НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Анализ данных»

Домашнее задание на тему:

«Лабораторная работа №2»

Выполнил: Осипов Лев,

студент группы 301ПИ (1).

Москва, 2015 г.

**СОДЕРЖАНИЕ**

**Теоретическая часть3**

**Задание 13**

**Задание 23**

**Задание 34**

**Практическая часть4**

**Задание 14**

**Задание 25**

**Задание 35**

**Список литературы7**

**Текст программы8**

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**ЗАДАНИЕ 1**

Для применения метода ближайшего соседа к данным с номинальными признаками можно представить каждый признак как значение из множества {0,1} или же {0,n}, если используем взвешенные признаки. С таким подходом каждый признак будет представлять собой измерение.

**ЗАДАНИЕ 2**

Да, проблема возникнет. Дело в том, что будут считаться равнозначными равные по номиналу значения пульса и веса, которые измеряются соответственно в ударах в минуту и граммах. Для того чтобы этого избежать, можно привести все к одной единице измерения с помощью коэффициента, который можно получить, например, поделив длины интервалов значений признаков.

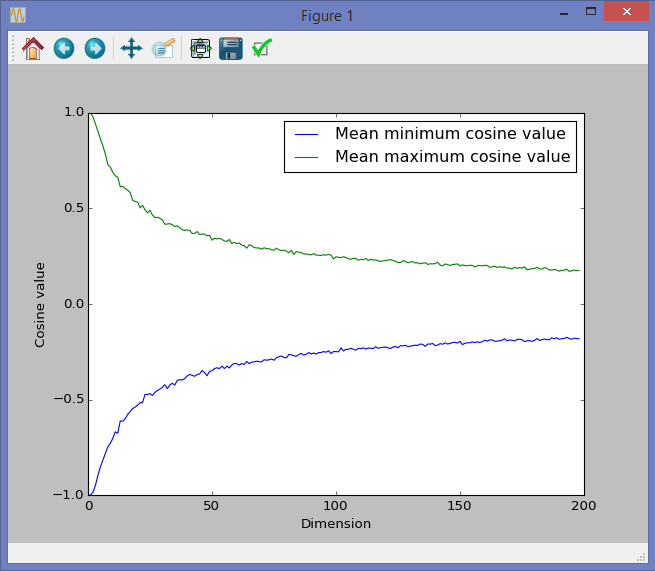
**ЗАДАНИЕ 3**

Сложность будет O(ns), так как расстояния для нулевых признаков считать не нужно (расстояние будет квадратом входного значения). Итерирование происходит только по s признаков каждого объекта.

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

**ЗАДАНИЕ 1**

Для решения задания были посчитаны средние максимальные и минимальные значений косинусов углов между случайно сгенерированных точками разных размерностей.

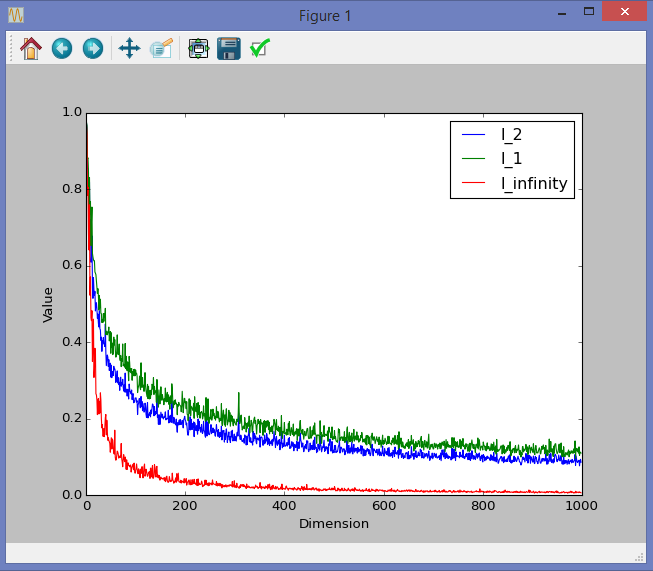
****

*Рис. 1*

По результатам программы (Рис. 1) видно, что средние с увеличением размерности стремятся к нулю, что как раз и говорит о том, что все точки становятся «практически ортогональны».

