CARLOS MAGNO NUNES

SELEÇÃO DE PRIMITIVAS UTILIZANDO ALGORITMO SUBIDA DE ENCOSTA OTIMIZADO EM PROBLEMAS DE RECONHECIMENTO DE CARACTERES

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do grau de Mestre em Informática Aplicada,
Área de Concentração: *Ciência da Imagem.*

CURITIBA

2004

CARLOS MAGNO NUNES

SELEÇÃO DE PRIMITIVAS UTILIZANDO ALGORITMO SUBIDA DE ENCOSTA OTIMIZADO EM PROBLEMAS DE RECONHECIMENTO DE CARACTERES

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do grau de Mestre em Informática Aplicada,
Área de Concentração: *Ciência da Imagem.*
Orientador:
Prof. Dr. Celso Antonio Alves Kaestner
Co-orientadores:
Prof. Dr. Alceu de Souza Britto Júnior e
Prof. Dr. Robert Sabourin

CURITIBA

2004

Nunes, Carlos Magno
Seleção de Primitivas Utilizando Algoritmo Subida de Encosta Otimizado em Problemas de Reconhecimento de Caracteres, / Carlos Magno Nunes, 2004
095 folhas

Dissertação. Pontifícia Universidade Católica do Paraná, PUCPR, Programa de Pós-graduação em Informática Aplicada, PPGIA, 2004
Orientador: Prof. Dr. Celso Antonio Alves Kaestner
Co-orientadores: Prof. Dr. Alceu de Souza Britto Júnior e Prof. Dr. Robert Sabourin

1. seleção de primitivas. 2. algoritmos de busca. 3. reconhecimento de padrões 4. reconhecimento automático de caracteres

___ – ___.__

Dedico este trabalho à
Elenise,
Eduardo e
Fernanda.

Agradecimentos

Primeiramente agradeço ao Prof. Alceu de Souza Britto Jr., pela confiança e apoio depositados em meu trabalho, principalmente pela serenidade quando eu estava presente e pela paciência quando eu estava ausente. Aos Professores Orientadores e Co-orientadores Celso Antonio Alves Kaestner e Robert Sabourin, pelas críticas e comentários.

Pelo financiamento, agradeço a Pró-Reitoria de Pesquisa e Pós-graduação da Pontifícia Universidade Católica do Paraná.

Aos colegas de laboratório, professores e funcionários do PPGIA pelas informações e troca de experiências.

Agradeço de forma especial a minha família que compreendeu e aceitou os meus objetivos.

Sumário

Lista de Figuras ..... **viii**

Lista de Tabelas ..... **x**

Lista de Abreviaturas ..... **xi**

RESUMO ..... **xii**

ABSTRACT ..... **xiii**

1 Introdução **1**

1.1 Objetivos 2

1.2 Contribuições 2

1.3 Justificativa 3

1.4 Organização do Texto 4

2 Fundamentação Teórica **5**

2.1 Reconhecimento de Padrões 6

2.2 Classificadores 7

2.2.1 Redes Neurais Artificiais *8*

2.2.2 Perceptron Multicamadas MLP *10*

2.2.3 Considerações sobre a Camada Escondida *11*

2.2.4 Número de Camadas Escondidas *13*

2.3 Extração de Primitivas 13

2.3.1 Zoneamento *14*

2.3.2 Informações de Contorno *14*

2.3.3 Primitivas de Fundo *15*

2.3.4 Contagem de Pixels *17*

2.4 Seleção de Primitivas 17

2.4.1 Abordagem Filter *20*

2.4.2 Abordagem Wrapper *21*

2.4.3 Algoritmo de busca Subida de Encosta com Mutação Randômica *22*

2.5 Considerações Finais 25

3 Trabalhos Relacionados **26**

4 Método Proposto.....33

4.1 Estrutura Básica.....34

4.1.1 Extração de Primitivas.....34

4.1.2 Classificador.....36

4.1.3 Épocas de Treinamento.....37

4.2 Seleção de Primitivas.....37

4.2.1 Múltiplos máximos locais.....38

4.2.2 Análise da Sensitividade.....38

4.2.3 Critérios de Parada Considerados.....39

4.2.4 Validando máximos locais.....40

4.2.5 Etapa de análise de resultados.....40

4.3 Otimizações no processo de Seleção de Primitivas.....41

4.3.1 Prioridade de remoção.....41

4.3.2 Seleção do Número de níveis de prioridade.....43

4.3.3 Seleção da primeira primitiva a ser removida.....45

4.3.4 Algoritmo Proposto.....46

4.4 Máximos locais com tolerância na taxa de reconhecimento.....47

4.5 Classificador em Cascata.....48

4.5.1 Complexidade Computacional.....50

4.6 Considerações Finais.....51

5 Experimentos e Resultados.....52

5.1 Base de dados.....52

5.2 Experimentos iniciais.....53

5.2.1 Camada Escondida.....53

5.2.2 Treinamento da Rede Neural.....55

5.2.3 Resultados dos experimentos para ajuste do classificador.....57

5.3 Experimentos para Seleção de Primitivas.....59

5.3.1 Número de subconjuntos avaliados.....59

5.3.2 Experimentos com o algoritmo de busca inicial.....60

5.3.3 Experimentos com as Otimizações Propostas.....63

5.3.4 Resumo do Protocolo Experimental.....63

5.3.5 Experimentos com o conceito de Tolerância.....67

5.3.6 Conjunto de Validação 3 ..... *68*
5.4 Classificador em Cascata 70
5.5 Considerações Finais 72
6 Conclusões 73
Referências 77

Lista de Figuras

Figura 2.1: Elemento de Processamento Perceptron 9
Figura 2.2: Adaptada de [PRI00] - rede MLP com uma camada escondida; 10
Figura 2.3: Adaptada de [OLI01a] - (a) caractere (b) contorno do caractere (c) contorno ampliado parte superior direita (d) Vetor de Primitivas e (e) código de Freeman .. 15
Figura 2.4: Primitivas direcionais modificadas 16
Figura 2.5: Descritores de concavidade adaptado de [BRI01]; 16
Figura 2.6: Métodos de seleção de primitivas, adaptado de [DAS97] 18
Figura 2.7: Classificação de algoritmos, adaptado de [BAR03] 19
Figura 2.8: Taxonomia segundo Jain [JAI97b] 19
Figura 2.9: Abordagem *Filter* , adaptada de [DAV95] 20
Figura 2.10: Diagrama simplificado *Wrapper*, adaptada de [DAV95] 21
Figura 2.11: Simulação de algoritmo Subida de Encosta 24
Figura 2.12: Algoritmo tradicional Subida de Encosta 25
Figura 3.1: Reconhecimento por tamanho da base, adaptada de [LIU02, pág. 198] 27
Figura 4.1: Estrutura básica inicial 33
Figura 4.2: Ajustes iniciais do classificador 34
Figura 4.3: Divisão das bases de dados 35
Figura 4.4: Ajuste manual da rede neural 36
Figura 4.5: Valor médio por primitiva 39
Figura 4.6: Validação dos máximos locais 40
Figura 4.7: Taxa de erro resultante da remoção individual 42
Figura 4.8: Determinação da prioridade da remoção das primitivas 43
Figura 4.9: Seleção do número de níveis de prioridade 44
Figura 4.10: Algoritmo proposto 46
Figura 4.11: Classificador em cascata 50
Figura 5.1: Número de nós da camada escondida 54
Figura 5.2: Determinação do critério de parada da rede neural 55
Figura 5.3: Iterações do algoritmo convencional 61
Figura 5.4: Subconjuntos gerados pelo método inicial 61

Figura 5.5: Resultados das etapas de validação do algoritmo inicial.....62
Figura 5.6: Iterações com conceito de prioridade e nova semente 64
Figura 5.7: Subconjuntos gerados pelo método proposto 65
Figura 5.8: Taxa de erro resultante da remoção individual 66
Figura 5.9: Resultados da etapas de validação do algoritmo proposto 66
Figura 5.10: Número médio de primitivas dos máximos locais 69
Figura 5.11: Comparação do incremento da taxa de erro.....69
Figura 5.12: Comparação entre taxas de rejeição 71

Lista de Tabelas

Tabela 3.1: Abreviaturas dos classificadores utilizados....26
Tabela 3.2: Comparação de resultados sobre diferentes bases de dados 29
Tabela 3.3: Comparação de resultados entre algoritmo RMHC e 1-NN 30
Tabela 3.4: Resultados dos experimentos com técnica backward e forward 31
Tabela 5.1: Divisão dos arquivos da base de dados....53
Tabela 5.2: Resultados da aplicação da fórmula de Baum-Haussler 53
Tabela 5.3: Divisão das bases de dados 57
Tabela 5.4: Resultados iniciais obtidos para caracteres maiúsculos manuscritos 57
Tabela 5.5: Base de caracteres minúsculos manuscritos....58
Tabela 5.6: Resultados para caracteres minúsculos manuscritos 58
Tabela 5.7: Base de dígitos manuscritos 58
Tabela 5.8: Resultados para dígitos manuscritos....58
Tabela 5.9: Resultados para dígitos manuscritos....62
Tabela 5.10: Resultados para caracteres maiúsculos manuscritos....65
Tabela 5.11: Resultados para dígitos manuscritos....67
Tabela 5.12: Características dos módulos do classificador....68
Tabela 5.13: Resultados para rejeição com taxa de erro 0,5% 70
Tabela 5.14: Resultados para rejeição com taxa de erro 0%....71

Lista de Abreviaturas

1-NN	*1-Nearest Neighbor*
GED	*Gerenciamento Eletrônico de Documentos*
HMM	*Hidden Markov Models*
ICR	*Intelligent Character Recognition*
LASER	*Learning Algorithms using SEarch Ring*
MLP	*Multi Layer Perceptron*
NIST	*National Institute of Standards and Technology*
OCR	*Optical Character Recognition*
RMHC	*Random Mutation Hill Climbing*
SCRAP	*Subset selection using Case-based Relevance APproach*
RBF	*Radial Basis Function*
PC	*Polynomial Classifier*
LVQ	*Linear Vector Quantization*
MQDF3	*Modified Quadratic Discriminant Function*
RDA	*Regularized Discriminant Analysis*
QDF	*Quadratic Discriminant Function*
LDF	*Linear Discriminant Function*
SLNN	*Single-layer Neural Network*

RESUMO

O desenvolvimento constante de técnicas de reconhecimento de padrões, de técnicas de extração de primitivas, implementações de algoritmos genéticos e a utilização de força bruta oferecida pela utilização de redes neurais permitem a resolução de problemas de classificação cada vez mais complexos. Este conjunto de facilidades acaba permitindo a utilização de conjuntos de primitivas maiores e mais complexos. De forma proporcional ocorre, porém o aumento da complexidade computacional que pode acabar por inviabilizar a tarefa, pois a utilização de determinadas primitivas sem que se tenha uma avaliação individual, da sua importância ou relevância, pode apenas consumir recursos importantes. Este trabalho apresenta um método de busca Subida de Encosta Otimizado, que tem por objetivo reduzir o conjunto de primitivas necessárias a tarefa de classificação de padrões. Os experimentos trabalham com exemplares de caracteres e dígitos da base NIST, utilizando um conjunto inicial de 132 primitivas. O método proposto apresentou uma redução de até 42% do tamanho inicial do conjunto de primitivas, sem redução dos índices de classificação e, com a utilização do conceito de tolerância no controle do algoritmo foi possível reduzir em até 80% o tamanho do conjunto inicial com uma redução do índice de complexidade computacional de até 86% para uma redução de apenas 1,67% dos índices de classificação. O método apresentado se mostrou uma interessante técnica de implementação de uma abordagem *Wrapper* sem necessariamente utilizar complexas e caras estruturas de hardware.

ABSTRACT

The pattern recognition techniques, feature extraction techniques, genetic algorithms implementation and brute force offered by neural nets allows to solve classification tasks more complex. These facilities allow the use of more complex features groups. In a proportional way happens, even so the increase of computational complexity that can make unfeasible the task, because the feature can be used without an individual evaluation of its meaning or relevance, can just use important resources. This work presents a search method called Optimized Hill Climbing Algorithm that reduces the feature set necessary to patterns classification task. Experiments use the NIST database characters and digits, using a initial set of 132 features. The Optimized Hill Climbing Algorithm reduces up to 42% the feature set initial size. Without reduction in the classification rates and, with the use of the of tolerance concept in the algorithm control, was possible to reduce up to 80% the initial feature set size. With computational complexity reduction up to 86% with reduction of just 1.67% of the classification rates. Proposed method showed an interesting Wrapper implementation technique, without necessarily use complex and expensive hardware structures.

Capítulo

1 Introdução

O surgimento da capacidade de escrita humana representou um marco na história das civilizações. Simbolizou o fim do que ficou conhecido como período Pré-histórico. Teria ocorrido por volta do ano 3.100 a.C. na região sul da Mesopotâmia, e se constituía em gravações cuneiformes feitas em placas de barro. Esta forma particular de comunicação e de registro de conhecimento humano, conta com aproximadamente 5.000 anos de desenvolvimento. O que era, no início, um conjunto de aproximadamente 2.000 símbolos, hoje é representado por um conjunto bem mais reduzido contendo apenas algumas dezenas de símbolos. Os diversos modelos destes símbolos influenciaram a própria imprensa, que surgiu cerca de 500 anos atrás. A origem dos tipos gráficos minúsculos [LOF03 , PIR03] foi baseada nos desenhos de caracteres manuscritos italianos, romanos e carolíngios e seus tipos gráficos maiúsculos foram baseados nos caracteres manuscritos das lápides romanas. E, há apenas 50 anos [TRI96], teve início o estudo de técnicas que visam permitir aos computadores efetuar a leitura ou reconhecimento automático deste tipo de escrita, tão rica em variabilidade e sensível às particularidades de cada indivíduo [FRE02, p.36].

O uso crescente de computadores pode sugerir que a utilização da escrita humana esteja perdendo importância. Porém, o que se observa é justamente o contrário. O volume de documentos gerados, manuscritos ou impressos, continua aumentando, acompanhando uma crescente necessidade de informações para a realização das tarefas diárias. Considerar o conteúdo de tais documentos pode representar o diferencial necessário para uma correta tomada de decisão em Sistemas de Informações Gerenciais, que é definido por [OLI02a] como sendo o processo de transformação de dados em informações que são utilizadas na estrutura decisória da empresa. A digitalização de documentos é a tecnologia básica que permite tal utilização e o reconhecimento automático de caracteres em documentos é o processo específico, ambos ainda

trabalham na obtenção de um modelo definitivo. Assim como a escrita humana precisou passar por um longo período de evolução, tudo o que já foi pesquisado em termos de técnicas de "extração" de informações de documentos manuscritos ou impressos, por meio de sistemas de reconhecimento automático, forma hoje uma base de conhecimento, um ponto de partida para novas pesquisas e experimentações.

1.1 Objetivos

O principal objetivo deste trabalho é propor um método que possa identificar de maneira automática um subconjunto de primitivas, a partir de um conjunto de primitivas inicialmente proposto. Utilizando, para isto, uma estratégia de busca de baixo custo computacional, baseada no algoritmo de busca Subida de Encosta. Para tanto, destacam-se os seguintes objetivos secundários:

a) Verificar a viabilidade da utilização de algoritmos de busca heurística na eliminação de primitivas ineficientes ou redundantes no processo de reconhecimento de caracteres manuscritos;
b) Verificar a eficiência da seleção aleatória de primitivas a serem eliminadas do conjunto principal visando redução da dimensionalidade do espaço de soluções, método conhecido como Mutação Randômica;
c) Verificar o impacto da substituição de primitivas pelo seu valor médio em lugar da sua simples eliminação, conceito conhecido como Análise da Sensitividade;
d) Considerar vários conjuntos subótimos de resposta para entre eles, determinar efetivamente o melhor subconjunto como solução final;
e) Verificar a eficiência da ordenação e priorização na tarefa de remoção de primitivas;
f) Avaliar a utilização do conceito de tolerância sobre as taxas de erro, priorizando a redução do tamanho do conjunto de primitivas;
g) Avaliar o desempenho de um sistema classificador em cascata, que utiliza os classificadores obtidos com o método proposto.

1.2 Contribuições

Este trabalho contribui com um módulo reconhecedor de dígitos e caracteres manuscritos, podendo ser inserido dentro de um aplicativo mais amplo de OCR/ICR. De maneira mais específica coloca a disposição de outros pesquisadores um sistema base

para avaliação da relevância de primitivas, que façam parte de um conjunto maior, utilizadas em tarefas de classificação de padrões. O método apresentado evita a reconfiguração e retreinamento constante dos classificadores utilizados, reduzindo desta forma o tempo destinado às fases de treinamento e de obtenção de um classificador ajustado e adaptado para um determinado contexto. Algumas das otimizações apresentadas no método proposto constituem importantes contribuições para a redução do tempo de processamento e aumento da eficiência de algoritmos de busca, a saber:

a) Utilização de análise da sensitividade na remoção de primitivas. Técnica que dispensa a remontagem das bases de dados e retreinamento a cada nova iteração ou remoção de primitiva. Tem impacto direto na redução do tempo de processamento;
b) Priorização na remoção de primitivas menos relevantes. Conduz o algoritmo de forma mais eficiente na busca de um ponto de máximo da função;
c) Utilização do conceito de diferentes sementes no início da árvore de busca para avaliação de conjuntos de primitivas. Exige que a primeira primitiva removida após a obtenção de um máximo local seja diferente das primeiras anteriormente removidas. Atua como uma exigência de que a busca do máximo da função siga em direção diferente das testadas anteriormente;
d) Aplicação do conceito de tolerância ao analisar região próxima aos máximos locais. A tolerância permite ao algoritmo continuar a remover primitivas mesmo no caso de piorar a função critério, mas assegurando conjuntos ainda mais reduzidos de primitivas com pequena piora dos resultados finais de classificação;
e) Avaliação de um classificador modular, com diferentes taxas de rejeição. Demonstra que as técnicas apresentadas são perfeitamente adaptáveis a classificadores em cascata;

1.3 Justificativa

A questão da variabilidade dos padrões impõe limitações em relação ao uso de um conjunto restrito de características capazes de operar com resultados satisfatórios de reconhecimento. Anos de pesquisa colocaram a disposição grande quantidade de

técnicas de extração de primitivas, que podem ser classificadas como estatísticas, estruturais e sintáticas, tornando o conjunto de primitivas extremamente complexo e difícil de avaliar. Diferentes conjuntos de primitivas são adequados cada qual a uma pequena gama de aplicações. Mesmo ao considerar especificamente dígitos e caracteres manuscritos, a quantidade de primitivas disponíveis ainda é muito grande. Existem primitivas baseadas em propriedades invariantes, outras que prevêem distorções e variações. O problema está justamente em determinar qual o impacto da presença ou não de uma determinada primitiva sobre a efetiva taxa de reconhecimento do conjunto sob avaliação. Técnicas não automatizadas de seleção tornam-se inviáveis para conjuntos com elevado número de primitivas e quando submetidos a limitações no número de combinações experimentadas pode não verificar todo o espaço de soluções e terminar por apresentar resultados insatisfatórios. O uso de técnicas automáticas de seleção de primitivas, ao permitir a inclusão de regras e de conhecimentos previamente adquiridos na seleção de subconjuntos apresenta taxas de classificação iguais às obtidas com o conjunto completo, porém com maior capacidade de separação de classes por primitiva e com menor complexidade computacional.

1.4 Organização do Texto

O presente documento está assim dividido. O Capítulo 2, Fundamentação Teórica, apresenta o embasamento teórico necessário para uma compreensão geral dos métodos e técnicas utilizados. O Capítulo 3, Trabalhos Relacionados, é composto por uma análise de publicações recentes e que de maneira direta ou indireta contribuíram na elaboração da metodologia apresentada. O Capítulo 4, Método Proposto, descreve detalhadamente a metodologia de trabalho onde são feitas considerações a respeito dos experimentos. No Capítulo 5, Experimentos e Resultados, são apresentados os gráficos e tabelas dos experimentos efetuados. O documento encerra com as Conclusões e considerações finais, listando ainda as referências utilizadas neste trabalho e necessárias para o perfeito entendimento dos assuntos abordados.

Capítulo

2 Fundamentação Teórica

O uso comercial de aplicativos para OCR[1] sugere que existe mercado e também aplicativos de qualidade [MEL99] e que dentro de determinado contexto, ou limitações, contribuem para a digitalização de informações impressas. Aplicativos OCR podem estar inseridos em aplicativos mais completos voltados para aplicações de GED[2]. Porém não é comum encontrar em tais aplicações, capacidade para ICR[3], que se propõe ao reconhecimento de caracteres manuscritos, devido principalmente às deficiências e limitações que as técnicas mais atuais de ICR ainda possuem. O uso de técnicas computacionais baseadas em utilização maciça de Redes Neurais [OLI01a] e/ou Modelos Escondidos de Markov [BRI02] estão conduzindo a resultados satisfatórios para determinados conjuntos de padrões. É o caso do reconhecimento de números manuscritos [OLI02b], com taxas de reconhecimento para dígitos isolados que atingem cerca de 99% considerando-se a base NIST[4].

Um dos problemas iniciais a ser trabalhado na tarefa de construção de um sistema de reconhecimento automático de dígitos ou caracteres manuscritos, é o de escolher, ou selecionar, um conjunto de primitivas que seja robusto com relação à variabilidade dos padrões que serão submetidos à tarefa de classificação. Sabe-se "a priori" que a imensa capacidade humana de identificar, em uma determinada imagem, todos os caracteres ali presentes, acaba por permitir uma enorme quantidade de variações e imperfeições na escrita e conseqüentemente nas classes de símbolos ou padrões. Identificar e representar matematicamente ou estatisticamente esta capacidade de percepção humana [FRE02, p. 49] é o desafio central dos aplicativos ICR.

---

[1] OCR – Optical Character Recognition
[2] GED – Gerenciamento Eletrônico de Documentos
[3] ICR – Intelligent Character Recognition
[4] National Institute of Standards and Technology

2.1 Reconhecimento de Padrões

Você está no elevador junto com várias outras pessoas e determinados traços de semelhança começam a chamar a atenção para o rosto de uma delas. Ela está utilizando um crachá de identificação preso à roupa. Em meio ao movimento das pessoas é possível em um único relance observar parte do primeiro nome que está escrito em letras mais destacadas. Em mais uma fração de tempo é possível concluir: - não era quem parecia ser.

Esta situação mostra a facilidade com que o ser humano altera e incorpora novos padrões como parte de seu raciocínio. Fazer com que os computadores tenham a capacidade de resolver problemas práticos que envolvam reconhecimento, descrição e classificação é o objeto de estudo da área de Reconhecimento de Padrões.

De acordo com [JAI00] o interesse pela área de reconhecimento de padrões aumentou nos últimos anos devido a novas aplicações nas áreas computacionais de DataMining[1], classificação de documentos, organização e recuperação de bases de dados de multimídia, entre outras. São aplicações específicas e dentro de um contexto limitado. Algumas delas, tais como, o reconhecimento de caracteres e dígitos manuscritos exige atenção especial, principalmente em relação a grande variabilidade de exemplos dentro da mesma classe e também variabilidade para um mesmo escritor [JUS01]. Para atender a tantas aplicações, muitas técnicas de reconhecimento foram surgindo e a área de reconhecimento de padrões pode ser dividida, atualmente, nas seguintes abordagens: comparação de modelos ou padrões, estatística, sintática ou estrutural e baseada em redes neurais.

