Inteligencia Artificial
Grado en Ingeniería Informática en Sistemas de Información
ENSEÑANZAS PRÁCTICAS Y DE DESARROLLO

EPD 6: Machine Learning – Sistemas de recomendación

Objetivos

- Implementación en Python de un algoritmo de sistemas de recomendación.

Bibliografía Básica

Recommender systems. Charu C. Aggarwal. Springer, 2016. Disponible online: http://pzs.dstu.dp.ua/DataMining/recom/bibl/1aggarwal_c_c_recommender_systems_the_textbook.pdf

Recommender systems handbook. Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor. Springer, 2011. Disponible online: https://www.cse.iitk.ac.in/users/nsrivast/HCC/Recommender_systems_handbook.pdf

Ejercicios

Implementar un algoritmo que recomiende películas a los usuarios. Para ello, usar el fichero `"ex8_movies.mat"` que contiene datos de películas clasificadas por los usuarios en una escala del 1 al 5. En concreto, 943 usuarios han clasificado 1682 películas. Las películas se identifican con 10 características relativas a su contenido. El objetivo del algoritmo es predecir la puntuación que le daría un usuario a una película que no ha visto aún y recomendar a ese usuario las películas con las puntuaciones más altas.

EJ1. Cargar el dataset y prepararlo para el algoritmo usando 2 matrices. La matriz Y almacenará las clasificaciones de las películas y la matriz R contendrá solamente valores binarios donde `R(i,j) = 1` significará que el usuario `j` clasificó la película `i` y `R(i,j) = 0` indicará que no la clasificó. Ambas matrices tendrán como dimensión: número de películas x número de usuarios. La media de las puntuaciones que recibe la primera película (Toy Story) debe ser aproximadamente 3.878319. Almacenar en las matrices de parámetros X y Theta los valores pre-entrenados disponibles en el fichero `"ex8_movieParams.mat"`. Las dimensiones de X deben ser número de películas x número de características y las de Theta número de características x número de usuarios. Compruebe las dimensiones y actúe en caso de que no coincidan.

EJ2. Implementar la función coste sin regularización para un sistema de recomendación de filtrado colaborativo en `cofiCostFuncSinReg.py` siguiendo la fórmula indicada en EB. El coste se acumula para el usuario `j` y la película `i` sólo si `R(i,j)= 1`. Si usa las matrices de parámetros X y Theta almacenadas en el fichero para los 4 primeros usuarios, 5 primeras películas y 3 primeros atributos/características, el coste debe ser 22.22 aproximadamente

EJ3. Implementar la función gradiente sin regularización en `cofiGradientFuncSinReg.py`. Usar la función auxiliar `checkNNGradients.py` con el parámetro lambda inicializado a 0 para verificar que los gradientes están bien calculados.

EJ4. Implementar la función coste y la función gradiente con regularización en `cofiCostFuncReg.py` y `cofiGradientFuncReg.py` respectivamente. Se debe incluir el parámetro lambda inicializado a 1.5. La función coste debe devolver un coste de 31.34 aproximadamente si usa las matrices de parámetros X y Theta almacenadas en el fichero `"ex8_movieParams.mat"` para los 4 primeros usuarios, 5 primeras películas y 3 primeros atributos. Usar la función auxiliar `checkNNGradients.py` con el parámetro lambda inicializado a 1.5 para verificar que los gradientes están bien calculados.

EJ5. Inicializar de forma random con valores pequeños tanto la matriz X como la matriz Theta para todo el conjunto de datos, utilize la función np.random.rand() indicando las dimensiones en los parámetros de entrada. A continuación, entrenar con regularización para obtener los parámetros óptimos X y Theta usando la función `fmin_cg` de la librería scipy.optimize con 200 iteraciones y lambda con valor 1.5.

EJ6. Después del entrenamiento, conseguir la matriz de predicciones. Además, imprimir por pantalla la recomendación de las 10 películas con mejores puntuaciones para el usuario 2. Deben ser películas que no estuviesen previamente puntuadas por dicho usuario, para ello use `np.where()` con la correspondiente condición.