(SCC 5830) Processamento de Imagens
Prof. Dr. João do E.S. Batista Neto
Prof. Dr. Francisco Aparecido Rodrigues

# Relatório do Projeto de Classificação de Formas

Alceu Ferraz Costa alceufc@icmc.usp.br
Frizzi San Roman Salazar frizzi.ss@gmail.com

## 1. Introdução

Este projeto tem como objetivo explorar técnicas de classificação e agrupamento (*clustering*) de imagens com base em descritores de forma. Para realizar os experimentos, foi utilizada uma base de imagens de folhas divididas em seis categorias, como mostra a figura 1.1.


Figura 1.1: Exemplos de folhas para cada uma das seis categorias existentes na base utilizada neste trabalho.

O restante deste relatório está organizado da maneira que se segue. Na parte 1 são discutidas as funções implementadas para extração das características de forma. Elas compõem o vetor de características que é utilizado nas tarefas de classificação e agrupamento. Na parte 2 são discutidos os classificadores bayesianos implementados. Na parte 3 são discutidos os experimentos de classificação utilizando redes neurais e o classificador k-vizinhos da ferramenta Weka. Na seção 4 é discutido o método de seleção de atributos CFS e de projeção de atributos por PCA. Por fim, na seção 5 é discutida a implementação feita do método de agrupamento k-means.

# 2. Parte 1

Tabela 2.1. Código Matlab desenvolvido para parte 1.

| Arquivo .m | Descrição |
|---|---|
| generateLeavesArffFile.m | Aplica as funções de extração de características de forma às imagens de folha e gera um arquivo ARFF com os vetores de características. |
| extractBorder.m | Extrai o contorno paramétrico de uma forma representada por uma imagem binária usando o algoritmo de seguimento do contorno. |
| smoothBorder.m | Realiza uma filtragem passa-baixas no domínio da freqüência do contorno paramétrico da forma. Uma vez que a filtragem é feita utilizando um filtro gaussiano, o algoritmo toma como parâmetro de entrada o valor de sigma. Quanto menor o valor de sigma, mais suave a forma fica após o processo de filtragem. |
| extractCurvatureFourierDescriptors.m, extractFourierDescritptors.m, getArea.m, getBendingEnergy.m, getCenterOfMass.m, getDiameter.m, getNumberPeaks.m getPerimeter.m | Funções de extração de características de forma utilizadas para obter o vetor de características de uma imagem. |

Esta parte do projeto teve como objetivo realizar a extração de características de forma das imagens de folha. Os descritores de forma são arranjados na forma de vetores de características que são utilizadas nas tarefas de classificação e agrupamento realizadas nas etapas subseqüentes deste trabalho.

Antes de se realizar a extração de características, foi realizado um pré-processamento das imagens. Inicialmente, as imagens foram binarizadas para separar as folhas do fundo da imagem. Foi então aplicada a cada imagem a operação de fechamento morfológico para remoção de ruídos (arquivo **preprocessInputImage.m**). Por fim, foi utilizado o algoritmo de seguimento de contorno (arquivo **extractBorder.m**) para se obter uma representação paramétrica do contorno da forma. Esse contorno é então suavizado através de um filtro passa-baixas gaussiano (arquivo **smoothBorder.m**). A figura 2.1 mostra uma imagem binarizada, seu contorno paramétrico e os resultados do processo de suavização para os valores de sigma = 89 e sigma = 8,9.

Os descritores de forma utilizados foram:

- Área (**getArea.m**): Número de pixels pertencentes à forma.
- Perímetro (**getPerimeter.m**): Consiste na soma das distâncias entre todos os pontos vizinhos do contorno paramétrico.
- Diâmetro (**getDiameter.m**): Maior distância entre dois pontos do contorno.
- Centróide (**getCentroid.m**): Média das coordenadas dos pixels que compõem o contorno do objeto. Corresponde a dois valores numérico.

- *Bending Energy* (**getBendingEnergy.m**): Corresponde à energia do sinal de curvatura. Para extrair o sinal de curvatura, é utilizada a função **extractCurvature.m**.
- Número de Picos no Sinal de Curvatura (**getNumberPeaks.m**): Número de picos no sinal de curvatura.
- Descritores de Fourier (**extractFourierDescriptors.m**): Descritores de Fourier do contorno paramétrico da forma. Para sua extração, o contorno paramétrico é representado por um sinal complexo u(n) = x(n) + j y(n). Corresponde a 21 valores numéricos.
- Descritores de Fourier do Sinal de Curvatura (**extractCurvatureFourier Descriptors.m**): descritores de Fourier do sinal de curvatura da forma. Corresponde a 21 valores numéricos.

