Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по предмету:

«Машинное обучение»

По лабораторной работе №9

«Рекомендательные системы»

Выполнил: Сенькович Дмитрий Сергеевич

магистрант кафедры информатики

группы №858642

Проверил: Стержанов Максим Валерьевич

доцент, кандидат технических наук

Минск 2019

**Оглавление**

[1 Введение 2](#_Toc25864457)

[2 Построение рекомендательной системы 6](#_Toc25864458)

[2.1 Постановка задачи 6](#_Toc25864459)

[2.2 Чтение данных 6](#_Toc25864460)

[2.3 Выбор числа признаков 6](#_Toc25864461)

[2.4 Реализация функции стоимости коллаборативной фильтрации. L2-регуляризация 6](#_Toc25864462)

[2.5 Реализация функции вычисления градиентов коллаборативной фильтрации. L2-регуляризация 6](#_Toc25864463)

[2.6 Обучение модели 7](#_Toc25864464)

[2.7 Проверка модели 8](#_Toc25864465)

[2.8 Реализация рекомендательной системы с помощью SVD 9](#_Toc25864466)

# 1 Введение

Рассмотрим очередную задачу машинного обучения - построение рекомендательных систем. Разберем пример такой задачи для рекомендации фильмов пользователям.

Постановка задачи следующая: имеются данные о рейтингах, которые дали пользователя фильмам. Также, имеются данные о факте: ставил ли данный пользователь рейтинг для данного фильма. Требуется по данной информации построить рекомендации пользователям.

Введем некоторые обозначения. Обозначим количество фильмов через nm, а количество пользователей через nu. Построим матрицу y размерности nm x nu - матрица рейтингов, данных фильму пользователем. Обозначим через r матрицу из 0 и 1 той же размерности, что и матрица y. Эта матрица будет показывать, ставили пользователь j рейтинг фильму i.

Рассмотрим первый способ решения задачи: рекомендательные системы на основе содержания. Поставим каждому фильму некоторый вектор значений n фич + единицу в качестве нулевого вектора, чтобы получился (n+1)-мерный вектор фич xi. Например, “процент” романтики в фильме, “боевика” и т.д.. Представим, что у нас есть значения этих фич. Тогда для каждого пользователя можно подобрать такой вектор параметров θj, который, для определенного фильма, будет предсказывать рейтинг, который пользователь бы поставил данному фильму:



Рисунок 1.1. Предсказание рейтинга фильму i пользователем j

Таким образом, для каждого пользователя j задача свелась к оптимизации параметров θj - линейная регрессия. Функция стоимости в такой задаче будет следующая:

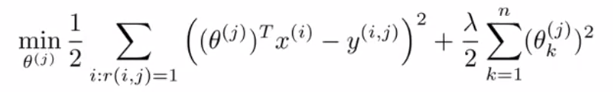


Рисунок 1.2. Функция стоимости для задачи оптимизации параметров θj для рекомендаций фильмов пользователя j

Задачу можно обобщить для всех пользователей с помощью следующей функцией стоимости:

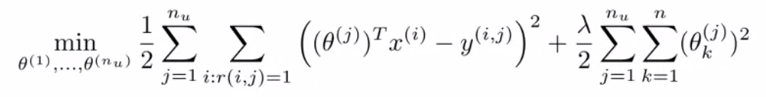


Рисунок 1.3. Функция стоимости для задачи оптимизации параметров для обобщенной задачи

Формулы для оптимизации параметров градиентным спуском будут выглядеть следующим образом:

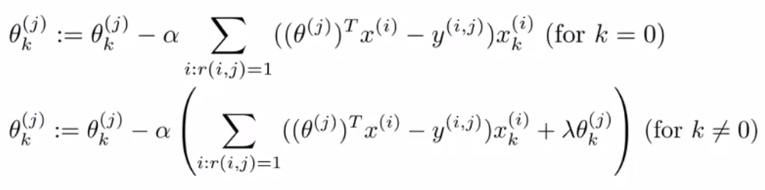


Рисунок 1.4. Формулы для оптимизации параметров

Описанный выше подход может решить рассматриваемую задачу, но у алгоритма существует очень серьезная проблема: значений фич обычно нет.

