Faculdade de Engenharia da Universidade do Porto

FEUP

Sistema de Processamento Automático de Cheques Portugueses

Luis Miguel Marques Soeiro Batista

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major em Telecomunicações

Orientador: Prof. Dr. Jaime S. Cardoso
Responsável local: Eng. Pedro Miguel Carvalho

Junho de 2008

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"Sistemas de Processamento Automático de Cheques Portugueses"

foi aprovada em provas realizadas 16/Julho/2008

o júri

Presidente Professor Doutor Aurélio Joaquim de Castro Campilho
Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Manuel João Oliveira Ferreira
Professor Auxiliar da Escola de Engenharia da Universidade do Minho

Professor Doutor Jaime dos Santos Cardoso
Professor Auxiliar Convidado da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - Luís Miguel Marques Soeiro Batista

Faculdade de Engenharia da Universidade do Porto

Resumo

Os avanços tecnológicos dos últimos anos contribuíram para alterações nas formas de pagamento, com o uso generalizado do cartão multibanco, a intensificação dos pagamentos via Internet e por débito directo nas contas de depósito à ordem. Apesar da massificação dos métodos electrónicos, o volume de cheques bancários é ainda significativo, com milhares de cheques processados diariamente. A verificação e validação dos mesmos envolvem actualmente o reconhecimento manual dos diversos campos, do montante à assinatura. A morosidade do processamento manual dos cheques convida à criação de um sistema automático capaz de validar automática, eficaz e rapidamente os cheques.

Este trabalho, desenvolvido entre Fevereiro e Junho de 2008, consistiu na evolução de um sistema que tem vindo a ser investigado no INESC Porto. O sistema era apenas capaz de reconhecer o valor de cortesia de um cheque. Inicialmente melhorou-se o pré-processamento, tendo sido posteriormente elaborado o processo de extracção da data. Finalmente investigou-se o reconhecimento do valor legal.

Para o desenvolvimento deste trabalho foi necessário estudar vários métodos de pré-processamento de forma a isolar não só os dígitos como as palavras. Foram também objecto de estudo algoritmos de aprendizagem automática para capazes de reconhecer automaticamente os dígitos e palavras. Foi ampliada a base existente de cheques portugueses bem como a de dígitos e foi criada uma base de palavras de forma a conseguir-se realizar o objectivo proposto.

Para o reconhecimento do campo da data são utilizados algoritmos OCRs (Optical Character Recognition) e para o valor legal são usados algoritmos ICRs (Intelligent Character Recognition). Para o campo de valor da data foram utilizadas Redes Neuronais e Máquinas Vector de Suporte e para o campo do valor legal foram estudadas para além destas os HMMs (Modelos Escondidos de Markov), Modelos Elásticos e kNN.

Página em branco

Abstract

The technological breakthroughs that have been happening along the last few years have altered the ways of payment with the use of the credit card and the internet. In spite of the massification of these alternative means of payment, thousands of cheques are still processed on a daily basis. The verification and validation of these cheques are made manually and because this process takes too long it is essential the development of an automatic system that is capable of an automatic, effective and quick cheque validation.

The work, developed between February and June of 2008, consisted in the continuation of a system that was being developed in INESC Porto. The system was able to recognize the courtesy amount. Initially the noise removal was improved and later the field concerning the data amount was extracted and recognized. Finally the legal field recognition was analyzed.

In order to develop such a work it was necessary to study several methods of pre-processing so that isolation of the digits and words would be possible. Algorithms that could recognize digits and words were also studied. The existing database of cheques and digits was also expanded, being necessary the creation of a word database in order to accomplish the goal that was proposed.

The OCRs (Optical Character Recognition) are used for the recognition of the data amount, and the ICRs (Intelligent Character Recognition) for the recognition of the legal amount. As for the data amount we used Neural networks and support vector machines. As for the legal amount, besides the algorithms used for the data amount we also used HMMs, Elastic Matching and kNN.

Página em branco

Agradecimentos

Ao Professor Doutor Jaime dos Santos Cardoso, orientador da FEUP, ao Engº. Pedro Miguel Carvalho, responsável local, pela paciência demonstrada e orientação disponibilizada ao longo de toda a dissertação.

Ao Mestre Filipe Emanuel Amaro Coelho, responsável pela minha rápida adaptação ao sistema, e pelas sugestões dadas durante toda a dissertação.

Gostaria de agradecer também ao INESC porto, nomeadamente aos elementos pertencentes à Unidade de Telecomunicações e Multimédia pelo apoio dado ao desenvolvimento de todo o sistema.

À minha família por toda a compreensão e paciência demonstrada durante a elaboração da dissertação.

Página em branco

Índice

Lista de Figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas

INESC	Instituto de Engenharia de Sistemas e Computadores
OCR	Optical Character Recognition
ICR	Intelligent Character Recognition
MLP	MultiLayerPerceptron
RBF	Radial Basis Function
KNN	K-nearest neighbour
SVM	Support Vector Machines
PDA	Personal Digital Assistant

Capítulo 1

Introdução

O sistema bancário português tem sofrido evoluções dramáticas nos últimos anos, em especial desde a criação da Sociedade Interbancária de Serviços SA (SIBS) em 1983 e em particular com o lançamento da rede multibanco (*Automatic Teller Machine* - ATM). Este cenário provocou uma transformação profunda nos padrões de pagamento na sociedade portuguesa. Em particular promoveu o decréscimo na utilização de instrumentos tradicionais de pagamento e o incremento na utilização do pagamento electrónico por transferência de crédito por ATM ou Internet. Contudo, esta supremacia dos pagamentos electrónicos está longe de se traduzir na morte do cheque bancário. A continuação da utilização da escrita, mesmo na era dos computadores, deve-se não só à fácil utilização da caneta e papel mas também à maior dificuldade de transporte de equipamento electrónico/informático. Além da dificuldade de transporte, os computadores têm um preço muito superior quando comparados com os instrumentos de escrita manual. A facilidade de se escrever em papel é outra vantagem para a sua contínua utilização.

Apesar da mudança gradual no padrão de pagamentos em Portugal com o aumento de pagamentos electrónicos, os cheques representam ainda um instrumento importante de pagamento, sendo mensalmente processados milhões de cheques em Portugal. Como consequência, os bancos têm que despender importantes recursos neste processamento. Grande parte dos recursos utilizados actualmente pela banca são recursos humanos, dada a capacidade humana em distinguir os vários padrões de escrita existentes. A multiplicidade de caligrafias, a falha no desenho de caracteres, a sobreposição de algumas letras e os diferentes espaçamentos entre palavras/letras dificultam a validação de um cheque por meios electrónicos. Contudo, o desenvolvimento de sistemas de processamento automático de cheques bancários permite acelerar o processamento (dando resposta às novas exigências colocadas à banca pelo Acordo de Basileia II), e poupar em recursos humanos. Optimizar este

processo de decisão obriga que a decisão seja consistente, objectiva e rápida, com o mínimo de erros e de perdas.

De maneira a reconhecer correctamente os valores contidos em cheques é importante investigar e desenvolver algoritmos avançados de processamento digital de imagem e *Machine Learning* que se adeqúem ao problema apresentado.

Para poder haver uma análise automática dos vários tipos de escrita é necessário uma conversão para um formato digital sendo o scanner a forma mais utilizada (offline), embora se possa utilizar o método online. Estes dois métodos diferem na base temporal, já que no método offline a análise é feita posteriormente enquanto que no método online é feita em tempo real. Os PDAs são os utilitários mais usados hoje em dia onde é aplicado o método online. Na forma offline é disponibilizada uma imagem para análise. O reconhecimento das palavras é feito a partir da imagem e de um léxico associado. O processo de reconhecimento é realizado de duas maneiras distintas: OCR (Optical Character Recognition) ou ICR (Intelligent Character Recognition). O OCR [1] é implementado para a análise de texto escrito à máquina enquanto que o ICR [1] envolve texto manuscrito.

1.1 - Objectivos

Os objectivos proposto para esta dissertação foram os seguintes:

- Melhorar o trabalho desenvolvido até então, ou seja, melhorar o processamento do cheque (eliminação do ruído) e melhorar o reconhecimento dos dígitos;
- Estudar/Implementar métodos para o processamento/reconhecimento das palavras escritas em cheques portugueses.

Os cheques emitidos em Portugal encontram-se "normalizados" desde a entrada em circulação do euro, ou seja, qualquer instituição bancária em Portugal ao emitir um cheque tem de cumprir um conjunto de regras, sendo o *layout* do cheque uma delas. Este conjunto de regras facilita a detecção dos vários campos de interesse num cheque (a localização do campo de cortesia, do valor legal, da data, entre outros), bem como as suas dimensões. Entre os vários campos existentes num cheque, dois destinam-se à especificação do montante: o valor de cortesia (algarismos) e o valor legal (por extenso).
Apesar de existirem vários tipos de sistemas de processamento automático de cheques estes não estão adaptados à língua portuguesa. Torna-se assim importante o desenvolvimento de um sistema capaz de reconhecer o valor legal adaptado ao *layout* dos cheques em Portugal e ao nosso idioma.

1.2 - Organização e temas abordados

Os capítulos seguintes descrevem a forma como está estruturada a dissertação:

Capítulo 2 - Estado de arte;

Onde se apresenta uma revisão científica do que foi elaborado até a data pela comunidade cientifica, incidindo sobre os artigos que mais impacto tiveram na dissertação. Apresenta-se também uma explicação resumida dos vários algoritmos utilizados na elaboração da parte experimental (reconhecimento de dígitos e palavras). É também apresentada uma breve explicação da arquitectura do sistema;

Capítulo 3 – Pré-processamento e extracção campo da data

Neste capítulo será apresentado o processamento sofrido pelo cheque bem como o reconhecimento do valor da data, onde se discute a melhor maneira de se retirarem os dígitos, eliminando o ruído, e onde são analisados algoritmos de *Machine Learning*, tais como, as Redes Neuronais e Máquinas Vector de Suporte;

Capitulo 4 - Extracção e tratamento do campo de valor legal;

Reconhecimento do valor legal onde são analisados algoritmos de *Machine Learning* como as Redes Neuronais, Máquinas de Vectores de Suporte, kNNs, Modelos Escondidos de Markov e "Elastic Matching" para verificar quais destes apresentam um melhor resultado no reconhecimento das palavras;

Capítulo 5 - Conclusões;

Onde é feita uma análise crítica dos resultados obtidos e das contribuições resultantes da investigação efectuada ao longo da dissertação. Discute-se ainda possível linhas de investigação de forma a continuar o trabalho.

1.3 - Resultados e contribuições relevantes

No estudo para identificar o melhor algoritmo de *Machine Learning* para o reconhecimento de algarismos, chega-se à conclusão de que as Máquinas Vectores de Suporte com núcleo Funções de Base Radial apresentam melhores resultados para o reconhecimento do campo da data. Ao contrário das abordagens mais tradicionais, que preferem um método no qual a análise da imagem é feita de forma directa, demonstra-se nesta dissertação que a característica de contorno obteve melhores resultados.

Para o reconhecimento de palavras conclui-se que as Máquinas Vectores de Suporte com núcleo Funções de Base Radial apresentam o melhor algoritmo para o reconhecimento. No estudo das características que melhor ajudam a identificar as palavras, concluiu-se que não

existe apenas uma característica a desempenhar este papel, mas antes um conjunto delas. As características, que serão explicadas em detalhe ao longo da dissertação, ascendentes/descendentes, o tamanho da palavra e os *loops* mostraram ser a combinação que atinge o melhor resultado.

As contribuições apresentadas por esta dissertação são:

1. Melhoria e extensão de uma *Framework* já criada para o reconhecimento da análise automática de cheques portugueses;
2. Criação de uma base de dados de palavras escritas em português, limitada a palavras possíveis de aparecer no valor legal de um cheque;
3. Aumento da base de dados de cheques portugueses bem como de dígitos;
4. Análise comparativa de vários algoritmos de *Machine Learning* para o reconhecimento do campo da data;
5. Análise comparativa de vários algoritmos de *Machine Learning*, tanto OCR, ICR e clustering para o reconhecimento do campo do valor legal.

Foi ainda elaborado o artigo científico "Automatic system for the recognition of amounts in handwritten checks", aceite para publicação na International Conference on Signal Processing and Multimedia Applications – SIGMAP 2008.

Capítulo 2

Estado da arte

Neste capítulo será apresentado o estado da arte para permitir aos leitores aperceberem-se do ponto da situação em que se encontra neste momento o reconhecimento de palavras na literatura científica, bem como se aperceberem dos vários entraves existentes ao problema apresentado. Serão também apresentados os vários algoritmos de aprendizagem usados na dissertação de forma a tentar ultrapassar os problemas inerentes ao reconhecimento de palavras e dígitos.

2.1 - Algoritmos de aprendizagem

Nesta secção será feita uma breve explicação dos vários algoritmos utilizados ao longo da dissertação tanto para o reconhecimento de dígitos como para o reconhecimento de palavras.

2.1.1 - kNN (Nearest Neighbors Classifier)

O kNN é um dos classificadores mais simples usados actualmente. O knn baseia-se nos k vizinhos mais próximos para classificar o ponto em questão [2], ou seja, este é colocado na classe mais "popular" entre os seus k vizinhos mais próximos. Como se pode observar na Figura 2.1, o ponto verde será classificado como triângulo vermelho se K=3, pois dos seus 3 vizinhos dois deles são triângulos enquanto que apenas um é quadrado amarelo. No caso de K=6 será classificado como quadrado amarelo.

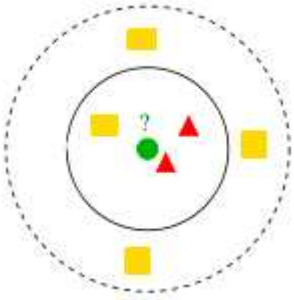


Figura 2.1– Classificação pelo método kNN.

