**UNIVERSIDADE FEDERAL DE MINAS GERAIS - UFMG**
**ELE075 - Sistemas Nebulosos - 2018/2**

**Professor:** Cristiano Leite de Castro

# Trabalho Computacional 1 - Fuzzy C-Means

**Aluno:** Rafael Carneiro de Castro **Matrícula:** 2013030210
**Curso:** Engenharia de Sistemas

## 1 - Introdução:

Este trabalho consiste da implementação do algoritmo *Fuzzy C-Means* (ou apenas *FCM*), assim como sua validação e testes. Os primeiros testes foram executados em uma base de dados sintéticos de duas dimensões. Dessa forma é possível plotar os pontos e os centróides calculados ao final da iteração do algoritmo implementado. Outro teste é feito a partir da segmentação de imagens RGB. Para cada imagem de teste um número de agrupamento de cores é escolhido, empiricamente, com base na observação das cores das diferentes regiões. Executando o algoritmo do *Fuzzy C-Means* sobre as imagens segmentadas, obtém-se uma matriz de pertinência, que pode ser usada para colorir cada região com a tonalidade do pixel que corresponde ao centro da região calculado pelo algoritmo. Os pixels que apresentarem maior grau de compatibilidade (pertinência) a uma dada região devem ser coloridos com a tonalidade do pixel central daquela região.

## 2 - Desenvolvimento:

Para a implementação do algoritmo, usou-se como base o algoritmo *KMeans* disponibilizado pelo professor. Tanto o código do professor como o desenvolvido aqui são em MatLAB. O primeiro passo a ser executado é a definição da pertinência de cada observação, de forma aleatória, a cada um dos grupos, cuja quantidade é um parâmetro, chamado de $K$. Para isso, usou-se a função *rand* do MatLAB. Em seguida calcula-se o valor da função objetivo para esta matriz de pertinência aleatória inicial. A função objetivo, no caso, é a variância da distância dos pontos ao centróide de seu grupo definido, valor este que se quer minimizar.

Após os passos iniciais, entra em execução o loop de iteração do algoritmo. Neste loop, o primeiro passo é o cálculo dos centróides a partir da matriz de pertinência atual. Este cálculo é feito com base na fórmula vista em sala de aula, apresentada a seguir. Nesta fórmula, o valor de $m$ é 2, $x$ são as observações e $u$ são os valores da matriz de pertinência. Pode ser feito no MatLAB utilizando como auxílio as operações matemáticas sobre matrizes e a função *repmat.*

$$c_k = \frac{\sum_{i=1}^{n} u_{i,k}^{m} x_i}{\sum_{i=1}^{n} u_{i,k}^{m}}$$

Com os centróides calculados, é possível calcular a nova matriz de pertinência $U$ também com a fórmula vista em sala de aula e apresentada a seguir. Mais uma vez o valor de $m$ é 2, $c$ são os centróides calculados no passo anterior, $K$ é a quantidade de grupos, passada por parâmetro, e $x$ são as observações. Novamente, para o cálculo desta função, são utilizadas as operações sobre matrizes do MatLAB e a função *repmat.* Vale salientar que os valores da matriz de pertinência são normalizados, para que a soma das pertinências de uma observação seja sempre igual a 1.

$$u_{i,k} = \left(\frac{(x_i - c_k)^2}{\sum_{t=i}^{K}(x_i - c_t)^2}\right)^{-2/(m-1)}$$

Em seguida é possível calcular a função objetivo, mais uma vez como sendo variância da distância dos pontos ao centróide de seu grupo definido, valor este que se quer minimizar. Os valores calculados da função objetivos são guardados num vetor para cada iteração. Desta forma, ao final da execução é possível plotar um gráfico com a evolução dos valores da função objetivo.

Por fim, passa-se pela decisão do critério de parada do loop do algoritmo. No *KMeans*, um critério possível é simplesmente avaliar se o grupo de todos os pontos se manteve o mesmo comparando-se à iteração anterior. Caso nada tenha mudado, as iterações são finalizadas. Contudo, para o *Fuzzy C-Means* o que define o grupo de cada observação é a matriz de pertinência. Cada ponto vai pertencer ao grupo em que possui maior valor de pertinência. Desta forma, de uma iteração para outra, a matriz de pertinência pode mudar sem que os grupos das observações mude. Neste cenário, duas melhores opções de critério de parada passam a ser as mudanças na matriz de pertinência ou as mudanças nos centróides. No algoritmo implementado para o trabalho, optou-se pela segunda opção, por ser uma menor quantidade de valores a se comparar, sendo um pouco mais performático. Assim, se os centróides se mantém inalterados por três iterações seguidas, elas são paradas. Com o fim das iterações, é possível plotar a evolução da função objetivo e os centróides finais alcançados. O resultado do desenvolvimento é a função presente no arquivo *FCM.m,* anexa junto a este documento.

