UNIVERSIDADE FEDERAL DE MINAS GERAIS

Tópicos em Inteligência Artificial - Sistemas Nebulosos

Exercício Computacional 3 - Fuzzy c-means

Luiz Henrique Silva Lelis - 2014034847

30/05/2017

Sumário

Lista de Figuras

1 Introdução

Na teoria clássica um dado elemento do universo (domínio) pertence ou não pertence ao referido conjunto. Na teoria dos conjuntos "fuzzy" (ou nebulosos) existe um grau de pertinência de cada elemento a um determinado conjunto. Esse grau de pertinência é determinado através de uma função denominada Função de Pertinência. A lógica fuzzy estende a lógica booleana, ao invés de permitir só dois valores possíveis (verdadeiro ou falso), ela permite uma gama infinita de valores.

A análise de cluster nos algoritmos c-means e k-means envolve a atribuição de pontos de dados a clusters (também chamados de grupos ou classes). Essa atribuição ocorre de modo que itens na mesma classe ou cluster sejam tão semelhantes quanto possível, enquanto itens pertencentes a classes diferentes são tão diferentes como possível. Os clusters são identificados através de medidas de similaridade. Essas medidas de similaridade incluem distância, conectividade e intensidade. Podem ser escolhidas diferentes medidas de similaridade com base nos dados ou na aplicação.

Os algoritmos k-means e c-means compartilham da mesma lógica e dos mesmos procedimentos em geral. C-means se diferencia do primeiro por estender a o algoritmo aos conceitos Fuzzy. (o que o torna mais robusto). Ou seja, ele estende a noção de associatividade de cada padrão aos grupos utilizando o conceito de função de pertinencia (cada padrão possui um grau de pertinência para cada grupo). O presente relatório propõe a implementação de um algoritmo fuzzy c-means que utiliza a distância euclediana como medida de similaridade.

O algoritmo fuzzy c-means pode ser dividido em quatro etapas de execução: (1) atribuir os valores iniciais, (2)calcular o centro de cada grupo utilizando a matriz de partição, (3) associar cada padrão (elemento) ao centro mais próximo (atualiza a matriz de partição), por fim, (4) pára caso o critério de convergência seja atingido, caso contrário volta para o passo 2. O algoritmo apresentado no presente relatório é subdividido dessa forma, e no final, imprime os resultados em forma de gráfico, caso a entrada seja um dataset ou em forma de imagem, caso a entrada também seja uma imagem.

2 Objetivos

Implementar o algoritmo fuzzy c-means aplicando um conjunto de dados disponibilizado (fcm_dataset.mat) para 4 grupos(clusters). Em seguida, comparar os resultados obtidos com a função fcm do MATLAB. Fazer uma adaptação do algoritmo anterior para o problema de segmentação de imagens, utilizando imagens disponibilizadas.

3 Metodologia

3.1 Atribuindo os valores iniciais

É nessa etapa que se carrega o conjunto de dados (imagem ou dataset) e que se define o número de clusters (k), de observações (n), de dimensões (d), máximo de iterações (maxIterations), e o valor da constante que determina a influência dos pesos (m). O número de observações e dimensões são atribuídos a partir da entrada (n = linhas, d = colunas). O valor atribuído para "m" foi 2 para que o peso dos centros fossem iguais ($\frac{2}{m-1} = 1$).

Também é nessa etapa que se define a matriz de partição inicial (U), ou seja, define-se valores para todos os U_{ij}. A matriz tem que ser definida de forma aleatória inicialmente, mas normalizada, ou seja, a soma dos elementos da linha devem ser iguais a 1. Segue abaixo a forma como foi construída:

```
 % Matriz de particao inicial normalizada (U -> n*k e 0 <
     U_ij < 1)
U = rand (n,k);
for i =1:n
    sum = 0;
    for j = 1:k
       sum = sum+U(i,j);
    end
    U(i,:) = U(i,:)/sum;
end
```

3.2 Calculando o centro de cada grupo utilizando a matriz de partição

Dadas todas as condições iniciais, foi feito um loop que finaliza caso o critério de convergência ou o máximo de iterações (maxIterations) seja atingido. A matriz de centros (C) é criada logo no início do loop. Essa matriz posssui uma dimensão de $k \times d$, ou seja, cada linha representa um centro de dimensão d.

O cálculo de C, foi feito segundo (Bezdec, 1981)[2]. A equação segue abaixo:

$$C_j = \frac{\sum_{i=1}^{n} \mu_{ij}^m X_i}{\sum_{i=1}^{n} \mu_{ij}^m} \tag{1}$$

O algoritmo foi feito da seguinte forma:

```
% Preenche a mtx de centros inicial com zeros (numerador+
    denominador)
cNum = zeros (k,d);
cDen = zeros (k,d);

% Obtem a matriz de Centros (C -> k*d)
for j=1:k
    for i=1:n
        cNum(j,:) = (U(i,j)^m)*X(i,:) + cNum(j,:);
        cDen(j,:) = (U(i,j)^m) + cDen(j,:);
    end
end

C = cNum./cDen;
```

3.3 Atualizando a matriz de partição

A matriz de partição (U) posssui uma dimensão de $n \times k$ e representa o grau de pertinência de cada elemento da matriz de entrada (X_i) a cada centro (nC_jk). Ou seja, representa o quanto cada elemento pertence a cada um dos centros.

