РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

дисциплина: Интеллектуальные системы

Студент: Койфман К.Д.

Группа: НПИбд-01-21

№ ст. билета: 1032217058

МОСКВА

2022 г.

Введение.

Цель работы.

Изучение принципов работы и реализация алгоритма распознавания образов.

Задачи.

1. Ознакомиться с теоретическим материалом по работе алгоритма распознавания образов.

2. Реализовать данный алгоритм, используя язык программирования C++.

3. Протестировать работу алгоритма.

1 задание.

Теоретическая часть была изучена и разобрана с использованием сторонних материалов и интернет-ресурсов [1,2].

2 задание.

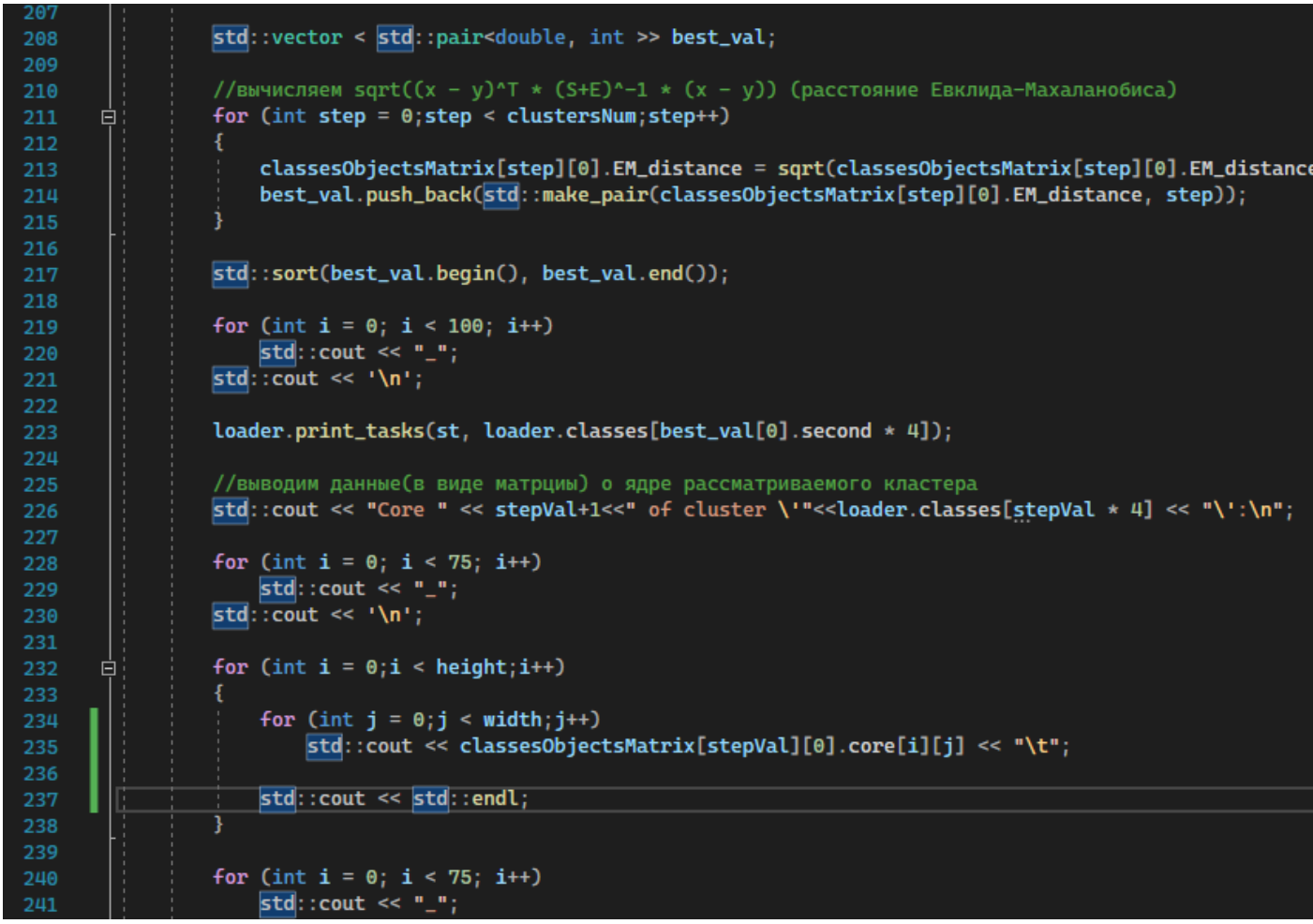
Реализуем алгоритм, используя приобретённые знания и высокоуровневый язык программирования C++(рис.1, рис.2, рис.3):

```
9
10  struct Cluster
11  {
12      std::vector<std::vector<double>> coordinates; // координата образа
13      std::vector<std::vector<double>> coordinates_diff; // разность координат объекта и ядра класса
14      std::vector<std::vector<double>> covariational_matrix; // матрица ковариации
15      std::vector<std::vector<double>> ucS; // (x-y)^T * (S+E)^(-1) (неполное значенеи EM_distance)
16      std::vector<std::vector<double>> core; // ядро кластера(класса)
17      double EM_distance = 0.0;// (x-y)^T * (S+E)^(-1) * (x-y) - расстояние Евклида-Махаланобиса
18  };
19
20  //функция вычисляет часть расстояния Евклида-Махаланобиса (S+E)^-1
21  void countEM(std::vector<std::vector<double>>& M)
22  {
23      double temp;
24
25      //объявляем единичную матрицу
26      std::vector<std::vector<float>> E(M.size(), std::vector<float>(M.size(), 0.0));
27
28      //определяем единичную матрицу E и складываем с матрицей ковариации S
29      for (int i = 0; i < M.size(); i++)
30          for (int j = 0; j < M.size(); j++)
31          {
32              if (i == j)  E[i][j] = 1.0;
33              M[i][j] += E[i][j];
34          }
35
36      for (int a = 0; a < M.size(); a++)
37      {
38          temp = M[a][a];
39          //все элементы a-ой строки матрицы M, кроме a-ого и до него, делим на разрешающий элемент t матрицы M(математического ожидания)
40          for (int b = a + 1; b < M.size(); b++)
41              M[a][b] = M[a][b] / temp;
42
43          //все элементы a-ой строки единичной матрицы E делим на разрешающий элемент t матрицы M(математичсекого ожидания)
44          for (int b = 0; b < M.size(); b++)
45              E[a][b] = E[a][b] / temp;
```

