Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: BCC 326 Processamento de Imagens

Trabalho de Implementação

O reconhecimento de padrões é o estudo de como as máquinas observam seu entorno, aprendem a distinguir padrões de interesse e tomam decisões razoáveis sobre as categorias dos padrões, um padrão é uma descrição de um objeto. Um computador consegue reconhecer padrões, convertendo-os em sinais digitais e comparando-os com outros sinais já armazenados na memória.

Um sistema de reconhecimento geralmente compreende três componentes principais: pré-processamento, extração de características e classificação. Na etapa de pré-processamento, os dados de entrada são manipulados por uma variedade de métodos que fazem operações, tais como remoção de ruído, segmentação e melhoramento da qualidade dos mesmos. Na extração de características, o objetivo é representar os dados de entrada em termos de medidas quantificáveis que possam ser utilizados facilmente na etapa de classificação. O problema do reconhecimento de padrões é reconhecer padrões que sejam, em algum sentido, "os mesmos" apesar de ter experimentado uma variedade de transformações permitidas. Os padrões na vida real apresentam transformações geométricas lineares (rotação, escala e translação), deformações não lineares e variância de iluminação e *background*, etc. Este tipo de reconhecimento pode ser uma tarefa simples para os seres humanos e para os animais, mas converte-se em um grande problema se tentamos realizá-lo através de um computador. Os métodos tradicionais de reconhecimento de padrões carecem da habilidade para reconhecer o mesmo padrão com certo tipo de variância.

1. Para cada imagem de números fornecida, além da original, gere mais 19 imagens com diferentes graus de rotações. Use o comando *imrotate()*.

2. Cada imagem rotaciona deve ser escalada em sete escalas diferentes: 0.5, 0.75, 1, 1.25, 1.5, 1.75 e 2. Use a função *imresize()*

3. Em cada imagem escalada, inserido ruido tipo sal e pimenta usando a função *imnoise()*, usar os seguintes valores de ruído : 0, 0.01, 0.02, 0.03 e 0.04.

4. Depois, extrair os momentos de Hu para cada imagem (código disponível no moodle).

5. Dividir a base em dois conjuntos, a primeira para treino e a segunda para teste. Use a seguinte porcentagem para cada grupo: treino (75%), e teste (25%). Utilize a função `crossvalind()` para executar essa tarefa. Caso a função não exista, use a o código *amostra.m* que realiza o mesmo processo. A continuação, a base de dados é dividida em 2 (dois) conjuntos.

```
% para Matlab
[train, test] = crossvalind('HoldOut', etiqueta, 0.5);
```

```
% para Octave
[train, test] = amostra(etiqueta, 0.5);
```

Divide a base em dois conjuntos, cada um corresponde ao 50% da base. A divisão da base é baseado no vetor de etiquetas (classes). Desta forma, a função consegue

saber quantas amostras existem por cada classe. Por exemplo, se dentro do vetor de *etiqueta* tem 3 classes com a seguinte quantidade de amostras por grupo: 80 para classe 1, 50 para classe 2 e 100 para classe 3, a função *crossvalind/amostra()* escolhe 50% de amostras de cada classe, ou seja, 40 da classe 1, 25 da classe 2 e 50 da classe 3. Retorna dentro do vetor *train* os índices das amostras usadas para treino e no vetor *test* os índices do conjunto de teste.

6. Utilize a base de treino para "aprender" os diferentes padrões. Use os comandos *fitcdiscr()* e *predict()* se estiver usando Matlab e os comandos *train_sc()* e *test_sc()* se estiver usando Octave.

```
% para Matlab
model = fitcdiscr(train_data, train_label);
pred = predict(model, test_data)
```

```
% para Octave
model = train_sc(train_data, train_label);
pred = test_sc(model, test_data)
```

onde `test_data` é o conjunto de teste, `train_data` o conjunto de treino e `train_label` as etiquetas do conjunto de treinamento. A função *fitcdiscr()/train_sc()* retorna o modelo treinado. A função *predict()/test_sc()* retorna como resultado as etiquetas do conjunto de teste.

Encontre as amostras de treino e suas respectivas etiquetas usando o vetor *train* retornado pela função *crossvalind()* ou *amostra()*.

```
train_data = data(train, :);
train_label = etiqueta(train);
```

onde *data* é a matriz de características, com os momentos de Hu de cada uma das imagens da base. Faça o mesmo processo para o conjunto de teste.

7. Calcule a taxa de acerto para cada tipo de digito. Além disso, gere uma matriz de confusão. Veja o uso da função *confusionmat()*

8. Testar o seu classificador identificando os caracteres na imagem *numeros2.jpg*. Use o modelo treinado com a função *fitcdiscr()/train_sc()*. Use a função *BoundingBoxPatches()* fornecida no Moodle para segmentar os caracteres da imagem *numeros2.jpg*. Depois, calcular os momentos de Hu de cada subimagem recortada. Logo, o vetor descritor gerado passar como parâmetro na função *predict()/test_sc()* para que retorne a etiqueta identificada.

O seguinte código permite realizar a leitura das imagens, com uma determinada extensão (".png", ".jpg", etc), que se encontram dentro de um diretório.

```
endereco = 'D:\pdi\images\';
ext = 'png'; % modificar segundo a extensao das imagens
% procura as imagens com extensao png dentro do diretorio images
arquivos = dir( fullfile(endereco, ['*.' ext]) ); %ler as imagens
quant_img = length(arquivos);
for i = 1:quant_img
    img = imread(fullfile(endereco, arquivos(i,1).name));
    ...
end
```