

Reporte Proyecto Redes Neuronales: Sistema Recomendador de Películas

Integrantes:

Ascencio Bolio Raúl (314340520)

Rodríguez Romero Sergio Alfonso (314187785)

Facultad de Ciencias, UNAM

Fecha de entrega: 12 de Junio de 2020

Introducción

En la actualidad, y debido a la pandemia generada por el virus COVID-19, las plataformas de streaming como Netflix, YouTube, Amazon Prime Video, etc, han sido uno de los medios de entretenimiento más usados para disminuir las tensiones y el estrés generado por la cuarentena. Ya que su contenido es accesible desde una variedad de dispositivos como computadoras, celulares, consolas de videojuegos, tablets y más, sin necesidad de salir del hogar.

En ellas podemos encontrar contenido de nuestro agrado, buscar música, películas, series, entre otras cosas. De la misma manera, y al tiempo que consumimos el contenido de las plataformas, las mismas nos recomiendan contenido personalizado en base a nuestro historial de reproducciones, géneros vistos más frecuentemente, contenido más popular en la región o el que tiene más rating.

Objetivo

La idea del proyecto surgió de la pregunta ¿Cómo es que estas plataformas de entretenimiento nos hacen ciertas recomendaciones de contenido mientras navegamos en ellas y calificamos lo que consumimos?

Es por lo mismo que nos planteamos hacer un Sistema Recomendador de Películas mediante el entrenamiento de redes neuronales para determinar qué contenido debería ser recomendado a un cliente o usuario en base a sus gustos, historial de visualizaciones y el rating que tienen las películas.

Métodos

Después de investigar cuál sería la manera óptima, decidimos usar **Collaborative Filtering basado en usuarios** para hacer el Sistema Recomendador de Películas.

La idea de usar **Collaborative Filtering** es, dado un grupo de usuarios, encontrar aquellos con gustos similares. Es decir, identificar usuarios que tengan valoraciones similares en ciertos items y aprender de ello para poder predecir si a un usuario en particular le puede gustar cierto contenido.

Para este proyecto usaremos el dataset de MovieLens para entrenar la red neuronal y realizar las recomendaciones personalizadas. El dataset puede ser encontrado en [MovieLens Latest Datasets](#).

Algunos de los elementos presentes serán:

1. **Items.** Serán las películas de nuestro dataset. Estos items tendrán valoraciones por parte de los usuarios.
2. **Usuarios.** Es el objetivo del proyecto, estos personajes evaluarán las películas, o items, para poder buscar similitudes entre usuarios y poder hacer las predicciones de contenido a ser recomendado.
3. **Calificaciones.** Será una calificación entre 0 y 5 brindada a las películas por los usuarios. Asumimos que un usuario tuvo que ver la película para poder calificarla.

Para darnos una idea de cómo es que están estructurados los datos comenzamos explorando nuestro dataset. Veremos los usuarios, películas y ratings o calificaciones que tienen las películas. Una vez que conocemos el dataset procederemos a hacer el manejo de datos.

Comenzaremos creando una matriz de usuarios-películas, donde la intersección de un usuario con una película será la calificación brindada por dicho usuario a la película correspondiente.

Lo siguiente que revisaremos será el sparsity que tenemos en la matriz, esto es los huecos o la cantidad de datos faltantes en la matriz, estos huecos corresponden a las películas que no han sido calificadas por los usuarios. Es normal tener una matriz con tantos huecos debido a que no todos los usuarios han visto todas las películas y lógicamente no las han evaluado.

Cabe mencionar que en caso que tuviéramos una matriz con pocos huecos o datos faltantes sería una matriz densa.

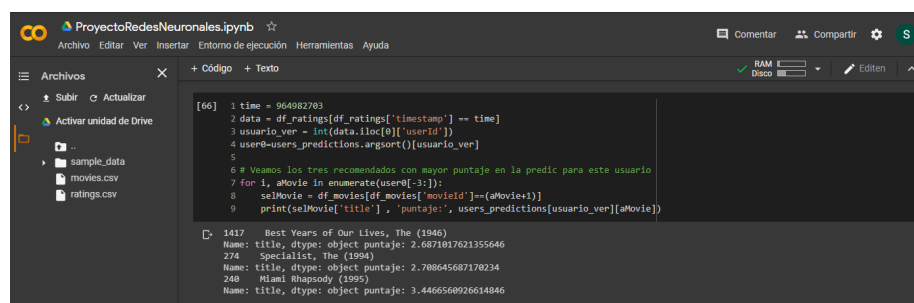
Buscaremos reemplazar esos con las recomendaciones obtenidas al entrenar la red neuronal.

Ya que se realizó esta matriz dividimos nuestros datos en los conjuntos: Train y Test. Una vez obtenidos esos conjuntos solo nos falta calcular la matriz de similitud, esta la realizamos por distancias de coseno y nos da una representación donde cada usuario es similar donde es más cercano su valor a 1.

