UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
AREA DE CONCENTRAÇÃO: PROCESSAMENTO DE IMAGEM

ADRIANO MENDES GIL

RECONHECIMENTO DE DÍGITOS MANUSCRITOS: BUSCA DE UM CLASSIFICADOR COM MÁXIMA TAXA DE ACERTO

MANAUS
2014

UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
AREA DE CONCENTRAÇÃO: PROCESSAMENTO DE IMAGEM

ADRIANO MENDES GIL

RECONHECIMENTO DE DÍGITOS MANUSCRITOS: BUSCA DE UM CLASSIFICADOR COM MÁXIMA TAXA DE ACERTO

Projeto de Qualificação apresentado ao Curso de Mestrado em Engenharia Elétrica, área de concentração de Controle e Automação de Sistemas do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Amazonas.

Orientador: Prof. Dr. Cícero Ferreira Fernandes Costa Filho
Co-Orientadora: Prof.ª Dra. Marly Guimarães Fernandes Costa

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Gil, Adriano Mendes
G463r Reconhecimento de dígitos manuscritos: busca de um classificador com máxima taxa de acerto / Adriano Mendes Gil. 2014
92 f.: il. color; 31 cm.

Orientador: Cícero Ferreira Fernandes Costa Filho
Coorientador: Marly Guimarães Fernandes Costa
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Amazonas.

1. Redes neurais . 2. Reconhecimento de padrões. 3. Dígitos manuscritos . 4. Máquina de vetores de suporte. I. Costa Filho, Cícero Ferreira Fernandes II. Universidade Federal do Amazonas III. Título

Dedico este trabalho a minha maior fonte de motivação e inspiração, minha filha,

Sofie Sánchez Gil

Agradecimentos

Ao Prof. Cícero Costa Filho, pelos seus conselhos valiosos, seu constante apoio e orientação ao longo do desenvolvimento desta pesquisa, e à Profa. Marly Guimarães Costa por incentivar meu interesse em processamento de imagens desde a graduação.

Ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da Universidade Federal do Amazonas e, em especial, ao Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação, CETELI, por disponibilizar a infraestrutura necessária para realização de minha pesquisa. À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, CAPES, pelo imprescindível apoio financeiro.

Ao meu pai, Prof. Antônio Gil, que de muitas formas apoiou e incentivou a conclusão da minha Dissertação.

A minha esposa Priscila Gil, pela ajuda e apoio nos momentos de dificuldade.

A todas as pessoas que direta ou indiretamente contribuíram de alguma forma para a conclusão deste trabalho.

Resumo

Sistemas de reconhecimento ótico de caracteres, também conhecidos como OCR, permitem identificar e reconhecer caracteres impressos por meio de imagens, uma funcionalidade já bem difundida em scanners, dispositivos móveis, entre outros. Existe uma crescente necessidade de reconhecimento de caracteres manuscritos para uso em várias situações, tais como reconhecimento de valores nominais em cheques de bancos, reconhecimento dos dígitos manuscritos de endereço postal para redirecionamento automatizado de cartas nos correios. Reconhecimento de dígitos manuscritos esbarra na dificuldade de lidar com uma grande variação intraclasse, devido a diferentes estilos de escrita, diferentes graus de inclinação dos caracteres. Este trabalho apresenta três estratégias utilizando três diferentes métodos de reconhecimento de padrões e dois métodos de extração de características. A primeira estratégia utilizou Descritores de Fourier e a técnica de transição de borda para extrair valores representativos do contorno dos caracteres e como camada de classificação utilizou uma rede neural MLP em associação com um conjunto de classificadores SVM para validar e corrigir eventuais erros da rede MLP. A segunda estratégia figurou como base comparativa para as demais estratégias por utilizar um algoritmo clássico de redes neurais convolutivas, LeNet5, e como características utilizou as próprias imagens dos dígitos. A terceira estratégia fez uso de um conjunto de classificadores SVM em uma árvore de decisão desbalanceada para a classificação dos dígitos a partir unicamente de suas imagens. Como resultados dos experimentos, a primeira estratégia provou não ser totalmente efetiva por obter resultados em torno de 80% de taxa de acerto. A segunda estratégia obteve 0,9% de taxa de erro que apesar de ter sido alta, ainda é muito menor se comparada com os melhores resultados obtidos na literatura. A terceira estratégia por sua vez logrou sucesso em reconhecer 100% das amostras de teste da base MNist de dígitos manuscritos, devido ao sucesso do treinamento de cada um dos classificadores SVM, que apesar de utilizarem uma enorme quantidade de vetores de suporte, atingiram individualmente 0% de taxa de erro.

Palavras-chave: redes neurais, reconhecimento de padrões, dígitos manuscritos, máquina de vetores de suporte.

Abstract

Optical character recognition system, aka OCR, allows identifying and recognizing printed characters from pictures. A wide range of devices already has such functionality, e.g, scanners and mobile devices. The current everyday tasks has an increasing demand for handwritten character recognition, for example, recognize specified amount on bank checks, identify postal address to automate some aspects of letter delivery. Handwritten digit recognition faces the difficulty of great intraclass variability, due to different writing stiles and different character slant degrees. This work presents three strategies to address handwritten digit recognition by means of three pattern recognition methods and two feature extraction algorithms. The first strategy makes use of Fourier Descriptor and Boundary Transition Technique to extract representative values from digits contours in order to recognize digits is used a neural network Multilayer Perceptron and a set of Support Vector Machines classifiers to validate neural network output. The second strategy represents this work's baseline using the classic convolutional neural networks algorithm from literature, LeNet5. Such algorithm received as input the raw digit images without preprocessing. The third strategy used a unbalanced decision tree in which support vector machines actuated as decision points and as representative feature received the raw digit images. Late experiments showed that first strategy was not effective enough to recognize digits; only about 80% of characters were successfully recognized. By means of Convolutional Neural Network was possible to achieve 0.9% of error rate, not so impressive if compared to literature best results. The third strategy was capable to recognize 100% of test samples from handwritten digits dataset of MNist. Each support vector machine classifier achieved 0% of error rate, due to an enormous amount of support vectors.

Keywords: neural network; pattern recognition; handwritten digits, support vector machines.

ÍNDICE DE ILUSTRAÇÕES

ÍNDICE DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Reconhecimento de caracteres nasceu da crescente demanda nas últimas décadas por rápida conversão de montanhas de informações impressas ou escritas à mão para o formato digital (CHERIET, KHARMA, *et al.*, 2007). Muito comumente estes dados existem no papel e precisam ser digitados em um computador por operadores humanos, por exemplo, bilhões de cartas nos correios que precisam ter o CEP reconhecido, cheques de banco onde o seu valor manuscrito deve ser identificado, entre outros. É um processo que consome muito tempo e com alta possibilidade de erros. Esta necessidade motivou a criação de sistemas *OCR* (Reconhecimento Ótico de Caracteres - *Optical Character Recognition*) que lêem dados escritos reconhecendo um caractere por vez.

Reconhecimento Ótico de Caracteres (*OCR*) é uma importante aplicação de reconhecimento de padrões. Existem muitos documentos de importância histórica, técnica e econômica que existem somente na forma impressa. Um sistema *OCR* pode ajudar a reduzir os custos de digitalização destes documentos. Existem muitas técnicas bem sucedidas de implementação de *OCR* que vêm sendo aplicadas em áreas como reconhecimento de texto manuscrito, reconhecimento de texto impresso de forma mecânica, e reconhecimento de notas musicais.

No campo do reconhecimento automático de caracteres existem dois modos de aquisição de dados: *online* e *offline* (CHERIET, KHARMA, *et al.*, 2007). O reconhecimento *online* diz respeito às informações capturadas durante o ato da escrita, tais como: velocidade, pressão, direção. O reconhecimento *offline* ocorre quando a entrada para o sistema é uma imagem do texto que foi capturada a partir de um scanner ou de uma câmera.

Reconhecimento de caracteres é um problema que traz grande dificuldade devido a enorme variação de estilos de escrita, ou seja, uma grande variação intra-classe, dado que o mesmo tipo de caractere pode ser escrito em diferentes tamanhos e ângulos de orientação.

Assim como existe uma grande variação entre diferentes caracteres, assim sofrendo também de uma grande variação inter-classe.

Um sistema de reconhecimento de caracteres, conforme mostrado na Figura **1**, divide-se nas seguintes fases:

- Aquisição de imagens: adquirir uma imagem colorida, ou em nível de cinza ou em formato binário
- Pré-processamento: aplicar técnicas de processamento de imagens visando aumentar as taxas de reconhecimento de caracteres. Também é feita a conversão das imagens adquiridas para o formato binário. Neste ponto a imagem é uma somatória dos pontos do plano de fundo com a região de interesse.
- Análise de layout: entender a estrutura textual da imagem. É uma etapa importante em sistemas que levam em consideração a disposição do texto como característica semântica, ou seja, que impacta na maneira como o texto será interpretado. Este trabalho não contempla a etapa de análise de layout, por focar em dígitos manuscritos isolados.
- Segmentação: Os caracteres são retirados das imagens e isolados em imagens separadas para reconhecimento de cada um.
- Classificação: utiliza reconhecimento de padrões para classificar as amostras encontradas nos caracteres corretos, também pode incluir análise de sintaxe.
- Pós-processamento: reúne todos os resultados de classificação dos caracteres para a reconstituição das palavras dos quais foram isolados.

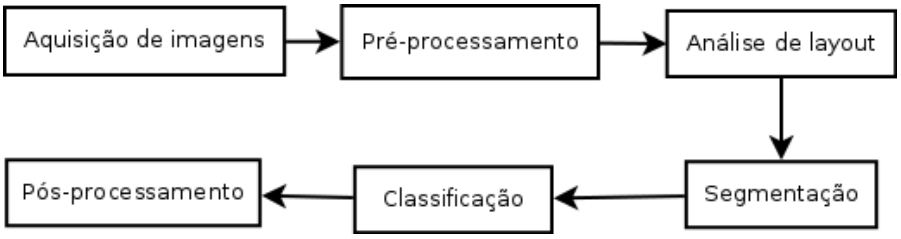

Figura 1 – Sistema de reconhecimento de caracteres

Existem vários trabalhos já publicados para o reconhecimento de dígitos manuscritos, tais como o uso de um classificador *SVM* com kernel polinomial de grau 9, realizado por Decoste e Schölkopf (2002) que obteve um resultado de 0,56% de erro na base MNIST de dígitos manuscritos. Em Ciresan, Meier, *et al.* (2011) um conjunto de redes convolutivas atingiu 0,35% de taxa de erro, taxa melhorada em Ciresan, Meier e Schmidhuber (2012), com expressivo resultado de 0,23%, o melhor resultado atual da literatura. Embora valores de taxas de acerto muito elevadas já tenham sido obtidas na literatura para a classificação de dígitos, esse trabalho centra o seu esforço na busca de uma otimização ainda maior desses valores.

Nesse sentido, duas estratégias foram utilizadas:

1) Procurou-se em primeiro lugar utilizar diferentes dados de entrada para os classificadores. Em pesquisas realizadas na literatura observou-se o emprego de dois grupos de dados de entrada: o primeiro grupo constituído de características extraídas dos dígitos em Cho (1997), Travieso, Alonso e Ferrer (2011), Chung e Wong (1997) e o outro grupo constituído da própria imagem do dígito em Ciresan, Meier e Schmidhuber (2012), LeCun, Bottou, *et al.* (1998). Dentre as características, as mais utilizadas estão os Descritores de Fourier. Dessa forma, procurou-se comparar o desempenho desses dois grupos de dados de entrada na classificação de dígitos, procurando identificar aquele que resultava em valores ótimos para a taxa de acerto.
2) Procurou-se em segundo lugar utilizar-se vários classificadores. Nesse trabalho três métodos de classificação foram empregados: redes neurais treinadas com o

algoritmo de backpropagation, redes convolutivas e Máquinas de Vetores de Suporte. Para esses métodos, os melhores resultados obtidos com a base MNIST que foi utilizada nesse trabalho consistiram em: 0,23% através de um conjunto de 35 classificadores convolutivos (CIRESAN, MEIER e SCHMIDHUBER, 2012), 0,27% utilizando 12 classificadores convolutivos (CIRESAN, MEIER, *et al.*, 2011) e 0,56% utilizando um classificador *SVM* de kernel polinomial de grau 9 (DECOSTE e SCHÖLKOPF, 2002).

Conforme pode-se depreender dos valores citados no parágrafo anterior, a tarefa de melhorar as taxas de acerto é uma tarefa de grande dificuldade na medida em que altos valores já foram obtidos. A determinação de se alcançar um método sem erros, com 100% de taxa de acerto, parecia-nos, no entanto, uma meta a ser alcançada, garantindo absoluta confiabilidade ao classificador de dígitos manuscritos.

Outra observação pertinente em relação as informações sobre taxas de acerto mostradas anteriormente é que os melhores resultados foram obtidos com redes convolutivas. Conforme será mostrado no decorrer do texto, no entanto, nesse trabalho a utilização de classificadores SVM associados a imagem como dado de entrada, possibilitou a obtenção de resultados ótimos para a taxa de acerto na classificação de dígitos manuscritos. No trabalho ora proposto a utilização de SVM, no entanto, se deu de forma diferente daquela empregada por outros trabalhos anteriores.

Na literatura, a utilização de SVM para a classificação de dígitos manuscritos pode ser encontrada no seguinte trabalho: em Decoste e Schölkopf (2002), um classificador *SVM* obtém na base MNIST uma taxa de acerto de 0,56%. Nesse trabalho, *SVM* é utilizado em contexto multiclasse em conjunto com a técnica de vetores de suportes virtuais, onde exemplos virtuais são gerados a partir dos vetores de suporte.

No trabalho ora apresentado utilizou-se classificadores SVM binários associados em uma árvore de decisão desbalanceada. Árvores são estruturas de dados compostas de unidades chamadas de nós que podem conter referências a outros nós, de maneira hierárquica, formando assim uma forma simbólica de árvore. Logo, árvores definem uma relação entre dados, de tal maneira que cada exemplar dessa estrutura se liga a apenas um único outro nó de hierarquia superior, nomeado como “nó pai”, e a um número ilimitado de nós de hierarquia inferior, os nós “filhos”. Nós sem filhos são chamados de “nós folhas”.

A forma como os pontos de decisões se distribuem na árvore é que determina se está ou não balanceada. Caso seus pontos estejam uniformemente distribuídos entre todas as ramificações, então é declarada como sendo uma árvore balanceada. Caso contrário, caso seus pontos de tomada de decisão estejam dispostos em determinadas ramificações mais do que em outras, ou seja, é definida como desbalanceada. A Figura 2 exemplifica esse conceito de balanceamento. Na imagem da esquerda, todos os nós com exceção dos nós folhas possuem o mesmo número de filhos, ou seja, é considerada balanceada. Enquanto na imagem da direita, os nós da árvore possuem diferentes números de filhos.

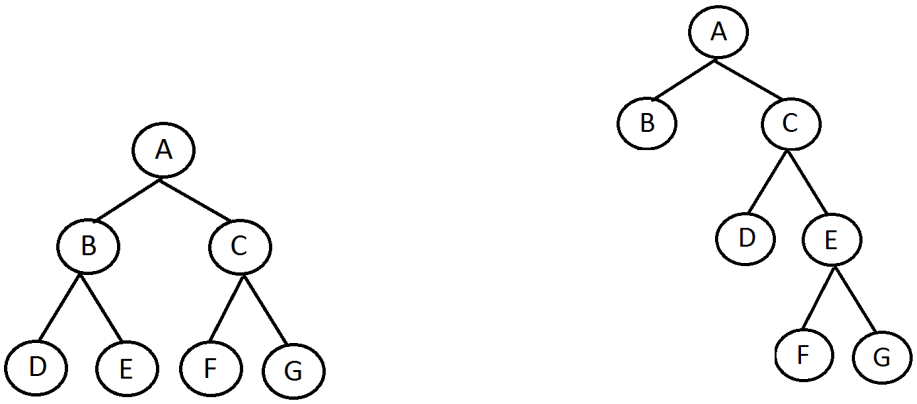

Figura 2 – Exemplos de árvores. (a) árvore balanceada, (b) desbalanceada.

Dentro da teoria de decisão, e até os anos oitenta, as árvores de decisão eram as ferramentas principais para representar problemas de decisão simples. Uma árvore de decisão

realiza a classificação por meio da segmentação do universo problema em diferentes conjuntos de decisão. Possui uma estrutura de ramificações hierárquicas, permitindo que diferentes condições de decisão sejam avaliadas de maneira sequencial ou paralela.

Com esse cenário em mente, são traçados a seguir os objetivos dessa dissertação:

2 OBJETIVOS

2.1 Geral

O objetivo geral deste trabalho de pesquisa é obter um valor máximo para a taxa de acerto da classificação de dígitos manuscritos, explorando a associação entre dados de entrada e método de classificação.

2.2 Específicos

Os objetivos específicos consistem em:

- Identificar um conjunto de dados de entrada para um classificador de dígitos manuscritos, que possibilite obter uma taxa de acerto máxima na classificação, utilizando uma base de dados de dígitos manuscritos consolidada e avaliada na literatura;
- Identificar um classificador de dígitos manuscritos que possibilite obter uma taxa de acerto máxima na classificação, utilizando uma base de dados de dígitos manuscritos consolidada e avaliada na literatura.
- Avaliar o uso de redes neurais de propagação direta (*feedforward)* treinadas com o método de retropropagação para diferenciação de dígitos manuscritos, avaliando o seu desempenho utilizando como entrada, de forma separada, dois tipos de informação: um conjunto com descritores de Fourier e imagens sem extração de características;
- Avaliar o uso de redes convolutivas para diferenciar os dígitos manuscritos em imagens sem extração de características;
- Avaliar a identificação de dígitos manuscritos em imagens sem extração de características por meio de uma árvore de decisão desbalanceada utilizando como pontos de decisão classificadores SVM binários.

2.3 Estrutura do trabalho

A divisão deste trabalho é feita em cinco capítulos, discorrendo acerca da pesquisa que foi realizada neste trabalho de dissertação. O Capítulo 1 apresenta uma breve introdução ao tema de reconhecimento de dígitos manuscritos e sua caracterização como problema na área de reconhecimento de padrões. O Capítulo 2 aborda os objetivos gerais e específicos propostos para a dissertação e descreve, de forma resumida, a estrutura do trabalho. O Capítulo 3 apresenta alguns trabalhos que contribuíram para a realização desta pesquisa e a revisão bibliográfica de artigos com temas relacionados à metodologia de sistemas de reconhecimento de caracteres. O Capítulo 4 inclui a fundamentação teórica, apresentado os conceitos básicos de remoção de padrões indesejados (*outliers*), normalização dos dados, a definição de Descritores de Fourier e do algoritmo de treinamento de redes neurais intitulado *backpropagation*, assim como a definição de redes convolutivas e de árvores de decisão. O Capítulo 5 aborda os materiais usados no desenvolvimento do trabalho e a metodologia que será utilizada para alcançar os objetivos propostos. O Capítulo 6 apresenta os resultados obtidos durante o mestrado. O Capítulo 7 realiza algumas conclusões acerca dos resultados e metodologia proposta.

3 REVISÃO BIBLIOGRÁFICA

Um sistema de reconhecimento de caracteres pode ser diferenciado segundo a base de amostras utilizada, os tipos de características que são extraídas, algoritmo de classificação e resultados obtidos, tal como evidenciado na Tabela **1**. Nas seções a seguir os artigos revisados serão abordados em função dos materiais, das características, dos classificadores e resultados obtidos. Procurou-se evidenciar quais trabalhos influenciaram na escolha e desenvolvimento da proposta dessa pesquisa.

