Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Aprendizaje automático

Practica 8: Sistemas de recomendación

Autora:
Nerea Gallego Sánchez (801950)

7 de diciembre de 2022

Índice

1. Introducción y objetivos

El objetivo de la práctica es implementar un sistema de recomendación de películas basado en las valoraciones de los usuarios.

2. Apartado 2

El enunciado pide completar la función **cofiCostFunc.m** para implementar el sistema de recomendación.
Para completar esta función, se recomiendan unos pasos a seguir. Además, el enunciado recomienda realizar una primera implementación con bucles y luego realizar una versión optimizada. En este caso, se ha realizado directamente la versión optimizada con cálculo matricial ya que me parecía más intuitiva.

Primero ha sido necesario implementar la función de coste en el fichero **cofiCostFunc.m**. Para ello, se ha utilizado la fórmula de coste incluida en las transparencias proporcionadas por los profesores de la asignatura:
$J(x^{(1)},...,x^{(n_m)},\theta^{(1)},...,\theta^{(n_u)}) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$
Siendo esta última la función a minimizar.
Por lo tanto, el coste de ha calculado de la siguiente manera:

```
J = (1/2) .* sum(sum(((X * Theta' - Y).*R).^2)) + (lambda /2) .* sum(sum((Theta).^2)) +
    (lambda/2).*sum(sum((X).^2));
```

Siendo X las características de cada película. $Theta$ contiene los parámetros de cada usuario. R contiene si el usuario a realizado esa valoración o no. Y contiene la clasificación de un usuario a una película. Este valor solo está definido si dicha componente de la matriz R tiene valor 1.

A continuación se pide calcular el descenso de gradiente de X y Theta.
Ambos gradientes se han calculado con las fórmulas indicadas en los apuntes tomados en clase. El descenso de gradiente se calcula de la siguiente manera:

$x_k^{(i)} := x_k^{(i)} - \alpha(\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)})$
$\theta_k^{(j)} := \theta_k^{(j)} - \alpha(\sum_{i:r(i,j)=1}((\theta^{j})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)})$

Por lo tanto los gradientes se calculan de la siguiente manera:

```
X_grad = ((X * Theta' - Y) .* R) * Theta + lambda * X;
Theta_grad = ((X * Theta' - Y) .* R)' * X + lambda * Theta;
```

Como se puede observar, se ha aplicado directamente la regularización aplicando el parámetro λ.

