1 Кластеризация

Задача кластеризации заключается в нахождении оптимального разбиения

$$\mathcal{X}^k(\mathbb{X}) = \{\mathbb{X}_1, \dots, \mathbb{X}_k\}$$

множества $\mathbb{X}$ на k непустых кластеров, таких что

$$(\mathbb{X}) = \bigcup_{i=1}^{k} \mathbb{X}_i$$

и

$$\mathbb{X}_i \cap \mathbb{X}_j = \emptyset, i \neq j.$$

Для достижения такого разбиения необходимо, чтобы выполнялось следующее свойство: элементы, принадлежащие одной группе, являются более “близкими”, чем элементы из разных групп. Для этого вводится некоторая функция различия (качества) q_i, определяющая “близость” для каждого кластера $\mathbb{X}_i, i \in 1..k$. Тогда задача кластеризации сводится к минимизации функционала:

$$F(\mathcal{X}^k) = Ef(\mathcal{X}^k, \mathbf{x}) = \sum_{i=1}^{k} \int_{\mathbb{X}} q_i(\mathcal{X}^k, \mathbf{x}) P(d\mathbf{x}) \to \min_{\mathcal{X}^k}. \tag{1}$$

Будем считать, что разбиение $\mathcal{X}_\|$ полностью определяется k векторами $x_1, x_2, \dots, x_k \in \mathbb{R}^d$, которые образуют $d \times k$ матрицу $\mathbf{X} = (x_1, x_2, \dots, x_k)$, и для $i \in 1..k$ и фиксированного $x \in \mathbb{X}$ каждая функция $q_i(\cdot, \mathbf{x})$ зависит только от x_i, т.е. $q_i(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{X} \to \mathbb{R}$. Тогда правило разбиения на кластеры задается следующим образом

$$\mathbb{X}_i(\mathbf{X}) = \{\mathbf{x} \in \mathbb{X} : q_i(x_i, \mathbf{x}) \leq q_j(x_j, \mathbf{x}), j = 1, \dots, k\}, \forall i \in 1..k,$$

что минимизирует 1.

Вектора x_i, $i \in 1..k$, обычно интерпретируются как *центры кластеров* (или *центроиды*), и $\mathbb{X}$ рассматривается как подмножество в Евклидовом пространстве $\mathbb{R}^d$. В этом случае функционал 1 принимает вид

$$F(\mathcal{X}) = \sum_{i=1}^{k} \int_{\mathbb{X}} q_i(x_i, \mathbf{x}) P(d\mathbf{x}) \to \min_{\mathcal{X}^k} \tag{2}$$

Рассмотрим стандартный случай, когда распределение $P(\cdot)$ является равномерным на $\mathbb{X}$ и функция $q_i(\cdot, \cdot)$ имеет вид

$$q_i(x_i, \mathbf{x}) = ||x_i - \mathbf{w}||^2, \quad i \in 1..k,$$

задающее Евклидово расстояния до центров $x_1, x_2, \dots, x_k$. Тогда функционал 2 записывается как

$$F(\mathbb{X}) = ||x_i - x_j||^2 \to \min_{\mathbb{X}} \tag{3}$$

Алгоритм K-средних

Рассмотрим один из самых распространенных алгоритмов кластеризации *алгоритм K-средних (K-means)*:

Вход: $\mathbb{X}$ – множество, которое необходимо разбить на кластеры, k – число кластеров, максимальное число итераций.

Выход: центры кластеров $\hat{\mathbb{X}}$ и разбиение $\mathcal{X}^k$ множества $\mathbb{X}$ на k кластеров.

