# Informe Avance Proyecto IIC3633

Rodrigo Alliende

## Introducción

En este proyecto se busca probar el funcionamiento de un método de optimización de hiper-parametros. En particular se busca mejorar el sistema de grid search. En grid search tradicional para cada hiperparametro se genera una serie de valores posibles luego se busca exhaustivamente todas las combinaciones de valores y se queda con la que obtiene un mejor resultado en un set de validación. El problema que este método puede tener es que a veces los resultados son volátiles es decir que pueden variar mucho con pequeños cambios en los hiper-parametros. La hipótesis que tengo es que un conjunto de hiper-parametros menos volátil, va a tener un mejor desempeño en un set distinto al de validación. Para encontrar un optimo más estable en vez de tomar el rendimiento del conjunto de hiper-parametros por cuenta propia, se tomo el promedio del rendimiento de ese conjunto y sus vecinos en la grilla.

El problema que intenta resolver este proyecto es similar a otros trabajos que han intento mejorar el proceso de selección de hiper-parametros. Por ejemplo, chan, et al [1] en su paper intentan buscar un método para cambiar los hiper-parametros del problema mientras van cambiando los datos de este. Si bien este problema es similar en este proyecto se esta buscando como optimizar los hiper-parametros dado un conjunto fijo de entrenamiento.

## Exploración de datos

Los datasets utilizados para este proyecto corresponden a last.fm y movielens, siendo estos datasets tradicionales. En movielens podemos ver que tanto la cantidad de películas por usuario (fig 1) como la cantidad de usuarios por película (fig 2) sigue una distriubcion exponencial. También tenemos que cada usuario ha visto por lo menos 20 películas, pero en el caso de las películas tenemos que el mínimo de usuarios al igual que la moda es 1. Para el caso de last.fm tenemos que la cantidad de artistas por usuario (fig 3) es mayoritariamente 50, con algunos usuarios con menos. Mientras que la cantidad de usuarios por artista(fig 4) sigue una distribución exponencial con un mínimo de 1 usuario por artista siendo esta la moda también.

