Изкуствен интелект
Домашно №2

Вариант 2
Клъстеризация с помощта на метода k-Means.

Ростислав Стоянов
ф-н 45244

1. Описание на използвания метод за решаване на задачата

За решаване на задачата е използван методът за обучение без учител k-Means. За целта първо се прочита входящата информация от приложения към условието на задачата файл iris.csv, като за входни данни се използват само атрибутите sepal length, sepal width, petal length и petal width. Всеки ред разглеждаме като отделна точка от четиримерно пространство и целим да ги групираме в три клъстера. След прочитането на файла се създават първите центроиди - в случая се избират реални точки от тези, които вече имаме. Следващата стъпка е да приобщим всяка точка към някой клъстер - пресмята се разстоянието от всеки центроид до съответната точка и точката става част от този клъстър, разстоянието до чиито центроид е най - малко. Продължаваме, като пресмятаме новите центроиди - те са средни на всички точки, които съставят даден клъстър, т.е всяка координата на центроидите се пресмята по следния начин: $\frac{(\sum_{i=1}^{n} a_i)}{n}$, където $_i$ е съответната координата в i - тия вектор, а n е броят на всички вектори в дадения клъстер. Тук се прави проверка дали центроидите са се изменили. В случай, че не са алгоритъмът прекратява своята работа и клъстеризацията е завършена. Ако са се изменили, то се повтаря цикълът на приобщаване на точки и пресмятане на нови центроиди, докато не се окаже, че центроидите са останали същите.

За пресмятане на разстоянието се използва т.нар. разстояние на Минковски, т.е. разстоянието между две точки X = (x1,x2,x3,...,xn) и Y = (y1,y2,y3,...,yn) $\in \mathbb{R}^n$ се пресмята по формулата: $D(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{(1/p)}$. В частност се използват случаите, когато p = 1,p = 2 и p = 4, т.е. използваме евклидово и манхатаново разстояние.

2. Псевдокод и описание на реализацията

Следва псевдокод на основните части от решението на задачата.

```
1: procedure kMeans ()
2:     centroids ← initializeCentroids()
3:     while true do
4:         assignToClusters()
5:         recalculateCentroids()
6:         if endCriteriaReached() then
7:             break
8:     printAns
```

Това е ключовата част на алгоритъма. В самата реализация центроидите получават своята начална стойност при създаването на класа KMeans - това са реални точки от подаденото входно множество.

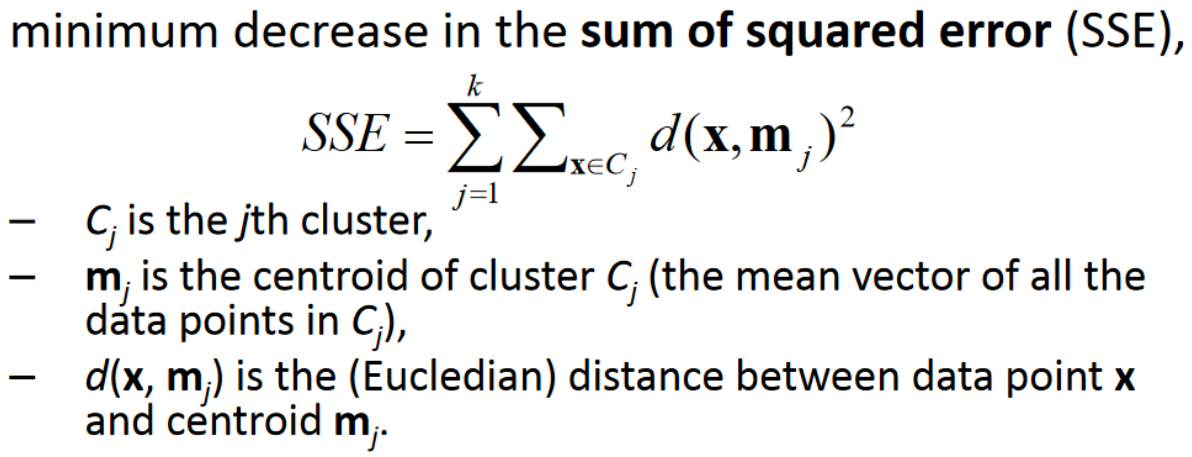
```
1: procedure assignToClusters ()
2:     distances[3] ← {}
3:     for item in items do
4:         for i in 1 : 3 do
5:             distances ← calculateDistance(item,centroid[i])
6:         minDistIdx ← getSmallestIndex(distances)
7:         item.setCluster(minDistIdx)
```