**ЗАДАНИЕ 2**

Для решения задания были посчитаны значения для случайно сгенерированных точек разных размерностей и для каждого из значений был произведен расчет по следующей формуле:

****

*Рис. 2*

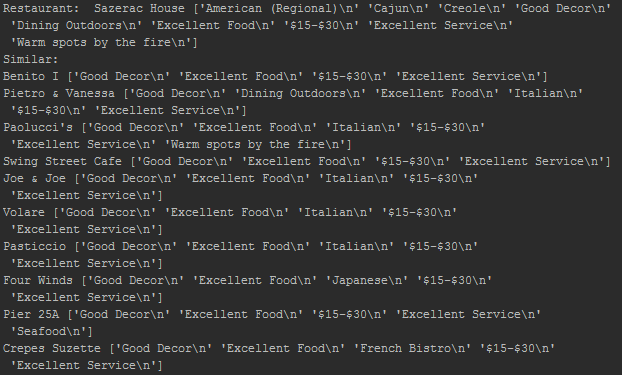
Если считать, что каждое из значений признаков независимо относительно выборки координат определенной точки и все значения имеют одинаковое распределение, то можно увидеть тенденцию стремления значений норм к одному значению с ростом размерности пространства. Именно поэтому величина, полученная при расчете, с увеличением размерности стремится к нулю, что можно наблюдать на графике, созданном программой (Рис. 2).

Из вышесказанного можно сделать вывод, что с увеличением размерности пространства расстояние между точками становится неинформативным.

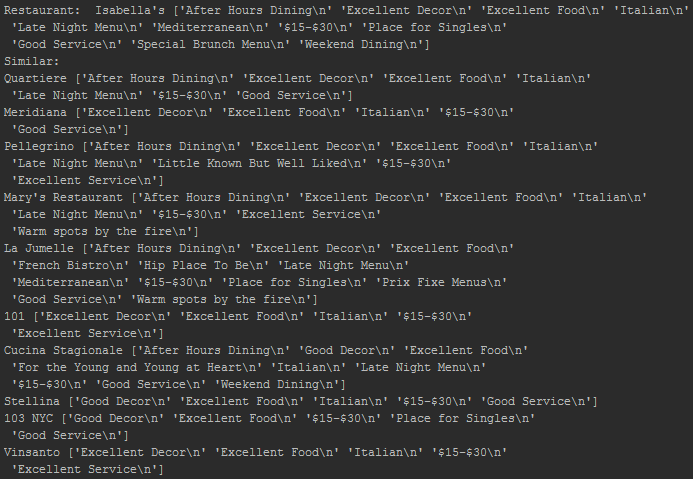
**ЗАДАНИЕ 3**

Для решения задания была написана программа поиска похожих ресторанов, основанная на методе ближайшего соседа.

Примеры работы программы представлены на Рис. 3, Рис. 4 и Рис. 5.

****

*Рис. 3. Результат работы программs на ресторане «Sazerac House».*

****

*Рис. 4. Результат работы программы на ресторане «Isabella’s».*

****

*Рис. 5. Результат работы программы на ресторане «Lucky Strike».*

**СПИСОК ЛИТЕРАТУРЫ**

1. **Анализ данных (Программная инженерия)** – http://wiki.cs.hse.ru/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7\_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85\_%28%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F\_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F%29#.D0.9E.D1.84.D0.BE.D1.80.D0.BC.D0.BB.D0.B5.D0.BD.D0.B8.D0.B5\_.D0.BF.D0.B8.D1.81.D0.B5.D0.BC

**ТЕКСТ ПРОГРАММЫ**

\_\_author\_\_ = 'Lev Osipov'  
  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** heapq **import** nsmallest  
  
  
**def cosine**(vector1, vector2):  
 **return** np.dot(vector1, vector2) / (np.linalg.norm(vector1) \* np.linalg.norm(vector2))  
  
# Task 1  
points = 100  
dimension = 200  
coordinates = []  
results = []  
**print** "Wait, please.."  
**for** dim **in** xrange(1, dimension):  
 coordinates.append(np.random.rand(points, dim) \* 2 - 1)  
 dim\_results = []  
 **for** i **in** xrange(points):  
 cos = []  
 **for** j **in** xrange(points):  
 **if** i != j:  
 cos.append(cosine(coordinates[dim - 1][i], coordinates[dim - 1][j]))  
 dim\_results.append([np.min(cos), np.max(cos)])  
 results.append(np.mean(dim\_results, axis=0))  
np\_results = np.array(results)  
plt.plot(np\_results[:, 0], label="Mean minimum cosine value")  
plt.plot(np\_results[:, 1], label="Mean maximum cosine value")  
plt.xlabel('Dimension')  
plt.ylabel('Cosine value')  
plt.legend()  
plt.show()  
  
# Task 2  
points = 1000  
dimension = 1000  
coordinates = []  
function = []  
**print** "Wait, please.."  
**for** dim **in** xrange(1, dimension):  
 coordinates.append(np.random.rand(points, dim) \* 2 - 1)  
 l2 = []  
 l1 = []  
 l\_inf = []  
 **for** i **in** xrange(points):  
 l2.append(np.linalg.norm(coordinates[dim - 1][i]))  
 l1.append(np.linalg.norm(coordinates[dim - 1][i], 1))  
 l\_inf.append(np.linalg.norm(coordinates[dim - 1][i], np.inf))  
 l2\_max = np.max(l2)  
 l1\_max = np.max(l1)  
 l\_inf\_max = np.max(l\_inf)  
 l2\_min = np.min(l2)  
 l1\_min = np.min(l1)  
 l\_inf\_min = np.min(l\_inf)  
 function.append([(l2\_max - l2\_min) / l2\_max, (l1\_max - l1\_min) / l1\_max, (l\_inf\_max - l\_inf\_min) / l\_inf\_max])  
np\_function = np.array(function)  
plt.plot(np\_function[:, 0], label="l\_2")  
plt.plot(np\_function[:, 1], label="l\_1")  
plt.plot(np\_function[:, 2], label="l\_infinity")  
plt.xlabel('Dimension')  
plt.ylabel('Value')  
plt.legend()  
plt.show()  
  
# Task 3  
features\_file = open("features.txt")  
feature\_list = []  
**for** feature **in** features\_file:  
 id, name = feature.split('\t')  
 feature\_list.append(name)  
features\_file.close()  
  
restaurants\_file = open("new\_york.txt")  
restaurants = {} # dictionary  
**for** restaurant **in** restaurants\_file:  
 id, name, features = restaurant.split('\t')  
 current\_features = np.zeros(len(feature\_list))  
 current\_features[[int(current\_feature) **for** current\_feature **in** features.split()]] = 1 # matching existing features  
 restaurants[name] = current\_features  
restaurants\_file.close()  
  
fl = np.array(feature\_list)  
  
  
**def find\_similar**(restaurant\_name, all\_restaurants, features\_array):  
 cf = all\_restaurants[restaurant\_name] # current features  
 **del** all\_restaurants[restaurant\_name] # deleting current restaurant  
 similar\_ones = nsmallest(10, all\_restaurants.items(), **lambda** item: np.linalg.norm(item[1] - cf)) # finding  
 **print** "Restaurant: ", restaurant\_name, features\_array[np.nonzero(cf)]  
 **print** "Similar:"  
 **for** similar **in** similar\_ones:  
 **print** similar[0], features\_array[np.nonzero(similar[1])]  
  
  
# Tests  
find\_similar("Sazerac House", restaurants.copy(), fl)  
find\_similar("Isabella's", restaurants.copy(), fl)  
find\_similar("Lucky Strike", restaurants.copy(), fl)