2.1.2 – MultiLayer Perceptron

O MultiLayer Perceptron é uma rede neuronal na qual as saídas são obtidas através do cálculo dos pesos de cada um das entradas aplicando-lhes depois uma função não linear. Esta operação pode ser representada matematicamente pela Equação 2,

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi(w^T x + b) \tag{2}$$

onde w é o vector de pesos das entradas, x é o vector de entradas, b é o "bias" e o φ é a função de activação. A função de activação habitualmente usada é a sigmóide indicada na Equação 3.

$$\text{sigmóide(x)} = \frac{1}{1+e^{x}} \tag{3}$$

Este tipo de função é usada porque muitos dos algoritmos de treino mais frequentes utilizam métodos para os quais esta função obtém os melhores resultados.

Para a estrutura da rede geralmente apenas se altera o número de camadas escondidas e o número de neurónios presentes em cada camada. Isto porque o número de entradas e saídas depende do problema em estudo.

A rede é treinada através do algoritmo *back-propagation*, ou seja, fornece-se à rede as entradas e suas respectivas saídas esperadas, sendo calculado o desvio entre a saída real e a saída esperada. Após o cálculo, o sinal é retro propagado até à entrada, sendo corrigidos os erros individuais de cada neurónio. O próprio peso dos neurónios pode alterar-se de forma a aproximar a saída real da saída esperada. Os parâmetros do algoritmo são dois: o *learning*

rate e o *momentum*. O primeiro consiste na velocidade com que é desejada que a rede aprenda, ou seja, quanto maior for o seu valor maior será a velocidade com que os pesos dos neurónios serão alterados. Uma das desvantagens de se aumentar em demasia o valor do l*earning rate* é a possibilidade da existência do fenómeno de *over-fitting* que consiste no facto das generalizações feitas pelo algoritmo não produzirem resultados satisfatórios. Este fenómeno ocorre pelo facto de a rede não ter o tempo necessário para aprender, ficando demasiado adaptada aos exemplos de treino fornecidos. No entanto, se o valor do *learning rate* for demasiado baixo causará um longo período de aprendizagem. O *momentum* foi criado para auxiliar a rede quando se utilizam valores de *learning rate* elevados, pois o aumento ou diminuição do *momentum* implicará uma alteração mais forte ou leve respectivamente.

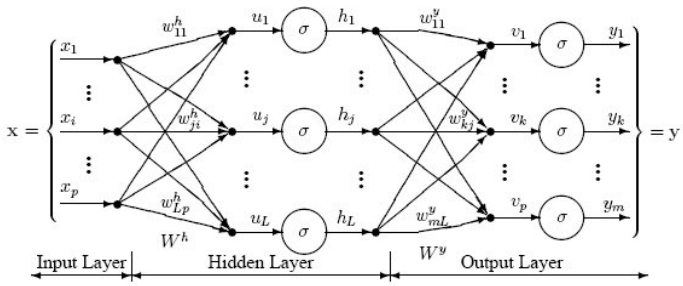


Figura 2.2 – MultiLayer Perceptron com 1 camada escondida com 3 neurónios de [3].

2.1.3 - Rede de Funções de Base Radial

As redes RBF (Radial Basis Function) são constituídas por uma única camada escondida de neurónios. Na entrada de cada neurónio é calculada a distância entre o seu centro e o vector de entrada. Ao valor da distância calculada será aplicada a função gaussiana, aparecendo na saída a soma das várias saídas produzidas pelos neurónios. Na Figura 2.3 e na Figura 2.4 encontram-se exemplos de duas redes neuronais com uma e várias saídas respectivamente.

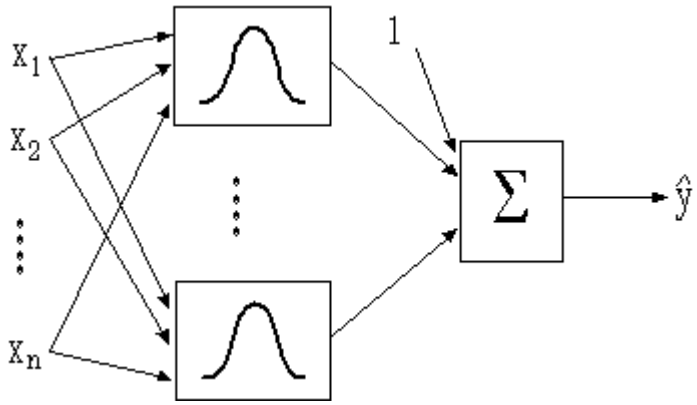


Figura 2.3 – Rede de Funções de Base Radial com uma saída de [4].

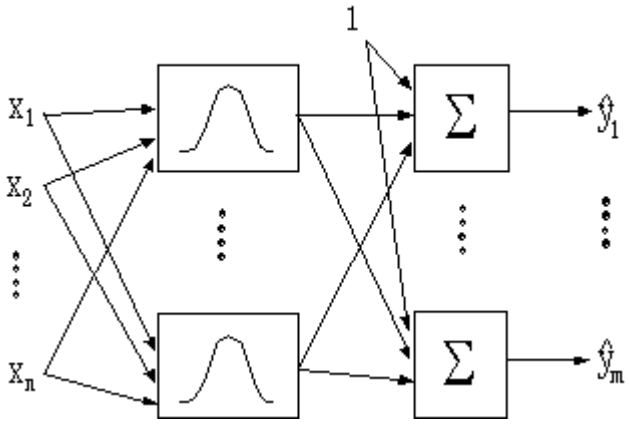


Figura 2.4 – Rede de Funções de Base Radial com várias saída de [4].

2.1.4 - Support Vector Machines (SVM)

Os SVMs são classificadores muito semelhantes às redes neuronais no qual o seu objectivo é encontrar um hiperplano óptimo de forma separar a informação de duas classes. Esta separação é feita de forma a minimizar o erro empírico de classificação enquanto maximiza a margem entre as classes.

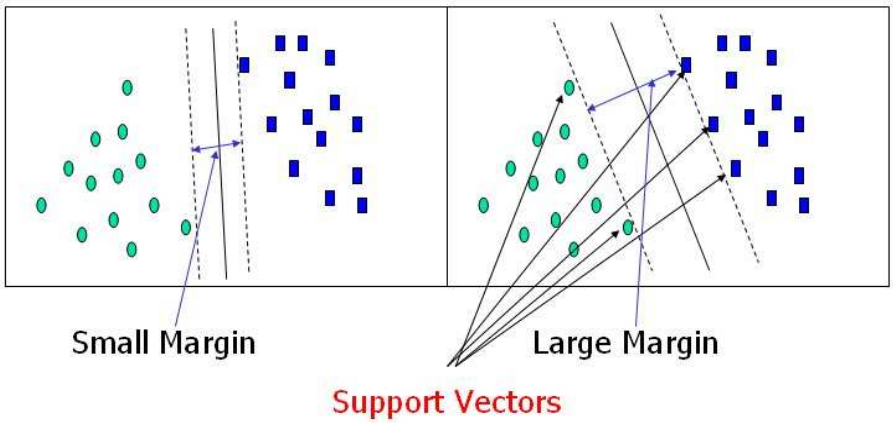


Figura 2.5 – SVM com margem e com vectores de suporte [5].

Os vectores são conjuntos de características que representam uma classe, sendo os vectores de suporte (*support vectors*) os vectores mais próximos do hiperplano que divide as classes.

Os SVMs possuem uma característica fundamental: a possibilidade de utilização da função de mapeamento de um kernel. Esta possibilidade permite ao classificador separar fronteiras complexas como se pode ver na Figura 2.6 e na Figura 2.7.

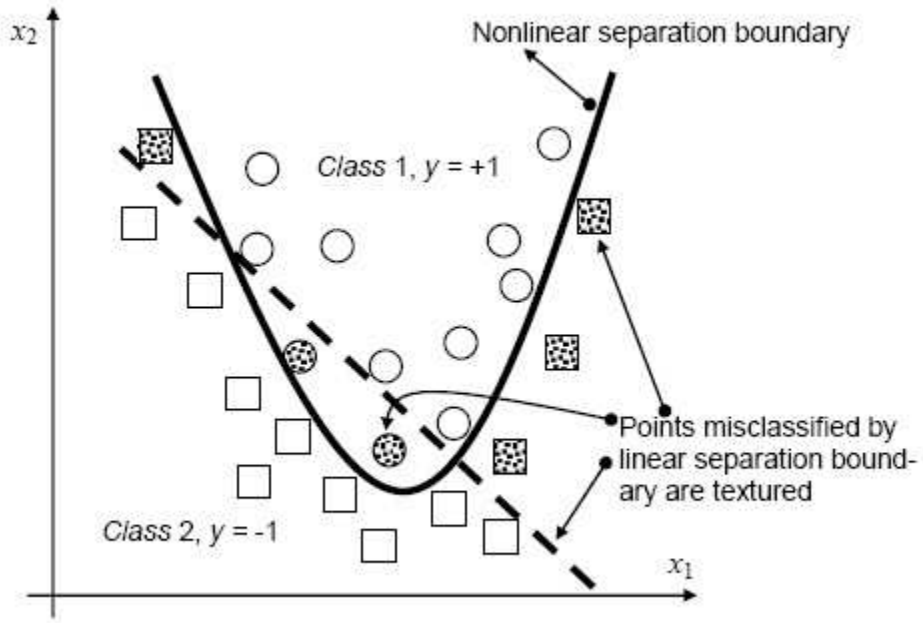


Figura 2.6 – SVM com kernel polinomial de [5].

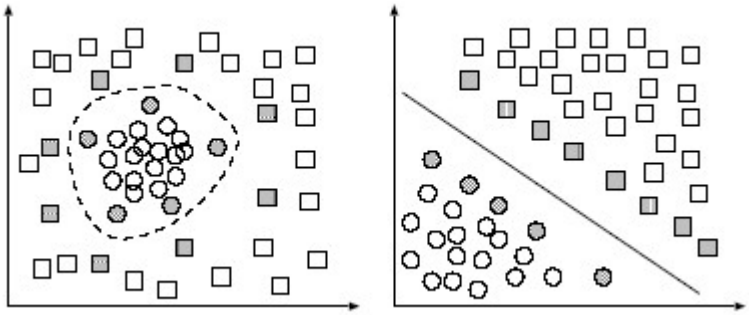


Figura 2.7 – SVM com kernel RBF. A esquerda encontra-se o mapeamento no espaço original e à direita no espaço de vectores de [5].

2.1.5 - Hidden Markov Models (HMMs)

Será apresentada uma breve explicação dos HMMs e como poderão ser aplicados ao reconhecimento de palavras. Consideremos um problema em relação ao tempo que se faz sentir no dia a dia. Iremos assumir que já passamos a fase de extracção de características e já temos um vector de características (*observation symbols sequence*) com tamanhos variáveis como se pode ver na Figura 2.8 i).

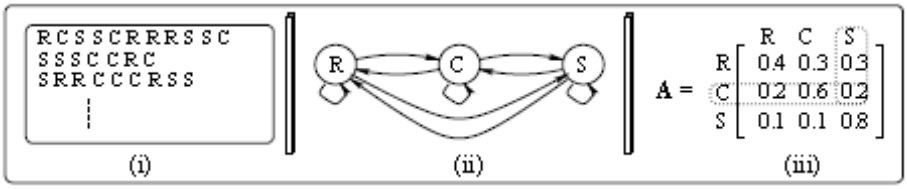


Figura 2.8 – Modelação com o Modelo de Markov de [6].

Como se pode verificar, o vector contem 3 valores diferentes (*codewords*) S, C, R que significam *Sunny*, *Cloudy* e *Rain*. Pode-se então modelar o problema do tempo com uma cadeia de Markov com 3 estados como se pode ver na figura 2.8 ii). Começando em qualquer dos estados, gera-se o símbolo associado a esse estado e depois poderá ser feita a transição para outro estado, isto dependendo da probabilidade de transição de estados. O modelo de Markov é representado simplesmente por uma matriz de transição de estados denominada de A como se pode ver na Figura 2.8 iii). Os coeficientes da matriz são calculados através da contagem do número de vezes que as transacções entre os estados ocorrem no vector de características. Embora este tipo de modelo de Markov seja suficiente para modelar um problema simples como este, não consegue modelar um problema com a complexidade do reconhecimento de palavras [5]. De forma a aumentar o "poder de resolução" dos modelos de Markov é necessária a introdução de mais uma matriz que terá o nome de *hidden layer* (dai deriva o nome de *Hidden Markov Models*). A segunda matriz pode ser vista na Figura 2.9.

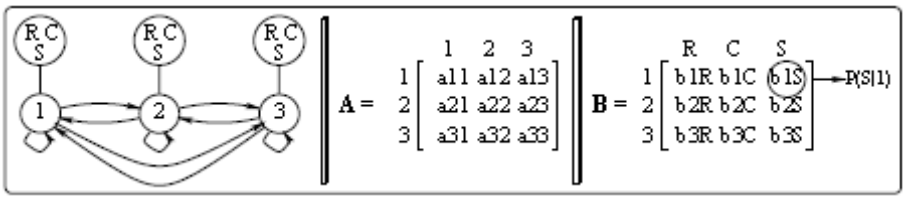


Figura 2.9 – Modelação com o HMM de [6].

Nos HMMs os símbolos já não se encontram associados a nenhum estado específico, embora a matriz de transição de estados (A) ainda represente a probabilidade de transição de estados. A segunda matriz, matriz de probabilidade de observações denominada de matriz B, contem a probabilidade de observações, ou seja, a matriz B diz a probabilidade de, por exemplo b1S, estar observar S (sunny) quando está no estado 1. Os parâmetros da matriz A e B são calculados automaticamente através da informação representada na Figura 2.8 i) usando o algoritmo de Baum-Welche [6].

No caso do reconhecimento de palavras manuscritas, e exemplificando com a palavra "dois", que se pode ver na Figura 4.11, constrói-se um HMM para modelar a palavra. Optando por apenas 2 *observation symbols sequence* A,D (ascendente, descendente) e considerando o número de estados igual ao número de letras, na palavra "dois" existem quatro estados. Poder-se-á assim obter a sequência mais provável através do algoritmo de *Viterbi*. Com a palavra "dois" o algoritmo de viterbi iria apresentar uma forte probabilidade de se encontrar um ascendente no estado 1 e fraca probabilidade de se encontrar um ascendente ou um descendente nos estados 2,3,4.

2.1.6 - Elastic Matching

Tem-se recorrido cada vez mais ao *Elastic Matching* para o problema do reconhecimento de palavras manuscritas. Retirando da base de dados uma imagem que será utilizada como instrumento de comparação, esta será distorcida até ficar o mais parecida possível com a imagem que se está a testar. No final do teste é obtida a percentagem de distorção que o *template* sofreu.