Функцията assignToClusters се използва за да присвои на всяка точка центроид. За целта се пресмята разстоянието от дадената точка до всеки от текущите центроиди, като за това се използва функцията calculateDistance. В реализацията за пресмятане на дистанция се използват евклидово и манхатаново разстояние, както и разстояние на Минковски с p = 4. Функцията getSmallestIndex ни връща числен идентификатор на най - близкия клъстер и на съответната точка се присвоява центроид, за което се използва функцията setCluster.

```
1: procedure recalculateCentroids ()
2:     clusterSize[3] ← {}
3:     values[3][4] ← {}
4:     for item in items do
5:         idx ← i.getClusterIdx()
6:         values[idx][0] ← values[idx][0] + i.getSepalLength()
7:         values[idx][1] ← values[idx][1] + i.getSepalWidth()
8:         values[idx][2] ← values[idx][2] + i.getPetalLength()
9:         values[idx][3] ← values[idx][3] + i.getPetalWidth()
10:        for i in 1 : 3 do
11:            for j in 1 : 4 do
12:                values[i][j] ← values[i][j]/clusterSize[i]
13:                oldCentroids[i][j] ← centroids[i][j]
```

14: $centroids[i][j] \leftarrow values[i][j]$

Функцията recalculateCentroids пресмята новите центроиди по начин, представен в първата секция на този документ. Променливата oldCentroids се използва при проверката за край - ако новите и старите центроиди съвпадат, то алгоритъмът приключва работата си. В реализацията е включен и друг критерий за прекратяване на работата на алгоритъма - минимално намаляне на сумата на квадратите на грешката (фиг. 1). Въпреки че е част от кода, този метод не се използва директно - необходима е промяна по кода за да се използва този критерий вместо описания по - горе.



Фигура 1: Алтернативен критерий за терминиране на работата на алгоритъма [1]

3. Инструкции за компилация

Компилацията може да се извърши по два начина, като за целта е необходим C++ компилатор поддържащ стандарта C++ 14 (и cmake версия>=3.14, ако се използва) Първият начин е да се компилират отделно всички класове и после да се свържат с main file-a. Това може да стане (използвайки g++) като се изпълният командите :
g++ -std=c++1y -c Item.cpp,
g++ -std=c++1y -c KMeans.cpp,
g++ -std=c++1y Item.o KMeans.o main.cpp .
Полученият изходен файл може да се изпълни от терминал. Алтернативно може да се използва CMake като за целта в архива се съдържа CMakeLists.txt файл, с чиято помощ да се компилира програмата. Това може да стане като се изпълнят следните (примерни) команди:
mkdir build
cd build
cmake ..
make

Резултат от изпълнението на командите е изпълнимият файл hw2.
Забележка: Използваните команди са примерни - в зависимост от използваната операционна система може да има разлика при компилацията.