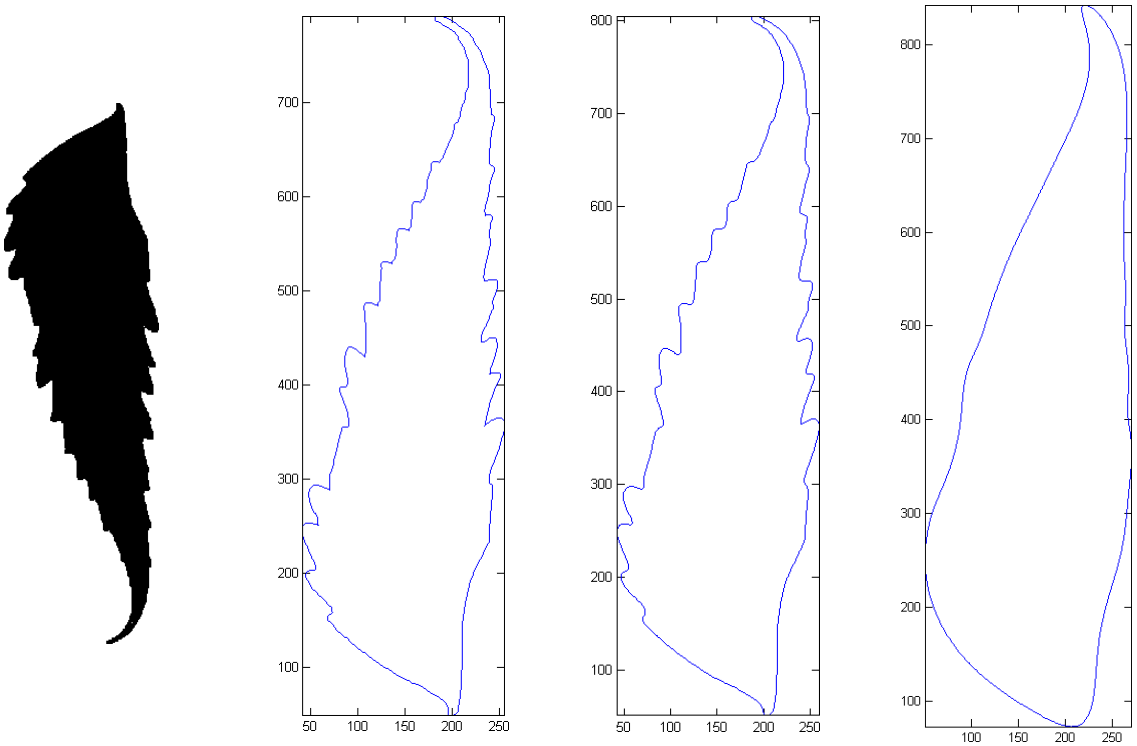


Figura 2.1. (a) Imagem binarizada. (b) Contorno parmétrico. (c-d) Resultados da suavição para valores de sigma = 89 e sigma = 8,9 respectivamente.

# 3. Parte 2: Classificação Bayesiana

Tabela 3.1: Código Matlab desenvolvido para parte 2.

| **Arquivo .m** | **Descrição** |
|---|---|
| classifierEuclideanDistance.m | Calcula a distância Euclidiana entre uma amostra e as classes, classifica segundo a distância menor. Retorna a matriz de confusão por meio da função **confusion** do Matlab. |
| classifierMahalanobisDistance.m | Calcula a distância de Mahalanobis entre uma amostra e as classes, classifica segundo a distância menor. Retorna a matriz de confusão por meio da função **confusion** do Matlab. |
| classifierParametricBayesian.m | Realiza a classificação paramétrica, calculando a probabilidade que uma amostra pertença a uma das classes. Retorna a matriz de confusão por meio da função **confusion** do Matlab. |
| classifierParzenWindows.m | Realiza classificação não paramétrica utilizando janelas de Parzen. Retonar as matrizes *targets* e *outputs* que podem ser utilizadas para obter a matriz de confusão por meio da função **confusion** do Matlab. |

## 3.1 Classificadores de Distância Mínima

Os classificadores de distância mínima calculam a distância entre o padrão desconhecido e cada um dos vetores protótipos. O método escolhe a menor distância para tomar uma decisão.