Para ilustrar este caso ir-se-á recorrer a um exemplo que se pode observar na Figura 2.10 a) e b).

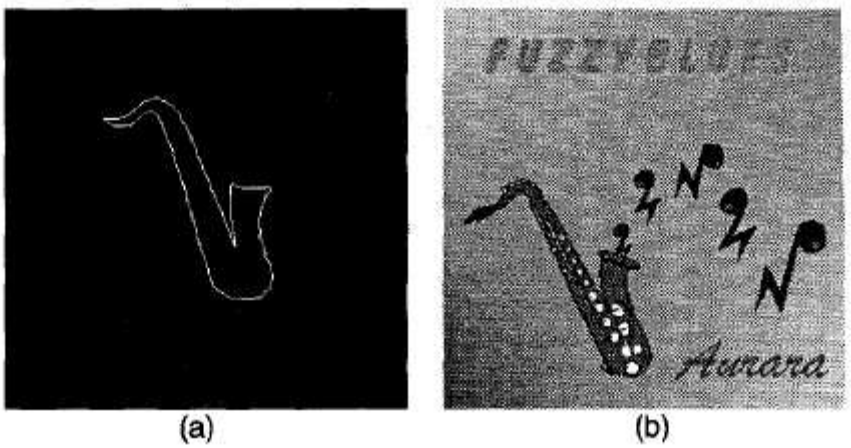


Figura 2.10 – a) *Template* de um saxofone com a característica de contorno b) imagem que contém um saxofone para servir de teste ao da figura (a) de [7].

Como o protótipo apenas descreve uma forma possível para o objecto, este terá de ser deformado de forma a igualar os objectos nas imagens. É assumido que o *template* é

desenhado num quadrado unitário $S=[0\ 1]^2$. Os pontos do quadrado serão mapeados pela Equação 3 de [14], onde D() representa uma função de deslocamento.

$$(x,y) \rightarrow (x,y) + (D^x(x,y),D^y(x,y)) \tag{3}$$

O espaço das funções de deslocamento é estendido pelas bases ortogonais que são dadas pelas Equações 4 e 5 de [7].

$$e^x_{mn}(x,y) = (2\sin(\pi mx)\cos(\pi my),0) \tag{4}$$

$$e^y_{mn}(x,y) = (0,2\sin(\pi mx)\cos(\pi my)) \tag{5}$$

As funções de deslocamento são dadas pela seguinte equação:

$$\mathrm{D(x,y)} = \sum_{\mathrm{m}=1}^{\infty}\sum_{n=1}^{\infty}\left(\frac{\varepsilon^x_{mn}e^x_{mn} + \varepsilon^y_{mn}e^y_{mn}}{\lambda_{mn}}\right) \tag{6}$$

$\varepsilon = \{(\varepsilon^x_{mn},\varepsilon^y_{mn}), m,n = 1,2...\}$ são projecções da função de deslocamento nas bases ortogonais e $\{\lambda_{mn} = \alpha\pi^2(n^2+m^2), m,n = 1,2...\}$ é a constante de normalização.

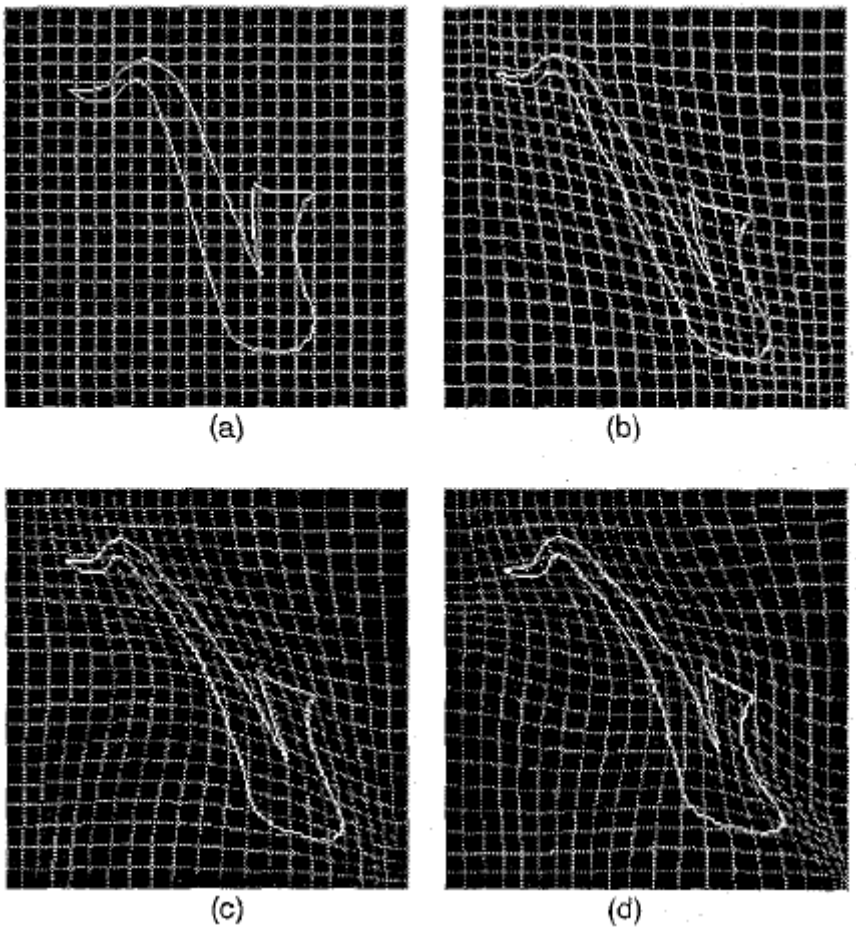


Figura 2.11 – a) *Template* original, b) deformação do *template* da ordem (m,n=1) c) deformação do template de ordem (m,n=2) d) deformação do template de ordem (m,n=3) de [7].

As imagens da Figura 2.11 demonstram o *template* com diferentes graus de deformação. Como se pode ver nas figuras acima, o D(x,y) pode representar deformações complexas, dependendo da escolha de ε_{mn}, de M e N, forçando-se uma imposição de densidade de probabilidade em D(x,y) $\Rightarrow \varepsilon_{mn}$, assumindo que existe independência ao longo de x e y bem como uma distribuição gausiana de média 0 e variância σ^2, resultando na Equação 7 de [14].

$$\mathrm{P}(\varepsilon) = \frac{1}{(2\pi\sigma^2)^{MN}} \exp\{-\frac{1}{2\sigma^2}\left[\sum_{m=1}^{M}\sum_{n=1}^{N}((\varepsilon^x mn)^2 + (\varepsilon^y mn)^2)\right]\} \qquad (7)$$

2.2 – Revisão Científica

Os sistemas automáticos existentes actualmente baseiam-se em técnicas de *Optical Character Recognition*, *Intelligent Character Recognition* e de clustering para o reconhecimento tanto de dígitos como de palavras.

Ávila [8] utilizou HMMs para o reconhecimento de palavras nas quais as características retiradas consistiam em características globais (ascendentes, descendentes, neutro) e locais (alfabeto dos 12 traços). As características globais serviam para a divisão das palavras do valor legal enquanto que as locais foram utilizadas para criar uma sequência de grafemas para ser processada pelo HMM. A proposta consistia na criação de uma lista de hipóteses de palavras obtidas por diferentes métodos de segmentação do valor legal. Aplicando o léxico era gerada a melhor solução para o valor legal.

Cotê [9] desenvolveu um sistema que usava características globais (ascendentes, descendentes, laços, concavidades e convexidades) para o reconhecimento de palavras. A autora definiu letras-chave para cada uma das características. Por exemplo, as letras "t", "l" foram associadas a ascendentes enquanto que a descendentes se associaram as letras "p" ou "q". Os laços foram relacionados a letras como o "o", "e", "a". As restantes características foram consideradas pela autora como primitivas condicionais pois são aquelas que se encontram interligadas, como é o caso da letra "d" que contém um ascendente e um laço. A autora dividiu as caracterizas em primárias, secundárias e vales. As características primárias permitem detectar as palavras-chave, isto é, palavras com ascendentes, descendente e laços. As secundárias só existem na presença das primarias e consistem nos laços associados ao "b", "d". As características de vale são as concavidades e convexidades que são retiradas através do contorno superior e inferior da imagem. Para a classificação utilizou redes neuronais artificiais obtendo resultados que variavam entre 74% para base de teste e 76% para base de treino.

Guillevic & Suen [10] apresentaram um sistema de reconhecimento de palavras que incorporava HMMs e kNNs. Para o classificador kNN é enviado um vector com características gerais. No caso do vector enviado para o HMM é utilizada uma janela que percorre as imagens da esquerda para a direita e guarda a partir de 8 direcções o número de traços que tenham ângulos com a horizontal igual a 0, 45 ,90 e 135. Foram treinados dois HMMs, tendo um deles uma modelação "*left-to-right*" e outro "*right-to-left*" combinando no final o resultado de ambas. Obteve-se desta forma um valor mais assertivo. Para testarem o sistema, os autores utilizaram uma base de dados com 4,500 cheques escritos em inglês e francês por 1,500 indivíduos diferentes obtendo, como se pode ver na Tabela 2.1, para um léxico pequeno resultados com percentagens bastante elevadas.

Tabela 2.1 - Resultados de reconhecimento de escrita francesa de [10].

Classifier	N= 1	2	5	10
kNN (Feature Set 0)	78.3	91.8	98.9	99.9
HMM (Feature Set 1)	84.7	92.9	97.9	99.4
kNN + HMM	86.7	94.6	98.7	99.9

Gorski et al [11] desenvolveram o "A2iA checkReader^{tm}", capaz de localizar os espaços onde são escritos os montantes por extenso e por dígitos. Este método permite detectar o montante em documentos escritos em francês e em inglês. O processamento é feito sobre dois aspectos diferentes: cheques escritos à máquina e cheques escritos à mão. Para ser mais eficiente o método A2IA CheckReader só analisa o valor legal (montante escrito por extenso) sempre que o valor de cortesia não seja suficiente para chegar a uma conclusão. Para o reconhecimento do montante de cortesia o método utiliza quatro OCRs (*optical character recognition*) porque embora inicialmente se possa pensar que o montante, mesmo escrito a máquina, é de fácil reconhecimento para países diferentes isso não é verdade uma vez que cada país tem símbolos especiais, delimitadores diferentes, entre outros. Todos os OCRs utilizados pelos autores são Redes Neuronais no qual se pode ver a eficiência destes algoritmo em dígitos escritos por cidadãos americanos na Tabela 2.2. O OCR_1 processa valores de pixéis de caracteres de imagens, o OCR_2 mede as várias características topológicas da imagem, o OCR_3 usa as características de contorno dos caracteres e o OCR_4 toma em consideração a informação textual (posição do caracter, tamanhos relativos, entre outros). No final têm uma Rede Neuronal Integradora que combina todos os resultados de cada um dos OCRs produzindo uma probabilidade final para cada classe. Para o reconhecimento de cheques de países diferentes é exigido que exista um treino individual para cada um dos países desejados, tornando o treino muito exigente em termos de informação necessária para o método ser eficiente.

Tabela 2.2 - Resultados de caracteres de 5 mil dígitos isolados em inglês (americano) de [11].

	OCR_1	OCR_2	OCR_3	OCR_4	Combination
Module trained on US characters	92.5	89.9	90.5	42.2	97.3
Module trained on UK characters	80.7	80.6	80.4	28.5	87.4

Como se referiu o método só irá analisar o montante legal se o de cortesia não for suficiente. Para o reconhecimento do valor legal são utilizados dois classificadores diferentes, o primeiro baseia-se em características Holísticas e o segundo é um HMM. Os resultados de ambos os classificadores podem-se observar na Tabela 2.3. Os resultados são posteriormente integrados na Rede Neuronal Integradora.

Tabela 2.3 - Resultados da performance em cheques americanos de 36 mil palavras isoladas de 38 classes de [11].

Markov Word Recognizer	Holistic Word Recognizer	Combination of both
81%	70%	89%

O método A2IA CheckReader, sendo um sistema geral, pode ser utilizado por vários bancos em países diferentes pelo que a sua capacidade de reconhecimento varia entre os 65% e os 85%.

Gomes [12] em 2001 decidiu reduzir a multiplicidade de escrita utilizando como características uma sequência ordenada de linhas curvas, rectas e de laços. Após o pré-processamento, a imagem é segmentada resultando em letras ou partes de letras. Posteriormente as letras ou partes de letras são decompostas em segmentos de linhas e são atribuídos valores de pertinência a conjuntos *fuzzy* (na teoria fuzzy os valores podem variar entre 0 e 1 inclusive não sendo forcados a ser ou 0 ou 1). Obtém-se assim uma sequência ordenada de linhas sendo que cada uma delas apresenta um valor de conjunto *fuzzy*. A classificação é feita por FHMMs (fuzzy HMMs). A base de dados integra 2.400 imagens de palavras referentes ao valor legal de cheques brasileiros. Este sistema teve um reconhecimento de 50%.

Uchida e Sakoe [13] desenvolveram um sistema de reconhecimento de palavras escritas à máquina baseado num modelo elástico. Este modelo elástico baseia-se nas *eigen-deformations* que são calculadas a partir das próprias imagens contidas na base de dados. As *eigen-deformations* são calculadas através da utilização do PCA nos vectores de deslocamento de cada classe (palavra), ou seja, são obtidas através dos vectores próprios da matriz de covariância de N vectores de deslocamento (sendo N o número de amostras utilizadas no treino). O vector próprio com o maior valor próprio é a primeira *eigen-deformation*. A Figura 2.12 ilustra os passos para a deformação da classe "A". As *eigen-deformations* são então a aplicação dos vectores próprios nas imagens.

