**Методические указания к лабораторной работе №4**

В данной работе мы продолжаем работать с библиотекой scikit-learn (<http://scikit-learn.org>), и рассмотрим ее возможности для решения задачи кластеризации данных.

Ниже приведены новые модули, которые будут использованы в данной работе.

euclidean\_distances - <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.euclidean_distances.html> - евклидово расстояние между парой векторов.

KMeans - <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> – Метод k-средних

Кроме того, будут использованы следующие модули из библиотеки Scipy (<https://scipy.org>)

Linkage - <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html> - Проведение иерархической кластеризации.

Dendrogram - <https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.cluster.hierarchy.dendrogram.html> - построение дендрограмм

fcluster - <https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.cluster.hierarchy.fcluster.html> - формирование кластеров из матрицы связей

Для вычисления критериев качества кластеризации могут быть полезными следующие методы массивов *numpy:*

shape - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html> – возвращает размерность массива

reshape - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html#numpy.reshape> – изменение размерности массива

mean - <https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.mean.html> - возвращает среднее значение элементов массива

where - <https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.where.html> - возвращает массив элементов, удовлетворяющих заданным условиям.

**Кластеризация данных**

Загрузка новых модулей может выглядеть следующим образом:

|  |
| --- |
| from sklearn.metrics.pairwise import euclidean\_distancesfrom scipy.cluster.hierarchy import linkage, dendrogram, fclusterfrom sklearn.cluster import KMeans |

Кроме того, необходимо подгрузить модуль pyplot для построения графиков и модуль numpy для обработки данных.

**Иерархическая кластеризация**

Для проведения иерархической кластеризации в данной работе мы воспользуемся модулем *linkage* из библиотеки *scipy*. В качестве параметров необходимо указать выборку данных и параметр *method* - метод вычисления расстояния между кластерами. На выходе модуль возвращает матрицу расстояний между всеми объектами. Например, для метода дальнего соседа:

|  |
| --- |
| mergings = linkage(X, method='complete') |

Теперь, в переменной *mergings* хранится матрица расстояний. Чтобы построить по ней дендрограмму, воспользуемся модулем *dendrogram*, обязательным параметром которого является матрица расстояний, и отобразим дендрограмму на графике:

|  |
| --- |
| dendrogram(mergings)plt.show() |

Для того, чтобы разбить объекты на кластеры согласно матрице расстояний, воспользуемся модулем *fcluster*, передав матрицу расстояний, порог и критерий формирования кластеров. Например, при указании критерия *‘distance’* и порога = *10*, данные будут разбиты на кластеры, расстояние между объектами которых не превышает 10:

|  |
| --- |
| T = fcluster (mergings, 10, 'distance')print (T) |

При указании критерия *‘maxclust’* и порога = 4, данные будут разбиты не более чем на 4 кластера.

Результатом будет массив длиной равной количеству объектов выборки, каждый элемент которого представляет собой номер кластера, к которому объект относится.

**Оценка качества кластеризации**

Для того чтобы оценить качество кластеризации, предлагается рассчитать такие функционалы качества, как **средняя** сумма квадратов расстояний до центра, **средняя** сумма средних внутрикластерных расстояний, средняя сумма межкластерных расстояний:

Сумма квадратов расстояний до центроида (inertia):



Сумма средних внутрикластерных расстояний:



Сумма межкластерных расстояний:



Необходимо обратить внимание, что для получения средних значений данных функционалов, их необходимо разделить на количество кластеров.

Для расчета данных характеристик предварительно необходимо рассчитать центроиды каждого кластера, для чего предлагается следующая функция, позволяющая рассчитать координаты центроидов при условии наличия двух кластеров:

|  |
| --- |
| import numpy as npdef update\_cluster\_centers(X, c): ix = np.where(c==1) mu[0,:] = np.mean(X[ix,:], axis=1) ix = np.where(c==2) mu[1,:] = np.mean(X[ix,:], axis=1) return mu |

На вход функции подается два массива одинаковой длины: *X* – исходная выборка объектов, *с* – номер кластера каждого объекта из выборки. При необходимости расчета центроидов более чем двух кластеров, предложенную функцию необходимо будет модифицировать.

Вывести на экран координаты центроидов, рассчитанные с помощью предложенной функции, можно следующим образом:

|  |
| --- |
| mu = np.array([[0.0,0], [0,0]])mu = update\_cluster\_centers(X, T)print(mu) |

Для дальнейшего расчета функционалов качества удобно использовать функции *shape, reshape, mean, where* библиотеки *numpy*, а также функцией *euclidean\_distances* из *sklearn*

**K-means**

Для кластеризации методом k-средних с помощью модулей библиотеки sklearn используется алгоритм, схожий с алгоритмом проведения классификации данных: создаем модель кластеризации с помощью модуля KMeans(), указав предполагаемое количество кластеров, настраиваем ее на выборке методом *fit()*. После этого, с помощью метода *predict()* можно записать результаты кластеризации в переменную. Метод *predict()* возвращает массив длиной равной количеству объектов выборки, каждый элемент которого представляет собой номер кластера, к которому объект относится.

|  |
| --- |
| kmeans = KMeans(n\_clusters=2)kmeans.fit(X)all\_predictions = kmeans.predict(X)print (all\_predictions) |

С помощью метода *inertia\_* можновывести значение суммы квадратов расстояний до центроида, а координаты центроидов – с помощью метода *cluster\_centers\_.*