Una vez se ha completado la función **cofiCostFunc.m** se ha comprobado que era correcta. Esto se ha realizado mediante el programa principal que proporcionan los profesores de la asignatura.
Este programa indica si la ejecución es correcta mostrando el valor esperado y el valor que se obtiene.
El programa muestra la siguiente salida:

```
%% =============== Part 1: Loading movie ratings dataset ================
Loading movie ratings dataset.

Average rating for movie 1 (Toy Story): 3.878319 / 5


Program paused. Press enter to continue.

%% ============ Part 2: Collaborative Filtering Cost Function ===========

Cost at loaded parameters: 22.224604
(this value should be about 22.22)
```

```
Program paused. Press enter to continue.

%% ============== Part 3: Collaborative Filtering Gradient ==============

Checking Gradients (without regularization) ...
    0.2388    0.2388
         0         0
    1.1365    1.1365
    0.8909    0.8909
    5.6437    5.6437
         0         0
    1.4585    1.4585
   -1.0486   -1.0486
    0.0144    0.0144
         0         0
   -0.8736   -0.8736
    1.0968    1.0968
   -3.0159   -3.0159
         0         0
   -0.3914   -0.3914
    0.4081    0.4081
   -0.1961   -0.1961
   -7.6078   -7.6078
         0         0
   -4.2017   -4.2017
    0.6418    0.6418
   -3.5646   -3.5646
   -1.3419   -1.3419
         0         0
    0.5013    0.5013
   -0.2508   -0.2508
    0.3573    0.3573

The above two columns you get should be very similar.
(Left-Your Numerical Gradient, Right-Analytical Gradient)

If your backpropagation implementation is correct, then
the relative difference will be small (less than 1e-9).

Relative Difference: 9.8125e-13

Program paused. Press enter to continue.

%% ========= Part 4: Collaborative Filtering Cost Regularization ========

Cost at loaded parameters (lambda = 1.5): 31.344056
(this value should be about 31.34)

Program paused. Press enter to continue.

%% ======= Part 5: Collaborative Filtering Gradient Regularization ======

Checking Gradients (with regularization) ...
    0.2967    0.2967
    0.5350    0.5350
   -2.6654   -2.6654
    0.8063    0.8063
   -2.6168   -2.6168
    1.2233    1.2233
    0.8487    0.8487
   -2.3209   -2.3209
    4.0014    4.0014
    0.8172    0.8172
    2.6249    2.6249
   15.2583   15.2583
   -1.5830   -1.5830
   -3.5198   -3.5198
   -0.2579   -0.2579
    1.9673    1.9673
   -1.7062   -1.7062
   -2.6195   -2.6195
    1.1126    1.1126
    2.4360    2.4360
```

```
    1.9898    1.9898
    0.0801    0.0801
   -4.3209   -4.3209
    3.9016    3.9016
   -6.8773   -6.8773
   -6.0830   -6.0830
   -4.4637   -4.4637

The above two columns you get should be very similar.
(Left-Your Numerical Gradient, Right-Analytical Gradient)

If your backpropagation implementation is correct, then
the relative difference will be small (less than 1e-9).

Relative Difference: 1.99739e-12

Program paused. Press enter to continue.

%% ============== Part 6: Entering ratings for a new user ===============

New user ratings:
Rated 4 for Toy Story (1995)
Rated 3 for Twelve Monkeys (1995)
Rated 5 for Usual Suspects, The (1995)
Rated 4 for Outbreak (1995)
Rated 5 for Shawshank Redemption, The (1994)
Rated 3 for While You Were Sleeping (1995)
Rated 5 for Forrest Gump (1994)
Rated 2 for Silence of the Lambs, The (1991)
Rated 4 for Alien (1979)
Rated 5 for Die Hard 2 (1990)
Rated 5 for Sphere (1998)

Program paused. Press enter to continue.

%% ================== Part 7: Learning Movie Ratings ====================

Training collaborative filtering...
Iteration     1 | Cost: 9.566722e+05
Iteration     2 | Cost: 6.280971e+05
Iteration     3 | Cost: 3.521716e+05
Iteration     4 | Cost: 2.675210e+05
Iteration     5 | Cost: 2.058185e+05
Iteration     6 | Cost: 1.612435e+05
Iteration     7 | Cost: 1.422506e+05
Iteration     8 | Cost: 1.304056e+05
Iteration     9 | Cost: 1.247439e+05
Iteration    10 | Cost: 1.172738e+05
Iteration    11 | Cost: 1.127219e+05
Iteration    12 | Cost: 1.094350e+05
Iteration    13 | Cost: 1.054822e+05
Iteration    14 | Cost: 1.039019e+05
Iteration    15 | Cost: 9.970849e+04
Iteration    16 | Cost: 9.415438e+04
Iteration    17 | Cost: 8.942725e+04
Iteration    18 | Cost: 8.700580e+04
Iteration    19 | Cost: 8.506022e+04
Iteration    20 | Cost: 8.236743e+04
Iteration    21 | Cost: 8.020178e+04
Iteration    22 | Cost: 7.884836e+04
Iteration    23 | Cost: 7.809062e+04
Iteration    24 | Cost: 7.772039e+04
Iteration    25 | Cost: 7.730658e+04
Iteration    26 | Cost: 7.689716e+04