Neste projeto o classificador por distância mínima foi utilizado para se classificar os dados compostos pelos descritores de forma gerados na primeira parte que compõem o vetor de características.

As matrizes de confusão para a distância mínima euclidiana e distância mínima de Mahalanobis são mostradas nas tabelas 3.2 e 3.3 respectivamente. Os atributos foram estandarizados através da função **zscore** do Matlab.

Tabela 3.2. Matriz de confusão para o classificador de distância mínima euclidiana.

| | | **Classe Predita** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 9 | 0 | 0 | 1 | 0 | 0 |
| | **bg** | 0 | 8 | 2 | 0 | 0 | 0 |
| | **cg** | 0 | 2 | 8 | 0 | 0 | 0 |
| | **lg** | 0 | 1 | 1 | 8 | 0 | 0 |
| | **me** | 1 | 0 | 0 | 0 | 8 | 1 |
| | **pe** | 0 | 0 | 1 | 1 | 1 | 9 |

Tabela 3.3. Matriz de confusão para o classificador de distância mínima de Mahalanobis.

| | | Classe Predita | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 10 | 0 | 0 | 0 | 0 | 0 |
| | **bg** | 0 | 10 | 0 | 0 | 0 | 0 |
| | **cg** | 0 | 0 | 10 | 0 | 0 | 0 |
| | **lg** | 0 | 0 | 0 | 10 | 0 | 0 |
| | **me** | 0 | 0 | 0 | 0 | 10 | 0 |
| | **pe** | 0 | 0 | 0 | 0 | 0 | 12 |

## 3.2 Classificador Bayesiana Paramétrica

O uso do classificador bayesiano geralmente é baseado no pressuposto de uma expressão analítica para as várias funções densidade, seguidas da estimativa dos parâmetros das expressões a partir de amostra de cada classe.

Neste projeto o classificador bayesiano foi utilizado para se classificar os dados compostos pelos descritores de forma gerados na primeira parte que compõem o vetor de características. A matriz de confusão é mostrado na tabela 3.4. Os atributos foram estandarizados através da função **zscore** do Matlab.

Tabela 3.4. Matriz de confusão para o classificador bayesiana paramétrica.

| | | Classe Predita | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 7 | 0 | 0 | 2 | 1 | 0 |
| | **bg** | 0 | 8 | 2 | 0 | 0 | 0 |
| | **cg** | 0 | 2 | 8 | 0 | 0 | 0 |
| | **lg** | 0 | 1 | 1 | 8 | 0 | 0 |
| | **me** | 1 | 0 | 0 | 0 | 9 | 0 |
| | **pe** | 0 | 0 | 1 | 1 | 8 | 2 |

## 3.3 Classificador por Janelas de Parzen

O classificador por janelas de Parzen é um método não paramétrico que pode ser utilizado quando não se conhece a distribuição dos dados. Para tanto, aproxima-se a FDP por um histograma.

Neste projeto o classificador por janelas de Parzen foi utilizado para se classificar os dados com base em duas das características que compõem o vetor de características. Foram escolhidas para esse propósito o 10º descritor de Fourier e o 8º descritor de Fourier da curvatura.

A figura 3.1 mostra o “histograma” gerado pelo método de janelas de Parzen para os dois atributos escolhidos considerando os vetores de características de todas as classes.

A matriz de confusão para o classificador por janelas de Parzen é mostrada na tabela 3.5. O tamanho da janela adotado foi de 0,2 sendo que o histograma foi calculado para um intervalo de valores entre -10 e 10 nas duas dimensões. Notar que os atributos foram estandarizados através da função **zscore** do Matlab.

