Sistemas Recomendadores
IIC-3633

Recomendación basada en factores latentes
Parte 2

Esta clase

1. Recomendación basada en factores latentes repaso
2. FunkSVD
3. Alternate Least Squares

Temas generales

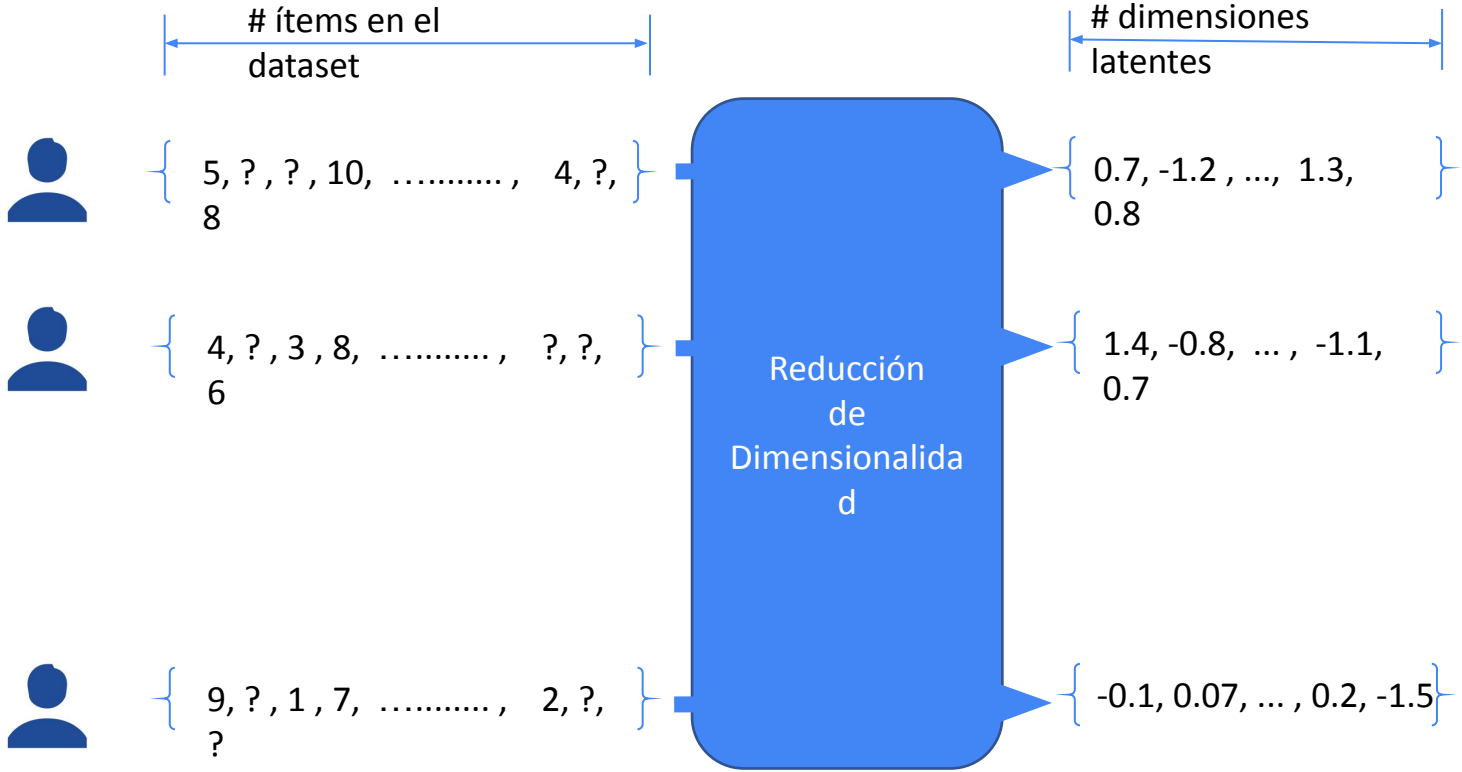
Lectura estará habilitada desde hoy en Perusall hasta el próximo jueves

Tarea 1: Enunciado se va a habilitar mañana. Incluiremos un challenge donde tendrán que proponer el mejor método y hacer un submission de tipo Kaggle.

En CANVAS van a tener que subir el pdf con el informe , el jupyter notebook con el código y el archivo con la submission (nosotros hacemos el ranking).