Поэтому рассмотрим следующий подход к решению этой задачи: коллаборативная фильтрация.

Оставим все обозначения, как есть и рассмотрим следующую задачу: представим себе, что некоторым образом мы получили эти параметры θj для каждого пользователя. В то же время, значений фич фильмов у нас нет и мы хотели бы их выучить. Будем следовать тем же рассуждениям, что и выше, но относительно xi как параметров обучения. В итоге, задача сведется к аналогичной задаче линейной регрессии со следующей функцией стоимости:

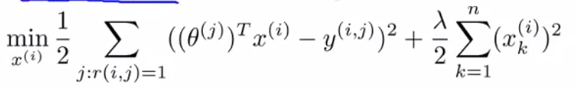


Рисунок 1.5. Формула стоимости для оптимизации xi в качестве параметров обучения линейной регрессией

Снова обобщим задачу, но теперь уже на все фильмы:

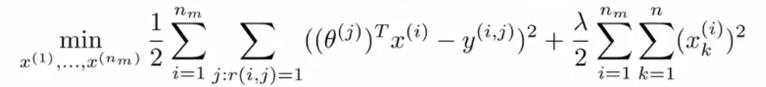


Рисунок 1.6. Формула стоимости для оптимизации xi для всех пользователей

Теперь заметим, что из первого метода, имея некоторые значения векторов θj пользователей мы можем найти оптимальные значения фич xi и наоборот. По сути, мы можем зафиксировать, например, некоторые параметры Θ и оптимизировать параметры X, затем оптимизировать Θ, затем X и т.д..

Основная идея коллаборативной фильтрации: объединить данные функции стоимости и решать уже одну одну задачу со следующей функцией стоимости (заметим, что первые единичные фичи уже не используются):

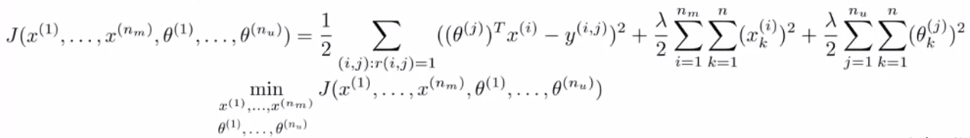


Рисунок 1.7. Формула стоимости в алгоритме коллаборативной фильтрации

В этом случае, градиентный спуск будет оптимизировать каждый из векторов θj и xi:

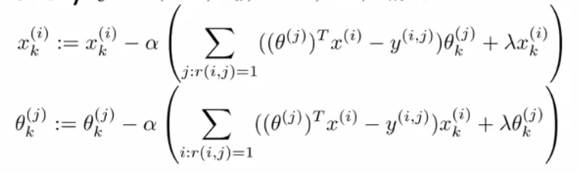


Рисунок 1.8. Формулы обновления параметров в алгоритме коллаборативной фильтрации

Вычисления выше можно ускорить, с помощью векторизации. Для начала, заметим, что при наличие всех параметров и фич предсказание рейтингов по всем фильмам для всех пользователей будет выглядеть следующим образом:

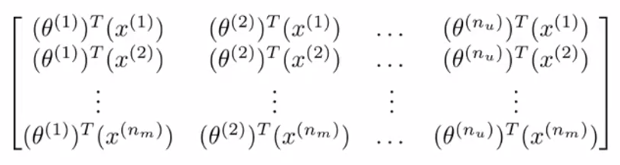


Рисунок 1.9. Матрица предсказаний рейтингов при имеющихся оптимизированных параметрах и фичах

Для векторизации вычисления этой матрицы можно составить матрицы X и Θ так, что матрицы X состоит из транспонированных векторов x для каждого фильма, а Θ - транспонированных векторов θ для каждого пользователя. Тогда следующая матрица разлагается на следующее произведения: X\*ΘT. Такое разложение называется low rank matrix factorization.

# 2 Построение рекомендательной системы

### 2.1 Постановка задачи

По уже известным рейтингам фильмов от пользователей построить рекомендательную систему.