## 3 - Testes e Resultados:

Para os primeiros testes, uma base de dados sintéticos foi utilizada. Ela foi disponibilizada pelo professor, e está no arquivo *fcm_dataset.mat.* Executando a função *FMC* desenvolvida com a linha de comando "FCM(x, 4, 10^-2, true);" (o "4" é a quantidade de grupos, "10^-2" é a tolerância na diferença das matrizes de pertinência e "true" é para se plotar os gráficos de função objetivo e de centróides) no MatLAB, após carregar o arquivo *fcm_dataset.mat* na variável x, o resultado é a evolução dos valores da função objetivo e centróides mostrados na figura a seguir. Como se pode ver, a função objetivo teve o valor diminuído durante a evolução das iterações. Os centróides estão, por inspeção, próximos aos centros dos quatro grupos de pontos visivelmente separados.

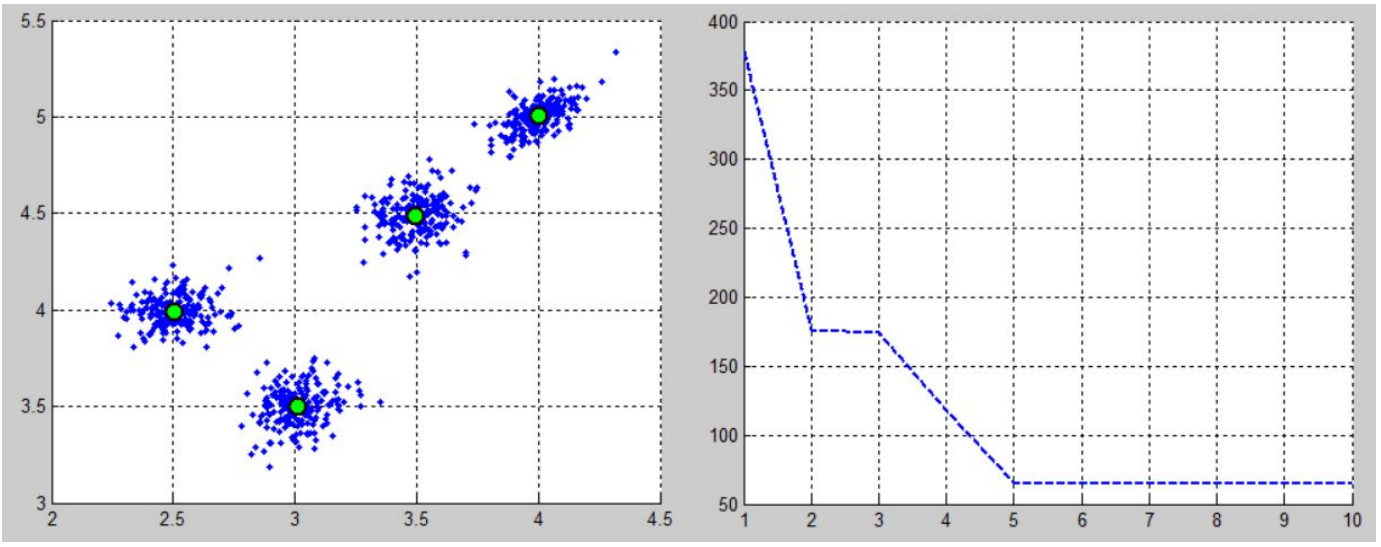


**Figura 1: Centróides e função objetivo após a execução do FCM.**

O código contido no arquivo *main.m*, também anexo a este relatório, foi criado para automatizar os testes seguintes. O algoritmo do *KMeans* disponibilizado pelo professor foi extraído para o arquivo *KMeans.m,* em forma de função. Em *main.m*, os algoritmos do *FCM* e do *KMeans* foram executados, paralelamente, por 30 iterações na base de dados sintéticos. Ao fim das 30 execuções, são mostradas as médias das quantidades de iterações rodadas em cada algoritmo, e a quantidade de vezes, dentre as 30, que cada algoritmo encontrou os centróides corretos. O resultado pode ser visto a seguir.

```
===== FCM
    CENTROS CORRETOS: 25
    MÉDIA DE ITERAÇÕES: 10.600000
===== KMeans
    CENTROS CORRETOS: 11
    MÉDIA DE ITERAÇÕES: 6.900000
```

**Figura 2: Comparação entre o FCM e o KMeans.**

Como se pode notar, o *FMC* teve uma média maior de iterações, principalmente por conta do critério de parada utilizado, que o força a ter três iterações com centróides iguais antes de finalizar os cálculos. Contudo, o *FCM* encontrou com sucesso os centróides em mais iterações que o *KMeans*, se mostrando ser mais robusto, convergindo de forma mais recorrente.

