and Convention*, 1960.

[Wilkens and Hatman, 1964] W. F. Wilkens and A. D. Hatman, "An electronic analog for the olfactory processes", *Ann. NY Acad. Sci., 116*: 608, 1964.

[Yamazaki, 2001] A. Yamazaki, "Reconhecimento de Padrões em um Nariz Artificial por Redes Neurais", Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Recife-PE, Junho, 2001.

[Yamazaki and Ludermir, 2001a] A. Yamazaki and T.B. Ludermir, "Classificação de safras de vinho por um nariz artificial com redes neurais", *Encontro Nacional de Inteligência Artificial* (ENIA'2001), pp. 1518-1527, Fortaleza-CE, 30 de julho a 03 de agosto de 2001.

[Yamazaki and Ludermir, 2001b] A. Yamazaki and T.B. Ludermir, "Classification of vintages of wine by an artificial nose with neural networks", *Proceedings of 8th International Conference on Neural Information Processing* (ICONIP'2001), Vol. 1, pp. 184-187, Shanghai, China, November 14 to 18, 2001.

[Yamazaki et al., 2001] A. Yamazaki, T.B. Ludermir and M.C.P. de Souto, "Classification of vintages of wine by an artificial nose using time delay neural networks", *IEE Electronics Letters*, 22nd November 2001, Vol. 37, N. 24, pp. 1466-1467, 2001.

[Yamazaki et al., 2002a] A. Yamazaki, M.C.P. de Souto and T.B. Ludermir, "Optimization of Neural Network Weights and Architectures for Odor Recognition using Simulated Annealing", *2002 International Joint Conference on Neural Networks* (IJCNN'02), Honolulu, Hawaii, pp. 547-552, May 12-17, 2002.

[Yamazaki et al., 2002b] A. Yamazaki, T.B. Ludermir and M.C.P. de Souto, "Simulated Annealing and Tabu Search for Optimization of Neural Networks", *26th Annual Conference of the Gesellschaft für Klassifikation* (GfKl), University of Mannheim, Germany, p. 193, July 22-24, 2002.

[Yamazaki et al., 2002c] A. Yamazaki, T.B. Ludermir and M.C.P. de Souto, "Global optimization methods for designing and

training neural networks", *VII Brazilian Symposium on Neural Networks*, pp. 130-135, Porto de Galinhas–PE, Brazil, November 11-14, 2002.

[Yamazaki and Ludermir, 2003] A. Yamazaki and T.B. Ludermir, "Neural Network Training with Global Optimization Techniques", *International Journal of Neural Systems*, Vol. 13, N. 2, pp. 77-86, April 2003.

[Yao, 1995] X. Yao, "Evolutionary artificial neural networks", *Encyclopedia of Computer Science and Technology* (A. Kent and J. G. Williams, eds.), vol. 33, pp. 137-170, New York, NY 10016: Marcel Dekker Inc., 1995.

[Yao, 1999] X. Yao, "Evolving Artificial Neural Networks", *Proceedings of the IEEE*, 87(9):1423-1447, September, 1999.

[Yao and Liu, 1997] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks", *IEEE Transactions on Neural Networks*, Vol. 8, n. 3, pp. 694-713, 1997.

Tese de Doutorado apresentada por **Akio Yamazaki** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Uma Metodologia para Otimização de Arquiteturas e Pesos de Redes Neurais"**, orientada pela Profa. Teresa Bernarda Ludermir e aprovada pela Banca Examinadora formada pelos professores:

Prof. Francisco de Assis Tenório de Carvalho
Centro de Informática / UFPE

Prof. Aluízio Fausto Ribeiro Araújo
Centro de Informática / UFPE

Prof. Fernando Buarque de Lima Neto
Centro de Informática / UFPE

Prof. André Carlos Ponce de Leon F. de Carvalho
Depto. de Ciências da Computação e Estatística / USP

Prof. Herman Martins Gomes
Departamento de Computação e Sistemas / UFCG

Visto e permitida a impressão.
Recife, 5 de março de 2004.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco