МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ
Институт системной и программной инженерии
и информационных технологий (Институт СПИНТех)

Лабораторный практикум по курсу "Интеллектуальные системы" (09/22 – 01/23)

Лабораторная работа 4

Кластеризация данных методом *k*-средних [1].

На этом занятии компьютерного практикума предстоит изучить кластеризацию метод*ом* *k*-средних и применить данный метод для сжатия изображения. Прежде чем приступить, собственно, к программированию, настоятельно рекомендуется ознакомиться с материалом лекций, а также с дополнительными материалами, имеющими отношение к задаче кластеризации методом *k*-средних.

Файлы, включенные в задание:

ex4.m - скрипт, реализующий пошаговое выполнение задания по разделу «Кластеризация методом *k*-средних»;
ex4data.mat – набор данных для метода *k*-средних;
bird_small.png – пример изображения;
displayData.m – отображает 2-мерные данные, сохраненные в матрице;
drawLine.m – рисует линию на существующем графике;
plotDataPoints.m – выводит в графическом виде центры для метода *k*-средних;
plotProgresskMeans.m – выводит в графическом виде каждый шаг метода *k*-средних во время расчета;
runkMeans.m – запуск метода *k*-средних.

* *findClosestCentroids.m* – находит ближайшие центры (для метода *k*-средних);
* *computeCentroids.m* – рассчитывает среднее значение центров (для метода *k*-средних)
* *kMeansInitCentroids.m* – задание центров (для метода *k*-средних)

Функции, отмеченные знаком *, следует написать самостоятельно.

В этой Лабораторной работе следует использовать скрипт *ex4.m*, не изменяя его. В данном скрипте (короткой программе) подготовлены обращения к исходным данным. Далее производится вызов функций, написанных Вами, и отображаются результаты вычислений. Необходимо дописать функции в файлах по инструкциям упражнения.

[1] Материал лабораторной работы составлен на основании аналогичного задания по курсу «Машинное обучение» на портале онлайн обучения Coursera.org (профессор Эндрю Ын, Стэнфордский университет - https://ru.wikipedia.org/wiki/Ын,_Эндрю)

1 Метод *k*-средних

В этом упражнении предстоит изучить как, собственно, сам метод *k*-средних, так и применить его для сжатия изображений. В первой части Вы применяете метод *k*-средних для обработки 2-мерного набора данных, что способствует лучшему пониманию алгоритма. Далее, метод *k*-средних применяется для сжатия изображения, посредством уменьшения количества цветов и сохранения только тех цветов, которые встречаются в данном изображении чаще всего. Используйте файл *ex4.m*.

Указание: Вы располагаете также написанными ранее (Лабораторные работы 1-2) программами, которые здесь можно использовать.

1.1 Использование метода *k*-средних

Метод *k*-средних автоматически объединяет в группы схожие данные. Например, пусть задан набор данных для обучения $\{x^{(1)}, ..., x^{(m)}\}$, где $x^{(i)} \in R^n$, и Вы хотите его объединить в несколько сплоченных «кластеров». Принцип работы метода *k*-средних, заключается в том, что он является итерационной процедурой, в которой начальные центры задаются некоторым образом, часто случайным, а далее повторно уточняются. Уточнение сводится к перегруппировке примеров (точек) к ближайшим центрам и последующему пересчету местоположения центров.

Указание: код для метода *k*-средних выглядит следующим образом:

```
% Задание начального положение центров
centroids = kMeansInitCentroids(X, K);
for iter = 1:iterations
% Присвоение каждой точки ближайшему центру
% idx(i) соответствует c^(i) - индексу центра i-го примера
idx = findClosestCentroids(X, centroids);
% Шаг перемещения центра: Вычисление среднего местоположения
centroids = computeMeans(X, idx, K);
end
```

Внутренний цикл осуществляет две операции: 1) группировка каждого примера $x^{(i)}$ к ближайшему центру кластеризации и 2) перерасчет среднего значения каждого центра, используя примеры, которые отнесены к его группе. Метод *k*-средних всегда сходится и находит конечное среднее для центров. Обратите внимание, что решение не всегда будет оптимальным и зависит от начальных значений центров. Соответственно, метод *k*-средних обычно запускают несколько раз с различными начальными параметрами. Один из способов выбрать между различными решениями - это найти то решение, для которого функция стоимости (искажение) имеет наименьшее значение.

В следующей части требуется выполнить два этапа метода *k*-средних по отдельности.

1.1.1 Нахождение ближайших центров

На первом этапе алгоритм метода *k*-средних, группирует каждый пример $x^{(i)}$ к его ближайшему центру для выбранного расположения центров. Для каждого примера *i*:

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

где $c^{(i)}$ - индекс ближайшего к $x^{(i)}$ центра, а μ_j - положение (численное значение) *j*-го центра. Обратите внимание на то, что в коде $c^{(i)}$ обозначен как *idx(i)*.