A abordagem por comparação de modelos ou padrões é uma das mais antigas e pode ser resumida numa operação de verificação de similaridade entre duas amostras de um mesmo tipo. A partir de um modelo ou protótipo do padrão a ser reconhecido, uma comparação com um exemplo do padrão é efetuada, verificando uma taxa de similaridade entre os dois padrões. Variações, desta abordagem, que façam uso de modelos deformáveis ou elásticos [JAI97a], também podem ser utilizadas.

Na abordagem estatística [SCH92, pág. 33], os exemplos dos padrões são representados por vetores n-dimensionais. O objetivo é fazer com que o conjunto de amostras, ou seus respectivos vetores, sejam distribuídos em um espaço multidimensional, de maneira que as amostras pertencentes a uma mesma classe fiquem

[1] Do inglês – Mineração de Dados

agrupadas e ao mesmo tempo distantes de todas as outras amostras de diferentes agrupamentos de classes. A eficiência da técnica pode ser verificada pela obtenção de fronteiras definidas entre as classes de padrões. A classificação de cada amostra pode ser previamente conhecida ou desconhecida.

A abordagem sintática [SCH92, pág. 127] para reconhecimento de padrões, pode ser útil em problemas mais complexos quando há necessidade de uma avaliação hierárquica. Os padrões são vistos como se fossem seqüências de primitivas e facilmente pode-se representar um grande número de diferentes classes com um conjunto reduzido de primitivas.

A utilização de redes neurais [SCH92, pág. 203] apresenta características de aprendizado, generalização, adaptação e tolerância a falhas aplicadas a um conjunto de pesos que automaticamente avalia a sua eficiência em produzir uma saída coerente com o conjunto de exemplos de padrões aplicados a ela.

2.2 Classificadores

Entre as várias definições para classificadores é possível citar [DUD73] onde se afirma que o problema central da classificação é o de dividir o espaço de primitivas em regiões, ou seja, uma região distinta do espaço de soluções para cada categoria ou classe. Em [HOR94, pág. 06] o termo classificação é apresentado como tendo dois significados distintos. O primeiro significado diz ser possível fazer um conjunto de observações com o objetivo de estabelecer a existência de classes ou *clusters*[1] de dados, e o segundo significado diz que a partir do conhecimento de que existem muitas classes, procura-se estabelecer uma regra de forma a tornar possível classificar uma nova observação em uma das classes já existentes. O primeiro modo é conhecido como aprendizagem não supervisionada ou "por agrupamento"[2] e o segundo modo é chamado de aprendizagem supervisionada.

A aprendizagem supervisionada sugere que existe alguém com capacidade para uma classificação perfeita, contemplando todos os exemplos propostos. Desta forma, é possível questionar sobre os motivos pelo qual deve-se substituir uma classificação perfeita ou exata por uma classificação com algum erro ou aproximação. Algumas justificativas para tal atitude:

---

[1] Grupos.
[2] Em inglês: clustering.

a) Os processos de classificação informatizados podem ser efetuados de maneira muito mais rápida;
b) Processos automáticos vão tomar decisões baseadas unicamente nas informações consideradas, enquanto que a decisão humana pode sofrer influências de uma análise subjetiva que comprometam a confiabilidade das decisões.

Em relação à aprendizagem não supervisionada, não existem exemplos de entrada cujas saídas sejam conhecidas. O processo de aprendizagem deve desenvolver suas representações de associação, baseando-se em cálculos efetuados com os exemplares atuais e com valores armazenados. Um exemplo de Redes de aprendizagem não supervisionada [PRI00], é a rede Kohonen, cujo objetivo é mapear um conjunto de entradas, pertencentes a um espaço contínuo para um conjunto de saídas pertencentes a um espaço discreto. Produz uma saída em um espaço de menor dimensão preservando a topologia dos exemplos de entrada. Desenvolvida na década de 80, procura aplicar modelos matemáticos cada vez mais próximos a modelos biológicos do córtex cerebral de animais superiores.

2.2.1 Redes Neurais Artificiais

As redes neurais artificiais podem ser entendidas como uma subdivisão da inteligência artificial, ou como uma técnica de modelagem matemática que se presta muito bem para a classificação e reconhecimento de padrões em sistemas complexos. Existem registros de estudos iniciais por volta do ano de 1942, onde pesquisas foram financiadas durante a Segunda Guerra Mundial. Apesar dos avanços obtidos com a idealização da máquina de Turing em 1936 e também da descoberta do Perceptron por Rosenblatt em 1957, que se propunha a ser um método simples de treinamento de pesos que convergiam para a solução de um problema caso ela existisse, em 1969 foram levantadas sérias dúvidas sobre a capacidade de resolução para problemas não linearmente separáveis.

Durante quase 20 anos pouca pesquisa foi patrocinada nesta área. Somente durante a década de 80 é que começaram a surgir inúmeras publicações a respeito, quando se verificou que redes implementadas com um maior números de camadas poderia avançar sobre problemas não linearmente separáveis.

Outro impulso recebido foi o desenvolvimento no *hardware* dos equipamentos envolvidos. Pesquisadores de redes neurais artificiais têm hoje a sua disposição, supercomputadores desenvolvidos com até 10.000 processadores [AMD04] trabalhando em paralelo, e com capacidades de processamento da ordem de 40 Teraflops[1]. Tais sistemas permitem a simulação de problemas complexos, tais como meteorologia, energia nuclear, etc.

O *Perceptron* é o elemento de processamento básico para a compreensão das redes neurais. Usado para a resolução de problemas quando o conjunto de treinamento apresenta padrões linearmente separáveis, não sendo possível sua aplicação a um tipo qualquer de conjunto de dados.

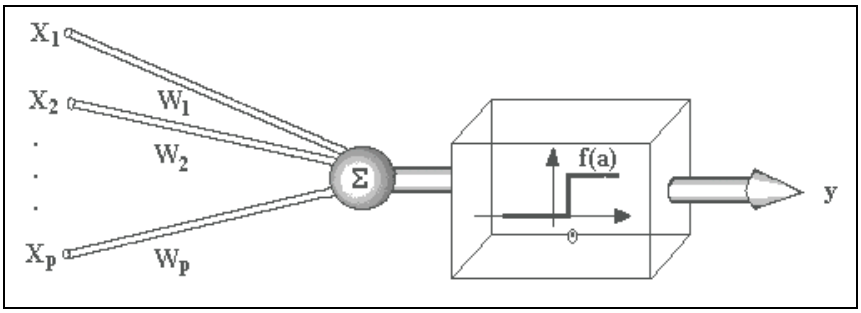

Figura 2.1: Elemento de Processamento Perceptron

É possível ainda incluir uma "tendência" no resultado final ao que é chamado de *bias*. Existindo um conjunto finito de exemplares presentes no conjunto de Treinamento, com componentes inteiros (ou racionais), o algoritmo de aprendizagem do *Perceptron* irá produzir, em tempo finito, um vetor de pesos que satisfaça a todos os exemplos de treinamento, para o caso de exemplares linearmente separáveis.

E para o caso de não serem separáveis a opção é tentar encontrar um vetor de pesos W* que classifique tantos exemplos de treinamento quanto possível. A este conjunto de pesos chamamos de "*ótimo"*. Este conjunto de pesos possui, desta forma, a capacidade de reter conhecimento, baseado na observação dos exemplares anteriores. Aprender com os erros e acertos transformando-os em auxílio na tomada de decisões futuras. A partir da estrutura básica que envolve o elemento de processamento *Perceptron*, é possível compreender esta importante característica apresentada pelas redes neurais.

[1] Representa trilhões de operações aritméticas com operadores de ponto flutuante.

Para que uma rede neural possa ser efetivamente utilizada, deve completar a sua etapa de aprendizagem, que consiste basicamente no ajuste do conjunto de pesos. Aproveitando a estrutura simples de um *Perceptron* é possível destacar os seguintes componentes: variáveis de entrada, conjunto de pesos, elementos de processamento e valor da saída. O aprendizado das redes neurais pode ser efetuado de duas maneiras citadas anteriormente: forma supervisionada e forma não supervisionada. No método aqui proposto a aprendizagem supervisionada de redes será a técnica utilizada. Para cada exemplo aplicado na entrada, o valor esperado da saída é previamente conhecido. O valor de saída é efetivamente calculado através da equação 2.1:

$$S = W_0 + \sum_{j=1}^{p} W_j . u_j \qquad \textbf{(2.1)}$$

A partir das variáveis de entrada é calculado um valor para a saída (S). Como "a priori" o valor da saída é conhecido, calcula-se um erro entre estes dois valores, o observado e o dado. Este erro é então utilizado para a correção (recálculo) dos pesos (W) na rede, com a intenção de que a rede aprenda a reconhecer um determinado padrão quando este lhe for apresentado às entradas.

2.2.2 Perceptron Multicamadas MLP

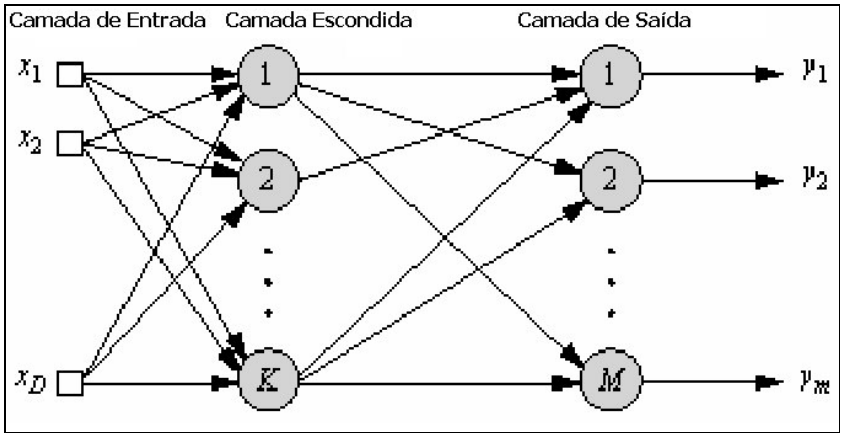

Figura 2.2: Adaptada de [PRI00] - rede MLP com uma camada escondida;

Uma limitação ao uso de redes *Perceptron* simples é a necessidade de um conjunto de padrões linearmente separáveis. Um caso extremamente simples e clássico desta limitação seria a tentativa do aprendizado da função lógica OU-Exclusivo, que não pode ser aprendida por uma rede *Perceptron*.

Com o desenvolvimento de redes multicamada em meados de 1980, tornou-se possível à resolução deste tipo de problema. Surgiu o algoritmo conhecido como *backpropagation*[1] trabalhando em redes multicamada. Uma rede MLP pode ter um número variável de camadas que são assim denominadas: camada de entrada, camada ou camadas escondidas e camada de saída. Cada uma destas camadas é composta por um número variável de nós ou elementos de processamento. Em relação às camadas de entrada e de saída, ocorre uma situação relativamente simples. Basta verificar o número de primitivas existentes (camada de entrada) e o número de classes conhecidas ou esperadas (camada de saída). Porém em relação aos nós das camadas escondidas, devem ser feitas mais algumas considerações.

2.2.3 Considerações sobre a Camada Escondida

Em relação ao efeito do número de nós da camada escondida, podem ocorrer duas situações extremas segundo [PRI00, pág. 143]:

a) Utilização de um número de nós em quantidade maior que a necessária ou;

b) Utilização de um número de nós em quantidade menor que a necessária.

Como primeiro caso serão considerados os problemas que ocorrem quando o número de nós é maior do que o necessário. No caso de um problema poder ser resolvido com três nós, ele também poderá ser resolvido com seis. A chance de se obter uma classificação correta aumenta e normalmente é requerido um número menor de iterações com um correspondente aumento do esforço computacional exigido. Um segundo problema consiste no fato da rede poder memorizar o conjunto de treinamento.

Caso seja escolhido um número de nós inferior ao necessário, pode ocorrer que a rede não consiga aprender a resolver o problema de forma correta. Os pesos irão tentar convergir para valores em que a maior quantidade de exemplos seja corretamente classificada. Para que isto ocorra os pesos ficam oscilando bruscamente seus valores entre as iterações. Uma maneira simples de verificar a ocorrência desta condição é observar esta oscilação dos pesos entre as iterações.

[1] Realiza uma retro-propagação do erro da saída para as camadas anteriores.

Ainda segundo [PRI02], o número de nós da camada escondida depende de uma série de fatores complexos:

a) Número de entrada e saídas;
b) Número de exemplos de treinamento;
c) Ruído presente nos exemplos;
d) Complexidade da função de classificação que deve ser aprendida;
e) Arquitetura de rede;
f) Tipo de função de ativação;
g) Algoritmo de treinamento;
h) Critério de parada.

Em várias situações não há meios de determinar o melhor número de nós da camada escondida sem efetivamente treinar muitas redes e considerar o erro de generalização de cada uma delas. Com poucos nós haverá alta taxa de erro e alta taxa de generalização e alto *bias*[1] devido ao efeito de *underfitting*[2]. No caso de uma rede configurada com muitos nós, é possível obter baixo erro de treinamento, mas com alta taxa de especialização e alta variância devido ao *overfitting*[3]. Em [GEM92] *apud* [SAR04] é apresentada discussão sobre a negociação entre *bias*/variância.

Alguns pesquisadores apresentam fórmulas empíricas[4] para auxiliar na escolha de uma arquitetura. Porém, várias destas regras deixam de fazer sentido, em uma visão mais genérica, por desconsiderarem vários dos fatores citados acima. Desta forma é possível construir contra-exemplos para desaprovar tais regras.

Uma boa forma de escolha do número de nós da camada escondida é através da configuração de várias redes com diferentes números de nós da camada escondida. Depois avaliar o erro para cada uma delas e escolher a rede com o menor deles. De qualquer modo pode-se lançar mão de uma das várias regras empíricas disponíveis. Com o objetivo inicial de proporcionar um ponto de partida junto ao processo de determinação da topologia utilizada. A equação 2.2 apresenta a descrição de uma destas regras [BAU88] chamada de Regra de Baum-Haussler, que sugere o seguinte:

---

[1] Desvio dos resultados, viés.
[2] Baixo nível de ajustamento em relação ao conjunto de treinamento.
[3] Alto nível de especialização em relação ao conjunto de treinamento.
[4] Do original inglês – "rules of thumbs".

$$N_{hidden} \leq \frac{N_{treino}.E_{tolerado}}{N_{pontos} + N_{saídas}} \qquad \textbf{(2.2)}$$

sendo : N_{hidden} número de nós da camada escondida; N_{treino} número de exemplos de treinamento; $E_{tolerado}$ erro permitido ou tolerado; $N_{saídas}$ número de nós da saída; N_{pontos} número de nós da entrada. Esta fórmula normalmente assegura que a rede generalize em lugar de memorizar.

2.2.4 Número de Camadas Escondidas

Convém citar que existem casos em que não há necessidade de se configurar uma topologia contendo camadas escondidas. Casos em que os exemplos de treinamento podem ser linearmente distribuídos. Por outro lado considerações, a respeito do aumento da complexidade, podem ser aplicadas ao uso de uma topologia com maior número de camadas escondidas. A rede neural com uma camada escondida é conhecida como um classificador universal, conseguindo atingir o objetivo de separar conjuntos de dados mais complexos.

2.3 Extração de Primitivas

Extração de primitivas [TRI96] é um dos fatores mais importantes para se atingir altos níveis de reconhecimento em sistemas de reconhecimento de caracteres. Existem diferentes métodos de extração, entre eles os baseados em informação de contorno e esqueletização[1].

Em Devijver e Kittler [DEV82] a extração de primitivas, é definida como um problema de "extrair de dados brutos, não trabalhados, a informação mais relevante para propósitos de classificação, de maneira a minimizar a variabilidade dentro da classe e maximizar a variabilidade entre classes".

Em [TRI96] tem-se a seguinte citação:

> "... Pode-se dizer que existe um número limitado de características independentes, que podem ser extraídas da imagem de um caractere e que desta forma o conjunto de características a ser usado não é importante.

[1] Técnica de afinamento da imagem de caracteres até que restem apenas linhas de um *pixel* de largura.

> Entretanto devemos lembrar que as características devem ser invariantes as distorções esperadas e as variações de cada caractere..."

Tomando esta citação por base, são descritas algumas técnicas, utilizadas em reconhecimento de caracteres, com o objetivo de construir um conjunto de primitivas robustas em relação a ruídos e invariantes às distorções apresentadas por cada classe. A base de dados utilizada é a base NIST [GAR92, BRI00 e KOE03] considerada uma referência como base de dados alfanuméricos manuscritos. A variabilidade presente nesta base permite atestar a qualidade do conjunto de primitivas utilizado.

2.3.1 Zoneamento[1]

Corresponde a uma divisão lógica da imagem a ser analisada. Pode ocorrer desde uma divisão exata em partes rigorosamente iguais em tamanho, totalmente independente da imagem, até uma divisão totalmente dependente do conteúdo da imagem [RAD03]. O zoneamento de imagens permite que sejam extraídas características de detalhes das imagens. Um mesmo tipo de primitiva é extraído em diferentes áreas da imagem do exemplar. As características do zoneamento aqui utilizado seguem o protocolo utilizado em [OLI01a]. A imagem do caractere é dividida em 6 partes de igual tamanho e número de *pixels*[2]. Isto permite uma redução na quantidade de primitivas diferentes extraídas sem perder informações de detalhes da imagem.

2.3.2 Informações de Contorno

O conjunto de primitivas é utilizado e descrito em [OLI01a]. As imagens dos caracteres são lidas ou analisadas normalmente da esquerda para a direita e da parte superior para a inferior, desta forma o primeiro *pixel* a ser analisado no exemplo apresentado na Figura 2.3 está representado na segunda parte da letra "t", veja parte (a). A partir de então é verificada a direção em que se encontra o próximo *pixel* preto, conforme Código Freeman[3] Figura 2.3 parte (e). De acordo com a orientação dos *pixels* pretos é então construído, um vetor que irá conter a informação de direção da linha de contorno do caractere. Este vetor tem oito posições que representam as oito direções possíveis.

---

[1] Em inglês, *zoning*.
[2] Do original inglês "picture element" ou elemento de imagem.
[3] Código direcional de 0 até 7, utilizado para indicar a direção de continuidade da imagem.

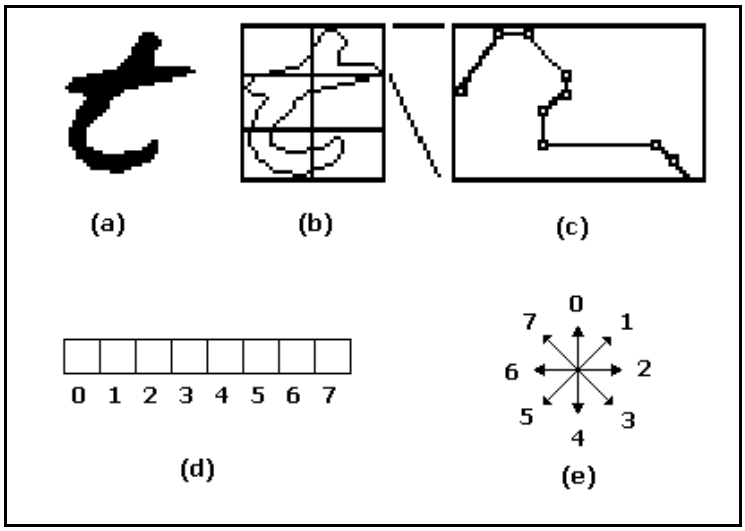

Figura 2.3: Adaptada de [OLI01a] - (a) caractere (b) contorno do caractere (c) contorno ampliado parte superior direita (d) Vetor de Primitivas e (e) código de Freeman

2.3.3 Primitivas de Fundo

As primitivas de fundo são baseadas em informações de concavidade das formas dos exemplares. São utilizadas para destacar propriedades topológicas e geométricas de cada classe. Cada primitiva de concavidade representa o número de *pixels* que pertence a uma específica configuração de concavidade. O nome para cada *pixel* branco, ou seja, os *pixels* do fundo, são escolhidos de acordo com um código direcional de quatro direções, baseado no código de Freeman. Cada direção é explorada até encontrar com um *pixel* preto ou até atingir os limites do caractere definidos por um *bounding box*[1]. O *pixel* branco é rotulado apenas quando forem encontrados *pixels* pretos em no mínimo duas direções.

[1] Moldura imaginária que toca as extremidades da imagem

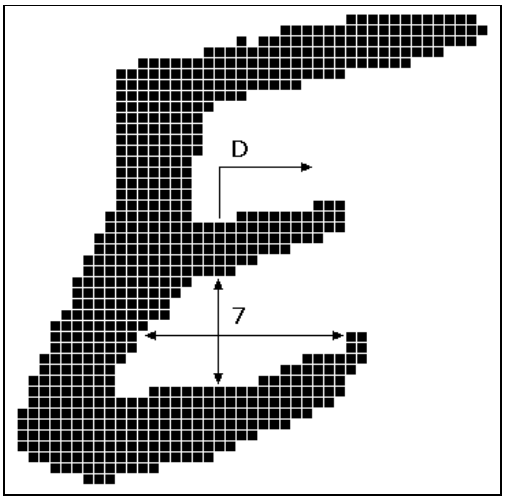

Figura 2.4: Primitivas direcionais modificadas

Na Figura 2-4 se destaca a necessidade de utilizar configurações mais elaboradas de forma a levantar todas as concavidades existentes no caractere. Tal abordagem foi utilizada por [BRI01] no reconhecimento de dígitos manuscritos, podendo ser aplicada para o efetivo reconhecimento, tanto de caracteres quanto de dígitos manuscritos. São 13 os tipos de rotulação que um *pixel* pode assumir.

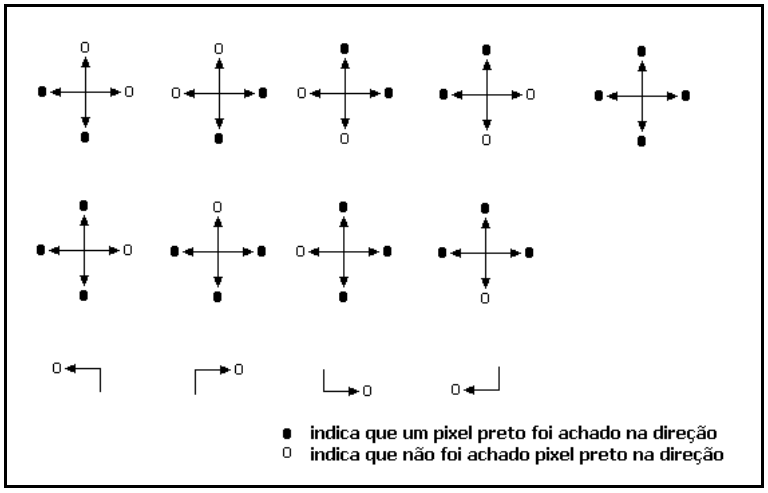

Figura 2.5: Descritores de concavidade adaptado de [BRI01];

Com tais descritores é montado um vetor que contém a somatória da ocorrência de cada uma das possibilidades de concavidade por zona da imagem do caractere.