Tabela 3.5. Matriz de confusão para o classificador por janelas de Parzen.

| | | **Classe Predita** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 2 | 0 | 0 | 1 | 3 | 4 |
| | **bg** | 0 | 8 | 2 | 0 | 0 | 0 |
| | **cg** | 0 | 2 | 8 | 0 | 0 | 0 |
| | **lg** | 1 | 0 | 2 | 0 | 7 | 0 |
| | **me** | 1 | 0 | 0 | 4 | 4 | 1 |
| | **pe** | 1 | 0 | 1 | 0 | 4 | 6 |

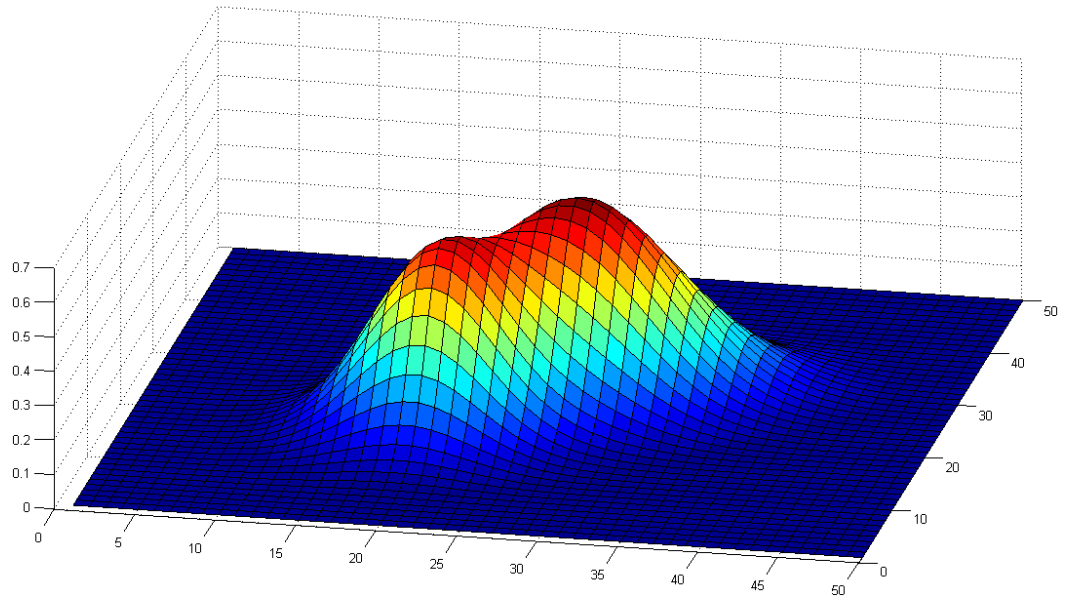


Figura 3.1. Histograma gerado pelo método de janelas de Parzen para os dois atributos escolhidos.

# 4. Parte 3: Classificadores

Esta parte do projeto teve como objetivo realizar a classificação das imagens utilizando o classificador k-vizinhos e redes neurais. Para tanto, foi utilizado o software Weka (Hall, et al. 2009) que faz uso de um arquivo no formato ARFF (*Attribute-Relation File Format*) para descrever a base de dados. O formato ARFF consiste em um arquivo de texto ASCII onde linha descreve o conjunto de atributos (no caso o vetor de características com os descritores de forma) e classe de uma imagem.

Para poder utilizar as funções de extração desenvolvidas em Matlab na parte 1 do projeto, foi implementado um script (**generateLeavesArffFile.m)** que extrai e salva os vetores de características de cada imagem no formato ARFF.

A seguir é feito um breve resumo dos dois classificadores utilizados e são mostrados os resultados obtidos na classificação (matrizes de confusão e as medidas derivadas).

## 4.1 Classificador k-vizinhos

O classificador k-NN ou k-vizinhos-mais-próximos (*k-nearest-neighbors*) é um exemplo algoritmo de aprendizado baseado em instâncias. Essa classe de algoritmos de aprendizado tem como característica utilizar instâncias de treinamento para realizar predições sem manter um modelo de classificação derivado dos dados (Tan, Steinbach and Kumar 2005).

Para realizar a classificação de um objeto, o k-NN procura pelos k exemplos da base de treinamento mais similares (vizinhos mais próximos) para verificar qual é a classe mais freqüente. É atribuído então, ao objeto a ser classificado, o rótulo da classe encontrada. No classificador k-NN cada objeto é representado como um ponto em um espaço n-dimensional, onde n é o número de atributos de cada instância. A similaridade entre dois objetos pode ser quantificada por meio de funções de distância tais como a euclidiana.

