EXERCÍCIO COMPUTACIONAL #3

UFMG - EEE920 Sistemas Nebulosos

TAIGUARA MELO TUPINAMBÁS

01 de outubro de 2017

Parte 1 – fcm_dataset

Inicialmente os dados são carregados no MatLab, os parâmetros do *fuzzy c-means* são definidos e os centros são aleatorizados.

```
load('fcm_dataset.mat');

%parametros
m = 2; %c-means
eps = 1e-3; %criterio de parada 1
N = 1e3; %criterio de parada 2
p = 4; %quantidade de centros

%aleatoriza os centros
index = round(size(x,1)*rand(p,1));
c = x(index,:);
```

Foi escolhido um m igual a 2, conforme indicado em sala de aula. A quantidade de centros foi escolhida com base na imagem plotada dos dados (Figura 2), onde notam-se 4 grupos a serem classificados.

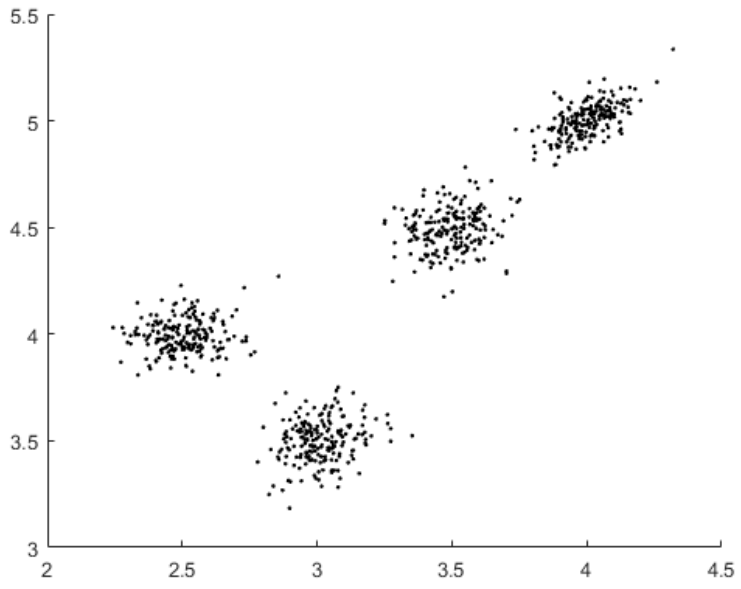

Figura 1 – Dados plotados

A primeira parte do algoritmo consiste no cálculo da matriz de pertinência U, que depende das distâncias entre cada ponto e cada centro, ou seja uma matriz de tamanho 800x4 (quantidade de pontos x quantidade de centros).

Para o cálculo das pertinências, foi inicializado uma matriz de zeros e, para cada ponto é verificado se aquele ponto é algum centro e, caso seja é dado o valor 1 para a pertinência daquele ponto naquele centro e 0 para os demais. Caso não seja o próprio centro, é calculado a distância do ponto a todos os centros através da fórmula (2) do artigo de Cannon, Dave & Bezdek (1986).

```
    %calculo da matriz de pertinencia
    U = zeros(size(x,1),size(c,1));

    for i=1:size(x,1)
        %checa se o ponto i é algum centro
        if sum(sum(repmat(x(i,:),[4,1])==c))==2;
            [~,ind]=max(repmat(x(i,:),[4,1])==c);
            U(i,:)=0; %atribui 0 de pertinencia aos outros centros
            U(i,ind(1))=1; %atribui 1 de pertinencia ao centro que coin-
cide com o ponto
        else
            %calcula as distancias
            for j=1:size(c,1)
                dist(j) = norm(x(i,:)-c(j,:));
            end
            for j=1:size(c,1)
                s=0;
                %somatoria do denominador conforme o artigo
                for k=1:size(c,1)
                    s=s+(dist(j)/dist(k))^2;
                end
                U(i,j) = 1/s;
            end
        end

    end
```