Tabela 1 – Tabela comparativa dos trabalhos utilizados como referência

Ano	***Autor***	***Título***	***Materiais***	***Características***	***Classificador***	***Taxa de Acerto***
1992	A. D. Mandalia, A. S. Pandya e R. Sudhakar	A hybrid approach to recognize handwritten alphanumeric characters	Não diz	Transformada de Hough	Rede Neural MLP juntamente com a teoria Dempster-Shafer para lidar com a incerteza dos dados	Não diz
1992	Gary M. T. Man e Joe C. H. Poon	An enhanced approach to character recognition by Fourier descriptor	Base própria	Descritores de Fourier	Casamento de modelos previamente definidos e Rede Neural MLP treinada com o algoritmo *backpropagation*	90%
1997	Yuk Ying Chung e Man To Wong	Handwritten Character Recognition by Fourier Descriptors and Neural Network	Base própria	Descritores de Fourier e Técnica de transição de borda	Rede Neural MLP utilizando algoritmo de treinamento backpropagation	96%
1999	Mohamed Masmoudi, MounirSamet, FaycalTaktak e Adel M. Alimi	A hardware implementation of neural network for the recognition of printed numerals	Base própria	A própria imagem sem alterações	Rede Neural treinada com o algoritmo *backpropagation*	90%

2002	Il-Seok Oh e Ching Y. Suen	A class-modular feedforward neural network for handwriting recognition	Base CENPARMI de dígitos manuscritos	Distribuição de distância direcional	Redes Neurais Modulares	97,30%
2004	Ernst Kussul e Tatiana Baidyk	Improved method of handwritten digit recognition tested on MNIST database	Base de dados MNist de dígitos manuscritos	A própria imagem sem alterações	Classificador baseado no perceptron de Rosenblatt	99,37%
2006	VelappaGanapathy e Charles C. H. Lean	Optical Character Recognition Program for Images of Printed Text using a Neural Network	Base própria	49 características relativas a propriedades da imagem	Rede Neural Auto-Organizável (SOM)	81%
2011	Carlos M. Travieso, Jesús B. Alonso e Miguel A. Ferrer	Combining different off-line handwritten character recognizers	Base NIST-SD19	Máscara de Kirsch e Descritores elípticos de Fourier	Combinação de três classificadores SVM	98,55%
2011	Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella e Jürgen Schmidhuber	Convolutional Neural Network Committees for Handwritten Character Classification	Base MNist de dígitos manuscritos	A própria imagem sem alterações	Combinação de classificadores convolutivos	99,73%
2012	Dan Ciresa, Ueli Meier e Jürgen Schmidhuber	Multi-column Deep Neural Networks for Image Classification	Base MNist de dígitos manuscritos	A própria sem extração de características	Combinação de classificadores convolutivos	99,77%

3.1 Materiais

Conforme mostrado na Tabela **1**, as principais bases de dados utilizadas para o treinamento e teste de sistemas de reconhecimento de caracteres são: NIST-SD19 e MNIST de dígitos manuscritos. Os trabalhos que utilizam uma base própria são de pouca utilidade,

pois não permitem que sejam estabelecidas comparações de desempenho com os algoritmos apresentados nos mesmos.

A base NIST *Special Database* 19 (SD 19) contém imagens binárias de 3.699 amostras de formulários manuscritos, totalizando 814.255 dígitos e caracteres alfanuméricos manuscritos segmentados. Cada caractere segmentado possui um tamanho de 128x128 pixels e estão rotulados como pertencentes a uma entre as 62 classes possíveis.

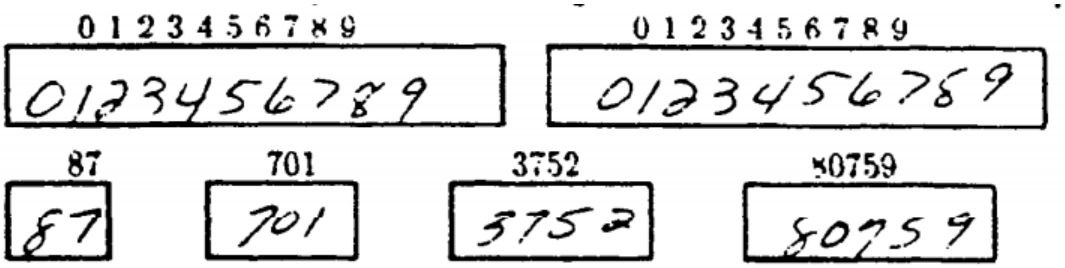

Figura 3 – Exemplo de amostras da base NIST-SD19

A base MNIST de dígitos manuscritos possui um conjunto de 60.000 amostras de treinamento e um conjunto de teste com 10.000 amostras. Cada amostra possui seu tamanho normalizado para 20x20 *pixels* centradas em um espaço de 28x28 *pixels*.

Figura 4 – Exemplo de amostras da base MNIST

Por ser mais utilizada, a base de dados do MNIST se destaca entre artigos de aprendizagem de máquina, o que a torna assim uma ótima referência para comparação de desempenho na proposta de novos algoritmos nessa área.

Um total de 68 classificadores são utilizados na tabela comparativa no web site da base de dados MNIST (LECUN, CORTES e BURGES, 1998), dentre eles, pode-se citar as seguintes categorias:

- Máquinas de vetor de suporte (SVM)

- Redes neurais (sem estrutura convolutiva)
- Redes convolutivas

Classificadores neurais tendem a possuir um desempenho significantemente melhor que outros tipos de classificadores nessa base. Algoritmos convolutivos lideram os registros das melhores taxas de classificação. O trabalho mais bem sucedido utiliza uma associação de redes convolutivas junto com o aumento da base de treinamento utilizando distorções elásticas, tendo obtido uma taxa de erro de 0,27%.

Buscando na literatura pelo uso de apenas um único classificador convolutivo, a melhor taxa de erro de classificação é de 0,35% (CIRESAN, MEIER, *et al.*, 2011). O trabalho de Deng (2012) analisa que o uso de distorções no aumento do conjunto de treinamento é necessário para se alcançar taxas de erro extremamente baixas. Pelos trabalhos que não utilizam essas distorções, conclui-se que a taxa de erro sobe de 0,35% para 0,53% (JARRETT, KAVUKCUOGLU, *et al.*).

Outras técnicas também são utilizadas, tais como k vizinhos mais próximos (*KNN*) e *SVM*. Segundo a avaliação de Deng (2012), para essas técnicas seria necessário uma etapa refinada de pré-processamento para obterem taxas de classificação relevantes.

Dada a grande quantidade de trabalhos publicados utilizando a base MNIST, o que possibilita comparar resultados obtidos, a base MNIST foi adotada como a base de dados para realização dos experimentos desta pesquisa.

3.2 Características

Uma das primeiras fases no projeto de um sistema de reconhecimento de caracteres é a definição de quais características serão utilizadas para representar os dados de interesse. Na Tabela **1** é perceptível que as principais características utilizadas na literatura para esse fim

foram: descritores de Fourier, a própria imagem, transformada de Hough, máscara de Kirsch e técnica de transição de borda.

Com o objetivo de comparar essas características foi elaborada a Tabela 2, onde são listadas as vantagens e desvantagens das mesmas.

Tabela 2 - Resumo das Características utilizadas na literatura

Características	Vantagens	Desvantagens
Descritores de Fourier	Invariantes a mudanças de deslocamento e translação	Sensíveis a pequenas mudanças nas bordas dos contornos
A própria imagem sem alterações	Preserva todas as características da imagem	Espaço de alta dimensionalidade, por exemplo, uma imagem 28x28 gera um espaço de característica de tamanho 784.
Transformada de Hough	Acurácia em detectar segmentos de linhas e curvas	Alto custo computacional
Máscara de Kirsch	Extrai informações pela inclinação dos caracteres	Grande variabilidade nos dados
Técnica de transição de borda	Utiliza propriedades topológicas para identificar indivíduos que são confundidos ao usar somente Descritores de Fourier, e.g, os dígitos '6' e '9' são confundidos devido ao fato de Descritores de Fourier serem invariantes a mudanças de rotação	Não é suficiente para garantir o reconhecimento dos caracteres, por não ser representativo no sentido de descriminar todas as classes. Seu papel é de complementar os Descritores de Fourier de maneira a contornar seu problema com amostras similares mas com ângulos de rotação diferente.

Nesse trabalho, as principais características utilizadas foram os descritores de Fourier com características topológicas e o uso das imagens puras sem pré-processamento. Dessa forma, será analisado com mais detalhes a seguir os trabalhos que utilizaram essas características.

3.2.1 Imagens Puras Como Vetor de Características

Muitos trabalhos de sucesso utilizam imagens sem pré-processamento como vetor de características para a etapa de classificação. Um exemplo é o trabalho de Masmoudi, Samet, *et al.* (2000), onde uma rede neural *backpropagation* é utilizada para implementação via hardware do reconhecimento de dígitos numerais impressos. Em Kussul e Baidyk (2004) utilizando uma base própria e fazendo uso das imagens dos dígitos puras, obteve 90% de acerto com uma arquitetura neural 35-10-10.

Utilizando três camadas de neurônios chamadas de camadas Sensor, Associativa e de Saída, o classificador LIRA (*Limited Receptive Area* - Área Limitada de Percepção) (KUSSUL e BAIDYK, 2004) conseguiu se sobressair com o resultado de 0,59% na base MNIST, através de uma variante desse classificador voltada para o reconhecimento em imagens em escala de cinza, o LIRA_*grayscale*. Também é efetuada a operação de normalização do tamanho das imagens para um tamanho padrão de 20 x 20. Devido a interpolação de cada imagem neste processo, as imagens outrora binárias são convertidas em escala de cinza. Em (KUSSUL e BAIDYK, 2004) são propostos dois classificadores, um para imagens binárias, o LIRA_*binary*, com taxa de erro de 0,63% e outro para imagens em escala de cinza, o LIRA_grayscale, que obteve taxa de erro de 0,59%, sendo ambos testados na base MNIST.

De forma a evitar que o classificador se especialize nas amostras de treinamento, o que prejudicaria a sua capacidade de generalização, o trabalho de Ciresan, Meier *et al.* (2011) utiliza uma série de deformações nas imagens do conjunto de treinamento de forma contínua a cada época do algoritmo de treinamento. Em adição a esse método, também são normalizados os tamanhos do caracteres para 6 diferentes conjuntos com os seguintes valores de largura: 10, 12, 14, 16, 18 e 20 pixels. Para cada base pré-processada em conjunto com a base original é criado um classificador convolutivo específico. Assim, realizando a média da saída dos sete

classificadores, foi possível obter o resultado de 0,27% de taxa de erro na base MNIST de dígitos manuscritos.

3.2.2 Descritores de Fourier

Os Descritores de Fourier são características únicas na representação e classificação de contornos, pois realçam a descrição de informações locais e distingue contornos similares. Segundo Poon e Man (1992), Descritores de Fourier são muito úteis no reconhecimento de formas, além de serem invariantes a transformações afins e a transformações que envolvem deslocamentos, e também tornam possível a partir dos próprios descritores de Fourier reconstruir a forma original completamente. Uma desvantagem é que, embora sejam característica globais, podem ser afetados por pequenas distorções nas bordas dos dígitos.

De acordo com Poon e Man (1992) é possível compensar os pontos fracos dos descritores de Fourier através de características topológicas ou assinaturas de curva. No presente trabalho será utilizada a técnica de transição de borda, como característica topológica, com intuito de complementar os descritores de Fourier. A técnica de transição de borda é uma das características recomendadas em Chung e Wong (1997) e em Dedgaonkar, Chandavale e Sapkal (2012) para evitar a confusão entre dígitos semelhantes.

Em Poon e Man (1992) é utilizada uma representação de objetos por um conjunto de Descritores de Fourier. Cada elemento desse conjunto refere-se a diferentes porcentagens do contorno total do objeto a ser reconhecido. A abordagem apresentada no artigo (MORNS e DLAY, 1996) define uma rede neural de propagação direta com supervisão dinâmica, cujas características de entrada são descritores de Fourier. Os autores alegam atingir taxas de acerto de 97.46%, o que sustenta a hipótese de boa representação deste tipo de característica.

Diferentes tipos de classificadores são utilizados na literatura em conjunto com descritores de Fourier, como rede neural treinada com o algoritmo *backpropagation* nos

trabalhos de Morns e Dlay (1996) e Chung e Wong (1997), redes SOM em Cho (1997) e conjuntos de classificadores SVM em Travieso, Alonso e Ferrer (2011).

Em Chung e Wong (1997) é citada a utilização dos descritores invariantes de Fourier em conjunto com uma rede neural treinada com o algoritmo *backpropagation*. Porém não é evidenciado no artigo quais as metodologias de treinamento e teste utilizadas, além de não especificar qual base de dados foi escolhida.

No trabalho de Cho (1997) são extraídos 15 descritores complexos de Fourier e também são usadas técnicas de compressão de imagem. Os vetores de características resultantes são passados para a etapa de classificação. Testes foram realizados em uma base de dígitos manuscritos da Universidade de Concordia em Montreal. O classificador de estrutura adaptativa de mapas auto-organizáveis (*SOM*) obteve 96,05%, enquanto um conjunto de classificadores *MLP* atingiu 97,69%.

No artigo de Travieso, Alonso e Ferrer (2011) é proposta uma combinação de classificadores *SVM* utilizando diferentes vetores de características, dentre eles foram extraídos os Descritores Elípticos de Fourier com correção de inclinação baseada em momentos de Zernike. Particularmente o classificador que utilizou esses descritores obtive o segundo melhor desempenho individual, de 7,38% de taxa de erro, enquanto os outros alcançaram respectivamente, 7% e 9,8%. Essa diferença de desempenho entre os classificadores pode ser um indicativo da boa capacidade representativa dos descritores de Fourier. Experimentos ao longo deste trabalho tentarão validar essa hipótese.

3.3 Classificadores

O papel do classificador é dividir o espaço de características em regiões que correspondem as classes do problema em análise (THEODORIDIS e KOUTROUMBAS,

2009). A escolha do classificador é uma das etapas mais importantes no projeto de um sistema de reconhecimento de padrões.

3.3.1 Redes Neurais treinadas com Algoritmo *Backpropagation*

O artigo de Masmoudi, Samet, *et al.* (2000) trabalha com o reconhecimento de numerais impressos, uma tarefa necessária para a interpretação de endereços postais, e para a determinação dos valores de um cheque bancário, por exemplo. Este trabalho trata da aplicação de um circuito de rede neural para o problema de reconhecimento de numerais. Foi utilizada uma rede neural de arquitetura 35-10-10 que, em uma base própria de 100 amostras, obteve uma taxa de 90% de acerto.

O trabalho de Poon e Man (1992) utiliza um classificador multi-categoria, que fornece os padrões de entrada para uma rede neural de três camadas treinada com algoritmo *backpropagation*. Testado em uma base própria, o algoritmo teve uma taxa de acerto de 96,5% nas amostras de treinamento e 90% nas amostras de teste.

Em Cho (1997) é treinado um conjunto de classificadores *MLP* utilizando uma base de dados da Universidade de Concordia com amostras coletadas do serviço postal americano, utilizando lógica fuzzy como componente de decisão entre os diferentes classificadores foi possível obter 97,69% de taxa de acerto.

3.3.2 Redes Neurais Modulares

O conceito de arquitetura neural modular foi desenvolvido em Jordan e Jacobs (1991). Essa arquitetura inicial possui muitas redes especialistas e uma rede de chaveamento. Uma única tarefa é então decomposta em múltiplas subtarefas e cada subtarefa é alocada a uma rede especialista. A rede de chaveamento controla a saída final após receber a saída de cada uma das redes especialistas.

Em Hrycej (1992) é apresentado o aprendizado modular de redes neurais. O autor notou o fato que uma rede neural é comumente vista como uma caixa preta desestruturada com uma capacidade de aprendizado. Quanto maior o tamanho da rede maior a sua complexidade. Baseando-se em suas características, pode ser feita uma tentativa de modularização da rede em vários aspectos, por exemplo, decomposição em partes não-supervisionadas e supervisionadas, ou partes lineares e não-lineares. Para demonstrar que a modularidade pode alavancar o desempenho do aprendizado, experimentos de reconhecimento são discutidos utilizando dados de uma desordem na tireóide e dígitos manuscritos.

No artigo de Oh e Suen (2002) é proposto uma arquitetura de *Perceptron* multicamada de alimentação direta com modularização de classe para o problema de reconhecimento de caracteres. São gerados K classificadores binários, treinados independentemente, ou seja, cada classificador é treinado para distinguir entre uma das K classes em detrimento do restante. O erro de propagação reversa é aplicado para cada um dos classificadores binários da mesma maneira que uma rede neural convencional. Sua arquitetura, tal como mostra a Figura 5, foi testada e comparada em relação à abordagem convencional através de quatro diferentes conjuntos de caracteres: dígitos (10 classes), letras maiúsculas (26 classes), dígitos em pares com pontos em comum (100 classes) e caracteres coreanos (352 classes). Todos os testes tiveram resultados melhores que suas contrapartidas de abordagem convencional.

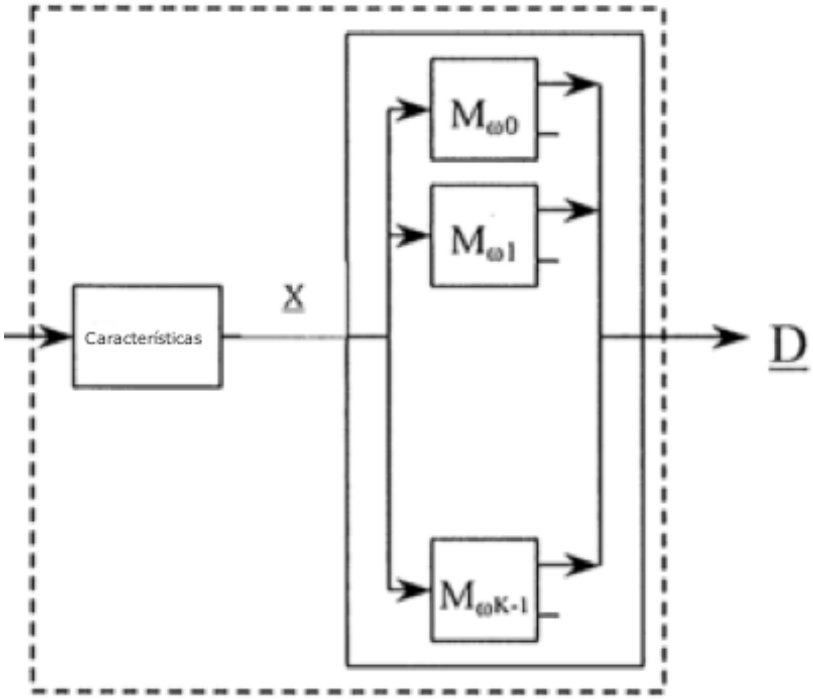

Figura 5 – Arquitetura modular definida em Oh e Suen (2002)

Uma arquitetura MLP com classes modulares para reconhecimentos dos nomes manuscritos dos meses do ano é proposta em Kapp, Freitas, *et al.* (2003). As características utilizadas neste artigo são características perceptuais e características baseadas em concavidades/convexidades e seus números de ocorrência, além de mecanismo de extração de primitivas por zoneamento. O artigo alega obter 81,75% de taxa de reconhecimento.

3.3.3 Redes Neurais Convolutivas

Redes convolutivas têm obtido bons resultados em inúmeras bases de dados (CIRESAN, MEIER, *et al.*, 2011), e logo se tornaram o método de comparação no *benchmarking* de novos algoritmos.

Segundo LeCun, Kavukcuoglu e Farabet (2010), os classificadores convolutivos trazem a ideia de que uma boa representação interna segue uma estrutura hierárquica, como pixels sendo agrupados em linhas, linhas em contornos, contornos em partes, partes em objetos e objetos em cena. O sistema de mapeamento de entrada e saída então se comporta

como uma forma de extração de característica hierárquicas que se altera continuamente durante a fase de treinamento. São conceitos inspirados no trabalho clássico de Hubel e Wiesel (1962) a respeito do córtex visual de gatos. Por este motivo são chamadas de arquiteturas multicamadas de Hubel-Wiesel (LECUN, KAVUKCUOGLU e FARABET, 2010).

O melhor resultado até agora publicado para dígitos manuscritos na base NIST SD 19 é o de Ciresan, Meier, Gambardella e Schmidhuber (2011). Utiliza um grupo de sete classificadores, redes neurais convolutivas, cada um com uma versão modificada da base de treinamento, cujas imagens são normalizadas para as larguras de 10,12,14,16,18 e 20 pixels. Segue assim a ideia de gerar diferentes classificadores especialistas, de modo que seus erros não estejam sobrepostos. O treinamento na base MNIST consumiu 800 épocas, obtendo um erro médio de $0{,}27 \pm 0{,}02\%$.

3.3.4 SVM

No contexto de reconhecimento de dígitos manuscritos, classificadores *SVM* podem ser utilizados em associação com um kernel *RBF* (VAMVAKAS, GATOS e PERANTONIS, 2010), utilizados com distorções na base de treinamento, com uso de vetores de suporte virtuais, ou como uma combinação de classificadores, seja em um conjunto não-hierárquico (TRAVIESO, ALONSO e FERRER, 2011) ou em uma estrutura hierárquica conhecida como árvore de decisão.

Segundo Chen e Wang (2009), *SVM* foi originalmente criado para classificação binária, de tal maneira que para serem capazes de resolver problemas multiclasse foram definidos dois tipos de estratégias. Uma é chamada máquina única, onde todos os dados entram em uma mesma fórmula de otimização. Outra abordagem é a criação e combinação de vários classificadores binários, que podem ser classificados de acordo com sua estrutura,

como um-contra-todos, um-contra-um, todos-contra-um, *SVM* de grafo cíclico direto e métodos baseados em árvores hierárquicas.