Com a matriz de centros criada, é necessário associar cada padrão (elemento) ao centro mais próximo. Essa medida de similaridade é realizada através distância euclediana. A equação utilizada para atuzalizar a matriz de partição (segundo (Bezdec, 1981)[2]) segue abaixo:

$$\mu_{ij} = \frac{1}{\sum_{l=1}^{k} \frac{||X_i - C_j||}{||X_i - C_l||}^{(\frac{2}{m-1})}} \tag{2}$$

O algoritmo foi feito da seguinte forma:

```
    % Preenche a nova matriz de particao com zeros (numerador+
        denominador)
    uNum = 0;
    uDen = 0;

    % Obtem a nova matriz de particao (U -> m*k)
    U = zeros(n,k);
    for j=1:k
        for i=1:n
            if (pdist2(X(i,:),C(j,:)) == 0)
                U(i,:) = 0;
                U(i,j) = 1;
                break
            else
                uNum = pdist2(X(i,:),C(j,:));
                aux = 0;
                for l=1:k
                   uDen = pdist2(X(i,:),C(l,:));
                   aux = aux + ((uNum/uDen)^(2/(m-1)));
                end
            end
            U(i,j) = 1/aux;
        end
    end
```

3.4 Critério de convergência

Por fim, o algoritmo deve repetir os passos 2 e 3 até atingir o número máximo de iterações (maxIterations) ou atingir o critério de convergência. Esse critério corresponde a um limite mínimo pré definido. A função objetivo (função de custo) tende a cair a cada iteração, ou seja, é uma função a ser minimizada. Quando o resultado da variação da função objetivo atinge esse limite, o loop deve ser interrompido. Quando ocorre a interrupção, significa que os centros (clusters) convergiram e nenhum padrão mudou de grupo novamente.

O cálculo da função objetivo (função de custo), foi feito segundo (Bezdec, 1981) [2]. Para o limite mínimo, foi atribuído o valor de 1^{-5}. A equação da função de custo segue abaixo:

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{k} U_{ij}^{m} ||X_i - C_j||^2 \tag{3}$$

O algoritmo da função objetivo e a verificação de sua variação foram feitos da seguinte forma:

```
function costValue = costFunction(X,C,U,m,n,k)

   % Calcula a funcao objetivo (ou funcao de custo) a ser
       minimizada
  costValue = 0;
  for j = 1:k
      for i = 1:n
          costValue = costValue + (U(i,j)^m)*(pdist2(X(i,:),(C
              (j,:)))^2);
      end
  end

end

% Dentro do loop da funcao principal:

% Verifica a condicao de parada
costValue(r) = costFunction(X,C,U,m,n,k);
if r ~= 1
    if abs(costValue(r) - costValue(r-1)) < 1e-5
        break;
    end
end
```

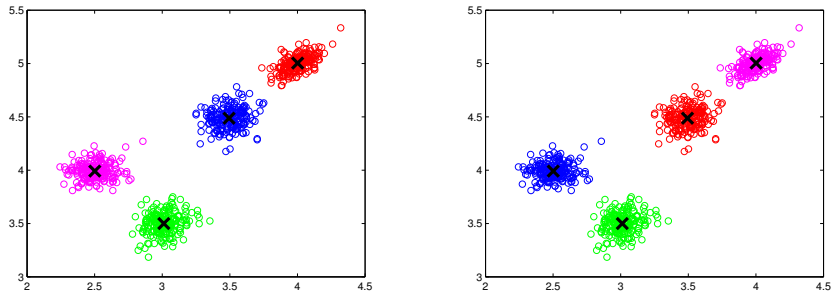
4 Resultados

4.1 Conjunto de dados - Dataset

Para o conjunto de dados, no início do código, a matriz de entrada recebe um arquivo de extensão *.mat*:

```
% Carrega o conjunto de dados (X -> m*d)
X = load('fcm_dataset.mat');
X = X.x;
```

No final do algoritmo, o resultado é plotado pela função plotGraphs. Essa função é responsável por pegar o valor máximo de cada uma das linhas da matriz de partição (U) e seus respectivos índices. A partir dessas informações, a função associa cada ponto da matriz de entrada (X) ao cluster mais relevante (imprimindo-os com a mesma cor). O resultado obtido para a matriz fcm_dataset segue abaixo junto ao resultado da função fcm nativa do MATLAB:



(a) fcm_dataset algoritmo - 4 clusters (b) fcm MATLAB (esperado) - 4 clusters

Figura 1: Testes para o fcm_dataset

4.2 Segmentação de imagens

Para a segmentação de imagens, no início do código, a matriz de entrada recebe um arquivo de extensão *.jpg*. Antes de recebê-lo, a imagem deve ser convertida para o formato de matriz. O que foi feito para solucionar o problema segue abaixo:

```
% Le, exibe e pega as dimensoes da imagem
img = imread('img02.jpg');
imshow(img);
[x,y,z] = size(img);

% De imagem a matriz de valores empilhados
newSize = [x*y 3];
stackedPixels = reshape(img, newSize);

% Carrega o conjunto de dados (X -> m*d)
X = double(stackedPixels);
```

No final do algoritmo, o resultado é plotado pela função plotImg. Essa função é responsável por pegar o valor máximo de cada uma das linhas da matriz de partição (U) e seus respectivos índices. A partir dessas informações, a função associa cada ponto (pixel) da matriz de entrada (X) ao cluster mais relevante (imprimindo-os com o valor RGB do respectivo cluster). A matriz resultante é convertida para imagem antes de ser plotada. Foram realizados 3 testes para cada imagem (para número de clusters = 3,5 e 7). Como são quatro imagens, obtém-se 12 imagens resultantes. Os resultados obtidos seguem abaixo:

(a) Imagem urso original

(b) 3 Clusters - Img urso segmentada

(c) 5 Clusters - Img urso segmentada

(d) 7 Clusters - Img urso segmentada

Figura 2: Testes para a imagem 02 - Urso

(a) Imagem confeito original

(b) 3 Clusters - Img confeito segmentada

(c) 5 Clusters - Img confeito segmentada

(d) 7 Clusters - Img confeito segmentada

Figura 3: Testes para a imagem 03 - Confeito

(a) Imagem macaco original

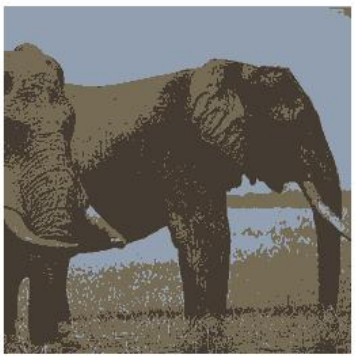
(b) 3 Clusters - Img macaco segmentada

(c) 5 Clusters - Img macaco segmentada

(d) 7 Clusters - Img macaco segmentada

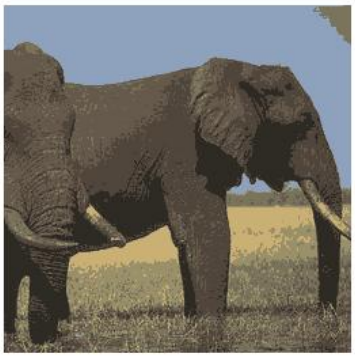
Figura 4: Testes para a imagem 04 - Macaco

(a) Imagem elefantes original



(b) 3 Clusters - Img elefantes segmentada

(c) 5 Clusters - Img elefantes segmentada



(d) 7 Clusters - Img elefantes segmentada

Figura 5: Testes para a imagem 01 - Elefantes

5 Discussão

Durante os experimentos observou-se uma queda no resultado das funções objetivo, o que era previso, já que a função deveria ser minimizada. Ou seja, os centros (clusters) convergiram tanto para o dataset quanto para as imagens.

Comparando os resultados obtidos com a função fcm do MATLAB, observa-se uma semelhança muito grande entre os gráficos. Isso é um indicativo de que o algoritmo foi implementado de forma correta.

Por fim, avaliando as imagens, observa-se um crescimento de segmentação à medida que o número de clusters cresce. Isso ocorre porque quando existem mais clusters, os pixels irão se identificar com os centros mais locais (mais grupos geram imagens mais segmentadas). Esse também era um resultado esperado, o que indica uma implementação correta do algoritmo.

Analisando o exposto acima, observa-se resultados bastante satisfatórios. Sendo assim, os objetivos propostos foram resolvidos, possibilitando que o exercício computacional fosse concluído com êxito.

Referências

[1] Hisao Ishibuchi and Tomaharu Nakashima. ***Effect of Rule Weights in Fuzzy Rule-Based Classification Systems***. IEEE TRANSACTIONS ON FUZZY SYSTEMS, VOL. 9, NO. 4, AUGUST 2001.

[2] Bezdec, J.C. ***Pattern Recognition with Fuzzy Objective Function Algorithms***. PLENUM PRESS, New York, 1981.

[3] ***Mathworks - Fuzzy c-means clustering***. Disponível em: `<https://www.mathworks.com/help/fuzzy/fcm.html>`. Acesso em 22 de maio de 2017.