Em seguida, com base na matriz U, calcula-se novos centros utilizando a fórmula (1) do mesmo artigo.

```
    %% novos centros
    c_new=zeros(size(c));
    for j=1:size(c,1)
        aux=U(:,j).^m; %matriz com as pertinencias daquele centro ao qua-
drado
        c_new(j,:)=sum([aux aux].*x,1)/sum(aux);%calcula o novo centro j
    end
```

Estas etapas de cálculo da matriz U e dos centros são executadas iterativamente, até que algum dos critérios de parada seja atingido: quantidade de iterações maior do que N (1e3) ou a norma da diferença entre o centro novo e o centro antigo seja menor do que o valor de epsilon (1e-3).

Cria-se, desta forma, um loop conforme código abaixo.

```
c_old = zeros(size(c));
iter=0;

while norm(c-c_old)>eps && iter<N

    %calculo da matriz de pertinencia
    U = zeros(size(x,1),size(c,1));

    for i=1:size(x,1)
        %checa se o ponto i é algum centro
        if sum(sum(repmat(x(i,:),[4,1])==c))==2;
            [~,ind]=max(repmat(x(i,:),[4,1])==c);
            U(i,:)=0; %atribui 0 de pertinencia aos outros centros
```

```
            U(i,ind(1))=1; %atribui 1 de pertinencia ao centro que coin-
cide com o ponto
        else
            %calcula as distancias
            for j=1:size(c,1)
                dist(j) = norm(x(i,:)-c(j,:));
            end
            for j=1:size(c,1)
                s=0;
                %somatoria conforme o artigo
                for k=1:size(c,1)
                    s=s+(dist(j)/dist(k))^2;
                end
                U(i,j) = 1/s;
            end
        end

    end

    %% novos centros
    c_new=zeros(size(c));
    for j=1:size(c,1)
        aux=U(:,j).^m; %matriz com as pertinencias daquele centro ao qua-
drado
        c_new(j,:)=sum([aux aux].*x,1)/sum(aux);%calcula o novo centro j
    end

    %critérios de parada
    c_old=c;
    c=c_new;
    iter=iter+1

end
```

Ao final das iterações, utiliza-se a matriz de pertinência U para saber qual centro possui maior pertinência para cada ponto, resultando na classificação desejada.

Cada ponto é então colorido conforme o seu centro e os resultados são plotados, para avaliar se a classificação foi feita adequadamente.

```
c_[~,ind]=max(U,[],2);

for k = 1 : length(ind)
    if ind(k)==1
        colorMap(k, :) = [1,0,0]; % Red
    elseif ind(k)==2
        colorMap(k, :) = [0,0,1]; % Blue
    elseif ind(k)==3
        colorMap(k, :) = [0,1,0]; % Green
    elseif ind(k)==4
        colorMap(k, :) = [1,0,1]; % Pink
    end
end

scatter(x(:,1),x(:,2),5,colorMap,'filled')
```

O resultado é apresentado na Figura 2.

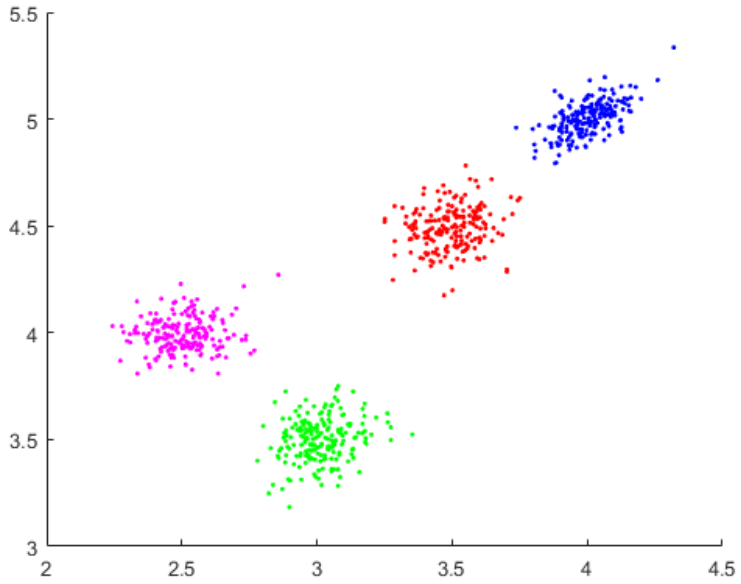

Figura 2 – Dados coloridos conforme centros de maior pertinência

Os resultados obtidos pelo algoritmo foram comparados com a saída da função fcm()

```
%% comparacao com a função fcm()

c_fcm = fcm(x,4)

c_fcm =

    3.0120    3.4999
    2.5012    3.9916
    3.4940    4.4876
    4.0014    5.0040


c =

    2.5013    3.9916
    4.0014    5.0040
    3.4940    4.4876
    3.0120    3.4999
```

Os centros obtidos pela função fcm() foram exatamente os mesmo obtidos pelo algoritmo desenvolvido, conforme esperado.

Parte 2 – Regiões de Classificação para 2 dimensões

O algoritmo foi adaptado para fazer a classificação das imagens disponibilizadas no moodle.

Inicialmente, carrega-se a imagem a ser classificada, e transforma os dados em uma matriz em que cada linha seja um ponto da imagem e cada coluna o valor que representa sua cor.

```
img = imread('img03.jpg');

x=zeros(size(img,1)*size(img,2),3);

%% transformando a imagem em uma matriz
ind=0;
for i=1:size(img,1)
    for j=1:size(img,2)
        ind=ind+1;
        x(ind,:)=img(i,j,:);
    end
end
```

Uma vez que os dados foram transformados para o mesmo formato tratado na parte 1, o algoritmo era executado da mesma forma, conforme código a seguir

```
%parametros
m = 2; %c-means
eps = 1e-2; %criterio de parada 1
N = 1e5; %criterio de parada 2
p = 4; %quantidade de centros
n = size(x,1); %qtd de padrões

% %aleatoriza os centros
index = round(n*rand(p,1));
c = x(index,:);

c_old = zeros(size(c));
iter=0;

while norm(c-c_old)>eps && iter<N

    %calculo da matriz de pertinencia
    U = zeros(size(x,1),size(c,1));
    for i=1:size(x,1)
        %checa se o ponto i é algum centro
        if sum(sum(repmat(x(i,:),[p,1])==c))>=3;
            [~,ind]=max(sum(repmat(x(i,:),[p,1])==c,2));
            U(i,:)=0; %atribui 0 de pertinencia aos outros centros
            U(i,ind(1))=1; %atribui 1 de pertinencia ao centro que coin-
cide com o ponto
        else
            %calcula as distancias
            for j=1:size(c,1)
                dist(j) = norm(x(i,:)-c(j,:));
            end
            for j=1:size(c,1)
                s=0;
                %somatoria conforme o artigo
                for k=1:size(c,1)
                    s=s+(dist(j)/dist(k))^2;
                end
                U(i,j) = 1/s;
```

```
            end
        end
    end
    %% novos centros
    c_new=zeros(size(c));
    for j=1:size(c,1)
        aux=U(:,j).^m; %matriz com as pertinencias daquele centro ao qua-
drado
        c_new(j,:)=sum(repmat(aux,[1,3]).*x,1)/sum(aux);%calcula o novo
centro j
    end

    %critérios de parada
    c_old=c;
    c=c_new;
    iter=iter+1;

end
```

Ao terminar a classificação, era necessário transformar os dados em imagem novamente, atribuindo a cada ponto as cores do centro de maior pertinência.
Para a imagem ser plotada, foi necessário transformar as variáveis em uint8.

```
%% retornando a imagem
ind=0;
img_class=zeros(size(img,1),size(img,2),3);
for i=1:size(img,1)
    for j=1:size(img,2)
        ind=ind+1;
        [~,centro]=max(U(ind,:));
        img_class(i,j,:)=c(centro,:);
    end
end

image(uint8(img_class))
```

2.5 Resultados

O algoritmo foi executado para a imagem 3, variando-se a quantidade de centros (de 2 a 7). Os resultados são apresentados na Figura 3.

Nota-se que todas as cores das balas aparecem apenas para quantidades de centros maior do que 5. A diferença entre a imagem classificada com 6 centros e com 7 centros é muito sutil.

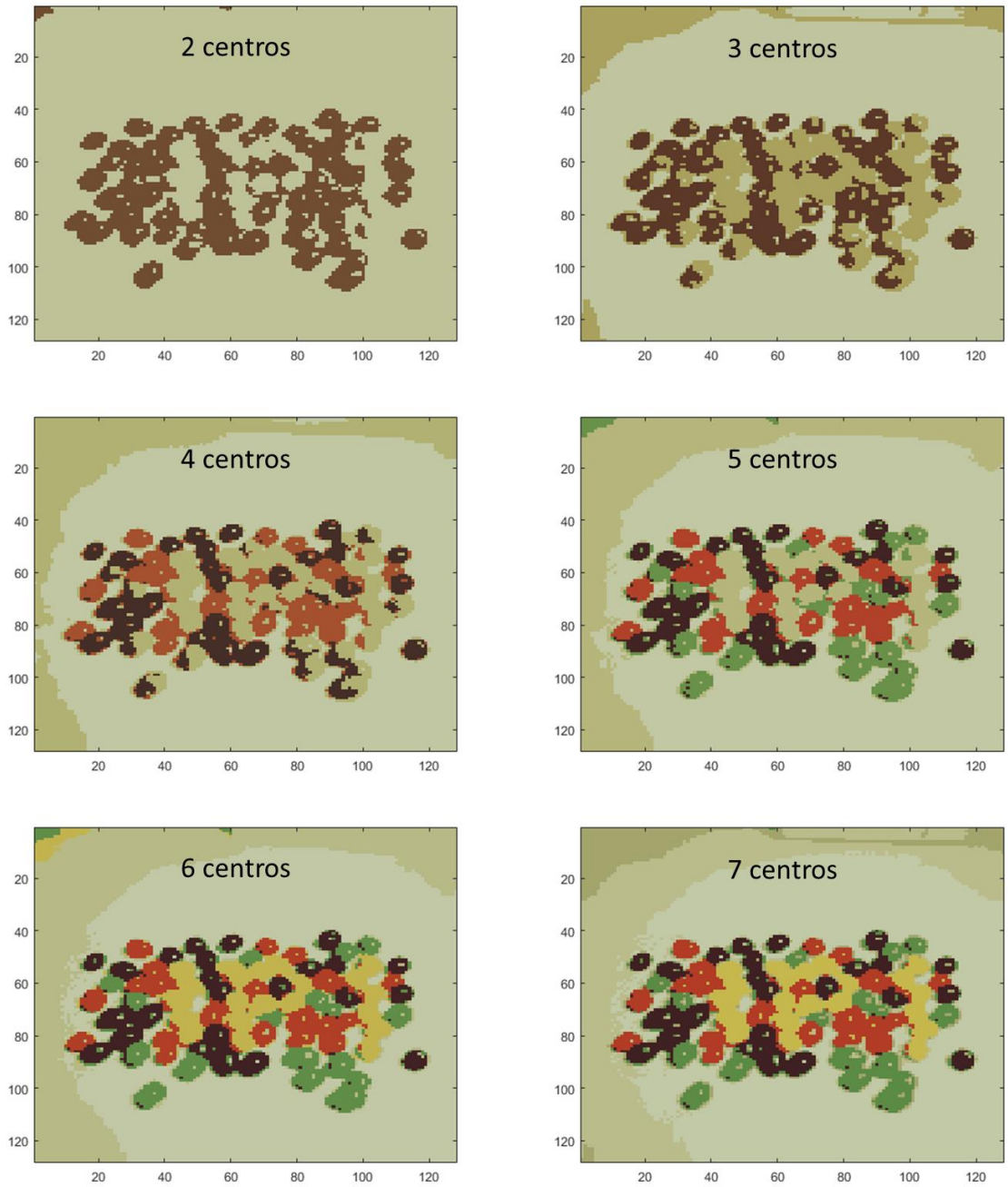

Figura 3 – img03 classificada para diferentes quantidades de centros.