2.3.4 Contagem de *Pixels*

Último conjunto de primitivas, simples de ser obtido, mas nem por isso menos eficiente. É extraído diretamente dos *pixels* do caractere. Efetua-se uma contagem de *pixels* pretos em cada uma das zonas da imagem dividida, o que totaliza seis primitivas.

2.4 Seleção de Primitivas

Forma de identificar quais as primitivas são mais relevantes e que contribuem para uma maior separação interclasse e, ao mesmo tempo, com uma menor dispersão intraclasse. Pode ser visto como uma maneira de eliminar informação irrelevante ou redundante de um vetor de primitivas.

Ao considerar o uso de bases de dados com número de exemplares relativamente pequeno em relação a um elevado número de dimensões, ou de primitivas, pode ocorrer uma condição conhecida como "maldição da dimensionalidade" [1], que é o aumento exponencial da complexidade em função do aumento linear do número de primitivas consideradas para a representação espacial ou separação em classes. Tal situação pode inviabilizar o uso do conjunto de primitivas. Esta já seria uma justificativa para uma redução do número de dimensões da base de dados através da redução do número de primitivas, mas além disto esta redução provoca uma redução do esforço computacional envolvido na classificação propriamente dita.

Em reconhecimento de padrões tal abordagem, que visa a uma redução da dimensionalidade, é conhecida também como o aprimoramento de técnicas de Seleção de Primitivas[2]. São encontradas diversas publicações que abordam o assunto seleção de primitivas, considerando-a sob diferentes aspectos. Tais publicações propõem uma personalizada taxonomia da área. Em [DAS97] é apresentada uma das classificações mais abrangentes, considerando aspectos de técnicas de geração dos possíveis subconjuntos de resposta. São consideradas três técnicas de geração: Completas (exaustivas), Heurísticas ou Randômicas. A Figura 2-6 apresenta todas as opções disponíveis:

[1] Em inglês: Curse of Dimensionality
[2] Em ingles: Feature Selection

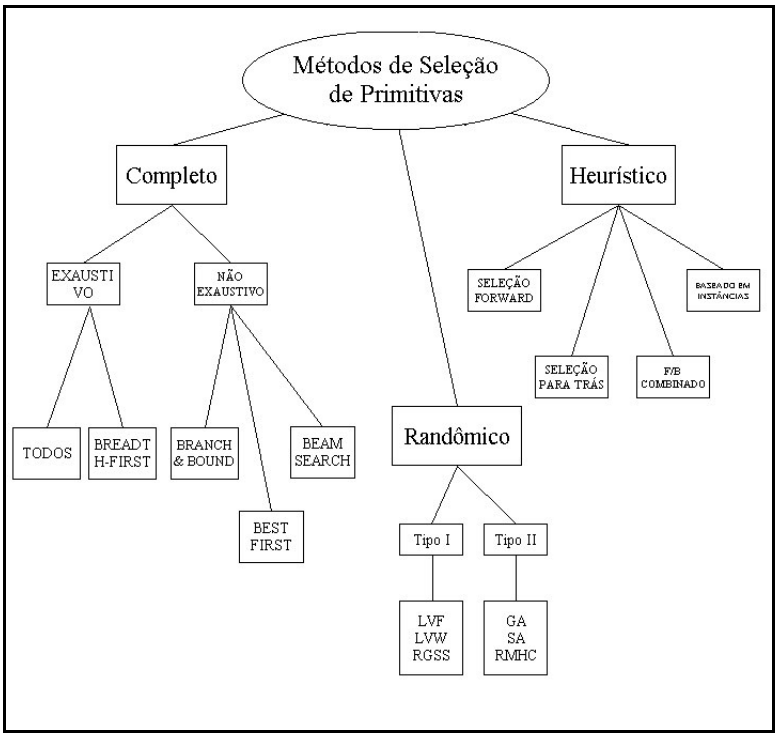

Figura 2.6: Métodos de seleção de primitivas, adaptado de [DAS97]

Em [RAM03] também são três os argumentos utilizados para a classificação. É considerada a relevância das técnicas em: construir hipóteses, melhorar a precisão da estimação e concepção das primitivas. A técnica de construção de hipóteses consistentes diz que não pode haver dois exemplos de primitivas iguais e que pertençam a classes diferentes. O segundo conjunto de técnica de seleção tenta otimizar a precisão da estimação do algoritmo de aprendizagem. A terceira técnica esta baseada na sensibilidade da primitiva em relação ao contexto. Indica a correlação entre a primitiva e o espaço alvo. A divisão é apresentada no seguinte diagrama:

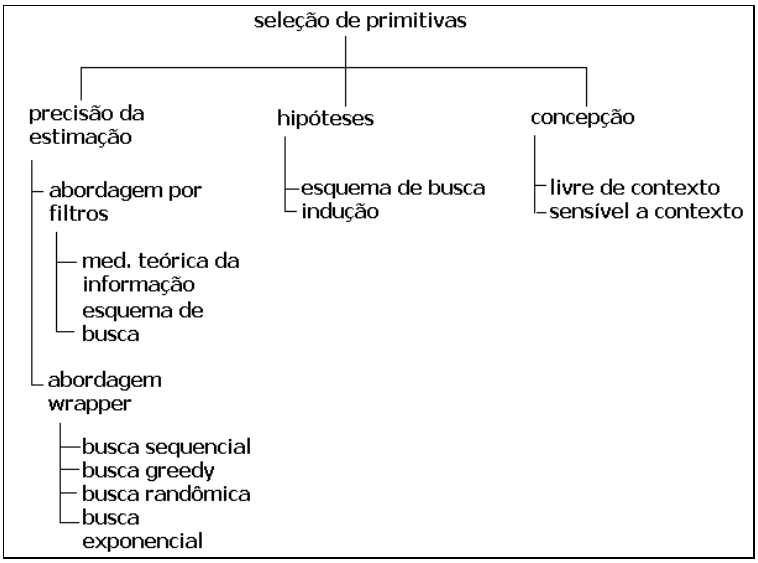

Figura 2.7: Classificação de algoritmos, adaptado de [BAR03]

Em [JAI97b] são feitas observações mais abrangentes, mas que consideram aspectos da implementação dos algoritmos de seleção de primitivas. A forma de implementação é dividida inicialmente em técnicas estatísticas e de redes neurais:

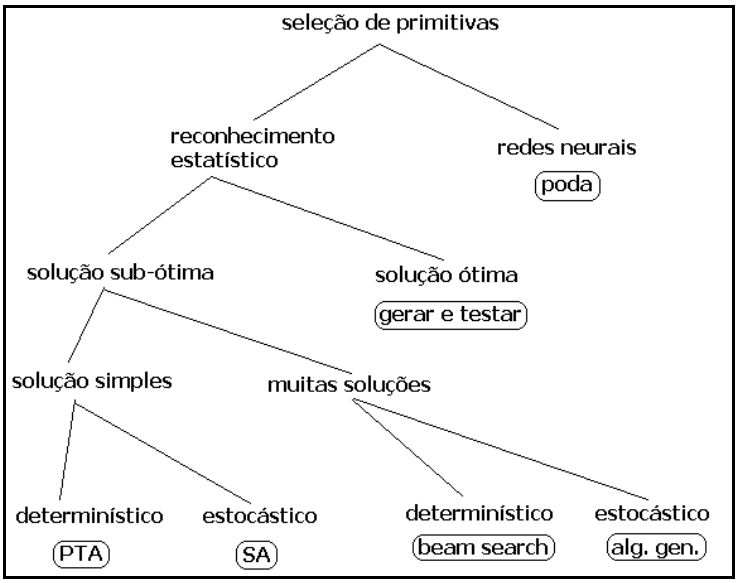

Figura 2.8: Taxonomia segundo Jain [JAI97b]

É provável que uma determinada técnica clássica de seleção apareça em mais de uma das classificações disponíveis. Tomando como exemplo a técnica: Subida de

Encosta com Mutação Randômica[1]. Tal abordagem está inserida dentro das diferentes classificações a disposição. Em [DAS97] é possível identificar de forma direta a técnica citada. Seria uma técnica randômica de tipo II: também conhecida pela sua sigla RMHC. Técnica a ser discutida nos próximos subitens. Em [BAR03] a técnica de Subida de Encosta pode ser classificada como uma técnica tipo precisão de estimação, mais precisamente do tipo abordagem *Wrapper* com pesquisa randômica. Em [JAI97] a técnica Subida de Encosta é classificada como estatística e subótima de múltiplas soluções com processo estocástico.

De todas as classificações apresentadas uma apresenta divisões em relação à forma de implementação do algoritmo de seleção. Em [BLU97] é apresentada uma revisão de métodos de seleção de primitivas com a seguinte classificação: métodos *Embedded*[2], métodos *Wrapper*[3] e métodos *Filter*[4]; sobre os dois últimos, é feita a seguir uma descrição detalhada.

2.4.1 Abordagem *Filter*

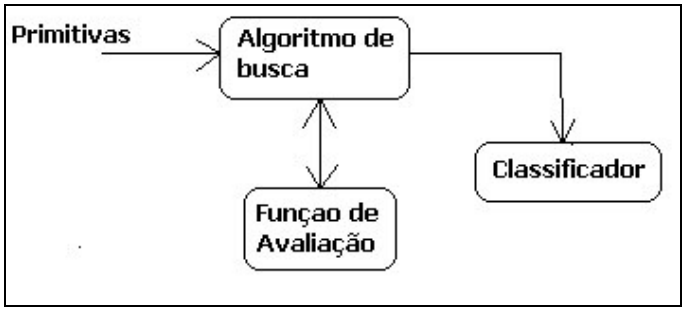

Figura 2.9: Abordagem *Filter* , adaptada de [DAV95]

Esta abordagem introduz um processo separado com o propósito de selecionar o subconjunto de primitivas, que irá ocorrer antes do processo de indução. Atua como um pré-processamento, considerando as características em geral da base de treinamento para escolher algumas e excluir outras.

[1] Do inglês: Random Mutation Hill Climbing
[2] Do inglês: embutido
[3] Do inglês: empacotado
[4] Do inglês: filtro

2.4.2 Abordagem *Wrapper*

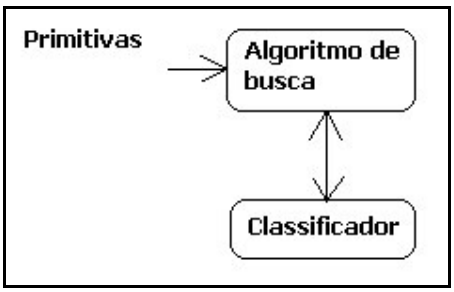

Figura 2.10: Diagrama simplificado *Wrapper*, adaptada de [DAV95]

O método *Wrapper* é um método que ocorre fora do método de indução básica. Tipicamente pesquisa o mesmo espaço de subconjuntos de primitivas, mas avalia os conjuntos alternativos rodando um algoritmo sobre os dados de treinamento e utiliza como métrica o resultado do classificador. Na literatura de reconhecimento de padrões estatísticos são vistos há bastante tempo, porém em aprendizado de máquinas é relativamente recente. O objetivo de um método *Wrapper* é utilizar um método de indução que aplicado sobre um subconjunto de primitivas forneça um resultado melhor do que o conseguido com o conjunto completo.

Uma desvantagem que o método *Wrapper* possui em relação à abordagem *Filter* é o custo computacional, resultado da necessidade de revalidar todo o conjunto, considerando a remoção de uma parte das primitivas.

Existem várias técnicas utilizadas para contornar este problema. Em [OLI02b] é descrita uma técnica que considera a sensibilidade da rede em estimar o relacionamento da primitiva considerada e o desempenho da rede. Esta sensibilidade é definida na equação 2.3:

$$S_\beta = \frac{1}{N}\sum_{j=1}^{N} ASE(\bar{x}_\beta) - ASE(x_\beta) \qquad \textbf{(2.3)}$$

Sendo:

$$\bar{x}_{\beta} = \frac{1}{N}\sum_{j=1}^{N} x_{\beta_j} \qquad \textbf{(2.4)}$$

onde na equação 2.4 temos que x_{β_j} é a β –enésima variável de entrada do *j*-enésimo exemplo. S_β mede o efeito no erro de treinamento (ASE)[1] da substituição do valor β da entrada pela sua média (x_β). Em [MOO92] é visto que para as variáveis com sensitividade pequena, sua remoção não influencia a classificação final. Desta forma não é necessário modificar a topologia da rede neural utilizada e nem mesmo treiná-la novamente. Uma última consideração é feita a respeito da necessidade de um novo conjunto de validação, que deve ser diferente do conjunto de validação utilizado na fase inicial de treinamento da rede. Em [OLI02b] é utilizada uma segunda base de dados de validação para verificar o desempenho destes novos subconjuntos de primitivas. De acordo com [AHA94] a estratégia *Wrapper* é superior a *Filter* porque evita o problema de usar uma função de avaliação que tenha um *bias* diferente do classificador.

2.4.3 Algoritmo de busca Subida de Encosta com Mutação Randômica

Pode receber outras designações como: *greedy search*[2] ou *stepest ascent*[3]. Seu nome deriva da semelhança que possui com a situação de uma pessoa perdida dentro de uma floresta, à noite. Na tentativa de encontrar um ponto mais alto na floresta, para poder se localizar, a pessoa mesmo sem poder ver, pode aleatoriamente escolher uma direção e então dar um passo. Caso a ascensão tenha resultado positivo, este novo local passa a ser o seu melhor ponto e também ponto de partida para o próximo passo. Em muitas circunstâncias a técnica de Subida de Encosta tende a encontrar uma solução melhor, ou seja, reduzida em número de primitivas, porém em [RIC94] estão descritos três tipos de problemas:

a) Existem falsos pontos de máximo, que forçam o algoritmo a um grande retrocesso para alcançar o máximo verdadeiro.
b) Existem condições de igualdade de escolha, semelhantes a planaltos, ou seja, para qualquer novo subconjunto não há registro de melhora nos índices de classificação;

[1] Average Square Error
[2] busca compulsiva
[3] subida em etapas

c) Existe um tipo especial de máximo local chamado de cume. É uma região do espaço de medida mais interessante do que as áreas vizinhas, porém não pode ser atingido em um único movimento, por ser uma região de "difícil acesso";

Existe uma maneira de contornar cada um destes problemas, porém pode-se adotar pequenas alterações nas formas padronizadas de contorná-los. Em relação aos problemas citados respectivamente é possível considerar:

a) O máximo local é considerado como uma solução subótima. Seu valor é registrado para uso posterior. A partir daí adota-se o tratamento convencional retornando para uma camada anterior e tentando seguir em uma próxima direção interessante;
b) Mesmo quando não ocorre alteração do resultado continuar a avançar na mesma direção tentando desta forma sair da região de planalto. Desta forma é possível alcançar regiões mais promissoras, ou em último caso, com um número reduzido de primitivas;
c) Usar uma variação da técnica anterior, ou seja, avançar na mesma direção mesmo quando o critério avaliado tenha sofrido uma piora, desde que seja respeitado outro critério de parada, chamado de tolerância ao menor erro obtido para aquele máximo local que está sendo considerado.

Na Figura 2-11 é apresentado um exemplo de funcionamento do algoritmo. Inicia com todas as primitivas sendo consideradas para a classificação, representadas pelas esferas pretas. O algoritmo de mutação seleciona, de maneira aleatória, uma primitiva a ser desconsiderada e submete as demais a um novo treinamento. Após o treinamento do conjunto de treinamento, confirma-se a relevância da primitiva desconsiderada junto ao conjunto original de primitivas. Caso o resultado aponte para uma solução mais interessante salte para outra camada, desconsiderando aleatoriamente mais uma primitiva. Assim sucessivamente até atingir a melhor opção, determinada por alguma condição heurística ou de parada, como um número máximo de iterações, por exemplo.

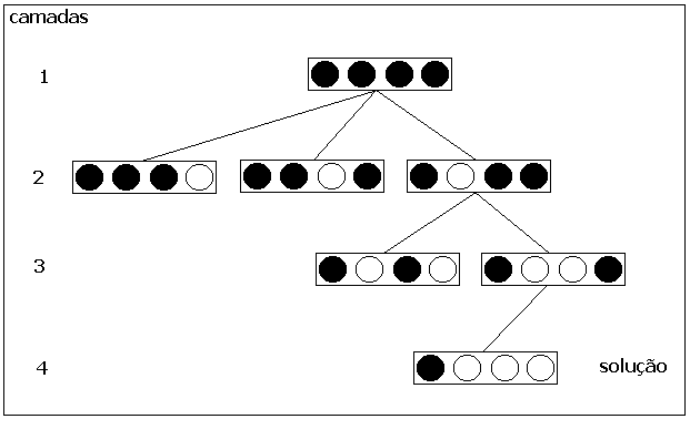

Figura 2.11: Simulação de algoritmo Subida de Encosta

Esta forma de tratamento em relação às primitivas é chamada de *backward,* pois a partir de um conjunto completo de primitivas o algoritmo vai aleatoriamente removendo primitivas. Uma variação desta técnica irá iniciar com apenas uma primitiva no conjunto e aleatoriamente irá incluir as outras primitivas até que o vetor esteja completo. Obviamente esta técnica conhecida como *forward* está sujeita aos critérios de parada estabelecidos dentro do algoritmo.

Em todos os experimentos serão apresentados resultados da forma padrão do algoritmo "Subida de Encosta com Mutação Randômica", que a partir d este ponto do trabalho será denominado de tradicional. A versão do algoritmo tradicional é apresentada a seguir com pequenas adaptações pertinentes ao método aqui proposto:

1. Início;
2. Carregar Classificador devidamente treinado;
3. Salvar condição do (Conjunto de primitivas atual) em (Conjunto de primitivas anterior);
4. Seleciona para remoção, aleatoriamente, uma primitiva que ainda não tenha sido removida;
5. Remove-a das bases e atualiza (Conjunto de primitivas atual);
6. Avalia Erro de Classificação sobre o conjunto Validação;
7. Se (erro atual <= erro anterior) confirma (Conjunto de primitivas atual);
8. Senão retorna (Conjunto de primitivas anterior) para (Conjunto de primitivas atual);
9. Se (erro atual <= erro mínimo) então Fim;
10. Se (número de Removidas < NR) então retorna ao passo 3;
11. Senão Retorna a primitiva já removida para o (Conjunto de primitivas Atual);
12. Salva condição (Conjunto de primitivas atual) como máximo local;
13. Se (número de máximos salvos == MAX_LOC) então Fim;
14. Retorna ao passo 3;
15. Fim;

MAX_LOC – número de máximos locais que o algoritmo armazena;
NR – número de primitivas removidas;

Figura 2.12: Algoritmo tradicional Subida de Encosta

Devido ao grande número de iterações possíveis, foi incluído um número máximo de resultados parciais que devem ser obtidos pelo algoritmo antes que seu funcionamento seja suspenso.

2.5 Considerações Finais

Os assuntos, abordados neste capítulo, revisam o conhecimento necessário para o entendimento das técnicas e métodos utilizados junto aos experimentos. Foram apresentados métodos de reconhecimento de padrões e técnicas de extração de primitivas que podem ser conceituados como "clássicos", e fazem parte dos objetivos principais deste trabalho ao garantir confiabilidade aos resultados obtidos em nossos experimentos, permitindo que sejam usados como base para o desenvolvimento de técnicas híbridas mais eficientes no quesito classificação de padrões.

Capítulo

3 Trabalhos Relacionados

Neste capítulo é apresentada uma seleção de publicações que estão diretamente relacionadas com os objetivos e métodos apresentados neste trabalho. Em alguns casos serviram como justificativa para a escolha de determinada técnica.

Em [LIU02] é apresentado um estudo comparativo de diversos modelos de classificadores:

Tabela 3.1: Abreviaturas dos classificadores utilizados

Sigla	Classificador	Sigla	Classificador
MLP	Multilayer Perceptron	MQDF3	Modified Quadratic Discriminant Function
RBF	Radial Basis Function	QDF	Quadratic Discriminant Function
PC	Polynomial Classifier	LDF	Linear Discriminant Function
LVQ	Linear Vector Quantization	SLNN	Single-layer Neural Network

A Figura 3-1 apresenta os resultados comparativos destes classificadores considerando a quantidade de exemplares da base de treinamento. Os resultados podem ser comparados também aos obtidos por classificadores Vizinho mais Próximo (1-NN) e Análise Discriminante Regularizada (RDA). Os diversos experimentos utilizam exemplares de dígitos da base NIST que é a mesma utilizada neste trabalho.

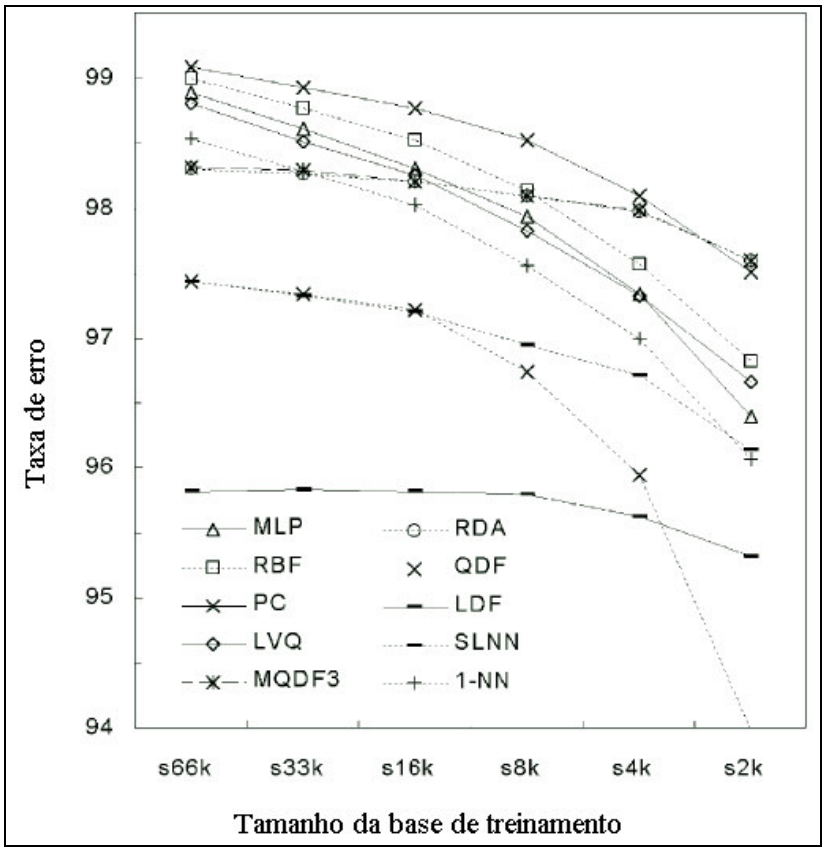

Figura 3.1: Reconhecimento por tamanho da base, adaptada de [LIU02, pág. 198]

Pela Figura 3-1 é possível notar que para conjuntos de treinamento com 66.000 exemplares da base NIST, as taxas de reconhecimento atingidas por alguns classificadores, entre eles PC, MLP, RBF e LVQ (ver tabela na pág. anterior), ficam próximas em torno de 99%.