РИС.1

```
80
81  int main(int argc, char* argv[]) //argc - argumnet counter, argv - argument values
82  {
83      //произодим загрузку входных данных( данные по кластерам(классам), образцам(шаблонам) кластеров(классов) и рассматриваемых объетам)
84      for(int i=0; i<argc; i++)
85          std::cout<<argv[i]<<"\n";
86      if(argc<2)
87      {
88          std::cout << "Name of the input XML file is not specified."<<std::endl;
89          return 1;
90      }
91      Loader loader;
92      loader.load_instance(argv[1]);
93      loader.print_examples();
94
95      //определяем число..
96      int clustersNum = 4;//кластеров( классов) объектов
97      int clusterPatternsNum = 4;//образов(шаблонов) в каждом кластере(классе)
98      int reviewedObjectsNum = 4;//рассматривемых объектов
99
100     // формируем двумерный массив классов-объектов
101     std::vector<std::vector<Cluster>> classesObjectsMatrix(clustersNum, std::vector<Cluster>(clusterPatternsNum));
102
103     //устанавливаем размеры матрицы для каждого объекта( как для образов(шаблонов) классов(кластеров), так и для рассматриваемых объект
104     int height = 10, width = 10;
```

РИС.2



```
207
208         std::vector < std::pair<double, int >> best_val;
209
210         //вычисляем sqrt((x - y)^T * (S+E)^-1 * (x - y)) (расстояние Евклида-Махаланобиса)
211         for (int step = 0;step < clustersNum;step++)
212         {
213             classesObjectsMatrix[step][0].EM_distance = sqrt(classesObjectsMatrix[step][0].EM_distance
214             best_val.push_back(std::make_pair(classesObjectsMatrix[step][0].EM_distance, step));
215         }
216
217         std::sort(best_val.begin(), best_val.end());
218
219         for (int i = 0; i < 100; i++)
220             std::cout << "_";
221         std::cout << '\n';
222
223         loader.print_tasks(st, loader.classes[best_val[0].second * 4]);
224
225         //выводим данные(в виде матрцицы) о ядре рассматриваемого кластера
226         std::cout << "Core " << stepVal+1<<" of cluster \'"<<loader.classes[stepVal * 4] << "\':\n";
227
228         for (int i = 0; i < 75; i++)
229             std::cout << "_";
230         std::cout << '\n';
231
232         for (int i = 0;i < height;i++)
233         {
234             for (int j = 0;j < width;j++)
235                 std::cout << classesObjectsMatrix[stepVal][0].core[i][j] << "\t";
236
237             std::cout << std::endl;
238         }
239
240         for (int i = 0; i < 75; i++)
241             std::cout << "_";
```

РИС.3

3 задание.