Um exemplo do uso de um conjunto de classificadores *SVM* é encontrado no artigo Travieso, Alonso e Ferrer (2011), onde é proposta a combinação de três classificadores *SVM*, cada um recebendo um tipo diferente de entrada: máscara de Kirsch e Descritores Elípticos. O resultado final da classificação é dado por uma média ponderada das saídas dos classificadores de acordo com pesos pré-definidos. Testado na base de dados NIST-SD19, obteve taxa de acerto de 98,55% com variância de 0,02.

O trabalho de Chen, Wang C. e Wang R. (2009) apresenta uma estrutura de árvores binárias adaptativas que obteve 97,02% de taxa de acerto na base MNIST. Os autores alegam atingir um tempo de classificação rápido pois o algoritmo proposto reduz o número de vetores de suporte ao invés de apenas reduzir o número de classificadores *SVM* binários.

Uma arquitetura baseada em uma árvore de decisão para classificadores *SVM* é proposta em Madzarov, Gjorgjevikj e Chorbev (2009). Sua forma de operação envolve decisões binárias sendo realizadas nos nodos. As vantagens incluem eficiência no custo computacional estrutura em árvore e alta de taxa de acerto dos classificadores *SVM*. O trabalho relata taxas de acerto de 97,55% na base MNIST.

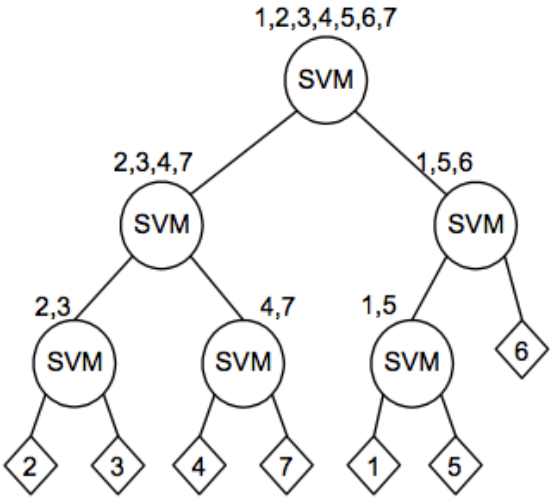

Figura 6 - Árvore de decisão binária de classificadores SVM **(MADZAROV, GJORGJEVIKJ e CHORBEV, 2009)**

4 FUNDAMENTOS TEÓRICOS

Neste capítulo, será apresentado a fundamentação teórica necessária para o entendimento do trabalho proposto. Na primeira parte, há uma breve descrição sobre os métodos de normalização de dados utilizados, como a remoção de *outliers*. Na seção seguinte, será abordado os métodos de extração dos Descritores de Fourier. A última parte deste capítulo será dedicada ao método de treinamento de redes neurais com retropropagação.

4.1 Remoção de padrões indesejados (*outliers*)

Um *outlier* é definido como um ponto que é muito distante da média da variável aleatória correspondente (THEODORIDIS e KOUTROUMBAS, 2009). Esta distância é medida com respeito a um dado limite, usualmente tomando como referência o desvio padrão. Sabe-se que para uma variável aleatória, com distribuição normal, uma distância de duas vezes o desvio padrão cobre 95% dos pontos, e a distância de três vezes o desvio padrão cobre 99% dos pontos. Segundo Theodoridis e Koutroumbas (2009) usualmente considera-se como limite uma distância de duas ou três vezes o desvio padrão em relação a média. Pontos situados além do limite estabelecido são descartados. Esse descarte ocorre porque pontos com valores muito diferentes do valor médio produzem grandes erros durante o treinamento e podem ter efeitos desastrosos.

4.2 Normalização de Dados

Em muitas situações um projetista pode ser confrontado com características cujos valores pertencem a limites muito dinâmicos e diferentes. Assim, características com faixas dinâmicas muito grandes podem influenciar muito mais um sistema de reconhecimento de padrões do que características com faixas dinâmicas menores, embora isso não reflita exatamente a real significância desta característica no projeto do classificador. Em

Theodoridis e Koutroumbas (2009) é sugerido que o problema pode ser solucionado ao normalizar a característica de tal forma que seus valores estejam em intervalos similares. Uma técnica muito utilizada é normalizar através das respectivas estimativas de média e variância. Theodoridis e Koutroumbas (2009) define um número l de características, e para um conjunto de N observações de uma característica x_k, o valor médio da característica x_k é definido por:

$$\bar{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_{ik} \tag{1}$$

para $k = 1,2,\ldots,l$

E o desvio padrão da característica x_k é dado por:

$$\sigma_k^2 = \left\{\frac{1}{N-1}\right\}\sum_{i=1}^{N}(x_{ik} - \bar{x}_k)^2 \tag{2}$$

O valor normalizado da característica x_k é dada então por:

$$\hat{x}_{ik} = \frac{x_{ik} - \bar{x}_k}{\sigma_k} \tag{3}$$

Em que

$$\sigma_k = \sqrt{\sigma_k^2} \tag{4}$$

Em outras palavras, todas as características normalizadas resultantes terão média zero e variância igual a 1. Esta operação é obviamente um método linear. Outras técnicas lineares limitam o valor das características a um intervalo fixo $[0,1]$ ou $[-1,1]$ através de um escalamento apropriado. Além dos métodos lineares, métodos não-lineares também têm sido empregados em casos onde os dados não são bem distribuídos ao redor da média (THEODORIDIS e KOUTROUMBAS, 2009). Em tais casos, transformações baseadas em funções não-lineares (por exemplo, logaritmo, ou *sigmoid*) podem ser usadas para mapear dados com intervalos especificados. Um método não linear de escalamento que goza de certa popularidade é o método *softmax*. Em Theodoridis e Koutroumbas (2009), é mostrado um método que consiste nos seguintes passos:

Passo 1: Calcular o novo valor de característica para distribuição Normal com desvio padrão 1 e média 0

$$y_{ik} = \frac{x_{ik} - \bar{x}_k}{\sigma_k} \tag{5}$$

Passo 2: Normalizar o valor de característica encontrado

$$\hat{x}_{ik} = \frac{1}{1 + e^{-y}} \tag{6}$$

Ou seja, é basicamente uma função de esmagamento, limitando os dados no intervalo de $[0, 1]$. Usando uma aproximação de expansão de séries, não é difícil perceber que para pequenos valores de y este método é aproximadamente uma função linear com respeito a x_{ik}. O intervalo de valores de x_{ik} que corresponde a parte linear depende do desvio padrão e do fator r, que é livremente definido. Valores longe da média são 'esmagados' exponencialmente.

4.3 Medidas de separabilidade entre classes

4.3.1 Divergência

De acordo com a regra de Bayes (THEODORIDIS, 2007), dadas duas classes $\boldsymbol{\omega}_1$ e $\boldsymbol{\omega}_2$ e um vetor de características x, pode-se dizer que x pertence a classe $\boldsymbol{\omega}_1$ se:

$$P(\omega_1|x) > P(\omega_2|x) \tag{7}$$

Dessa forma, o erro de probabilidade depende da diferença entre $P(\boldsymbol{\omega}_1|x)$ e $P(\boldsymbol{\omega}_2|x)$. Logo, a taxa $\frac{P(\boldsymbol{\omega}_1|x)}{P(\boldsymbol{\omega}_2|x)}$ pode fornecer uma boa informação acerca das capacidades discriminatórias com respeito as duas classes $\boldsymbol{\omega}_1$ e $\boldsymbol{\omega}_2$. Verifica-se que a mesma informação reside na relação $\boldsymbol{ln}\frac{p(x|\boldsymbol{\omega}_1)}{p(x|\boldsymbol{\omega}_2)} = D_{12}(x)$. Para classes completamente sobrepostas, o valor de $D_{12} = 0$. Considerando o valor médio sobre a classe $\boldsymbol{\omega}_1$, obtém-se:

$$D_{12} = \int_{-\infty}^{+\infty} p(x|\boldsymbol{\omega}_1)\, \boldsymbol{ln}\frac{p(x|\boldsymbol{\omega}_1)}{p(x|\boldsymbol{\omega}_2)} \tag{8}$$

Da mesma forma para a classe $\boldsymbol{\omega}_2$, tem-se:

$$D_{21} = \int_{-\infty}^{+\infty} p(x|\boldsymbol{\omega}_2)\, \boldsymbol{ln} \frac{p(x|\boldsymbol{\omega}_2)}{p(x|\boldsymbol{\omega}_1)} \quad (9)$$

A soma $d_{12} = D_{12} + D_{21}$ é conhecida como divergência (THEODORIDIS e KOUTROUMBAS, 2009) e pode ser usada como medida de separabilidade para as classes $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, com respeito ao vetor de características x. Para problemas multiclasse, a divergência é calculada para cada par $\boldsymbol{\omega}_i$, $\boldsymbol{\omega}_j$:

$$d_{ij} = D_{ij} + D_{ji} \quad (10)$$

$$d_{ij} = \int_{-\infty}^{+\infty} (p(x|\boldsymbol{\omega}_i) - p(x|\omega_j))\boldsymbol{ln} \frac{p(x|\boldsymbol{\omega}_i)}{p(x|\boldsymbol{\omega}_j)} dx \quad (11)$$

E a separabilidade média de classe pode ser calculada utilizando a divergência média:

$$d = \sum_{i=1}^{M} \sum_{j=1}^{M} P(\boldsymbol{\omega}_i) P(\boldsymbol{\omega}_j) d_{ij} \quad (12)$$

Assumindo que as funções de densidade são Gaussianas $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ e $\mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ a divergência para o caso unidimensional pode ser dada por:

$$d_{ij} = \frac{1}{2}\left(\frac{\sigma_j^2}{\sigma_i^2} + \frac{\sigma_i^2}{\sigma_j^2} - 2\right) + \frac{1}{2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^2 \left(\frac{1}{\sigma_i^2} + \frac{1}{\sigma_j^2}\right) \quad (13)$$

A medida de separabilidade de classe não depende somente da diferença entre valores de média, também depende da variância. Assim, d_{ij} pode ser maior mesmo para valores de media iguais porém com variâncias que tenham uma diferença significativa. Dessa forma, a separação entre classe é possível mesmo se os valores de média das classes coincidirem.

4.3.2 Fisher's Discriminant Ratio (FDR)

FDR é utilizado algumas vezes para quantificar a capacidade de separabilidade de características individuais, considerando as características como estatisticamente independentes (THEODORIDIS e KOUTROUMBAS, 2009). Um maior valor da razão FDR, tal como é mostrada na equação abaixo, significa uma medida maior de separabilidade entre as classes.

$$FDR = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \quad (14)$$

Segundo Theodoridis e Koutroumbas (2009), Para problema multiclasse, FDR pode ser estendido para:

$$FDR = \sum_i^M \sum_{j \neq i}^M \frac{(\mu_i - \mu_j)^2}{\sigma_i^2 + \sigma_j^2} \quad (15)$$

Onde:

- $\boldsymbol{\mu}_i$ é a média das amostras da classe i
- $\boldsymbol{\mu}_j$ é a média das amostras da classe j
- $\boldsymbol{\sigma}_i^2$ é a variância das amostras da classe i
- $\boldsymbol{\sigma}_j^2$ é a variância das amostras da classe j

4.4 Seleção de Características

Um grande problema associado com o reconhecimento de padrões é a assim chamada maldição da dimensionalidade (THEODORIDIS e KOUTROUMBAS, 2009). Esse fenômeno ocorre quando existe um grande número de características a disposição de um projetista de um sistema de reconhecimento de padrões. Existem algumas razões para reduzir o número de características para o mínimo necessário. Uma delas é a questão da redução da complexidade computacional. Outra razão é que, embora duas características possam carregar uma boa informação de classificação quando tratadas separadamente, há pouco ganho quando são combinadas em um vetor de características, por conta de sua alta correlação mútua.

Do ponto de vista das propriedades de generalização do classificador, quanto maior for a relação do número de padrões de treinamento N para o número de parâmetros livres do classificador, melhores são as qualidades de generalização do classificador resultante (LECUN, BOTTOU, *et al.*, 1998). Um maior número de característica é diretamente traduzido em um maior número de parâmetros do classificador, por exemplo, pesos sinápticos

em uma rede neural, pesos de um classificador linear. Assim, de acordo com Theodoridis e Koutroumbas (2009), para um número N de padrões de treinamento e número l de parâmetros do classificador, manter uma alta taxa $\frac{N}{l}$, aumenta o desempenho da classificador, aumentando sua estimativa de erro de classificação.

4.4.1 Seleção Escalar de Características

Seleção escalar de características trata do problema de selecionar um subconjunto de características a partir de um vetor de características, tratando uma a uma, de forma individual, ou seja, como escalares (THEODORIDIS e KOUTROUMBAS, 2009). É necessário escolher um método de medida de separabilidade de classe, um critério $C(k)$ calculado para cada característica $k = 1,2, \dots, l$. As características então são organizadas em ordem decrescente de valores de $C(k)$, e são escolhidas os melhores valores de $C(k)$ para formar um novo vetor de característica.

A maior vantagem de lidar com características individualmente é a redução da complexidade computacional. Embora, tais abordagens não levem em conta as correlações existentes entre as características. Assim, o coeficiente de correlação cruzada entre duas características i e j, é definido em Theodoridis e Koutroumbas (2009) como:

$$p_{ij} = \frac{\sum_{n=1}^{N} x_{ni} x_{nj}}{\sqrt{\sum_{n=1}^{N} x_{ni}^2 \sum_{n=1}^{N} x_{nj}^2}} \tag{16}$$

De acordo com Theodoridis e Koutroumbas (2009), o procedimento de seleção de característica se desenvolve nos seguintes passos:

- Selecione um critério $C(k)$ de separabilidade de classe e compute todo os seus valores para todas as características disponíveis x_k para $k = 1,2, \dots, l$. Ordene de forma decrescente e escolha o melhor valor de $C(k)$. Vamos chamar de x_{i_1}

- Para selecionar uma segunda característica, calcule o coeficiente de correlação cruzada entre a característica escolhida x_{i_1} e cada um das restantes $l - 1$ características
- Escolha a característica x_{i_2} para o qual

$$i_2 = \ argmax_{x_j}\{\alpha_1 \ C(j) - \alpha_2 \rho_{i_1 j}\} \tag{17}$$

onde $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ são fatores de peso para determinar a importância relativa que damos para os dois termos.

4.5 Traçado da Borda

O próposito do algoritmo é extrair informação da borda (contorno) de um caractere segmentado e apresentá-lo em uma forma mais compactada. Existem vários métodos de traçado de borda. O método de adjacência de '4-vizinhança' é adotado neste trabalho.

Basicamente, o algoritmo de Cheriet, Kharma, *et al.* (2007) percorre toda a imagem binária até encontrar a borda. Uma vez que um pixel de borda é detectado, continua a busca até encontrar o pixel seguinte. Dessa forma, o traçado seguirá o contorno automaticamente.

Quando o primeiro pixel é encontrado, o programa atribui as coordenadas daquela posição para indicar que esta é a origem do traçado (CHERIET, KHARMA, *et al.*, 2007). O algoritmo de traçado busca então pelos pixels mais próximos. A busca ocorre seguindo a sentido horário se acordo com o que mostra a Figura **7**. Assim, de acordo com Cheriet, Kharma, *et al.* (2007) a sequência de busca é iniciada em P1 para P2, então P3 e finalmente P4.

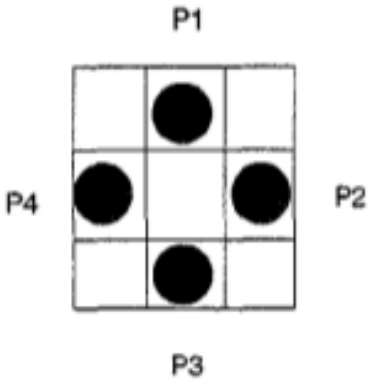

Figura 7 – Ilustração da sequência de buscas do algoritmo de traçado de borda (CHERIET, KHARMA, *et al.*, 2007)

Conforme o algoritmo de traçado se movimenta pelos contornos da imagem, as coordenadas correspondentes são armazenadas em um vetor, com o objetivo de calcular os Descritores de Fourier.

Durante o processo de busca pelo contorno, o programa verifica sempre se as primeiras coordenadas obtidas são iguais as últimas. Atingindo esta condição, o algoritmo termina sua operação, pois todo o contorno deverá ter sido traçado.

4.6 Descritores de Fourier

Na literatura foram desenvolvidos inúmeros métodos para a extração da forma de imagens, embora os descritores de Fourier sejam um dos métodos mais populares e eficientes. Em princípio, representa a forma do objeto no domínio da frequência. Por fazer isso, os descritores de Fourier tiram proveito de múltiplas vantagens: forte habilidade de discriminação, baixa sensitividade a erros, fácil normalização e preservação de informação. A sequência discorre sobre as diferentes assinaturas de forma e então explica minuciosamente os descritores padrões de Fourier.

De acordo com Theodoridis e Koutroumbas (2009), Descritores de Fourier são usados para descrever a forma (curva fechada) de qualquer objeto encontrado em uma imagem de entrada. Um pré-requisito para computar os Descritores de Fourier é digitalizar e extrair as

informações de borda do objeto e normalizar todas as informações. Durante a normalização, os pontos de dados da forma das bordas do objeto são amostrados para terem o mesmo número de pontos. Como os descritores de Fourier requerem uma representação unidimensional da informação de borda, assinaturas de forma são utilizadas. A assinatura da forma mapeia uma representação 2D de uma forma para uma representação 1D. Embora existam inúmeros tipos de assinaturas de forma, serão considerados apenas três tipos: coordenadas complexas, distância de centróide e curvatura.

Em Cheriet, Kharma, *et al.* (2007) é mencionado que a transformada de Fourier para uma forma assinatura formada com N pontos de amostragem, assumindo que s(t) é normalizada, é dada por:

$$u_n = \frac{1}{N}\sum_{t=0}^{N-1} s(t)e^{\left(\frac{-j\pi nt}{N}\right)} \quad (18)$$

Os coeficientes são chamados de descritores de Fourier da amostra, denotados por FD_n (CHERIET, KHARMA, *et al.*, 2007).

4.7 Redes Neurais

Redes Neurais são compostas de simples elementos operando em paralelo (HAGAN, DEMUTH e BEALE, 1995). Estes elementos são inspirados no sistema nervoso biológico. Assim como na natureza, as conexões entre elementos determinam a função da rede. É possível treinar uma rede neural para realizar uma função particular através de ajustes nos valores das conexões, ou seja, um ajuste de pesos, entre elementos.

Segundo Hagan, Demuth e Beale (1995), quanto ao treinamento, existem dois tipos de redes neurais: redes baseadas em treinamento supervisionado e redes baseadas em treinamento não supervisionado. Essas últimas destinam-se a aplicações de agrupamento de dados, enquanto que as primeiras destinam-se a aplicações de reconhecimento de padrões.

No treinamento supervisionado, as redes neurais são ajustadas, ou treinadas, de forma que uma entrada particular leve a uma saída alvo específica. A rede é ajustada, baseada na comparação da saída e o alvo, até que a saída da rede se torne igual ao valor alvo. O treinamento da rede necessita de pares de entrada e valores alvo, permitindo que a rede atinja o seu objetivo de projeto.

As redes neurais convencionais utilizadas nesse trabalho têm uma arquitetura de propagação direta, sem realimentações, com três camadas, conforme mostrado na Figura **8**. A primeira camada é denominada de camada de entrada. A segunda camada é denominada de camada intermediárias ou camada escondida, enquanto que a última camada é denominada de camada de saída. A arquitetura de uma rede de três camadas é usualmente descrita por três números: $n - m - o$, em que: n é o número de neurônios da camada de entrada, m é o número de neurônios da camada intermediária e o é o número de neurônios da camada de saída.

