Trabalho Prático 1

Identificação de Imagens por Visão Computacional

Programação de Baixo Nível

04/2018

1 Resumo

O objetivo deste trabalho é a criação de um programa C de **Análise de Imagens**. Um dos objetivos da Análise de Imagens é identificar características de uma imagem de maneira que seja possível, através de um processo de Aprendizado de Máquina, saber se esta imagem é, por exemplo, de uma maçã ou de uma pera, ou se um piso de um grama ou de areia. Para que seja possível executar o Aprendizado de Máquina, é preciso gerar um conjunto de **atributos**, realizar o treino de um classificador e a seguir testar este classificador.

Desta forma, o processo de identificação de imagens por Aprendizado de Máquina pode ser resumido na Figura 1.

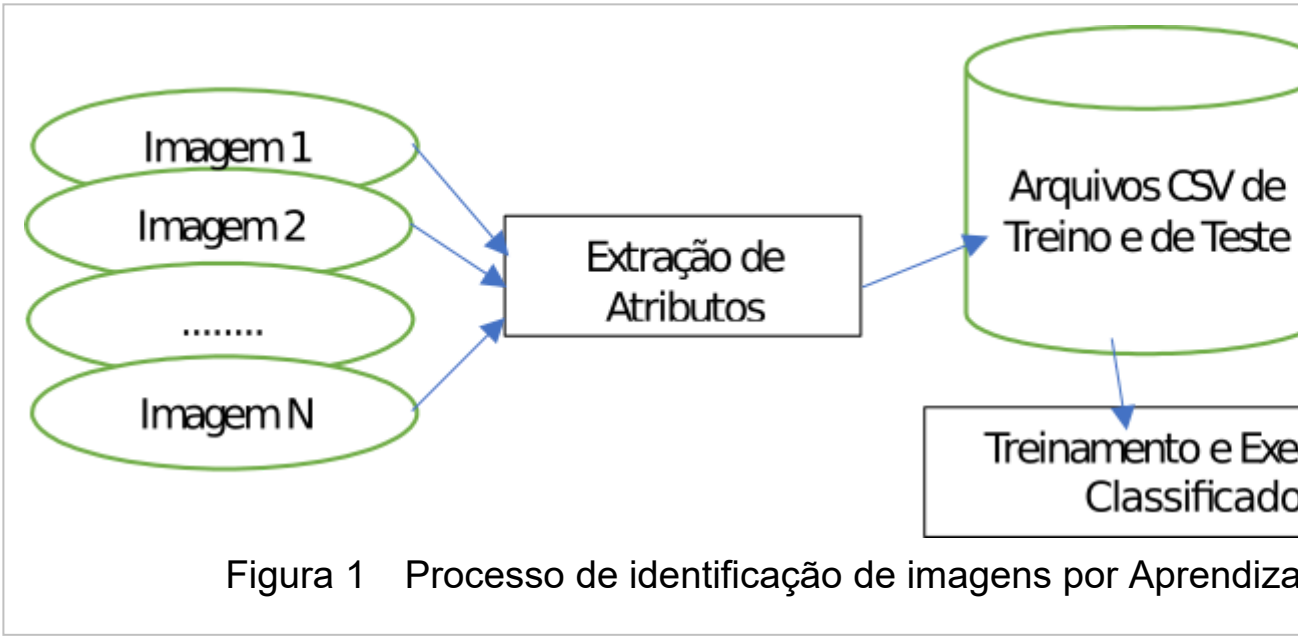


Figura 1 Processo de identificação de imagens por Aprendiza

Neste trabalho, você deverá realizar a parte referente à Extração de Atributos.

O programa C que você irá criar, deverá realizar a classificação de um conjunto de imagens, executando os seguintes passos:

- Carregar uma imagem;
- Gerar um conjunto de atributos a partir dos pixels da imagem;
- Gravar os atributos de cada imagem em uma linha de um arquivo CSV;
- Repetir as etapas anteriores para todas as imagens disponíveis, gravando os atributos no mesmo CSV.

