UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
DISCIPLINA: SISTEMAS INTELIGENTES
PROFESSORA: DEBORAH MAGALHÃES

3a PRÁTICA COMPUTACIONAL

Algoritmo K-means

1. Fase 1: implementando o K-means

a. Descrição da etapa

O objetivo dessa prática reside em adquirir noções básicas de clusterização, através do algoritmo k-means. A **primeira** etapa consiste em implementar o algoritmo K-means. A **segunda** consiste em utilizá-lo no processamento de uma imagem. Primeiro, vamos começar com um dataset 2D (**ex7data2.mat)**, para nos familiarizarmos com o funcionamento do K-means. Em seguida, o algoritmo K-means é aplicado na compressão de uma imagem através da redução do número de cores da imagem para somente àquelas que são mais comuns. Os códigos necessários para execução desta etapa são:

- main.m -> código principal que realiza a chamada para as demais funções;

* findClosestCentroids.m -> associa cada amostra de entrada ao centróide mais próximo;

*computeCentroids.m -> calcula o novo centróide a partir do ponto médio de todas as amostras atribuídas aquele centróide;

- runkMeans.m -> código responsável pela execução do K-means

*kMeansInitCentroids.m ->escolhe aleatoriamente K exemplos do conjunto de dados para inicializar os centróides

- plotProgresskMeans.m -> responsável por plotar cada iteração realizada pelo k-means;

- plotDataPoints.m -> colore os pontos conforme o cluster associado;

- drawLine.m -> desenha uma linha sobre figura existente.

* São as funções que você deve modificar. Todas elas estão disponíveis no sigaa.

b. O que deve ser feito?

1. **Encontrar o centróide mais próximo:** o algoritmo atribui a todos os exemplos de treinamento x(i), o centróide mais próximo, dadas as posições atuais dos centróides. Especificamente, para cada exemplo, devemos definir:

$$c^{(i)} := j \; ||x^{(i)} - \mu_j||^2,$$

onde c(i) é o índice do centróide mais próximo de x(i) e μj é a posição do j-ésimo centróide. Note que c(i) corresponde a idx(i) no código. Sua tarefa é modificar o código **findClosestCentroids.m**. Essa recebe uma matriz de dados X e a posição inicial dos K centróides. Ela deve retornar um vetor **idx** que contém o índice (valor do centróide [1, ... K]) do centróide mais próximo para todos os exemplos de treinamento. Se você implementou corretamente, os clusters retornados para os 3 primeiros exemplos seguem a seguinte ordem: 1, 3, 2 respectivamente ;

2. **Calcular o ponto médio do agrupamento e definir como novo centróide**: na etapa de movimentação do centróide, o algoritmo calcula, para cada centróide, a média dos pontos que foram atribuídos a ele. Especificamente, para cada centroid k definimos:

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$$

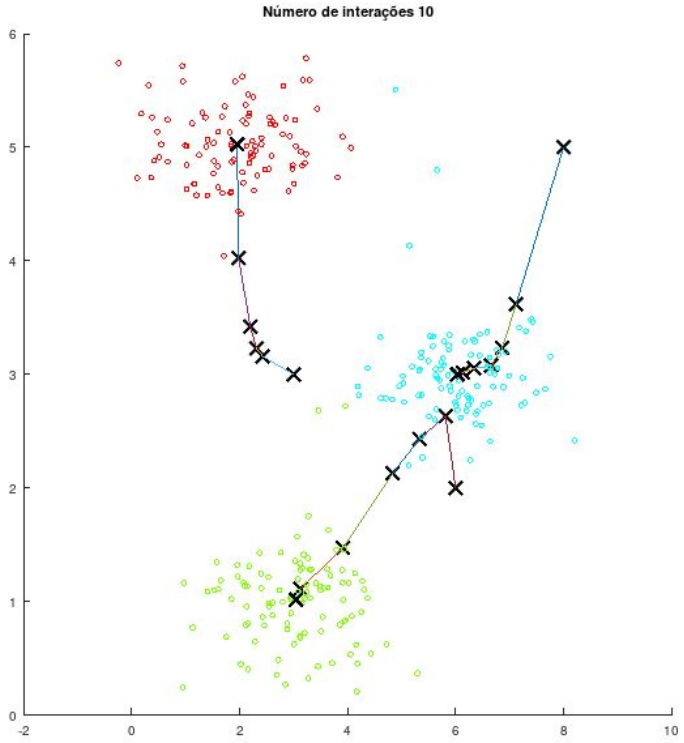
onde Ck é o conjunto de exemplos que são atribuídos ao centróide k. Você deve completar o código **computeCentroids.m** para calcular o novo ponto médio de cada cluster. Esse código deve retornar uma matriz chamada centroids cujas linhas correspondem a cada centróide e as colunas correspondem as coordenadas x e y. Para o exemplo do código, os resultados encontrados devem ser:

1 - (2.428301 3.157924)

2 - (5.813503 2.633656)

3 - (7.119387 3.616684)

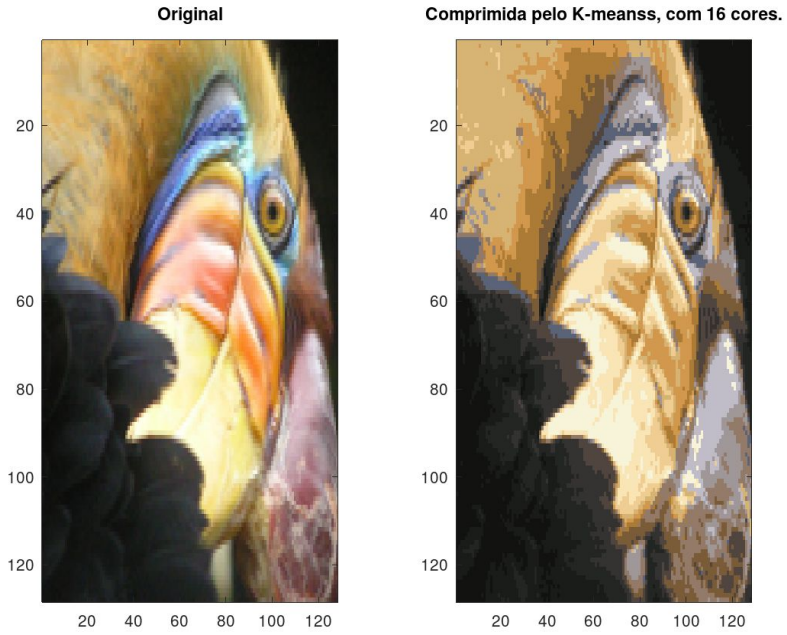
Se você implementou as duas funções **findClosestCentroids.m** e **computeCentroids.m**, o próximo passo consiste em executar o K-means em um dataset 2D (**ex7data2.mat**) para você visualizar como o K-means funciona. A função **runKmeans.m** implementa o algoritmo K-means e chama as funções que você implementou. Recomendo fortemente que você dê uma olhada em como essa função funciona. Se tudo estiver corretamente implementado, você deverá obter a saída ilustrada na Figura 1, que mostra o processo de aprendizado do K-means a cada iteração e a posição final dos centróides.



3. **Inicializar os centróides randomicamente:** nesta etapa, você usará o K-Means para comprimir uma imagem. Para fazer isso, primeiro você executará K-Means nas cores dos pixels da imagem e, então, mapeará cada pixel em seu centróide mais próximo. Na prática, uma boa estratégia consiste em inicializar os centróides randomicamente. Nessa etapa, você deverá completar a função **kMeansInitCentroids.m**. Onde as amostras devem ser aleatoriamente embaralhadas e, em seguida, os K primeiros exemplos devem ser selecionados como centróides. Dica: veja a função **randperm**.

4. **Aplicar a compressão em outra imagem de sua escolha:** Na codificação RGB, cada pixel é representado por três inteiros de 8 bits (variando de 0 a 255) que especificam os valores de intensidade vermelho, verde e azul, resultando em 24 bits. A imagem do pássaro contém milhares de cores. O objetivo é reduzir o número de cores para 16. Ao fazer essa redução, é possível compactar a imagem de maneira eficiente, pois é possível representar cada pixel por 4 bits onde é possível representar as 16 possibilidades de cores.

   Neste exercício, você usará o algoritmo K-means para selecionar as 16 cores que serão usadas para representar a imagem compactada. Depois de encontrar as K=16 cores (centroids) para representar a imagem, você utilizará a função **findClosestCentroids.m** para atribuir cada posição do pixel ao seu centróide mais próximo. É possível recuperar a imagem comprimida através dos índices dos centróides mapeados no vetor idx. Se você executou tudo corretamente, deve encontrar através da execução do main.m, você deve encontrar a imagem ilustrada na Figura abaixo.



   Nesta etapa, sua tarefa consiste em variar os parâmetro **K** e **max_iters** e verificar o impacto na imagem comprimida. Ainda, você deve utilizar outra imagem diferente do pássaro. Tal imagem pode ser tanto .mat como .png. Dica: se você optou por utilizar uma imagem png, o comando **imread** será útil.

2. Avaliação

Este trabalho corresponde a uma parte da segunda avaliação parcial da disciplina e deverá ser entregue no dia **19/11**. O trabalho poderá ser apresentado de dupla e assumirá o valor de **0-2.5**.

Os seguintes critérios serão considerados na avaliação:

1. Atender ao que foi pedido na descrição deste documento;
2. Compreender os conceitos discutidos em sala;
3. Código está executando sem erros.

Atenção: se identificada a cópia de código, a nota **zero** será atribuída aos envolvidos.