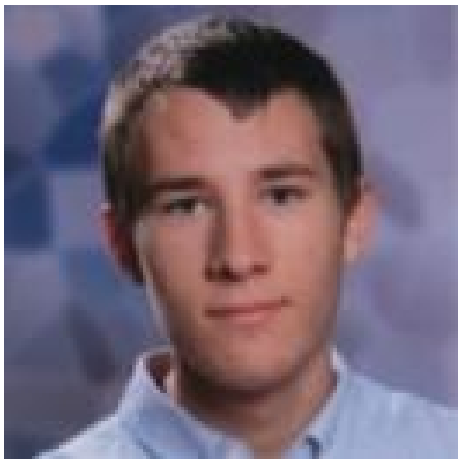
Aprendizaje automático

Práctica 8: Sistemas de recomendación

GAJAN Antoine (894825)



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

Introducción

Como parte de la asignatura “Aprendizaje automático", los estudiantes se familiarizarán con los conceptos de inteligencia artificial. Más concretamente, durante el cuatrimestre se estudiarán las bases del aprendizaje supervisado y no supervisado.

Durante esta última práctica, los estudiantes se familiarizarán con los sistemas de recomendaciones. En particular, pondremos en práctica el conocimiento visto en el curso magistral para poder crear un sistema de recomendación de películas basado en las notas dadas a las películas por el usuario.

Esta memoria pondrá de relieve el proceso emprendido para responder a las preguntas, y producirá una reflexión personal sobre los resultados obtenidos.

1. Estudio previo

Durante este estudio previo de la práctica, me tomé el tiempo de revisar los algoritmos propuestos en el curso magistral y hacerlos en papel. En particular, me tomé el tiempo para revisar y estar seguro de entender bien la función de costo utilizado que debemos minimizar después. También me tomé el tiempo de recalcular manualmente los gradientes para obtener la fórmula propuesta.

2. Memoria de la práctica

2.1. Implementar la función de coste

En un primer momento, utilizando la fórmula propuesta en el curso, establecí una primera versión del código para calcular J de manera iterativa. Este método tiene la ventaja de ser fácil de entender para un principiante, pero nos penaliza porque requiere un tiempo de ejecución importante.

Por lo tanto, me tomé el tiempo para optimizarlo con la ayuda de las calificaciones matriciales de Matlab. En particular, para el primer término de J, que consiste en calcular la suma de las diferencias entre valores reales y valores predichos, es necesario analizar donde R(i,j) = 1. En lugar de hacer un bucle, simplemente podemos multiplicar por término la diferencia por el término correspondiente de R utilizando la calificación ".*". Así, si R(i,j) vale 0, multiplicamos por 0, lo que equivale a no tener en cuenta este punto.

De este modo, conseguimos un código más potente, con menos líneas. Sin embargo, esto se hace a expensas de la legibilidad para un principiante en matemáticas e informática.

Asi tenemos :

```
% Compute the difference between predicted and actual ratings only if R
    (i,j) equals to 1
diff = (X * Theta' - Y) .* R;
% Compute the sum of squared errors unregularized
J = sum(sum(diff .^ 2)) / 2;
```

2.2. Añadirle el gradiente

Para calcular los gradientes, de nuevo, en lugar de proceder a plazo, podemos trabajar con productos de matriz para ahorrar tiempo de ejecución. En efecto, nos damos cuenta rápidamente de que la fórmula utilizada término por término es idéntica en todos los casos. Así, utilizando el producto de matriz, podemos ejecutar todos los cálculos simultáneamente y ahorrar tiempo.

De este modo, conseguimos un código más potente, con menos líneas. Sin embargo, esto se hace a expensas de la legibilidad para un principiante en matemáticas e informática.

Asi tenemos :

```
% Compute X_grad which contains the partial derivatives w.r.t. to each
    element of X
X_grad = ((R .* (X * Theta' - Y)) * Theta) + lambda * X;
```

```
% Compute Theta_grad which contains the partial derivatives w.r.t. to
    each element of Theta
Theta_grad = ((R .* (X * Theta' - Y))' * X) + lambda * Theta;

% Store gradients in a unique matrix
grad = [X_grad(:); Theta_grad(:)];
```

2.3. Añadirle la regularización

A continuación, para calcular la suma relacionada con la regularización, también podríamos hacer un doble bucle para iterar sobre los elementos, ponerlos al cuadrado y sumarlos. En su lugar, podemos directamente terminar la matriz al cuadrado con la ayuda de ".^ luego sumar todos los términos de la matriz A con la instrucción(sum(sum(A)).