Depois da execução destes passos, você deverá executar o script Python que realiza a classificação.

Cada uma destas etapas é detalhada a seguir.

2 Carregar uma imagem

Para realizar a leitura da imagem utilizaremos uma biblioteca pronta denominada *SOIL*, já integrada no projeto fornecido. O programa exemplo contido no projeto abre um arquivo contendo uma imagem e preenche um vetor com dos dados da mesma.

Uma imagem é essencialmente uma matriz de pontos em memória (*pixels*), onde cada cor geralmente é definida por três componentes: vermelho (R),verde (G) e azul (B). Cada uma dessas componentes é usualmente codificada em um byte, o que produz 3 bytes por pixel (24 bits) – ou seja, 16 milhões de possíveis cores (2^{24}). Em outras palavras, as intensidades (R, G, B) variam de 0 a 255, onde 0 é escuro e 255 é claro.

Veja na Figura 2 como diversas cores são representadas nesse formato - cada cor está expressa pelas componentes RGB em hexadecimal.

FFFFFF	000000	333333	666666	999999	CCCCCC	CCCC99	9999CC
660000	663300	996633	003300	003333	003399	000066	330066
990000	993300	CC9900	006600	336666	0033FF	000099	660099
CC0000	CC3300	FFCC00	009900	006666	0066FF	0000CC	663399
FF0000	FF3300	FFFF00	00CC00	009999	0099FF	0000FF	9900CC
CC3333	FF6600	FFFF33	00FF00	00CCCC	00CCFF	3366FF	9933FF
FF6666	FF6633	FFFF66	66FF66	66CCCC	00FFFF	3399FF	9966FF
FF9999	FF9966	FFFF99	99FF99	66FFCC	99FFFF	66CCFF	9999FF
FFCCCC	FFCC99	FFFFCC	CCFFCC	99FFCC	CCFFFF	99CCFF	CCCCFF

Figura 2 Exemplo de cores codificadas em RGB

Para simplificar, nós trabalharemos com imagens em tons de cinza, isto é, imagens onde R == G == B. Com isso, a representação interna também muda: ao invés de 3 bytes, só é necessário um byte por pixel,

que representa a intensidade (ou tom) de cinza – onde 0 corresponde ao preto e 255 corresponde ao branco.

3 Gerar um conjunto de atributos a partir dos pixels da imagem

Atributos são valores que buscam descrever características de uma imagem. Para tanto são analisadas as cores dos pixels, ou intensidades da imagem e coletadas estatísticas destes dados.

Neste texto, assume-se que a imagem está armazenada em forma de uma matriz e que as cores são intensidades, também conhecidos como tons de cinza. Desta forma, a análise das cores dos pixels deve ser feita analisando os dados de uma matriz de inteiros.

Nas seções seguir são apresentados os atributos que deverão ser gerados para o processo de análise.

3.1 Histogramas de níveis de cinza

Um histograma de níveis de cinza descreve uma imagem como um vetor de inteiros que armazena o número de vezes que um determinado tom de cinza aparece em uma imagem. Desta forma, se a imagem tem N tons de cinza distintos, então este vetor terá N elementos.

A Figura 3 mostra um exemplo de criação de um histograma. A matriz da esquerda representa uma imagem que possui tons de cinza no intervalo [0..10], o vetor da direita, apresenta o histograma da imagem e o gráfico na parte inferior apresenta o traçado do histograma.