Задание: Ваша задача завершить код в *findClosestCentroids.m*. Это функция, использует матрицу X и расположение центров для вывода одномерного массива *idx*. Массив *idx* содержит индекс (значение принадлежащее *{1,., K}* где *K* – число центров) ближайшего центра к каждому примеру. Это может быть реализовано циклом для каждого примера для каждого центра. После завершения кода в *findClosestCentroids.m*, программа *ex4.m* запустит код и Вы увидите результат [1 3 2], соответствующий ближайшим центрам для первых 3-х примеров.

1.1.2 Расчет среднего расположения центров

После распределения всех точек к ближайшему центру, на втором этапе алгоритм находит новые положения центров: среднее положение точек которые к нему принадлежат к конкретному центру. Для каждого центра *k* имеем

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

где C_k набор примеров, которые относятся к центру *k*. Например, если два примера $x^{(3)}$ и $x^{(5)}$ относятся к центру $k = 2$, тогда следует обновить

$$\mu_2 = 1/2(x^{(3)} + x^{(5)}).$$

Задание: Завершить код в программе *computeCentroids.m*. Это можно сделать, например, используя цикл для каждого центра или цикл для каждого примера, но если расчеты будут производиться в векторном виде, программа будет работать значительно быстрее. После завершения кода в *computeCentroids.m*, скрипт *ex4.m* запустит вашу работу и отобразит центры после первого этапа метода К-среднего.

1.2 Метод *k*-средних с набором данных из примера

После завершения работы двух функций (*findClosestCentroids.m* и *computeCentroids.m*) программа *ex4.m* применит метод *k*-средних с 2-мерным набором данных. Функции, которые Вы написали, вызываются из *runkMeans.m*. Для лучшего понимания материала рекомендуется ознакомиться с этим кодом. Обратите внимание на то, что функции вызываются в цикле. Следующий шаг программы - это последовательное отображение каждой итерации алгоритма. Нажмите клавишу *«Enter»* несколько раз, чтобы увидеть изменение расположения центров. Окончательный вариант должен выглядеть так, как представлено на рис. 1.

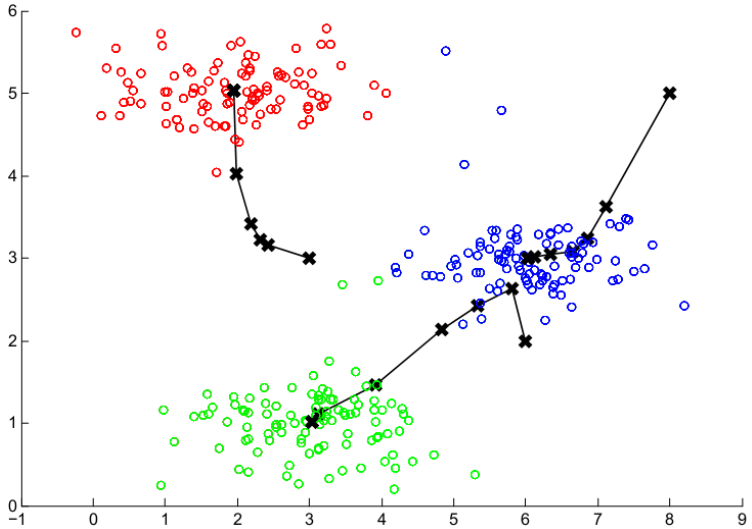

Рис. 1. Ожидаемый результат (10 итераций)

1.3 Случайный запуск

Начальное расположение центров для предыдущего набора данных было выбрано таким образом, чтобы Вы смогли увидеть такую же картинку, как на рис. 1. В действительности, лучшим подходом будет случайный выбор начального расположения центров. В этой части упражнения Вам следует завершить код в *kMeansInitCentroids.m*, добавив следующие строчки:

```
% Случайное задание центров из примеров
% Случайно перегруппировать индексы примеров
randidx = randperm(size(X, 1));
% Взять первые K примеров в качестве центров
centroids = X(randidx(1:K), :);
```

Здесь случайным образом перемешиваются индексы примеров (используя *randperm*) и выбираются первые *K* примеров в качестве центров.

Указание: функция *p = randperm(n)* выполняет случайную перестановку целых чисел от 1 до n. Это позволяет случайно выбрать данные без риска совпадения для двух центров.

1.4 Сжатие изображений, используя метод *k*-средних

Рис. 2. Исходное изображение 128 x 128

Используйте метод *k*-средних для сжатия изображения. В обычном, 24-х битовом цветовом представлении каждый пиксель соответствует трем 8-ми битным целым числам (в интервале от 0 до 255) - красный, зеленый, синий. Данную цветовую модель обычно обозначают как *RGB*. Приведенная картинка птицы содержит тысячи цветов и в этой части упражнения предстоит уменьшить их количество до 16, что потребует только 4 бита. Используйте метод *k*-средних для выбора 16-ти цветов, которые будут отображать рассматриваемое изображение. Каждый пиксель представляется как набор данных и метод *k*-средних находит 16 цветов, которые лучше всего группируют пиксели в 3-х мерном *RGB*-пространстве. После нахождения центров Вы замените цвет каждого пикселя исходного изображения на один из 16-ти цветов.