Ejemplo de tarea 2022

Relación con Matriz Usuarios-Items



Factorización Matricial - Introducción

Filtrado Colaborativo

Recomendar lo que usuarios similares han consumido

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{j \in V(u)} sim(u,j)(r_{j,i} - \bar{r}_j)}{\sum_{j \in V(u)} sim(u,j)}$$

Factorización Matricial (SVD)

Genera vectores latentes por cada usuario e ítem del set de datos

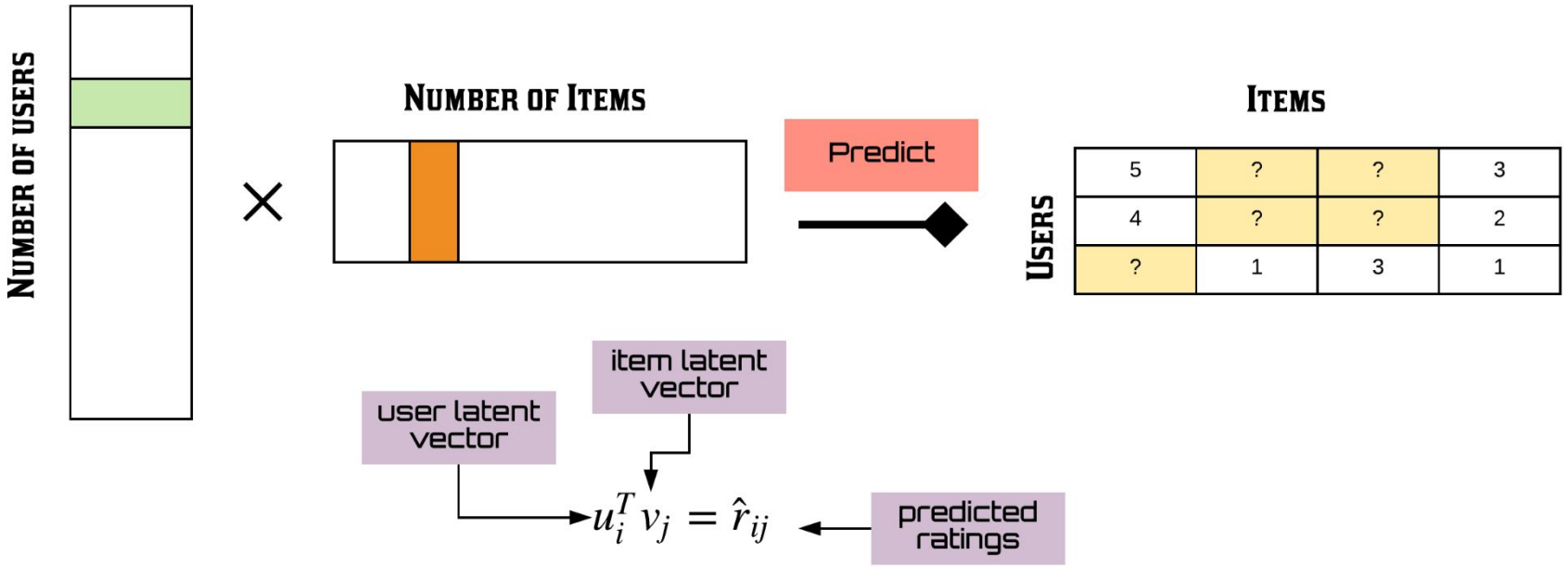
$$\hat{r}_{ui} = \vec{q}_i \cdot \vec{p}_u$$

donde

$\vec{q}_i$: vector latente para el ítem i

$\vec{p}_u$: vector latente para el usuario u

Recomendación basada en factores latentes (factorización matricial SVD).



Un poco de historia

Netflix Prize:

El ganador del concurso implementó el mejor método en términos de rendimiento, pero es muy difícil de llevar a producción porque es un ensemble de modelos.

