1. Загрузите данные **ex9_movies.mat** из файла.

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

```
load('ex8_movies.mat');
n = 100;
```

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.

```
function [J, grad] = cofiCostFunc(params, Y, R, num_users, num_movies, ...
num_features, lambda)
X = reshape(params(1:num_movies*num_features), num_movies, num_features);
Theta = reshape(params(num_movies*num_features+1:end), ...
num_users, num_features);
J = 0;
X_grad = zeros(size(X));
Theta_grad = zeros(size(Theta));

J = 1/2 * sum(sum((R.* ((X*Theta') - Y)).^2));
X_grad = (R .* (X*Theta' - Y)) * Theta;
Theta_grad = (R .* (X*Theta' - Y))' * X;

% With regularization
J = J + lambda/2 * (sum(sum(Theta.^2)) + sum(sum(X.^2)));
X_grad = X_grad + lambda * X;
Theta_grad = Theta_grad + lambda * Theta;

grad = [X_grad(:); Theta_grad(:)];

end
```

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

```
%  Load data
load('ex8_movies.mat');

%  Y is a 1682x943 matrix, containing ratings (1-5) of 1682 movies by 943 users
%  R is a 1682x943 matrix, where R(i,j) = 1 if and only if user j gave a rating to movi
%  Add our own ratings to the data matrix
Y = [my_ratings Y];
R = [(my_ratings ~= 0) R];

%  Normalize Ratings
[Ynorm, Ymean] = normalizeRatings(Y, R);

%  Useful Values
num_users = size(Y, 2);
num_movies = size(Y, 1);
num_features = 10;
```

```
% Set Initial Parameters (Theta, X)
X = randn(num_movies, num_features);
Theta = randn(num_users, num_features);
initial_parameters = [X(:); Theta(:)];

% Set options for fmincg
options = optimset('GradObj','on','MaxIter',100);

% Set Regularization
lambda = 10;
theta = fmincg(@(t)(cofiCostFunc(t, Ynorm, R, num_users, num_movies, num_features,lambd
```

```
Iteration     1 | Cost: 3.295098e+05
Iteration     2 | Cost: 1.342659e+05
Iteration     3 | Cost: 1.065690e+05
Iteration     4 | Cost: 8.626164e+04
Iteration     5 | Cost: 6.219411e+04
Iteration     6 | Cost: 4.982501e+04
Iteration     7 | Cost: 4.607981e+04
Iteration     8 | Cost: 4.376115e+04
Iteration     9 | Cost: 4.286476e+04
Iteration    10 | Cost: 4.197543e+04
Iteration    11 | Cost: 4.129062e+04
Iteration    12 | Cost: 4.085120e+04
Iteration    13 | Cost: 4.074387e+04
Iteration    14 | Cost: 4.041061e+04
Iteration    15 | Cost: 4.013374e+04
Iteration    16 | Cost: 4.002092e+04
Iteration    17 | Cost: 3.975343e+04
Iteration    18 | Cost: 3.963572e+04
Iteration    19 | Cost: 3.951533e+04
Iteration    20 | Cost: 3.945638e+04
Iteration    21 | Cost: 3.944413e+04
Iteration    22 | Cost: 3.936649e+04
Iteration    23 | Cost: 3.932996e+04
Iteration    24 | Cost: 3.930156e+04
Iteration    25 | Cost: 3.927348e+04
Iteration    26 | Cost: 3.922682e+04
Iteration    27 | Cost: 3.919700e+04
Iteration    28 | Cost: 3.918104e+04
Iteration    29 | Cost: 3.917037e+04
Iteration    30 | Cost: 3.916454e+04
Iteration    31 | Cost: 3.915602e+04
Iteration    32 | Cost: 3.915049e+04
Iteration    33 | Cost: 3.914397e+04
Iteration    34 | Cost: 3.912361e+04
Iteration    35 | Cost: 3.909918e+04
Iteration    36 | Cost: 3.909200e+04
Iteration    37 | Cost: 3.908628e+04
Iteration    38 | Cost: 3.907699e+04
Iteration    39 | Cost: 3.906858e+04
Iteration    40 | Cost: 3.906409e+04
Iteration    41 | Cost: 3.906073e+04