Considerando que a base de treinamento utilizada nos experimentos é ainda maior, possuindo 195.000 exemplares, em princípio, poderia ter sido escolhido qualquer um destes quatro classificadores. A escolha recaiu sobre a rede neural MLP, devido a considerações de facilidade de implementação e modificações de topologia, necessárias aos diversos experimentos efetuados e também para permitir a comparação de resultados obtidos em relação a outras publicações. O método aqui proposto possui o mesmo protocolo experimental utilizado em [OLI01b] onde é apresentado um sistema de seleção de primitivas, baseado em algoritmos genéticos com o objetivo de gerar novos subconjuntos de primitivas e utiliza rede neural MLP como classificador. As primitivas não são efetivamente removidas, sendo substituídas por um valor médio

conforme utilizado em [MOO91]. Esta técnica é também utilizada aqui, neste trabalho, e a formulação necessária consta do Capítulo 2 Fundamentação Teórica, sub-item Métodos *Wrapper*. A semelhança do protocolo experimental, entre os dois trabalhos, permite uma comparação dos resultados obtidos. Em [OLI01a] são experimentados dois métodos com algoritmos genéticos para a criação dos subconjuntos de primitivas a serem avaliados e redes neurais para a classificação. Os métodos são AGS, Algoritmo Genético Simples[1] SGA e AGI, Algoritmo Genético Iterativo[2]. Com o método AGS o conjunto de primitivas foi reduzido de 132 primitivas para 95 primitivas com uma taxa final de reconhecimento em 97,52% e com o método AGI o conjunto de primitivas foi reduzido para 104 para uma taxa final de reconhecimento de 97,42%.

Em [OLI02c] é apresentado experimento similar, utilizando também uma técnica de algoritmo genético chamada Algoritmo Genético de Ordenação não Dominante[3] NSGA e que apresentou resultados melhores principalmente em relação às taxas de erro apresentadas. Com o método NSGA o conjunto final foi reduzido para 100 primitivas e a taxa final de reconhecimento ficou em 99,16%. Estes experimentos foram conduzidos com a utilização de um cluster contendo 16 computadores escravos e um equipamento mestre, todos apresentando as seguintes características CPU 1.1 Ghz e 512 MB RAM. Para o método aqui proposto, foi utilizado apenas um computador bi-processador. A expectativa inicial de redução de primitiva que era de 30%, conforme os resultados obtidos por [OLI01a, OLI02b], foi superada atingindo mais de 40% de primitivas removidas conforme pode ser verificado no capítulo de experimentos.

Em [BLU97] é apresentado um estudo sobre seleção de exemplos e primitivas e também uma análise sobre os conceitos de relevância de uma primitiva em relação ao conjunto completo. Na página 5 é apresentado o seguinte conceito:

> "...Vamos agora discutir algoritmos de seleção de primitivas e, de forma mais abrangente, algoritmos que trabalhem com conjuntos de dados que contenham número elevado de primitivas irrelevantes. Um paradigma conveniente para muitas destas abordagens (especialmente as que executam seleção de característica de forma direta) é a busca heurística, com cada estado no espaço de soluções correspondendo a um subconjunto de possíveis primitivas..."

[1] Do original inglês: SGA Simple Genetic Algorithm
[2] Do original inglês: IGA Iterative Genetic Algorithm
[3] Do original inglês: NSGA Non dominated Sorting Genetic Algorithm

Este conceito auxilia na justificativa da escolha dos métodos de busca utilizados neste trabalho: seleção de primitivas com método de busca heurística Subida de Encosta.

Uma análise de abordagens *Wrapper* na seleção de primitivas é apresentada em [KOH97]. São utilizadas bases de dados reais e geradas artificialmente. De uma maneira geral a utilização de métodos de seleção de primitivas não consegue apresentar melhoras no índice de classificação das bases de dados reais, mas melhora o desempenho das bases artificiais, o aumento da taxa de classificação é bastante significativa, da ordem de 20 a 25% com a utilização de método de seleção de subconjuntos de primitivas.

A Tabela 3-2 [KOH97, pág. 14] compara a performance do classificador ID3[1], que é baseado em árvores de decisão, sem e com a utilização de um algoritmo de seleção de primitivas ID3-FSS[2]:

Tabela 3.2: Comparação de resultados sobre diferentes bases de dados

Bases de dados	ID3	ID3-FSS
Bases reais		
1 breast cancer	94.57	94.71
2 cleve	72.35	78.24
3 crx	81.16	85.65
4 DNA	90.64	94.27
5 horse-colic	81.52	83.15
6 Pima	68.73	69.52
7 sick-euthyroid	96.68	97.06
8 soybean-large	90.62	90.77
Bases geradas artificialmente		
9 Corral	100.00	75.00
10 m-of-n-3-7-10	91.60	77.34
11 Monk1	82.41	75.00
12 Monk2-Local	82.41	67.13
13 Monk2	69.68	67.13
14 Monk3	90.28	97.22

[1] Induction Decision Tree . Utilização do homófono 3 <> tree;
[2] Feature Subset Selection

Os nomes das bases de dados foram mantidos no idioma original. As bases de dados de números 1 até 8 são bases reais, enquanto as bases de 9 até 14 são bases construídas artificialmente. De maneira geral o que se conclui é de que a seleção de primitivas pode, dependendo da base, melhorar as taxas de classificação ou obter os mesmos índices porém com um número reduzido de primitivas.

Em [RAM03, pág. 08] são avaliadas técnicas de aprendizagem utilizando seleção de primitivas e de exemplos. O artigo afirma que a utilização de seleção randômica da primitiva a ser removida evita um dos problemas básicos apresentados pelo algoritmo de busca de abordagem *Wrapper*, tipo *Hill Climbing,* que é o fato do algoritmo acabar convergindo para um máximo local. Desta forma, uma vez que o método de busca *Hill Climbing* tenha sido escolhido para o trabalho aqui apresentado, a utilização da função randômica de escolha de primitivas a ser eliminada é necessária principalmente para garantir a geração de subconjuntos, ou de máximos locais, de melhor qualidade.

Em [SKA94] é apresentado um estudo comparativo entre o classificador convencional 1-NN e implementações com métodos heurísticos de Monte Carlo MC e Random Mutation Hill Climbing RMHC. A Tabela 3-3 apresenta os resultados de classificação de algumas bases de dados, considerando também o percentual de memória requerida para o método com RMHC em relação ao método tradicional 1-NN.

Tabela 3.3: Comparação de resultados entre algoritmo RMHC e 1-NN

Base de Dados	Armazenamento	RMHC-P	1-NN
Iris	2,0%	93,3%	93,3%
Cleveland	1,0%	82,3%	74,3%
Breast Cancer	1,0%	70,9%	65,6%
Soybean	8,5%	97,8%	100%

Os experimentos mostram que o classificador com RMHC possui desempenho comparável ao método tradicional 1-NN em termos de classificação, porém utiliza apenas 2% da memória requerida e executa as etapas de treinamento de 10 até 200 vezes mais rápido.

Em [BOZ02] é conduzida uma série de experimentos que incluem algoritmos *Wrapper* com técnicas de seleção de primitivas *backward* e *forward,* ou seja, a busca começa com o vetor de primitivas completo ou começa com o vetor de primitivas vazio.

As alternativas são chamadas de SBE *Sequential Backward Elimination* e SFS *Sequential Forward Selection.* Na Tabela 3-4 são apresentados os resultados de classificação de algumas bases de dados reais, semelhantes às bases utilizadas aqui neste trabalho, porém com um número de exemplares bem inferior. Sendo uma base composta por 435 exemplares de uma base de votos com 16 primitivas e apenas 2 classes e a outra de doenças de grãos de soja com 683 exemplares 35 primitivas e 19 classes. São apresentadas duas opções de algoritmo: uma com critério de parada (ES – *early stop*) e outra de busca exaustiva percorrendo todo o espaço de soluções (TA- *test all*):

Tabela 3.4: Resultados dos experimentos com técnica backward e forward

	BASE	**BASE**
ALGORITMO	vote	soybean
SFS-ES	95,66%	89,03%
SFS-TA	95,20%	89,48%
SBS-ES	95,88%	88,43%
SBS-TA	94,97%	88,70%

São duas bases de dados reais onde é possível verificar a semelhança entre os valores obtidos entre as técnicas que utilizam a busca exaustiva e com critério de parada. Também não são verificadas diferenças de desempenho entre as técnicas de *forward* e *backward*.

Em [RAM03] é proposta uma estrutura de trabalho que utiliza um algoritmo de filtragem seqüencial denominado SCRAP e também um método de seleção de exemplos denominado LASER. Enquanto a maioria dos algoritmos estáveis de aprendizado funciona bem com informações relevantes, eles degradam na presença de informação redundante ou irrelevante. O Aprendizado Seletivo ou Focado é uma solução para este problema. Os dois componentes de aprendizado seletivo são: observação seletiva (seleção de primitivas) e utilização seletiva (seleção de exemplos):

a) Filtro de Pesquisa Seqüencial - chamado de Seleção de SubConjunto que é a abordagem de Caso Relevante;
b) Algoritmo de Aprendizado – usando Estrutura em Anel de Pesquisa para efetuar seleção de exemplos para Algoritmos de Pesquisa.

A aplicação de ambos os esquemas a um classificador Naive Bayes resultou em melhor exatidão de prognóstico; Melhoras da ordem de 3 % foram encontradas. Vendo os algoritmos de forma um pouco mais detalhada:

c) Algoritmo de Seleção de Primitivas SCRAP – trata-se de um filtro de pesquisa seqüencial, que constrói clusters. Toda a vizinhança é considerada um nó para avaliação de primitivas. Cada vizinhança é identificada unicamente por dois pontos de mudança de classe. O primeiro ponto é onde a construção começa e o segundo é onde termina;
d) Algoritmo LASER – um método de seleção que consiste de duas partes principais: Esquema de seleção de exemplo e Classificador (*Learner*) de alvo.

Considerações Finais

A análise dos resultados obtidos nas publicações, citadas acima, direcionaram a montagem de uma estrutura de trabalho robusta, em condições de permitir a execução dos diversos experimentos propostos e implementados neste trabalho. A seguir é apresentado um resumo das conclusões obtidas a partir da leitura destas publicações:

a) A utilização de métodos de busca heurística na seleção de primitivas apresenta resultados melhores do que sistemas que não utilizam;
b) Métodos de busca com abordagem *Wrapper* utilizam uma função de avaliação sintonizada com o classificador;
c) A rede neural pode ser um classificador tão eficiente quanto outras técnicas mais modernas, desde que a base de dados possua um número de exemplares elevado[1];
d) O método de busca Subida de Encosta apresenta desempenho semelhante ao busca de Melhor Escolha para bases de dados reais;
e) A remoção aleatória de primitivas, aplicada ao método de busca Subida de Encosta, assegura melhores resultados do que a remoção sequencial;
f) Em bases de dados reais não se verificam diferenças de desempenho entre as técnicas de busca *forward* e *backward;*

[1] Uma relação aproximada seria um mínimo de 2.000 exemplares por classe.

Capítulo

4 Método Proposto

O objetivo central do presente trabalho é apresentar um algoritmo de busca eficiente na remoção de primitivas irrelevantes e redundantes, porém com baixo custo computacional, que permita viabilizar a sua aplicação em bases de dados com número elevado de primitivas. Para a remoção das primitivas, são considerados dois fatores: inicialmente uma seleção aleatória e então sobre a primitiva escolhida é observado o conceito de relevância ou a influência individual sobre a taxa de reconhecimento final. Para a realização dos experimentos, são necessárias etapas preliminares e para facilitar o entendimento do método proposto é apresentado a seguir um diagrama dos principais blocos da estrutura de trabalho utilizada:

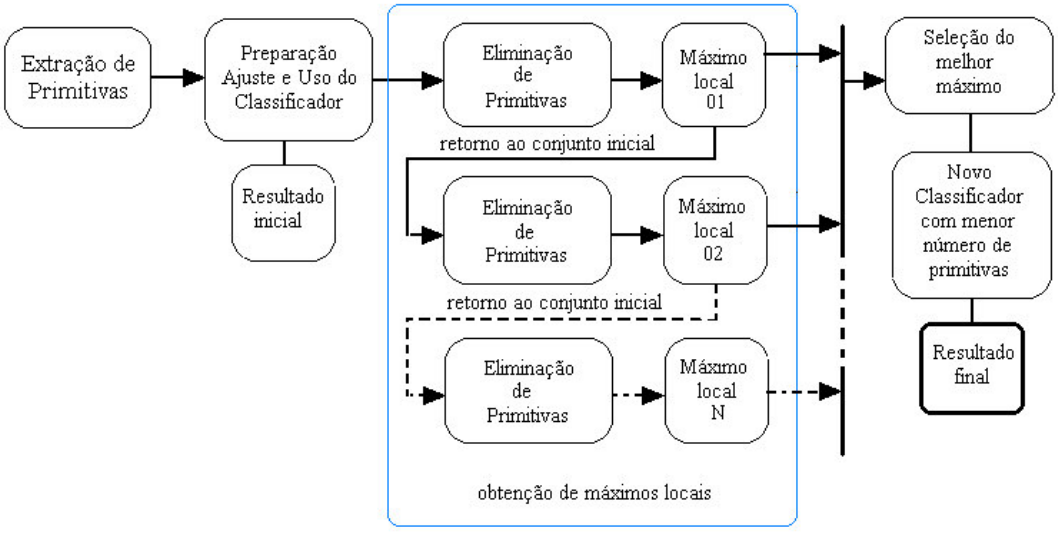

Figura 4.1: Estrutura básica inicial

4.1 Estrutura Básica

Consiste na montagem das etapas inicias, que permitem efetuar a classificação das bases de dados. É dividida em três tarefas, apresentadas na Figura 4-2:

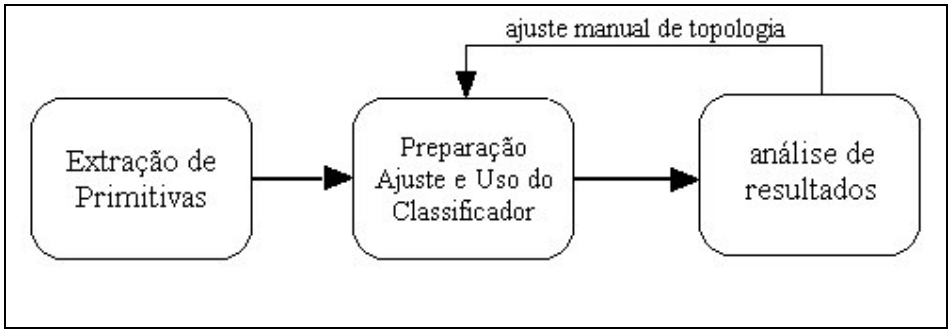

Figura 4.2: Ajustes iniciais do classificador

O objetivo principal desta etapa é adaptar os arquivos de bases de dados às características do classificador utilizado. Permite obter um classificador ajustado e pronto para a tarefa de reconhecimento considerando o conjunto completo de primitivas.

4.1.1 Extração de Primitivas

A partir da utilização de imagens em formato TIFF[1] compactado, são extraídas informações de cada um dos exemplares da base. Considerando o zoneamento aplicado na imagem, é montado um vetor, com um total de 6 x 8 primitivas contendo informações da orientação do contorno da imagem do caractere, 6 x 13 primitivas com informações de fundo e 6 informações de contagem de *pixels*, que completam o conjunto principal totalizando 132 primitivas. Os valores obtidos para os exemplares foram devidamente divididos em conjuntos de vetores de primitivas para sua utilização junto ao classificador.

[1] Especificação de um formato de arquivo gráfico, criado pela Adobe Systems.

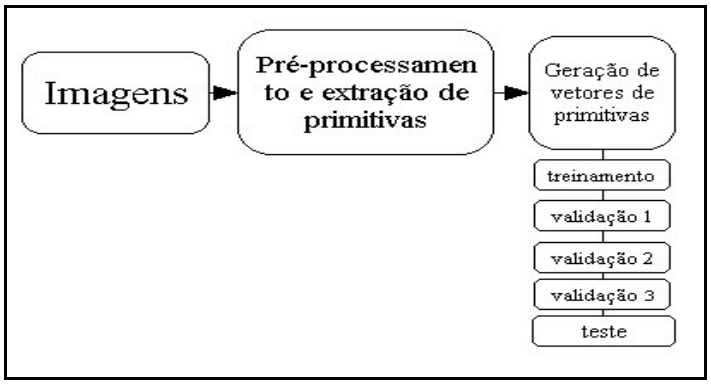

Figura 4.3: Divisão das bases de dados

O processo de extração de primitivas é efetuado apenas uma vez. Os arquivos de vetores são reutilizados, uma vez que a redução do número de primitivas não altera o método de extração. Em relação à função de cada um dos conjuntos, estes podem ser descritos como:

a) Treinamento, com aproximadamente 70 % dos exemplares da base. Usado para treinamento da rede neural, incluindo todos os passos necessários para o ajuste da rede;

b) Validação 1, com aproximadamente 5% dos exemplares da base. Usado na fase de treinamento para verificação do comportamento da rede treinada. Pode ser considerado um conjunto de validação convencional e a taxa de reconhecimento é o critério de parada adotado para o treinamento inicial da rede neural. Devido ao elevado número de exemplares em relação ao número de classes das bases de dados, não será utilizado um processo de validação cruzada;

c) Validação 2, com aproximadamente 5% dos exemplares da base. Normalmente não é utilizado pelos métodos tradicionais de classificação através de redes neurais. Porém, aqui tem grande importância. É utilizado junto ao algoritmo de seleção e remoção de primitivas para avaliar o novo subconjunto de primitivas gerado. A rede neural é submetida a uma nova etapa de validação, sendo que para verificar seu comportamento é utilizado o conjunto de Validação 2. As taxas de erro de classificação obtidas sobre este conjunto servirão de controle do algoritmo de busca. O algoritmo gera uma série de resultados e conseqüentemente de taxas de erro, que serão armazenadas e comparadas umas com as outras, auxiliando o algoritmo a

decidir qual é seu próximo passo. Os máximos locais são determinados dependendo do comportamento da rede sobre este conjunto de validação;

d) Validação 3, com aproximadamente 5% dos exemplares da base. Utilizado em uma fase adicional de validação, após terem sido determinados os máximos locais. Mais uma vez a rede neural será submetida a uma etapa de validação, tendo a função de identificar o melhor dos máximos locais obtidos. Foram conduzidos também alguns experimentos sem a utilização deste conjunto de validação;
e) Testes, com aproximadamente 15% dos exemplares da base. Utilizado para efetivamente verificar a nova taxa de reconhecimento com a utilização de um número menor de primitivas. Com a rede neural devidamente ajustada, já com número reduzido de primitivas, ocorre o treinamento deste conjunto.

4.1.2 Classificador

A Figura 4-4 representa as etapas de treinamento e validação iniciais, ou seja, a rede neural configurada está sendo treinada e validada através de sua taxa efetiva de reconhecimento, considerando os conjuntos de Treinamento e Validação 1. Eventuais ajustes devem ser feitos nesta fase, o que pressupõe uma intervenção manual na topologia e parâmetros. O ajuste do número de nós da camada escondida é feito manualmente, bem como o ajuste dos pesos e das constantes de aprendizagem da rede neural:

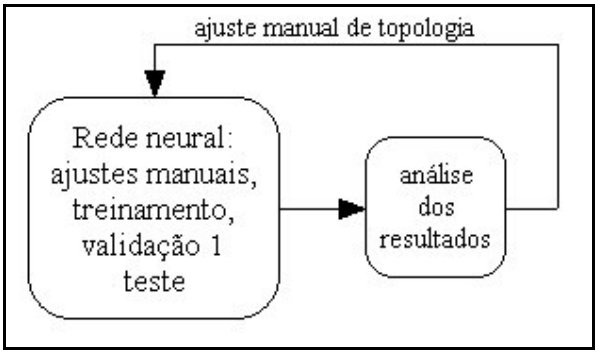

Figura 4.4: Ajuste manual da rede neural

A rede neural artificial utilizada é uma MLP com algoritmo Back-Propagation, cuja topologia é a seguinte:

a) Número de nós da camada de entrada: deve ser igual ao número de primitivas utilizadas que está estimado em 132 primitivas, considerando o vetor inicial de primitivas;
b) Número de nós da camada de saída. Deve-se considerar cada experimento efetuado utiliza um diferente número de classes de saída: 10 classes para dígitos, 26 classes para caracteres maiúsculos e 26 classes para caracteres minúsculos, que representam diretamente o número de nós na camada de saída;
c) Número de Camadas Escondidas: seleciona-se apenas uma camada escondida, visto que esta configuração de rede é conhecida como um classificador universal, o que atende a nossa necessidade de separação em classes. Não é nosso objetivo principal avaliar diferentes topologias que convirjam mais rapidamente do que outras;
d) Número de nós na Camada Escondida: devido à inexistência de axiomas, utiliza-se uma regra empírica conhecida como Regra de Baum-Haussler, como ponto de partida para o ajuste deste item da topologia.

4.1.3 Épocas de Treinamento

O comportamento da taxa de erro de validação em relação à taxa de erro de treinamento determina a parada do treinamento do classificador. A partir do arquivo de resultados gerado pelo SNNS[1] é obtido, através de uma conversão, o valor da taxa de reconhecimento efetiva, em lugar do erro MSE[2], Esta taxa de erro é utilizada no algoritmo de busca para tomar a decisão de avançar ou não na tarefa de remoção de mais primitivas.

4.2 Seleção de Primitivas

O método de busca utilizado apresenta otimizações em relação ao método tradicional Subida de Encosta (*Hill Climbing*), de abordagem *Wrapper*. Tal método consiste em avaliar uma melhora em determinada função heurística, que corresponde a uma menor taxa de erro de classificação. O objetivo da função é atingir o menor nível de erro possível. O algoritmo trabalha com subconjuntos de primitivas que são

[1] SNNS – Stuttgart Neural Network Simulator. Aplicativo para construção e treinamento de redes neurais
[2] Do inglês: *Mean Square Error.*= Erro médio Quadrático.

determinados de forma aleatória. Forma de criação de subconjuntos conhecida como Mutação Randômica (*Random Mutation*).

4.2.1 Múltiplos máximos locais

Máximos locais são considerados candidatos a melhor solução possível como o conjunto de primitivas proposto. Permitem avaliar e comparar a eficiência de diversos pontos do espaço de soluções, fugindo de uma possibilidade de especialização da rede neural em relação ao conjunto de Validação 2. Todos os máximos obtidos são utilizados em novo processo de validação, onde é utilizado o conjunto de Validação 3 para identificar qual o melhor deles. Também é experimentada outra maneira de determinar qual o melhor subconjunto obtido. Opção que não utiliza esta segunda validação, pois o melhor máximo é escolhido por possuir o menor número de primitivas.

4.2.2 Análise da Sensitividade

A eliminação de primitivas de um conjunto de validação permite avaliar a taxa de classificação global sem a influência da primitiva removida, porém exige um árduo trabalho de re-treinamento do classificador. Em conjuntos com grande número de primitivas e de exemplos tal operação torna-se proibitiva. Os vetores de primitivas teriam que ser recriados com um diferente número de primitivas para cada nova iteração do algoritmo.