Imagem

1	1	1	4	4	4
1	1	1	9	9	9
1	1	4	9	9	9
4	4	0	0	1	1
5	5	0	11	1	1
10	10	10	1	1	1

Histograma de Níveis de Cinza

Tom de Cinza	0	1	2	3	4	5	6	7	8	
Quantidade	3	15	0	0	6	2	0	0	0	

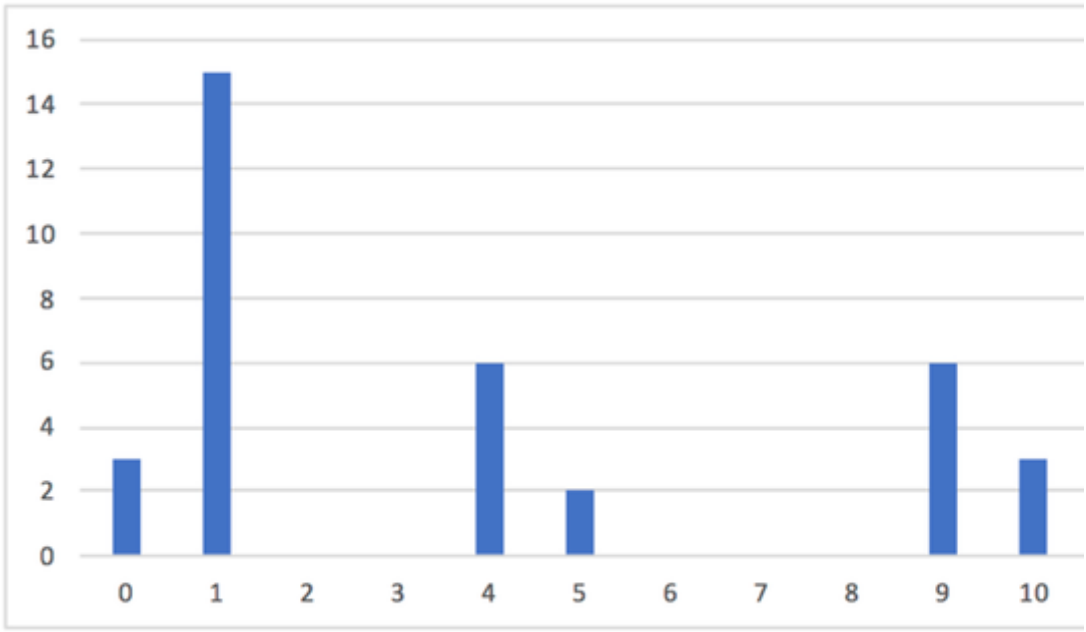


Figura 3 Geração de um histograma de níveis de cinza

Como muitas vezes o histograma tem uma quantidade muito grande de entradas, costuma-se simplificá-lo, dividindo o vetor em regiões. No exemplo da Figura 4, observa-se a divisão do histograma em 3 regiões.

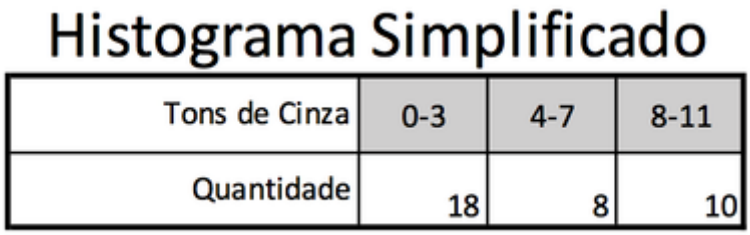


Figura 4 Simplificação de um histograma de níveis de cinza

A partir destes dados os valores 18, 8 e 10 são considerados como atributos da imagem.

Além destes atributos, é possível extrair medidas estatísticas de um histograma, ou mesmo diretamente da imagem e usá-las como atributos. Exemplos destas medidas são média, desvio padrão, curtose e mediana. A Tabela 1 mostra os valores destes atributos tanto para a imagem quanto para o histograma da Figura 3.

Tabela 1 Exemplos de descritores estatísticos

Atributo	Da imagem	Do histograma
Média	4	3
Desvio padrão	3,749285646	4,390071443
Curtose	1,211455812	5,01128515
Mediana	2,5	1,5

3.2 Matrizes de Co-ocorrência de Níveis de Cinza

Outro tipo de atributo muito usado para descrever características de uma imagem são as **Matrizes de Co-ocorrência de Níveis de Cinza**, o inglês *Gray Level Coocurrence Matrices* (GLCMs).