Iteration    42 | Cost: 3.905547e+04
Iteration    43 | Cost: 3.904375e+04
Iteration    44 | Cost: 3.903670e+04
Iteration    45 | Cost: 3.903429e+04
Iteration    46 | Cost: 3.902717e+04
Iteration    47 | Cost: 3.902351e+04
Iteration    48 | Cost: 3.902324e+04
Iteration    49 | Cost: 3.902113e+04
```

```
Iteration    50 | Cost: 3.901937e+04
Iteration    51 | Cost: 3.901752e+04
Iteration    52 | Cost: 3.901201e+04
Iteration    53 | Cost: 3.900409e+04
Iteration    54 | Cost: 3.900161e+04
Iteration    55 | Cost: 3.900053e+04
Iteration    56 | Cost: 3.899909e+04
Iteration    57 | Cost: 3.899799e+04
Iteration    58 | Cost: 3.899657e+04
Iteration    59 | Cost: 3.899558e+04
Iteration    60 | Cost: 3.899443e+04
Iteration    61 | Cost: 3.899293e+04
Iteration    62 | Cost: 3.899198e+04
Iteration    63 | Cost: 3.899134e+04
Iteration    64 | Cost: 3.898977e+04
Iteration    65 | Cost: 3.898814e+04
Iteration    66 | Cost: 3.898764e+04
Iteration    67 | Cost: 3.898732e+04
Iteration    68 | Cost: 3.898618e+04
Iteration    69 | Cost: 3.898257e+04
Iteration    70 | Cost: 3.897974e+04
Iteration    71 | Cost: 3.897903e+04
Iteration    72 | Cost: 3.897828e+04
Iteration    73 | Cost: 3.897787e+04
Iteration    74 | Cost: 3.897770e+04
Iteration    75 | Cost: 3.897745e+04
Iteration    76 | Cost: 3.897722e+04
Iteration    77 | Cost: 3.897707e+04
Iteration    78 | Cost: 3.897684e+04
Iteration    79 | Cost: 3.897668e+04
Iteration    80 | Cost: 3.897652e+04
Iteration    81 | Cost: 3.897635e+04
Iteration    82 | Cost: 3.897594e+04
Iteration    83 | Cost: 3.897542e+04
Iteration    84 | Cost: 3.897484e+04
Iteration    85 | Cost: 3.897470e+04
Iteration    86 | Cost: 3.897459e+04
Iteration    87 | Cost: 3.897446e+04
Iteration    88 | Cost: 3.897388e+04
Iteration    89 | Cost: 3.897364e+04
Iteration    90 | Cost: 3.897362e+04
Iteration    91 | Cost: 3.897331e+04
Iteration    92 | Cost: 3.897320e+04
Iteration    93 | Cost: 3.897314e+04
Iteration    94 | Cost: 3.897307e+04
Iteration    95 | Cost: 3.897282e+04
Iteration    96 | Cost: 3.897270e+04
Iteration    97 | Cost: 3.897269e+04
Iteration    98 | Cost: 3.897255e+04
Iteration    99 | Cost: 3.897246e+04
Iteration   100 | Cost: 3.897240e+04
```

```
% Unfold the returned theta back into U and W
X = reshape(theta(1:num_movies*num_features), num_movies, num_features);
Theta = reshape(theta(num_movies*num_features+1:end), num_users, num_features);
```

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

```
% Load movvie list
movieList = loadMovieList();
```

```
% Initialize my ratings
my_ratings = zeros(1682, 1);

my_ratings(1) = 4;
my_ratings(11) = 4;
my_ratings(42) = 5;
my_ratings(50)= 4;
my_ratings(56) = 5;
my_ratings(72)= 4;
my_ratings(94)= 4;
my_ratings(96) = 5;
my_ratings(122) = 4;
my_ratings(202) = 4;
my_ratings(294)= 4;

fprintf('\n\nNew user ratings:\n');
```

```
New user ratings:
```

```
for i = 1:length(my_ratings)
    if my_ratings(i) > 0
        fprintf('Rated %d for %s\n', my_ratings(i), movieList{i});
    end
end
```

```
Rated 4 for Toy Story (1995)
Rated 4 for Seven (Se7en) (1995)
Rated 5 for Clerks (1994)
Rated 4 for Star Wars (1977)
Rated 5 for Pulp Fiction (1994)
Rated 4 for Mask, The (1994)
Rated 4 for Home Alone (1990)
Rated 5 for Terminator 2: Judgment Day (1991)
