Sistema de Recomendación para **LastFM**.

Universidad de los Andes

Sergio Alberto Mora Pardo [*]
github: sergiomora03
s.morap@uniandes.edu.co
Bogotá D.C.

Jahir Stevens Rodriguez Riveros
github: jarorid
js.rodriguezr@uniandes.edu.co
Bogotá D.C.

Cindy Zulima Alzate Roman
github: czalzate
c.alzate@uniandes.edu.co
Bogotá D.C.

13 de diciembre de 2020

1. Introducción

Una aplicación de música quiere actualizar su aplicación online para que genere recomendaciones a sus usuarios de nuevos artistas para escuchar. El sistema de recomendación debe tomar en cuenta las preferencias de cada usuario, con el fin de ofrecer recomendaciones automáticas y personalizadas.

Por ello se le pide, desarrollar un algoritmo de recomendación de artistas para cada usuario.

1.1. Definición del problema

Actualmente, la compañia desea actualizar la aplicación online para que genere recomendaciones a nuevos usuarios de artistas para escuchar. La información se encuentra en Celma 2010 en la página de lastfm-360K.

1.2. Pregunta de Investigación

¿Cómo generar recomendaciones automáticas y personalizadas teniendo en cuenta las preferencias de cada usuario?

2. Metodología

Usaremos la librería **Surprise** es un scikit[1] de Python para construir y analizar sistemas de recomendación que traten con datos explicitos de acuerdo con Hug 2020. En ella se enmarcan divesos modelos para construir sistemas de recomendación inspirados[2] y basados en *skit-learn* una librería de Python para machine learning según Pedregosa y col. 2011.

La librería **Surprise** provee diversos algoritmos *listos para usar* como lo son: Metódos de vecindades o Factorización matricial. Así mismo, provee herramientas para evaluar, analizar y

[*] Administrador del repositorio sergiomora03/deep-learning-intermediate

[1] SciKits (abreviatura de SciPy Toolkits) son paquetes complementarios para SciPy, alojados y desarrollados por separado e independientemente de la distribución principal de SciPy. Todos los SciKits tienen licencia aprobada por OSI.

[2] De esto se conversará más adelante, en la sección de presentación de modelos.

comparar el desempeño de los algorítmos. De acuerdo con Hug 2020 muchas de estas herramientas han sido basadas en *skit-learn* como lo son la validación cruzada y la busqueda exaustiva sobre un conjunto de parámetros.

Para el desarrollo y actualización del problema. Se hace necesario crear una API[3]. No obstante, el tiempo es limitado. Generando que nos concentremos en el desarrollo de los modelos y la mejora de los mismos. En este sentido, nuestra metodología es la siguiente:

1. Análisis Exploratorio de Datos.

   Número de usuario, número de artistas, análisis de Pareto sobre las reproducciones.

   Dinámica de usuarios con artistas.

   Análisis de la distribución de reproducciones por usuario.

   Distribución de reproducciones por artista.

   Análisis de distribución por percentiles.

   Análisis de asimetría y curtosis en la distribución de las reproducciones.

2. Benchmark de modelos.

   Selección de métricas.

   Selección de base de datos.

   Análisis de una muestra.

3. Selección del modelo.

   Train test split

4. Hiperparámetros del mejor modelo.

   Mejora de desempeño del modelo con mejor ajuste.

2.1. Identificación y Presentación de los modelos

Para la identificación de los modelos, se implementó un *benchmark* con en el cual se corrieron los siguientes modelos:

- ***Algoritmos basados en factorización matricial***

  SVD[4] Este modelo es equivalente a la Factorización Matricial Probabilistica por Salakhutdinov y Mnih 2007

  Non-negative Matrix Factorization. Algoritmo de filtro colaborativo basado en una factorización matricial no negativa, equivalente al de Wang y Zhang 2013 pero en su forma no regularizada.

- ***Pronósticos Aleatorias***

  Normal Predictor. Supone una distribución normal en la distribución del conjunto de entrenamiento y pronóstica una clasificación aleatoria.