Uno de los métodos que participó fue Simon Funk con una versión mejorada de Factorización Matricial que además de ser más rápido permite:

- Permite incorporar nuevos usuarios sin un costo computacional tan grande.
- Optimización basada en descenso de gradiente.
- Incluye un factor de regularización (overfitting).

https://sifter.org/simon/journal/20061211.html

Monday, December 11, 2006

Netflix Update: Try This at Home

Solución parcial: FunkSVD

- FunkSVD a diferencia de SVD agrega un factor de regularización, a diferencia de SVD encuentra directamente vectores de usuarios e ítems.

$$\min_{p^*,q^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T \cdot p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

donde

r_{ui}: rating que asignó el usuario u al ítem i

q_i: vector latente del ítem i

p_u: vector latente del usuario u

λ: constante de regularización

Factorización Matricial – Vectores Latentes

- Los vectores latentes capturan atributos de los ítems y si a los usuarios les gustan dichos atributos

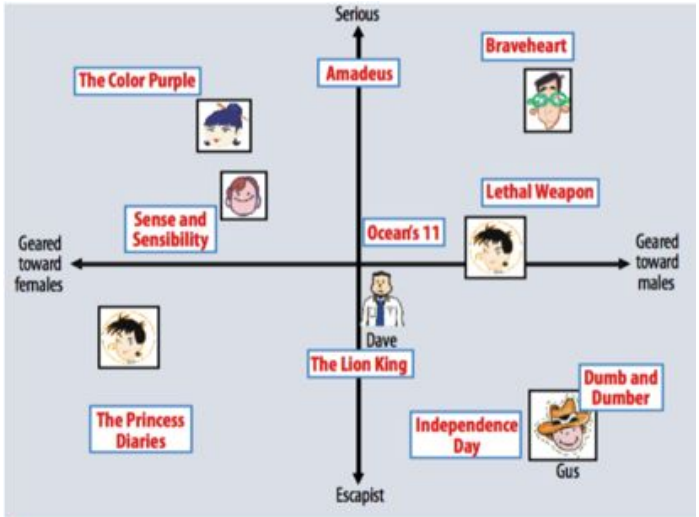


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

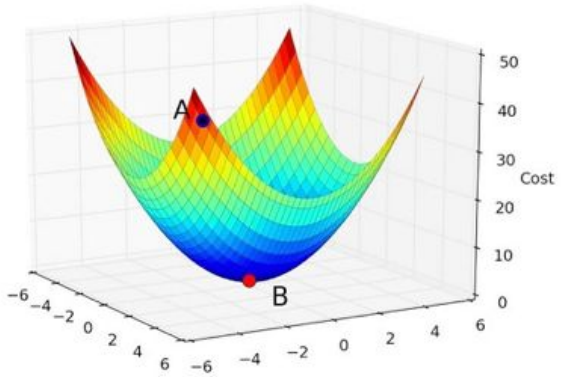
Factorización Matricial - Descenso de Gradiente

- Descenso de gradiente es un procedimiento para optimizar funciones
- La intuición es mover los parámetros del modelo en dirección contraria del gradiente
- En el caso de *FunkSVD* la función de aprendizaje se aplica para cada par (u, i)

Función objetivo $\min_\theta \frac{1}{2}\sum_i (y_i - f(x_i, \theta))^2$

Función de error $E_i = \frac{1}{2}(y_i - f(x_i, \theta))^2$

Actualización de parámetros $\Delta w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$



¿Cómo aprendemos qu y pi?

1. Inicializar aleatoriamente ambos vectores.
2. Usando las reglas de actualización voy iterativamente actualizando los valores de qu y pi
3. Repetir hasta que converja el error.

Alternate Least Squares

Alternate Least Squares

- ALS es un método de optimización para la factorización de matrices que busca aprender vectores de usuarios e ítems de menor dimensionalidad.

- Busca minimizar la diferencia entre las observaciones originales de ratings y las predicciones.

Diferencias con FUNK SVD

- El entrenamiento de ALS es distinto: ALS minimiza dos funciones de pérdida de manera alternada;
  - Primero congela la matriz de usuario y calcula los gradientes de la matriz de ítems.
  - Luego congela de la items y actualiza la de usuarios.

- En cada iteración, se pueden actualizar todas las filas de la matriz de usuarios (o ítems) de forma independiente o todas las columnas de la matriz de ítems (o usuarios). ALS puede aprovecharse eficientemente en sistemas distribuidos.

- ALS puede trabajar también con feedback implícito (no solo ratings).

Idea principal de ALS

- La idea es dividir un problema grande (por ejemplo, determinar una matriz de ratings) en problemas más pequeños y más manejables.

- En cada iteración, ALS mantiene un conjunto de variables constante mientras optimiza el otro.