Теперь протестируем его работу на 4 кластерах с уникальным образом в каждом, 4-мя образцами в каждом кластере и 4-мя рассматриваемыми объектами, класс каждого из которых наш алгоритм должен будет определить (рис.5, рис.6, рис.7, рис.8). Но для начала посмотрим на образцовые объекты каждого из классов (рис.4):

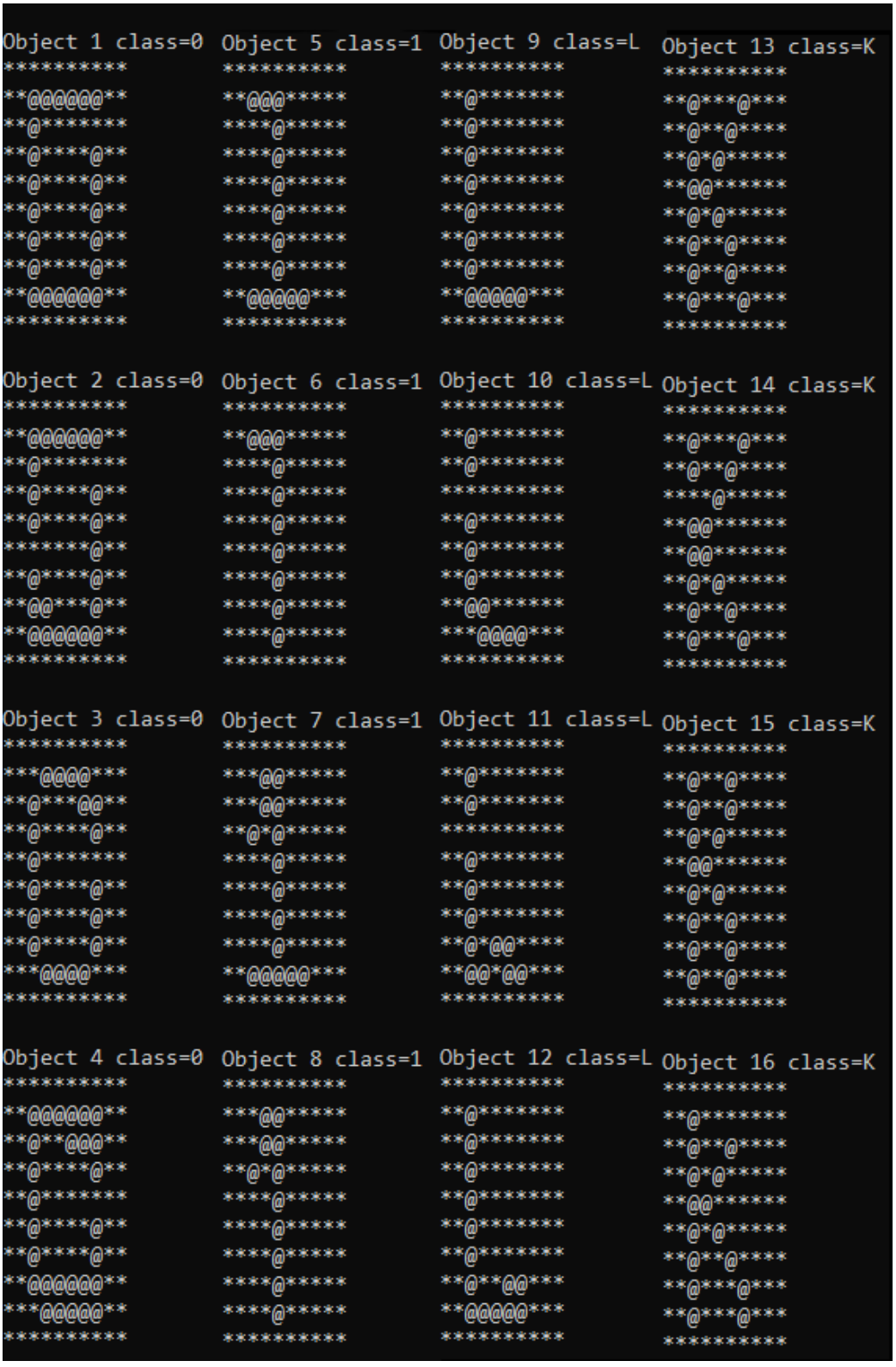


РИС.4(классы символов ‘0’, ‘1’, ‘L’ и ‘K’, представленные их бинарными матричными изображениями)

```
Task 1 supposed class = K
**********
**@**@****
*****@****
**@*@*****
**@@******
**@@@*****
**@**@****
**@*******
**@**@****
**********
Core 1 of cluster '0':
```