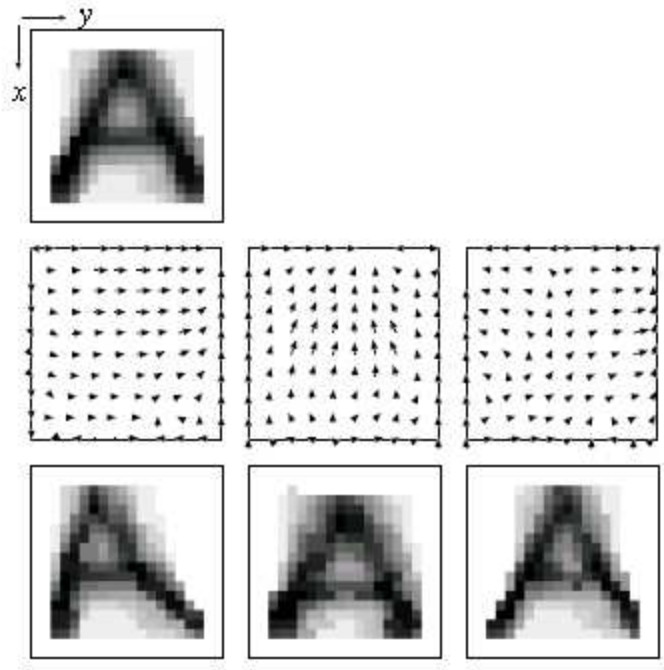


Figura 2.12 - Amostra padrão da classe "A" no topo, aplicação das primeiros três eigen-deformations no meio, resultado das três primeiras eigen-deformations na classe "A" em baixo de [13].

A base de dados utilizada para o teste deste modelo elástico consistia em 26 classes com 1100 amostras isoladas de dígitos para cada classe num total de 28 600 amostras.

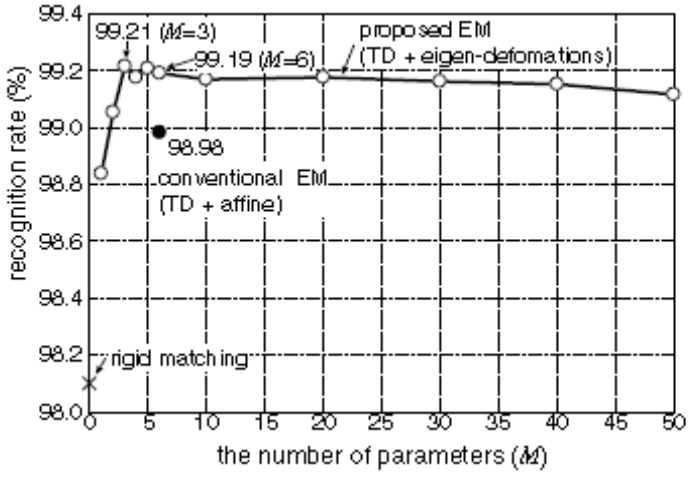


Figura 2.13 – Taxa de reconhecimento obtido pelo modelo elástico proposto de [13].

Como se pode ver na Figura 2.13 o reconhecimento teve o seu melhor resultado para M=3 com 99.21%. Note-se que é a taxa de reconhecimento para caracteres e não para palavras.

Morita et al [14] apresentaram um sistema híbrido HMM-MLP (Hidden Markov Models – MultiLayerPerceptron) para o reconhecimento do campo da data em cheques brasileiros. A rede neuronal (MLP) era utilizada para o reconhecimento de dígitos (situados nos campos do dia e do ano) enquanto que o HMM era utilizado para o reconhecimento de palavras (encontradas no mês). Os autores utilizaram uma mistura de características que envolvia as de contorno, concavidades e globais, obtendo um vector para os dígitos. Como as palavras foram divididas em grafemas foram obtidos dois vectores.

Tabela 2.4 - Performance do sistema apresentado (NV resultados sem verificação, V resultados com verificação) de [14].

	Date	**Month**	**1 - digit Day**	**2 - digit Day**	**2 - digit Year**	**4 - digit Year**
NV	79.8%	87.2%	76.1%	93.9%	96.9%	87.5%
V	82.0%	89.0%	90.4%	94.4%	96.9%	87.5%

Como se pode ver pela Tabela 2.4 o reconhecimento das palavras variou entre os 87% e os 89%, enquanto que nos dígitos variou entre os 76% e os 97%.

2.3 - Revisão tecnológica

O sistema foi desenvolvido tendo em conta duas áreas tecnológicas: o processamento digital de imagem, na qual foi utilizada a plataforma AForge.NET1 v1.51 pelo facto de possuir técnicas de processamento de imagem, e o *Machine Learning*, na qual foi utilizada a plataforma Weka v3.4 devido ao elevado número de classificadores que a plataforma possui possibilitando um estudo aprofundado dos algoritmos. O WEKA (Waikato Environment for Knowledge Analysis) é uma ferramenta de código-aberto bastante utilizada. Esta ferramenta, desenvolvida na Universidade de Waikato de Hamilton, Nova Zelândia, implementa mais de vinte algoritmos diferentes de aprendizagem de dados convencionais.

A arquitectura do sistema desenvolvido encontra-se dividido em três grandes módulos, que podem ser observados na Figura 2.14:

- Aplicação web – que permite a submissão de novos cheques, novas imagens de dígitos e palavras.
- Base de Dados – que contém os cheques submetidos (Anexo A), dígitos e palavras necessárias para o módulo de treino.

- Motor do sistema – encontra-se em constante funcionamento, interrogando a base de dados se tem cheques novos para processar. Contém também 2 submódulos:
    1. Processamento – onde o cheque é tratado em relação ao ruído e ângulo.
    2. Reconhecimento – onde se faz a extracção/reconhecimento dos dígitos e das palavras.

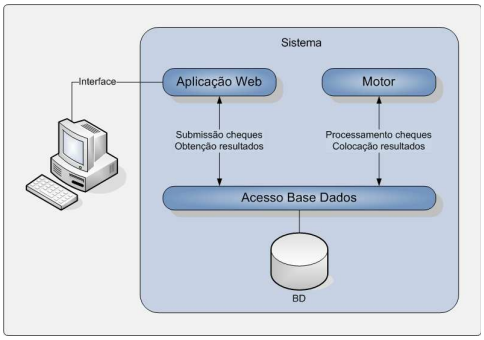


Figura 2.14 – Arquitectura do sistema de [15].

O trabalho desenvolvido neste projecto beneficiou da implementação já existente desta arquitectura [15], tendo incidido em desenvolvimentos a nível do Motor do Sistema, como descrito nos capítulos seguintes.

Capítulo 3

Pré-processamento e extracção do campo da data

Neste capítulo será descrito todo o processamento elaborado para as palavras existentes no campo do valor legal, desde a extracção do campo referente ao valor legal, passando pelo processamento (eliminação de ruídos) até ao reconhecimento das palavras.

3.1 – Pré-processamento do cheque

Antes de se proceder à extracção do campo da data é necessário fazer um pré-processamento do cheque de forma a eliminar todo o ruído para que a extracção dos dígitos e das palavras seja feita da melhor forma possível.

Quando o cheque passa por um scanner é introduzido ruído que precisa de ser eliminado pois afecta negativamente a qualidade da imagem além de dificultar a detecção do ângulo de rotação que muitas vezes vem associado à imagem como se pode ver na Figura 3.1.

Figura 3.1 – imagem que contém associada a ela um ângulo de rotação

Para a eliminação do erro é utilizado um filtro mediana [15] que considera cada pixel e através da observação dos pixéis vizinhos substitui o pixel em questão pela mediana dos valores, como se pode observar na Figura 3.2.

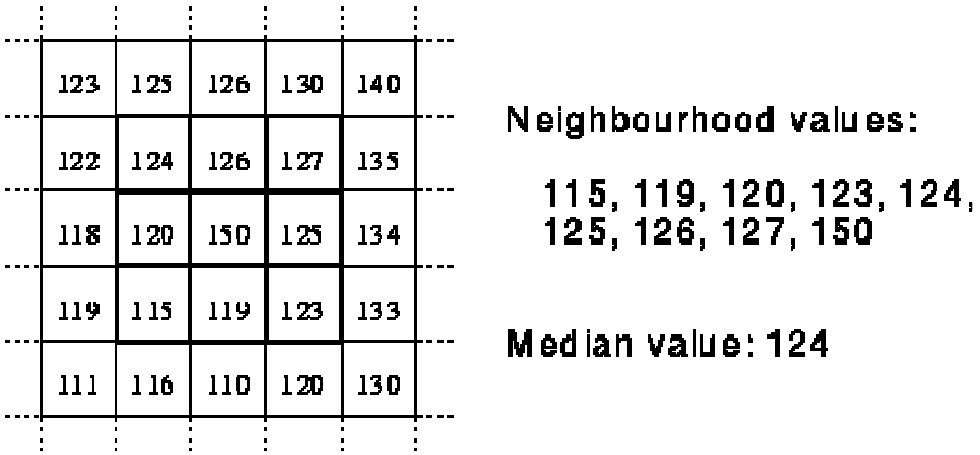


Figura 3.2 – Algoritmo mediana de [15].

Após a eliminação geral do ruído é detectado e corrigido o ângulo que está associado à imagem através da Transformada de Fourier [16] e do PCA (Principal Components Analasys) [17] que foram seleccionados após comparativo que se pode ver em detalhe em Coelho [15]. A Transformada de Fourier será afectada pelo mesmo ângulo de inclinação que o cheque, facilitando assim ao PCA a sua detecção.

Uma vez que na biblioteca Aforge.Net se encontra disponível o algoritmo de *Fast Fourier Tranform*, é apenas necessário aplicá-lo. Após a sua utilização e a aplicação de um *threshold*, foi obtida a imagem da Figura 3.3.

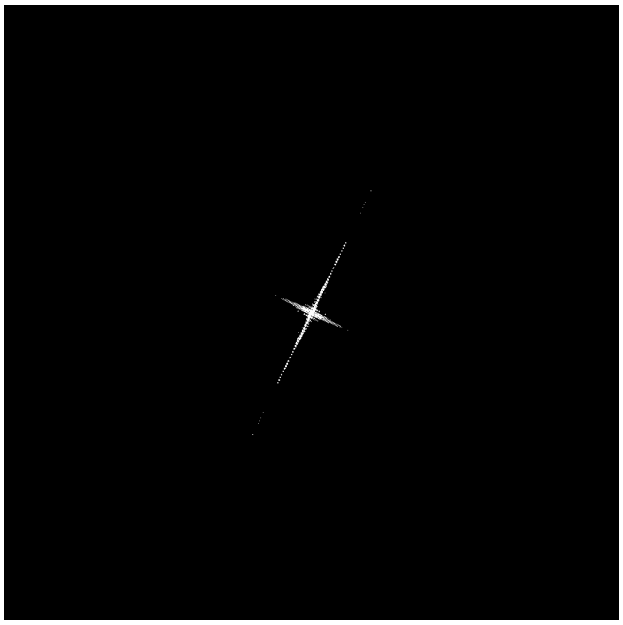


Figura 3.3 – Imagem após a aplicação a um cheque da Transformada de Fourier e de um *threshold*.

Aplica-se depois o método PCA, que consiste no cálculo dos valores próprios da matriz de covariância. O maior valor próprio encontrado corresponde ao ângulo de inclinação existente na imagem. Uma vez calculado o ângulo, é aplicado o ângulo inverso de forma a corrigir a orientação do cheque obtendo-se a Figura 3.4.

Figura 3.4 – Imagem após a correcção do ângulo de rotação.

Uma vez feita a correcção do ângulo será preciso encontrar os limites do cheque e corta-lo para finalmente obtermos o cheque pronto para ver os seus montantes reconhecidos.

Para a detecção dos limites foi utilizado a projecção horizontal/vertical que consiste no varrimento horizontal e vertical da imagem, contando o número de pixéis pretos presentes na imagem para cada linha e coluna respectivamente. Ao valor de transição do cheque para não cheque é atribuído 10% do valor do máximo detectado para que a detecção não seja afectada por ruído aleatório.

Finalmente é utilizada a função *crop*, que irá recortar a imagem em função dos valores dados pelo limite, como se pode ver na Figura 3.5.

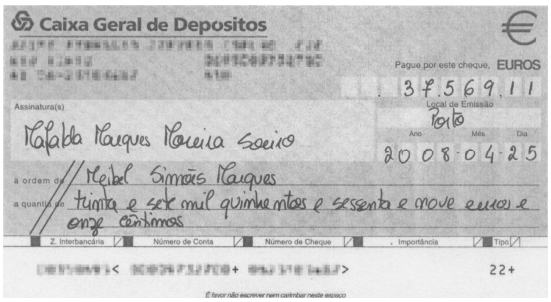


Figura 3.5 - Imagem após o crop.

3.2 Extracção e tratamento do campo de valor da data

Nesta secção é descrita todo o processamento elaborado para a detecção dos dígitos existentes no campo da data, desde a extracção do campo referente à data, passando pelo processamento (eliminação de ruídos) até ao seu reconhecimento.

3.2.1 – Extracção do campo da data

O campo da data foi extraído pelo mesmo método que Coelho [15] utilizou para o método do campo de valor legal e de cortesia. O método consiste na utilização do tamanho do cheque para se obter os campos pretendidos, ou seja, visto o cheques se encontrarem normalizados em relação não só ao tamanho mas também às posições relativas dos vários campos, podemos extrair manualmente qualquer campo pretendido do cheque através do seu comprimento. Neste caso o campo pretendido é o campo da data que se encontra evidenciado por um circulo a vermelho na Figura 3.6. Sabendo que para o desenho de um rectângulo é necessário um ponto (x,y) além da altura e largura, para a extracção do campo da data os valores utilizados foram relativamente ao comprimento do cheque os seguintes: 25,50% para o topo, 67,50% para a esquerda, 32,25% para o comprimento e 4,25% para a altura, obtendo-se assim a figura 3.7.

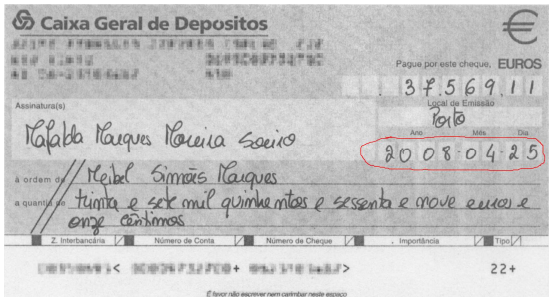


Figura 3.6 – Imagem que contém dentro do circulo o campo desejado

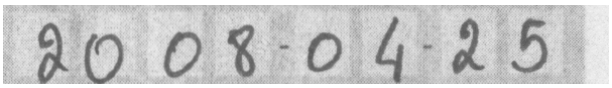


Figura 3.7 – Campo do valor da data

3.2.2 - Pré-processamento

Após se obter o campo da data, aplica-se a binarização obtendo-se uma imagem como se pode ver na Figura 3.8.