Para se obter um desempenho satisfatório para o classificador k-NN é importante que se escolha um valor apropriado para o parâmetro k. Em geral, para valores de k muito pequenos têm-se uma alta susceptibilidade a ruídos. Em contrapartida, autovalores de k tendem a privilegiar a classe majoritária e reduzir a flexibilidade da fronteira de decisão. Neste projeto, foi utilizado um procedimento em que o valor de k foi variado de 1 até 5 para encontrar o valor que, por meio de uma validação cruzada com dez partições, resultasse em uma maior acurácia na classificação. O melhor valor encontrado foi k = 2. A tabela 4.1 mostra a matriz de confusão obtida para o classificador k-NN.

## 4.2 Redes Neurais

Uma rede neural artificial é um modelo computacional inspirado pela estrutura funcional das redes neurais biológicas. Uma rede neural consiste em um grupo de neurônios artificiais interconectados.

Para este projeto foi utilizada a rede neural MLP (*multilayer perceptron*). Essa reder consiste em múltiplas camadas (*layers*) de nós que podem ser vistos como um grafo direcionado. Cada nó deste grafo corresponde a um neurônio que possui uma função de ativação não linear. O processo de aprendizado deste tipo de rede MLP, denominado de *backpropagation*, consiste em atualizar os pesos das conexões da rede (as arestas do grafo) com base no erro existente entre o rótulo

verdadeiro e o predito para um exemplo já rotulado. A tabela 4.2 mostra a matriz de confusão obtida para a classificação realizada utilizando a rede neural MLP.

Tabela 4.1. Matriz de confusão para o classificador k-vizinhos.

| | | **Classe Predita** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 10 | 0 | 0 | 0 | 0 | 0 |
| | **bg** | 0 | 8 | 2 | 0 | 0 | 0 |
| | **cg** | 0 | 1 | 8 | 0 | 0 | 1 |
| | **lg** | 0 | 1 | 0 | 9 | 0 | 0 |
| | **me** | 2 | 0 | 0 | 0 | 6 | 2 |
| | **pe** | 0 | 0 | 1 | 0 | 3 | 8 |

Tabela 4.2. Matriz de confusão para a rede neural MLP.

| | | **Classe Predita** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 10 | 0 | 0 | 0 | 0 | 0 |
| | **bg** | 0 | 9 | 1 | 0 | 0 | 0 |
| | **cg** | 0 | 3 | 7 | 0 | 0 | 0 |
| | **lg** | 0 | 0 | 0 | 10 | 0 | 0 |
| | **me** | 1 | 0 | 0 | 0 | 7 | 2 |
| | **pe** | 0 | 0 | 1 | 0 | 1 | 10 |

## 4.3 Comparação do desempenho da rede neural MPL e do classificador k-vizinhos

Na figura 4.1 é mostrada uma comparação do desempenho da rede neural MPL e do classificador k-vizinhos para classificação das imagens. Para realizar a comparação foram adotadas quatro métricas derivadas das matrizes de confusão: a acurácia, a precisão, a medida-F e a área sob a curva ROC. Pode-se perceber que a rede neural MPL foi a que apresentou o melhor desempenho.