- Luego, cambia el conjunto de variables que se mantiene constante y repite el proceso hasta que se alcanza la convergencia.

Formulación de ALS

$$L = \Sigma (R - U * V^T)^2 + \lambda(||U||^2 + ||V||^2)$$

Donde:

R es la matriz de ratings.

U es la matriz de features (latentes) de los usuarios.

V es la matriz de features (latentes) de los ítems.

$||.||^2$ representa la norma de Frobenius (suma de cuadrados de cada elemento de la matriz).

λ es un parámetro de regularización para prevenir el sobreajuste.

El algoritmo ALS procede fijando U y optimizando V, luego fija V y optimiza U, y repite hasta que converja. Como el problema es convexo en U y V por separado, cada uno de estos pasos de optimización encontrará un óptimo global.

Veamos un ejemplo básico (excel)

https://docs.google.com/spreadsheets/d/1uNk8PvnQMwQwwjiNZIrjAdN9MNf5KkW6NlpuLV1wcxM/edit?usp=sharing

Implementación de ALS en python

https://colab.research.google.com/drive/1heW5V0P_A6T8VrLnUvmZ6FQTNEFRVEXj?usp=sharing

Ventajas de ALS

- Puede manejar matrices grandes y dispersas, lo que lo hace útil para sistemas de recomendación con muchos usuarios y elementos.

- Los modelos de factorización de matrices, como los creados con ALS, a menudo proporcionan recomendaciones más precisas que otros tipos de sistemas de recomendación.

- Es paralelizable, lo que significa que se puede acelerar distribuyendo el cálculo en varios procesadores o máquinas.

Desafíos y desventajas de ALS

- Puede tener dificultades para manejar datos “*sparsed*” o cuando la matriz de ratings está incompleta.

- Es un método iterativo y puede ser computacionalmente intensivo.

- No puede manejar directamente la información contextual o características adicionales sobre usuarios y elementos.

Large-scale Parallel Collaborative Filtering for the Netflix Prize

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber and Rong Pan

HP Labs, 1501 Page Mill Rd, Palo Alto, CA, 94304
{yunhong.zhou, dennis.wilkinson, rob.schreiber, rong.pan}@hp.com

Abstract. Many recommendation systems suggest items to users by utilizing the techniques of collaborative filtering (CF) based on historical records of items that the users have viewed, purchased, or rated. Two major problems that most CF approaches have to resolve are scalability and sparseness of the user profiles. In this paper, we describe *Alternating-Least-Squares with Weighted-λ-Regularization* (ALS-WR), a parallel algorithm that we designed for the Netflix Prize, a large-scale collaborative filtering challenge. We use parallel Matlab on a Linux cluster as the experimental platform. We show empirically that the performance

Hallazgos del paper

- Uso de ALS en el concurso para netflix prize, obteniendo resultados competitivos.

- Combinaron su método ALS con otras técnicas, como SVD y el filtrado colaborativo basado en vecinos, para mejorar el rendimiento.