As GLCMs servem para descrever **correlações** entre pixels vizinhos de uma imagem. A ideia desta correlação é determinar a **frequência** com que duas intensidades i e j ocorrem em pontos “vizinhos” de uma imagem.

No caso, o conceito de *vizinho* é definido por uma distância em colunas (*dx*), e em linhas (*dy*) entre dois pontos da imagem. Desta forma, para cada distância definida por um vetor D=(dx,dy) é possível criar uma matriz de N linhas e N colunas, sendo N o número de intensidades diferentes na imagem. Esta matriz recebe o nome de **Matriz de Co-Ocorrência**, ou **MCO**.

A Figura 5 mostra o exemplo de uma imagem e três GLCMs, para as distâncias D(1,0), D(0,1) e D(1,-1). Como a imagem tem 3 tons de cinza distintos, a MCOs geradas têm 3 linhas e 3 colunas cada.

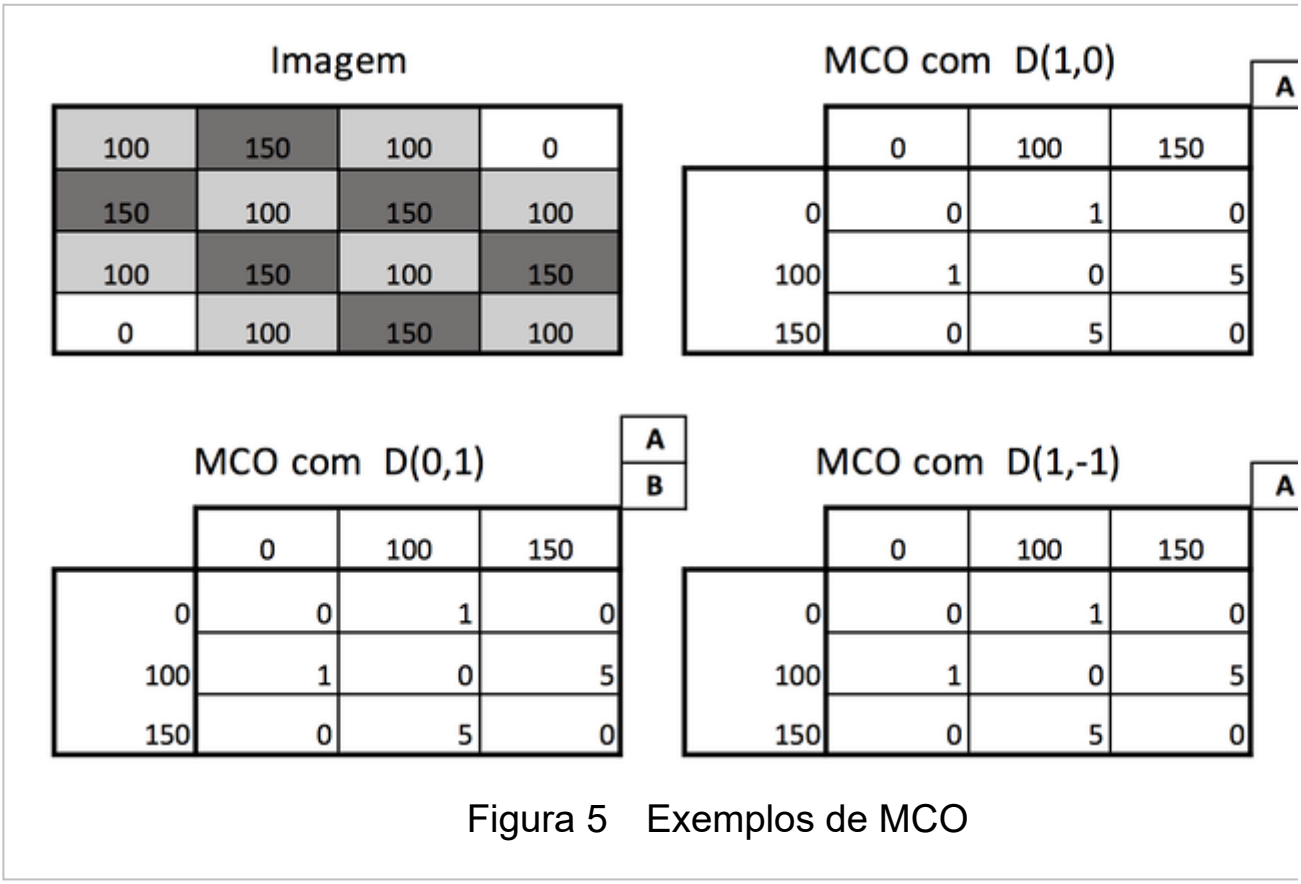
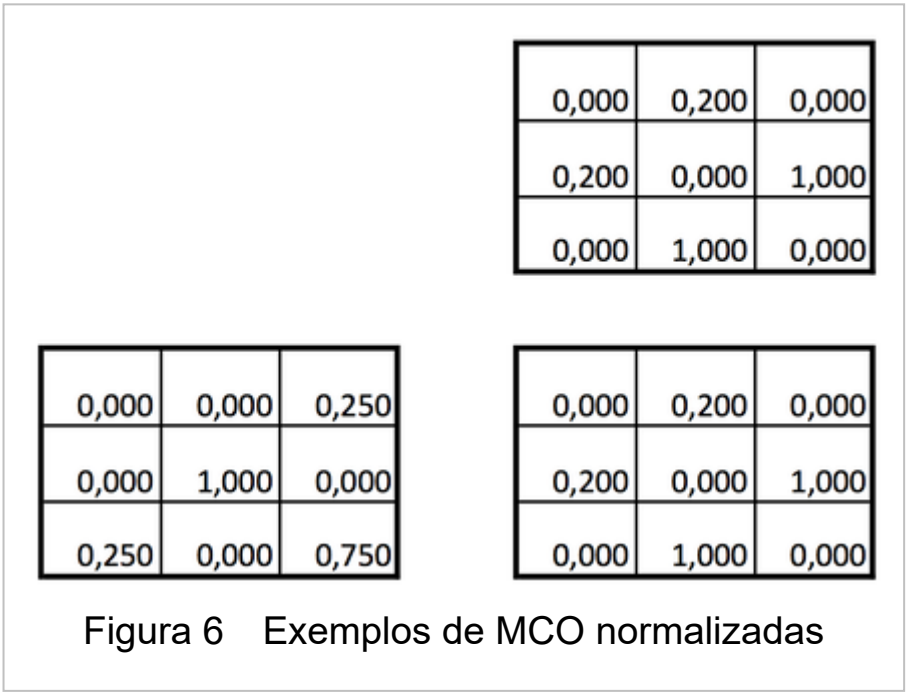