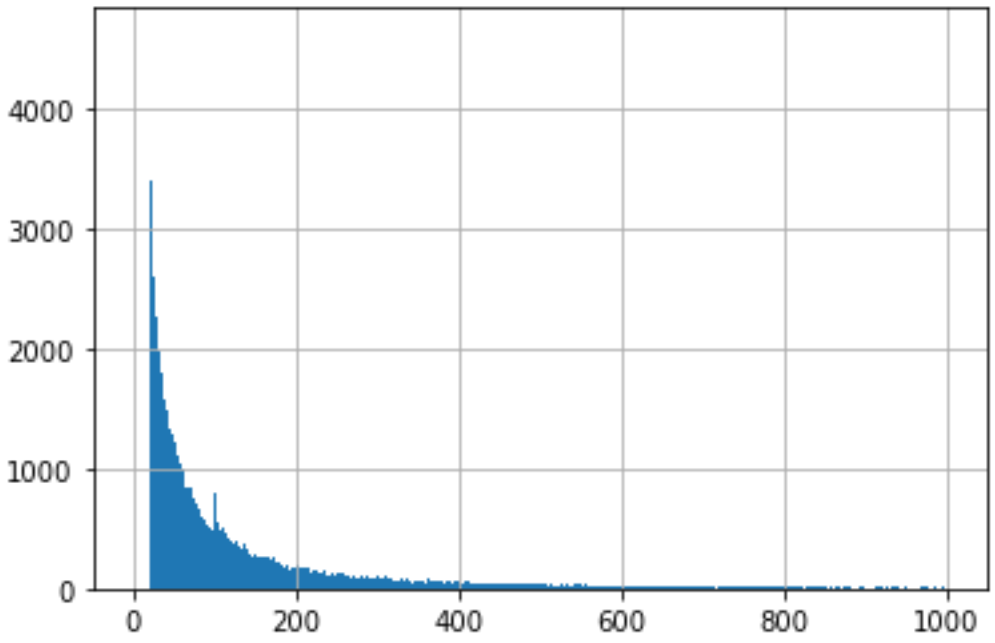


Fig 1 cantidad de películas por usuario

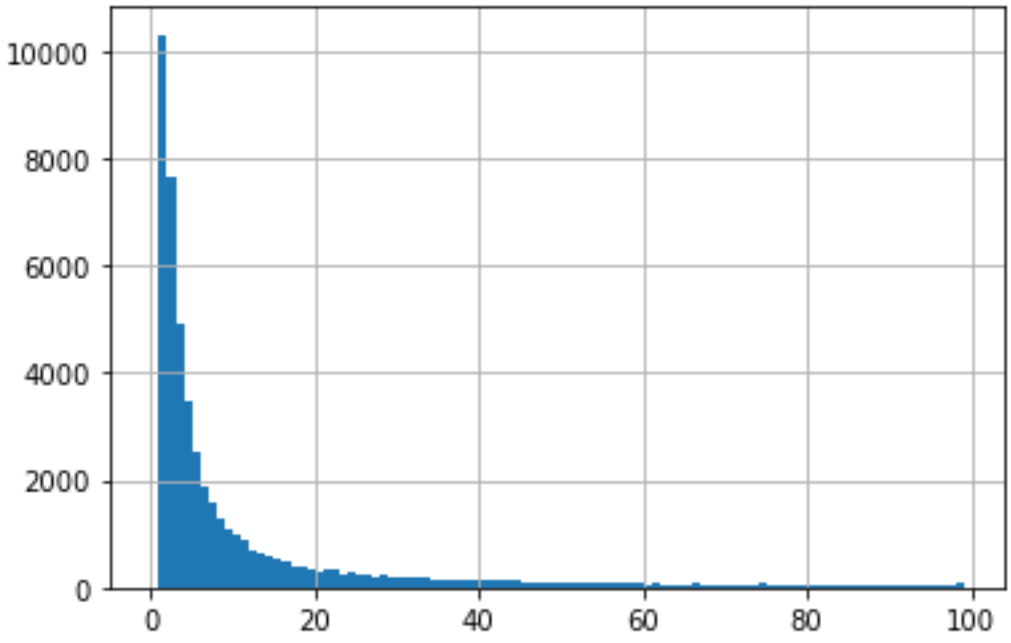


Fig 2 cantidad de usuarios por película

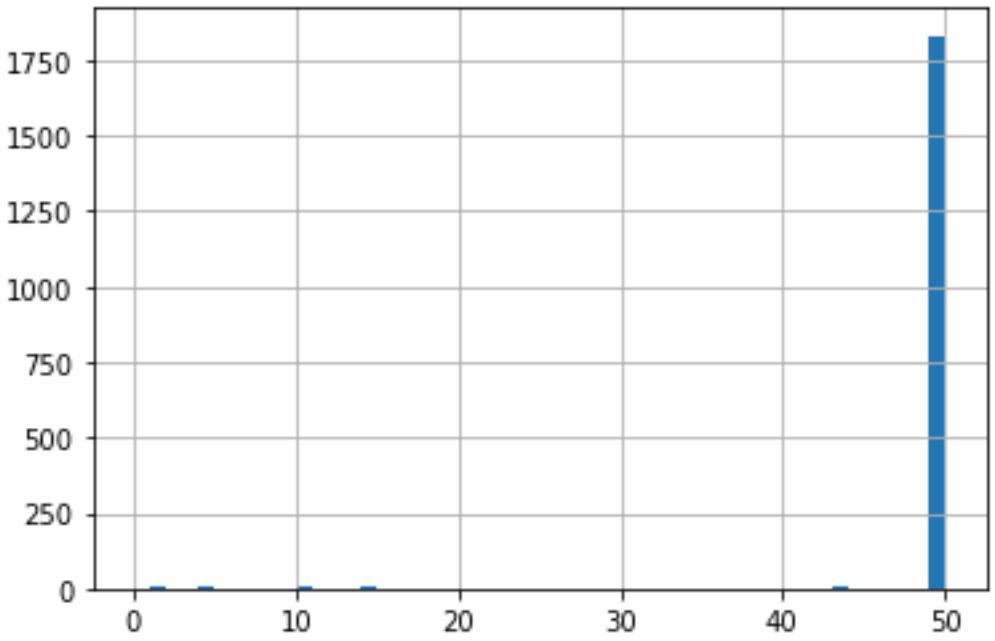


Fig 3 cantidad de artistas por usuario

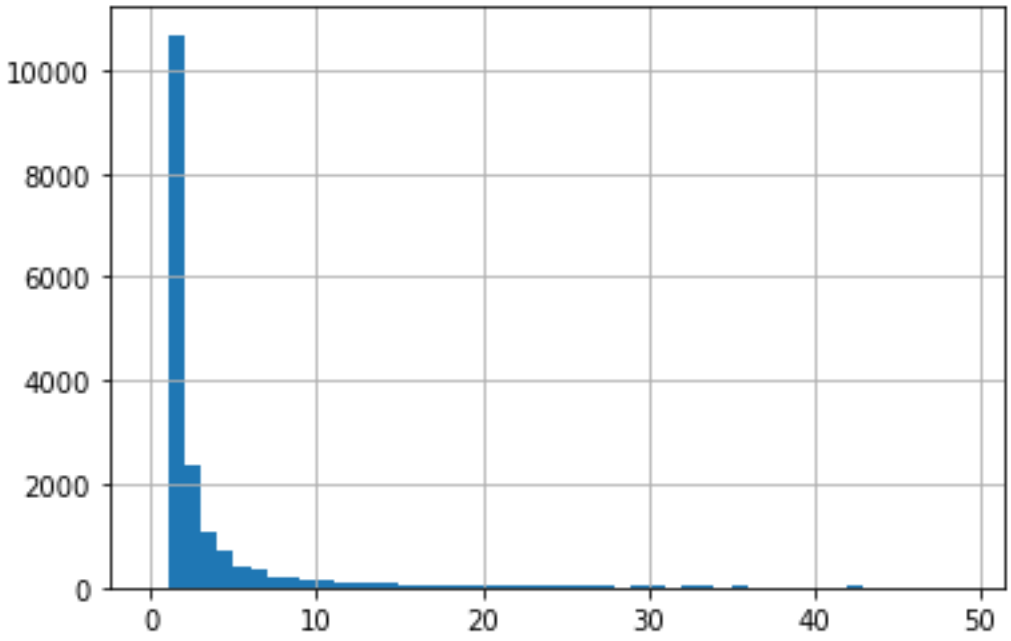


Fig 4 cantidad de usuarios por artistas

# Resultados Preliminares

Para esta entrega probamos el método utilizando únicamente el dataset de Movielens utilizando BPR como método. Para BPR se probaron con 7 valores distintos de numero de factores latentes, tasa de aprendizaje, y factor de regularización. Dando un total de 343 combinaciones de hiper-parametros posibles. El dataset fue divido en 3 sets. El set de testeo que corresponde al 10% del total de los datos. El set de entrenamiento que corresponde al 72% del total de los datos y un set de validación que corresponde al 18% del total de los datos. La métrica utilizada para evaluar el rendimiento del modelo es NDCG 20. Para el método propuesto se utilizo un vecindario de tamaño 2(es decir se consideraron los puntos que en la grilla cuyos hiper parámetros estén dentro de 2 hiper-parametros de distancia). Los valores de los hiper parámetros fueron elegidos utilizando los resultados en el set de validación.

Dado esto los resultados en el set de testeo los podemos ver en la tabla siguiente:

| Método utilizado | NDCG 20 |
|---|---|
| Grid search promedio vecinos | 0.31317603896514645 |
| Grid search normal | 0.3118476817002973 |