### 2.2 Чтение данных

Считаем данные:

data = loadmat('ex9\_movies.mat')

y = data['Y']

r = data['R']

### 2.3 Выбор числа признаков

Пусть количество фич будет 3:

features\_count = 3

### 2.4 Реализация функции стоимости коллаборативной фильтрации. L2-регуляризация

Реализуем функцию стоимости с регуляризацией:

def calculate\_error(x, theta, r, y):

return np.multiply(x.dot(theta.T), r) - y

def cost(params, y, r, features\_count, l):

movies\_count = y.shape[0]

users\_count = y.shape[1]

x = np.reshape(params[:movies\_count\*features\_count], (movies\_count, features\_count))

theta = np.reshape(params[movies\_count\*features\_count:], (users\_count, features\_count))

error = 0.5\*np.sum(calculate\_error(x, theta, r, y)\*\*2)

penalty = 0.5\*l\*np.sum(x\*\*2) + 0.5\*l\*np.sum(theta\*\*2)

### 2.5 Реализация функции вычисления градиентов коллаборативной фильтрации. L2-регуляризация

Реализуем функцию с регуляризацией:

def gradient(params, y, r, features\_count, l):

movies\_count = y.shape[0]

users\_count = y.shape[1]

x = np.reshape(params[:movies\_count\*features\_count], (movies\_count, features\_count))

theta = np.reshape(params[movies\_count\*features\_count:], (users\_count, features\_count))

error = calculate\_error(x, theta, r, y)

x\_gradient = error.dot(theta) + l\*x

theta\_gradient = error.T.dot(x) + l\*theta

return np.concatenate([x\_gradient.ravel(), theta\_gradient.ravel()])

Заметим, что и функция стоимости, и функция вычисления градиента реализованы с векторизацией.

### 2.6 Обучение модели

Проинициализируем фичи фильмов и параметры пользователей случайными значениями, преобразуем данные в unrolled форму, нормализуем данные, обучим модель с помощью библиотечной реализации алгоритма TNC:

# ------------ initialize parameters ---------

x = np.random.randn(y.shape[0], features\_count)

theta = np.random.randn(y.shape[1], features\_count)

params0 = np.concatenate([x.ravel(), theta.ravel()])

print(cost(params0, y, r, features\_count, 0.1))

# ------------ initialize parameters ---------

# ------------ run optimization ---------

y\_normalized, y\_mean = normalize(y, r)

optimal\_params\_unrolled = optimize.minimize(fun=cost, x0=params0, args=(y\_normalized, r, features\_count, 0.1), method='TNC', jac=gradient, options={'maxiter': 10000, 'disp': True})

movies\_count = y.shape[0]

users\_count = y.shape[1]

optimal\_x = np.reshape(optimal\_params\_unrolled.x[:movies\_count \* features\_count], (movies\_count, features\_count))

optimal\_theta = np.reshape(optimal\_params\_unrolled.x[movies\_count \* features\_count:], (users\_count, features\_count))

# ------------ run optimization ---------

### 2.7 Проверка модели

Добавим несколько оценок от себя. Представим, что нам нравятся фильмы линейки Star Trek и Batman. Добавим вектор наших оценок последней колонкой:

# ------------ add my ratings ---------

my\_y = np.zeros(shape=(y.shape[0], 1))

my\_y[226] = 5 # Star Trek VI: The Undiscovered Country

my\_y[227] = 4 # Star Trek: The Wrath of Khan

my\_y[228] = 5 # Star Trek III: The Search for Spock

# check 229 Star Trek IV: The Voyage Home

my\_y[28] = 5 # Batman Forever

my\_y[230] = 5 # Batman Returns

my\_y[253] = 4 # Batman & Robin

# check 402 Batman

y = np.append(y, my\_y, axis=1)

my\_r = np.zeros(shape=(y.shape[0], 1))

my\_r[226] = 1 # Star Trek VI: The Undiscovered Country

my\_r[227] = 1 # Star Trek: The Wrath of Khan

my\_r[228] = 1 # Star Trek III: The Search for Spock

my\_r[28] = 1 # Batman Forever

my\_r[230] = 1 # Batman Returns

my\_r[253] = 1 # Batman & Robin

r = np.append(r, my\_r, axis=1)