Figura 5 Exemplos de MCO

Para transformar a MCO em uma matriz com valores entre 0 e 1, cada elemento é dividido pelo maior de todos os valores existente na MCO, gerando a chamada *MCO Normalizada*. A Figura 6 mostra o resultado da normalização das matrizes da Figura 5.



0,000	0,200	0,000
0,200	0,000	1,000
0,000	1,000	0,000

0,000	0,000	0,250
0,000	1,000	0,000
0,250	0,000	0,750

0,000	0,200	0,000
0,200	0,000	1,000
0,000	1,000	0,000

Figura 6 Exemplos de MCO normalizadas

A partir de uma MCO normalizada são gerados descritores estatísticos, conforme as seguintes fórmulas:

- Energia:

$$E = \sum_{i,j=0}^{levels-1} P_{i,j}^2$$

- Entropia:

$$\sqrt{E}$$

- Contraste:

$$\sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2$$

- Variância:

$$\sum_{i,j=0}^{levels-1} P_{i,j}(i-j)$$

- Homogeneidade:

$$\sum_{i,j=0}^{levels-1} P_{i,j} \left[\frac{P_{i,j}}{1+(i-j)^2} \right]$$

Nestas fórmulas, a expressão $P_{i,j}$ é o valor da MCO na posição [i,j].

