Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет информатики и радиоэлектроники”

Факультет компьютерных систем и сетей

Специальность “Обработка больших объемов информации”

Лабораторная работа №9

**Рекомендательные системы**

Выполнил:

магистрант гр. 858641 Кальман В.А.

Проверил:

Стержанов М. В.

Минск 2019

**Данные.**

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 Rij 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.

**1. Загрузите данные ex9\_movies.mat из файла:**

**file\_path = 'ex9\_movies.mat'**

**dataset = sio.loadmat(file\_path)**

**# Y is a 1682x943 matrix, containing ratings (1-5) of**

**# 1682 movies on 943 users**

**# R is a 1682x943 matrix, where R(i,j) = 1**

**# if and only if user j gave a rating to movie i**

**Y, R = dataset['Y'], dataset['R']**

**X = np.zeros**((1682, 10)) # 1682 X 10 matrix , num\_movies X num\_features matrix of movie features

Theta = np.zeros((943, 10)) # 943 X 10 matrix, num\_users X num\_features matrix of user features

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

num\_users, num\_movies, num\_features = 4, 5, 3

**3-6. Реализуйте функцию стоимости для алгоритма. Реализуйте функцию вычисления градиентов. Добавьте L2-регуляризацию в модель:**

def cost\_function(params, Y, R, num\_users, num\_movies, num\_features, Lambda):

"""

Returns the cost and gradient for the collaborative filtering problem

"""

# Unfold the params

X = params[:num\_movies\*num\_features].reshape(num\_movies,num\_features)

Theta = params[num\_movies\*num\_features:].reshape(num\_users,num\_features)

predictions = np.dot(X, Theta.T)

err = (predictions - Y)

J = 1/2 \* np.sum((err\*\*2) \* R)

# compute regularized cost function

reg\_X = Lambda/2 \* np.sum(Theta\*\*2)

reg\_Theta = Lambda/2 \* np.sum(X\*\*2)

reg\_J = J + reg\_X + reg\_Theta

# Compute gradient

X\_grad = np.dot(err\*R, Theta)

Theta\_grad = np.dot((err\*R).T, X)

grad = np.append(X\_grad.flatten(), Theta\_grad.flatten())

# Compute regularized gradient

reg\_X\_grad = X\_grad + Lambda\*X

reg\_Theta\_grad = Theta\_grad + Lambda\*Theta

reg\_grad = np.append(reg\_X\_grad.flatten(), reg\_Theta\_grad.flatten())

return J, grad, reg\_J, reg\_grad

X\_test = X[:num\_movies, :num\_features]

Theta\_test= Theta[:num\_users, :num\_features]

Y\_test = Y[:num\_movies, :num\_users]

R\_test = R[:num\_movies, :num\_users]

params = np.append(X\_test.flatten(), Theta\_test.flatten())

# Evaluate cost function

J, grad = cost\_function(params, Y\_test, R\_test, num\_users, num\_movies, num\_features, 0)[:2]

print("Cost at loaded parameters:", J)

J2, grad2 = cost\_function(params, Y\_test, R\_test, num\_users, num\_movies, num\_features, 1.5)[2:]

print("Cost at loaded parameters (lambda = 1.5):", J2)

Результат выполнения:

('Cost at loaded parameters:', 0.0)

('Cost at loaded parameters (lambda = 1.5):', 0.0)

**7. Обучите модель с помощью градиентного спуска или других методов оптимизации:**

Ynorm, Ymean = normalize\_ratings(Y, R)

num\_users = Y.shape[1]

num\_movies = Y.shape[0]

num\_features = 10

# Set initial Parameters (Theta,X)

X = np.random.randn(num\_movies, num\_features)

Theta = np.random.randn(num\_users, num\_features)

initial\_parameters = np.append(X.flatten(), Theta.flatten())

Lambda = 10

# Optimize parameters using Gradient Descent

paramsFinal, J\_history = gradient\_descent(initial\_parameters, Y, R, num\_users, num\_movies, num\_features, 0.001, 400, Lambda)

Результат выполнения:

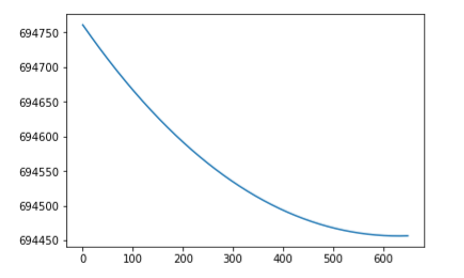


Рисунок 1 – график зависимости функции стоимости в зависимости от количества итераций

**8. Добавьте несколько оценок фильмов от себя. Файл movie\_ids.txt содержит индексы каждого из фильмов:**

def get\_my\_dataset():

my\_ratings = np.zeros((1682, 1))

# Set my own estimation

my\_ratings[11] = 1

my\_ratings[22] = 2

my\_ratings[33] = 3

my\_ratings[44] = 4

my\_ratings[55] = 5

return my\_ratings

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

svd = TruncatedSVD(n\_components=3, n\_iter=250, random\_state=42) thetas = svd.fit\_transform(Y) xes = svd.components\_

predictions = thetas.dot(xes)[:,-1] predictions.argsort()[-5:][::-1]

**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.