Em lugar de remover determinadas primitivas do vetor de primitivas e recriar o vetor com um tamanho diferente do inicial, a primitiva escolhida apenas tem o seu valor substituído pelo seu valor médio, técnica chamada de Análise da Sensitividade[1] [OLI02]. Valor calculado de forma direta pela média de valores apresentados por determinada primitiva, considerando todos os exemplares presentes no conjunto de Validação 2. Os valores médios das primitivas são calculados e armazenados em um arquivo, logo no início do processo, sendo consultado durante o funcionamento do algoritmo.

A "sensitividade" das primitivas elimina a dificu ldade que representa recriar os vetores de primitivas. Ao tornar determinada primitiva irrelevante junto ao processo de classificação, seu valor é substituído em cada um dos exemplos por um valor obtido a

[1] Sensitivity Analysis

partir de informações de todos os valores da primitiva ao longo do conjunto de Validação 2.

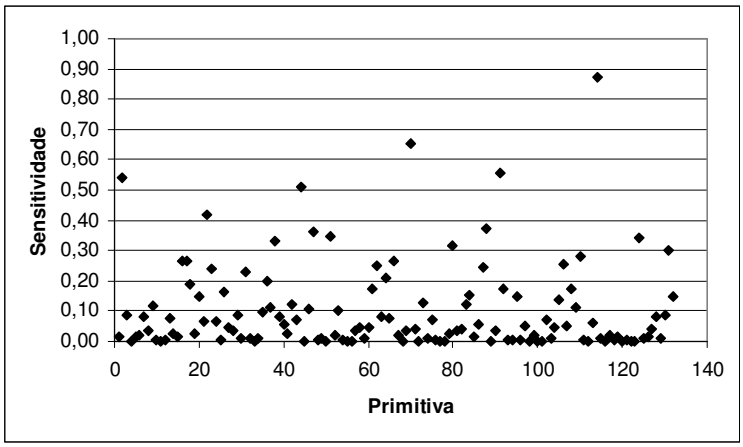

Figura 4.5: Valor médio por primitiva

Na Figura 4-5, é apresentada uma visualização dos valores médios obtidos, considerando a base utilizada como sendo a de Validação 2. Um único vetor é gerado com o valor médio de cada uma das primitivas. A cada novo subconjunto gerado pelo algoritmo o vetor de valores médios é consultado. Os valores fornecidos substituem as primitivas removidas em cada um dos exemplos do conjunto de Validação 2.

4.2.3 Critérios de Parada Considerados

Alguns critérios de parada considerados para o algoritmo:

a) Obtenção de taxa de erro de classificação igual a zero. É uma condição interessante, principalmente se estiver associada a uma grande redução do número de primitivas;

b) Impossibilidade de melhorar a função critério. Após efetuar todas as remoções de primitivas possíveis sem que ocorra uma melhora nos índices da função critério, o algoritmo está preparado para executar um retrocesso à condição do conjunto de primitivas inicial, de maneira a poder reiniciar o processo e explorar regiões diferentes em busca de melhores resultados de classificação.

c) Obtenção do número de máximos locais possível, com as técnicas de otimização utilizadas e que serão explicadas ao longo deste capítulo,

permitindo então que o método progrida para a fase de escolha do melhor subconjunto obtido.

4.2.4 Validando máximos locais

Nesta etapa os diversos subconjuntos selecionados, que durante o funcionamento do algoritmo de seleção de primitivas foram considerados máximos locais ou de outro tipo relevante, irão ser testados com a base de Validação 3 para verificar qual atinge a melhor taxa de classificação. Conforme Figura 4-6, serão executados novamente os passos definidos na 2ª etapa, inclusive os ajustes manuais necessários à nova topologia da rede neural:

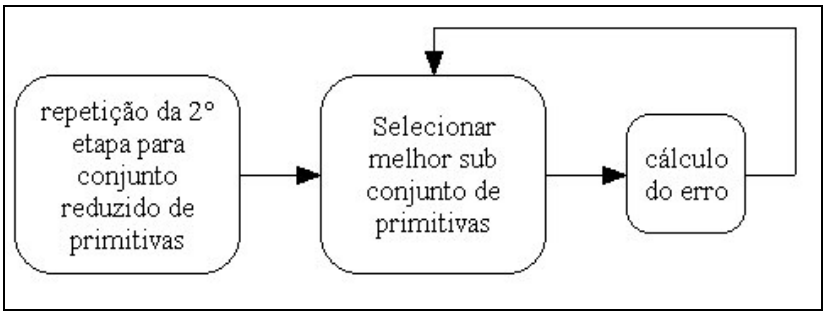

Figura 4.6: Validação dos máximos locais

Especial atenção é reservada a esta etapa de validação. Do conjunto de máximos locais obtidos é que sai o subconjunto de primitivas a ser utilizado na preparação das novas bases de treinamento e teste.

Conforme explicado anteriormente, a forma de busca dos máximos locais baseia-se na taxa de classificação do conjunto de Validação 2 que é utilizado junto ao algoritmo de busca. Este tipo de seleção pode permitir certa especialização em relação ao conjunto utilizado. A etapa de validação de máximos locais consiste na utilização de uma terceira base de Validação 3 com o objetivo de assegurar a escolha do máximo local com um bom nível de generalização.

4.2.5 Etapa de análise de resultados

Esta é a última etapa necessária no método proposto. É onde se comparam os resultados de classificação obtidos com o conjunto completo de primitivas aos

resultados obtidos com o conjunto reduzido de primitivas obtido, dentro das etapas anteriores.

4.3 Otimizações no processo de Seleção de Primitivas

Após encontrar um máximo local, o algoritmo convencional perde muito de sua eficiência ao ficar exaustivamente testando regiões muito próximas ao máximo local encontrado. Considerando que uma determinada primitiva, relevante ao processo de classificação, tenha sido removida antecipadamente do conjunto, é possível que o máximo local encontrado, em termos de resultado seja inferior ao resultado máximo possível para o problema. Considerando o número de combinações possíveis para um conjunto de 1 a 132 primitivas, e o fato de que caso a busca reinicie removendo novamente uma primitiva "relevante" ao processo de classif icação, o algoritmo ficaria perdendo tempo procurando soluções onde é impossível encontrar. Seria o caso de começar todo o processo do início e esperar que primitivas relevantes não fossem aleatoriamente removidas do conjunto.

4.3.1 Prioridade de remoção

Uma análise sobre os resultados obtidos com o algoritmo inicial permitiu observar que primitivas removidas e que provocam um aumento nos índices de erro são mantidas no conjunto após o final da busca, ou seja, os subconjuntos sem elas não se tornam máximos locais. Uma maneira mais rápida e eficiente do algoritmo convergir para uma região de máximo local é utilizar este conhecimento para priorizar a remoção de determinadas primitivas do conjunto e dificultar a remoção de outras.

Remover uma primitiva, em termos de espaço de primitivas, pode ser visto como a variação experimentada em uma região local pela ausência de uma primitiva. Se o algoritmo de busca Subida de Encosta faz paralelo com a subida de uma montanha no escuro, pode-se dizer que o erro obtido pela falta de uma primitiva dá a idéia de um degrau. Sua posição na escada ou escalada não é conhecida, porém é possível afirmar se sua condição é de subida, descida ou neutra:

a) Neutra – indica que não existem alterações significativas com a sua remoção. É forte candidato a ser removido uma vez que sua falta não é sentida;

b) Descida – o objetivo do algoritmo é subir a encosta. Uma condição de descida significa piorar a função objetivo. Não deve ser removida, pois sua remoção aumenta o erro;
c) Subida – sua remoção diminui a taxa de erro, deve também ser removida.

A maneira escolhida para acelerar o encontro de um máximo local é atribuir uma ordem de prioridade para a remoção de primitivas. Não se trata de uma simples ordenação baseada na relevância das primitivas. Fica mantido o conceito de remoção aleatória de primitivas, porém as primitivas escolhidas serão avaliadas em relação a sua relevância junto ao processo de classificação e removidas apenas após terem sido escolhidas aleatoriamente um determinado número de vezes. Outras formas de utilização da relevância na condução de algoritmos de busca também podem ser encontradas em [MOL02, FRA00, BOZ02]. Antes de iniciar a operação efetiva do algoritmo é determinada a taxa de erro individual, ver Figura 4-7, que ocorre no conjunto de Validação 2.

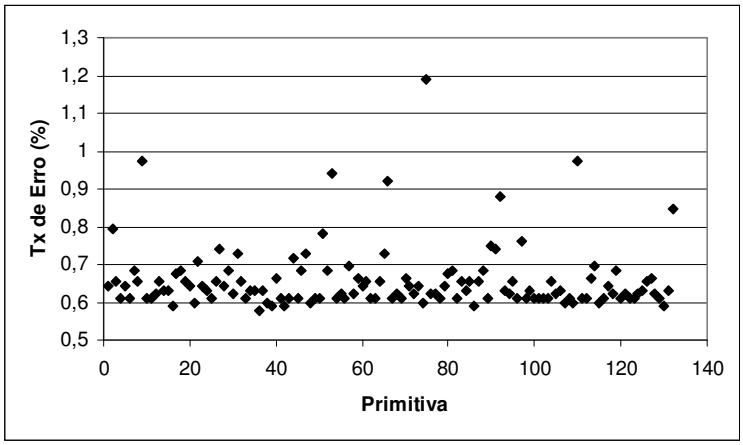

Figura 4.7: Taxa de erro resultante da remoção individual

Esta taxa é obtida pela remoção individual de cada uma das primitivas do vetor. Obviamente fazendo a substituição de seu valor pelo valor médio, obtido quando da determinação da sensitividade das primitivas. Para estabelecer a prioridade é necessário que os diversos valores obtidos passem por uma quantização, porém o número de níveis a ser utilizado nesta quantização deve ser determinado de forma experimental.

4.3.2 Seleção do Número de níveis de prioridade

O número de níveis de prioridade a ser adotado precisa ser determinado e uma variável foi estabelecida com esta finalidade, denominada NNIVEIS, e que indica o número de níveis de quantização em que se deve classificar as primitivas pertencentes ao vetor completo. O melhor valor NNIVEIS deve ser aquele que permite ao algoritmo eliminar o maior número de primitivas no menor número de iterações. Foram utilizados experimentalmente os valores 0, 5, 10, 20 e 50, conforme a Figura 4-9, onde 0 corresponde ao algoritmo inicial sem priorização. Quanto maior o valor de NNIVEIS mais criterioso será o algoritmo na remoção de primitivas. Caso o nível fosse igual ao número de primitivas do vetor inicial, o algoritmo poderia escolher uma única primitiva em cada iteração e estaria perdido o conceito de remoção aleatória de primitivas. Porém efetuaria uma remoção seqüencial, baseada na relevância inicial de cada uma das primitivas junto ao processo de classificação.

Deve-se lembrar que esta condição de prioridade é obtida considerando o início do algoritmo e tal condição pode variar conforme forem reduzindo o número de primitivas. A implementação desta técnica junto ao algoritmo ocorre da seguinte forma:

1. Calcular a taxa de reconhecimento com cada uma das primitivas removidas (ERRO IND);
2. Entre todos os valores identificar a maior e a menor taxa de reconhecimento (MAX e MIN);
3. Escolher um número inteiro para (NNIVEIS);
4. Calcular passo entre os níveis de prioridade (PASSO= (MAX-MIN)/ NNIVEIS);
5. Criar um vetor de prioridades para as primitivas sendo os valores determinados por VETOR [primitivas]= (- NNIVEIS + (MAX – ERRO IND)/ PASSO).

Figura 4.8: Determinação da prioridade da remoção das primitivas

De uma forma geral a prioridade *P* é função do erro obtido quando da remoção da primitiva, onde e_p é o erro obtido com a remoção de cada uma das primitivas e e_T é o erro obtido com o conjunto completo de primitivas.:

$$P = \{0 \quad até \quad (-NNIVEIS) \quad se\,(e_p \geq e_T)$$

Cada primitiva passa a ter uma prioridade para a sua remoção que irá variar entre os valores (0) como prioridade máxima para remoção e (- NNIVEIS) como prioridade mínima para remoção. Com exceção do caso da prioridade máxima os outros valores são negativos.

A convergência para uma região de máximo local é dependente da seqüência de remoção de primitivas escolhida. Pela Figura 4-9 é possível verificar que para NNIVEIS igual a 10, ou seja, dez níveis de prioridade, o funcionamento do algoritmo apresenta uma rápida convergência para uma região de máximo local. Após aproximadamente 120 iterações o algoritmo já havia removido 40 primitivas do conjunto inicial.

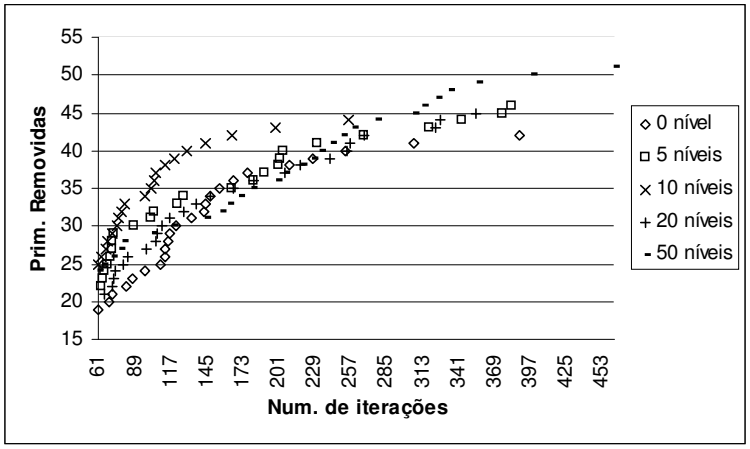

Figura 4.9: Seleção do número de níveis de prioridade

Desta forma inicia-se a operação do algoritmo de busca considerando como -10 a prioridade mínima para remoção e 0 como prioridade máxima. A seleção aleatória da primitiva, ou *Random Mutation,* a ser removida é confrontada com a informação da prioridade de remoção. Caso seja igual a 0 (zero) a primitiva é removida do conjunto, caso seja menor do que 0 (valores entre -1 e -NNIVEIS) é adicionado (1) ao seu valor. Quando a prioridade de determinada primitiva atingir o valor zero ela é removida. Tal arranjo permite ao algoritmo convergir rapidamente para uma região de máximo local, eliminando inicialmente apenas primitivas que pouco alteram a taxa de erro, e quando uma região próxima a um máximo local for alcançada o algoritmo irá experimentar todas as opções de remoção de primitivas que ainda estiverem disponíveis. Isto atende a duas necessidades, converge rapidamente evitando perder tempo removendo primitivas relevantes ao reconhecimento e quando próximo a um máximo local, permite que todas as primitivas que restaram no conjunto tenham a chance de serem removidas mantendo as características do método proposto.

Uma vantagem em relação a outras técnicas é que neste caso a relevância das primitivas é obtida dentro do próprio método. Não se recorre a classificadores externos. A "relevância" está baseada na variação do erro de classificação que a remoção de uma

primitiva provoca sobre o resultado final. Como último comentário deve-se frisar que uso desta técnica tem pouco impacto sobre o tempo total de funcionamento do algoritmo, pois os critérios de parada dependem de outros fatores e não apenas de uma rápida convergência.

4.3.3 Seleção da primeira primitiva a ser removida

Após encontrar um máximo local, retornar a condição inicial, com o conjunto completo de primitivas e obrigatoriamente selecionar uma primitiva inicial ainda não utilizada como primeira primitiva removida. Esta opção tira o caráter aleatório da escolha da primeira remoção. Utilizando um vetor com 132 primitivas, o espaço de soluções possui um incontável número de soluções possíveis. Caso fosse permitido que uma semente já utilizada fosse novamente removida, o algoritmo poderia estar retornando a mesma região de máximo local de que acabou de sair e testar praticamente sempre a mesma região. Entretanto, ao exigir uma nova primitiva, não removida por primeira antes, abre-se a possibilidade de pesquisar outras áreas do espaço de soluções, permitindo encontrar diferentes regiões do espaço de soluções mais promissoras, que é o objetivo desta etapa do método. O algoritmo apresentado mantém um controle especial sobre a primeira primitiva removida, que a partir daqui passa a se chamar "semente". O número de sementes é no máximo igual ao número de primiti vas do conjunto inicial, neste caso 132. Cada primitiva terá condições de ser a primeira a ser removida. Para cada uma delas o algoritmo irá tentar remover o maior número possível de outras primitivas, até encontrar um máximo local.

Em princípio, a utilização de sementes determinaria o número de máximos locais possíveis, que em nosso caso seria igual em 132 máximos, porém deve-se considerar o item anterior, Prioridade na Remoção de Primitivas, onde é possível remover apenas primitivas que melhorem ou pelo menos mantenham o nível de reconhecimento obtido pelo subconjunto anterior. O algoritmo proposto após remover as primitivas com menor impacto sobre a taxa de erro, efetua também a remoção das demais primitivas, porém o resultado obtido nestes casos é pior do que o resultado obtido com o conjunto completo. O algoritmo não prossegue e a primitiva é colocada na condição de já utilizada como semente. O subconjunto não é considerado, pelo fato de não ter sido efetivamente removida nenhuma primitiva. O algoritmo prossegue

escolhendo outra semente até que todas sejam utilizadas, que se constitui em um dos critérios de parada de algoritmo.

4.3.4 Algoritmo Proposto

Na Figura 4-10 é apresentada uma visão geral do algoritmo proposto. As modificações efetuadas em relação ao algoritmo inicial estão em negrito e serão explicadas a seguir:

1. Início;
2. **Calcular valor médio para cada uma das primitivas;**
3. **Estabelecer prioridade para remoção de primitivas;**
4. Carregar classificador previamente ajustado e treinado;
5. Salvar conjunto atual de primitivas em conjunto anterior de primitivas;
6. Seleciona aleatoriamente, para remoção, uma primitiva;
7. **Se (prioridade da primitiva selecionada = zero) então remove-a (substitui seu valor pelo seu valor médio) e atualiza máscara de removidas;**
8. **Senão incrementa o valor da prioridade da primitiva e retorna ao passo 5;**
9. Avaliar taxa de classificação sobre Validação 2;
10. Se (taxa atual <= taxa anterior) confirma (conjunto atual de primitivas);
11. Senão retorna (conjunto anterior de primitivas);
12. Se (taxa atual <= erro mínimo) então Fim;
13. Se (número de removidas < NR) então retorna ao passo 5;
14. **Retorna conjunto inicial com todas as primitivas;**
15. Salvar condição (conjunto atual de primitivas) como máximo local em memória e disco;
16. Se (número sementes = NR) então Fim;
17. Retorna ao passo 5;
18. Fim;

NR – número de primitivas removidas.

Figura 4.10: Algoritmo proposto

O algoritmo inicia determinando o valor médio de cada uma das primitivas do conjunto inicial considerando os exemplares do conjunto de Validação 2, que será utilizado quando da remoção de uma primitiva. Segue determinando a influência que a remoção de uma única primitiva tem sobre a taxa de classificação do conjunto de Validação 2. Baseado nesta influencia é determinada uma prioridade de remoção para cada uma das primitivas do conjunto inicial. No passo seguinte é carregado o classificador, que consiste em uma rede neural ajustada e treinada com o conjunto inicial de primitivas. Na parte central do algoritmo inicia o processo randômico de seleção randômica de primitivas a ser removida. Caso a prioridade da primitiva selecionada seja igual a zero, ela será removida. Caso contrário terá o valor da prioridade diminuído de uma unidade e uma outra primitiva será selecionada aleatoriamente. Ocorre uma repetição destes passos até que uma primitiva seja

efetivamente removida. Na ocorrência de uma remoção o algoritmo segue calculando a taxa de erro de classificação sobre o conjunto de Validação 2.

Outra otimização apresentada neste método consiste na utilização do conceito de tolerância sobre o erro obtido nesta etapa de validação, este parâmetro não utilizado nesta explicação será discutido e avaliado nos experimentos deste trabalho.

Uma decisão é tomada baseada na taxa de erro obtida em relação à taxa de erro obtida na iteração anterior. Um máximo local é obtido após ocorrer a avaliação todas as primitivas do conjunto. Após registrar as informações relativas ao máximo local obtido, o algoritmo retorna a condição inicial do conjunto de primitivas em lugar de retornar apenas uma única camada, ou seja, uma única primitiva. O conjunto completo de primitivas é novamente considerado para a tarefa de remoção de uma primitiva, porém com a observação de somente poder remover de forma aleatória uma primitiva não removida na primeira iteração do algoritmo, ou seja, uma nova semente deve ser escolhida. O objetivo desta técnica é investigar outras áreas do espaço de soluções. Outra observação é de que a prioridade para remoção é uma forma de guiar o processo de busca enquanto o conceito de remoção aleatória é mantido.

A etapa de validação deste experimento utiliza um conjunto de Validação 3 com o objetivo de identificar o máximo local de melhor desempenho e conseqüentemente menos especializado em relação ao conjunto de Validação 2.

4.4 Máximos locais com tolerância na taxa de reconhecimento

No capítulo anterior destacou-se que um os problemas enfrentados pelo algoritmo tradicional de Subida de Encosta é a existência de regiões, no espaço de soluções, que se assemelham a planaltos. Muitos máximos locais também podem apresentar estas características. A forma tradicional de enfrentar tal situação seria prosseguir, removendo primitivas mesmo que a função de controle esteja dizendo justamente o contrário.

Uma variação desta solução seria modificar o algoritmo proposto, priorizando uma redução ainda maior do número de primitivas, mesmo que isto acarrete um aumento nos erros de classificação. Este aumento no índice de erros de classificação passa a ser uma das variáveis do sistema. Devido a forma de trabalho do algoritmo, é possível determinar, de forma direta, uma certa tolerância máxima no valor da taxa de classificação que ocorre junto ao conjunto de Validação 2 e de forma indireta sobre a

taxa de reconhecimento final, que é efetivamente determinada somente ao final de todo o processo.

Determinação de um critério de parada

Partindo dos máximos locais obtidos na etapa anterior, onde não havia sido permitido nenhum acréscimo nas taxas de erro, aumenta-se gradativamente a tolerância sobre a taxa de reconhecimento do conjunto de Validação 2, o que conduz também a uma gradativa redução do número de primitivas utilizadas. Porém, deve-se ressaltar que a taxa de reconhecimento final é a taxa obtida sobre o conjunto de Testes e não sobre o conjunto de Validação 2. Ao remover um número cada vez maior de primitivas, enquanto se observa um aumento da taxa de erro sobre o conjunto de Validação 2, é promovida uma especialização em relação ao conjunto de exemplares. Ao fim desta etapa do processo, novos máximos locais são obtidos, sendo submetidos a nova validação frente ao conjunto de Validação 3, que irá determinar o melhor dos máximos locais.

Como esta etapa trabalha com um número reduzido de primitivas, e até certo ponto especializadas no conjunto de Validação 2, é de se esperar grande variação da taxa de erro de classificação frente à Validação 3 para cada um dos máximos locais. O resultado final desta seleção tende a apresentar o máximo local que generalize melhor entre os conjuntos de Validação 2 e 3, não importando o número de primitivas. Vale lembrar que o número de primitivas varia de máximo local para máximo local.

Este termo de equilíbrio entre os conjuntos de validação ocorrerá quando o algoritmo estiver trabalhando a uma determinada tolerância sobre a taxa de erro de classificação. O final de todo o processo deve ocorrer quando não se apresentar como resultado final a escolha de um conjunto de primitivas menor que o anterior, ou seja, mesmo com o aumento da tolerância sobre a classificação do conjunto de Validação 2 não é mais possível obter, como resposta final, um máximo local com número menor de primitivas que o obtido para a taxa de erro anterior.