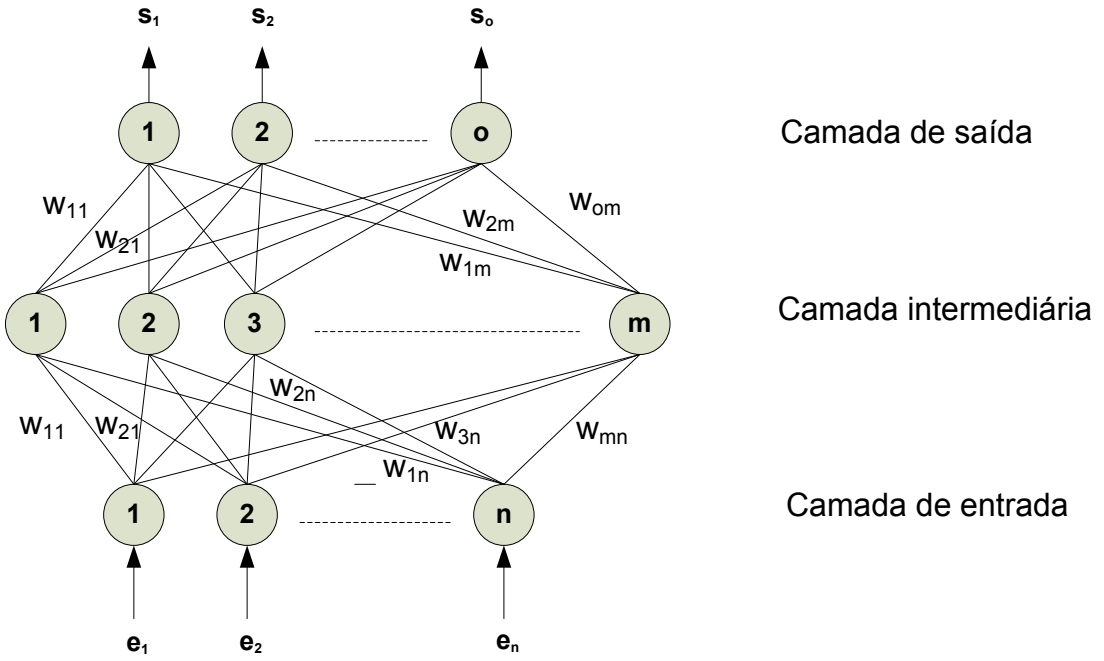

Figura 8 – Arquitetura de uma Rede neural de três camadas

Redes Neurais têm sido utilizadas para realizar funções complexas em vários campos, incluindo reconhecimento, identificação e classificação de padrões, aplicações envolvendo voz, visão e em sistemas de controle (HAGAN, DEMUTH e BEALE, 1995). Problemas difíceis para computadores convencionais ou para seres humanos também podem ser resolvidos através de redes neurais treinadas para este fim.

4.7.1 Redes Neurais Multi-Layer Perceptron com o algoritmo de treinamento Backpropagation

Métodos de aprendizagem baseada em gradientes tem sido usados desde a década de 1950, mas eles tem sido limitados a sistemas lineares. O algoritmo *backpropagation* é de longe o algoritmo de aprendizagem mais utilizado para o treinamento de redes neurais artificiais através de ajustes de pesos (LECUN, BOTTOU, *et al.*, 1998). A idéia básica do método de *backpropagation* é que gradientes podem ser calculados de maneira eficiente por meio de propagação da saída para a entrada. Esta idéia foi descrita na literatura de teoria de controle das décadas de 1960.

Conhecido também como método de gradiente descendente, o algoritmo *backpropagation* se baseia no cálculo do gradiente do erro total, calculado durante o ajuste dos pesos, tem como pré-requisito que as funções de ativação dos nodos sejam contínuas e, geralmente, não decrescentes, assim tornando possível o cálculo da derivada da função.

O algoritmo minimiza a função de custo que é definida pelo erro total cometido pela rede, ou seja, a média da soma dos erros quadráticos. De acordo com Hagan, Demuth e Beale (1995) o sinal de erro na saída do neurônio j na iteração n é calculado como:

$$e_j(n) = d_j(n) - y_j(n) \tag{19}$$

Logo a média da soma dos erros quadráticos pode ser definida por:

$$E_{av} = \frac{1}{2N}\sum_{j=1}^{N}\left(d_j - y_j\right)^2 \tag{20}$$

onde E_{av} é o erro total médio cometido pela rede, N é a quantidade de padrões pertencentes ao conjunto de treinamento, d_j é a saída desejada para a amostra j fornecida a rede como entrada e y_j é o valor produzido pela rede como saída para a amostra j.

O algoritmo *backpropagation* segue uma abordagem de aprendizagem supervisionada, pois lida com pares de valores (amostra de entrada, saída desejada). Métodos não supervisionados por sua vez, são definidos quando não há o conhecimento a priori da saída desejada para o padrão de entrada. Supervisionado e não supervisionado são as duas principais classificações de algoritmos quanto à forma de aprendizagem.

Durante o treinamento da rede neural utilizando o algoritmo *backpropagation*, ocorrem duas fases distintas: fase de propagação direta, onde são calculadas as saídas de cada nodo para cada padrão de entrada apresentado à rede; e fase de retro-propagação, que propaga o erro cometido pela rede da saída para entrada, atualizando os pesos das conexões das redes. Assim, o algoritmo *backpropagation* define os seguintes passos (THEODORIDIS e KOUTROUMBAS, 2009):

- Passo de inicialização: Inicializar todos os pesos com pequenos valores aleatórios.
- Cálculo de propagação direta: Para cada um dos vetores de características $x(i)$ das amostras de treinamento, sendo $i = 1, \ldots, N$, calcular os valores de saída de cada neurônio.
- Calcular a função de custo para os valores atuais dos pesos, ou seja, calcular o valor de propagação do erro médio.
- Atualizar pesos e valor de polarização de cada camada da rede.

A partir da equação $e_j(n) = d_j(n) - y_j(n)$ (19,

temos a derivada do erro de saída, calculado somente para um neurônio j da camada de saída:

$$e_j(n) = \Big(d_j(n) - y_j(n)\Big) f'\Big(z_j(n)\Big) \tag{21}$$

onde, $d_j(n)$ é o valor de saída desejado, $y_i(n)$ é saída calculada para o neurônio j, f' é a função de ativação derivada com respeito ao valor de ativação $z_j(n)$, que é representado pela equação:

$$z_j(n) = \sum_{i=1}^{N} x_i(n) w_{ji} \tag{22}$$

Em Hagan, Demuth e Beale (1995) o valor n representa o número de nodos da camada imediatamente anterior conectados ao nodo j, x_i é a saída de cada nodo da camada anterior que serve como entrada do nodo j e w_{ji} é o peso associado à conexão entre cada nodo da camada imediatamente anterior e o nodo j.

Dado um neurônio j^{M-1} da camada intermediária, ou seja, da camada $(M-1)$, seu erro é dado pela equação

$$\delta_{j^{M-1}} = f'\Big(net_{j^{M-1}}\Big) \sum_n \delta_n w_{nj}^{M-1} \tag{23}$$

O erro δ_n é relacionado aos neurônios imediatamente anteriores a camada $(M-1)$ e w_{nj}^{M-1} significa o valor dos pesos que ligam o neurônios j aos da camada anterior.

Dessa maneira, em Hagan, Demuth e Beale (1995) conclui que a equação de ajuste dos peso é dada por:

$$W_{ij}(k+1) = W_{ij}(k) + \Delta W_{ij} \tag{24}$$

$$\Delta W_{ij} = \eta \delta_j p_i \tag{25}$$

onde η é a taxa de aprendizagem cujo valor deve pertencer ao intervalo [0,1], δ_j representa o erro calculado em um neurônio da camada de saída ou camada anterior e p_i é o valor de entrada do neurônio j.

Para evitar custos computacionais desnecessários, um critério de parada do algoritmo é estabelecido: ao atingir uma taxa de erro mínima ou quando ultrapassar um número máximo de iterações.

Na literatura diversos métodos foram desenvolvidos para acelerar o treinamento de uma rede neural. Nesse trabalho utilizou-se o algoritmo Levenberg-Marquadt . Esse algoritmo é uma variação do método de Newton que foi projetada para minimizar funções que são a soma dos quadrados de outras funções não-lineares. Segundo Hagan, Demuth e Beale (1995) este algoritmo converge em menos iterações do que a maioria dos métodos conhecidos na literatura. Apesar da desvantagem de possuir um custo computacional maior por iteração, dado que é necessário calcular a matriz inversa, este algoritmo é o método de treinamento de redes neurais mais rápido. Devido as características de agilidade no treinamento, este método foi escolhido para ser utilizado neste trabalho.

4.7.2 Aprendizagem Profunda (*Deep Learning*)

O termo "Deep Learning", que pode ser traduzido literalmente para "Aprendizagem Profunda", representa um conjunto de métodos para o aprendizado de características hierárquicas através da composição de características mais simples. Esse uso de múltiplos níveis de abstração é particularmente similar ao funcionamento do sistema visual dos primatas (SERRE, KREIMAN, *et al.*, 2007), com uma sequência de estágios de processamento: detecção de bordas, formas primitivas até gradualmente identificando formatos visuais mais complexos.

O aprendizado de características em múltiplos níveis de abstração permite ao um sistema aprender complexas funções de mapeamento dos dados de entrada para saída, de maneira independente de características criadas manualmente. Ou seja, é uma forma de automatizar a geração de características que sejam mais representativas de um determinado problema de reconhecimento de padrões.

4.7.3 Redes Neurais Convolutivas

Redes Neurais convolutivas é um método de *deep learning* especialmente projetado para lidar com a variabilidade em dados bidimensionais (2D), como por exemplo, imagens em sua forma matricial (LECUN, BOTTOU, *et al.*, 1998). Na busca por algum grau de invariância a deslocamento, escala e distorção, combinam as seguintes idéias arquiteturais:

- Tem seu mapeamento de características, ou campos receptivos locais, como um dos pontos de maior similaridade com as redes neurais biológicas, conceito descoberto por Hubel e Wiesel em experimentos com córtex visual de gatos, que garante uma maior robustez a distorções locais.
- Pesos compartilhados permitem uma redução dos parâmetros livres e invariância geométrica.
- Sub-amostragem temporal ou espacial reduz o tamanho total dos mapas de características a cada camada, chegando assim na última camada apenas como valores unidimensionais, o que nesse ponto em diante as torna equivalentes a uma rede neural MLP.

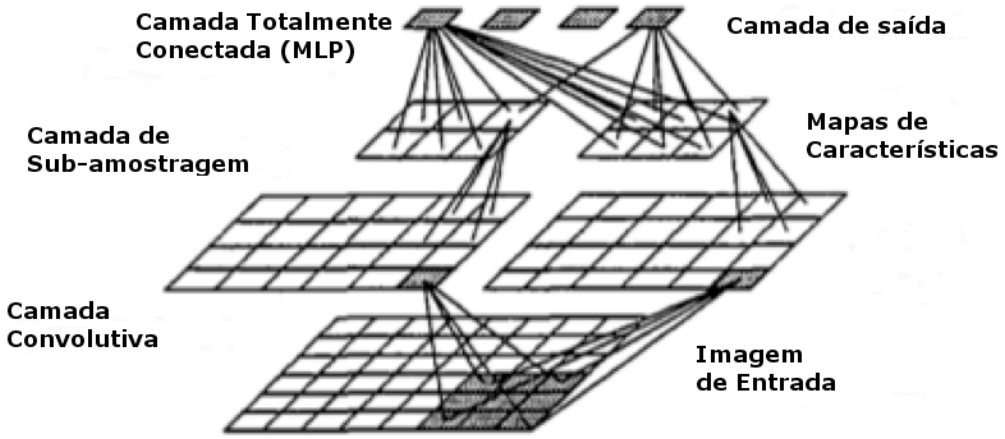

Figura 9 - Mapas de características de uma arquitetura convolutiva (LI, 1999, tradução nossa)

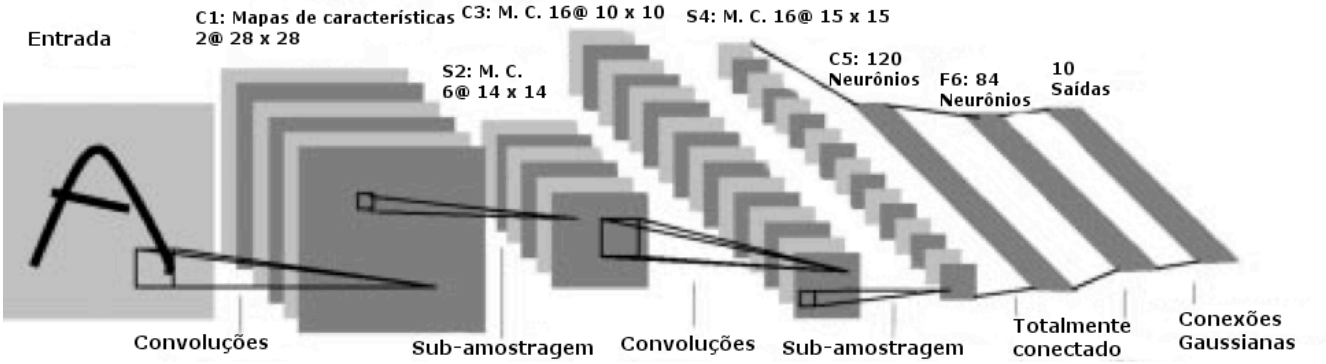

Figura 10 – Visão geral da arquitetura de uma rede neural convolutiva **(LECUN, BOTTOU, *et al.*, 1998)**

A partir de uma imagem como elemento de entrada, cada unidade nas camadas seguintes mapeia uma determinada região da matriz de entrada em uma região específica da matriz de saída. Por este comportamento, a matriz resultante de cada camada é chamada de mapa de características.

Essa implementação sequencial do mapa de características contempla uma passagem total da imagem de entrada por meio dos correspondentes locais no mapa de características. Tal operação é equivalente a uma convolução, sendo esse portanto o motivo do nome das "redes convolutivas". O Kernel da convolução seria assim os pesos da conexão usados pelas unidades do mapa de características.

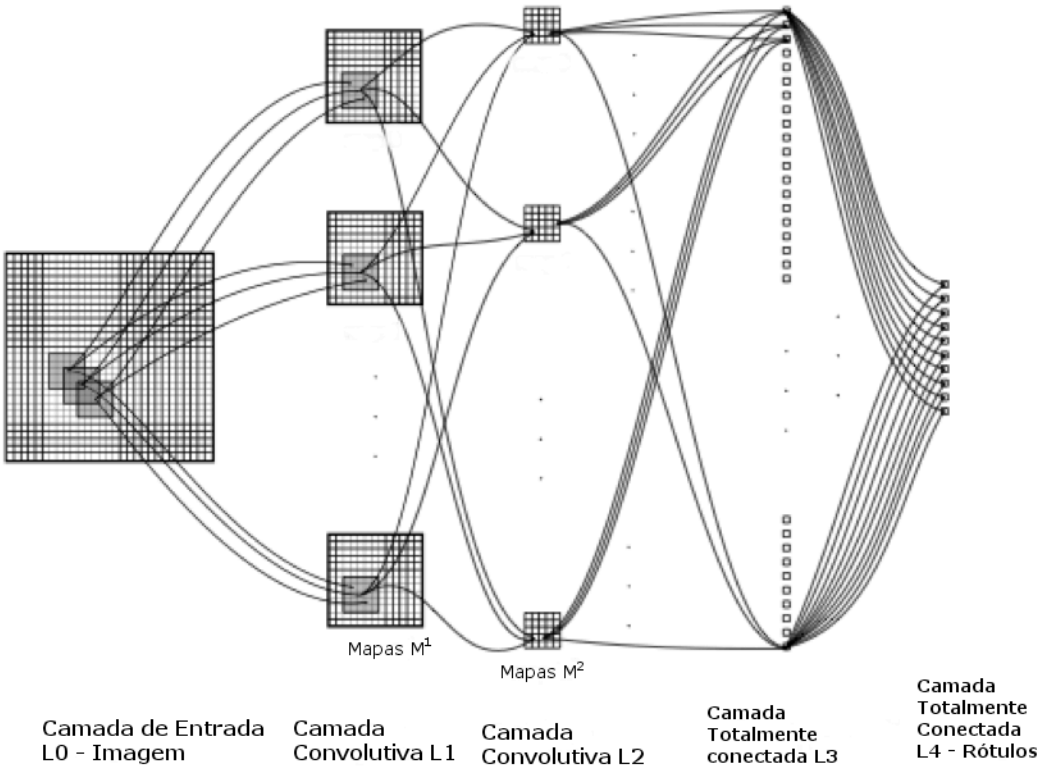

Figura 11 - Arquitetura de rede neural convolutiva com kernels de tamanho 5x5 **(CIREşAN, MEIER, *et al.*, 2011)**

Utilizando o algoritmo *backpropagation* é computado o gradiente da função de perda com respeito a todos os pesos em todas as camadas da rede convolutiva. O algoritmo *backpropagation* padrão usada em redes MLP deve ser ligeiramente modificado para lidar com o compartilhamento de pesos. Uma maneira rápida de implementar esse ajuste é primeiro calcular as derivadas parciais da função de perda com respeito a todas conexões, como se a rede fosse uma rede neural multicamada convencional. Então as derivadas parciais de todas conexões que compartilham o mesmo parâmetro são adicionadas para formar o derivativo com respeito a esse parâmetro.

4.8 Máquina de Vetores de Suporte (*Support Vector Machines - SVM*)

O classificador SVM é largamente usado devido a sua alta acurácia, habilidade de lidar com dados de alta dimensionalidade e flexibilidade em modelar diversas fontes de dados. Definido por Harrington (2012) como um método de destaque em aprendizagem de máquina, SVM é basicamente voltado para classificação binária, ou seja, problemas de reconhecimento entre duas classes, apesar de existirem versões modificadas para lidar com problemas multiclasse.

Segundo Harrington (2012) para um dado hiperplano, pode-se denotar por $x_+(x_-)$ o ponto mais próximo do hiperplano entre as amostras positivas e negativas. A norma do vetor w denotado por $\|w\|$ é o seu comprimento, e é dado por $w^T w$.

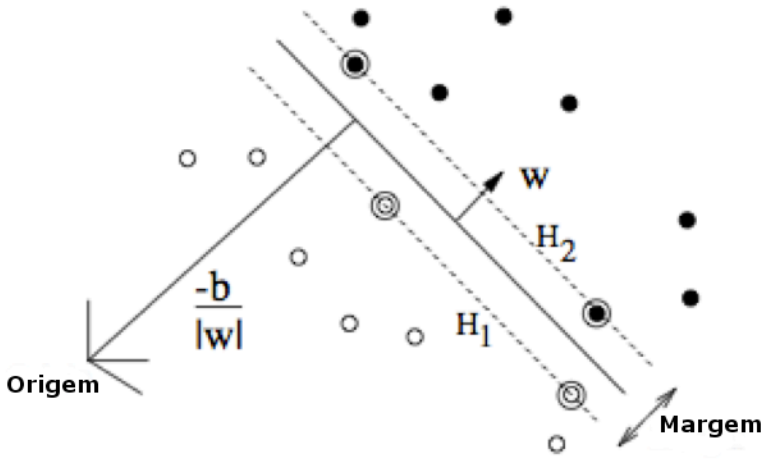

Figura 12 - Hiperplano de decisão **(BURGES, 1998)**

Dado um conjunto de amostras de treinamento em pares com suas saídas desejadas $(x_i, y_i), i = 1 \ldots m,$ onde $x_i \in R^n$ e $y_i \in \{1, -1\}^m$, o SVM (HARRINGTON, 2012) então seleciona o hiperplano de decisão ótimo que maximiza a margens cujo resultado resolve a equação:

$$\mathbf{min}_{w,b}\left(\frac{1}{2}\boldsymbol{w^T w}\right) \text{ sujeito a } \boldsymbol{y_i(w^T x_i + b) \geq 1} \quad (26)$$

onde **w** é o vetor de pesos e **b** a polarização. Quando os pontos de treinamento são não-linearmente separáveis, a função de custo é reformulada ao introduzir uma variável de folga $\xi_i \geq 0,\ i = 1,2, \ldots, m$

$$\mathbf{min}_{w,b}\left(\frac{1}{2}\boldsymbol{w^T w} + \boldsymbol{C}\sum_{i=1}^{m}\boldsymbol{\xi_i}\right) \text{ sujeito a } \begin{matrix}\boldsymbol{y_i(w^T x_i + b) \geq 1 - \xi_i} \\ \boldsymbol{\xi_i \geq 0}\end{matrix} \quad (27)$$

onde $C > 0$ é um parâmetro de penalidade do termo de erro. Embora, quando a função de decisão não é linear, o esquema acima não pode ser usado diretamente e o SVM requer que a solução para o problema de otimização seja definida da seguinte maneira:

$$\mathbf{min}_{w,b}\left(\frac{1}{2}\boldsymbol{w^T w} + \boldsymbol{C}\sum_{i=1}^{m}\boldsymbol{\xi_i}\right) \text{ sujeito a } \begin{matrix}\boldsymbol{y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i} \\ \boldsymbol{\xi_i \geq 0}\end{matrix} \quad (28)$$

Em Harrington (2012) é mencionado que o classificador SVM pertence a categoria geral de métodos que fazem uso de kernels. Um método baseado em kernel é um algoritmo que opera uma mudança de base vetorial através da operação de multiplicação escalar. Muitas vezes uma função Kernel calcula uma multiplicação escalar em um espaço de características de alta dimensão. Outra vantagem é a possibilidade de gerar fronteiras de decisão não-lineares usando métodos projetados para classificadores lineares. O uso de funções kernel permitem aplicar em um classificador dados que não possuem representação em um espaço vetorial de dimensão fixada.