- ***Baseline***

  Baseline Only. Algoritmo que predice la estimación de la línea de base para un usuario y artículo determinados siguienda a Koren, Bell y Volinsky 2009.

- ***Co-Clustering***

  Co-Clustering. Es un algoritmo de filtrado colaborativo basado en agrupación conjunta. Existe una implementación usada en la librería **Surprise** de George 2005.

- ***Algoritmos basado en Vecindades***[5].

  K-NN Básico.

  K-NN Baseline. Un algoritmo de filtrado colaborativo básico que tiene en cuenta una calificación de referencia.

[3] Buitinck y col. 2013, Algunas API basadas en experincias con la librería de *skit-learn*.
[4] Popularizado por *Simon Funk* por el premio de Netflix
[5] Hug 2020, En la librería Surprise se construyó estos con algoritmos que se derivan directamente de un enfoque básico de vecinos más cercanos.

2.2. Estimación de Coeficientes

George y Merugu 2005 indica que La mayoría de los métodos de filtrado colaborativo existentes se basan en criterios de correlación, descomposición de valor singular (SVD) y factorización matricial no negativa. (NNMF) han demostrado proporcionar una alta precisión predicciones de calificaciones.

En la librería **Surprise** se enmarcan los modelos y algoritmos usados. En este sentido, se destina el apéndice para comentar sobre la estimación de los coeficientes[6]. Para la construcción de los coeficientes y comentarios, se destina el apendice A.

3. Resultados y Análisis

3.1. Análisis de los Modelos

3.1.1. Construcción de los modelos

3.1.2. Interpretación de los coeficientes

3.1.3. Supuestos de los Modelos

3.2. Análisis de los Resultados

4. Conclusiones

[6] Por defecto se utiliza el RMSE como la métrica de error a minimizar para la predicción.

Referencias

[1] Lars Buitinck y col. "API design for machine learning software: experiences from the scikit-learn project". En: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, págs. 108-122.

[2] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[3] Thomas George. "A scalable collaborative filtering framework based on co-clustering". En: *Fifth IEEE International Conference on Data Mining*. 2005, págs. 625-628.

[4] Thomas George y Srujana Merugu. "A Scalable Collaborative Filtering Framework based on Co-clustering". En: *Department of Computer Science, Texas A M University* (2005). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6458&rep=rep1&type=pdf.

[5] Nicolas Hug. "Surprise: A Python library for recommender systems". En: *Journal of Open Source Software* 5.52 (2020), pág. 2174. DOI: 10.21105/joss.02174. URL: https://doi.org/10.21105/joss.02174.

[6] Y. Koren, R. Bell y C. Volinsky. "Matrix Factorization Techniques for Recommender Systems". En: *Computer* 42.8 (2009), págs. 30-37. DOI: 10.1109/MC.2009.263.

[7] F. Pedregosa y col. "Scikit-learn: Machine Learning in Python". En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[8] Ruslan Salakhutdinov y Andriy Mnih. "Probabilistic Matrix Factorization". En: *Department of Computer Science, University of Toronto* (2007). URL: https://papers.nips.cc/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf.

[9] Yu-Xiong Wang y Yu-Jin Zhang. "Nonnegative Matrix Factorization: A Comprehensive Review". En: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* 25.6 (2013). URL: http://oa.ee.tsinghua.edu.cn/~zhangyujin/Download-Paper/E224%3DTKDE-13.pdf.

A. Algoritmos en Surprise

A.1. Notaciones

R : the set of all ratings.
R_{train}, R_{test} and $\hat{R}$ denote the training set, the test set, and the set of predicted ratings.
U : the set of all users. u and v denotes users.
I : the set of all items. i and j denotes items.
U_i : the set of all users that have rated item i.
U_{ij} : the set of all users that have rated both items i and j.
I_u : the set of all items rated by user u.
I_{uv} : the set of all items rated by both users u and v.
r_{ui} : the true rating of user u for item i.
$\hat{r}ui$: the estimated rating of user u for item i.
b_{ui} : the baseline rating of user u for item i.
μ : the mean of all ratings.
μ_u : the mean of all ratings given by user u.
μ_i : the mean of all ratings given to item i.
σ_u : the standard deviation of all ratings given by user u.
σ_i : the standard deviation of all ratings given to item i.
$N_i^k(u)$: the k nearest neighbors of user u that have rated item i. This set is computed using a similarity metric.
$N_u^k(i)$: the k nearest neighbors of item i that are rated by user u. This set is computed using a similarity metric.

