

Лабораторная работа №9 “Рекомендательные системы”

Долматович Алина, 858641

Набор данных ex9_movies.mat представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две матрицы Y и R - рейтинг 1682 фильмов среди 943 пользователей. Значение R_{ij} может быть равно 0 или 1 в зависимости от того оценил ли пользователь j фильм i. Матрица Y содержит числа от 1 до 5 - оценки в баллах пользователей, выставленные фильмам.

```
In [1]: 1 from scipy.io import loadmat
        2 import math
        3 import numpy as np
        4 import pandas
        5 import matplotlib.pyplot as mp
        6 from sklearn.decomposition import TruncatedSVD
```

Загрузите данные ex9_movies.mat из файла.

```
In [2]: 1 data = loadmat('ex9_movies.mat')
        2 Y = data['Y']
        3 R = data['R']
        4 print(Y.shape, R.shape)
        5
        6 nm, nu = Y.shape
```

```
((1682, 943), (1682, 943))
```

Выберите число признаков фильмов (n) для реализации алгоритма коллаборативной фильтрации.

```
In [3]: 1 n=2
```

Реализуйте функцию стоимости для алгоритма.

```
In [4]: 1 def cost(x, thetha, y, lmbda=10.):
2         cost = (np.sum((x.dot(thetha.T)-y)**2)) / 2
3         cost += lmbda*(np.sum(x**2)) / 2
4         cost += lmbda*(np.sum(thetha**2)) / 2
5         return cost
6
7
8 def defaultValue(nm, nu):
9     ones = np.ones((nm, 1))
10    np.random.seed(41)
11    x = np.random.rand(nm, n) / 100000000
12    x = np.hstack((ones, x))
13
14    zeros = np.zeros((nu, 1))
15    theta = np.random.rand(nu, n) / 100000000
16    theta = np.hstack((zeros, theta))
17    return x, theta
18
19 x, theta = defaultValue(nm, nu)
20 c = cost(x, theta, Y)
21 print(c)
```

694762.0

Реализуйте функцию вычисления градиентов.

При реализации используйте векторизацию для ускорения процесса обучения.

Добавьте L2-регуляризацию в модель.

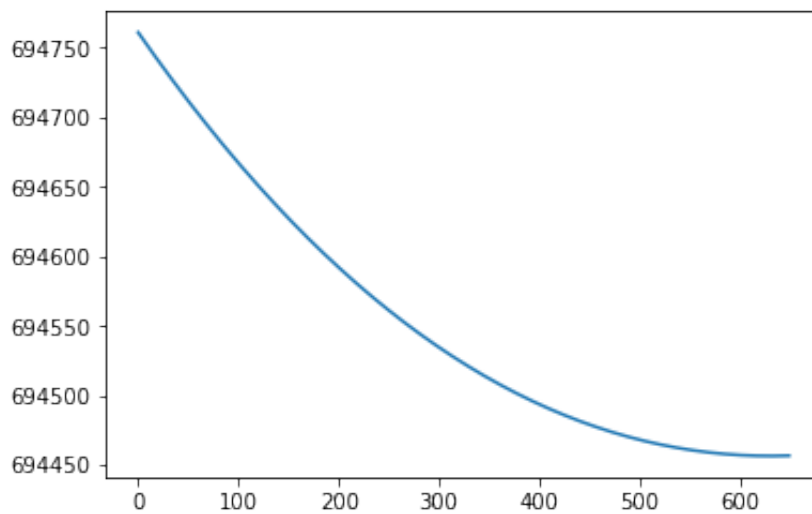
Обучите модель с помощью градиентного спуска или других методов оптимизации.

```
In [5]: 1 def derivateX(x, i, j, k, thetha, y, lmbda):
2         return np.dot((x[i].dot(thetha[j].T)-y[i,j]), thetha[j,k]) + lmbda*x[i,k])
3
4 def derivateThetha(x, i, j, k, thetha, y, lmbda):
5     return np.dot((x[i].dot(thetha[j].T)-y[i,j]), x[i,k]) + lmbda*thetha[j,k])
6
7 def gradient(x, y, R, thetha, alpha=.000001, lmbda=-0.1, iterations=1000):
8     values = dict()
9     for iteration in range(iterations):
10        for film in range(x.shape[0]):
11            for user in range(x.shape[1]):
12                for k in range(n+1):
13                    di = []
14                    dj = []
15                    ...
```

```

15         if R[film, user]:
16             deriv = derivateX(x, film, user, k, theta,
17                               di.append(deriv)
18             deriv2 = derivateTheta(x, film, user, k, t
19                               dj.append(deriv2)
20             x[film, k] -= alpha * sum(di)
21             theta[user, k] -= alpha * sum(dj)
22         values[iteration] = cost(x, theta, Y)
23
24     return x, theta, values
25
26 x, theta, values = gradient(x, Y, R, theta)
27 # c = cost(x, theta, Y)
28
29 mp.plot(values.keys(), values.values())
30 mp.show()
31
32 print(min(values.values()))