O último teste executado sobre o algoritmo do *FCM.m* desenvolvido foi o teste de segmentação de imagens descrito na Introdução. As 11 imagens de teste utilizadas, disponibilizadas pelo professor, estão no diretório *ImagensTeste*. No arquivo *main.m*, a quantidade de cores para execução do *FCM* em cada imagem é definida. Também é feita a segmentação das imagens, execução do algoritmo para o cálculo das matrizes de pertinência para cada uma delas e exibição do resultado final, mostrado a seguir. Nos pares, as imagens à esquerda são as originais, e à direita as criadas após o agrupamento por cores do *FCM.*

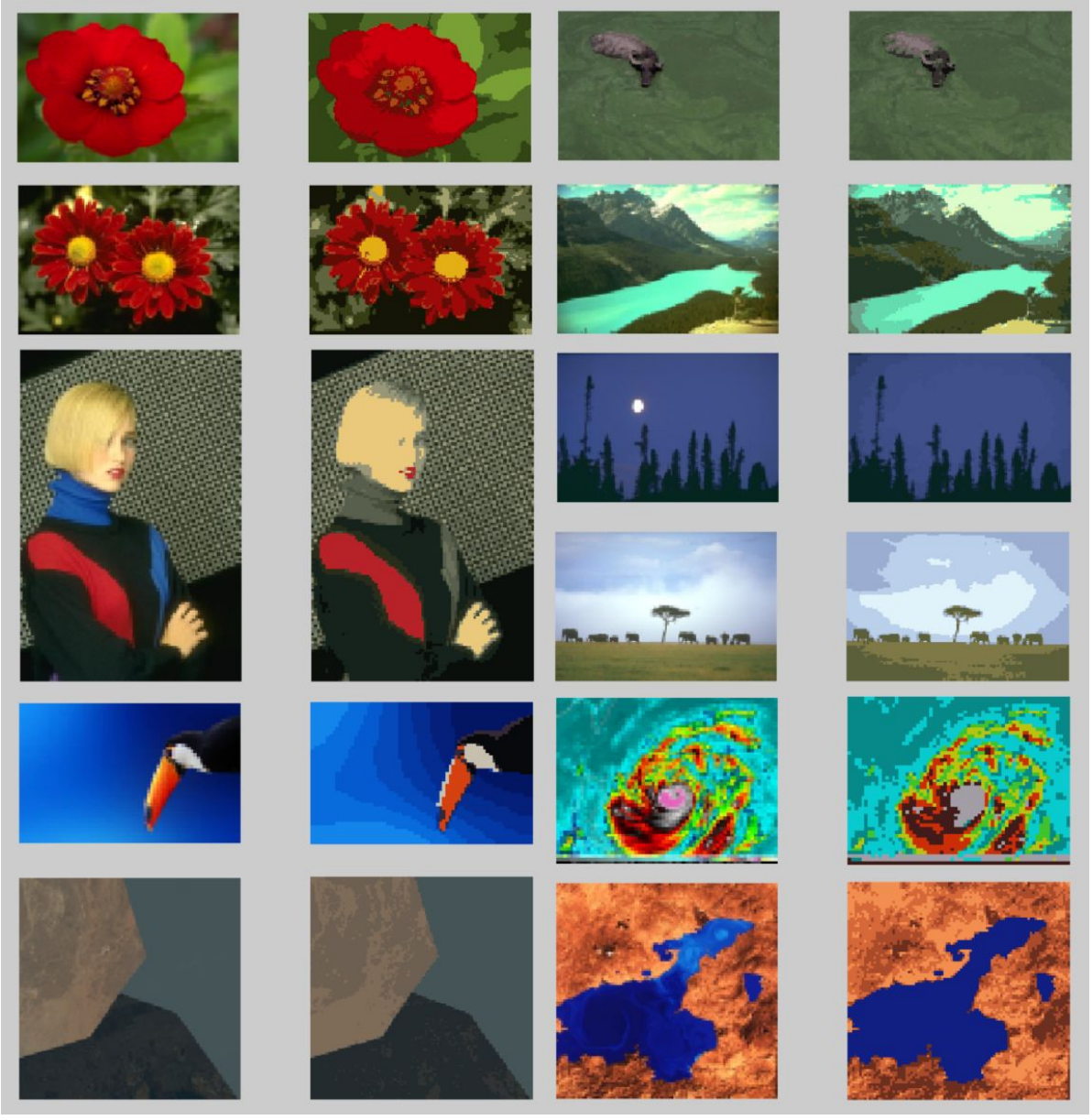

**Figura 3 - Testes do FCM com segmentação de imagens.**

Como se pode notar, o resultado final foi bem interessante, com as imagens geradas estando bem próximas das originais, considerando suas limitações de cores. Listando a partir da primeira, no canto superior esquerdo, passando pela segunda, à sua direita, indo até a última, no canto inferior direito, a quantidade de cores definidas são: 7, 17, 8, 9, 11, 7, 5, 12, 7, 5 e 5.

## 4 - Conclusão:

A implementação do algoritmo do *FCM* apresentado aqui se mostrou uma ótima maneira para se exercitar os conceitos aprendidos em sala de aula, através de uma aplicação prática. Após a superação dos obstáculos de implementação, como a execução das fórmulas no MatLAB e definição do critério de parada, considera-se que os objetivos do trabalho foram alcançados, e os resultados apresentados foram satisfatórios.