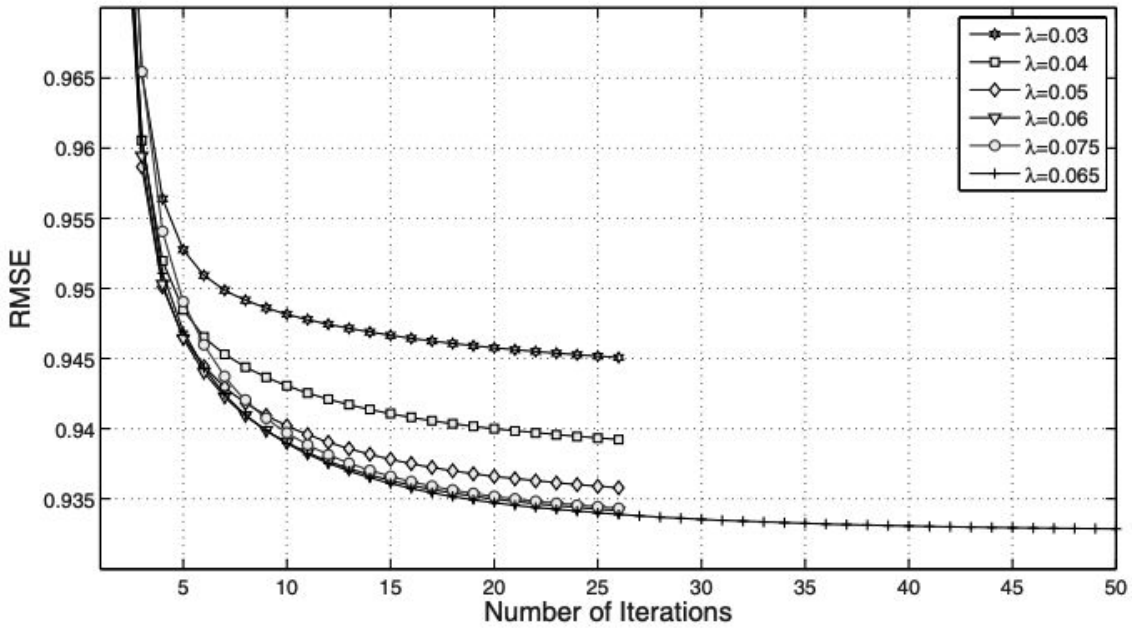


Fig. 1. Comparisons of different λ values for ALS-WR with $n_f = 8$. The best performer with 25 rounds is $\lambda = 0.065$. For this fixed λ, after 50 rounds, the RMSE score still improves but only less than 0.1 bps for each iteration afterwards.

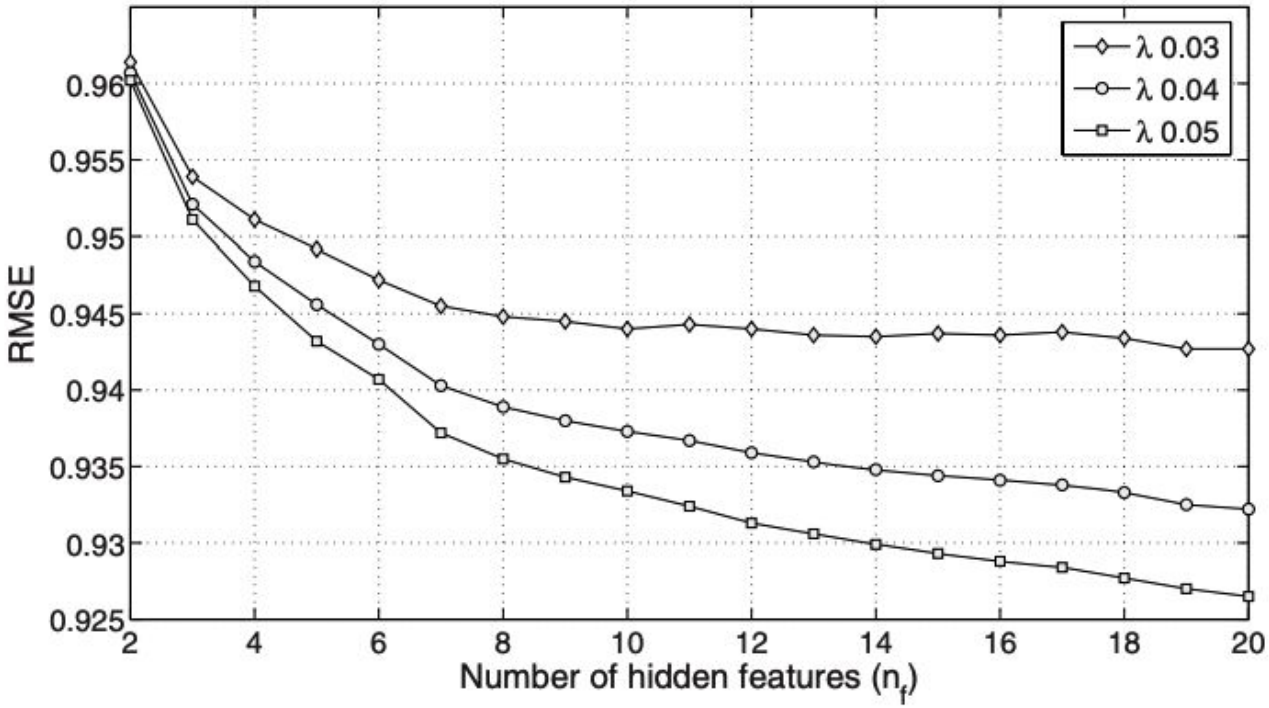


Fig. 2. Performance of ALS-WR with fixed λ and varying n_f