4.5 Classificador em Cascata

Com os resultados obtidos no experimento anterior implementou-se um classificador em cascata. Cada módulo é composto por um dos resultados de tolerância.

Os módulos passam a ser identificados de C5 até C0. O conjunto completo de exemplares é aplicado ao módulo C5.

O arquivo de exemplares "rejeitados" aplicado ao primeiro módulo do classificador, C5, contém na verdade a indicação de todos os exemplares como rejeitados, a fim de que todos sejam avaliados pelo módulo. O último módulo não irá gerar arquivo de rejeitados, pois todos os exemplares restantes devem ser analisados. Tarefas executadas em cada um dos módulos:

a) Carregar do arquivo de resultados gerado pela rede neural, para o módulo em questão, apenas os exemplares considerados rejeitados;
b) Identificar as duas classes com a maior probabilidade de serem a saída correta. Estabelecer a diferença aritmética entre os dois valores. Esta diferença é utilizada para determinar, entre todos os exemplares, os que possibilitam maior chance de confusão entre classes e terão prioridade para rejeição dentro do algoritmo;
c) São removidos alguns exemplares, aqueles que apresentam a maior probabilidade de confusão;
d) Desconsiderando os caracteres rejeitados, é verificada a taxa de reconhecimento para cada uma das classes;
e) A taxa de reconhecimento da classe é comparada ao valor inicialmente estabelecido, por exemplo, igual a 99,5%. Caso este valor tenha sido atingido não serão mais removidos exemplares desta classe.
f) O algoritmo continua sua execução retornando ao item "c", até que a taxa de reconhecimento especificada ocorra em todas as classes;
g) Cria-se uma lista determinando os exemplares a serem rejeitados e enviados para análise no próximo módulo.

Uma condição especial, para o último módulo classificador, é imposta. Onde não é mais possível haver rejeição de exemplares, sendo que a taxa de reconhecimento considerada como critério de parada será igual a 100%.

Devido à taxa de reconhecimento dos dígitos estar próxima de 100%, o índice de erro máximo de 0,5% foi utilizado junto aos módulos. São apresentados também os resultados para os módulos trabalhando com taxa de erro igual a 0%.

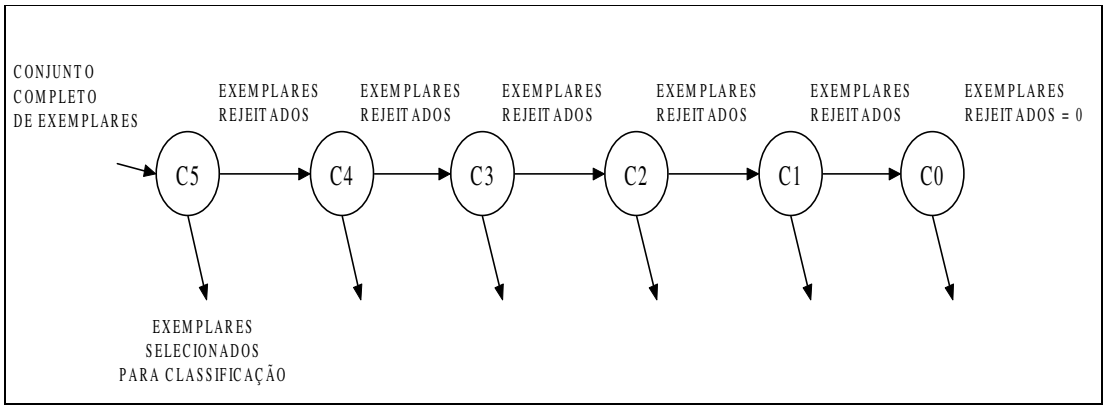

Figura 4.11: Classificador em cascata

Não existem alterações no conjunto de primitivas utilizados nos diferentes módulos. As diferentes taxas de reconhecimento, são devidas ao diferente número de primitivas que existem em cada módulo. Os exemplares que não estão em condições de serem corretamente classificados em C5 serão enviados para C4, e assim sucessivamente até que os últimos caracteres, ainda sem condições de serem classificados, serão submetidos ao último módulo, onde não há rejeição e todos os caracteres restantes serão submetidos à classificação.

Não se espera uma melhora na taxa de classificação. A taxa final deve ser igual à taxa considerando a utilização do conjunto completo de primitivas. Entretanto é possível observar uma significativa redução da complexidade computacional para a tarefa de classificação.

4.5.1 Complexidade Computacional

O cálculo da complexidade leva em conta a simplificação do fator tempo, que deixa de ser utilizado nas formulações propostas. De maneira que o valor obtido representa o número de multiplicações efetuadas para cada um dos classificadores avaliados. Para explicar melhor, considera-se a complexidade do classificador original como sendo:

$$Compl_{OC} = ns \times nc \qquad \textbf{(4.1)}$$

onde:

ns: número de exemplares da base;
nc: número de conexões do classificador MLP;

e, a complexidade do classificador modular como:

$$Compl_{CC} = \sum_{i=1}^{NC} \left(ns_i \times nc_i \right)_I \qquad \textbf{(4.2)}$$

onde:

NC: número de classificadores;

ns_i: número de exemplares classificados pelo i^{th} classificador;

nc_i: número de conexões do i^{th} classificador MLP.

4.6 Considerações Finais

O método proposto tem por objetivo realizar experimentos que explorem todas as otimizações propostas, de maneira a reduzir ao máximo o número de primitivas necessárias à classificação bem como reduzir o esforço computacional. O uso conjunto de prioridade para remoção de determinadas primitivas pouco relevantes a classificação, utilização do conceito de nova semente para obter os máximos locais e a aplicação de tolerância da taxa de erro para que o algoritmo possa avaliar a região do espaço de soluções ao redor do máximo local, visa assegurar que os resultados obtidos sejam consistentes e com boa generalização em relação aos exemplares da base de dados. A redução do esforço computacional é um indicador interessante, pois representa de uma maneira adimensional os cálculos realizados pelo método proposto, permitindo a comparação da complexidade apresentada pelos resultados de outros algoritmos.

Capítulo

5 Experimentos e Resultados

A realização dos experimentos segue, quando possível, a ordem apresentada na metodologia e tiveram por objetivo atestar de forma individual a qualidade das otimizações propostas. A partir destas verificações foi possível assegurar o cumprimento do objetivo principal deste trabalho. Toda a estrutura utilizada foi aproveitada em experimentos adicionais, que incluem o conceito de tolerância sobre os máximos locais e um exemplo de classificador em cascata. Os experimentos foram todos efetuados em um servidor bi-processador, modelo Intel de 1,1 GHz e com 3 GB de memória em ambiente Red Hat 8.0 Linux.

5.1 Base de dados

Uma das primeiras tarefas consistiu em obter informações a respeito da base de dados NIST utilizada em todas as fases e experimentos do trabalho. A escolha permite que o método proposto tenha sua eficiência verificada e comparada a outras publicações. Maiores informações podem ser obtidas nas referências bibliográficas [GAR92, BRI00, KOE03]. A base de dados está dividida em dígitos (0 até 9), caracteres maiúsculos (A até Z) e minúsculos (a até z) e trata-se de uma base de dados extensa, com muitas variações para cada padrão. Todas as imagens dos exemplares utilizados foram cedidas para este trabalho já binarizadas, pré–processadas e armazenadas em arquivos gráficos de formato TIFF. A fase de pré-processamento existente neste projeto está restrita apenas a correção em escala dos exemplares, condição necessária para a correta utilização do zoneamento dos caracteres.

Os vetores de primitivas serão distribuídos em quantidades pré-determinadas e em arquivos assim caracterizados:

Tabela 5.1: Divisão dos arquivos da base de dados

CONJUNTOS	CARACTERES MINÚSCULOS		CARACTERES MAIÚSCULOS		DÍGITOS	
		Do total		Do total		Do total
Treinamento	37.440	61%	37.440	61%	195.000	77%
Validação 1	3.859	6%	4.030	7%	9.336	4%
Validação 2	3.859	6%	4.031	7%	9.336	4%
Validação 3	3.860	6%	4.031	7%	9.336	4%
Testes	12.000	20%	11.941	19%	30.089	12%
Totais	61.018		61.473		253.097	

O número de exemplares em cada arquivo é determinado também pelo número de exemplos nos arquivos originais da base NIST. Utilizam-se as séries hsf_0, 1, 2, 3, 4 e 7. Cada uma com um diferente número de exemplares.

5.2 Experimentos iniciais

Os primeiros experimentos tiveram por objetivo disponibilizar um classificador para dígitos e caracteres manuscritos. Os resultados possuem qualidade suficiente para não comprometer a eficiência dos experimentos seguintes. Uma das maneiras de verificar esta qualidade é através da comparação de resultados com o trabalho recente de outros pesquisadores. Ao longo da apresentação dos diversos resultados, serão apresentadas comparações.

5.2.1 Camada Escondida

A determinação do número de nós da camada escondida consiste em testar várias configurações e alguns números foram definitivamente adotados. A fórmula de Baum-Haussler foi utilizada e, considerando-se as variáveis do contexto proposto, apresentou os seguintes resultados:

Tabela 5.2: Resultados da aplicação da fórmula de Baum-Haussler

	Num de exemplares	Camada de entrada	Camada de saída	Erro tolerado	Valor calculado
Caracteres Maiúsculos	37.440	132	26	0,1	24 nós
Dígitos	195.000	132	10	0,01	14 nós

Para caracteres maiúsculos a resposta obtida foi um número de nós igual a 24, que é menor do que o número de classes de saída. Para dígitos manuscritos o valor obtido foi igual a 14 que também é menor do que o número de entradas. Porém, de maneira geral, a regra Baum-Haussler serviu de ponto de partida. O critério de decisão, efetivamente utilizado, foi o acompanhamento da taxa de classificação efetiva. Havendo igualdade de desempenho entre duas topologias testadas, a topologia escolhida é a que possui o menor número de nós, visando diminuição da complexidade computacional. Os valores que foram escolhidos para determinar a topologia da rede a ser usada com caracteres maiúsculos manuscritos foram os seguintes: 24[1], 40, 80, 100 e 120 nós e apresentaram as seguintes taxas de classificação para o conjunto de Teste: 88,80%, 88,86%, 88,90%, 90,00%, 90,39% e 89,35% respectivamente.

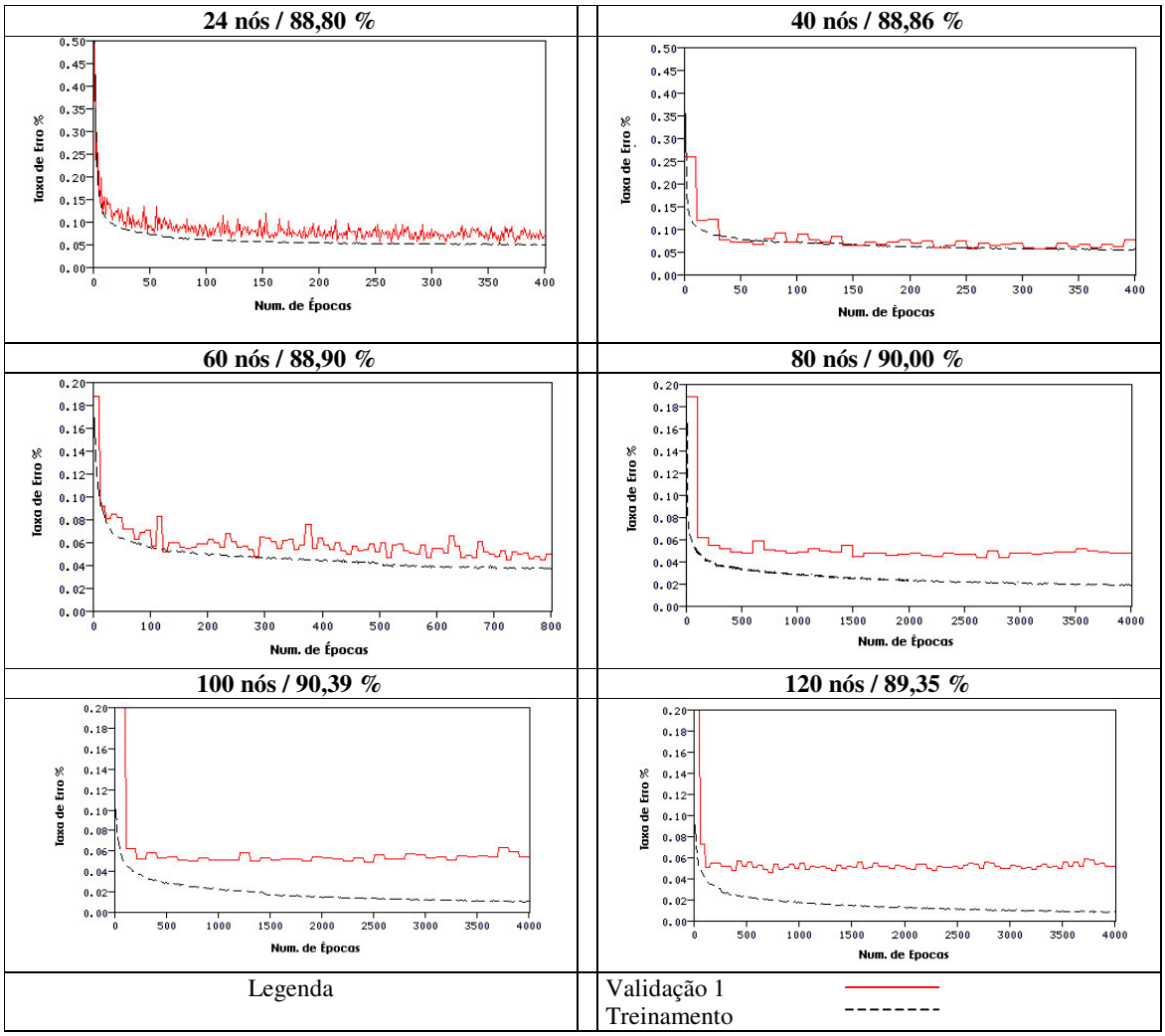

Figura 5.1: Número de nós da camada escondida

[1] Determinado pela aplicação direta da fórmula de Baum-Haussler.

O critério de parada utilizado foi a taxa de reconhecimento efetiva em relação ao número de nós da camada escondida. Uma análise visual dos gráficos obtidos permite verificar que a taxa de reconhecimento obtida pelos conjuntos de treinamento e de Validação 1 passam a apresentar valores cada vez mais divergentes. A partir de 100 nós a taxa de reconhecimento apresenta diminuição do valor. Portanto, quanto maior o número de nós da camada escondida, maior é a especialização da rede em torno do conjunto de Treinamento. O valor igual a 100 nós foi o escolhido para ser utilizado nos experimentos seguintes.

5.2.2 Treinamento da Rede Neural

A correta determinação do número de épocas de treinamento é outro critério importante para evitar especialização da rede neural. A Figura 5-2 apresenta, como exemplos, os gráficos de treinamento e de validação que ocorrem durante a preparação de uma rede neural na aplicação SNNS. É possível notar que a curva de validação, linha superior, que atua sobre o conjunto de Validação 1, não apresenta um ponto definido do momento em que passa a ocorrer o sobre-treinamento, condição a ser evitada sob pena de redução da taxa efetiva de reconhecimento sobre o conjunto de Teste.

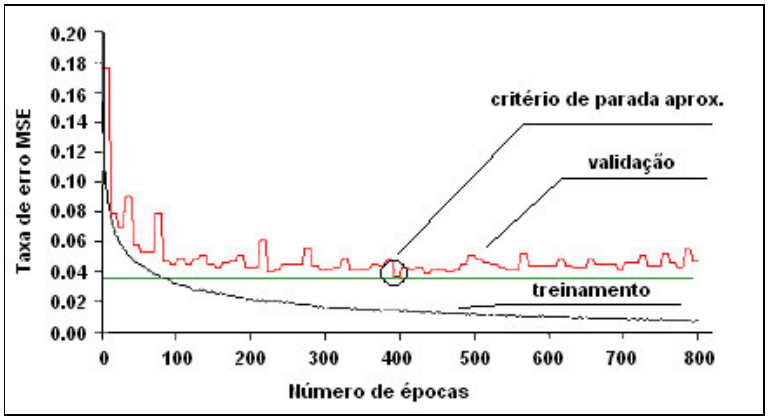

Figura 5.2: Determinação do critério de parada da rede neural

Os treinamentos iniciais permitiram a obtenção de um valor para o número de épocas de treinamento em que deve ocorrer a interrupção do treinamento da rede, porém este só é corretamente observado através de um gráfico deste tipo, gerado durante o trinanemento da rede, não sendo o ideal para se obter as melhores taxas de classificação.

Então, para se obter o melhor número de épocas de treinamento foi necessário uma verificação efetiva do número de caracteres reconhecidos corretamente a cada época de treinamento. Isto foi possível, devido ao fato de que, a cada época de treinamento, os resultados da classificação eram disponibilizados em arquivo. Baseando-se no conteúdo deste arquivo foi feita uma contagem da quantidade de caracteres reconhecidos corretamente. Um índice foi gerado e enviado para controle do algoritmo. Desta forma o critério de parada foi determinado de forma precisa através de rotina automática que verifica a condição da taxa de erro de reconhecimento sobre o conjunto de Validação 1. Trabalhando da seguinte forma:

a) É armazenada inicialmente a taxa de erro de classificação igual a 100% para a primeira iteração ou época 0;
b) O treinamento da rede ocorre por até 800 épocas. Observações iniciais dos gráficos, que representam as curvas de treinamento e validação, demonstraram ser este valor suficiente para os experimentos em questão;
c) O processo de treinamento da rede inicia e roda livremente até que um determinado valor máximo de MSE seja atingido. Desta forma, não é perdido tempo avaliando as primeiras épocas que apresentam taxas de erro ainda elevadas;
d) Quando determinada época atinge o MSE máximo permitido, é calculada a taxa efetiva de reconhecimento a que ele corresponde. O arquivo disponibilizado contém, para cada exemplo testado, o valor da classe de entrada e os valores obtidos para cada classe de saída. De posse destes dados é possível determinar quantos caracteres foram corretamente classificados e quantos não foram, obtendo então uma taxa de reconhecimento efetiva. Este valor é armazenado junto à informação do número da época. Armazena-se também a rede treinada, ou seja, o conjunto completo de pesos da rede treinada;
e) O algoritmo continua a treinar a rede, e sempre que o MSE ficar abaixo do limite máximo, será verificada a taxa efetiva. Caso seja menor do que a previamente armazenada, esta nova época assume a condição de melhor época, descartando-se a anterior;
f) O algoritmo segue treinando normalmente até a época 800, mesmo que não sejam mais observados valores melhores de MSE. O critério de parada indica

efetivamente quando parar de armazenar a rede treinada, porém não interrompe o treinamento.

Após esta fase de ajuste, efetua-se uma etapa de teste para verificar a efetiva taxa de reconhecimento atingido sobre um conjunto de Teste, que não participou das etapas de validação e treinamento. A melhor rede treinada é armazenada e será utilizada para os experimentos seguintes. O algoritmo que determina com exatidão qual época de treinamento apresenta a melhor taxa de reconhecimento efetivo sobre o conjunto de Validação 1, foi utilizado para cada um dos experimentos.

5.2.3 Resultados dos experimentos para ajuste do classificador

Os resultados obtidos para caracteres maiúsculos manuscritos são comparados aos resultados obtidos em [KOE03]. O protocolo experimental adotado foi o mesmo. O objetivo é testar o classificador obtido, utilizando a base NIST para treinamento, validação, teste com um conjunto inicial de 132 primitivas e um número de classes igual a 26.

Tabela 5.3: Divisão das bases de dados

Séries NIST	**(hsf 0,1,2,3)**	**(hsf 7)**	**(hsf 4)**
Conjunto	Treinamento	Validação	Testes
Exemplos	37.440	12.092	11.941

Os valores obtidos foram os seguintes:

Tabela 5.4: Resultados iniciais obtidos para caracteres maiúsculos manuscritos

Experimentos	**# Primitivas**	**Treinamento**	**Teste**
Classificador ajustado	132	97,23%	93,05%
Resultados em [KOE03]	132	97,87%	92,49%

Os resultados obtidos mostraram-se interessantes e foram até 0,5% superiores, indicando adequação do conjunto de primitivas proposto para a tarefa de classificação de caracteres maiúsculos manuscritos.

Um dos objetivos do trabalho de [KOE03] é efetuar uma combinação manual de diferentes conjuntos de primitivas, para obter a melhor taxa de reconhecimento sobre a mesma base de dados NIST.

A segunda base utilizada foi a de caracteres minúsculos manuscritos. Foi mantido o mesmo protocolo experimental. Utilizando a base de dados composta de exemplares das séries NIST para treinamento, validação, teste com um conjunto inicial de 132 primitivas e um número de classes igual a 26.

Tabela 5.5: Base de caracteres minúsculos manuscritos

Séries NIST	**(hsf 0,1,2,3)**	**(hsf 7)**	**(hsf 4)**
Conjunto	Treinamento	Validação	Testes
Exemplos	37.440	11.578	12.000

Os valores obtidos foram os seguintes:

Tabela 5.6: Resultados para caracteres minúsculos manuscritos

Experimentos	**# Primitivas**	**Treinamento**	**Teste**
Classificador ajustado	132	95,25%	84,39 %
Experimentos em [KOE03]	132	95,82%	86,73%

Os resultados obtidos para caracteres minúsculos manuscritos ficaram 2,34% abaixo dos valores obtidos por [KOE03]. Um dos possíveis motivos para a ocorrência desta diferença pode ser a presença de primitivas menos eficientes na avaliação da dimensão vertical dos exemplares. O conjunto inicial de primitivas foi sido escolhido, inicialmente, com a intenção de efetuar a classificação de dígitos manuscritos que apresentam menor variação de altura inter-classe.

A terceira base utilizada corresponde à base de dígitos manuscritos. O conjunto de primitivas utilizado permaneceu o mesmo dos experimentos anteriores e segue o protocolo experimental baseado no protocolo utilizado em [OLI01a, OLI02b]. Estruturado de forma a permitir uma comparação direta dos resultados obtidos.

Tabela 5.7: Base de dígitos manuscritos

Séries NIST	**(hsf 0,1,2,3)**	**(hsf 0,1,2,3)**	**(hsf 7)**
Conjunto	Treinamento	Validação	Teste
Exemplos	195.000	9.336	30.089

Os valores obtidos foram os seguintes:

Tabela 5.8: Resultados para dígitos manuscritos

Experimentos	**# Primitivas**	**Treinamento**	**Teste**
Classificador ajustado	132	99,77%	99,10%
Resultados em [OLI02]	132	99,66%	99,13%

De maneira geral os resultados obtidos são semelhantes indicando boa qualidade do conjunto de primitivas. Para um processo de remoção de primitivas, baixos índices de reconhecimento poderiam impedir uma correta avaliação do processo de remoção de primitivas.