Segundo Harrington (2012), uma abordagem simples de tornar um classificador em não-linear é mapear seus dados de entrada para o espaço de entrada *X* para um espaço de características $\mathcal{F}$, usando uma função não-linear $\phi: X \to \mathcal{F}$.

Métodos que utilizam kernels lidam com o problema da complexidade ao evitar o passos de mapeamento explícito de dados para o espaço de característica de alta dimensionalidade. Um exemplo de kernel largamente usado é o Gaussiano:

$$k(x, x') = \exp(-\gamma \|x - x'\|^2) \quad (29)$$

onde $\gamma > 0$ é um parâmetro que controla a largura da Gaussiana.

Percebe-se assim que podem ser aplicados Kernels nas fronteiras de decisão, ou seja, sua dependência dos dados é somente através de produto escalar. Muitos algoritmos de

aprendizagem de máquina podem ser expressados usando kernels, incluindo, o algoritmo do *perceptron* e *SVM*.

4.9 Árvores de Decisão

Uma árvore de decisão representa uma função cuja entrada é um vetor de atributos e saída é um resultado de decisão, sob a forma de um valor único de saída (RUSSELL e NORVIG, 2010). Os valores de entrada ou saída podem ser discretos ou contínuos. Uma árvore de decisão realiza sua decisão através de uma sequencia de testes.

Cada nodo da árvore é definido como uma condição de teste. Cada resultado possível do teste é uma ramificação desse nodo, levando a dois tipos de caminhos possíveis: 1) nós folha que representam o resultado final da decisão ou resultado da classificação da árvore; 2) para novos nós de teste. Um exemplo é mostrado na Figura **13**.

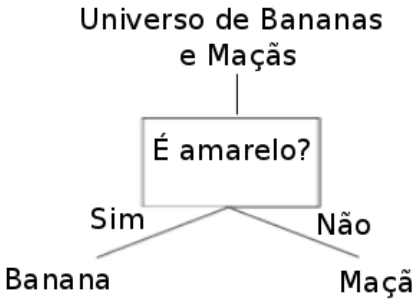

Figura 13 – Simples árvore de decisão para a classificação entre Bananas e Maçãs.

A Figura **14** exemplifica uma árvore de decisão para o problema de um pai ao lidar com um bebê chorando, onde as classes de decisão representam ações que o pai poderá tomar. Em cada nó é realizado algum tipo de teste para avaliar qual a decisão a ser tomada.

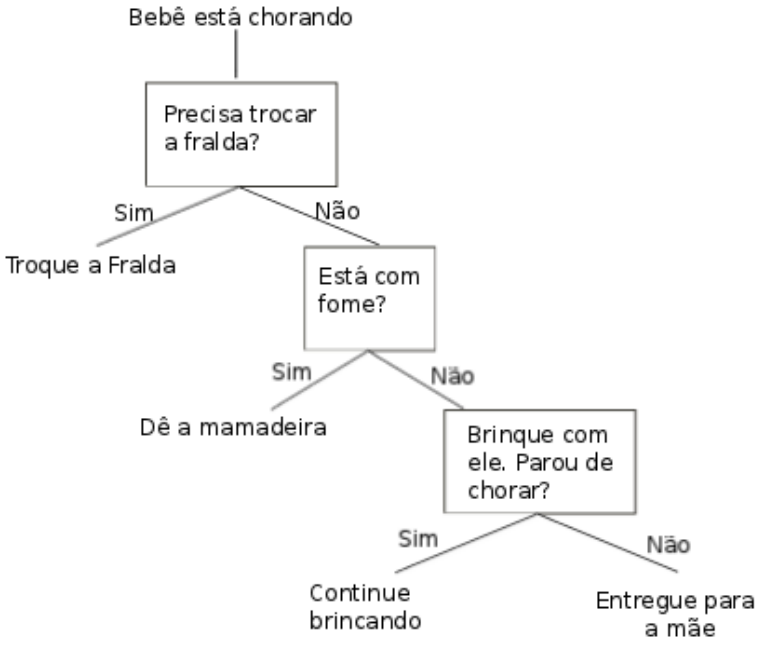

Figura 14 – Árvore de decisão para possíveis ação de um pai ao lidar com um bebê chorando.

4.10 Combinação de Classificadores para Reconhecimento de Problemas Multi-classe

Segundo Ramanan, Suppharangsan e Niranjan (2007) *SVM* foi originalmente desenvolvido para solucionar problemas de classificação binária, entretanto mais de um classificador *SVM* podem ser combinados para lidar com problemas multiclasse. Existem algumas combinações de classificadores projetadas de forma a lidar com problemas multiclasse:

Cada uma das amostras a serem classificadas são apresentadas para a estrutura de classificadores combinados, que retorna como saída, uma das classes do problema na qual a amostra é reconhecida.

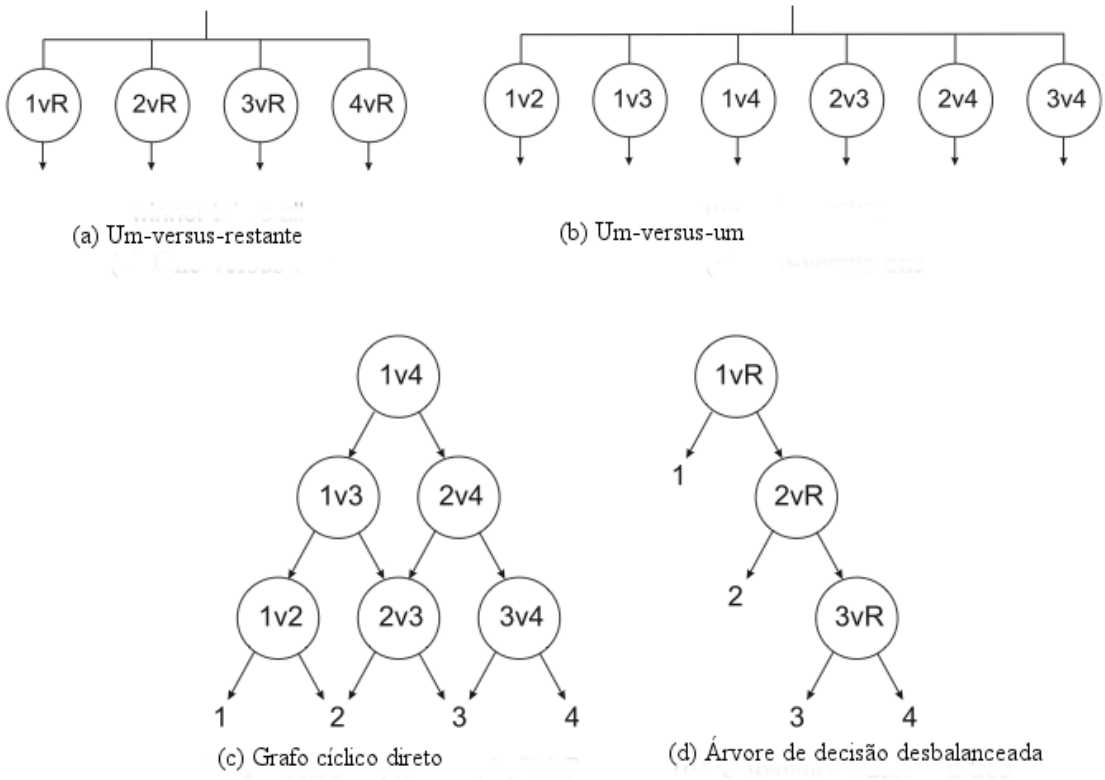

Figura 15 – Arquiteturas de combinação de classificadores binários para um problema de 4 classes **(HASSAN e DAMPER, 2012)**

A seguir serão detalhadas algumas arquiteturas encontradas na literatura para lidar especialmente com problemas multiclasses. Nota-se que essas soluções não são dependentes do tipo de classificador, sendo assim perfeitamente aplicáveis a qualquer classificador binário.

4.10.1 Um-versus-Restante

Para a arquitetura de "um-versus-restante", o problema de distinguir entre m classes requer a criação de m classificadores. Cada classificador C_i é treinado para reconhecer as amostras da classe i, retornando uma saída positiva, e uma saída negativa caso a amostra seja reconhecida como pertence as demais classes.

O grande benefício da arquitetura "Um-versus-restante" é ser necessário testar somente m classificadores. Entretanto para casos em que a base de treinamento é altamente

desbalanceada, ou seja, uma diferença muito grande entre as quantidades de amostras para cada classe, pode comprometer seriamente o desempenho desta associação de classificadores.

4.10.2 Um-versus-um

A arquitetura "Um-versus-um" usa a estratégia de votação majoritária, ou seja, uma amostra é definida como sendo da classe i se existir uma quantidade maior de votos para essa classe. São construídos $m(m-1)/2$ classificadores binários, um para cada par possível e distinto entre as classes, sendo avaliados em paralelo.

Cada classificador C_{ij} é treinado utilizando somente as amostras da classe i e da classe j. Para uma dada amostra x_n, se o classificador C_{ij} reconhece como sendo da classe i, então um voto é adicionado para a classe i, caso contrário, se for reconhecida como da classe j, então um voto é computado para a classe j. Quando a amostra x_n passar pela avaliação de todos os classificadores, a classe que tenha mais votos é a que será atribuída à amostra como resultado final da classificação.

O método do "um-versus-um" não é afetado tanto pelo desbalanceamento das classes na base de treinamento quanto o "um-versus-restante", pois cada par de classe é tratado isoladamente, permitindo definir uma quantidade proporcional de uma classe em relação a outra. Embora a arquitetura "um-versus-um" exija uma quantidade maior de classificadores.

4.10.3 Grafo Acíclico Direto

Um conjunto de classificadores binários também pode ser estruturado como um grafo acíclico direto (DAG – Directed Acyclic Graph), como pode ser visto um exemplo para um problema de 4 classes na imagem (c) da Figura **15**. Em cada nodo de classificação, são treinados classificadores "um-para-um", ou seja, que reconhecem exclusivamente entre duas classes do problema. Percebe-se que para este tipo de arquitetura é necessário $m(m-1)/2$

classificadores, porém apenas são avaliados $(m-1)$ nodos, ou seja, a amostra corrente passa somente por $(m-1)$ classificadores até obter sua classificação final. Na construção do fluxo de avaliações a serem executas, é observado que caso a saída de um classificador C_{ij} seja a classe i, então nos nodos seguintes a classe j não será mais apresentada, e por esse motivo apenas $(m-1)$ classificadores são acionados durante a classificação de uma amostra.

4.10.4 Árvore de Decisão Desbalanceada

UDT (Unbalanced Decision Tree) é uma implementação proposta em Ramanan, Suppharangsan e Niranjan (2007) seguindo o conceito "um-versus-restante" em cada nodo de decisão mas em termos de estrutura geral é projetado em formato cascata, pendendo sempre para a direita, como pode ser visto na imagem (d) da Figura **15**. Embora em comparação com arquitetura de "um-versus-restante" são necessários apenas $(m-1)$ classificadores ao invés de m classificadores.

A definição do resultado final, ou o resultado de classificação, para essa arquitetura segue alguns passos. Primeiro a avaliação se inicia no nodo 1, localizado no ponto mais superior da árvore. Para cada nodo n é feita a avaliação do seu respectivo classificador C_i, o processo decisório então segue para esquerda caso seja classificada como pertencente a classe do qual esse classificador é especialista, terminando o processo de classificação. Caso contrário, caso a amostra seja classificado como restante, recomeça o processo de avaliação para o próximo nodo. O nodo final (m - 1) possui então duas possibilidades de classificação, ou a amostra pertence ao classe (m - 1) ou é considerada como uma classe m. De acordo com Hassan e Damper (2012), a estrutura UDT segue uma abordagem "nocaute", que no pior dos casos requer apenas (m - 1) avaliações de classificadores, e no melhor dos casos apenas um.

5 MATERIAIS E MÉTODOS

A metodologia adotada para obter o classificador com o melhor desempenho possível foi através de uma comparação entre diferentes métodos. Serão exploradas três abordagens: o uso de classificadores binários para validar a saída de uma rede neural *MLP*, o uso de redes neurais convolutivas LeNet5 e classificadores *SVM* em uma árvore de decisão desbalanceada. Nas seções seguintes serão discutidas as etapas da solução proposta para o problema do reconhecimento de dígitos manuscritos descritas no diagrama em blocos da Figura **16**.

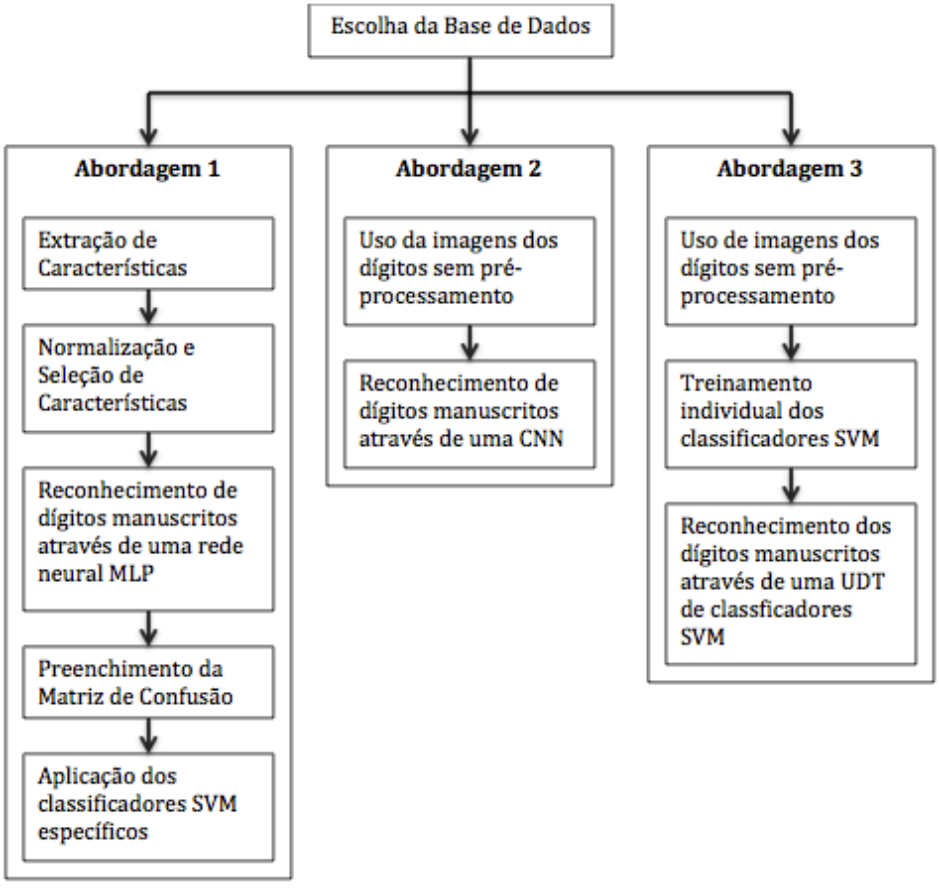

Figura 16 – Etapas da Metodologia utilizada neste trabalho

5.1 Escolha da base de dados

A base de dados MNIST foi escolhida para a realização desse trabalho conforme análise realizada na seção 3.1. Esta é uma base dados contendo 60.000 padrões de treinamento e 10.000 padrões de teste. Os padrões foram obtidos de aproximadamente 250 autores

diferentes. A base de dados é adequada para testes de algoritmo de aprendizagem para reconhecimento e classificação de padrões. Os números são armazenados em escala de cinza e normalizados para o tamanho 20×20 pixels, onde se encontram centralizados. A vantagem desta base é o fato de padrões não necessitarem de uma etapa de pré-processamento extensa ou complexa, sendo também amplamente conhecida e bem avaliada na literatura.

5.2 Abordagem do uso de classificadores específicos

Nesta abordagem é treinada uma rede MLP para distinguir as 10 classes de dígitos manuscritos, utilizando descritores de Fourier e a técnica de transição de borda como características de entrada. Como um estágio de classificação complementar é definido um conjunto de classificadores binários específicos para validar a saída da rede neural. As etapas necessárias no desdobramento dessa abordagem são explicadas nas subseções seguintes.

5.2.1 Extração de Características

Utilizando amostras contidas na base de dados MNIST de dígitos manuscritos, são extraídos um conjunto de características representativas de cada amostra. Cada conjunto é formado por 20 valores relativos aos Descritores de Fourier e 8 valores obtidos pela Técnica de transição de borda, totalizando 28 valores. Estas características foram escolhidas entre algumas levantadas na literatura, conforme pode ser visto na Tabela 2.

5.2.2 Normalização e Seleção das Características

Para selecionar um conjunto de características ótimo para o processo de reconhecimento utilizou-se o *Fisher's Discriminant Ratio* (*FDR*) associado à seleção escalar de características, conforme explanado na seção Seleção de Características. Para a seleção da primeira característica calcula-se o valor FDR para todas as características. Aquela com maior

valor FDR é selecionada. Em seguida, selecionam-se as demais características que melhor complementam a primeira. Para o processo de seleção das características seguintes, utiliza-se o conceito de seleção escalar de características apresentado anteriormente. Para este trabalho, com o propósito de obter o melhor conjunto de características que servirá de entrada à rede neural, serão testados conjuntos com números de características diferentes.

5.2.3 Reconhecimento dos dígitos através de uma rede neural de múltiplas camadas

Nessa etapa será utilizada uma rede neural de três camadas de propagação direta, treinada com o algoritmo de retropropagação, para o reconhecimento dos dígitos. Várias arquiteturas de rede serão testadas e será escolhida aquela que apresentar um melhor desempenho.

Para realizar o treinamento da rede neural, utilizou-se de um microcomputador com processador Intel Xeon 2.4 GHz e 24 GB de RAM, operando sob a plataforma Windows 7. Os scripts para treinamento e simulação da rede neural foram desenvolvidos parte em linguagem MATLAB, escritos e executados no programa MATLAB R2009a operando com a versão 1.6.0_22 da máquina virtual Java.

O conjunto de treinamento da rede será constituído por 60.000 amostras extraídas aleatoriamente da base de treinamento MNIST, um total 6.000 amostras por classe. O conjunto de teste da rede será constituído por 10.000 padrões de teste provenientes da base de teste MNIST, sendo 1.000 amostras por classe. Para cada amostra utilizada em ambos contextos de treinamento e teste, será feita a extração das características definidas na Seção 5.2.1, as quais servem de entrada para o classificador neural escolhido.

Testes iniciais realizados mostraram que a aplicação dessa rede não é suficiente para uma perfeita distinção entre os dígitos. Existem pares de dígito como 5 e 6, 2 e 9 e outros,

que, em muitos casos, são confundidos uns com os outros. Assim sendo, na próxima seção será mostrado como montar uma matriz de confusão, em função dos resultados dessa rede.

5.2.4 Preenchimento da matriz de confusão

A matriz de confusão é montada segundo mostrado na Figura **17**. As células dessa matriz são preenchidas utilizando o conjunto de teste da rede neural descrito na seção anterior. Assim, após a aplicação da rede neural anteriormente treinada no conjunto de teste, será determinado, por exemplo, quantas vezes o dígito 5 foi reconhecido pela rede como ele próprio ou como cada um dos outros dígitos (0, 1,2,3,4,6,7,8,9). O valor correspondente ao número de vezes que ele foi reconhecido como ele próprio será utilizado para preencher a célula mostrada na Figura **17** correspondente a interseção da linha 5 com a coluna 5. O valor correspondente ao número de vezes que ele foi reconhecido como o dígito 1 será utilizado para preencher a célula mostrada na Figura **17** correspondente a interseção da linha 5 com a coluna 1. E assim sucessivamente.

Determinada essa matriz de confusão, serão analisadas as células com maiores valores. As linhas e colunas das células que correspondem a esses maiores valores correspondem aos pares de dígitos que a rede neural mais se confundiu no reconhecimento. No sentido de obter um melhor reconhecimento desses pares de dígitos, serão treinados vários classificadores binários específicos, de forma a distinguir entre cada classe individual do restante, utilizando conjuntos de treinamentos distintos, um para cada rede. Esses conjuntos de treinamento serão constituídos por dois conjuntos, um das amostras pertencente a classe e outro do restante.

Dígito	1	2	3	4	5	6	7	8	9	0
1										
2										
3										
4										
5										
6										
7										
8										
9										
0										

Figura 17 – Matriz de confusão

5.2.5 Aplicação de classificadores binários específicos para distinção de uma classe versus restante

Afim de complementar a rede neural apresentada na Seção 5.2.4, os classificadores específicos utilizados nessa etapa foram binários, onde cada um é especializado em uma classe diferente, assim projetados para fornecer apenas dois valores de saída, positivo caso o padrão de amostras pertença a classe a qual é especialista, ou negativo caso essa amostra pertence ao conjunto das demais classes.

