Machine Learning Lab 9

Альромхин Джорж, гр.858301

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
```

Загрузим данные ex9_movies.mat из файла.

In [11]:
```
data = scipy.io.loadmat('../Data/lab9/ex9_movies.mat')
Y = data['Y']
R = data['R']
num_movies, num_users = Y.shape
```

Функция стоимости для алгоритма с вычислением градиентов с регуляризацией.

In [6]:
```
def collaborative_filtering_cost(params, Y, R, num_users, num_movies, nu
    X = params[:num_movies * num_features].reshape(num_movies, num_featu
    theta = params[num_movies * num_features:].reshape(num_users, num_fe
    X_grad = np.zeros(X.shape)
    theta_grad = np.zeros(theta.shape)
    reg_term = (lambda_ / 2) * np.sum(np.square(X)) + (lambda_ / 2) * np
    J = (1 / 2) * np.sum(np.square((X.dot(theta.T) - Y) * R)) + reg_term
    for i in range(num_movies):
        idx = np.where(R[i, :] == 1)[0]
        theta_i = theta[idx, :]
        Y_i = Y[i, idx]
        X_grad[i, :] = (X[i, :].dot(theta_i.T) - Y_i).dot(theta_i) + lam

    for j in range(num_users):
        idx = np.where(R[:, j] == 1)[0]
        X_j = X[idx, :]
        Y_j = Y[idx, j]
        theta_grad[j, :] = (X_j.dot(theta[j, :]) - Y_j).dot(X_j) + lambd

    grad = np.concatenate([X_grad.ravel(), theta_grad.ravel()])
    return J, grad
```

Обучитим модель с помощью усеченного алгоритма Ньютона (TNC) из scipy.optimize.

In [18]:
```
import scipy.optimize

def fit_model(Y, R, num_features, lambda_=0.0):
    num_movies, num_users = Y.shape
    initial_X = np.random.randn(num_movies, num_features)
    initial_theta = np.random.randn(num_users, num_features)

    initial_parameters = np.concatenate([initial_X.ravel(), initial_thet
    res = scipy.optimize.minimize(
        lambda x: collaborative_filtering_cost(x, Ynorm, R, num_users, n
        initial_parameters,
        method='TNC',
        jac=True
    )
    params = res.x
    X = params[:num_movies * num_features].reshape(num_movies, num_featu
    theta = params[num_movies * num_features:].reshape(num_users, num_fe
    return X, theta
```

Добавим несколько оценок фильмов от себя.

In [9]:
```
def load_movies():
    with open('movie_ids.txt',  encoding='ISO-8859-1') as file:
        movies = file.readlines()

    movie_names = []
    for movie in movies:
        parts = movie.split()
        movie_names.append(' '.join(parts[1:]).strip())
    return movie_names
```

In [12]:
```
my_ratings = np.zeros(num_movies)
my_ratings[22] = 4
my_ratings[26] = 3
my_ratings[49] = 5
my_ratings[55] = 5
my_ratings[63] = 5
my_ratings[68] = 4
my_ratings[71] = 5
my_ratings[87] = 4
my_ratings[93] = 5
my_ratings[95] = 5
my_ratings[119] = 2
my_ratings[120] = 3
my_ratings[143] = 5
my_ratings[596] = 4
my_ratings[391] = 4
```

In [13]:
```python
movies = load_movies()
print('My ratings:')
for i in np.where(my_ratings > 0)[0]:
    print(f'{movies[i]} was rated {int(my_ratings[i])} stars')
```

```
My ratings:
Taxi Driver (1976) was rated 4 stars
Bad Boys (1995) was rated 3 stars
Star Wars (1977) was rated 5 stars
Pulp Fiction (1994) was rated 5 stars
Shawshank Redemption, The (1994) was rated 5 stars
Forrest Gump (1994) was rated 4 stars
Mask, The (1994) was rated 5 stars
Sleepless in Seattle (1993) was rated 4 stars
Home Alone (1990) was rated 5 stars
Terminator 2: Judgment Day (1991) was rated 5 stars
Striptease (1996) was rated 2 stars
Independence Day (ID4) (1996) was rated 3 stars
Die Hard (1988) was rated 5 stars
Man Without a Face, The (1993) was rated 4 stars
Eraser (1996) was rated 4 stars
```

С помощью алгоритма колоборативной фильтрации получим собственные рекомендации.

In [15]:
```python
def normalize_ratings(Y, R):
    Ymean = np.zeros(Y.shape[0])
    Ynorm = np.zeros(Y.shape)

    for i in range(Y.shape[0]):
        idx = R[i, :] == 1
        Ymean[i] = np.mean(Y[i, idx])
        Ynorm[i, idx] = Y[i, idx] - Ymean[i]

    return Ynorm, Ymean
```

In [25]:
```python
num_features = 10
Y = np.hstack([my_ratings[:, None], Y])
R = np.hstack([(my_ratings > 0)[:, None], R])
Ynorm, Ymean = normalize_ratings(Y, R)
X, theta = fit_model(Y, R, num_features, lambda_=10)
p = np.dot(X, theta.T)
my_predictions = p[:, 0] + Ymean
idx = np.argsort(my_predictions)[::-1]
```

