Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Кафедра информатики

Лабораторная работа №9
«Рекомендательные системы»

Выполнил: Чёрный Родион Павлович
магистрант кафедры информатики
группа №858642

Проверил: доцент кафедры информатики
Стержанов Максим Валерьевич

Минск 2019

Постановка задачи

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.
2. Выберите число признаков фильмов (n) для реализации алгоритма коллаборативной фильтрации.
3. Реализуйте функцию стоимости для алгоритма.
4. Реализуйте функцию вычисления градиентов.
5. При реализации используйте векторизацию для ускорения процесса обучения.
6. Добавьте L2-регуляризацию в модель.
7. Обучите модель с помощью градиентного спуска или других методов оптимизации.
8. Добавьте несколько оценок фильмов от себя. Файл **movie_ids.txt** содержит индексы каждого из фильмов.
9. Сделайте рекомендации для себя. Совпали ли они с реальностью?
10. Также обучите модель с помощью сингулярного разложения матриц. Отличаются ли полученные результаты?

Описание реализации

1. Загружаем данные об оценках фильмов пользователями из файла ex9_movies:

```
mat = loadmat("ex9_movies")
rating_matrix = mat['R']
y = mat['Y']
```

2. Положим число признаков n равным 4.

3. Реализуем функцию стоимости для алгоритма коллаборативной фильтрации. При реализации будем использовать векторизацию, а также добавим L2 регуляризацию:

```
def cost(self):
    ratings_vector = self.y[self.rate_idx]
    theta = self.theta[self.rate_idx[1]]
    x = self.x[self.rate_idx[0]]
    predictions = np.sum(theta * x, axis=1)
    error = ratings_vector - predictions
    cost = np.dot(error.T, error)
    reg = self.reg_coeff * (np.sum(self.x ** 2) = np.sum(self.theta ** 2))
    return (cost + reg) * 0.5
```

4. Реализуем функцию вычисления градиентов для матрицы X (признаки фильмов) и Θ (признаки пользователей). При реализации будем использовать векторизацию, а также добавим L2 регуляризацию:

```
def compute_gradient(self):
    x_grad, theta_grad = np.zeros(self.x.shape), np.zeros(self.theta.shape)
    for movie_id in range(self.n_movies):
        theta = self.theta[self.r[movie_id] == 1]
        rates = self.y[movie_id][self.r[movie_id] == 1]
        predictions = np.sum(theta * self.x[movie_id], axis=1)
        error = predictions - rates
        for k in range(self.n):
            reg = self.reg_coeff * self.x[movie_id][k]
            x_grad[movie_id][k] = np.sum(np.multiply(error, theta.T[k])) + reg
    for user_id in range(self.n_users):
        x = self.x[self.rt[user_id] == 1]
        rates = self.yt[user_id][self.rt[user_id] == 1]
        predictions = np.sum(self.theta[user_id] * x, axis=1)
        error = predictions - rates
        for k in range(self.n):
            reg = self.reg_coeff * self.theta[user_id][k]
            theta_grad[user_id][k] = np.sum(np.multiply(error, x.T[k])) + reg
```

```
    return x_grad, theta_grad
```

7. Произведем 500 итераций градиентного спуска с шагом обучения 0.0005. Кривая обучения изображена на рисунке 1.