Iteration    27 | Cost: 7.635542e+04
Iteration    28 | Cost: 7.614002e+04
Iteration    29 | Cost: 7.610619e+04
Iteration    30 | Cost: 7.570972e+04
Iteration    31 | Cost: 7.559762e+04
Iteration    32 | Cost: 7.548353e+04
Iteration    33 | Cost: 7.506984e+04
Iteration    34 | Cost: 7.461818e+04
Iteration    35 | Cost: 7.424012e+04
Iteration    36 | Cost: 7.401899e+04
```

```
Iteration    37 | Cost: 7.373516e+04
Iteration    38 | Cost: 7.346913e+04
Iteration    39 | Cost: 7.336825e+04
Iteration    40 | Cost: 7.332562e+04
Iteration    41 | Cost: 7.327150e+04
Iteration    42 | Cost: 7.320970e+04
Iteration    43 | Cost: 7.317729e+04
Iteration    44 | Cost: 7.315995e+04
Iteration    45 | Cost: 7.309958e+04
Iteration    46 | Cost: 7.306026e+04
Iteration    47 | Cost: 7.303753e+04
Iteration    48 | Cost: 7.300608e+04
Iteration    49 | Cost: 7.298687e+04
Iteration    50 | Cost: 7.297265e+04
Iteration    51 | Cost: 7.294101e+04
Iteration    52 | Cost: 7.289603e+04
Iteration    53 | Cost: 7.285133e+04
Iteration    54 | Cost: 7.283725e+04
Iteration    55 | Cost: 7.280648e+04
Iteration    56 | Cost: 7.271487e+04
Iteration    57 | Cost: 7.259383e+04
Iteration    58 | Cost: 7.253675e+04
Iteration    59 | Cost: 7.251427e+04
Iteration    60 | Cost: 7.250465e+04
Iteration    61 | Cost: 7.249188e+04
Iteration    62 | Cost: 7.248084e+04
Iteration    63 | Cost: 7.246827e+04
Iteration    64 | Cost: 7.244199e+04
Iteration    65 | Cost: 7.240378e+04
Iteration    66 | Cost: 7.236723e+04
Iteration    67 | Cost: 7.236376e+04
Iteration    68 | Cost: 7.236194e+04
Iteration    69 | Cost: 7.236072e+04
Iteration    70 | Cost: 7.235568e+04
Iteration    71 | Cost: 7.235307e+04
Iteration    72 | Cost: 7.235010e+04
Iteration    73 | Cost: 7.233590e+04
Iteration    74 | Cost: 7.232207e+04
Iteration    75 | Cost: 7.230868e+04
Iteration    76 | Cost: 7.229632e+04
Iteration    77 | Cost: 7.225547e+04
Iteration    78 | Cost: 7.221763e+04
Iteration    79 | Cost: 7.219665e+04
Iteration    80 | Cost: 7.218507e+04
Iteration    81 | Cost: 7.218105e+04
Iteration    82 | Cost: 7.216930e+04
Iteration    83 | Cost: 7.216258e+04
Iteration    84 | Cost: 7.215927e+04
Iteration    85 | Cost: 7.215250e+04
Iteration    86 | Cost: 7.214624e+04
Iteration    87 | Cost: 7.214250e+04
Iteration    88 | Cost: 7.214146e+04
Iteration    89 | Cost: 7.213884e+04
Iteration    90 | Cost: 7.213653e+04
Iteration    91 | Cost: 7.213314e+04
Iteration    92 | Cost: 7.213055e+04
Iteration    93 | Cost: 7.212901e+04
Iteration    94 | Cost: 7.212703e+04
Iteration    95 | Cost: 7.212563e+04
Iteration    96 | Cost: 7.212359e+04
Iteration    97 | Cost: 7.212152e+04
Iteration    98 | Cost: 7.212031e+04
Iteration    99 | Cost: 7.211976e+04
Iteration   100 | Cost: 7.211760e+04

Recommender system learning completed.

Program paused. Press enter to continue.

%% ================== Part 8: Recommendation for you ====================

Top recommendations for you:
Predicting rating 8.5 for movie Titanic (1997)
Predicting rating 8.4 for movie Shawshank Redemption, The (1994)
```

```
Predicting rating 8.4 for movie Star Wars (1977)
Predicting rating 8.3 for movie Schindler's List (1993)
Predicting rating 8.2 for movie Raiders of the Lost Ark (1981)
Predicting rating 8.1 for movie Usual Suspects, The (1995)
Predicting rating 8.1 for movie Good Will Hunting (1997)
Predicting rating 8.1 for movie Braveheart (1995)
Predicting rating 8.0 for movie Godfather, The (1972)
Predicting rating 8.0 for movie Wrong Trousers, The (1993)


Original ratings provided:
Rated 4 for Toy Story (1995)
Rated 3 for Twelve Monkeys (1995)
Rated 5 for Usual Suspects, The (1995)
Rated 4 for Outbreak (1995)
Rated 5 for Shawshank Redemption, The (1994)
Rated 3 for While You Were Sleeping (1995)
Rated 5 for Forrest Gump (1994)
Rated 2 for Silence of the Lambs, The (1991)
Rated 4 for Alien (1979)
Rated 5 for Die Hard 2 (1990)
Rated 5 for Sphere (1998)
```

Se puede observar como la ejecución es correcta.

En el último apartado se pide realizar tus propios ratings. Para eso se ha cambiado la aleatoriedad de la variable *my_ratings*, sustituyendolo por el índice de predicción que realizan con X y Theta y guardando ahora en *my_ratings* los índices calculados.

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

En el apartado 8 de la salida obtenida se muestran los resultados de las recomendaciones.
Se obtienen esas salidas ya que los usuarios que han clasificado de manera similar esas películas, han recomendado también las películas obtenidas como recomendación.