3.3 Preparação das imagens para o cálculo das MCOs

Note que em uma imagem real, como as fornecidas para este trabalho, os tons de cinza podem variar no intervalo entre 0 e 255. Entretanto, muitos destes tons não ocorrem de fato na imagem.

Com isto, as MCOs geradas terão muitas células contendo o valor 0, o que irá degradar a capacidade descritiva das medidas estatísticas.

Para solucionar este problema, pode-se reduzir a quantidade de tons de cinza das imagens simplesmente dividindo o valor do tom de cinza por uma constante, como no exemplo a seguir. Neste exemplo, a quantidade de tons é reduzida de 255 para **QTD_TONS**, o que irá gerar uma MCO de **QTD_TONS** linhas e **QTD_TONS** colunas.

```
for(x=0; x<Largura x++)
{
  for(y=0; y< Altura; y++)
  {
     Cor[x][y] = Cor[x][y]/(255/QTD_TONS);
  }
}
```

Listagem 1 Algoritmo para redução da quantidade de tons de uma imagem

4 Gravar os atributos de cada imagem em um arquivo CSV

Após a geração dos atributos, estes deverão ser gravados em dois arquivos distintos. Um de **Treino** e outro de **Teste**. O arquivo de treino permite treinar

um classificador e o segundo permite classificar um conjunto de imagens.

A seção 5 apresenta o script Python que realiza os processos de Treino e Teste.

4.1 Geração do CSV de Treino

A partir da geração dos atributos de uma imagem, estes devem ser gravados em um arquivo no formato *CSV* (*Comma-Separated Values*), para possibilitar o treinamento do classificador.

Neste arquivo, a primeira linha descreve os nomes dos atributos. As demais linhas descrevem os atributos separados por vírgula. Estes dados são formados pelo nome da imagem, seguido pela sequência de atributos separados por vírgula e finalizado pela classe a que pertence à imagem.

Tomando como exemplo os dados descritos na Tabela 2, o CSV gerado deve ser como descreve a Listagem 2.

Tabela 2 Dados de Imagens

Nome	Média	Desvio	Curtose	Mediana	QTD1	QTD2	QTD3
Grass1	4	3.749	-1.211	2.5	18	8	10
Grass2	3	2.564	1.003	3	8	3	5
Grass3	9	2.018	0.876	1.5	10	4	10
Sand1	2	1.5	2.564	2.4	2	4	4
Sand2	3	1.5	1.104	1.5	1	10	7
Sand3	1	1.5	-1.876	0.5	1	7	10

```
NOME,Media,Desvio,Curtose,Mediana,QTD1,QTD2,QTD3,Classe
Grass1,4,3.749,-1.211,2.5,18,8,10,Grama
Grass2,3,2.564,1.003,3.0,8,3,5,Grama
Grass3,9,2.018,0.876,1.5,10,4,10,Grama
Sand1,2,1.5,2.564,2.4,2,4,4,Areia
Sand2,3,1.5,1.104,1.5,1,10,7,Areia
Sand3,1,1.5,-1.876,0.5,1,7,10,Areia
```

Listagem 2 Exemplo de arquivo CSV para TREINO

4.2 Geração do CSV de Teste

Após o treino, é possível testar o classificador com um segundo conjunto de imagens, da qual se tem o mesmo conjunto de atributos. Para tanto, um novo CSV deve ser gerado, conforme o exemplo da Listagem 3.

```
NOME,Media,Desvio,Curtose,Mediana,QTD1,QTD2,QTD3
Grass15,2,3.44,-1.211,1.5,8,18,0,Grama
Sand18,1,3.4,0.003,5.3,2,5,4,Areia
```

Listagem 3 Exemplo de arquivo CSV para TESTE

5 Executar o script Python que realiza a classificação

A partir da existência dos arquivos de Treino e Teste, nos formatos descrito na seção anterior, é possível treinar um classificador com os dados e classificar novas imagens, com base no script Python apresentado na Listagem 4.