Así, tenemos:

```
% Add regularization term to J
reg_term = lambda / 2 * (sum(sum(Theta .^ 2)) + sum(sum(X .^ 2)));
% Get final value of J
J = J + reg_term;
```

2.4. Comprobar la función de coste completa

Ahora que todas estas fórmulas han sido adaptadas en forma optimizada con Matlab, es hora de comprobar que funcionan.

Vamos a proceder paso a paso con el código que se nos propone en P8.m. En particular, primero probaremos si el valor de J devuelto es correcto, luego verificaremos si los gradientes obtenidos son consistentes y finalmente agregaremos la regularización y nos aseguraremos de que funcione.

2.4.1. Valor de J

En primer lugar, comprobaremos el valor de J obtenido al llamar a la función sin regularización (es decir, con lambda = 0).

Obtenemos esto :

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


Program paused. Press enter to continue.
Cost at loaded parameters: 22.224604
(this value should be about 22.22)

Program paused. Press enter to continue.
```

Figura 1: Test del valor de J sin regularización

Vemos que el resultado es consistente con el esperado

2.4.2. Valor de los gradientes

. Ahora podemos probar el valor de los gradientes obtenidos. Obtenemos esto sin regularización:

```
   -2.6735   -2.6735
   -9.7838   -9.7838
   -3.4104   -3.4104
   -2.6130   -2.6130
    2.6955    2.6955
    3.2854    3.2854
   -1.6036   -1.6036
    0.1181    0.1181
    0.4230    0.4230
   -0.5504   -0.5504
    1.0851    1.0851

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: 1.62269e-12

Program paused. Press enter to continue.
Cost at loaded parameters (lambda = 1.5): 31.344056
(this value should be about 31.34)
```

Figura 2: Test del valor de los gradientes sin regularización

De nuevo, notamos que nuestros resultados son relevantes.

2.4.3. Test con la regularización

Entonces podemos verificar los resultados con la regularización añadida. Los resultados son los siguientes:

```
Checking Gradients (with regularization) ...
   -1.6982   -1.6982
   -0.9089   -0.9089
   -4.0487   -4.0487
    2.9958    2.9958
    1.5548    1.5548
   -0.6700   -0.6700
   -3.4329   -3.4329
    3.0518    3.0518
    0.7130    0.7130
    0.9384    0.9384
   -0.6588   -0.6588
    2.8047    2.8047
   -1.5580   -1.5580
    3.2453    3.2453
   -4.1653   -4.1653
   -0.2734   -0.2734
    1.2052    1.2052
   -0.5594   -0.5594
   -1.2411   -1.2411
```

Figura 3: Test del valor de los gradientes sin regularización (1)

```
   -1.9310   -1.9310
   -2.8653   -2.8653
   -2.2591   -2.2591
   -0.2595   -0.2595

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: 2.10697e-12

Program paused. Press enter to continue.
```

Figura 4: Test del valor de los gradientes sin regularización (2)

Una vez más, comprobamos con certeza que nuestro código propuesto funciona perfectamente. Así podremos pasar al entrenamiento del modelo para probar cuáles son las películas más adecuadas en función del usuario.

2.5. Introducir tus calificaciones

Para que el sistema funcione, en primer lugar tenemos que proporcionarle valores de entrada, que corresponden a las calificaciones de las películas vistas. Así, basado en mi cultura y mis gustos cinematográficos, noté una decena de películas vistas con una escala del 1 al 5.