Com intuito de realizar uma comparação de desempenho entre diferentes classificadores, realizou-se duas configurações de experimentos:

- O primeiro em que todos os classificadores específicos foram implementados cada um como uma rede neural MLP com algoritmo backpropagation.
- O segundo com somente classificadores *SVM* binários específicos associados com o Kernel RBF para validar a saída do classificador principal

A etapa de treinamento foi realizada com uma parcela da base de treinamento, sendo 6000 para a classe que representa o a saída positiva e 6000 tomados aleatoriamente das classes restantes representando a saída negativa. O número de classificadores utilizados nessa

etapa é igual ao número de classes de dígitos, ou seja, um total de 10 classificadores. Esses classificadores foram denominados de C_i, em que $0 \leq i \leq 9$.

A interação com o classificador principal, a rede neural MLP, dá-se através da saída da mesma. A cada valor de saída i a amostra corrente é apresentada ao classificador C_i específico da classe i identificada pela rede. Em caso de confirmação, o classificador C_i retorna uma saída positiva e o processo de decisão termina resultando na classe i. Caso a saída de C_i seja negativa, a amostra corrente é apresentada ao classificador seguinte C_{i+1} e assim sucessivamente até que o último classificador C_9 é responsável por rotular a imagem ou como sendo da classe 9 ou como sendo um erro de identificação. Espera-se que a utilização desses classificadores específicos possa aumentar de forma significativa o desempenho do sistema de reconhecimento de dígitos proposto.

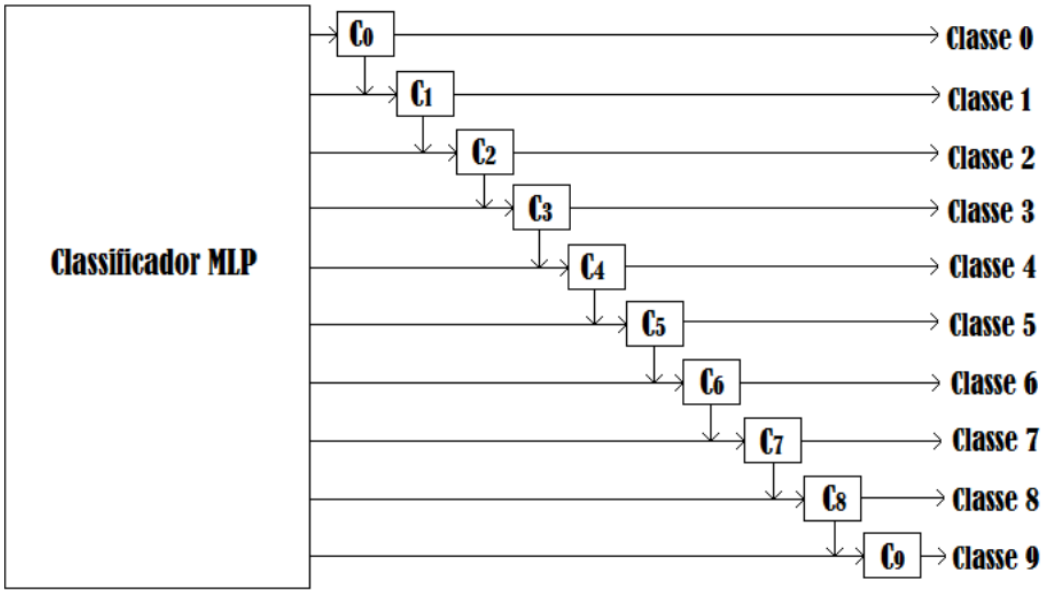

Figura 18 - Esquema de validação de saída da rede neural MLP

5.3 Abordagem do uso de Redes Convolutivas

Nesta abordagem foram utilizadas Redes Neurais Convolutivas para o reconhecimento dos dígitos manuscritos, método escolhido devido à alta taxa de sucesso apontada na literatura (CIRESAN, MEIER e SCHMIDHUBER, 2012), onde o melhor resultado de 0,27% utiliza um grupo de 7 classificadores convolutivos, e utilizando versões modificadas da base de treinamento. Neste trabalho, pela finalidade comparativa em termos de desempenho, foi empregada a versão mais simplificada da Rede Convolutiva, a *LeNet5* proposta em LeCun, Bottou, *et al.* (1998).

A arquitetura da *CNN* (Rede Neural Convolutiva - *Convolutional Neural Network*) segue a descrição da Figura **11**. Um ponto positivo para a implementação é o fato do compartilhamento de pesos entre várias conexões de neurônios auxiliar na redução do número de parâmetros livres, o que pode implicar em um treinamento mais rápido do que uma rede neural MLP.

A implementação utilizou a biblioteca *Theano*, uma biblioteca para *deep learning* (aprendizagem profunda) feita na linguagem de programação *Python*. A decisão de migrar para o desenvolvimento em *Python* foi motivado por questões de desempenho e por um uso menor de memória comparado ao mesmo experimento utilizando redes neurais MLP rodando em MATLAB.

O desenvolvimento desse *benchmarking* será feito por meio do código disponibilizado em LeNet, onde serão extraídas as matrizes representando as imagens da base de treinamento de dígitos manuscritos do MNIST. A classificação dos dígitos é implementada por meio de um regressor logístico ao invés de uma rede RBF.

Para aferir o resultado do treinamento do classificador são utilizadas as amostras da base de teste da MNIST, 10000 exemplos de imagens isoladas de dígitos manuscritos. Os

parâmetros escolhidos para analisar o desempenho são a taxa de acerto, ou seja, a porcentagem de amostras corretamente identificadas, e o tempo total de treinamento.

5.4 Abordagem do uso de classificadores *SVM* em uma árvore de decisão desbalanceada (UDT – Unbalanced Decision Tree)

A terceira abordagem envolve utilizar um conjunto de classificadores *SVM* com Kernel RBF estruturados em uma árvore de decisão desbalanceada (RAMANAN, SUPPHARANGSAN e NIRANJAN, 2007), utilizada com sucesso na literatura (HASSAN e DAMPER, 2012) em tarefas como reconhecimento de emoções na fala.

A implementação foi feita por meio da biblioteca de algoritmos de aprendizagem *scikit-learn* (Support Vector Machines) em *Python*. O desenvolvimento dos scripts seguiu uma abordagem de orientação a objetos com a modelagem por meio de diagrama de classes como pode ser visto na Figura **19**. *Classifier* representa a classe mais abstrata para representar os classificadores e define as implementação do geração da matriz de confusão. *NeuralClassifier* e *SVMClassifier* são, respectivamente, as classes dos classificadores de Redes Neurais e *SVM*, cada qual com sua definição de saída e algoritmo de treinamento de seus classificadores. Para lidar com um classificador binário de maneira genérica foi desenvolvida a classe *ModularClassifier*, que implementa métodos privados de particionamento da base de dados. *ClassifierSet* representa um conjunto de classificadores, possuindo duas classes filhas: *ModularClassifierSet* que implementa o comportamento de uma árvore de decisão desbalanceada utilizando várias instâncias de *ModularClassifier* e *SupportClassifier* que reproduz o comportamento de validação descrito na seção 5.2.5.

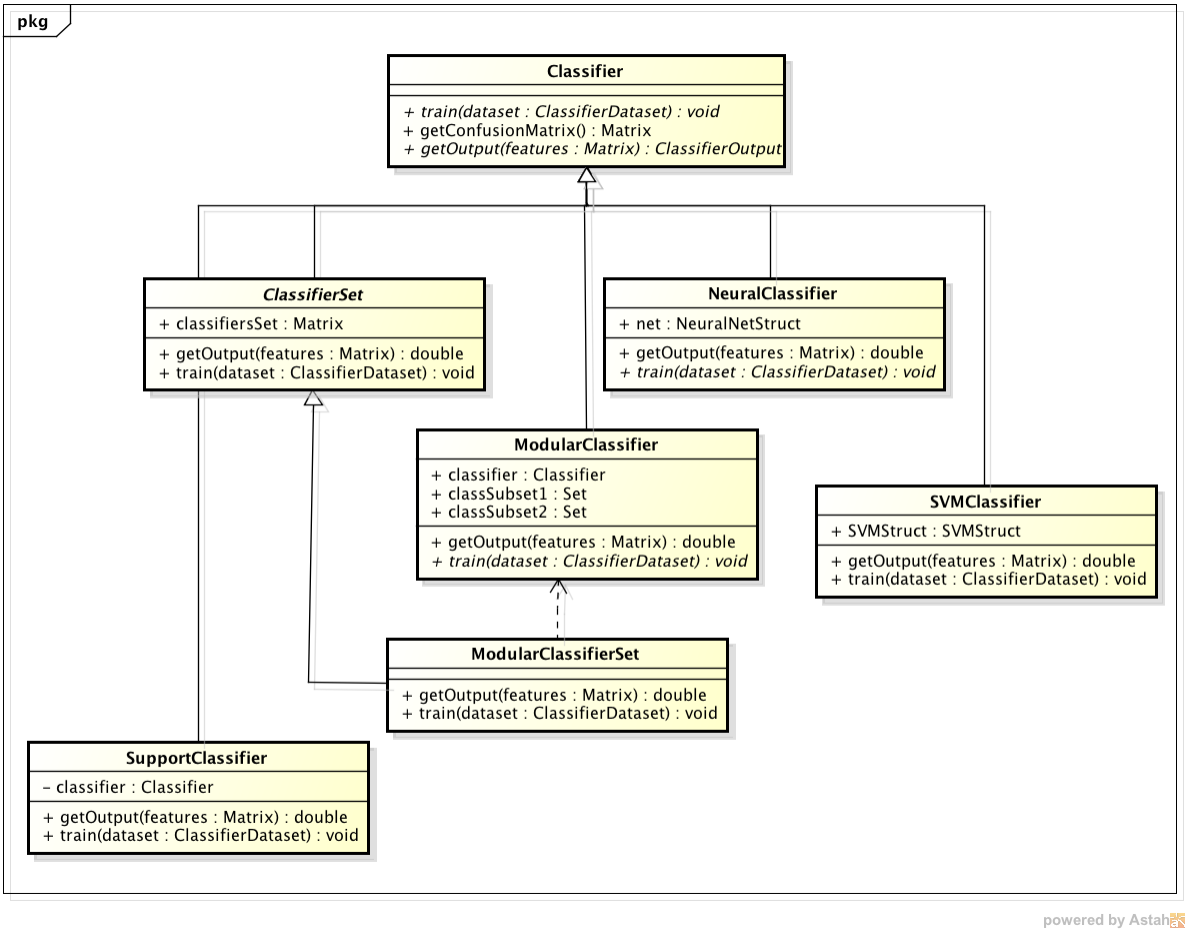

Figura 19 – Diagrama de Classes da implementação dos experimentos em *Python*

Para cada classificador C_i, a base de treinamento é particionada de forma a definir dois conjuntos, um conjunto positivo relativo as amostras da classe i e outro conjunto negativo representando as amostras de todas as classes restantes. De forma a não prejudicar a capacidade de generalização dos classificadores, cada conjunto recebe a mesma quantidade de amostras. Dado que a base de MNIST possui um total de 60.000 amostras, em torno de 6.000 para cada classe, apenas 12.000 são utilizados para o treinamento de cada classificador. Cerca de 660 amostras são escolhidas aleatoriamente de cada classe restante para compor o conjunto de saída negativa, totalizando 6.000 amostras. Como saída positiva são utilizadas as amostras de uma única classe, também 6.000 amostras.

De maneira a calcular a taxa de acerto do treinamento do conjunto de classificadores são utilizadas as 10.000 amostras da base de teste da MNIST. Dessa maneira, são 1.000

amostras por classe para validar cada saída do árvore de decisão. No teste de desempenho de cada classificador, 1.000 amostra são associadas ao padrão positivo e 9.000 amostras das classes restantes compõem o padrão negativo.

6 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados os resultados e discussões acerca do processo de treinamento de classificadores para o reconhecimento de dígitos manuscritos utilizando a base de dados da MNIST. Para cada estratégia definida anteriormente, uma determinada configuração de arquitetura de classificador ou conjunto de classificadores foi treinada e são listados os resultados obtidos, tais como taxa de acerto e matriz de confusão.

Nas seções seguintes serão abordadas as soluções desenvolvidas: classificadores específicos para validar a saída de uma rede neural MLP, redes neurais convolutivas utilizando imagens de dígitos sem pré-processamento, classificadores SVM utilizando imagens de dígitos e uma arquitetura de árvore de decisão desbalanceada.

6.1 Abordagem do uso de classificadores específicos

Essa estratégia de classificação é composta de três camadas: a primeira consiste da extração de características; a segunda consiste da seleção de características e, a terceira, da classificação de dígitos. Na primeira camada são extraídas como características os Descritores de Fourier, complementados pela técnica de transição de borda. Conforme explicado na seção 5.2.2 um passo seguinte é realizar a seleção de características. De forma complementar a seleção de características foi realizado um experimento para relacionar o número máximo de características com o desempenho do classificador. A Tabela **3** mostra as taxas de erro para duas variações de arquitetura e três características escolhidas para o treinamento de uma rede neural MLP, utilizando um número fixo de épocas. Pelos resultados obtidos, a arquitetura e quantidade máxima de características que resultou em melhores resultados de classificação, foi, respectivamente, 60-40-10 e 9 valores de Descritores de Fourier.

Tabela 3 – Taxa de Erro de classificação versus Arquitetura da rede neural e número de características selecionadas: experimentos preliminares

	Número de características		
Arquitetura	**13**	**11**	**9**
60-60-10	*58,31%*	*58,03%*	*57,43%*
60-40-10	*55,87%*	*56,01%*	*55,63%*

Seguindo a estrutura da Figura **18**, são definidas duas subetapas do processo de decisão: uma rede neural MLP inicial, que consiste do método principal de classificação, seguida de classificadores específicos, que servem como uma etapa de validação do resultado da rede MLP. No método principal de classificação é treinada uma rede neural MLP com o algoritmo *backpropagation*. Sucessivos experimentos foram realizados na busca pela arquitetura com melhor desempenho para o problema. A Tabela **4** demonstra que a melhor taxa de acerto obtida com as nove melhores características obtidas nos experimentos preliminares foi obtida com a arquitetura "20-30-10".

Tabela 4 – Valores da taxa de acerto do classificador MLP para diferentes arquiteturas

Arquitetura	**Taxas de Acerto**	
	Treinamento	**Teste**
20-30-10	83,50%	79,28%
30-40-10	86,86%	77,95%
30-32-10	87,44%	77,41%
40-50-10	84,88%	74,55%
20-60-10	80,57%	73,39%
30-75-10	80,45%	67,54%
30-100-10	88,32%	65,84%

Para essa arquitetura de rede neural *MLP* foi obtido a taxa de acerto de *79,28%* para o conjunto de teste. Mesmo a arquitetura escolhida de '20-30-10' não conseguiu ser, portanto, totalmente eficaz na tarefa de discriminação de todas as classes, posto que muitos exemplos da literatura já trazem acertos muito superiores a 80%.

A Figura **20** representa um terminal MATLAB executando um *script* para calcular a matriz de confusão. Na diagonal principal da matriz se encontra o número de acertos da rede neural para cada classe, ou seja, na coluna 0, representando a classe 0 ocorreram 815 acertos, na classe 1 foram 855 acertos, e assim em diante. Os números em cada coluna que não fazem parte da diagonal principal correspondem aos casos de erro. Dessa forma, para a classe 0, somando-se todos os outros números da coluna 0, temos, somente para esta classe, 185 erros de classificação.

Analisando a matriz da Figura **20**, percebe-se que não houve grandes erros localizados especificamente em uma mesma classe. Os erros se encontram distribuídos ao longo de todas as colunas, indicando uma falha de generalização do classificador para todas as classes.

```
>> confusion_matrix

confusion_matrix =

   815    15     9    36    11    34    23    23    18    14
    28   855    13    39    26    38    40    29    26     7
     6     3   898    34    43    23    58    15    11    17
    21    33     9   736    20    51    30    19    13    19
    25     0     7    26   771    18    59    16    10    30
    18    11    12    18     3   704    35    27    19    12
    34    21     9    16    30    35   657    12    11    16
     7    28     8    27    48    28    36   802    19    31
    23    20    14    43    24    32    35    43   852    16
    23    14    21    25    24    37    27    14    21   838
```

Figura 20 - Matriz de confusão para a rede MLP principal.

A segunda etapa dessa abordagem é a camada de validação, que valida a saída da rede neural MLP, com a intenção de reduzir o número de padrões classificadores incorretamente. Para esse fim foram treinadas 10 redes neurais, cada uma específica para a tarefa de distinguir uma classe de dígito das demais, de acordo com a arquitetura definida na Figura **15**. A Tabela **5** lista as taxas de acerto para cada classificador *MLP*. O uso destes classificadores não

resultou em aumento da taxa de acerto. Infelizmente, a etapa de validação não foi bem sucedida, devido ao baixo desempenho do treinamento individual dos classificadores específicos.

Tabela 5 - Taxas de acerto das redes específicas MLP

	Taxas de Acerto	
Classificador	**Treinamento**	**Teste**
C0	96,83%	94,52%
C1	88,25%	82,47%
C2	78,65%	76,87%
C3	84,53%	82,94%
C4	89,77%	87,76%
C5	79,83%	74,64%
C6	73,63%	71,58%
C7	86,58%	71,44%
C8	89,75%	86,37%
C9	77,43%	69,56%

Já na Tabela **6** são listadas as taxas de acerto obtidas com classificadores *SVM* específicos para cada classe de dígito manuscrito. Utilizando este conjunto de classificadores foi possível aumentar a taxa de acerto média para 82,36%.

Tabela 6 - Taxas de acerto das redes específicas SVM

	Taxas de Acerto	
Classificador	**Treinamento**	**Teste**
C0	97,43%	93,50%
C1	98,62%	96,56%
C2	90,31%	85,31%
C3	90,25%	91,44%
C4	93,97%	90,50%
C5	96,59%	93,87%
C6	93,63%	91,63%
C7	96,49%	92,94%
C8	89,75%	89,56%
C9	94,42%	90,28%

A matriz de confusão da Figura **21** exibe o melhor desempenho obtido com os classificadores *SVM*. Percebe-se que as primeiras classes tiveram resultados melhores que as últimas classes, provavelmente devido à natureza iterativa da abordagem proposta, em que cada classificador específico impõe um erro acumulativo para as classes seguintes.

```
>>    ConfusionMatrix

    ConfusionMatrix =

   739     0     7     1     8     0    21     8    14     2
     2   747    15     0     6     2     9     3     8     8
     8     0   685    18    17     4    44     8     7     9
     5     0    75   665     2     7    11    10    17     8
    12     5    19     0   692     3    10     6    10    43
     3     2    27    11     8   669    34     7    22    17
    12    12    18     5    11    20   694     4    16     8
     4     7    29    10    12     2     8   698    13    17
    31    10    28    20    17     8    36     8   585    57
     4    11    18    10    44     7     4    22    42   638
```

Figura 21 - Matriz de confusão para rede MLP com validação das redes específicas calculada no ambiente MATLAB

6.2 Abordagem do uso da Rede Neural Convolutiva (CNN)

Na segunda abordagem, foram descartados os vetores de características compostos pelos Descritores de Fourier, e foram utilizadas somente as imagens puras ou sem pré-processamento. Assim para todas as amostras de treinamento e teste, as características que alimentaram o processo de classificação foram as próprias imagens representadas por suas matrizes de valores binários.

O classificador a ser utilizado nesta abordagem foi uma rede neural convolutiva (*CNN*) com arquitetura LeNet5 (LECUN, BOTTOU, *et al.*, 1998). Sua implementação foi realizada na linguagem *Python* através da biblioteca *Theano*. Sua execução requisitou 1GB de memória, em uma máquina virtual Linux rodando em computador Windows Vista. Foram

necessárias 12hs para treinar a rede utilizando biblioteca de aprendizado e otimização *scikit-learn*.

Utilizando este método foi possível obter 99,1% de taxa de acerto na base de teste dos dígitos manuscritos. A Figura **22** mostra a confusão de classificação por meio da matriz que relaciona as classes de saída com as classes das amostras, a assim chamada matriz de confusão. Uma análise subjetiva em cima da matriz aponta que o número de confusão é consideravelmente menor que os erros apontados pela matriz de confusão da Figura **21**.

```
>>> confusion_matrix
array([[ 975,    0,    2,    0,    0,    0,    1,    1,    1,    0],
       [   0, 1131,    1,    1,    0,    1,    1,    0,    0,    0],
       [   0,    2, 1026,    0,    1,    0,    0,    2,    1,    0],
       [   0,    0,    2, 1004,    0,    2,    0,    1,    1,    0],
       [   0,    0,    0,    0,  976,    0,    1,    0,    1,    4],
       [   2,    0,    0,    6,    0,  882,    1,    1,    0,    0],
       [   6,    1,    0,    1,    1,    4,  944,    0,    1,    0],
       [   0,    1,    4,    2,    1,    0,    0, 1018,    1,    1],
       [   3,    0,    4,    1,    1,    0,    0,    1,  963,    1],
       [   2,    3,    0,    1,    7,    3,    0,    2,    0,  991]])
```

Figura 22 - Matriz de confusão para a rede neural convolutiva

6.3 Validação do uso de classificadores *SVM* em uma árvore de decisão desbalanceada (UDT – Unbalanced Decision Tree)

Na terceira abordagem, também são utilizadas as imagens dos dígitos como características de entrada para a camada de classificação. Em sequência foi utilizada também uma árvore de decisões desbalanceada, tal como esquematizada na Seção 5.4. A modelagem adotada seguiu uma metodologia *top-down*, tratando o problema de reconhecimento de padrões de maneira abstrata por meio de classes genéricas e só então foi implementado especificamente a interação entre classificadores tal como sugere a Figura **19**.