5.3 Experimentos para Seleção de Primitivas

Para os próximos experimentos, que tratam da seleção de primitivas, algumas restrições foram colocadas. Neste ponto da experimentação a observação de alguns resultados obtidos sugerem que os esforços sejam direcionados para um número menor de bases de dados. As seguintes situações verificadas nos etapas anteriores serviram de apoio a esta decisão:

a) Elevado tempo de processamento necessário ao algoritmo de busca. É necessário um elevado número de iterações para a obtenção dos diversos máximos locais, mesmo qunado se considera o algoritmo otimizado;

b) Os experimentos para ajuste do classificador apresentaram desempenho apenas regular para classificação de caracteres minúsculos manuscritos.

c) As bases de dados utilizadas podem ser consideradas independentes devido ao elevado número de exemplares e presença de grande variabilidade nos padrões.

Desta forma os experimentos com o algoritmo de busca utilizaram apenas as bases de dados de caracteres maiúsculos e dígitos manuscritos. Tal simplificação dos experimentos não compromete a qualidade dos resultados, pois são utilizadas bases completas e o objetivo principal do trabalho é a experimentação das otimizações aplicadas ao algoritmo tradicional.

5.3.1 Número de subconjuntos avaliados

Algoritmos de busca heurística possuem o inconveniente de, no pior dos casos, efetuarem uma busca exaustiva. Portanto, critérios de parada devem ser definidos. Para

o algoritmo inicial, sem otimizações, o critério de parada escolhido foi o do número de máximos locais armazenados. Tal algoritmo, não atinge rapidamente este critério de parada. A possibilidade de uma busca exaustiva ainda existe e representa a realização de um incontável número de iterações. A cada iteração do algoritmo um novo subconjunto é gerado e avaliado.

O algoritmo proposto, contando com todas as otimizações propostas, atinge um dos critérios de parada definidos em valores próximos a 10.000 ou 16.000 iterações, considerando as bases de caracteres maiúsculos manuscritos e dígitos manuscritos, respectivamente. Nestas condições, o algoritmo conseguiu utilizar todas as sementes possíveis, gerando o conjunto de máximos locais, disponível para a próxima etapa do processo.

Como forma de poder comparar os resultados obtidos pelas otimizações implementadas, foi permitido ao algoritmo inicial efetuar também 10.000 ou 16.000 testes de remoção de um diferente número de primitivas, ou seja, o número de iterações permitidas foi limitada aos valores atingidos com as otimizações. Estes números foram adotados como critério de parada e, neste momento o importante é ressaltar a necessidade de um número igual de iterações entre o algoritmo inicial e o algoritmo otimizado.

5.3.2 Experimentos com o algoritmo de busca inicial

A cada iteração do algoritmo sobre a base de dados de caracteres maiúsculos manuscritos, ocorre a geração de um novo subconjunto de primitivas. Estes subconjuntos são avaliados e caso não apresentem um bom resultado não são utilizados após o término da busca. Seus resultados não são nem armazenados pelo algoritmo, pois são apenas resultados parciais. Porém, são úteis para ilustrar o funcionamento do algoritmo. A Figura 5-3 apresenta o conjunto de 10.000 iterações realizadas pelo algoritmo inicial sobre a base de caracteres maiúsculos. Conforme explicado, este algoritmo efetuou bem mais do que 10.000 iterações, porém para efeito de comparação, com o algoritmo otimizado proposto, foram consideradas apenas as 10.000 iterações iniciais.

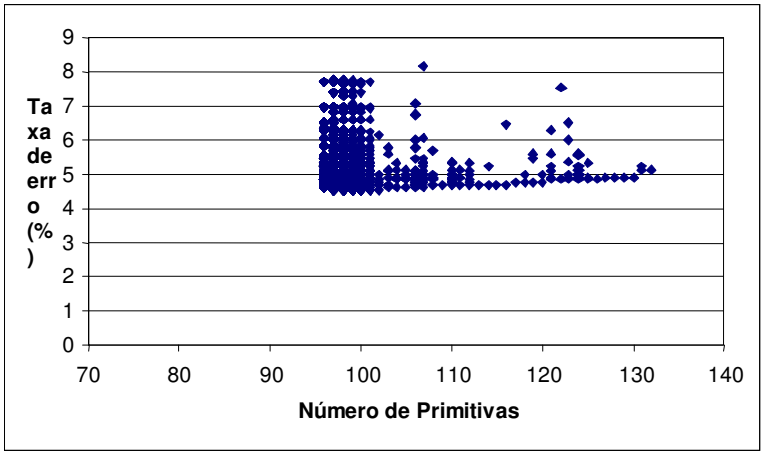

Figura 5.3: Iterações do algoritmo convencional

Da mesma forma que o experimento anterior, o algoritmo inicial foi aplicado a base de números manuscritos. São 16.000 subconjuntos gerados e representam o conjunto das primeiras iterações realizadas pelo algoritmo convencional.

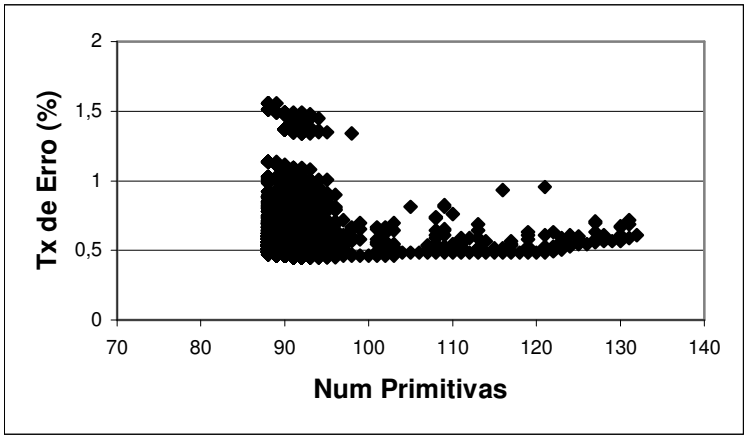

Figura 5.4: Subconjuntos gerados pelo método inicial

Nas Figuras 5-3 e 5-4 ocorre uma concentração dos valores obtidos na parte inferior, entre 85 e 98 primitivas, ou seja, foram realizados poucos testes fora desta faixa de valores. Isto ocorre devido ao algoritmo de busca ter como objetivo uma redução do erro de classificação. A cada iteração esta condição é testada, impedindo que determinadas combinações de primitivas que apresentem resultados contrários à função critério sejam utilizadas.

A Figura 5-5 apresenta apenas os valores obtidos para os máximos locais, indicados pela taxa de erro do conjunto de Validação 2. No caso dos dígitos manuscritos

é igual a 47 máximos locais. Estes máximos locais são submetidos a novo processo de validação, agora sobre o conjunto de Validação 3.

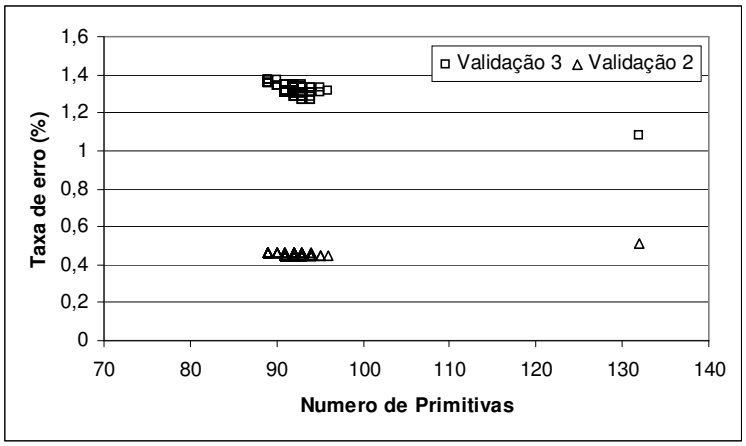

Figura 5.5: Resultados das etapas de validação do algoritmo inicial

Os valores obtidos para os máximos locais são semelhantes. Ocorre uma concentração de resultados em torno do conjunto de 47 máximos locais. O número de primitivas removidas para os máximos locais varia entre 36 e 43. A concentração dos resultados é causada pelo fato do algoritmo inicial ter dificuldade em explorar outras regiões do espaço de soluções. Efetua esta exploração de forma lenta, após um número elevado de iterações. A tabela 5-9 apresenta os resultados obtidos com a remoção de primitivas do conjunto inicial com a utilização do algoritmo de Seleção de Primitivas Inicial cujo acrônimo passa a ser ASP-I.

Tabela 5.9: Resultados para dígitos manuscritos

Experimentos	**# Primitivas**	**Treinamento**	**Teste**
Conjunto inicial de primitivas	132	99,77%	99,10%
ASP-I (usando val_3)	92	99,50%	99,04%
ASP-I (menor conjunto)	81	99,50%	98,92%

São apresentados os valores para o algoritmo ASP-I referente à seleção do subconjunto de acordo com o resultado junto ao conjunto de Validação 3 e pela escolha direta do menor subconjunto gerado entre os máximos locais obtidos. A diferença entre os dois valores é mínima indicando que os máximos locais possuem boas características de generalização, conforme é possível verificar na Figura 5-5 e pela pequena variação da taxa de erro de classificação do conjunto de Validação 3 .

5.3.3 Experimentos com as Otimizações Propostas

Utiliza o algoritmo de busca Subida de Encosta com mutação randômica otimizado, chamado aqui de Algoritmo de Seleção de Primitivas Otimizado, cujo acrônimo a ser usado é ASP-Ot. Prevê a utilização do conceito de prioridade para remoção de determinadas primitivas e também, ao encontrar um máximo local, armazena-o e efetua um retorno ao conjunto inicial de primitivas. Seleciona para remoção, uma primitiva que ainda não tenha sido a primeira a ser removida e desta forma procura evitar percorrer regiões do espaço de soluções que já tenham sido testadas.

5.3.4 Resumo do Protocolo Experimental

A seguir é apresentada uma lista dos principais pontos do protocolo para teste do algoritmo proposto:

Fase	Nome	Passos	Objetivos
1ª	Treinamento inicial do classificador	1. Treinamento considerando todas as primitivas e a base de treinamento; 2. Validação utilizando a conjunto de Validação 1; 3. Testar o comportamento da rede, em relação ao erro de classificação considerando o conjunto de Testes;	Obter uma rede ajustada considerando todas as primitivas.
2ª	Usar algoritmo Subida de Encosta com Mutação Randômica	1. Aleatoriamente selecionar as primitivas que serão substituídas pelos seus valores médios, ou seja, redução do número de primitivas; 2. Efetuar validação com conjunto de Validação 2; 3. Ao atingir um dos critérios de parada, armazenar informações deste máximo local; 4. Efetuar teste com conjunto de Validação 3, armazenar resultado; 5. Parar algoritmo quando atingir um dos critérios de parada final;	Identificar no sub conjunto de máximos locais aquele que apresenta o menor erro de classificação.

Fase	Nome	Passos	Objetivos
3ª	Reajuste da topologia da rede neural	1. Alterar a topologia da rede neural de acordo com o resultado obtido como sendo o melhor sub conjunto de primitivas; 2. Treinar, validar e testar a nova rede;	Com o novo resultado efetivamente verificar a melhoria da classificação em relação ao conjunto original de primitivas.

Quadro 5-1: Etapas para experimentação do método proposto

Novamente os experimentos utilizam a base de caracteres maiúsculos manuscritos e a Figura 5-6 apresenta o resultado de 10.000 iterações. Porém é possível notar que os subconjuntos testados apresentam-se mais distribuídos em relação ao número de primitivas removidas. É possível comparar diretamente com a Figura 5-3. As taxas de erro de classificação ficam um pouco menores do que as apresentadas quando da utilização do algoritmo inicial.

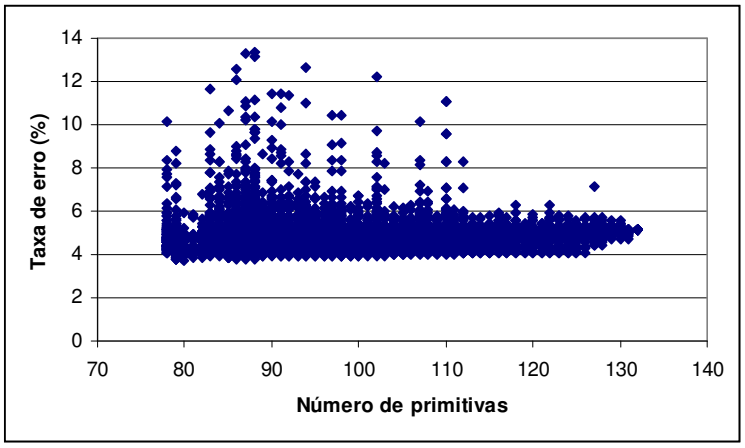

Figura 5.6: Iterações com conceito de prioridade e nova semente

As taxas finais de classificação também foram obtidas de diferentes formas. Utilizando o melhor subconjunto de acordo com o conjunto de Validação 3 e utilizando o menor conjunto obtido. Os valores ainda são comparáveis aos obtidos por [KOE03], mesmo quando é considerado um subconjunto de apenas 79 primitivas.

Tabela 5.10: Resultados para caracteres maiúsculos manuscritos

Experimentos	# Primitivas	Treinamento	Teste
Conjunto de primitivas original	132	97,23%	93,05%
ASP-Ot (Validação 3)	101	97,57%	92,51%
ASP-Ot (menor conj. máximo local)	79	97,68%	92,50%
Resultado em [KOE03]	132	97,87%	92,49%

Considerando agora a base de dígitos manuscritos, na Figura 5-7 são representados os 16.000 subconjuntos testados durante o funcionamento de método otimizado ASP-Ot:

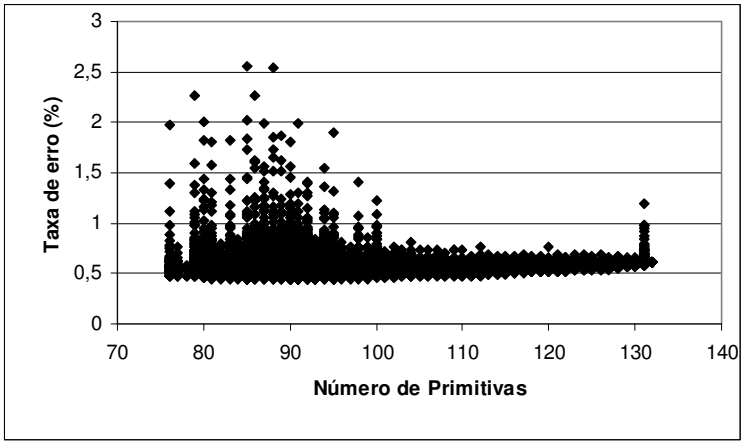

Figura 5.7: Subconjuntos gerados pelo método proposto

Ao estabelecer uma prioridade para a remoção de determinadas primitivas evita-se que primitivas importantes para a classificação sejam removidas. Na Figura anterior, isto pode ser visualizado na região entre 100 e 132 primitivas. Não existem conjuntos com altas taxas de erro nesta região, devido ao fato de que não é permitida a remoção de primitivas relevantes ao classificador. O número de primitivas removidas nos máximos locais atinge valores maiores, variando de 31 até 55 primitivas.

No conjunto de Validação 2 existem apenas 47 primitivas que atendem a condição de poder iniciar o processo de remoção. É possível verificar na Figura 5-8, duplicada neste capítulo intencionalmente para maior clareza, que a taxa de erro obtida com a remoção de cada uma das primitivas de forma individual, em alguns casos pioram a taxa de erro. Para o conjunto completo de primitivas a taxa de erro de classificação para o conjunto Validação 2 é igual a 0,6105%. As primitivas que

apresentam marcas piores que esta não serão consideradas removíveis na primeira camada.

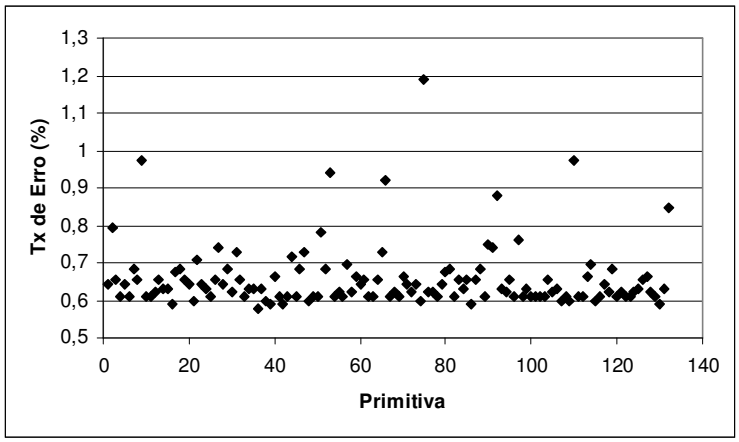

Figura 5.8: Taxa de erro resultante da remoção individual

Tais primitivas não estão em condições de serem removidas, pois vão contra o critério de busca estabelecido o que acaba por determinar o número de máximos locais a serem obtidos pelo algoritmo. O número de máximos locais obtidos é igual a 47.

A Figura 5-9 apresenta os resultados obtidos dos máximos locais do algoritmo otimizado. Pelo resultado fica evidente a maior dispersão dos valores obtidos, e que são resultado das otimizações apresentadas.

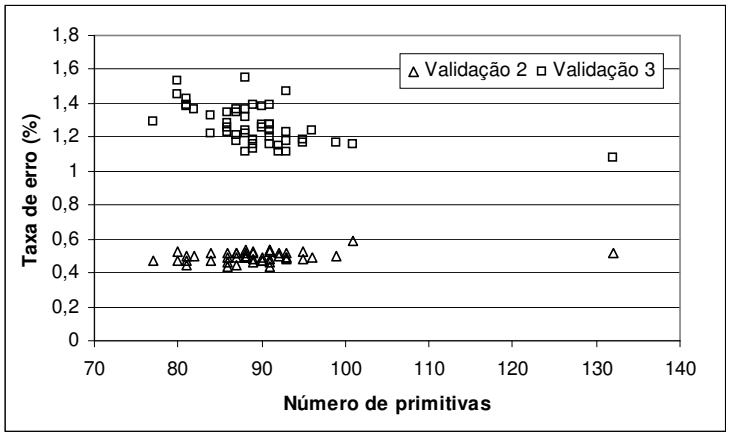

Figura 5.9: Resultados da etapas de validação do algoritmo proposto

A variação do erro sobre o conjunto de Validação 3 apresenta um delta de 0,5 %. Pela variação apresentada é possível prever que o subconjunto de máximo local, obtido pela análise do conjunto de Validação 3, tem pouco impacto sobre a taxa final de classificação.

A Tabela 5-11 apresenta os resultados obtidos. Uma redução de 42% do número de primitivas ocasionou uma redução de 0,16% nas taxas de classificação.

Tabela 5.11: Resultados para dígitos manuscritos

Experimentos	**# Primitivas**	**Treinamento**	**Teste**
Conjunto de primitivas original	132	99,77%	99,10%
ASP-Ot (usando val_3)	87	99,95%	98,95%
ASP-Ot (menor conjunto de primitivas)	77	99,86%	98,94%

5.3.5 Experimentos com o conceito de Tolerância

Os experimentos utilizando o algoritmo otimizado, identificaram máximos locais com boas características de generalização para a tarefa de reconhecimento de dígitos e caracteres manuscritos. Utilizar o conceito de tolerância é uma forma de explorar, ainda mais, a região local de cada um dos máximos locais, tentando identificar um conjunto ainda mais reduzido em relação ao número de primitivas. Os experimentos foram conduzidos considerando as seguintes tolerâncias para a taxa de classificação: 1%, 2%, 4%, 8% e 12%. Para isto é levado em conta que, junto aos máximos locais obtidos na primeira fase as taxas de erro obtidas para o conjunto de Validação 2, variaram entre 0,44% e 0,54%.

Acrescenta-se ao algoritmo o parâmetro ERROR-TOL (Tolerância no erro). Existe uma preocupação em sair de uma região de planalto e buscar melhores taxas de erro, porém o objetivo principal desta técnica é buscar uma redução do número de primitivas do vetor final.

Para estes experimentos foram consideradas apenas as bases de dígitos manuscritos. Os experimentos foram obtidos seguindo o mesmo protocolo adotado anteriormente. Porém o ponto de partida do algoritmo de remoção passa a ser o conjunto de máximos locais, em número igual a 47, previamente obtido e armazenado. Uma condição de TOLERÂNCIA foi utilizada sobre os resultados obtidos, ou seja, em cada máximo local obtido, o procedimento de remoção de primitivas reiniciou, até que

um determinado nível de erro de classificação fosse obtido. O algoritmo atua sobre o conjunto de Validação 2 e as taxas de tolerância do erro, dizem respeito a este conjunto de validação (composto por 9336 exemplares de dígitos manuscritos). A tabela 5-12 apresenta os valores obtidos:

Tabela 5.12: Características dos módulos do classificador

Módulo Classificador	Tolerância (ERROR_TOL)	#	Treinamento	Teste	Incremento na Taxa de Erro
C1	0%	77	99,86%	98,94%	
C2	1%	59	99,30%	98,72%	-0,32%
C3	2%	52	99,40%	98,55%	-0,49%
C4	4%	33	98,92%	97,85%	-1,19%
C5	8%	25	98,65%	97,43%	-1,61%
C6	12%	26	98,06%	96,93%	-2,11%

A 6ª coluna – Taxa de Erro, indica a variação ocorrida em relação ao melhor valor obtido no experimento anterior considerando um subconjunto de 92 primitivas e com taxa de reconhecimento de 99,04%. O critério de parada adotado para o incremento da taxa de tolerância foi o da variação da taxa de erro no conjunto de Treinamento em relação ao número de primitivas do subconjunto escolhido como melhor.

Estes valores foram obtidos após testes sobre o conjunto de Validação 3 da forma como normalmente utilizado nos experimentos anteriores. Detalhes são apresentados no próximo subitem.

5.3.6 Conjunto de Validação 3

Etapa importante no método proposto, a taxa de classificação do conjunto de Validação 3 indica a generalização do máximo local. Na Figura 5-10, é apresentado o resultado conforme o aumento da tolerância permitida: 1%, 2%, 4%, 8% e 12%. É possível notar a redução gradual do número de primitivas dos máximos locais obtidos. Os valores são compostos pelo valor médio do número de primitivas de cada um dos máximos locais obtidos.

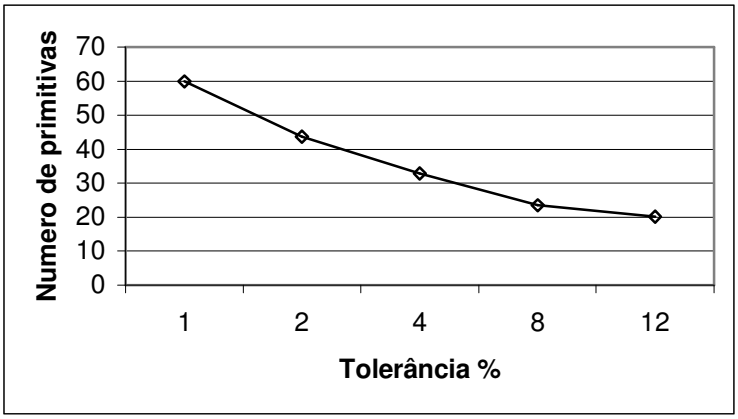

Figura 5.10: Número médio de primitivas dos máximos locais

Outro detalhe importante é notar que os máximos locais tornam-se cada vez mais especializados em relação ao conjunto de Validação 2. As taxas de erro impostas para a tolerância de erro de classificação de 1 a 12% são respeitadas junto ao conjunto de Validação 2, porém apresentam valores bem mais elevados quando observadas junto aos conjuntos de Validação 3. Na Figura 5-11 é apresentado o valor médio da taxa de erro de classificação para cada um dos conjuntos de validação em relação à tolerância permitida.