0	0	0	0	0	0	0	0	0	0
0	0	0.75	1	1	1	1	0.75	0	0
0	0	1	0	0	0.25	0.5	0.5	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0.5	0	0
0	0	0.75	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0.5	0.25	0.25	0.25	1	0	0
0	0	0.5	1	1	1	1	0.75	0	0
0	0	0	0	0	0	0	0	0	0

```
Distance of cluster '0' : 4.38748
Distance of cluster '1' : 4.21307
Distance of cluster 'L' : 3.36341
Distance of cluster 'K' : 2.09165
```

РИС.5

```
Task 2 supposed class = 1
**********
***@******
****@*****
****@*****
****@*****
****@*****
****@*****
***@@@****
**********
Core 2 of cluster '1':
```

0	0	0	0	0	0	0	0	0	0
0	0	0.5	1	1	0	0	0	0	0
0	0	0	0.5	1	0	0	0	0	0
0	0	0.5	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0.5	0.5	1	0.5	0.5	0	0	0
0	0	0	0	0	0	0	0	0	0

```
Distance of cluster '0' : 5.06335
Distance of cluster '1' : 2.44194
Distance of cluster 'L' : 4.37903
Distance of cluster 'K' : 4.68686
```

РИС.6

```
Task 3 supposed class = 0
**********
**@@@@@***
**@****@**
**@****@**
**@****@**
**@****@**
**@****@**
**@****@**
***@@@@@**
**********
Core 3 of cluster 'L':
```

0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0.5	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0.25	0.25	0.5	0.25	0	0	0
0	0	0.75	1	0.75	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

```
Distance of cluster '0' : 2.70432
Distance of cluster '1' : 4.91853
Distance of cluster 'L' : 4.08553
Distance of cluster 'K' : 4.80228
```

РИС.7

```
Task 4 supposed class = L
**********
**@*******
**@*******
**@*******
**@*******
**@*******
**********
**@***@***
**@@@@****
**********
Core 4 of cluster 'K':
```

0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0.25	0.5	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	0.75	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	0.25	0.75	0	0	0	0	0
0	0	1	0	0.25	0.75	0	0	0	0
0	0	1	0	0	0.75	0.25	0	0	0
0	0	1	0	0	0.25	0.75	0	0	0
0	0	0	0	0	0	0	0	0	0

```
Distance of cluster '0' : 3.99429
Distance of cluster '1' : 4.43492
Distance of cluster 'L' : 2.71993
Distance of cluster 'K' : 3.83109
```

РИС.8

Исходя из полученных результатов, которые можно увидеть на рис.5, рис.6, рис.7 и рис.8 алгоритм успешно с максимальной точностью определяет, к какому классу относится тот или иной объект. Об этом свидетельствуют значительные разницы показателей “Distance of cluster” у каждого соответствующего класса и рассматриваемого объекта.

Однако важно заметить, что алгоритм может совершать погрешности и ошибки при недостаточно точном и чётком описании каждого отдельного класса при помощи его экземпляров (например, если описать символы ‘2’ и ‘3’, но при этом намеренно или нет сделать их экземпляры слишком подобными, то это может привести к некорректным результатам работы алгоритма. То есть он отнесёт, например, символ ‘2’ к классу ‘3’ или наоборот).

Заключение.

В ходе проделанной лабораторной работы мной были усвоены принципы работы алгоритма для распознавания образов.

Источники.

[1] https://translated.turbopages.org/proxy_u/en-ru.ru.b4c78c42-6359cd4f-52636cf1-74722d776562/https/en.wikipedia.org/wiki/Pattern_recognitionя (turbopages.org)

[2] (miighttps://www.miigaik.ru/vtiaoai/tutorials/19.pdf?ysclid=l9qbh947v8836698347aik.ru)