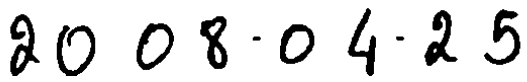


Figura 3.8 – Valor da data após binarização.

Para eliminar o ruído ainda existente na imagem, como é o caso dos pontos entre os dígitos correspondentes aos meses e anos e aos meses e dias, que se pode observar na figura anterior, é utilizado o *blob filtering* [19] obtendo-se a Figura 3.9. O *blob filtering* é um filtro que elimina qualquer objecto encontrado nas imagens que não satisfaz os requisitos impostos pelos dados fornecidos pelo autor.

Figura 3.9– Valor do campo da data após blob filtering.

Para se obter isoladamente cada um dos dígitos utilizou-se o método utilizado por Coelho [15], no qual se executa, inicialmente, um varrimento vertical da imagem para a detecção dos pixéis brancos, sendo registados numa lista. O varrimento vertical começa pela canto esquerdo da imagem e quando detecta um pixel branco guarda a sua posição na lista. Após a primeira detecção, o varrimento continua até acabar de detectar pixéis brancos guardando a posição do último pixel detectado, obtendo assim os limites verticais do primeiro dígito. O varrimento irá continuar guardando consecutivamente os limites verticais até chegar ao fim da imagem. Quando acabar o varrimento vertical começará o horizontal que será feito dentro dos limites verticais guardados na lista. O varrimento horizontal é muito parecido com o

vertical porque começa também no topo da imagem (fazendo o varrimento de cima para baixo) e quando detecta um pixel branco guarda a sua posição numa lista mas, ao contrário do acontece no caso do varrimento vertical, o varrimento já não continua até ao final da imagem. Após ter detectado o primeiro pixel branco começa o varrimento de baixo para cima e guardará o primeiro pixel branco detectado. Esta opção permite salvaguardar situações de descontinuidade entre traços do mesmo dígito, por exemplo, o "5" como se pode ver na Figura 3.10.

Figura 3.10 – Descontinuidade no dígito "5" após a aplicação do filtro "edges".

Figura 3.11 – exemplo de dígitos isolados.

3.2.3 - Extracção de características dos Dígitos

Contorno

Para a extracção da característica de contorno decidiu-se normalizar os dígitos para 32x64 sendo posteriormente usado um filtro de "edges" como se pode ver na Figura 3.12.

Figura 3.12 – Dígitos após a aplicação do filtro edges.

As imagens sofreram uma normalização pois o objectivo era retirar a distância da caixa ao contorno. O *array* terá então 32+64+32+64=192 posições com a distância respectiva (topo, esquerda, direita, baixo) de cada pixel de contorno à caixa.

Grayscale

Para a extracção da característica de grayscale decidiu-se normalizar os dígitos para 8 x16 sendo estes retirados da imagem directamente como se pode ver na Figura 3.13.

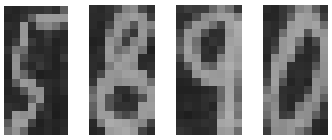


Figura 3.13 – Dígitos em formato grayscale.

No caso desta característica a lista terá 8 x 16=128 posições com o valor do pixel que varia entre 0 e 255.

Black and White (B&W)

Para a extracção da característica de dígitos binarizados decidiu-se normalizar os dígitos para 8x16 sendo estes retirados da imagem directamente após binarização como se pode ver na figura 3.14.

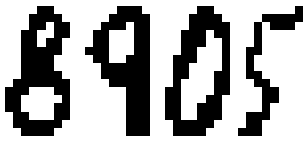


Figura 3.14 – Dígitos binarizados.

No caso desta característica a lista terá 8x16=128 posições com o valor do pixel que, ao contrário do grayscale que varia entre 0 e 255 o B&W, pode ser 0 ou 255.

Podem ser observados mais exemplo de dígitos com as diferentes características no Anexo B.

3.2.4 – Comparação de resultados

As classes de algoritmos estudadas para o reconhecimento dos dígitos foram três (o *MultilayerPerceptron*, a rede de Funções de Base Radial e o SVM). No caso do SVM foram utilizados dois tipos de SVM, um com o *kernel* polinomial e outro com *kernel* RBF. O processo de aprendizagem/teste utilizado foi o 10-fold cross validation.

Os parâmetros utilizados para a realização dos testes com as Máquinas Vectores de Suporte Polinomiais foram: o expoente igual a 1 e a complexidade com o valor de 100. Para as Máquinas Vectores de Suporte com RBF o expoente manteve-se igual a 1 embora a complexidade usada tenha tido o valor 9 e o gama tenha sido igual 0.025.

Para o Multilayerperceptron os parâmetros utilizados foram para os neurónios de camada escondida (n entradas + n saídas) /2, para o "*learning rate*" 0.3, para o "*momentum*" 0.2 e para o "*training time*" foi de 500.

Para as Redes de Função de Base Radial o número de clusters utilizado foi o 10.

Tabela 3.1 – Reconhecimento em percentagem dos vários algoritmos e características

	Contorno	Grayscale	Binarizados	Média da taxa de reconhecimento dos algoritmos
MLP	93,0%	65,8%	78,7%	79,2%
RBF	87,3%	75,1%	70,3%	77,6%
SVM Polinomial	92,1%	65,3%	77,7%	78,4%
SVM com RBF	94,0%	79,3%	81,2%	84,8%
Média das características	91,6%	71,4%	77,0%	

Como podemos observar pela Tabela 3.1 o SVM com RBF obtém os melhores resultados para todas as características, tendo como melhor resultado a de contorno (94%). Estes resultados para o campo da data confirmam os resultados que Coelho [15] obteve para o campo do valor de cortesia. Os resultados do campo da data podem ser vistos mais exaustivamente no Anexo C.

Capítulo 4

Extracção e tratamento do campo de valor legal

Neste capítulo será descrito todo o processamento elaborado para as palavras existentes no campo do valor legal, desde a extracção do campo referente ao valor legal, passando pelo processamento (eliminação de ruídos) até ao reconhecimento das palavras.

4.1 – Extracção do campo de valor legal

A extracção do campo do valor legal é feito utilizando o método já considerado para o campo da data que se encontra evidenciado a vermelho na Figura 4.1. Os valores introduzidos para a extracção do campo do valor legal foram: topo com `33.75%`, esquerda `11.50%`, comprimento com `87.75%` e altura com `8.50%` obtendo assim a Figura 4.2.

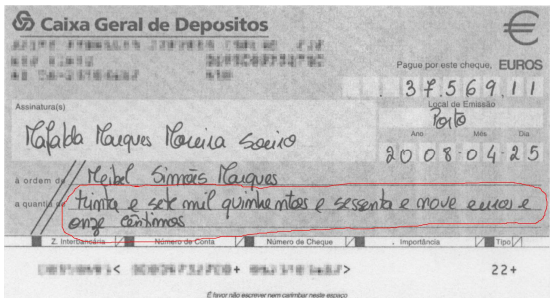


Figura 4.1– Imagem que contém dentro do circulo o campo desejado.

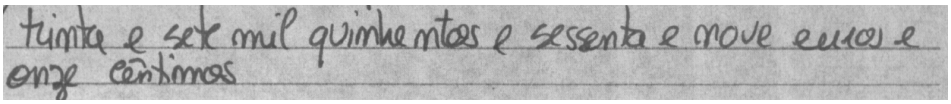


Figura 4.2 – Campo do valor da data após extracção.

4.2 - Pré-processamento do campo de valor legal

Inicialmente, antes de se utilizar qualquer filtro para eliminar o ruído, foi necessário corrigir o ângulo de inclinação visto que para alguns dos cheques havia ângulos de inclinação que rondavam 1 grau, o que poderia ter influência na remoção das linhas horizontais existentes no campo do valor legal. Para a correcção do ângulo foi utilizado mais uma vez o algoritmo anteriormente descrito e já desenvolvido em Coelho [15].

Extracção da linha do cheque

Para a extracção das linhas do cheque é feito um corte na horizontal, dividindo a imagem em duas. Na primeira imagem estará o valor legal da linha superior e na segunda imagem o valor legal da linha inferior. Para cada uma das imagens é feita uma contagem horizontal de pixéis brancos existentes na imagem sendo colocado num array o valor da sua contagem, sendo que cada posição do array indica a linha da imagem de onde o valor provem. Posteriormente o array é percorrido e no caso de o valor presente na posição actual ser superior ou igual a 65% do valor máximo do array e superior a um valor mínimo de pixéis brancos, então na imagem será percorrida a linha horizontal da qual o valor foi retirado. É necessário salvaguardar um valor mínimo de pixéis brancos para garantir que o algoritmo não extraía uma linha em que estejam incluídas palavras. Na imagem será feita um varrimento horizontal ao longo de todo o comprimento da imagem, sendo que em cada X será verificado a altura Y, de forma a descobrir se se trata de uma linha, e então apagá-la, ou se é uma letra que se encontra sobre a linha, mantendo-a. Esta sequência pode observar-se nas Figuras 4.3, 4.4, 4.5.

Figura 4.3 – Valor legal antes da divisão e extracção das linhas do cheque.

Figura 4.4 – Valor legal da linha superior após divisão e extracção da mesma.

Figura 4.5 – Valor legal da linha inferior após divisão e extracção da mesma.

Extracção das linhas

Para a extracção das linhas que são apresentadas quando o campo do valor legal não é utilizado na sua totalidade utilizou-se uma função do Aforge, o *blob filtering*. Exemplo do *blob filtering* encontra-se na passagem da Figura 4.6 para a Figura 4.7.

Figura 4.6– Valor legal antes do blob filtering para a extracção das linhas

Figura 4.7 – Valor legal após blob filtering para a extracção das linhas

Isolamento das palavras

Após o tratamento do ruído, a imagem encontra-se limpa, e serão feitos três tipos de varrimentos. O primeiro varrimento é o varrimento vertical que irá guardar num array a posição do primeiro pixel branco encontrado e do último consecutivamente. De modo a evitar os espaços que possam existir entre as letras de uma palavra foi introduzido um "*threshold*". O segundo varrimento é o horizontal, que terá as mesmas características que o varrimento horizontal feito para os dígitos. O terceiro varrimento é do tipo horizontal mas não é feito para percorrer a imagem toda. Como os cortes para a divisão da linha de cima e de baixo são rígidos é possível encontrarmos na linha de cima traços da linha de baixo e vice versa. Deste modo foi necessário para cada um dos casos fazer o varrimento horizontal e eliminar esses traços. O varrimento na linha de cima era feito a partir dos 70% da imagem até ao final e quando se notava uma descontinuidade ela era eliminada. O varrimento feito na imagem da linha de baixo tinha as mesmas características sendo feita desde o início da imagem até aos 40%. Posteriormente é feito um corte com os pontos que se encontram em ambos os arrays obtendo-se a palavra como se pode ver na figura 4.8. Pode-se encontrar mais exemplos de palavras isoladas no Anexo D.

Figura 4.8 – Exemplos de palavras isoladas.

4.3 - Extracção de características

As características retiradas para a análise de palavras foram o contorno, histograma, ascendentes/descendentes, o tamanho e os loops.

Contorno

A característica de contorno foi retirada e usada da mesma maneira que foi elaborado no caso dos dígitos, sendo que apenas o tamanho da normalização da imagem foi diferente. A normalização das palavra foi de 150x60 (obtendo um *array* de 150+150+60+60=420), sendo posteriormente usado o filtro "edges" obtendo-se imagens como a da Figura 4.9.

Figura 4.9 – Exemplo da característica de dígitos.

Histograma

Para o histograma foi feita a contagem horizontal/vertical dos pixéis e armazenadas num array.

Figura 4.10– a) Imagem da palavra euros b) Exemplo de um histograma horizontal da palavra euros c) Exemplo de um histograma vertical da palavra euros.

Ascendentes/descendentes

Os ascendentes/descendestes de uma palavra são as letras que têm uma altura acima da média da palavra, altura esta que se pode reflectir acima da palavra ou abaixo como é o caso da letra T (ascendente), Z (descendente). Para serem calculados os ascendentes e descendentes percorre-se a imagem contando o número de pixéis e guarda-se o máximo. A imagem é percorrida horizontalmente, de Y=0 até Y=altura, e quando é encontrado 20% do máximo, é desenhada uma linha de cinzento que percorre de X=0 até X=comprimento. É adoptado o mesmo

procedimento para Y=altura até Y=0. Refira-se que logo que seja encontrado o valor de 20%, é desenhada apenas uma linha para cada uma das situações como se pode ver na figura. Após terem sido desenhadas as linhas em cinzento calcula-se a altura dos pixéis em branco que se encontram acima e abaixo das linhas cinzentas indicando-se a sua posição relativamente ao comprimento da imagem. É considerado um ascendente/descendente se forem respeitados dois requisitos: o primeiro requisito é ter uma altura mínima, tanto acima da linha como abaixo para que não seja considerado um qualquer traço ascendente/descendente. No exemplo da Figura 4.11 pode ver-se que o "i" da palavra "dois" está acima da linha e o "o" está abaixo da linha a cinzento. Em ambos os casos não são considerados ascendentes e descendentes respectivamente por não terem uma altura mínima. O outro requisito é de o ascendente/descendente tocar na linha de cinzento, ou seja, mesmo que tenha altura mínima se este não tocar na linha a cinzento não será considerado. Este requisito foi implementado para salvaguardar casos como o da Figura 4.12 na qual a "pinta" do "i" ou o traço horizontal do "t" seja considerado ascendente.

Figura 4.11 – Exemplo de uma imagem com um ascendente.

Figura 4.12 – "pinta" do "i" para não ser considerado ascendente.

Tamanho

O tamanho é obtido através do campo "*length*" da instância de classe bitmap.