```



694456.390146

Добавьте несколько оценок фильмов от себя. Файл movie_ids.txt содержит индексы каждого из фильмов.

```

In [6]: 1 rates = [
2         [0, 3],
3         [66, 4],
4         [68, 5],
5         [71, 4],
6         [93, 5],
7         [140, 5],
8         [150, 4],

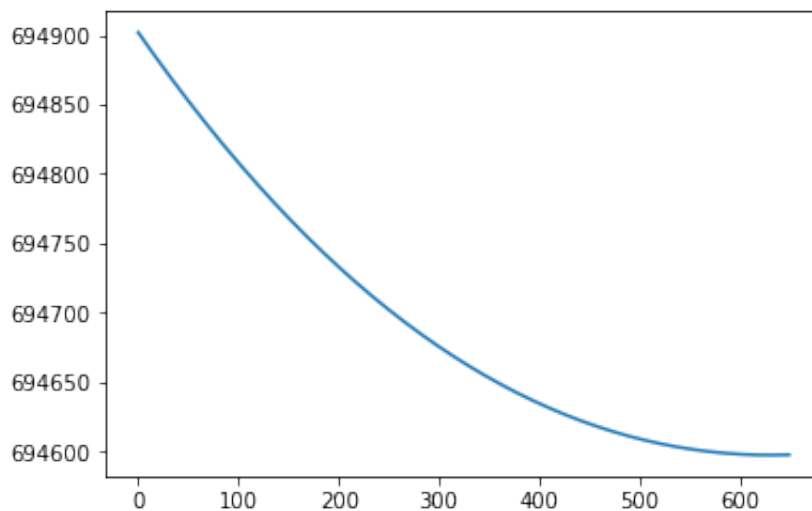
```

```

9      [189, 4],
10     [210, 4],
11     [249, 5],
12     [312, 5],
13     [538, 3],
14     [738, 5],
15     [893, 3],
16     [1442, 5]
17 ]
18
19 myY, myR = np.zeros((nm, 1)), np.zeros((nm, 1))
20 for rate in rates:
21     myY[rate[0]] = rate[1]
22     myR[rate[0]] = 1
23 print(myY.shape, myR.shape)
24
25 Y = np.hstack((Y, myY))
26 R = np.hstack((R, myR))
27
28
29 x, theta = defaultValue(nm, nu+1)
30 x, theta, values = gradient(x, Y, R, theta)
31
32 mp.plot(values.keys(), values.values())
33 mp.show()
34
35 print(min(values.values()))

```

((1682, 1), (1682, 1))



694597.390146

Сделайте рекомендации для себя. Совпали ли они с реальностью?

```
In [7]: 1 prediction = np.dot(x, theta.T)[-1]
        2 prediction.argsort()[-5:][::-1]
```

```
Out[7]: array([ 0,  1,  2, 236, 755])
```

Также обучите модель с помощью сингулярного разложения матриц. Отличаются ли полученные результаты?

```
In [8]: 1 svd = TruncatedSVD(n_components=3, n_iter=250, random_state=42)
        2 thetas = svd.fit_transform(Y)
        3 xes = svd.components_
        4
        5 predictions = thetas.dot(xes)[:,-1]
        6 predictions.argsort()[-5:][::-1]
```

```
Out[8]: array([ 49, 180, 120,  0, 116])
```

Вывод

Рекомендательные системы — программы, которые пытаются предсказать, какие объекты (фильмы, музыка, книги, новости, веб-сайты) будут интересны пользователю, имея определенную информацию о его профиле.

Две основные стратегии создания рекомендательных систем — фильтрация на основе содержания и коллаборативная фильтрация. При фильтрации на основе содержания создаются профили пользователей и объектов, профили пользователей могут включать демографическую информацию или ответы на определённый набор вопросов, профили объектов могут включать названия жанров, имена актёров, имена исполнителей и другую атрибутивную информацию в зависимости от типа объекта. Например, в Music Genome Project музыкальный аналитик оценивает каждую композицию по сотням различных музыкальных характеристик, которые могут использоваться для выявления музыкальных предпочтений пользователя. При коллаборативной фильтрации используется информация о поведении пользователей в прошлом — например, информация о покупках или оценках. В этом случае не имеет значения, с какими типами объектов ведётся работа, но при этом могут учитываться неявные характеристики, которые сложно было бы учесть при создании профиля. Основная проблема этого типа рекомендательных систем — «холодный старт»: отсутствие данных о недавно появившихся в системе пользователях или объектах.

В процессе работы рекомендательные системы собирают данные о пользователях, используя сочетание явных и неявных методов. Примеры явного сбора данных:

- запрос у пользователя оценки объекта по дифференцированной шкале;
- запрос у пользователя ранжировки группы объектов от наилучшего к наихудшему;
- предъявление пользователю двух объектов с вопросом о том, какой из них лучше;
- предложение создать список объектов, любимых пользователем. Примеры неявного сбора данных:
- наблюдение за тем, что осматривает пользователь в интернет-магазинах или базах данных другого типа;
- ведение записей о поведении пользователя онлайн;
- отслеживание содержимого компьютера пользователя.

Рекомендательные системы сравнивают однотипные данные от разных людей и вычисляют список рекомендаций для конкретного пользователя. Некоторые примеры их коммерческого и некоммерческого использования приведены в статье о коллаборативной фильтрации. Для вычисления рекомендаций используется граф интересов. Рекомендательные системы — удобная альтернатива поисковым алгоритмам, так как позволяют обнаружить объекты, которые не могут быть найдены последними. Любопытно, что рекомендательные системы часто используют поисковые машины для индексации необычных данных.