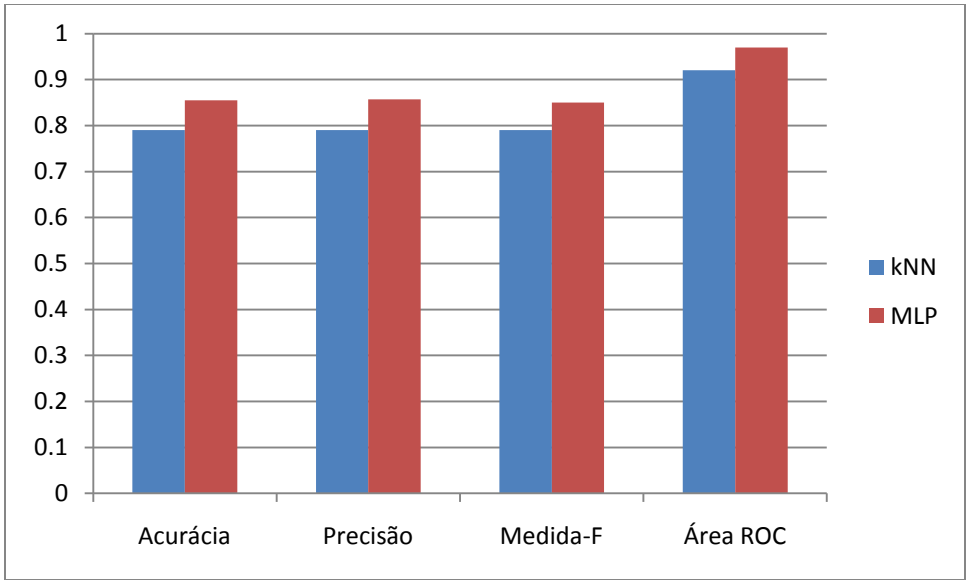

Figura 4.1. Comparação entre o desempenho do classificador k-vizinhos e a rede neural MPL.

# 5. Parte 4: Seleção de Atributos

## 5.1 Parte A: Seleção

Uma vez que muitos dos descritores de forma extraídos de uma certa imagem podem ser ou irrelevantes ou redundantes entre si, é possível realizar a seleção de um subconjunto de atributos sem que haja, necessariamente, perda de informação e reduzindo, desta maneira, o problema da maldição da dimensionalidade. São três as abordagens para se realizar a seleção de atributos. A primeira delas é a embarcada (*embedded*), na qual o processo de seleção de atributos ocorre como parte do algoritmo de classificação. A segunda consiste em utilizar filtros que selecionam os atributos por uma metodologia independente do algoritmo de classificação. A última abordagem, denominada *wrapper*, consiste em utilizar o próprio classificador para encontrar um subconjunto de atributos (neste caso descritores de forma).

Neste projeto foi adotado o algoritmo CFS (*Correlation-based Feature Selection*) (M. A. Hall 2000) para realizar a seleção de atributos sobre os descritores extraídos da base de imagens de folhas. O CFS é um algoritmo de seleção de atributos do tipo filtro que faz de uma métrica de avaliação de subconjuntos de atributos que leva em conta tanto a capacidade de um atributo em predizer a classe de uma instância quanto o nível de intercorrelação existentes entre os atributos do subconjunto.

A implementação do CFS utilizada neste projeto é a oferecida pelo software Weka. Para realizar a busca no espaço de atributos foi utilizado um algoritmo de busca gulosa pelo espaço de atributos. Uma vez que uma busca exaustiva pelo espaço de atributos é excessivamente custosa computacionalmente, foi definido como critério de parada para a realização de no máximo 30 níveis de *backtracking*. Ou seja, se durante a busca forem encontrados 30 subconjuntos seguidos sem que haja uma melhora na métrica do CFS, a busca é terminada. A tabela 5.1 mostra os três atributos mais importantes encontrados.

Tabela 5.1. Os três atributos mais importantes encontrados pelo algoritmo de seleção CFS utilizando uma busca gulosa com 30 níveis de *backtracking*.

| **Atributo** | **Descrição** |
|---|---|
| center_of_mass_x | Coordenada x do centro de massa. |
| fourier10 | 10º descritor de Fourier. |
| curve_fourier8 | 8º descritor de Fourier da curvatura. |

As matrizes de confusão para os classificadores k-vizinhos e a rede neural com MLP com a seleção de atributos são mostradas nas tabelas 5.2 e 5.3. Na figura 5.1 é feita uma comparação entre os resultados obtidos com e sem seleção de atributos. É importante observar que os resultados obtidos após a seleção de três atributos é bastante próximo dos resultados obtidos com o vetor de características completo, com 49 atributos.

Tabela 5.2. Matriz de confusão para o classificador k-vizinhos com seleção de atributos.

| | | Classe Predita | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 10 | 0 | 0 | 0 | 0 | 0 |
| | **bg** | 0 | 8 | 2 | 0 | 0 | 0 |
| | **cg** | 0 | 4 | 6 | 0 | 0 | 0 |
| | **lg** | 0 | 2 | 0 | 8 | 0 | 0 |
| | **me** | 3 | 0 | 0 | 0 | 6 | 1 |
| | **pe** | 0 | 0 | 1 | 0 | 2 | 9 |

Tabela 5.3. Matriz de confusão para a rede neural MLP com seleção de atributos.