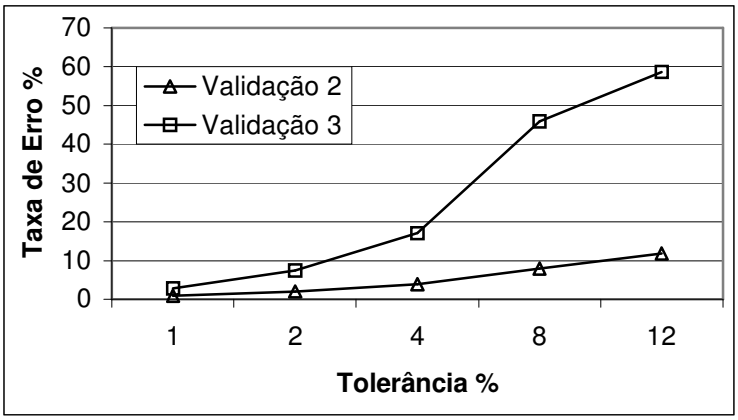

Figura 5.11: Comparação do incremento da taxa de erro

Para o valor de 8% de tolerância na taxa de erro, o valor médio da taxa de erro sobre o conjunto de Validação 3 foi de 46%. Em [OLI02], onde se utiliza uma estrutura semelhante com conjuntos de Validação 2 e 3, a diferença entre as taxas de erro dos dois conjuntos é bem mais expressiva mesmo ao considerar índices de tolerância 0, 1 ou 2%.

5.4 Classificador em Cascata

Os resultados obtidos nos experimentos anteriores foram reunidos em um classificador em cascata, como uma maneira de testar a eficiência desta técnica de classificação. Deve-se observar que a taxa de classificação final deve ser semelhante à taxa de classificação quando considerado o conjunto inicial de primitivas.

Estes testes foram efetuados sobre os exemplares de dígitos manuscritos. Quanto mais criterioso for um determinado módulo maior será o número de exemplares rejeitados. Como as taxas de reconhecimento iniciais giram em torno de 99,00%, optou-se por uma taxa de reconhecimento de 99,5% como padrão para cada módulo. Este valor produziu uma taxa de rejeição de aproximadamente 12% dos exemplares, considerada para a operação de um módulo inicial ou C1.

Tabela 5.13: Resultados para rejeição com taxa de erro 0,5%

	Classificadores Individuais						Combinação
Classificador	C_5	C_4	C_3	C_2	C_1	OC (classificador original)	
# primitivas	26	33	52	59	77	132	
# exemplares reconhecidos corretamente	26565	722	1381	430	179	528	29805
# exemplares reconhecidos incorretamente	129	1	4	1	0	149	284
# exemplares rejeitados	3395	2672	1287	856	677	0	0
% reconhecimento	99,5%	99,5%	99,5%	99,5%	99,5%	78,0%	99,1%
% rejeição	11,28%	78,70%	48,16%	66,51%	79,08%	0%	0%
% erros	0,5%	0,5%	0,5%	0,5%	0,5%	22%	0,9%
# conexões MLP	936	1419	2704	4071	6699	14200	

Havendo interesse é possível elevar ainda mais a taxa de reconhecimento de cada um dos módulos. A taxa de erro mínima para a operação de um módulo do classificador é igual a 0%. O resultado obtido para esta taxa de erro é apresentado a seguir como forma de auxiliar na observação do número de exemplares rejeitados, principalmente junto aos módulos iniciais. Desta forma é possível determinar o nível de rejeição que permita ao primeiro módulo classificar um bom número de exemplares e ao mesmo tempo justificar a existência de módulos com conjuntos de primitivas mais completos.

Tabela 5.14: Resultados para rejeição com taxa de erro 0%

	Classificadores Individuais						Combinação
Classificador	C_5	C_4	C_3	C_2	C_1	OC (classificador original)	
# exemplares reconhecidos corretamente	10534	6413	5292	2232	1160	4183	
# exemplares reconhecidos incorretamente	0	0	0	0	0	275	
# exemplares rejeitados	19555	13142	7850	5618	4458	0	0
% reconhecimento	100%	100%	100%	100%	100%	93,83%	99,1%
% rejeição	65,00%	67,21%	59,73%	71,57%	85,44%	0%	0%
% erros	0%	0%	0%	0%	0%	6,17%	0,9%
# conexões MLP	936	1419	2704	4071	6699	14200	

A taxa de erro de classificação igual a 0% provoca a rejeição de 65% dos exemplares no primeiro módulo, o que pode inviabilizar o uso dos módulos em cascata.

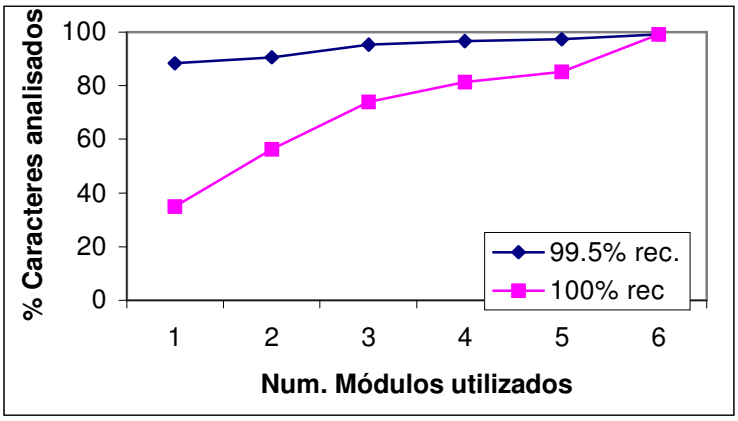

Figura 5.12: Comparação entre taxas de rejeição

Porém, para o caso de se utilizar módulos compostos de diferentes conjuntos de primitivas é possível aumentar ainda mais a taxa de rejeição pela correspondente diminuição do limite de erro de classificação para valores entre 0,5 e 0%.

Complexidade Computacional

A taxa de reconhecimento final, obtida pelo uso de um classificador em cascata, é semelhante a obtida com o conjunto de primitivas completo. Entretanto é possível observar uma significante redução em termos de complexidade computacional. Para

explicar melhor, considera-se a complexidade do classificador original, ou com conjunto completo de primitivas, como sendo:

$$\boxed{Compl_{OC} = ns \times nc} \quad \textbf{(5.1)}$$

onde,

ns: número de amostras;
nc: número de conexões do classificador MLP.

e, a complexidade de classificador em cascata é:

$$\boxed{Compl_{CC} = \sum_{i=1}^{NC} \left(ns_i \times nc_i \right)_I} \quad \textbf{(5.2)}$$

onde,

NC: número de módulos classificadores;
ns_i: número de amostras classificadas pelo i^{th} classificador;
nc_i: número de conexões do i^{th} classificador.

Os valores para $Compl_{oc}$ e $Compl_{cc}$ são 427.263.800 e 57.532.586, respectivamente. Isto indica uma redução de 86,54% em termos de complexidade computacional na tarefa de classificar 30.089 amostras de dígitos.

5.5 Considerações Finais

Uma atenção especial foi dedicada as etapas iniciais dos experimentos, como forma de garantir a confiabilidade dos resultados obtidos, uma vez que os experimentos estão parcialmente interligados. A observação dos resultados parciais permitiu ajustes nas otimizações propostas até o ponto de permitir criar um subproduto do método, que é o classificador em cascata.

Capítulo

6 Conclusões

O trabalho apresentou inicialmente o método de busca de soluções ótimas conhecido como Subida de Encosta em sua forma tradicional, com suas limitações também conhecidas. Utilizando como critério avaliar o mesmo número de iterações que o algoritmo otimizado, identificou um número 4 vezes maior de máximos locais, porém todos eles obtidos muito próximos, ou seja, com o conjunto de primitivas muito semelhante. Outra característica do algoritmo tradicional é a falta de um critério para a seleção da primitiva a ser removida, permitindo que primitivas importantes para o processo de classificação sejam removidas logo no início do funcionamento do algoritmo impedindo a obtenção de máximos locais de boa qualidade.

Foram adicionadas modificações e otimizações ao método tradicional, que neste trabalho passou a ser chamado de Método de Subida de Encosta Otimizado. Estas otimizações permitiram obter resultados melhores, em relação ao número de primitivas eliminadas, com menor variação entre os resultados parciais obtidos. O método proposto se mostrou uma interessante estratégia para a implementação de uma abordagem *Wrapper*. O fato de evitar o retreinamento em diversas etapas do método permite trabalhar com bases de dados extensas e com número elevado de primitivas.

Este trabalho de pesquisa utilizou técnicas conhecidas como auxiliar no processo de otimização:

a) Determinação do número de nós da camada escondida: A utilização de uma regra empírica para serviu como ponto de partida do ajuste deste item da topologia da rede neural. Apesar do resultado inicialmente obtido ficar distante do valor final, é uma forma automática de determinar um dos parâmetros da topologia da rede a ser utilizada;

b) Seleção aleatória de primitivas: conceito mantido em todas os experimentos. Permite ao algoritmo explorar melhor o espaço de soluções ajudando a evitar que a pesquisa ocorra sobre os mesmos locais;

Contribuições originais apresentadas neste trabalho:

a) Utilização do conceito de relevância inicial das primitivas é uma técnica já utilizada, porém associá-la ao conceito de prioridade de remoção consiste de uma contribuição original deste trabalho. Dificultando, porém sem impedir, a seleção de determinadas primitivas permite ao algoritmo remover um maior número de primitivas irrelevantes, ao processo de reconhecimento, logo nas primeiras iterações. A seleção do número de níveis de prioridade apresentou um número ótimo para este parâmetro do algoritmo;
b) Após a identificação de um máximo local o algoritmo efetua um retorno ao conjunto inicial de primitivas para a utilização de uma nova semente no início da busca de um novo máximo local. O tempo de busca não necessariamente reduziu, porém permitiu que a busca fosse realizada em diferentes regiões do espaço de soluções, testando subconjuntos diferentes uns dos outros em pelo menos uma primitiva;
c) De posse do conjunto de máximos locais, foram utilizadas diferentes formas de determinação do melhor conjunto e a taxa final de classificação obtida foi praticamente a mesma, indicando qualidade dos máximos locais obtidos com o algoritmo apresentado;
d) A partir do conjunto de máximos locais, o uso de tolerância na taxa do erro de classificação permitiu explorar melhor estas regiões, agora com o objetivo de redução do número de primitivas dos subconjuntos obtidos. Foi possível observar uma crescente especialização dos resultados em relação ao conjunto de Validação 2, com conseqüente aumento da importância do conjunto de Validação 3 no processo de identificação de um conjunto final de primitivas com boa generalização;
e) O conceito de tolerância na taxa de erro não é aplicado diretamente sobre os valores finais de classificação, portanto permitir uma taxa de erro de até 8% sobre o conjunto de Validação 2 não implica que a taxa final de reconhecimento

piore em 8%, pois o subconjunto de primitivas selecionado é submetido a novo processo de treinamento do classificador. Neste caso os experimentos indicaram uma piora de 1,67% para a taxa de erro de classificação final;

f) Os resultados obtidos junto aos experimentos de tolerância foram transformados em módulos de um classificador em cascata que apresentou desempenho equivalente ao classificador único, porém com redução da complexidade computacional em até 86,54%;

Em relação aos resultados dos experimentos:

a) Apresentou redução de até 42% do tamanho do conjunto de primitivas, de 132 para 77, usadas na tarefa de classificação de dígitos manuscritos, com variação de 0,16% na taxa de reconhecimento.
b) Utilizando-se do conceito de tolerância, atingiu a expressiva redução de 80% do tamanho do conjunto de primitivas, de 132 para 26, com redução da taxa de classificação de 99,10% para 97,43%, ou seja, 1,67% menor.
c) Não foram exigidos recursos especiais de hardware tornando possível sua utilização em problemas reais.

Os trabalhos aqui apresentados foram publicados nos seguintes congressos internacionais: Statistical Pattern Recognition SPR2004 – IAPR e International Workshop on Frontiers in Handwriting Recognition IWFHR2004, respectivamente nas referências [NUN04a, NUN04b].

A experimentação permitiu que o problema inicialmente proposto recebesse novo ponto de observação. Dificuldades e objetivos foram identificados, alguns foram devidamente experimentados e já se encontram descritos, outros farão parte de trabalhos futuros. Não se propõe um encerramento das atividades de pesquisa, mas uma fonte de embasamento que permita avaliar os novos objetivos.

A continuação deste trabalho deve incluir:

a) Avaliação de diferentes bases de dados que, por exemplo, possuam um número ainda maior de primitivas a serem avaliadas e um número de exemplares restrito. A estabilidade das taxas de reconhecimento obtidas junto aos máximos locais e o comportamento, quando da utilização do conceito de tolerância, indica que o

método proposto pode obter sucesso ao trabalhar com bases que disponham de um número restrito de exemplares por classe.

b) Avaliação da utilização das otimizações propostas junto a outros métodos de busca poderia demonstrar a influência que a escolha do método Subida de Encosta, baseada em experimentos apresentados em outras publicações, teve sobre os resultados obtidos.

c) Utilização de técnicas de seleção de exemplares, associadas a classificadores em cascata semelhante à apresentada neste trabalho. O número de exemplares da base de dados aqui utilizada e a seleção automática de exemplares para cada um dos módulos do classificador em cascata podem permitir que a especialização atingida em cada um dos módulos seja benéfica para os índices de classificação global.

Referências

[AHA94] AHA, D. W.; BANKERT R. L., *Feature Selection for Case-Based Classification of Cloud Types: An Empirical Comparison,* AAAI-94, Workshop on Case Based Reasoning, Technical Report WS-94-01, 1994

[AMD04] ADVANCED MICRO DEVICES, Inc. *Cray, Inc. Adopts Upcoming AMD Opteron™ Processor* Disponível em http://www.amd.com/gb-uk/Corporate/VirtualPressRoom/0,,51_104_543_8001~56268,00.html acesso feito em 18/05/2004, publicado em 21/Outubro/2002

[BAU88] BAUM, E. B. ; HAUSSLER, D. *What size net gives valid generalization?*, Neural Computation, vol. 1, pp. 151-160, 1988

[BLU97] BLUM, A. L.; LANGLEV, P. *Selection of Relevant and Examples in Machine Learning,* Special issue of Artificial Intelligence – Relevance, 97, vol. 97, no. (1-2), pp. 245-271, 1997

[BOZ02] BOZ, O. *Feature Subset Selection by Using Sorted Feature Relevance*, Proceedings of the 2002 International Conference on Machine Learning and Applications – ICMLA, CSREA Press, pp. 147-153, 2002

[BRI02] BRITTO, A. S. *A two-stage HMM-Based Method for Recognizing Handwritten Numeral Strings,* Tese de Doutorado, Pontifícia Universidade Católica do Paraná, PUCPR, 2002

[DAS97] DASH, M.; LUI, H., *Feature Selection for Classification*, IDA Intelligent Data Analysis, vol. 1(3), pp. 131-156, 1997

[DEV82] DEVIJVER, P.A.; KITTLER, J. *Pattern Recognition, a Statistical Approach*, Prentice Hall, Englewood Cliffs, London, 1982

[DUD73] DUDA, R. O.; HART, P. E. *Pattern Classification and Scene Analysis*, Stanford Research Institute, 1973

[FRA00] FREITAG, D.; KUSHMERICK, N. *Boosted Wrapper Induction,* Proc. American Association for Artificial Intelligence Conference, pp. 577-583, 2000

[FRE02] FREITAS, C. O. DE A. *Percepção Visual e Reconhecimento de Palavras Manuscritas.* Monografia apresentada para Concurso de Professor Titular da Pontifícia Universidade Católica do Paraná, PUCPR, 95p., 2002

[GAR92] GARRIS, M. D. *Design and Collection of a Handwriting Sample Image Database,* Advanced Systems Division National Institute of Standards and Technology, Social Science Computing Journal, Vol. 10, pp. 196-214, 1992

[GEM92] GEMAN, S., BIENENSTOCK, E.; DOURSAT, R. *Neural Networks and the Bias/Variance Dilemma*, Neural Computation, vol. 4, pp. 1-58, 1992

[HOR94] HORWOOD, E. *Machine Learning, Neural and Statistical Classification*, Editors: D. Michie, D. J. Spiegelhalter, C. C. Taylor, Series in Artificial Intelligence, disponível em http://www.amsta.leeds.ac.uk/%7Echarles/statlog/, 1994

[JAI00] JAIN, A. K.; DUIN, R. P. W.; MAO, J. *Statistical Pattern Recognition: A Review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22 no 1, pp. 4-37, January, 2000

[JAI97a] JAIN, A. K.; ZONGKER, D. *Representation and Recognition of Handwritten Digits Using Deformable Templates*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, no. 12, pp. 1386-1391, December, 1997

[JAI97b] JAIN, A.; ZONGKER, D. *Feature Selection: Evaluation, Application and Small Sample Performance,* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, num. 2, 1997

[JUS01] JUSTINO, E. J. R. *O grafismo e os modelos escondidos de Markov na verificação automática de Assinaturas,* Tese de Doutorado, Pontifícia Universidade Católica do Paraná, Programa de Pós-Graduação em Informática Aplicada, Disponível em http://www.ppgia.pucpr.br/~justino/, 131 p., Curitiba, 2001

[KOE03] KOERICH, A. L. *Unconstrained Handwritten Character Recognition Using Different Classification Strategies,* IAPR International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR 2003), Firenze, Italy, September 2003

[KOH97] KOHAVI, R.; JOHN, G. H. *Wrappers for Feature Subset Selection,* journal on Artificial Intelligence, vol. 97, num. 1-2, pp. 273-324, 1997

[LIU02] LIU, C.; SAKO, H.; FUJISAWA, H. *Performance Evaluation of Pattern Classifiers for Handwritten Character recognition*, IJDAR, International Journal on document Analysis and Recognition, Volume 4, Number 3, March 2002

[LOF03] LÖFFLER, K. *Arnold Pannartz and Konrad Sweinheim*, versão eletrônica The Catholic Encyclopedia, disponível em http://www.newadvent.org/cathen/ 11444b.htm, publicada em 2003,

acesso efetuado em 18/05/2004

[MEL99] MELLO, C. A. B.; LINS, R. D. *A Comparative Study On OCR Tools*, Vision Interface'99, Quebéc, Canadá, pp. 224-233, 1999

[MOL02] MOLINA, L. C.; BELANCHE, L.; NEBOT, À. *Feature Selection Algorithms: A Survey and Experimental Evaluation,* ICDM, International Conference of Data Mining, Japan, pp. 306-313, December, 2002

[NUN04a] NUNES, C. M.; BRITTO, A. S. JR.; KAESTNER, C. A. A.; SABOURIN, R. *Feature Subset Selection using an Optimized Hill Climbing Algorithm for Handwritten Character Recognition*, Statistical Pattern Recognition SPR2004 – IAPR, Lisboa, Portugal, 2004

[NUN04b] NUNES, C. M.; BRITTO, A. S.; KAESTNER, C. A. A.; SABOURIN, R. *An Optimized Hill Climbing Algorithm for Feature Subset Selection: Evaluation on Handwritten Character Recognition*, International Workshop on Frontiers in Handwritting Recognition IWFHR2004, Tóquio, Japão, 2004

[OLI01a] OLIVEIRA L. S.; SABOURIN R.; BORTOLOZZI F.; SUEN, C. Y. *A Modular System to Recognize Numerical Amounts on Brazilian Bank Checks,* 6th International Conference on Document Analysis and Recognition ICDAR, pp. 389-394, Seattle-USA, IEEE Computer Society Press, September, 2001

[OLI01b] OLIVEIRA, L. S.; BENAHMED, N.; SABOURIN, R.; BORTOLOZZI, F.; SUEN, C. Y. *Feature Subset Selection Using Genetic Algorithms for Handwritten Digit Recognition*, 14th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2001), pages 362-369, Florianópolis, Brazil, IEEE CS

Press, October 15-18, 2001

[OLI02a] OLIVEIRA, D. de P. R. *Sistemas de Informações Gerenciais: estratégicas, táticas, operacionais*. 8ª. Edição, Ed. Atlas, São Paulo, 2002

[OLI02c] OLIVEIRA, L. S.; SABOURIN, R.; BORTOLOZZI, F.; SUEN, C. Y. *Feature Selection Using Multi-Objective Genetic Algorithms for Handwritten Digit Recognition*, 16th International Conference on Pattern Recognition (ICPR 2002), pages 568-571, Vol. 1, Quebec City-Canada, IEEE CS Press, August 11-15, 2002

[OLI02c] OLIVEIRA, L. S.; SABOURIN R.; BORTOLOZZI F.; SUEN, C. Y. *Seleção de Primitivas usando algoritmos Genéticos Multi-Objetivos,* Proc. Int. Conf. on Pattern Recognition, vol. 1, 568-571, Quebec City, August, 2002

[PIR03] PIRES, C. *Antecedentes Históricos da Escrita*, revista eletrônica Contato, Revista Temas Portugal, Disponível em www.revista-temas.com/contacto/NewFiles/ Contacto12.html, publicado em 12/03/2003, acesso efetuado em 18/05/2004

[PRI00] PRINCIPE, J. C.; EULIANO N. R.; LEFEBVRE W. C. *Neural and Adaptive Systems,* John Wiley & Sons, Inc., New York, pg 656, 2000

[RAD03] RADTKE, P. V. W.; OLIVEIRA, L.S.; SABOURIN, R.; WONG, T. *Intelligent Zoning Design Using Multi-Objective Evolutionary Algorithms*, in the Proceedings of the 7th International Conference on Document Analysis and Recognition, ICDAR2003, pp.824-828, Edinburgh, Scotland, 3-6 August 2003

[RAM03] RAMAN, B.; IOEGER, T. R. *Enhancing Learning using Feature and Example Selection,* Submitted to Journal of Machine Learning

Research, January, 2003

[RAM03] RAMAN, B.; IOERGER, T. R. *Enhancing Learning using Feature and Example selection* – a ser publicado no Journal of Machine Learning Research – 2003

[RIC94] RICH, E.; KNIGHT, K., *Inteligência Artificial*, Segunda Edição, Editora Makron Books, 1994

[SAR04] SARLE, W. *How many hidden units should I use?,* Disponível em http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-10.html acesso feito em 18/05/2004

[SCH92] SCHALKOFF, R. J. *Pattern Recognition: Statistical, Structural and Neural Approaches*, Ed. John Wiley & Sons, Inc. 1992

[SKA94] SKALAK, D. B. *Prototype and Feature Selection by Sampling and Random Mutation Hill climbing Algorithms,* Proceedings of the Eleventh International Conference Machine Learning, ICML-94, pp. 293-301, New Brunswick, NJ: Morgan Kauffmann, 1994

[TRI96] TRIER, O. D.; JAIN, A. K.; TAXT, T. *Feature Extraction Methods for Character Recognition, A Survey*, Pattern Recognition 29, pp. 641-662, 1996