Loops

Para o cálculo de loops é utilizada a função floodfill que irá preencher a imagem com a cor branca, ficando a cor preta os loops. Utilizando a função do aforge blob count obtemos o número de blobs como se pode ver na Figura 4.13 e na Figura 4.14.

Figura 4.13 – Imagem antes do floodfill.

Figura 4.14 – Imagem após o floodfill, donde se poderá retirar o número de blobs.

4.4 – Base de dados de palavras

A base de dados de palavras é constituída por 42 palavras diferentes. Esta deveria incluir 43 palavras mas não foi possível encontrar imagens diferentes para todas as palavras. O número total de imagens obtidas foi de 320 podendo-se verificar no Anexo E o número de imagens obtidas para cada palavra. É importante salientar o limitado número de exemplos disponível para algumas palavras, com consequências no treino e estimação de desempenho dos algoritmos descritos na secção seguinte.

Visto o weka não aceitar *strings* de letras, foi necessário atribuir a cada palavra um número, por exemplo, à palavra "duzentos" foi atribuído o número "200" para que a plataforma weka possa fazer o reconhecimento.

4.5 - Comparação de resultados

Para o resultado do reconhecimento das palavras foram usados 3 classificadores presentes no *weka*: o kNN, o MultiLayerPerceptron e o SVM e analisadas três situações: utilizando todas as classes (palavras) possíveis, utilizando apenas as classes com mais de 7 e apenas as classes com mais de 9 exemplos. Para os resultados obtidos com os HMMs foram utilizados os métodos de 10-fold cross validation, base de dados de classes com mais de 7 e 9 imagens. Para os resultados obtidos, após as deformações do método de *Elastic Matching*, com kNN com o mesmo número de classes usadas para os HMMs.

4.5.1 – Resultados obtidos através do WEKA

Para o weka é enviado um ficheiro arrf que contém para cada palavra analisada uma lista com as várias características que podem ser observadas na Tabela 4.1. Os resultados obtidos utilizando o método de teste 10-fold cross validation, encontra-se na Tabela 4.1.

Tabela 4.1 – Resultados obtidos com as várias características, com todas as classes (palavras).

	Contorno	Ascendentes/ Descendentes	Contorno, Ascendentes/ Descendentes	Ascendentes/ Descendentes, Tamanho	Ascendentes/ Descendentes, Tamanho, histograma	Ascendentes/ Descendentes, Tamanho, loops	Ascendentes/ Descendentes, Tamanho, Loops, Histograma	Média da taxa de reconhecimento dos algoritmos
MLP	-------	27.4 %	----------	36.5%	37.6%	37.4 %	39.5%	35.7%
KNN	13.4%	25.3 %	16.9 %	36.5%	33.8%	36.2 %	33.1%	27.9%
SVM Polinomial	17.6%	24.0 %	18.3%	38.6%	39.0 %	38.3%	40.2%	30.8%
SVM com RBF	19.0 %	28.6 %	18.6 %	40.1%	40.7%	41.0%	43.2%	33.0%
Média das características	16.7%	26.3%	17.9%	37.9%	37.8%	38.2%	39.0%	

Como se pode ver pela Tabela 4.1 as Máquinas Vectores de Suporte mostraram mais uma vez ser a melhor escolha para todas as características. O conjunto de características que mostrou um melhor resultado foi o conjunto de características de ascendentes/descendentes, tamanho, loops e histograma. De seguida foi introduzido o *elastic matching*: as imagens de treino eram deformadas pelo elastic matching de forma a ampliar a base de dados e algoritmo assim desenhado era avaliado no conjunto de teste. Os resultados obtidos utilizando este método podem ser observados na Tabela 4.2. Os resultados podem ser vistos mais exaustivamente no Anexo F.

Tabela 4.2 – Resultados obtidos com dois conjuntos de características.

	Ascendentes/Descendentes, Tamanho, loops	Ascendentes/Descendentes, Tamanho, Loops, histograma	Média da taxa de reconhecimento dos algoritmos
MLP	33.3 %	23.0%	28.2%
KNN	35.6 %	20.7%	28.2%
SVM Polinomial	35.6%	21.8%	28.7%
SVM com RBF	44.8%	21.8%	33.3%
Média das características	37.3%	21.8	

Como se pode ver na tabela 4.2 as Máquinas Vectores de Suporte mostraram obter um reconhecimento superior em comparação com o resto dos classificadores. A base de dados de treino contém 656 imagens de palavras enquanto que a de teste contém 87. Através da análise da primeira coluna da Tabela 4.2 pode-se verificar que não ocorreram grandes alterações na percentagem de palavras reconhecidas. Verificando a segunda coluna já se pode chegar à conclusão que não é ideal incluir-se a característica "histograma" no reconhecimento de palavras, já que os níveis de reconhecimento baixam em todos os classificadores.

O desempenho alcançado até ao momento é insuficiente; para se descobrir se a causa desse baixo reconhecimento reside na base de dados limitada, procedeu-se ao estudo dos algoritmos com apenas as palavras mais representadas na base de dados. Os resultados obtidos utilizando apenas as palavras com mais de 7 imagens na base de dados pode-se ver na Tabela 4.3; os resultados com as palavras com mais de 9 exemplos na base de dados encontram-se na Tabela 4.4. Estes resultados podem ser vistos mais exaustivamente no Anexo F.

Tabela 4.3 – Resultados obtidos com classes com mais de 7 imagens.

	Ascendentes/Descendentes, Tamanho, Loops
MLP	49.8%
KNN	45.8%
SVM Polinomial	49.4%
SVM com RBF	51.4%
Média das características	49.1%

Como se pode verificar, o SVM com RBF continua a apresentar o melhor resultado, além de ter havido um aumento significativo da taxa de reconhecimento de palavras. Este aumento do reconhecimento permitiu concluir que o reduzido número de imagens que representam cada classe é efectivamente responsável pela redução do desempenho dos algoritmos. Naturalmente que a redução do número de classes envolvidas no treino também simplifica o problema. Ainda assim é de esperar que as classes menos representadas estejam a contribuir para o mau desempenho do sistema completo.

Tabela 4.4 – Resultados obtidos com classes com mais de 9 imagens.

	Ascendentes/Descendentes, Tamanho, Loops
MLP	73.9%
KNN	74.6%
SVM Polinomial	73.9%
SVM com RBF	78.5%
Média	75.2%

4.5.2 – Resultados obtidos através dos HMM

Para a utilização do algoritmo de HMM a lista de características usada é diferente da utilizada para a do weka pois o HMM envolve uma evolução temporal. Para a demonstração da evolução temporal da palavra decidiu-se normalizar o tamanho da palavra e utilizar janelas, dentro das quais foi verificada se a palavra continha ou não as características desejadas. As características que o programa extraiu foram ascendentes/descendentes, tamanho (retirado antes da normalização) e loops.

Os resultados obtidos usando o método de 10 cross validation encontram-se na Tabela 4.5.

Tabela 4.5 – Resultados obtidos pelos HMMs com o 10 cross validation com números de classes diferentes.

	Ascendentes/Descendentes, Tamanho, Loops
Todas as classes	27%
Classes com mais de 7 imagens	36%
Classes com mais de 9 imagens	73%
Média de características	45%

Na Tabela 4.5 podemos observar que a taxa de reconhecimento aumenta com o aumento do número de imagens existentes nas classes confirmando o resultado em obtido na Tabela 4.4. A utilização de HMMs não revelou melhorias em relação ao desempenho obtido com os modelos mais simples anteriores.

4.5.3 - Resultados obtidos após a utilização do Elastic matching

As imagens foram retiradas da base de dados tendo 75% das imagens sido utilizadas para treino e 25% para teste. As imagens que fizeram parte do treino sofreram deformações através do algoritmo de *Elastic Matching*. As deformações sofridas pelas imagens retiradas da base de dados (*template*) foram até ao grau 4 (como se pode ver na sub-secção 2.1.6 na Figura 2.11, o grau de deformação é tanto maior quanto maior for o n,m. Neste caso o m,n foram até ao número 4). Posteriormente, através do kNN e MultiLayerPerceptron, as imagens de teste foram comparadas com estas imagens e os resultado obtidos podem ser observados na Tabela 4.6 e Tabela 4.7 respectivamente.

Tabela 4.6 – Resultados obtidos pelos *Elastic Matching* com o KNN com números de classes diferentes.

	Ascendentes/Descendentes, Tamanho, Loops
Todas as classes k=1	33%
Classes com mais de 7 imagens e k=1	54%
Classes com mais de 9 imagens e K=3	65%
Média de características	50.7%

Como se pode observar na Tabela 4.6 a taxa de reconhecimento aumenta à medida que o número de imagens que representam cada classe aumenta.

Tabela 4.7 – Resultados obtidos pelos Elastic Matching com o MultiLayerPerceptron com números de classes diferentes.

	Ascendentes/Descendentes, Tamanho, Loops
Todas as classes	59%
Classes com mais de 7 imagens	64%
Classes com mais de 9 imagens	68%
Média de características	63.7%

Como se pode observar na Tabela 4.7 a taxa de reconhecimento aumenta ligeiramente à medida que o número de imagens que representa cada classe aumenta.

Verificando os resultados apresentados pelas várias plataformas aliada aos vários métodos e características pode-se chegar à conclusão que os SVM com RBF obtêm um resultado superior aos restantes obtendo 43.2% de reconhecimento para toda a base de dados. Utilizando a base de dados que inclui apenas classes com mais de 7 imagens o resultado foi de 51.4%. Finalmente foi obtido 78.5% de taxa de sucesso de reconhecimento usando a base de dados que contém as classe com mais de 9 imagens.

É de realçar que embora os HMMs não tenham obtido os melhores resultados pode-se observar que após a utilização de classes nas quais existissem um número significativo de

exemplos o desempenho aumentou consideravelmente. Estes exemplos permitiram ao algoritmo aprender melhor quais as características de cada classe. Os resultados obtidos utilizando a base de dados contendo todas as classes foram de 27%, passando para 36% se se considerarem as classes com mais de 7 imagens, acabando com um reconhecimento de 73% se forem analisadas as classes com mais de 9 imagens.

A utilização de *Elastic Matching* com o kNN, obtem resultados acima dos 50% a partir de classes com imagens superior a 7, embora para classes com mais de 9 imagens este método encontra-se longe dos resultados obtidos pelos restantes. Quando se utiliza o MultiLayerPerceptron os resultados são muito melhores quando comparados com os restantes métodos. Ao reduzirmos a base de dados mantendo apenas as classes com mais de 9 imagens o resultado torna-se bastante inferior podendo-se concluir que provavelmente a taxa de reconhecimento não iria aumentar se a base de dados tivesse imagens suficientes para todas as classes.

Conforme se pode observar na Tabela 4.1, em relação às características para os conjuntos de combinações de ascendente/descendente, tamanho, loops e histograma não se obteve uma diferença significativa na taxa de sucesso em relação à combinação de ascendente/descendente, tamanho e loops, sendo estes resultados em média de 39% e 38,2% respectivamente. No entanto, conforme se pode verificar na Tabela 4.2, já se pode observar uma redução drástica na taxa de reconhecimento quando se comparam os resultados para as características Ascendentes/Descendentes, Tamanho, Loops, histograma assistindo-se a uma diminuição da eficácia do reconhecimento de 39% para 21.8% enquanto que a combinação de ascendente/descendente, tamanho e loops se manteve nos 38.2%.

Capítulo 5

5.1 – Conclusões

Pode-se concluir que os objectivos propostos em 18 de Fevereiro foram atingidos até 30 de Junho (20 semanas).

Este trabalho incidiu no estudo e aplicação de métodos de classificação num sistema de processamento de cheques portugueses. Investigaram-se diversos métodos para detectar automaticamente a data e o valor legal presentes no cheque. Avaliaram-se diversas configurações dos métodos de aprendizagem automática e diferentes conjuntos de características extraídas do cheque.

Como se pôde observar, houve além de uma melhoria do processamento dos campos do cheque, um aumento da eficácia na extracção/reconhecimento dos dígitos. A melhoria da taxa de reconhecimento dos dígitos do campo de cortesia para o campo da data foi de 1%.

Conseguiu-se ainda elaborar o processamento do campo do valor legal, além de se demonstrar que se se tivesse utilizado uma base de dados com um maior número de imagens, de forma facilitar o processo de aprendizagem das características discriminantes de cada classe, se obteria uma taxa do reconhecimento pouco inferior a 80%.

5.2 – Perspectivas Futuras

A continuação do desenvolvimento do protótipo em estudo passará principalmente pelo aumento da base de dados das palavras de forma a verificar os resultados obtidos nesta dissertação. No futuro dever-se-á também investir no processamento dos vários campos do cheque de forma a permitir retirar-se as várias características tanto dos dígitos como, e principalmente, das palavras. Uma das melhorias no que se refere ao processamento das palavras seria implementar um *threshold* iterativo aquando da separação das várias palavras. Outra melhoria a implementar seria a correcção da inclinação das letras nas palavras pois

permitiria a melhor obtenção dos ascendentes/descendentes em situações de inclinação demasiado acentuada. A extracção das características apresentadas deveria sofrer um melhoramento além de se dever prosseguir com a extracção de novas características de forma a intensificar o estudo. No reconhecimento de palavras dever-se-ia introduzir uma função que conseguisse detectar perante qual dos três tipos distintos de escrita nos encontramos (palavras escritas com formato de palavras escritas à máquina, palavras com formato manuscrito e palavras em que existe uma mistura entre letras à máquina e à mão). Embora nesta tese não se tenham considerado imagens com palavras escritas somente à máquina, foram consideras palavras em que existe uma mistura entre os dois tipos de escrita, o que inevitavelmente diminui a taxa de reconhecimento.

A extracção do campo da assinatura de forma a detectar a falsificação de assinaturas também seria uma melhoria que se deveria introduzir no sistema.

Embora não seja essencial para o desempenho do sistema melhorar o módulo da aplicação web e o seu aspecto gráfico, tal é um factor relevante para fases mais avançadas do sistema já que permitiria a introdução de mais do que uma imagem de cada vez na base de dados.

Finalmente dever-se-ia iniciar um conjunto de testes em ambiente de trabalho real numa instituição bancária onde fosse instalado o protótipo de teste. Assim seria possível verificar a aplicação e desempenho do sistema perante as situações do dia a dia da instituição e corrigir eventuais falhas para que depois de devidamente testado o sistema pudesse ser largamente produzido e implementado em todas as instituições bancárias.