| | | Classe Predita | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 9 | 0 | 0 | 0 | 1 | 0 |
| | **bg** | 0 | 9 | 1 | 0 | 0 | 0 |
| | **cg** | 0 | 2 | 8 | 0 | 0 | 0 |
| | **lg** | 0 | 1 | 0 | 9 | 0 | 0 |
| | **me** | 3 | 0 | 0 | 0 | 6 | 1 |
| | **pe** | 0 | 0 | 1 | 0 | 2 | 9 |

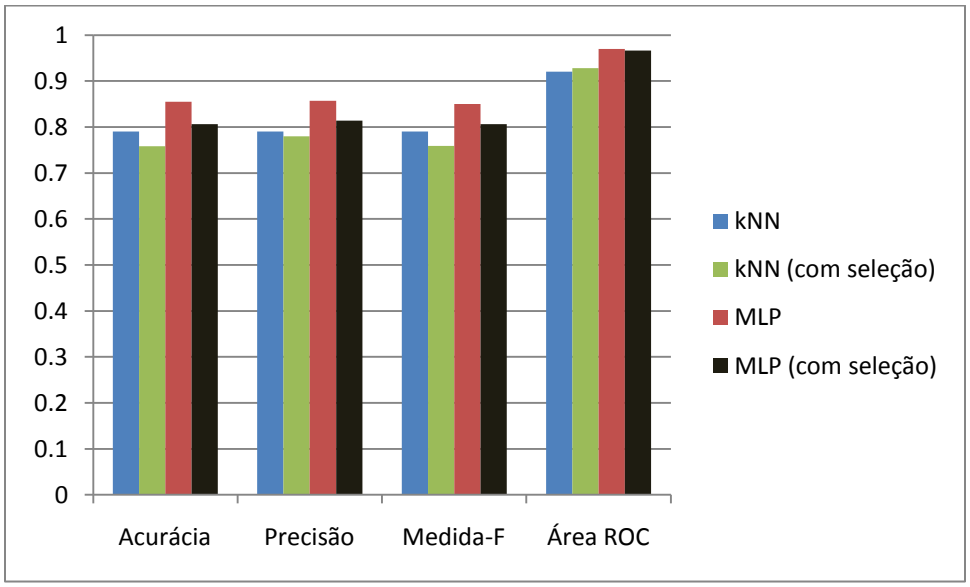


Figura 5.1. Comparação do desempenho dos classificadores k-vizinhos e MLP com e sem seleção de atributos. O vetor original possui 49 atributos e após a seleção foram utilizados somente três atributos.

## 5.2 Parte B: Projeção PCA

A função **pcaProjection.m** toma como entrada a matriz NxC de vetores de características, onde N corresponde ao número de vetores de características e C ao número de componentes do vetores. A função retorna uma matriz com N linhas e duas colunas. As colunas correspondem aos componentes dos vetores de características projetados utilizando o método PCA (implementado pela função **princomp** do Matlab). A tabela 5.4 mostra a matriz de confusão obtida para classificação por janelas de Parzen com os vetores projetados. Assim como foi feito na parte 2 deste projeto, foi adotada uma janela de tamanho 0,2 e o histograma foi calculado para um intervalo de valores entre -10 e 10. A acurácia obtida para classificação com os atributos projetados por PCA foi de 69%, que corresponde a um desempenho superior ao obtido na parte dois com dois atributos, que apresentou uma acurácia de 54%.

Tabela 5.4. Matriz de confusão para o classificador por janelas de Parzen para atributos projetados utilizando o método PCA.

| | | **Classe Predita** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **ag** | **bg** | **cg** | **lg** | **me** | **pe** |
| **Classe Verdadeira** | **ag** | 9 | 0 | 0 | 0 | 0 | 0 |
| | **bg** | 0 | 0 | 10 | 0 | 0 | 0 |
| | **cg** | 0 | 0 | 10 | 0 | 0 | 0 |
| | **lg** | 0 | 0 | 0 | 0 | 2 | 8 |
| | **me** | 0 | 1 | 1 | 8 | 0 | 0 |
| | **pe** | 1 | 0 | 2 | 9 | 0 | 0 |

# 6. Parte 5: Agrupamento (*Clustering*)

A implementação do algoritmo de agrupamento é realizada no arquivo **kMeansPI.m**. A função toma como entrada um *cell array* onde cada linha corresponde ao vetor de características e a classe de uma imagem. A saída da função é uma matriz de n linhas por duas colunas, onde n é o número de imagens na base, a primeira coluna corresponde à classe da imagem e a segunda coluna corresponde ao agrupamento ao qual a imagem foi atribuída.