1. *Инициализация:* Случайным образом из элементов множества $\mathbb{X}$ выбирается k начальных центров кластеров $\hat{\mathbb{X}}^0 = (\hat{x}_1^0, \hat{x}_2^0, \ldots, \hat{x}_k^0)$

2. *Классификация:* На итерации n для каждой точки данных $\mathbf{x}$ вычисляется значение $||\hat{x}_i^n - \mathbf{x}||^2, \quad i \in 1..k$, на основе которого $\mathbf{x}$ относится к одному из k кластеров:

$$\mathbb{X}_i^n = \{\mathbf{x} \in \mathbb{X} : ||\hat{x}_i^n - \mathbf{x}||^2 \leq ||\hat{x}_j^n - \mathbf{x}||^2, \quad j = 1, \ldots, k\}, \quad i \in 1..k.$$

3. *Минимизация:* На итерации n вычисляются средние значения (центроиды) для каждого i-го кластера $i \in 1..k$:

$$\hat{x}_u^{n+1} = \frac{1}{|\mathbb{X}_i^n|} \sum_{\mathbf{x_j} \in \mathbb{X}_{\sqcup}^{\ltimes}} \mathbf{x}_j$$

4. Шаги 2 и 3 повторяются пока центроиды не перестанут меняться или не будет достигнуто максимальное число итераций

Проблема локального минимума

Описанный выше алгоритм является достаточно эффективным, но гарантирует сходимость только к локальному минимуму функционала 3. Для того, чтобы результат получался оптимальным существуют различные подходы. Будем использовать подход, основанный на внутрикластерных расстояниях: алгоритм K-средних запускается заданное количество раз, и выбирается результат, у которого наименьшее значение функционала 3

2 Задание

Файл с данными

Задание заключается в разбиении изображения на области, близкие по цветовому наполнению с помощью алгоритма кластеризации K-средних. Такой подход аналогичен простейшему “сжатию” изображения.

3 Решение задания

Алгоритм K-средних

Необходимо реализовать описанный выше алгоритм K-средних с выбором наилучшего результата для решения проблемы локального минимума. На вход алгоритму должна поступать матрица $\mathbb{X}$, описанная в предыдущем пункте, число кластеров, максимальное число итерации (обычно 100-200), число повторных запусков для решения проблемы локального минимума. Возвращаться должны центроиды и метки кластеров $(1, 2, \ldots, k)$ лдя каждого элемента множества $\mathbb{X}$

"Сжатие" изображения

Каждый пиксель изображения можно представить как точку в трехмерном цветовом пространстве RGB: $\mathbf{x}_i = (x_r, x_g, x_b)$, где x_r – значение Red для i-ого пикселя, x_g – значение Green для i-ого пикселя, x_b – значение Blue для i-ого пикселя, т.е. $\mathbb{X} \in \mathbb{R}^3$.Таким образом получается матрица входных данных $\mathbb{X}$

4 Содержание ответа

Для получения зачета по этому заданию необходимо на адрес natalia.kizhaeva@gmail.com прислать следующее

- Три изображения, полученных для трех разных чисел кластеров k.
- Краткое описание того, что получилось
- Код всей программы, содержащий реализацию алгоритма и его применение

В результате Вы получите сообщение о том, что задание зачтено или замечания, которые нужно будет устранить.

Код

Чтение изображения из файла и преобразования его в удобный для использования формат может создать определенные трудности. Ниже приведен код на Python и R, иллюстрирующий возможный способ чтения изображения, его преобразования в матрицу и применение результатов работы алгоритма и сохранение обработанного изображения в файл. Можно использовать этот код, вместо new_X или new.X должны стоять матрицы, в которых строки заменены значениями соответствующих центроидов.

Код на Python

```
from PIL import Image
import numpy

input_image_file = 'input_image_file.jpg'
output_image_file = 'output_image_file.jpg'

image = numpy.array(Image.open(input_image_file))
X = image.reshape((image.shape[0] * image.shape[1], image.shape[2]))

new_X = ...

new_image = new_X.reshape(image.shape)
Image.fromarray(new_image).save(output_image_file)
```

Код на R

```
library(jpeg)

input.image.file <- "input_image_file.jpg"
output.image.file <- "output_image_file.jpg"

image <- readJPEG(input.image.file)
X <- array(image, dim=c(dim(image)[1] * dim(image)[2], dim(image)[3]))

new.X <- ...

new.image <- array(new.X, dim(image))
writeJPEG(new.image, output.image.file)
```