Rated 4 for Cable Guy, The (1996)
Rated 4 for Groundhog Day (1993)
Rated 4 for Liar Liar (1997)
```

```
%  Load data
load('ex8_movies.mat');

%  Y is a 1682x943 matrix, containing ratings (1-5) of 1682 movies by 943 users
%  R is a 1682x943 matrix, where R(i,j) = 1 if and only if user j gave a rating to movi
%  Add our own ratings to the data matrix
Y = [my_ratings Y];
R = [(my_ratings ~= 0) R];

%  Normalize Ratings
[Ynorm, Ymean] = normalizeRatings(Y, R);

%  Useful Values
num_users = size(Y, 2);
num_movies = size(Y, 1);
num_features = 10;

% Set Initial Parameters (Theta, X)
```

```
X = randn(num_movies, num_features);
Theta = randn(num_users, num_features);
initial_parameters = [X(:); Theta(:)];

% Set options for fmincg
options = optimset('GradObj','on','MaxIter',100);

% Set Regularization
lambda = 10;
theta = fmincg(@(t)(cofiCostFunc(t, Ynorm, R, num_users, num_movies, num_features,lambd
```

```
Iteration     1 | Cost: 3.306268e+05
Iteration     2 | Cost: 1.284247e+05
Iteration     3 | Cost: 1.066610e+05
Iteration     4 | Cost: 8.354596e+04
Iteration     5 | Cost: 6.215504e+04
Iteration     6 | Cost: 5.196101e+04
Iteration     7 | Cost: 4.704334e+04
Iteration     8 | Cost: 4.410469e+04
Iteration     9 | Cost: 4.360855e+04
Iteration    10 | Cost: 4.230828e+04
Iteration    11 | Cost: 4.153591e+04
Iteration    12 | Cost: 4.102106e+04
Iteration    13 | Cost: 4.077304e+04
Iteration    14 | Cost: 4.046977e+04
Iteration    15 | Cost: 4.020025e+04
Iteration    16 | Cost: 4.009521e+04
Iteration    17 | Cost: 3.979742e+04
Iteration    18 | Cost: 3.968766e+04
Iteration    19 | Cost: 3.955546e+04
Iteration    20 | Cost: 3.948291e+04
Iteration    21 | Cost: 3.939622e+04
Iteration    22 | Cost: 3.933025e+04
Iteration    23 | Cost: 3.928511e+04
Iteration    24 | Cost: 3.923237e+04
Iteration    25 | Cost: 3.921105e+04
Iteration    26 | Cost: 3.918431e+04
Iteration    27 | Cost: 3.916012e+04
Iteration    28 | Cost: 3.913781e+04
Iteration    29 | Cost: 3.910957e+04
Iteration    30 | Cost: 3.909200e+04
Iteration    31 | Cost: 3.908192e+04
Iteration    32 | Cost: 3.906874e+04
Iteration    33 | Cost: 3.906259e+04
Iteration    34 | Cost: 3.905760e+04
Iteration    35 | Cost: 3.904493e+04
Iteration    36 | Cost: 3.903046e+04
Iteration    37 | Cost: 3.902368e+04
Iteration    38 | Cost: 3.902017e+04
Iteration    39 | Cost: 3.901554e+04
Iteration    40 | Cost: 3.901284e+04
Iteration    41 | Cost: 3.901157e+04
Iteration    42 | Cost: 3.900903e+04
Iteration    43 | Cost: 3.900710e+04
Iteration    44 | Cost: 3.900614e+04
Iteration    45 | Cost: 3.900309e+04
Iteration    46 | Cost: 3.900162e+04
Iteration    47 | Cost: 3.899954e+04
Iteration    48 | Cost: 3.899819e+04
Iteration    49 | Cost: 3.899704e+04
Iteration    50 | Cost: 3.899529e+04
Iteration    51 | Cost: 3.899337e+04
Iteration    52 | Cost: 3.899290e+04
```

```
Iteration    53 | Cost: 3.899240e+04
Iteration    54 | Cost: 3.899132e+04
Iteration    55 | Cost: 3.898766e+04
Iteration    56 | Cost: 3.898455e+04
Iteration    57 | Cost: 3.898261e+04
Iteration    58 | Cost: 3.898088e+04
Iteration    59 | Cost: 3.897984e+04
Iteration    60 | Cost: 3.897890e+04
Iteration    61 | Cost: 3.897818e+04
Iteration    62 | Cost: 3.897728e+04
Iteration    63 | Cost: 3.897620e+04