# ------------ add my ratings ---------

Здесь индексы фильмов взяты и файла movie\_ids.txt. Будем ожидать, что наша рекомендательная система предскажет мне не оцененные один фильм Star Trek (ну или что-нибудь про космос) и один фильм про Бэтмена. Обучим систему и выдадим топ 10 рекомендаций для последнего пользователя, то есть нас:

recommendations\_count\_to\_show = 10

p = optimal\_x.dot(my\_theta.T)

my\_rating\_prediction = (p.reshape(-1, 1) + y\_mean).flatten()

max\_ratings = my\_rating\_prediction.argsort()[::-1][:recommendations\_count\_to\_show]

print('My recommendations:')

for movie\_index in max\_ratings:

print('Movie: %s' % movies[movie\_index])

В результате обучения алгоритма и предсказывания фильмов может получиться следующий список фильмов:

My recommendations:

Movie: Turbo: A Power Rangers Movie (1997)

Movie: Gumby: The Movie (1995)

Movie: 8 Seconds (1994)

Movie: Hurricane Streets (1998)

Movie: For Ever Mozart (1996)

Movie: Scarlet Letter, The (1995)

Movie: Harlem (1993)

Movie: Head Above Water (1996)

Movie: Winter Guest, The (1997)

Movie: Race the Sun (1996)

Рекомендации вышли не совсем ожидаемые. Только первый фильм связан с супергероями, а остальные в основном драмы с комедиями. Ни одного боевика, фильма о супергероях, космосе. Обучения производилось с чуть большим количеством фичей, несколькими значениями параметра регуляризации и максимального количества итераций алгоритма, но удовлетворительного результата получить не удалось. Вероятно, следовало бы выбрать большей фич, как в следующем пункте с реализацией с помощью svd, ввиду долгой работы алгоритма проверить это не представилось возможным за разумное время.

### 2.8 Реализация рекомендательной системы с помощью SVD

Найдем сингулярное разложение матрицы y. Левая матрица будет показывать, насколько пользователя “любят” каждую из фич, а правая - насколько каждая из фичей релевантна каждому из фильмов. В итоге для некоторого прогноза нам нужно выбрать несколько фич, скажем, 50, разложить матрицу сингулярным разложением, взять 50 первых сингулярных векторов по количеству дисперсии (в нашем случае мы используем функцию, которая сразу возвращает только k максимальных по дисперсии сингулярных векторов) и перемножить получившиеся 3 матрицы обратно:

def predict\_svd(u, s, vt, y\_mean):

return predict\_svd\_no\_mean(u, s, vt) + y\_mean.reshape(-1, 1)

def predict\_svd\_no\_mean(u, s, vt):

return u.dot(s).dot(vt)

y = y.T

r = r.T

y\_normalized, y\_mean = normalize(y, r)

u, s, vt = svds(y\_normalized, k = best\_k)

s = np.diag(s)

all\_user\_predicted\_ratings = predict\_svd(u, s, vt, y\_mean)

recommendations\_count\_to\_show = 10

my\_rating\_prediction = all\_user\_predicted\_ratings[943]

max\_ratings = my\_rating\_prediction.argsort()[::-1][:recommendations\_count\_to\_show]

print('My recommendations:')

for movie\_index in max\_ratings:

print('Movie: %s' % movies[movie\_index])

В этом случае получим намного лучшие предсказания:

My recommendations:

Movie: Batman Returns (1992)

Movie: Alien 3 (1992)

Movie: Batman Forever (1995)

Movie: Star Trek III: The Search for Spock (1984)

Movie: Star Trek V: The Final Frontier (1989)

Movie: Silence of the Lambs, The (1991)

Movie: Heat (1995)

Movie: Time to Kill, A (1996)

Movie: So I Married an Axe Murderer (1993)

Movie: Star Trek: The Motion Picture (1979)

Некоторые фильмы пересекаются с теми, что мы уже смотрели, впрочем, это уже лучше. Из фильмов, которые мы не смотрели, только 1 не соответствует нашим вкусам: последний. Это фильм о романтике. Среди остальных, еще не просмотренных нами фильмов есть один фильм про инопланетян, один психологический триллер и 3 криминальных драмы, что вполне совпадает с реальностью для такого человека, которому нравится линейка фильмов Star Trek и Batman.