O código necessário foi desenvolvido em *Python* fazendo uso da biblioteca *scikit-learn* para o treinamento e teste de desempenho dos classificadores *SVM*. O tempo total de treinamento dos 9 classificadores foi de 6 horas rodando em uma máquina virtual Ubuntu.

A Tabela **7** lista todos os resultados do treinamento dos classificadores *SVM* específicos. Cada classificador recebeu como base de dados, uma versão modificada da base MNIST, separando a em dois conjuntos, um referente a cada classe, e outro de mesmo tamanho com uma seleção arbitrária das amostras pertencentes as outras classes. Conclui-se assim que cada classificador conseguiu efetivamente distinguir cada classe individualmente das demais.

Tabela 7 - Taxas de acerto dos classificadores *SVM* da UDT

	Taxas de Acerto	
Classificador	**Treinamento**	**Teste**
C0	100%	100%
C1	100%	100%
C2	100%	100%
C3	100%	100%
C4	100%	100%
C5	100%	100%
C6	100%	100%
C7	100%	100%
C8	100%	100%

Os classificadores resultantes então são agrupadas hierarquicamente em uma árvore de decisão desbalanceada tal como demonstra a imagem (d) da Figura **15**. Para cada amostra recebida, o conjunto de classificadores inicia um processo de avaliação iterativo, em que a cada passo um classificador distingue entre a sua classe de especialidade e as demais classes, caso a amostra não seja da classe da qual é especialista, no passo seguinte é avaliada a próxima classe, até restar no final apenas duas classes a serem avaliadas. Esse método de avaliação permite reduzir o número de classificadores, considerando N como o número de classes, a redução então é de N para $N - 1$, além de fornecer uma resposta média mais rápida,

devido ao fato de que para obter o resultado é necessário a avaliação de no mínimo um classificador e no máximo $N - 1$.

Analisando a matriz de confusão gerada para a *UDT* na Figura **23**, percebe-se que os valores de saída concentram-se unicamente na diagonal da matriz, ou seja não ocorre confusão na classificação das diferentes classes das amostras. Assim, o conjunto de classificadores logrou sucesso no reconhecimento de todas as amostras, possibilitando notar também que para a base de teste do MNIST não existe um mesmo número de amostras para todas as classes.

```
>>> modular_classifier.getConfusionMatrix(mnist_experiments.MnistData.TestingData)
array([[ 980,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0, 1135,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1032,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 1010,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,  982,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,  892,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,  958,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1028,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  974,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0, 1009]])
>>> modular_classifier.testingHit
1.0
```

Figura 23 - Matriz de confusão e taxa de acertos para o uso de classificadores modulares *SVM* como uma *UDT*

6.4 Comparação de resultados com a literatura

Na literatura o melhor resultado envolvendo Descritores de Fourier e redes neurais MLP atinge um valor de 97,15% em Cho (1997) para uma base de dígitos manuscritos coletados do serviço postal americano, em que se diferencia da primeira abordagem proposta aqui por contar com 15 descritores complexos de Fourier em adição ao uso de máscaras de Kirsch para o treinamento de um conjunto de redes MLP.

Em se tratando do uso de redes neurais convolutivas (*CNN*), a literatura aponta como referência de maior destaque o trabalho de Ciresan, Meier e Schmidhuber (2012) com

resultado de taxa de acerto de 99,77%, devido ao uso de um conjunto de classificadores treinados na GPU com geração de amostras adicionais através de técnicas de distorção das imagens da base MNIST.

O uso de classificadores *SVM* na base MNIST tem seu melhor resultado em Madzarov, Gjorgjevikj e Chorbev (2009), com taxa de acerto de 97,54%. Utiliza um conjunto de classificadores *SVM* em uma árvore de decisão binária (*BDT*) e como características, são extraídas projeções verticais, horizontais e diagonais das imagens dos dígitos. O *SVM-BDT* se distingue da árvore de decisão desbalanceada (*UDT*) pela abordagem de balancear a árvore de classificadores onde os nós iniciais filtram entre dois grandes grupos de classes, os nós seguintes entre grupos menores, até chegar os nós folha que separam entre duas classes individuais, como pode ser visto na Figura **6**. A vantagem da *UDT* em relação a *BDT* é permitir que cada classificador se especialize em uma única classe, permitindo assim uma maior generalização.

Uma vez que cada classificador da *UDT* foi capaz de atingir um desempenho de 100% é possível afirmar que cada ponto da árvore de decisão, realiza um processo decisório com taxa de erro de 0%, o que significa que o erro acumulado de todos os classificadores também é 0%. Assim obtivemos um método de máxima taxa de acerto para a base MNIST de dígitos manuscritos, com erro inferior a 0,23%, melhor resultado publicado na literatura Ciresan, Meier e Schmidhuber (2012).

7 CONCLUSÃO

Neste trabalho foram propostas três abordagens na busca por um classificador de máximo desempenho no problema de reconhecimento de dígitos manuscritos utilizando a base de dados MNIST. A primeira abordagem utilizou Descritores de Fourier como vetor de característica, uma rede neural *MLP* e uma etapa de validação através de classificadores modulares específicos. A segunda abordagem já utiliza redes convolutivas e as próprias imagens dos dígitos manuscritos como características de entrada para classificação. A terceira abordagem faz uso de um conjunto de classificadores *SVM* em uma estrutura hierárquica de árvore de decisão desbalanceada.

A primeira abordagem teve o objetivo de adicionar uma camada de validação a um sistema de reconhecimento de caracteres. Nesse sistema foram incluídas várias etapas: extração dos Descritores de Fourier como características, normalização de características e seleção de características afim de aumentar o desempenho do classificador, classificação das amostras por meio de uma rede neural *MLP* de três camadas treinada com o algoritmo backpropagation. Em uma etapa de validação, um conjunto de classificadores modulares binários foram utilizados para validar cada saída da rede *MLP*.

Redes Neurais Convolutivas é o método de maior destaque na literatura para reconhecimento de dígitos manuscritos, sendo que a maior taxa de acerto registrada é de um conjunto de classificadores convolutivos em Ciresan, Meier e Schmidhuber (2012). A segunda abordagem proposta é a da utilização de um classificador convolutivo como *benchmarking* quanto ao uso da base de dados e métodos propostos nas outras abordagens.

A terceira abordagem é o uso de uma árvore de decisão desbalanceada cujos nós de decisão representam classificadores *SVM* binários. Cada classificador realiza um particionamento do universo problema de tal forma a gerar para cada classe um hiperplano de separação, representado pelo classificador *SVM*.

Por meio de uma única rede neural *MLP* foi obtido uma taxa de erro de 0,2074% com uma arquitetura 20-30-10. Ao utilizar a etapa de validação com classificadores *SVM* binários foi possível reduzir essa taxa de erro para 0,1477%. A baixa taxa de acerto comparada com a literatura pode ser um indicativo de que as características de Descritores de Fourier não possuem suficiente efetividade na representação de dígitos manuscritos.

Rede Neural Convolutiva é um método de *deep learning* amplamente conhecido e com vários exemplos na literatura de boas taxas de reconhecimento para a base MNIST de dígitos manuscritos. Neste trabalho, a aplicação de um classificador convolutivo *LeNet5* alcançou na base de teste, uma taxa de erro de 0,9%, muito inferior à obtida com a primeira abordagem, destacando o como um bom método de benchmarking para os classificadores obtidos neste trabalho.

Desenvolvido a poucos anos, o método de Árvores de Decisão Desbalanceada (*UDT*) fornece uma estrutura simples para o processo de decisão multiclasse utilizando classificadores binários. Ao adotar classificadores *SVM* em uma arquitetura *UDT* foi possível obter taxa de erro de 0% na base MNIST, a melhor taxa de reconhecimento que se tem notícia atualmente. O sucesso deste método foi atingido pela alta capacidade de reconhecimento obtido no treinamento dos classificadores *SVM*, em que cada um conseguiu 100% de taxa de acerto nas amostras de teste. Uma desvantagem a ser notada para essa abordagem, é o grande número de vetores de suporte utilizados, demandando uma quantidade grande de dados estarem disponíveis no momento da classificação, cerca de 1 GB.

7.1 Trabalhos Futuros

A proposta deste trabalho lança alguns pontos de exploração para futuros trabalhos. Modificações na estruturação de classificadores *SVM*, métodos de geração de amostras baseados em distorções para melhorar o desempenho da rede convolutiva e treinamento da

arquitetura de validação utilizando uma base de validação, são alguns exemplos de técnicas empregadas na literatura que podem ser aplicadas como ramificações dos métodos propostos.

A arquitetura *UDT* pode ser explorada em relação a flexibilidade em permitir o uso de qualquer classificador binário, assim podem ser usados redes neurais, redes neurais convolutivas, classificadores *SVM*, ou seja, diferentes métodos de classificação. O uso de classificadores com um número menor de parâmetros pode contribuir com uma solução mais leve e portável para diferentes cenários e plataformas.

O sucesso da proposta também abre margens para trabalhos comparativos com redes neurais convolutivas, sendo esta a técnica vigente dos últimos trabalhos que apresentam as maiores taxas de acerto LeCun, Cortes e Burges (1998).

O método de árvore de decisão desbalanceada também pode ser aplicado em outros domínios de problemas, semelhantes ao do reconhecimento de caracteres manuscritos. Em um cenário de grande número de classes, a estrutura de árvore de decisão pode melhor gerenciar o processo decisório, pois simplifica o treinamento de cada classificador individual ao reduzir o escopo de generalização, contribuindo assim para taxas de acerto maiores.

REFERÊNCIAS BIBLIOGRÁFICAS

BURGES, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. **Data Mining and Knowledge Discovery**, Hingham, MA, USA, v. 2, n. 2, p. 121 - 167, 1998.

CHEN, J.; WANG, C.; WANG, R. Adaptive binary tree for fast SVM multiclass classification. **Neurocomputing**, 2009.

CHERIET, M. et al. **Character Recognition Systems**. Hoboken, New Jersey: Wiley, 2007.

CHO, S.-B. Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals. **Neural Networks, IEEE Transactions on**, p. 43-53, 1997.

CHUNG, ; WONG,. **Handwritten character recognition by Fourier descriptors and neural network**. TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE. [S.l.]: [s.n.]. 1997. p. 391 - 394.

CIREşAN, D. C. et al. **Flexible, high performance convolutional neural networks for image classification**. Proceedings of the Twenty-Second international joint conference on Artificial Intelligence. Barcelona, Catalonia, Spain: AAAI Press. 2011. p. 1237-1242.

CIRESAN, D. C. et al. **Convolutional Neural Network Committees for Handwritten Character Classification**. Document Analysis and Recognition (ICDAR), 2011 International Conference on. [S.l.]: [s.n.]. 2011. p. 1135 -1139.

CIRESAN, D.; MEIER, U.; SCHMIDHUBER, J. **Multi-column Deep Neural Networks for Image Classification**. Dalle Molle Institute for Artificial Intelligence. Manno, Switzerland. 2012.

CONVOLUTIONAL Neural Networks (LeNet). **Deep Learning**. Disponivel em: <http://deeplearning.net/tutorial/lenet.html>. Acesso em: 14 Março 2014.

DECOSTE, D.; SCHÖLKOPF, B. **Training Invariant Support Vector Machines**. The Netherlands, p. 161-190. 2002.

DEDGAONKAR, S. G.; CHANDAVALE, A. A.; SAPKAL, A. M. Survey of Methods for Character Recognition. **International Journal of Engineering and Innovative Technology (IJEIT)**, 2012.

DENG, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research. **IEEE Signal Processing Magazine**, 10 November 2012. 141-142.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. Third Edition. ed. [S.l.]: Pearson Prentice Hall, 2008.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. **Digital Image Processing Using Matlab**. [S.l.]: Gatesmark Publishing, v. 2, 2009.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H. **Neural Network Design**. [S.l.]: PWS Pub., 1995.

HARRINGTON, P. **Machine Learning in Action**. Shelter Island: Manning Publications, 2012.

HASSAN, A.; DAMPER, R. I. Classification of emotional speech using 3DEC hierarchical classifier. **Speech Communication**, n. 54, p. 903-916, 2012.

HRYCEJ, T. **Modular Learning in Neural Networks:** A Modularized Approach to Neural Network Classification. New York: John Wiley & Sons, Inc., 1992.

JARRETT, K. et al. **What is the best multi-stage architecture for object recognition?** Computer Vision, 2009 IEEE 12th International Conference on. [S.l.]: [s.n.].

JORDAN, M. I.; JACOBS, R. A. **A competitive modular connectionist architecture**. Advances in Neural Information Processing Systems. San Maeto, CA: Morgan Kaufmann Publisher Inc. 1991. p. 767 - 773.

KAPP, M. N. et al. **Evaluating the conventional and class-modular architectures feedforward neural network for handwritten word recognition**. Computer Graphics and Image Processing, 2003. SIBGRAPI 2003. XVI Brazilian Symposium on. [S.l.]: [s.n.]. 2003. p. 315 - 319.

KUSSUL, E.; BAIDYK, T. **Improved method of handwritten digit recognition tested on MNIST database**. 15th International Conference on Vision Interface. [S.l.]: Elsevier. 2004. p. 971-981.

LATHI, B. P. **Sinais e Sistemas Lineares**. 2. ed. Porto Alegre: Bookman, 2007.

LE, H. M.; DUONG, A. T.; TRAN, A. **Multiple-Classifier Fusion Using Spatial Features for Partially Occluded Handwritten Digit Recognition**. 10th International Conference ICIAR. Póvoa do Varzim, Portugal: Springer Berlin Heidelberg. 2013. p. 124-132.

LECUN, Y. et al. **Gradient-Based Learning Applied to Document Recognition**. Proceeding of the IEEE. [S.l.]: [s.n.]. 1998. p. 2278-2324.

LECUN, Y.; CORTES, C.; BURGES, C. J. C. The MNIST database of Handwritten Digits. **The MNIST database of Handwritten Digits**, 31 December 1998. Disponivel em: <http://yann.lecun.com/exdb/mnist/>. Acesso em: 2 January 2013.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. **Convolutional networks and applications in vision**. Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. [S.l.]: [s.n.]. 2010. p. 253-256.

LI, B.-Q.; LI,. **Building pattern classifiers using convolutional neural networks**. Neural Networks, 1999. IJCNN '99. International Joint Conference on. [S.l.]: [s.n.]. 1999. p. 3081-3085.

MADZAROV, G.; GJORGJEVIKJ, D.; CHORBEV, I. A Multi-Class SVM Classifier Utilizing Binary Decision Tree. **Informatica**, p. 225-233, 2009.

MANTAS, J. An overview of character recognition methodologies. **Pattern Recognition**, v. 19, n. 6, p. 425 - 430, 1986.

MASMOUDI, M. et al. **A hardware implementation of neural network for the recognition of printed numerals**. Microelectronics, 1999. ICM '99. The Eleventh International Conference on. [S.l.]: [s.n.]. 2000. p. 113 - 116.

MATLAB Image Processing Toolbox User Guide. Disponivel em: <http://www.mathworks.com/help/pdf_doc/images/images_tb.pdf>. Acesso em: 10 Outubro 2010.

MATLAB Neural Network Toolbox User Guide. Disponivel em: <http://www.mathworks.com/help/pdf_doc/nnet/nnet.pdf>. Acesso em: 15 Novembro 2010.

MORNS, I. P.; DLAY, S. S. **The dynamic supervised forward-propagation neural network for handwritten character recognition using Fourier descriptors and incremental training**. Electronics, Circuits, and Systems, 1996. ICECS '96., Proceedings of the Third IEEE International Conference on. Rodos, Greece: [s.n.]. 1996. p. 1123 - 1126.

OH, I.-S.; SUEN, Y. A class-modular feedforward neural network for handwriting recognition. **Pattern Recognition**, v. 35, n. 1, p. 229 - 244, 2002.

POON, J. C. H.; MAN, G. M. T. **An enhanced approach to character recognition by Fourier descriptor**. Singapore ICCS/ISITA '92. 'Communications on the Move'. Cingapura: [s.n.]. 1992. p. 558 - 562.

RAMANAN, A.; SUPPHARANGSAN, S.; NIRANJAN, M. **Unbalanced Decision Trees for Multi-class Classification**. International Conference on Industrial and Information Systems. Sri Lanka: [s.n.]. 2007.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence:** A Modern Approach. 3rd. ed. New Jersey: Pearson, 2010.

SERRE, T. et al. A quantitative theory of immediate visual recognition. **Progress in Brain Research. Computational Neuroscience: Theoretical Insights into Brain Function.**, 2007.

SUPPORT Vector Machines. **Scikit Learn**. Disponivel em: <http://scikit-learn.org/stable/modules/svm.html>. Acesso em: 14 mar. 2014.

THEODORIDIS, K. **Pattern Recognition**. [S.l.]: John Wiley & Sons, Inc., 2007.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. 4. ed. San Diego, California: Elsevier, 2009.

TRAVIESO, C. M.; ALONSO, J.; FERRER, M. A. **Combining different off-line handwritten character recognizers**. INES 2011, 15th International Conference on Intelligent Engineering Systems. Propad, Slovakia: [s.n.]. 2011.

VAMVAKAS, G.; GATOS, B.; PERANTONIS, S. J. Handwritten character recognition through two-stage foreground sub-sampling. **Pattern Recognition**, 2010.

APÉNDICES

Apéndice A – Artigo submetido para "*International Conference on Network and Service Management 2014*"

Handwritten Digit Recognition Using SVM Binary Classifiers and Unbalanced Decision Trees

Adriano Mendes Gil[2], Cícero Ferreira Fernandes Costa Filho[2], Marly Guimarães Fernandes Costa[1]

[1]*Instituto Nokia de Tecnologia*, Manaus, Brazil
adrianomendes.gil@gmail.com

[2]*Universidade Federal do Amazonas/Centro de Pesquisa Desenvolvimento em Tecnologia Eletrônica e da Informação* – UFAM/CETELI, Manaus, Brazil
ccosta@ufam.edu.brmailto:cffcfilho@gmail.com, mcosta@ufam.edu.br

Abstract. In this work, we use SVM binary classifiers coupled with a binary classifier architecture, a unbalanced decision tree, for handwritten digit recognition. According to input variables, two classifiers were trained and tested. One using digit characteristics and the other using the whole image as input variables. Developed recently, the unbalanced decision tree architecture provides a simple structure for a multiclass classifier using binary classifiers. In this work, using the whole image as input, 100% handwritten digit recognition accuracy was obtained in the MNIST database. These are the best results published in the literature for the MNIST database.

Keywords: Handwritten digit recognition, MNIST database, support vector machine, unbalanced decision tree, binary classifiers.

1 Introduction

In recent decades, character recognition technology has been driven by the increasing demand of converting an enormous amount of printed or handwritten information to a digital format [1]. This conversion from paper to computer in the past required human operators who processed billions of checks, mail correspondence, etc. This process was time consuming and error prone, motivating the development of optical character recognition (OCR), a technique for reading data and recognizing one character after another. OCR is an important pattern recognition technique. There are vast amounts of historical, technical and economic documents only in a printed form. An OCR system drastically reduces cost of digitalizing them. There are some successful techniques for OCR implementation applied in digitalization of handwritten and mechanical printed texts, and musical scores.

Character recognition is a very difficult problem, due to the great variability in writing styles, in other words, wide interclass variability: the same character can be written in different sizes and orientation angles.

As shown if Fig. 1, an OCR system is comprised of certain steps: image acquisition – a color, gray level or binary image is acquired; pre-processing – image processing techniques are applied to improve image quality; layout analysis – the text structure is understood to facilitate text interpretation; word segmentation in characters; classification – pattern recognition is employed for character recognition and post-processing – gather the recognized characters to obtain the original words (opposite for word segmentation).

In this work we focused attention only on the classification step of digit recognition. Table 1 provides details about some digit recognition studies published in the literature. The columns of this table include: database, input data, classifier used and results.