```
%  Recommendation for myself : initialize my ratings
my_ratings = zeros(1682, 1);

 % Toy Story
my_ratings(1) = 5;
% Taxi Driver
my_ratings(23) = 5;
% Belle de jour
my_ratings(30) = 1;
% Star Wars
my_ratings(50) = 2;
% Pulp Fiction
my_ratings(56) = 3;
% Forest Gump
my_ratings(69) = 4;
% Lion King
my_ratings(71) = 3;
% The Mask
my_ratings(72) = 1;
```

```
% Jurrasik Park
my_ratings(82) = 3;
% Aladdin
my_ratings(95) = 5;
% The Aristocats
my_ratings(102) = 4;
% Batman
my_ratings(254) = 1;
% Scream
my_ratings(288) = 2;
% Schindler's List
my_ratings(318) = 5;
% The Game
my_ratings(333) = 4;
```

Para todas las películas que no he visto, he dejado la evaluación en 0, lo que me permite saber que no he anotado la película. Una vez hecho esto, podemos agregar las notas de la persona al vector Y colocando estas notas en la primera columna. También podemos agregar en la primera columna de R si el usuario anotó la película.

Una vez hecho esto, podemos agregar las notas de la persona al vector Y colocando estas notas en la primera columna. También podemos agregar en la primera columna de R si el usuario anotó la película.

2.6. Entrenar el recomendador

Ahora que he indicado las películas que he visto y mi grado de apreciación, somos capaces de entrenar el sistema de recomendación.

Para lograr esto, podemos en primer lugar, como se propone en el código P8.m, normalizar las notas de las películas (para mejorar el descenso de gradientes), y luego calcular los vectores X y Theta correspondientes a las características de las películas y a las características de las personas. Para ello, utilizamos el descenso de gradientes en 100 iteraciones minimizando la función de coste con fmincg.

A continuación, obtenemos los vectores X y Theta del aprendizaje de 100 iteraciones de descenso de gradientes.

2.7. Predecir tus propios ratings y generar recomendaciones para ti mismo

Ahora tenemos los vectores X y Theta del aprendizaje. Para poder calcular las predicciones del usuario, podemos utilizar el cálculo matricial. De hecho, sabemos por el curso que la expresión de las predicciones son la forma X * Theta(usuario)' o Theta(usuario) * X dependiendo de cómo se almacena la información en el vector.

En nuestro caso, será la primera escritura. En efecto, esto se verifica fácilmente con las dimensiones de las matrices.

Ahora podemos mostrar el vector que contiene las predicciones para el usuario. Las películas con la calificación supuestamente más alta son las recomendaciones para el usuario. Así, en mi caso, con las películas que vi y disfruté, obtuve los siguientes resultados:

```
Top recommendations for you:
Predicting rating 3.6 for movie Shawshank Redemption, The (1994)
Predicting rating 3.6 for movie Schindler's List (1993)
Predicting rating 3.5 for movie Good Will Hunting (1997)
Predicting rating 3.5 for movie Titanic (1997)
Predicting rating 3.5 for movie Braveheart (1995)
Predicting rating 3.4 for movie Jerry Maguire (1996)
Predicting rating 3.3 for movie Forrest Gump (1994)
Predicting rating 3.3 for movie One Flew Over the Cuckoo's Nest (1975)
Predicting rating 3.3 for movie Raiders of the Lost Ark (1981)
Predicting rating 3.3 for movie It's a Wonderful Life (1946)


Original ratings provided:
Rated 5 for Toy Story (1995)
Rated 5 for Taxi Driver (1976)
Rated 1 for Belle de jour (1967)
Rated 2 for Star Wars (1977)
Rated 3 for Pulp Fiction (1994)
Rated 4 for Forrest Gump (1994)
Rated 3 for Lion King, The (1994)
Rated 1 for Mask, The (1994)
Rated 3 for Jurassic Park (1993)
```

Figura 5: Recomendaciones de películas para mi mismo

Al no conocer muchas de las películas de esta lista, fui a ver en Internet si en realidad podían coincidir conmigo. Me di cuenta de que algunas películas tenían grandes posibilidades de gustarme y otras no me interesarían.

Conclusión

Para concluir, durante esta práctica hemos podido diseñar un sistema de recomendaciones de películas basadas en las calificaciones atribuidas a las películas vistas. En función de las puntuaciones asignadas a cada película por cada usuario, podemos deducir las características emergentes relativas a los usuarios y a las películas. Esto nos permitió añadir un nuevo usuario (aquí me he añadido con las notas de las películas que he visto). Gracias a las predicciones calculadas, finalmente pudimos saber qué películas podían o no corresponder a mis expectativas basadas en mis experiencias cinematográficas anteriores.

La realización de sistema de recomendación me ha interesado especialmente, ya que es algo que manejamos cotidianamente sin saberlo: Netflix, Google, Amazon,...

Índice de figuras