4. Резултати

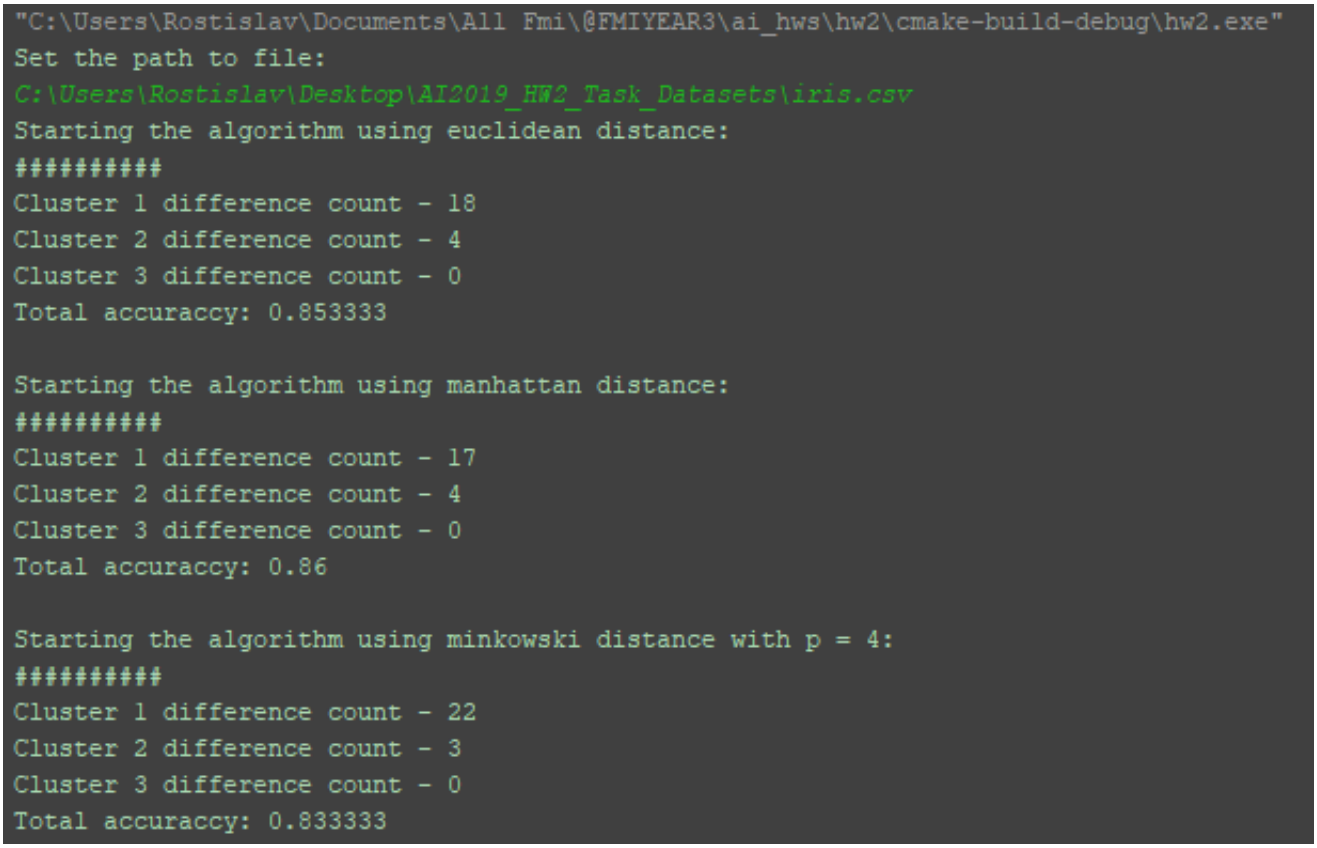
Резултатите от работата на алгоритъма, зависят както от терминалния критерий, така и от избора на началните средни[1],[2]. При своето изпълнение, програмата реализираща описаното решение печата на стандартния изход броя на елементите, които са различни от най - често срещания елемент във всеки клъстер, както и точността, с която сме разпределили елементите (брой сгрешени елементи / брой всички елементи). За целта се използва структурата на дадения файл и факта, че цветята са групирани по 50 елемента от вид, което отговаря на индексите на примерите във файла. На фигури 2,3 и 4 могат да се видят примерни резултати след изпълнението на програмата - във всеки един от примерите различна функция за растояние дава най - добър резултат като във всеки един от примерите и трите начина за намиране на разстояние работят върху един и същи начални центроиди генерирани в началото на работата на програмата.

```
"C:\Users\Rostislav\Documents\All Fmi\@FMIYEAR3\ai_hws\hw2\cmake-build-debug\hw2.exe"
Set the path to file:
C:\Users\Rostislav\Desktop\AI2019_HW2_Task_Datasets\iris.csv
Starting the algorithm using euclidean distance:
##########
Cluster 1 difference count - 0
Cluster 2 difference count - 2
Cluster 3 difference count - 14
Total accuraccy: 0.893333

Starting the algorithm using manhattan distance:
##########
Cluster 1 difference count - 0
Cluster 2 difference count - 2
Cluster 3 difference count - 15
Total accuraccy: 0.886667

Starting the algorithm using minkowski distance with p = 4:
##########
Cluster 1 difference count - 0
Cluster 2 difference count - 3
Cluster 3 difference count - 14
Total accuraccy: 0.886667
```

Фигура 2: Примерен резултат от изпълнението на програмата - евклидовото разстояние дава най - добър резултат



Фигура 3: Примерен резултат от изпълнението на програмата - манхатановота разстояние дава най - добър резултат

```
"C:\Users\Rostislav\Documents\All Fmi\@FMIYEAR3\ai_hws\hw2\cmake-build-debug\hw2.exe"
Set the path to file:
C:\Users\Rostislav\Desktop\AI2019_HW2_Task_Datasets\iris.csv
Starting the algorithm using euclidean distance:
##########
Cluster 1 difference count - 0
Cluster 2 difference count - 3
Cluster 3 difference count - 14
Total accuraccy: 0.886667

Starting the algorithm using manhattan distance:
##########
Cluster 1 difference count - 0
Cluster 2 difference count - 2
Cluster 3 difference count - 15
Total accuraccy: 0.886667

Starting the algorithm using minkowski distance with p = 4:
##########
Cluster 1 difference count - 0
Cluster 2 difference count - 4
Cluster 3 difference count - 12
Total accuraccy: 0.893333
```

Фигура 4: Примерен резултат от изпълнението на програмата - разстоянието на Минковски при p = 4 дава най - добър резултат

Литература

[1] *http://www.mit.edu/~9.54/fall14/slides/Class13.pdf* .

[2] *Лекции по ИИ - летен семестър,2018/2019 учебна година, ФМИ, Проф. д-р Мария Нишева-Павлова*