1.4.1 Метод k-средних и пиксели (“элизы”)

В пакете Matlab изображения можно считать следующим образом:

```
% Считать 128x128 цветное изображение (bird_small.png)
A = imread('bird_small.png');
```

```
% Для корректной работы imread необходимо иметь
% установленный компонент по работе с изображениями
% или использовать вместо этого другую команду
% load('bird_small.mat');
% Читает изображение в переменную A
```

Этот код создаст 3-х мерную матрицу A, в которой первые два элемента обозначают местоположение пикселя, а третий обозначает интенсивность красного, зеленого или синего цветов. Например, A(50, 33, 3) обозначает пиксель с координатами 50, 30 и интенсивностью синего цвета 3. Код в программе ex4.m загружает изображение и переделывает ее в матрицу размером $m \times 3$ (где $m = 16384 = 128 \times 128$), и запускает метод *k*-средних.

Задание: После определения первых $K = 16$ цветов для представления изображения, требуется распределить пиксели в зависимости от ближайшего к ним центра, используя функцию *findClosestCentroids*. Это преобразует исходное изображение, используя ближайшие для каждого пикселя центры. Обратите внимание на то, что количество битов, необходимых для хранения изображения значительно уменьшилось. Исходное изображение требовало 24 бита для каждого из 128×128 пикселей, занимая в итоге 128×128×24 = 393.216 битов. Новое отображение требует «словаря» из 16 цветов, занимающих 24 бита каждый и 4 бита для каждого пикселя, получая 16×24 + 128×128×4 = 65.920 битов, что соответствует сжатию исходного изображения примерно в 6 раз.

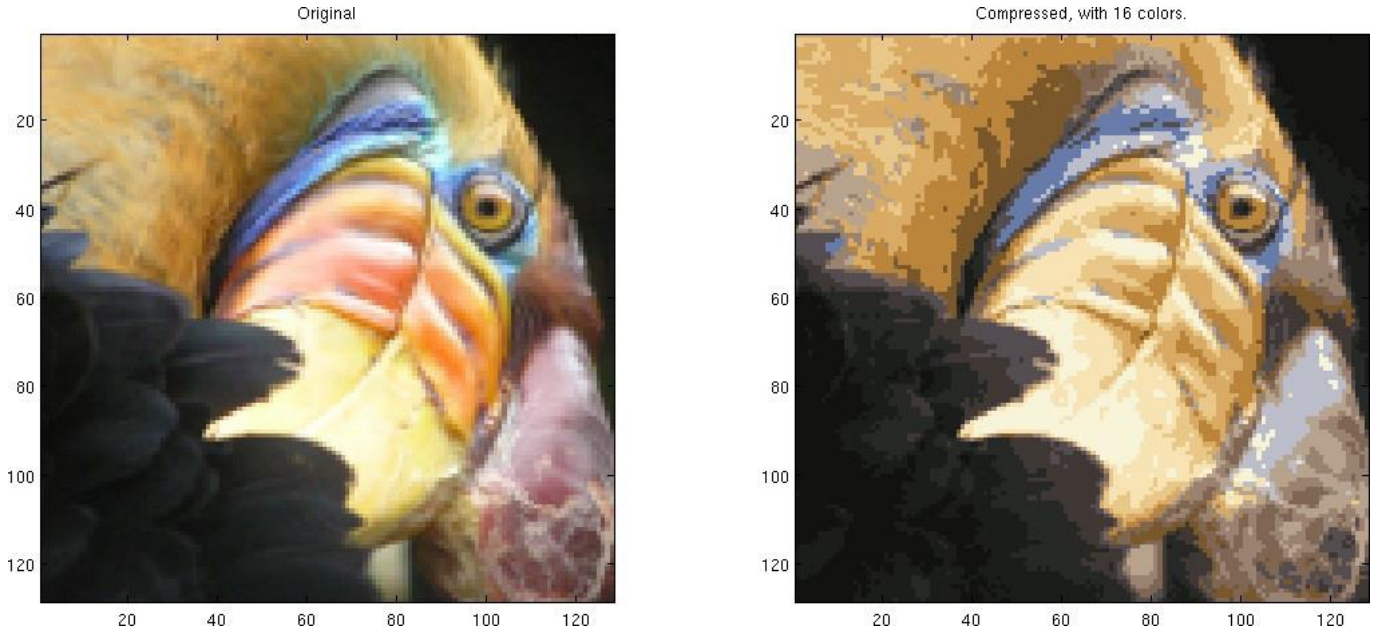

Рис. 3: Исходное и преобразованное изображение (метод *k*-средних).

Результат сжатия с помощью метода *k*-средних можно увидеть на рис. 3. Каждый пиксель можно заменить на среднее значение центра, к которому он относится. Несмотря на то, что изображение сохраняет большинство своих характеристик, последствия сжатия весьма заметны.

1.5 Сжатие собственного изображения

В этом упражнении измените предыдущую программу, применив её для Вашего собственного изображения. Обратите внимание на то, что если изображение имеет большой размер, то для обработки может потребоваться достаточно долгое время. Рекомендуется вначале его уменьшить. Вы также можете использовать различное количество центров и посмотреть на разницу при сжатии.