Para a execução, este script deve ser chamado recebendo os nomes dos arquivos de treino e de teste como parâmetros, da seguinte forma:

```
python Classificador.py Treino.csv Teste.csv
```

```
#!/usr/bin/env python
import sys
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

if len(sys.argv) != 3:
    print('Requeridos 2 argumentos: arquivo de treino e arquivo de teste
    sys.exit(1)

# Lê os arquivos
treino = pd.read_csv(sys.argv[1])
teste = pd.read_csv(sys.argv[2])
atributos = np.asarray(treino.columns)
if not np.array_equal(atributos, teste.columns):
    print('Os arquivos possuem atributos distintos')
    sys.exit(1)

# O primeiro argumento é o identificador
id = atributos[0]
# Os outros, exceto o último, são os preditivos
preditivos = atributos[1: - 1]
# E que o atributo alvo é o último
alvo = atributos[-1]
print('Atributo identificador:', id)
print('Atributos preditivos:', ', '.join(preditivos))
print('Atributo alvo:', alvo)
print('Numero de registros de treino:', len(treino))
print('Numero de registros de teste:', len(teste))

# Treino do classificador
clf = MLPClassifier(hidden_layer_sizes=[30, 10], max_iter=500)
clf.fit(treino[preditivos], treino[alvo])

# Teste
respostas = clf.predict(teste[preditivos])
for i, reg in teste.iterrows():
    identificador = reg[id]
    classe_real = reg[alvo]
    classe_predita = respostas[i]
    resposta = 'ACERTOU' if classe_real == classe_predita else 'ERROU'
    print('%s, classe real: %s, classe predita: %s. %s!' %
          (identificador, classe_real, classe_predita, resposta))

print('-------------------------------------')
acc = accuracy_score(teste[alvo], respostas, normalize=True) * 100
print('Acuracia: %0.2f%%' % acc)
```

Listagem 4 Exemplo de classificador em Python

6 Tarefas

Para a realização deste trabalho, devem ser executadas as seguintes tarefas:

- Criar um programa que gere, para cada imagem do dataset disponível abaixo, o conjunto de atributos descritos na Tabela 3;
- Dividir os dados gerados em dois arquivos. Um para treino e outro para teste;
- Rodar o classificador, fornecendo como entrada os arquivos gerados.

Tabela 3 Atributos a serem gerados e sua origem

Atributo	Origem
Média	Histograma da imagem
Mediana	Histograma da imagem
Desvio padrão	Histograma da imagem
Curtose	Histograma da imagem
Energia	GLCM da imagem
Entropia	GLCM da imagem
Contraste	GLCM da imagem
Variância	GLCM da imagem
Homogeneidade	GLCM da imagem

Links:

- Projeto exemplo para ler a imagem
- Dataset com as imagens
- Script Python para classificação
- Exemplos de arquivos de treino e teste (já prontos)

7 Avaliação

Leia com atenção os critérios de avaliação:

- Os trabalhos são **em duplas ou individuais**. A pasta do projeto deve ser compactada em um arquivo .zip e este deve ser submetido pelo *Moodle* até a data e hora especificadas.
- Não envie .rar, .7z, .tar.gz - apenas .zip.
- O código deve estar identado corretamente (o *Code::Blocks* **faz isso automaticamente**).
- A nota do trabalho depende da apresentação deste no laboratório, na data marcada. Trabalhos entregues mas não apresentados terão sua nota anulada automaticamente. Durante a apresentação será avaliado o domínio da resolução do problema, podendo inclusive ser possível invalidar o trabalho quando constatada a falta de conhecimento sobre o código implementado.
- **A cópia parcial ou completa do trabalho terá como conseqüência a atribuição de nota ZERO ao trabalho dos alunos envolvidos. A verificação de cópias é feita inclusive entre turmas.**
- **A cópia de código ou algoritmos existentes da Internet também não é permitida**. Se alguma idéia encontrada na rede for utilizada na implementação, sua descrição e referência deve constar no artigo.

Document generated by eLyXer 1.2.5 (2013-03-10) on 2018-04-04T15:27:34.844156