Referências

[1] OCR (Optical Character Recognition), ICR (Intelligent character recognition). Disponível em http://www.datacap.com/primer/recog/. Acesso em 12/Junho/2008

[2] Ian H. Witten & Eibe Frank, "Data Mining, Pratical Machine Learning Tools and Techniques, Second Edition", Elsevier, 2005

[3] SVM (Support Vector Machines). Disponível em http://www.dtreg.com/svm.htm, Acesso em 12/Junho/2008

[4] SVM (Support Vector Machines). Disponível em http://www.dtreg.com/mlfn.htm, Acesso em 12/Junho de 2008

[5] Radial Basis Function Networks. Disponível em http://documents.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.5.2.html, Acesso em 12/Junho/2008

[6] D. Guillevic, C.Y.Suen. *HMM-KNN Word Recognition Engine for Bank Cheque Processing* in Fourteenth International Conference on Volume 2, Issue , 16-20 Aug 1998 Page(s):1526 - 1529 vol.2

[7] A. Jain, Y.Zhon, S. Lakshmanan, Object Matching Using Deformable Templates, IEEE Trasn. Pattern Anal. Mach. Intell, vol 18, no. 3, March 1996 pp 267- 278,

[8] M. Ávila. *Optimisation des Modeles Markoviens pour la Reconnaissance de L'escrit*. Phd thesis, Université de Touen, France, 1996

[9] M.Cotê. *Utilisation dún modèle d'accés lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs*. Phd thesis, Ècole Nationale Uupérieure des Télécommunications, France, 1997.

[10] D. Guillevic. *Unconstrained handwriting recognition applied to the processing of banks cheques*. Phd thesis, Concordia University, Canada, December 1995.

[11] N.Gorsky, V. Anisimov, E. Augustin, O. Baret, S. Maximov. *Industrial bank check processing : the A2iA CheckReaderTM* in International Journal on Document Analysis and Recognition on volume 3, 4 Dezembro 2000 Page(s) :196-206

[12] N. R. Gomes, L. L. Lee. *Feature extraction based on fuzzy set theory for handwriting recognition*. In Sixth International Conference on Document Analysis and Recognition. IEEE Computer Society, 2001.

[13] Uchida e Sakoe Handwritten character recognition using elastic matching based on a class-dependent deformation model **Seventh International Conference on** 3-6 Aug. 2003 page(s): 163- 167 vol.1

[14]M. Morita, L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. *An hmm-mlp hybrid system to recognize handwritten dates*. In Proceedings of the International Joint Conference on Neural Networks, pages 867–872. IEEE Press, 2002.

[15]F. Coelho. Sistema Automático de Reconhecimento do montante do Cheque. Mestrado Integrado, Universidade do Porto, Fevereiro 2008

[16]Spatial Filters – Median Filters. Disponível em http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm, Acesso em 12 de Junho de 2008

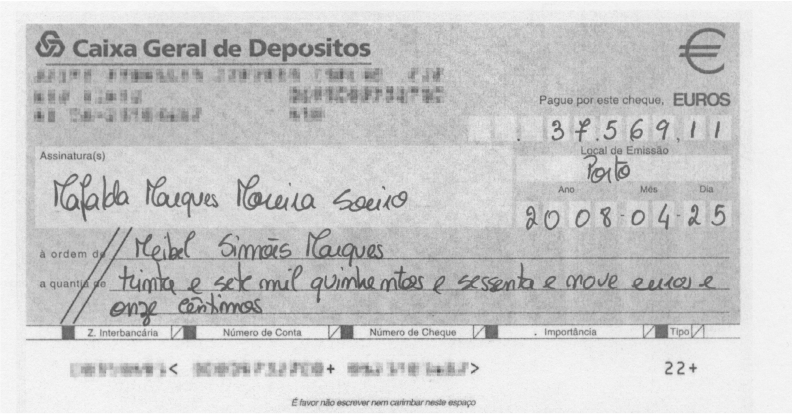
[17]Transformada de Fouier. Disponível em http://pt.wikipedia.org/wiki/Transformada_de_Fourier Acesso em 12 de Junho de 2008

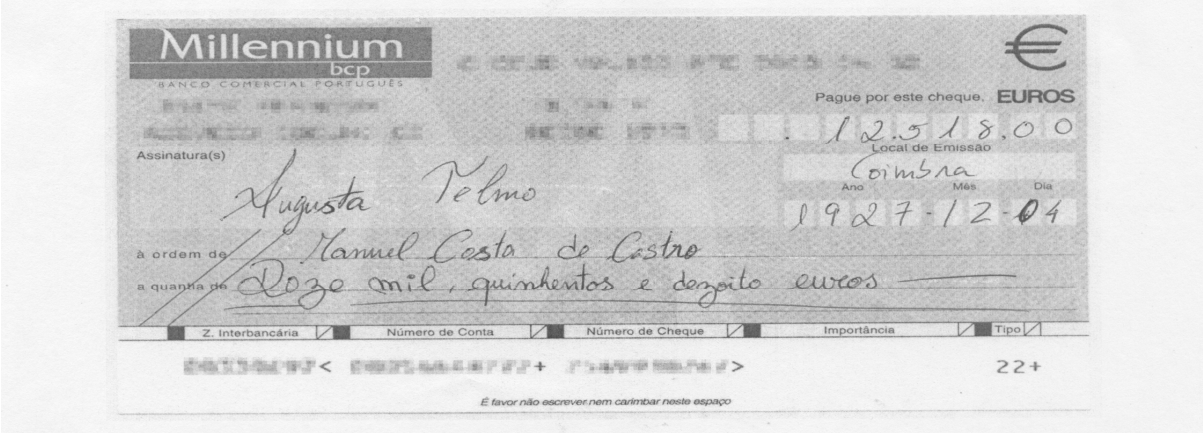
[18]PCA (principal Components Analisys. Disponível em http://kybele.psych.cornell.edu/~edelman/Psych-465-Spring-2003/PCA-tutorial.pdf, Acesso em 12 de Junho de 2008

[19]Blob filtering. Disponível em http://aforge.googlecode.com/svn/tags/AForge-1.6.0/Release%20notes.txt, Acesso em 12 de Junho de 2008

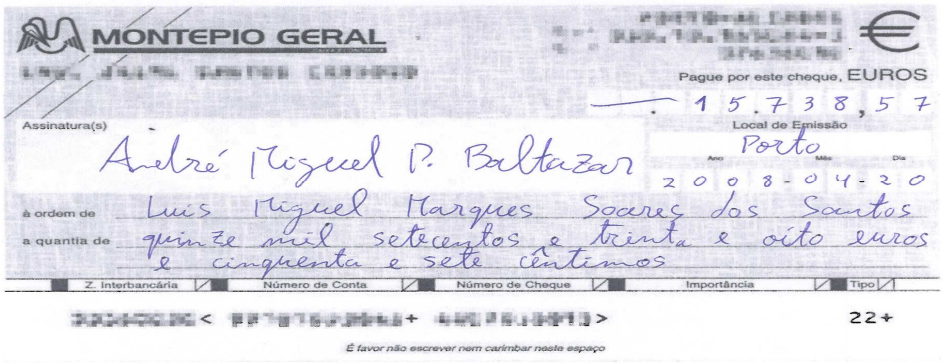
Anexos

ANEXO A: imagens de cheques usado





46 ANEXO A: imagens de cheques usado


MONTEPIO GERAL

Pague por este cheque, EUROS

15.738,57

Assinatura(s)

André Miguel P. Baltazar

Local de Emissão

Porto

Ano Mês Dia

2008-04-20

à ordem de Luis Miguel Marques Soares dos Santos

a quantia de quinze mil setecentos e trinta e oito euros e cinquenta e sete cêntimos

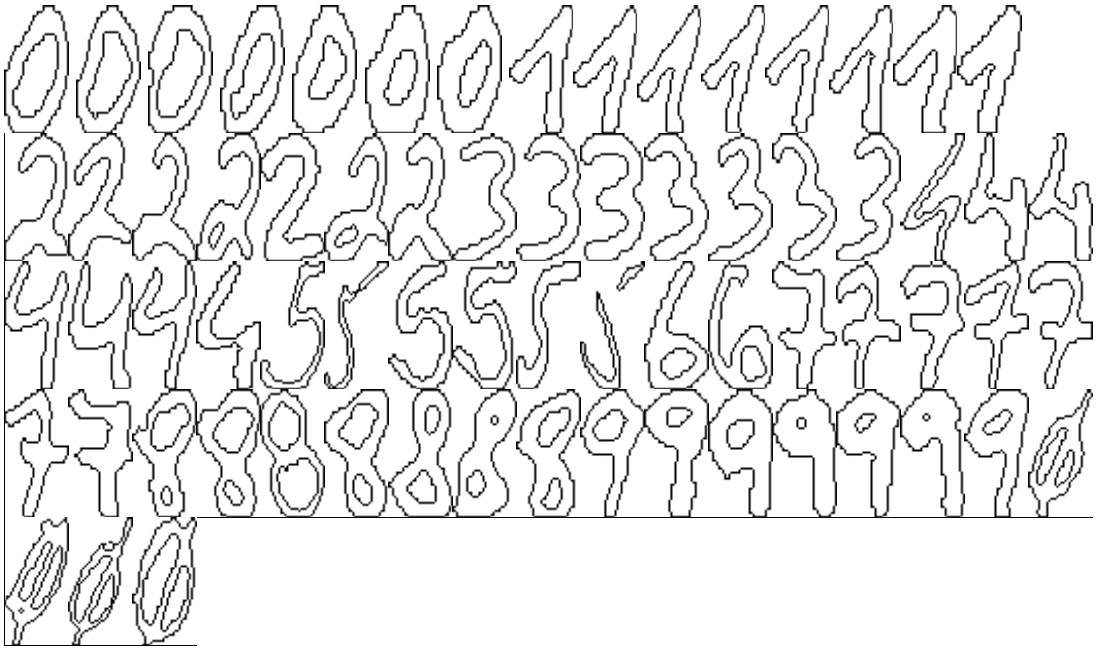
Z. Interbancária | Número de Conta | Número de Cheque | Importância | Tipo

22+

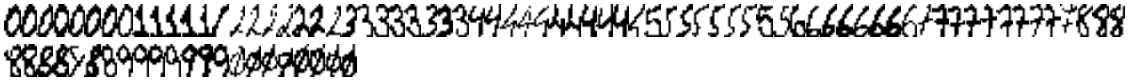
É favor não escrever nem carimbar neste espaço

ANEXO B: imagens de algarismos usado para o treino de cada uma das características

Imagens com a característica de contorno



Imagens com a característica B&W



Imagens com a característica Greyscale

Anexo C: Resultados obtidos no weka – Reconhecimento do valor da data

Resultados obtidos com a característica de contorno

SMO com RBF

Scheme: weka.classifiers.functions.SMO -C 9.0 -E 1.0 -G 0.03 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

Relation: digits

Instances: 316

Attributes: 193

Test mode: 10-fold cross-validation

Correctly Classified Instances	297	93.9873 %
Incorrectly Classified Instances	19	6.0127 %
Kappa statistic	0.9323	
Mean absolute error	0.149	
Root mean squared error	0.2638	
Relative absolute error	91.8953 %	
Root relative squared error	92.65 %	
Total Number of Instances	316	

=== Confusion Matrix ===

```
  a  b  c  d  e  f  g  h  i  j  k   <-- classified as
 34  1  0  0  0  0  0  0  0  0  0 |  a = 0
  0 55  1  0  0  0  0  0  1  0  0 |  b = 1
  0  1 41  0  0  0  1  0  0  0  0 |  c = 2
  0  1  1 25  0  0  0  0  0  2  0 |  d = 3
  0  4  0  0 12  0  0  0  0  0  0 |  e = 4
  0  0  0  0  0 24  0  0  0  0  0 |  f = 5
  0  0  0  0  0  0 10  0  0  0  0 |  g = 6
  0  0  0  0  0  0  0 33  0  0  0 |  h = 7
  1  1  1  0  0  0  0  0 22  0  0 |  i = 8
  0  1  0  0  0  0  0  0  0 28  0 |  j = 9
  2  0  0  0  0  0  0  0  0  0 13 |  k = 10
```

Resultados obtido com os dígitos originais

SMO com RBF

Scheme: weka.classifiers.functions.SMO -C 100.0 -E 1.0 -G 0.2 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

Relation: digits

Instances: 213

Attributes: 129

Correctly Classified Instances	169	79.3427 %
Incorrectly Classified Instances	44	20.6573 %
Kappa statistic	0.7677	
Mean absolute error	0.1514	
Root mean squared error	0.2683	
Relative absolute error	92.9561 %	
Root relative squared error	94.0542 %	
Total Number of Instances	213	

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	k	<-- classified as
35	1	1	0	0	0	0	0	0	0	0	a = 0
0	23	3	0	0	1	0	0	0	0	0	b = 1
2	2	17	1	0	1	0	1	0	0	0	c = 2
0	0	3	10	0	0	0	0	0	0	0	d = 3
0	3	0	0	6	1	0	0	0	1	0	e = 4
0	0	1	1	0	20	0	0	0	0	0	f = 5
1	0	1	0	0	1	5	0	1	0	0	g = 6
0	0	2	0	0	1	0	18	0	0	0	h = 7
0	2	3	0	0	1	0	1	11	0	0	i = 8
0	0	1	0	0	2	0	0	1	14	0	j = 9
0	1	1	0	0	0	0	1	0	0	10	k = 10

Resultados obtido com os dígitos B&W

SMO com RBF

Scheme: weka.classifiers.functions.SMO -C 15.0 -E 1.0 -G 0.04 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