O algoritmo foi comparado com a função fcm() para 6 centros, e os valores encontrados foram muito próximos:

```
c_fcm =

  177.2482   61.8892   39.2863
   96.0946  142.3443   71.8919
  194.4292  179.0304   79.4511
   64.7794   34.4912   35.7027
  183.2395  184.6549  133.7520
  193.9716  200.1131  165.6885
```

```
c_algoritmo =

  193.9722  200.1140  165.6911
  194.4161  179.0187   79.4584
  177.2486   61.8887   39.2861
   96.0922  142.3433   71.8910
   64.7793   34.4909   35.7026
  183.2459  184.6616  133.7622
```

Em seguida, as outras imagens foram classificadas para uma quantidade de centros arbitrária e os resultados são apresentados nas figuras a seguir

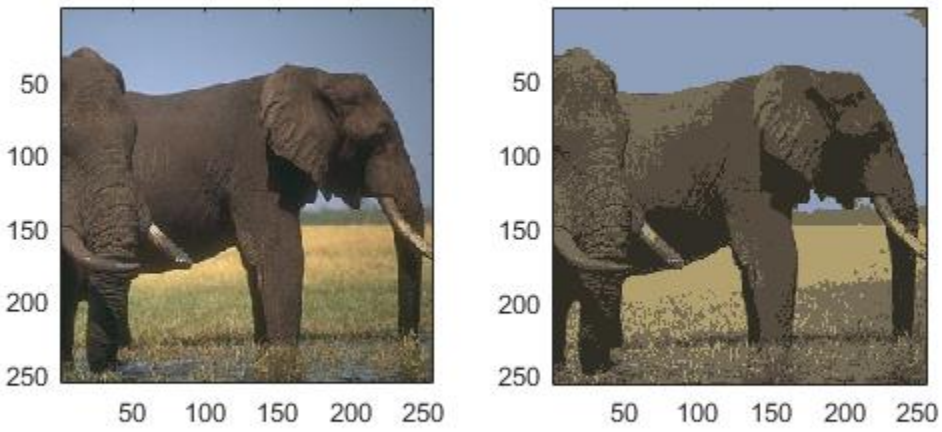

Figura 4 – img-03 original (esquerda) e a classificada com 6 centros (direita)

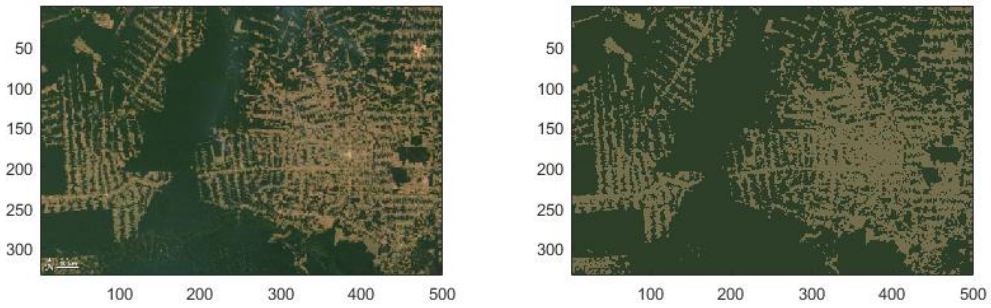

Figura 5 – img2 original (esquerda) e classificada com 2 centros (direita)

Bibliografia

[1] R. L. Cannon, J. V. Dave, and J. C. Bezdek, “Efficient Implementation of the Fuzzy c-Means Clustering Algorithms,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 2, pp. 248–255, 1986.