Concerning input characteristics, the studies can be divided into two main groups: the first group consists of studies using digit extracted characteristics as input data [5,6,9,10,11] and the second one consists of studies using the whole image as input data [2,3,4,7,8].

Concerning the databases used, the studies shown in Table 1 can be divided into four groups: MNIST database, proprietary databases, CENPARMI database and NIST-SD19 database.

For performance comparison between different studies it is necessary that a common database be used for all them. In this work the MNIST handwritten digits database is adopted as the common database [12]. This database is suited for training and testing digit recognition algorithms and consists of 60,000 training patterns and 10,000 testing patterns. The patterns were obtained from 250 different authors. One digit is centralized in a gray level figure with 20x20 pixel size. This database presents two advantages: the digits need not be pre-processed and it is extensively used in the literature, enabling a performance comparison between different algorithms.

Consulting the web site of MNIST database, it can be verified that a total of 68 classifiers have been used for digit recognition [12]. The most used are: SVM, MLP and neural networks using convolutional algorithms.

In general, neural classifiers perform better than other classifiers. Convolutional algorithms have the best classifier performance. The best results for the accuracy in the classification step using convolutional algorithms, 99.73%, were obtained by Ciseran et al. [4]. In this study, the authors expanded the training and testing database, including elastic distortions. Deng [13] concluded that the use of distortion to expand the database is necessary to obtain high accuracy in digit classification. Studies that do not use distortion obtained low accuracy rates, varying between 99.47% and 99.65%.

Concerning the MNIST database and Table 1, it should also be noted that classifiers that use a whole image as input characteristics perform better than those that use digit characteristics as input.

Although impressive results for digit recognition using the MNIST database have been reported in the literature, this work focuses on improving state-of-the-art digit recognition, investigating the use of SVM.

In the literature, using SVM, the best results for digit recognition in the MNIST database, an accuracy of 99.44%, was obtained by Decoste and Scholkpf [2]. The authors employed a multiclass SVM classifier associated with the support virtual-vectors technique.

In this work, we intend to use SVM binary classifiers associated with a multiclass binary architecture, the unbalanced decision tree. According to input variables, two classifiers were trained and tested. One of them used digit characteristics and the other used the whole image as input variables.

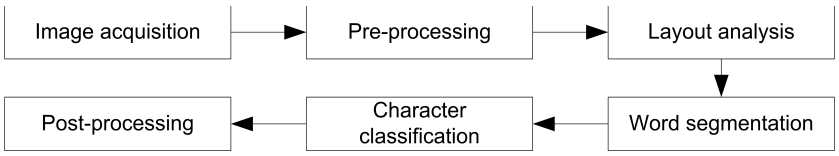

Fig. 1 Block diagram of an optical character recognition system

Table 1. A brief review of digit recognition

Reference	Database	Input data	Classifier	Results (accuracy)
[2]	MNIST Database	Whole image	SVM	99.44%
[3]	MNIST Database	Whole image	Combination of Convolutional Neural Networks	99.73%
[4]	MNIST Database	Whole image	Combination of Convolutional Neural Networks	99.77%
[5]	Proprietary Database	Fourier Descriptors + Border Transition Technique	MLP	96%
[6]	Proprietary Database	Fourier Descriptors	MLP + Models Previously Defined	90%
[7]	MNIST Database	Whole image	Perceptron	99.37%
[8]	Proprietary Database	Whole image	MLP	90%
[9]	Not cited	Hough Transform	MLP + Dempster-Shafer Theory	Not cited
[10]	NIST-SD19 Database	Kirsch Masks and Elliptic Fourier Descriptors	Combination of SVM Classifiers	98.55%
[11]	CENPARMI Database	Directional Distances	Modular Neural Networks	97.30%

2 Methods

2.1 Multiclass binary architectures

In both items 2.2 and 2.3, which address, respectively, the use SVM classifiers for digit recognition using digit characteristics and the whole image as input data, unbalanced decision trees, a type of multiclass binary architecture, is employed for digit recognition. So, in this item, we briefly review the different architectures of binary classifiers and, particularly, unbalanced decision trees.

According to Hassan and Demper [14], there are four different multiclass architectures using binary classifiers: one-against-rest, one-against-one, acyclic direct graph - ADG and unbalanced

decision tree - UDT. Fig. 2 shows these architectures for a special case of four classes. In each one of these architectures, the output is the selection of only one class.

To distinguish between m classes, the architecture one-against-rest requires the training of m classifiers. Each classifier C_i is trained for recognizing class *i*. C_i returns a 1 if a given sample belongs to class *i* and 0 if a given samples does not belong to class *i*. It is only necessary to train m classifiers. When the training set is highly unbalanced, the performance of this architecture can be seriously affected. For a sample classification, m classifiers are used.

The one-against-one architecture uses the major voting rule. One sample is defined as belonging to class *i* if there are more votes for this class than for the others. A total of $m(m-1)/2$ binary classifiers are constructed, one for each different class pair. These classifiers are evaluated in parallel. Each classifier C_{ij} is trained using only samples of classes *i* and *j*. If a sample *x* is recognized by classifier C_{ij} as belonging to class *i*, a vote is assigned to class *i*. Otherwise, if it is recognized as belonging to class *j*, a vote is assigned to class *j*. After the sample is classified by all classifiers, the class that received more votes is considered the one to which the sample belongs. For a sample classification, $m(m-1)/2$ classifiers are used.

A set of binary classifiers can also be structured as an ADG. For this architecture, $m(m-1)/2$ binary classifiers are also necessary. In the architecture shown in Fig. 2, it can be observed that if the output of a classifier C_{ij} is class *i,* in the following node the class *j* is no longer considered a possible output class. This is why only $m-1$ classifiers are used for a pattern classification. Differing from the one-against-one architecture, only m-1 classifiers are evaluated to obtain a sample classification.

UDT was proposed by Ramanan et al. [15]. In each node, a decision is made regarding the type one-against-rest. Comparing with the architecture one-against-rest previously presented, this architecture uses only *m-1* classifiers. A sample classification begins in the node located on the top of the tree, using the classifier C_i. If the sample does not belong to class *i,* the decision process follows with the next right classifier of the tree. The classification process finishes when the sample is recognized as belonging to class *n*, by classifier C_n. As noted in Fig. 2, the lowest node of the tree decides only between two classes. According to Hassan & Damper [14], UDT follows a knockout strategy that, in the worst case, for a sample classification, requires $m-1$ classifiers. For a sample classification, on average, $(m-1)/2$ classifiers are used. Table 2 summarizes the main information of the four multiclass binary architectures. As shown, the UDT classifiers require a smaller number of classifiers both for training and classification. This is why in this paper we used SVM binary classifiers with a UDT multiclass architecture for digit recognition.

.

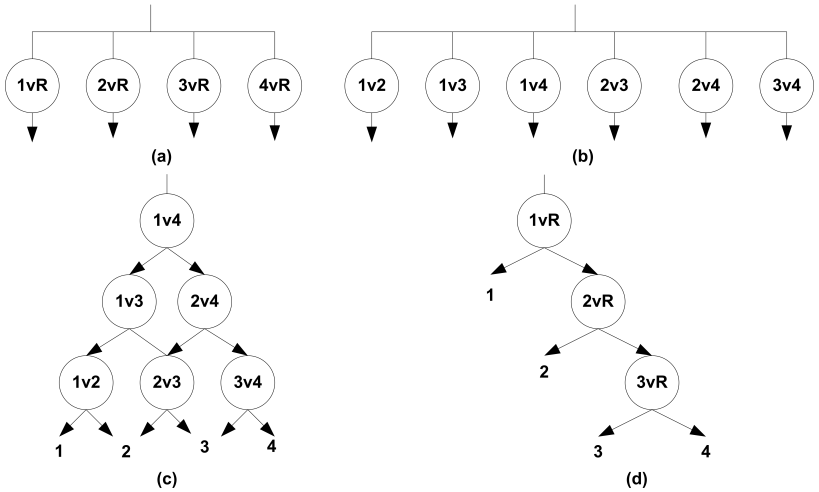

Fig. 2 Multiclass binary architectures: (a) one against rest; (b) one against one, (c) acyclic direct graph, (d) unbalanced decision tree.

Table 2. Summary binary classifier architectures

Architecture	Number of classifiers	Classifiers used for a sample classification
one-versus-rest	*m*	*m*
one-versus-one	*(m*(m-1))/2*	*(m*(m-1))/2*
acyclic direct graph	*(m*(m-1))/2*	*m-1*
unbalanced decision tree	*m-1*	*(m-1)/2 **

* Average value

2.2 Digit recognition using multiclass binary architecture with SVM binary classifiers and digit characteristics as input data

The block diagram of Fig. 3 shows a block diagram of the pattern recognition system used for digit recognition, using digit characteristics as inputs.

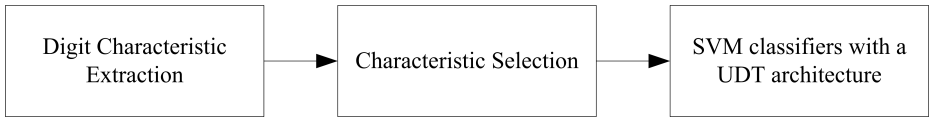

Fig. 3 Block diagram of a digit recognition system using multiclass binary architecture with SVM binary classifiers and digit characteristics as input data.

Digit Characteristic Extraction. A set of 28 characteristics was used: twenty parameters corresponding to Fourier descriptors and eight parameters associated with border transition technique.

The twenty Fourier descriptors selected were the low frequency ones. The higher frequencies coefficients were discarded because they have insignificant values.

The border transition technique divides the digit image into four quadrants. For each quadrant, it calculates the transitions of pixel values from 0 to 1. In other words, a summation of the first order gradient in vertical and horizontal directions is done, totaling 8 parameters. In this work this complementary technique was used associated with Fourier descriptors, because the latter is invariant with rotation and displacement, impairing the distinction between ´6´and ´9´.

Characteristic Selection. Not all the 20 Fourier descriptors were used for classification. To select the best Fourier descriptors the scalar characteristic selection was used [16]. This is an "ad-hoc" technique that incorporates correlation information combined with criteria tailored for scalar characteristics. The procedure is divided into three parts. The first part is devoted to selecting only the first characteristic. The second part is devoted to selecting the second characteristic and the third part is used to select the other characteristics. In the first part, a class separability measure is selected and its value is computed for all the available characteristics. These values are ranked in descending order and the characteristic with higher value is chosen. In this paper, for this first part, a Fisher´s Discriminant Ratio (FDR) was used.

According to Theodoridis and Koutroumbas [16], FDR is sometimes used to quantify the separability capabilities of individual characteristics in a two-class problem, as is the case in this paper (pixels belong to bacillus or to background). FDR is defined as:

$$FDR = \frac{(\mu_1 - \mu_2)}{\sigma_1^2 + \sigma_2^2} \qquad (1)$$

Where μ_1 and σ_{12} represent the mean value and standard deviation, respectively, of a characteristic in class $\mu_1\omega_1;\omega_1$ μ_2 and σ_{22} $\mu_2\sigma_2^2$ represent the mean value and standard deviation, respectively, of the same characteristic in class $\omega_2\omega_1\omega_2$.

In the second and third parts, two other separability class measures are used: the divergence separability measure and the cross-correlation coefficient. The divergence measure between two classes ω_i and ω_j, for a given characteristic with mean value and standard deviation μ_i and σ_{i2} and μ_j and σ_{j2}, respectively, is defined as:

$$d_{ij} = \frac{1}{2}\left(\frac{\sigma_j^2}{\sigma_i^2} + \frac{\sigma_i^2}{\sigma_j^2} - 2\right) + \frac{1}{2}(\mu_i - \mu_j)^2\left(\frac{1}{\sigma_i^2} + \frac{1}{\sigma_j^2}\right) \qquad (2)$$

To define the cross-correlation coefficient between two characteristics, let x_{nk}, n =1,2,....N and k=1,2,....m, be the kth characteristic of the nth pattern. The cross-correlation coefficient between any two characteristics is defined as [11]:

$$\rho_{ij} = \frac{\sum_{n=1}^{N} x_{ni} x_{nj}}{\sqrt{\sum_{n=1}^{N} x_{ni}^2 \sum_{n=1}^{N} x_{nj}^2}} \quad (3)$$

The second part selects x_{i_2} which

$$i_2 = \arg\max_j \{\alpha_1 \min_{i,j} d_{ij} - \alpha_2 | \rho_{i_1 j} |\}, for \text{ all } j \neq i \quad (4)$$

where α_1 and α_2 are weighting factors that determine the relative importance given to the two terms inside the brackets.

The third part selects x_{i_k}, k=3,...l, which

$$i_k = \arg\max_j \left\{\alpha_1 \min_{i,j} d_{ij} - \frac{\alpha_2}{k-1} \sum_{r=1}^{k-1} | \rho_{i_r j} |\right\} \quad (5)$$

With this technique, sets with the best 18, 17, 16, 15, 14, 13, 12 ,11, 10 and 9 Fourier descriptors were selected.

SVM Classifiers. Support vector machines (SVM) can be defined as binary learning machines used to separate data belonging to two classes using a hyperplane that maximizes the separation margin [17].

According to Theodoridis and Koutroumbas [16], for separable classes, the parameters of the hyperplane that maximize the margin are calculated through the determination of weight vector w and polarization w0, such that expression (6) is minimized and the Karush-Kuhn-Tucker (KKT) conditions are satisfied.

$$J(\mathbf{w}) \equiv \frac{1}{2} || \mathbf{w} ||^2 \quad (6)$$

For nonseparable classes, the same parameters can be calculated minimizing expression (7), where new variables ξi, known as slack variables, are introduced. The goal now is to make the margin as large as possible but at the same time to keep the number of points with ξ > 0 as small as possible [16].

$$J(\mathbf{w}, w_0, \boldsymbol{\xi}) = \frac{1}{2} || \mathbf{w} ||^2 + C \sum_{i=1}^{N} \xi_i \quad (7)$$

Parameter C in expression (7) is a constant positive that controls the tradeoff between the slack variable penalty and the margin. The value of the C parameter used in this work was 0.5.

SVMs use kernels to map the characteristic vector into a high dimensional space to exploit the nonlinear power of this tool. In this work, radial base function kernels were used, as shown in expression (8).

$$\exp(-\gamma || \mathbf{x} - \mathbf{z} ||^2)^d, \gamma > 0 \quad (8)$$

3 Results

For SVM binary classifiers with an UDT architecture and digit characteristics as input data, the best results were obtained using the set of the best nine Fourier descriptors selected with the scalar selection technique, with the eight parameters obtained with a border transition technique, totaling 16 input variables for the SVM classifier. The nine best Fourier descriptors were the ones corresponding to the nine lower frequencies. For pattern classification, nine SVM binary classifiers were used with a UDT architecture, as shown in Fig. 4. Fig. 5 shows the confusion matrix obtained. Table 3 shows the accuracy obtained for the ten digit classification.

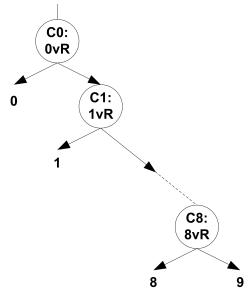

Fig. 4 UDT architecture used with SVM classifiers $C_0, C_1 \ldots C_8$.

```
>>    ConfusionMatrix

   ConfusionMatrix =

   739     0     7     1     8     0    21     8    14     2
     2   747    15     0     6     2     9     3     8     8
     8     0   685    18    17     4    44     8     7     9
     5     0    75   665     2     7    11    10    17     8
    12     5    19     0   692     3    10     6    10    43
     3     2    27    11     8   669    34     7    22    17
    12    12    18     5    11    20   694     4    16     8
     4     7    29    10    12     2     8   698    13    17
    31    10    28    20    17     8    36     8   585    57
     4    11    18    10    44     7     4    22    42   638
```

Fig. 5 Confusion matrix for multiclass binary architecture with SVM binary classifiers and digit characteristics as input data.

For digit recognition using multiclass binary architecture with SVM binary classifiers and the whole image as input data, the number of inputs of each SVM classifier used in the UDT architecture (shown if Fig. 4) was 400, which corresponds to the pixels of an image with 20x20 pixels. Fig. 6 shows the confusion matrix obtained. As shown in Fig. 6 no classification error occurred. So the obtained accuracy was 100%.

Table 2. Accuracy obtained for digit recognition using multiclass binary architecture with SVM binary classifiers and digit characteristics as input data.

Digit	Accuracy	
	Training	Test
0	97.43%	93.50%
1	98.62%	96.56%
2	90.31%	85.31%
3	90.25%	91.44%
4	93.97%	90.50%
5	96.59%	93.87%
6	93.63%	91.63%
7	96.49%	92.94%
8	89.75%	89.56%
9	94.42%	90,28%
Mean Value	93.85%	91.56%

```
>>> modular_classifier.getConfusionMatrix(mnist_experiments.MnistData.TestingData)
array([[ 980,    0,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0, 1135,    0,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0, 1032,    0,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 1010,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,  982,    0,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,  892,    0,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,  958,    0,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0, 1028,    0,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,  974,    0],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0, 1009]])
```

Fig. 6 Confusion matrix for multiclass binary architecture with SVM binary classifiers and the whole image as input data (400 pixel values as input data).

4 Conclusion

Developed recently, UDT architecture provides a simple structure for a multiclass classifier using binary classifiers. In this work, the association of SVM binary classifiers with a UDT architecture, using the whole image as input, makes it possible to obtain an accuracy of 100% with handwritten digit recognition, using the MNIST database. This is the best recognition rate found in the literature for the MNIST database. As stated earlier, before this work, the best results for the accuracy in MNIST database was obtained by Ciseran et al. [4], 99.73%, using neural networks with convolutional algorithms.

The large number of support vectors used is a drawback of this approach. These vectors must be available at classification time, requiring nearly 1GB of memory.

The UDT architecture can be explored using any type of binary classifier, such as MLP and neural networks using convolutional algorithms, etc.

Acknowledgments: **We thank AcademicEnglishSolutions.com for revising the English.**

References

1. Cheriet, M., Kharma, N., Liu, C.L. and Suen, C.: Character Recognition Systems, Wiley, New Jersey (2007).
2. Decoste, D. and Schölkopf, B.: Training Invariant Support Vector Machines. The Netherlands: Kluwer Academic Publishers (2002).

3. Ciresan, D. C., Meier, U., Gambardella, L. M. and Schmidhuber, J.: Convolutional Neural Network Committees for Handwritten Character Classification. International Conference on Document Analysis and Recognition (ICDAR), 1135 -1139 (2011).
4. Ciresan, D., Meier, U. and Schmidhuber, J.: Multi-column Deep Neural Networks for Image Classification. Dalle Molle Institute for Artificial Intelligence. Manno, Switzerland: IDSIA / USI-SUPSI (2012).
5. Chung, Y. Y. and Wong, M. T.: Handwritten character recognition by Fourier descriptors and neural network. IEEE Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications, 1, 391 – 394 (2007).
6. Poon, J. C. and Man, G. M.: An enhanced approach to character recognition by Fourier descriptor. ICCS/ISITA '92 ', 2, 558 – 562 (1992).
7. Kussul, E. and Baidyk, T. Improved method of handwritten digit recognition tested on MNIST database. 15th International Conference on Vision Interface, 22, 971-981 (2004).
8. Masmoudi, M., Samet, M., Taktak, F. and Alimi, A. M. A hardware implementation of neural network for the recognition of printed numerals. The Eleventh International Conference on Microelectronics, 113 – 116 (1999).
9. Mandalia, A.D., Pandya, A.S. and Sudhakar, R.: A hybrid approach to recognize handwritten alphanumeric characters. International Conference on System, Man and Cybernetics, 1, 723-726 (1992).
10. Travieso, C. M., Alonso, J., and Ferrer, M. A.: Combining different off-line handwritten character recognizers. 15th International Conference on Intelligent Engineering Systems. Propad, 315-318 (2011).
11. Oh, I.-S. and Suen, C. Y.: A class-modular feedforward neural network for handwriting recognition. Pattern Recognition, 35(1), 229 – 244 (2002).
12. LeCun, Y., Cortes, C. and Burges, C. J.: The MNIST database of Handwritten Digits. Available at: http://yann.lecun.com/exdb/mnist/. Accessed in January 02, 2014.
13. Deng, L.: The MNIST Database of Handwritten Digit Images for Machine Learning Research. IEEE Signal Processing Magazine, 141-142 (2012).
14. Hassan, A. and Damper, R. I.: Classification of emotional speech using 3DEC hierarchical classifier. Speech Communication, 54, 903-916 (2012).
15. Ramanan, A., Suppharangsan, S. and Niranjan, M.: Unbalanced Decision Trees for Multi-class Classification. International Conference on Industrial and Information Systems. Sri Lanka, 291-294 (2007).
16. Theodoridis , S. and Koutroumbas, K.: Pattern Recognition, Elsevier Academic Press, San Diego (2006).