> NOTA: Estas notaciones son extridas directamente de la librería **Surprise** al igual que las ecuaciones descritas a continuación.

A.2. Algoritmos base:

A.2.1. NormalPredictor

Este algoritmo predice un rating aleatorio asumiendo que la muestra de entrenamiento proviene de una distribución Normal:

$$\hat{r}_{ui} \sim Normal(\hat{\mu}, \hat{\sigma}) \tag{1}$$

donde,

$$\hat{\mu} = \frac{1}{|R_{entrenamiento}|} \sum_{r_{ui} \in R_{entrenamiento}} r_{ui} \tag{2}$$

$$\hat{\sigma} = \sqrt{\sum_{r_{ui} \in R_{entrenamiento}} \frac{(r_{ui} - \hat{\mu})^2}{|R_{entrenamiento}|}} \tag{3}$$

A.2.2. BaselineOnly

El algoritmo obtiene su estimación a partir del rating medio y las desviaciones observadas para la pelicula i y el usuario u:

$$\hat{r}_{ui} = \mu + b_i + b_u \tag{4}$$

donde μ es el rating promedio de los datos de entrenamiento, b_i que es el rating promedio del item i menos μ y b_u que es el rating promedio del usuario u menos μ.

A.3. Algoritmos de vecindades:

A.3.1. KNNBasic

El **KNNBasic** es un modelo que estima los ratings de acuerdo con los k vecinos más cercanos, ya sea por usuario o por item, de acuerdo con:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{5}$$

ó

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot r_{uj}}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{6}$$

donde sim es la función de similitud

A.3.2. KNNWithMeans

Este algoritmo modifica el **KNNBasic** tomando además en cuenta los ratings promedios de los usuarios:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{7}$$

ó

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{8}$$

A.3.3. KNNWithZScore

Este algoritmo toma en cuenta las similitudes y los ratings estandarizados:

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{9}$$

ó

$$\hat{r}_{ui} = \mu_i + \sigma_i \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)/\sigma_j}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{10}$$

A.3.4. KNNBaseline

Este algoritmo toma en cuenta el rating medio y las desviaciones observadas para la pelicula i y el usuario u:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{11}$$

ó

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)} \tag{12}$$

A.4. Algoritmos de factores latentes:

A.4.1. SVD

Este algoritmo corresponde con la factorización de la matriz de ratings (visto en la actividad anterior):

$$\hat{r}_{ui} = q_i' p_u + \mu + b_i + b_u \tag{13}$$

A.4.2. SVDpp

Este algoritmo extiende el SVD al tomar en cuenta los ratings implícitos, ó la cantidad de **feedback** implícito:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i' \left(p_u + |I_u|^{-1/2} \sum_{j \in I_u} y_j \right) \tag{14}$$

donde y_j es un valor binario que captura el hecho de que el usuario u haya calificado o revelado su rating para j (sin importar el valor del rating).

A.4.3. NMF

Este algoritmo corresponde con la factorización no-negativa de la matriz de ratings, y sigue la misma formulación del SVD. Solo que se garantiza que los factores sean no-negativos.

A.5. Algoritmo de clustering:

A.5.1. Co-clustering

En este algoritmo, los usuarios y los items son asignados a los clusters C_u, C_i y C_{ui}:

$$\hat{r}_{ui} = \bar{C_{ui}} + (\mu_u - \bar{C_u}) + (\mu_i - \bar{C_i}) \tag{15}$$

donde $\bar{C_{ui}}, \bar{C_u}, \bar{C_i}$ son respectivamente los rating promedio de los clusters C_{ui}, C_u y C_i. Los clusters se asignan de acuerdo con K-medias.