De lo que podemos ver el método utilizado proporciona un aumento leve en la precisión de los datos, pero este aumento es bastante pequeño y no es estadísticamente significativo.

# Problemas encontrados

Un problema que tuve es como lidiar con los hiperparametros que se encuentran al borde la grilla, las opciones que tuve fueron ignorarlos o calcular el promedio utilizando solo los vecinos disponibles. Se opto por la segunda opción.

Otro problema que surgió es que para que el método sea relevante se necesitan varios valores de hiper-parametros lo que lleva a muchas combinaciones posibles y lo que hace que la búsqueda sea muy lenta. BPR es un método relativamente rápido de entrenar, y aun así tomo un par de horas en mi computador. Me gustaría poder probar este método utilizando una grilla con un mayor número de valores bajo el mismo rango, pero no creo que mi computador aguante los datos, y colab me va a cerrar la sesión en ese caso. El problema de poder de computo también sería un problema en caso de querer entrenar un modelo de Deep learning, que son considerablemente más lentos de entrenar.

# Plan futuro

En las entregas posteriores planeo expandir el experimento a más métodos, probando su funcionamiento para disintos tamaños de grilla, distintos métodos y además incluir el dataset de last.fm. También podría incluir el funcionamiento utilizando otras métricas de precisión para optimizar el método, como lo son map n o precision promedio.

# Referencias

[1] Chan, et al. Continuous Hyperparameter Optimization for Large-scale Recommender Systems http://www0.cs.ucl.ac.uk/staff/l.capra/publications/bigd-chan.pdf