Lo siguiente que haremos es realizar las predicciones o recomendaciones para un usuario. Se graficó dicha matriz con ayuda de *pyplot* y observamos que hay pocas recomendaciones que tengan puntuaciones altas, esto es debido a que el dataset es relativamente pequeño y tenemos pocos usuarios que han calificado las películas.

Resultados

Para observar los resultado obtenidos, tomamos un usuario como ejemplo y observamos las tres recomendaciones con mayor puntaje en la predicción obtenida para ese usuario en particular. Podemos observar que los puntajes de las

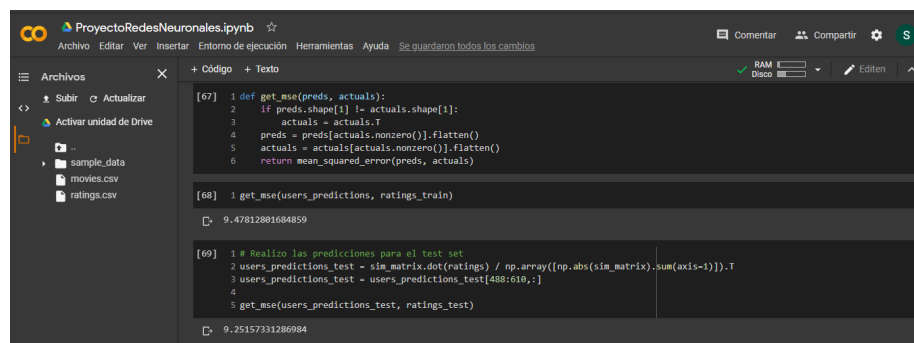


```
[66]: 1 time = 964982783
      2 data = df_ratings[df_ratings['timestamp'] == time]
      3 usuario_ver = int(data.iloc[0]['userId'])
      4 users = users_predictions.argsort()[usuario_ver]
      5
      6 # Veamos los tres recomendados con mayor puntaje en la predic para este usuario
      7 for i, aMovie in enumerate(users[0:-3]):
      8     selMovie = df_movies[df_movies['movieId']==(aMovie+1)]
      9     print(selMovie['title'], ' ', 'puntaje:', users_predictions[usuario_ver][aMovie])

1417 Best Years of Our Lives, The (1946)
      Name: title, dtype: object puntaje: 2.6871017621355646
274 Specialist, The (1994)
      Name: title, dtype: object puntaje: 2.788645687178234
240 Miami Rhapsody (1995)
      Name: title, dtype: object puntaje: 3.4466568926614846
```

recomendaciones no corresponden a los puntajes más altos que se pueden obtener, es decir, de calificación 5. Esto se debe a lo mencionado anteriormente, no tenemos un dataset tan grande y muy pocos usuarios han calificado películas. Por lo que se asume que el resultado obtenido es el óptimo dentro de nuestro dataset.

Para verificar los resultado creamos una función *get_mse* (mean squared error) que nos ayudará a validar los errores. Como podemos observar, para el



```
[67]: 1 def get_mse(preds, actuals):
      2     if preds.shape[1] != actuals.shape[1]:
      3         actuals = actuals.T
      4     preds = preds[actuals.nonzero()].flatten()
      5     actuals = actuals[actuals.nonzero()].flatten()
      6     return mean_squared_error(preds, actuals)

[68]: 1 get_mse(users_predictions, ratings_train)

9.47812861684859

[69]: 1 # Realizo las predicciones para el test set
      2 users_predictions_test = sim_matrix.dot(ratings) / np.array([np.abs(sim_matrix).sum(axis=1)]).T
      3 users_predictions_test = users_predictions_test[488:610,:]
      4
      5 get_mse(users_predictions_test, ratings_test)

9.25157331286984
```

conjunto train y el conjunto test, el error es bastante cercano. Un indicador de

que no estamos realizando buenas predicciones sería si el error en el test fuera mucho más grande que el valor de train.

Conclusiones

Obtuvimos un error de 9 según mean squared error, lo cual a pesar de ser un error grande nos dice que es más por el modelo utilizado que por entrenamiento, ya que cuando se revisa el error con el con de prueba da uno similar siendo la diferencia menor a 1 por lo que lo que genera el error es el modelo y la falta de información y no el entrenamiento.

Bibliografía

- Cross-validation: evaluating estimator performance.
https://scikit-learn.org/stable/modules/cross_validation.html?fbclid=IwAR3oB7Yw9rI1XcbpewvYBZS8U49BZMr-tKhb2uAzvfibnXv8hi8e4jNwe3g#cross-validation
- MovieLens Dataset.
<https://grouplens.org/datasets/movielens/latest/>
- Validación cruzada.
https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada
- Collavorative Filtering.
https://en.wikipedia.org/wiki/Collaborative_filtering