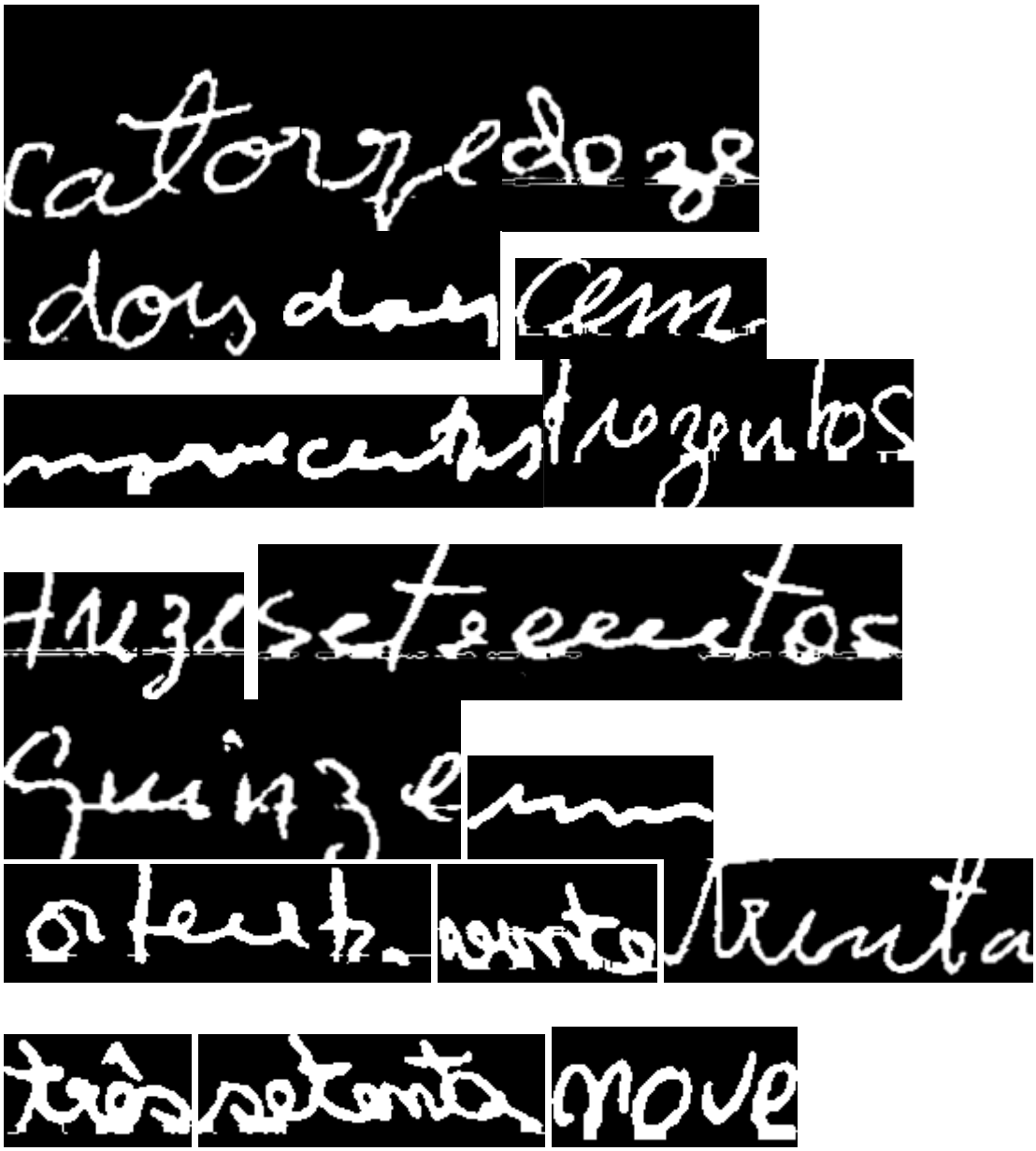
Correctly Classified Instances	164	81.1881 %
Incorrectly Classified Instances	38	18.8119 %
Kappa statistic	0.7884	
Mean absolute error	0.1501	
Root mean squared error	0.2659	
Relative absolute error	92.124 %	
Root relative squared error	93.1952 %	
Total Number of Instances	202	

=== Confusion Matrix ===

```
 a  b  c  d  e  f  g  h  i  j  k   <-- classified as
34  0  0  0  0  1  0  0  0  0  0 |  a = 0
 0 23  2  0  0  0  0  0  1  0  0 |  b = 1
 2  1 16  1  0  1  0  0  1  0  0 |  c = 2
 0  0  3  8  0  0  0  0  0  1  0 |  d = 3
 0  3  0  0  6  0  0  1  0  1  0 |  e = 4
 0  1  0  1  0 18  0  0  0  0  0 |  f = 5
 1  1  1  0  0  1  4  0  0  0  0 |  g = 6
 0  1  1  0  0  0  0 18  0  0  0 |  h = 7
 0  3  2  0  0  0  0  0 10  1  0 |  i = 8
 0  0  0  0  0  0  0  0  2 17  0 |  j = 9
 0  2  1  0  0  0  0  0  0  0 10 |  k = 10
```

ANEXO D: imagens de algarismos usado para o treino de cada uma das características

Algumas imagens binarizadas após a aplicação do filtro de inversão de cores



Anexo E: Número palavras diferentes e número de imagens de cada palavra

Identificador	palavra	Números de imagens de cada palavra
101	Um	8
102	Dois	9
103	Três	8
104	Quatro	5
105	Cinco	13
106	Seis	9
107	Sete	8
108	Oito	7
109	Nove	9
115	Dez	5
11	Onze	4
12	Doze	8
13	Treze	2
14	Catorze	3
15	Quinze	5
16	Dezasseis	0
17	Dezassete	2
18	Dezoito	7
19	Dezanove	4
20	Vinte	4
30	trinta	7
40	Quarenta	6
50	Cinquenta	8
60	Sessenta	4
70	Setenta	9
80	Oitenta	6
90	Noventa	10
100	Cem	2
200	Duzentos	6
300	Trezentos	5
400	Quatrocentos	10
500	Quinhentos	7
600	Seiscentos	4
700	Setecentos	4
800	Oitocentos	6
900	Novecentos	8
1000	Mil	10
100000	Milhão	0
100001	milhões	5
111	E	34
112	Cêntimos	28
113	euros	17
110	cento	4
total		320

Anexo F : Resultados obtidos no weka – Reconhecimento do valor legal

Ascendentes/descendentes+loops+histograma+tamanho

SMO com RBF

Scheme: weka.classifiers.functions.SMO -C 1.0E7 -E 1.0 -G 0.07 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

Relation: words

Instances: 329

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	142	43.1611 %
Incorrectly Classified Instances	187	56.8389 %
Kappa statistic	0.4018	
Mean absolute error	0.0457	
Root mean squared error	0.1506	
Relative absolute error	99.6896 %	
Root relative squared error	99.4888 %	
Total Number of Instances	329	

=== Confusion Matrix ===

```
  a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab ac ad ae af ag ah ai aj ak al am an ao ap   <-- classified as
 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  a = 11
 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 |  b = 12
 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  c = 13
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 |  d = 14
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 1 1 0 0 0 0 |  e = 15
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  f = 17
 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 |  g = 18
 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 |  h = 19
 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 |  i = 20
 2 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 |  j = 30
 0 0 0 0 1 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |  k = 40
 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 |  l = 50
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 0 0 0 1 0 0 0 0 0 0 0 0 0 |  m = 60
 0 0 0 0 0 0 0 0 0 0 0 0 0 4 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 |  n = 70
 0 1 0 0 0 0 0 0 0 0 0 0 2 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 |  o = 80
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 2 0 0 |  p = 90
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 |  q = 100
```

```
 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  1  0  0  0  0  1  2  0  0  0  0  0  0  0  0  0  0  0  0  |  r = 101
 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  2  1  0  0  1  0  0  0  0  2  0  2  0  0  0  0  0  0  0  0  0  0  0  |  s = 102
 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  4  0  0  1  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  |  t = 103
 0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  |  u = 104
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  8  0  0  0  0  0  2  2  0  0  0  0  0  0  0  0  0  1  0  0  |  v = 105
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  4  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  1  0  0  |  w = 106
 0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  2  0  2  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  |  x = 107
 0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  0  0  1  0  0  0  0  0  0  0  0  1  0  0  |  y = 109
 0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  2  0  1  0  0  1  0  0  0  0  0  0  0  0  0  2  0  0  |  z = 108
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  | aa = 110
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0 30  1  0  0  0  0  0  0  0  0  0  1  0  0  | ab = 111
 0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  2 22  0  0  0  0  0  0  0  0  0  0  0  | ac = 112
 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  1  0  1  0  0  0  1 11  0  0  0  0  0  0  0  0  1  0  0  | ad = 113
 1  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  | ae = 115
 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  1  0  2  0  0  0  0  0  0  0  0  0  | af = 200
 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  0  1  0  0  0  0  0  0  0  | ag = 300
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  1  0  0  | ah = 400
 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  2  0  1  0  0  0  1  0  0  0  1  0  0  | ai = 500
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  1  1  0  0  | aj = 600
 0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  | ak = 700
 0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  | al = 800
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  5  1  0  0  | am = 900
 0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  2  1  0  0  0  0  0  0  1  0  0 12  0  0  | an = 1000
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  | ao = 1000000
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  2  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  | ap = 1000001
```

Com um arrf de treino e um de teste

Com ascendentes/descendentes, loops e tamanho

SMO com RBF

Scheme: weka.classifiers.functions.SMO -C 100000.0 -E 1.0 -G 1.0 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

Relation: words

Instances: 656

=== Evaluation on test set ===

=== Summary ===

Correctly Classified Instances	39	44.8276 %
Incorrectly Classified Instances	48	55.1724 %
Kappa statistic	0.4327	
Mean absolute error	0.0457	
Root mean squared error	0.1504	
Relative absolute error	98.4225 %	
Root relative squared error	98.6638 %	

Total Number of Instances 87

=== Confusion Matrix ===

```
 a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab ac ad ae af ag ah ai aj ak al am an ao ap   <-- classified as
 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  a = 11
 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  b = 12
 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  c = 13
 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  d = 14
 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  e = 15
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  f = 17
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  g = 18
 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  h = 19
 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  i = 20
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  j = 30
 0 0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  k = 40
 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  l = 50
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  m = 60
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  n = 70
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  o = 80
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  p = 90
 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  q = 100
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  r = 101
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  s = 102
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |  t = 103
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  u = 104
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  v = 105
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  w = 106
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 |  x = 107
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  y = 109
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |  z = 108
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | aa = 110
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ab = 111
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ac = 112
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ad = 113
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ae = 115
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | af = 200
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 | ag = 300
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 | ah = 400
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ai = 500
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 | aj = 600
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 | ak = 700
 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 | al = 800
 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | am = 900
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 | an = 1000
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ao = 1000000
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 | ap = 1000001
```

Com 21 classes e com ascendentes/descendentes/tamanho/loops

SMO com RBF

=== Run information ===

Scheme: weka.classifiers.functions.SMO -C 100.0 -E 1.0 -G 1.5 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

Relation: words

Instances: 247

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	127	51.417 %
Incorrectly Classified Instances	120	48.583 %
Kappa statistic	0.4814	
Mean absolute error	0.045	
Root mean squared error	0.1492	
Relative absolute error	100.0971 %	
Root relative squared error	99.7803 %	
Total Number of Instances	247	

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	aa	ab	ac	ad	ae	af	ag	ah	ai	aj	ak	al	am	an	ao	ap	<-- classified as
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| a = 11
0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	| b = 12
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| c = 13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| d = 14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| e = 15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| f = 17
0	0	0	0	0	0	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	| g = 18
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| h = 19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| i = 20
0	0	0	0	0	0	2	0	0	1	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| j = 30
0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	| k = 40
0	0	0	0	0	0	1	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	| l = 50
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| m = 60
0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	| n = 70
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| o = 80
0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	| p = 90
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| q = 100
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	| r = 101
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	| s = 102
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	| t = 103
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| u = 104
0	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0	2	1	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	| v = 105
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| w = 106
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	| x = 107
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| y = 109
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	4	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	| z = 108
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| aa = 110
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| ab = 111
0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	2	0	1	0	0	0	0	18	0	0	0	0	0	0	0	0	0	2	0	0	0	| ac = 112
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	1	0	0	0	1	0	0	1	| ad = 113
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| ae = 115
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| af = 200
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| ag = 300
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	1	0	0	0	| ah = 400
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	0	1	1	0	0	| ai = 500
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| aj = 600
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	| ak = 700
0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	| al = 800

```
0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 4 0 0 0 | am = 900
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 2 0 0 0 4 0 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0 0 0 9 0 0 0 | an = 1000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | ao = 1000000
0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 | ap = 1000001
```

Com 7 classes e com ascendentes/descendentes/tamanho/loops

SMO com RBF

=== Run information ===

Scheme: weka.classifiers.functions.SMO -C 100000.0 -E 1.0 -G 0.01 -A 250007 -L 0.0010 -P 1.0E-12 -N 0 -R -V -1 -W 1

Relation: words

Instances: 130

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	102	78.4615 %
Incorrectly Classified Instances	28	21.5385 %
Kappa statistic	0.7377	
Mean absolute error	0.0404	
Root mean squared error	0.1413	
Relative absolute error	97.8312 %	
Root relative squared error	100.0306 %	
Total Number of Instances	130	

=== Confusion Matrix ===

```
  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z aa ab ac ad ae af ag ah ai aj ak al am an ao ap   <-- classified as
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  a = 11
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = 12
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = 13
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  d = 14
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = 15
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = 17
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  g = 18
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  h = 19
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  i = 20
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  j = 30
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  k = 40
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  l = 50
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  m = 60
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  n = 70
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  o = 80
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  p = 90
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  q = 100
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  r = 101
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  2  0  0  0  0 |  s = 102
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  t = 103
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  u = 104
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  5  0  0  0  0  0  0  5  1  0  0  0  0  0  0  0  0  1  0  0  0 |  v = 105
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  w = 106
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  x = 107
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  y = 109
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  z = 108
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | aa = 110
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 34  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ab = 111
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0 22  1  0  0  0  0  0  0  0  1  0  0 | ac = 112
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2 13  0  0  0  2  0  0  0  0  0  0  0  0 | ad = 113
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ae = 115
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | af = 200
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ag = 300
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  9  0  0  0  0  0  0  0  0  0 | ah = 400
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ai = 500
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | aj = 600
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ak = 700
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | al = 800
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | am = 900
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  2  2  0  0  0  0  0  0  0  0  0 14  0  0 | an = 1000
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ao = 1000000
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | ap = 1000001
```