O valor de k (total de agrupamentos) adotado para execução do algoritmo foi o igual ao total de classes da base de imagens, ou seja, seis. As sementes iniciais para execução do algoritmo são seis instâncias diferentes escolhidas aleatoriamente. O algoritmo é finalizado quando nenhuma instância tem sua classe atualizada durante uma iteração.

Foram utilizadas duas métricas parar realizar a avaliação dos agrupamentos obtidos. A primeira delas é a entropia dada pela fórmula:

$$E = -\sum_{j=1}^{L} p_{ij}\, log_2(p_{ij}) \tag{6.1}$$

onde $L$ é o número de classes e $p_{ij}$ corresponde a probabilidade de um membro do agrupamento $i$ pertencer à classe $j$. Quanto menor o valor de entropia de um agrupamento, melhor o resultado do algoritmo de agrupamento. A segunda métrica utilizada foi a pureza (*purity)* dada por:

$$P = \max_{j} p_{ij} \tag{6.2}$$

A tabela 6.1 mostra os resultados obtidos para uma das execuções do algoritmo k-means. As médias dos valores de entropia e pureza (linha na parte de baixo da tabela) são ponderadas pelo número de elementos no agrupamento. A tabela 6.2 mostra as métricas de entropia e pureza obtidas para três realizações (escolhas diferentes de sementes iniciais) do algoritmo *k-means*.

Tabela 6.1. Resultados obtidos para o algoritmo de agrupamento k-means utilizando k=6.

| | Classe | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Grupo** | **ag** | **bg** | **cg** | **lg** | **me** | **pe** | **Entropia** | **Pureza** |
| 1 | 0 | 0 | 0 | 8 | 0 | 0 | 0,00 | 1,00 |
| 2 | 4 | 0 | 0 | 0 | 5 | 4 | 1,58 | 0,38 |
| 3 | 6 | 0 | 0 | 0 | 5 | 7 | 1,57 | 0,39 |
| 4 | 0 | 2 | 3 | 2 | 0 | 1 | 1,91 | 0,37 |
| 5 | 0 | 0 | 5 | 0 | 0 | 0 | 0,00 | 1,00 |
| 6 | 0 | 8 | 2 | 0 | 0 | 0 | 0,72 | 0,80 |
| **Total/Média** | 10 | 10 | 10 | 10 | 10 | 12 | 1,15 | 0,58 |

Tabela 6.2. Resultados obtidos para execução do *k-means* para diversas realizações.

| | **Experimento 1** | | | **Experimento 2** | | | **Experimento 3** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Grupo** | **Tam.** | **E** | **P** | **Tam.** | **E** | **P** | **Tam.** | **E** | **P** |
| 1 | 8 | 0,00 | 1,00 | 10 | 1,21 | 0,56 | 5 | 0,00 | 1,00 |
| 2 | 13 | 1,58 | 0,38 | 6 | 0,00 | 1,00 | 9 | 0,28 | 0,85 |
| 3 | 18 | 1,57 | 0,39 | 8 | 0,82 | 0,74 | 20 | 2,18 | 0,32 |
| 4 | 8 | 1,91 | 0,37 | 17 | 1,71 | 0,39 | 12 | 1,24 | 0,35 |
| 5 | 5 | 0,00 | 1,00 | 6 | 0,92 | 0,85 | 6 | 0,00 | 1,00 |
| 6 | 10 | 0,72 | 0,80 | 7 | 0,00 | 1,00 | 10 | 0,33 | 0,78 |
| **Total/ Média** | 62 | 1,15 | 0,58 | 62 | 0,86 | 0,58 | 62 | 1,04 | 0,59 |

# Bibliografia

Hall, M. A. "Correlation-based feature selection for discrete and numeric class machine learning." *Machine Learning Internation Workshop.* 2000. 359-366.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. "The WEKA Data Mining Software: An Update." *SIGKDD Explorations*, 2009.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.