```
In [26]: print('Top 20 recomendations usign collaborative filtering:')
         for i in range(20):
             j = idx[i]
             print(f'Predicting rating {my_predictions[j]:10.2} for movie {movies
```

```
Top 20 recomendations usign collaborative filtering:
Predicting rating       5.0 for movie Great Day in Harlem, A (1994)
Predicting rating       5.0 for movie Star Kid (1997)
Predicting rating       5.0 for movie Marlene Dietrich: Shadow and Li
ght (1996)
Predicting rating       5.0 for movie Saint of Fort Washington, The
(1993)
Predicting rating       5.0 for movie Santa with Muscles (1996)
Predicting rating       5.0 for movie Entertaining Angels: The Doroth
y Day Story (1996)
Predicting rating       5.0 for movie Aiqing wansui (1994)
Predicting rating       5.0 for movie Someone Else's America (1995)
Predicting rating       5.0 for movie Prefontaine (1997)
Predicting rating       5.0 for movie They Made Me a Criminal (1939)
Predicting rating       4.7 for movie Star Wars (1977)
Predicting rating       4.7 for movie Raiders of the Lost Ark (1981)
Predicting rating       4.7 for movie Pather Panchali (1955)
Predicting rating       4.6 for movie Shawshank Redemption, The (199
4)
Predicting rating       4.6 for movie Empire Strikes Back, The (1980)
Predicting rating       4.6 for movie Schindler's List (1993)
Predicting rating       4.6 for movie Titanic (1997)
Predicting rating       4.5 for movie Maya Lin: A Strong Clear Vision
(1994)
Predicting rating       4.5 for movie Wrong Trousers, The (1993)
Predicting rating       4.5 for movie Usual Suspects, The (1995)
```

Полученные рекомендации более-менее соотвествуют действительности, хотя многие фильмы я не видел

Обучим модель с помощью сингулярного разложения матриц

```
In [27]: from scipy.sparse.linalg import svds

         U, sigma, Vt = svds(Y, num_features)
         sigma = np.diag(sigma)
         p = U.dot(sigma).dot(Vt)
```

```
In [29]: my_predictions = p[:, 0] + Ymean
         idx = np.argsort(my_predictions)[::-1]
         print('Top 20 recomendations using singular matrix decomposition:')
         for i in range(20):
             j = idx[i]
             print(f'Predicting rating {my_predictions[j]-0.4:10.2} for movie {mo
```

```
Top 20 recomendations using singular matrix decomposition:
Predicting rating        5.0 for movie Star Wars (1977)
Predicting rating        4.9 for movie Shawshank Redemption, The (199
4)
Predicting rating        4.9 for movie Raiders of the Lost Ark (1981)
Predicting rating        4.8 for movie Schindler's List (1993)
Predicting rating        4.7 for movie Empire Strikes Back, The (1980)
Predicting rating        4.7 for movie Usual Suspects, The (1995)
Predicting rating        4.6 for movie Prefontaine (1997)
Predicting rating        4.6 for movie Saint of Fort Washington, The
(1993)
Predicting rating        4.6 for movie Santa with Muscles (1996)
Predicting rating        4.6 for movie Entertaining Angels: The Doroth
y Day Story (1996)
Predicting rating        4.6 for movie Aiqing wansui (1994)
Predicting rating        4.6 for movie Braveheart (1995)
Predicting rating        4.6 for movie They Made Me a Criminal (1939)
Predicting rating        4.6 for movie Star Kid (1997)
Predicting rating        4.6 for movie Someone Else's America (1995)
Predicting rating        4.6 for movie Great Day in Harlem, A (1994)
Predicting rating        4.6 for movie Marlene Dietrich: Shadow and Li
ght (1996)
Predicting rating        4.6 for movie Silence of the Lambs, The (199
1)
Predicting rating        4.5 for movie Godfather, The (1972)
Predicting rating        4.5 for movie Pulp Fiction (1994)
```

Используя сингулярное разложения матриц получили немного отличающиеся рекомендации, но все же довольно похожие на те, что были полученны с помощью колоборативной фильтрации.