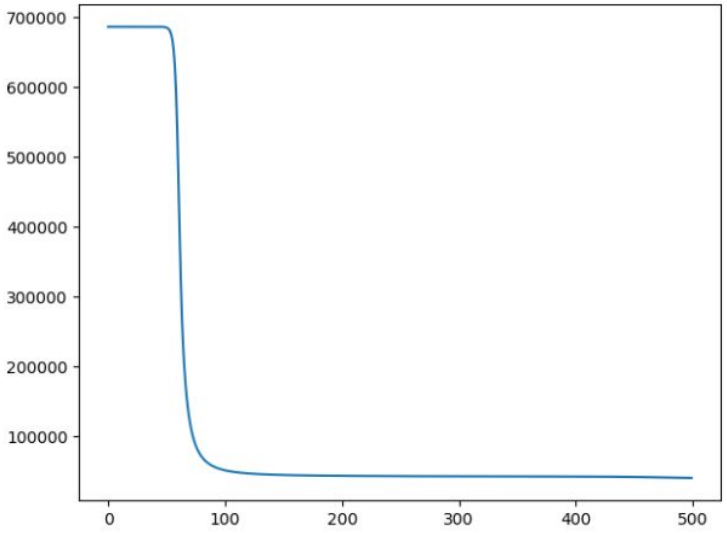

Рисунок 1 - Кривая обучения коллаборативной фильтрации с шагом 0.0005

8. Добавим свои личные оценки в исходные данные. Id нового пользователя будет 943.

User 943 rating Toy Story (1995) with 5.
User 943 rating From Dusk Till Dawn (1996) with 5.
User 943 rating Clerks (1994) with 5.
User 943 rating Pulp Fiction (1994) with 5.
User 943 rating Forrest Gump (1994) with 5.
User 943 rating 2001: A Space Odyssey (1968) with 5.
User 943 rating Good, The Bad and The Ugly, The (1966) with 5.
User 943 rating Mission: Impossible (1996) with 5.
User 943 rating Stalker (1979) with 5.

9. Зная векторы предпочтений для каждого пользователя и зная векторы признаков для каждого фильма, предскажем оценки, которые пользователь 943 поставит всем фильмам. Отсортируем список в порядке убывания рейтинга и выведем 10 фильмов, которые бы пользователю наиболее понравились.

Рекомендации перечислены в таблице 1.

№	Название фильма
1	Pather Panchali (1955)
2	Faust (1994)
3	Wallace & Gromit: A Close Shave (1995)
4	Wallace & Gromit: The Wrong Trousers (1993)
5	Schindler's List (1993)
6	Maya Lin: A Strong Clear Vision
7	Casablanca (1942)
8	The Shawshank Redemption
9	Kaspar Hauser (1993)
10	Wallace & Gromit: The Best of Aardman Animation (1996)

Таблица 1 - Рекомендации новому пользователю

Рекомендации крайне интересные. В списке есть два фильма, которые я уже видел (но оценки к которым не добавил): The Shawshank Redemption и Schindler’s List, но которые мне очень нравятся. Тут можно сказать, что алгорітм попал в самую точку. Из-за наличия в списке оцененных фильмов одного мультфильма - Toy Story, алгоритм решил мне порекомендовать сразу 3 фильма про Уоллеса и Громмита! Что, к слову, тоже неплохая рекомендация. Ничего из остального в списке я не смотрел, но судя по описанию в интернете, это кино на очень большого любителя (возможно всему виной наличие “Сталкера” Тарковского среди оцененных фильмов).

10. Найдем векторы предпочтений пользователей и векторы признаков фильмов при помощи сингулярного разложения матриц.

```
u, s, vt = svds(y.astype(float), k=n)
```

Матрица u - матрица признаков фильмов, матрица vt - транспонированная матрица предпочтений пользователей. Рекомендации, используя эти значения матриц указаны в таблице 2.

№	Название фильма
1	Twelve Monkeys (1995)
2	Trainspotting (1996)
3	Pulp Fiction (1994)
4	Fargo (1996)
5	Reservoir Dogs (1992)
6	The Usual Suspects (1995)
7	Heat (1995)
8	Seven (Se7en)
9	Rumble in the Bronx (1995)
10	The Professional (1994)

Таблица 2 - Рекомендации новому пользователю с использованием алгоритма сингулярного разложения матриц

Рекомендации, честно говоря, поразили попаданием в самую точку. Те фильмы из списка, которые я уже видел, мне безумно нравятся, а те, кто не видел - давно хочу посмотреть.

Результаты, безусловно, отличаются от прошлых рекомендаций. Рекомендации из таблицы 1 более неожиданные, фильмы из того списка я бы вряд ли самостоятельно для себя обнаружил. Результаты из таблицы 2 хоть и попали в самую точку, но ничего неожиданного для себя я там не обнаружил.