Iteration    64 | Cost: 3.897514e+04
Iteration    65 | Cost: 3.897457e+04
Iteration    66 | Cost: 3.897437e+04
Iteration    67 | Cost: 3.897375e+04
Iteration    68 | Cost: 3.897338e+04
Iteration    69 | Cost: 3.897322e+04
Iteration    70 | Cost: 3.897298e+04
Iteration    71 | Cost: 3.897279e+04
Iteration    72 | Cost: 3.897267e+04
Iteration    73 | Cost: 3.897251e+04
Iteration    74 | Cost: 3.897201e+04
Iteration    75 | Cost: 3.897142e+04
Iteration    76 | Cost: 3.897092e+04
Iteration    77 | Cost: 3.897032e+04
Iteration    78 | Cost: 3.896986e+04
Iteration    79 | Cost: 3.896934e+04
Iteration    80 | Cost: 3.896873e+04
Iteration    81 | Cost: 3.896841e+04
Iteration    82 | Cost: 3.896830e+04
Iteration    83 | Cost: 3.896762e+04
Iteration    84 | Cost: 3.896736e+04
Iteration    85 | Cost: 3.896697e+04
Iteration    86 | Cost: 3.896676e+04
Iteration    87 | Cost: 3.896659e+04
Iteration    88 | Cost: 3.896596e+04
Iteration    89 | Cost: 3.896551e+04
Iteration    90 | Cost: 3.896522e+04
Iteration    91 | Cost: 3.896399e+04
Iteration    92 | Cost: 3.896341e+04
Iteration    93 | Cost: 3.896336e+04
Iteration    94 | Cost: 3.896272e+04
Iteration    95 | Cost: 3.896257e+04
Iteration    96 | Cost: 3.896227e+04
Iteration    97 | Cost: 3.896123e+04
Iteration    98 | Cost: 3.895947e+04
Iteration    99 | Cost: 3.895807e+04
Iteration   100 | Cost: 3.895697e+04
```

```
% Unfold the returned theta back into U and W
X = reshape(theta(1:num_movies*num_features), num_movies, num_features);
Theta = reshape(theta(num_movies*num_features+1:end), num_users, num_features);
```

9. Сделайте рекомендации для себя. Совпали ли они с реальностью?

```
p = X * Theta';
my_predictions = p(:,1) + Ymean;

movieList = loadMovieList();
```

```
[r, ix] = sort(my_predictions,'descend');
for i=1:10
    j = ix(i);
    if i == 1
        fprintf('\nTop recommendations for you:\n');
    end
    fprintf('Predicting rating %.1f for movie %s\n', my_predictions(j), movieList{j});
end
```

```
Top recommendations for you:
Predicting rating 5.0 for movie Aiqing wansui (1994)
Predicting rating 5.0 for movie Santa with Muscles (1996)
Predicting rating 5.0 for movie Someone Else's America (1995)
Predicting rating 5.0 for movie Great Day in Harlem, A (1994)
Predicting rating 5.0 for movie Entertaining Angels: The Dorothy Day Story (1996)
Predicting rating 5.0 for movie Marlene Dietrich: Shadow and Light (1996)
Predicting rating 5.0 for movie Saint of Fort Washington, The (1993)
Predicting rating 5.0 for movie Prefontaine (1997)
Predicting rating 5.0 for movie They Made Me a Criminal (1939)
Predicting rating 5.0 for movie Star Kid (1997)
```

```
for i = 1:length(my_ratings)
    if i == 1
        fprintf('\n\nOriginal ratings provided:\n');
    end
    if my_ratings(i) > 0
        fprintf('Rated %d for %s\n', my_ratings(i), movieList{i});
    end
end
```

```
Original ratings provided:
Rated 4 for Toy Story (1995)
Rated 4 for Seven (Se7en) (1995)
Rated 5 for Clerks (1994)
Rated 4 for Star Wars (1977)
Rated 5 for Pulp Fiction (1994)
Rated 4 for Mask, The (1994)
Rated 4 for Home Alone (1990)
Rated 5 for Terminator 2: Judgment Day (1991)
Rated 4 for Cable Guy, The (1996)
Rated 4 for Groundhog Day (1993)
Rated 4 for Liar Liar (1997)
```