IIC3633 — Sistemas Recomendadores

Functional Matrix Factorizations for Cold-Start Recommendation

Juan Navarro

Referencia

Zhou, K., Yang, S., and Zha, H.
Functional matrix factorizations for cold-start recommendation.
In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011* (2011), pp. 315–324.

Problema: Recomendación a usuarios nuevos

Usuarios nuevos

- No hay información sobre el usuario: ratings
- La disponibilidad del usuario es limitada
- No todas los ratings posibles entregan la misma información

Una solución: Cuestionario de inicio

Preguntar al usuario sobre algunos items al comenzar

Tres respuestas posibles

- “Like”
- “Dislike”
- “Unknown”

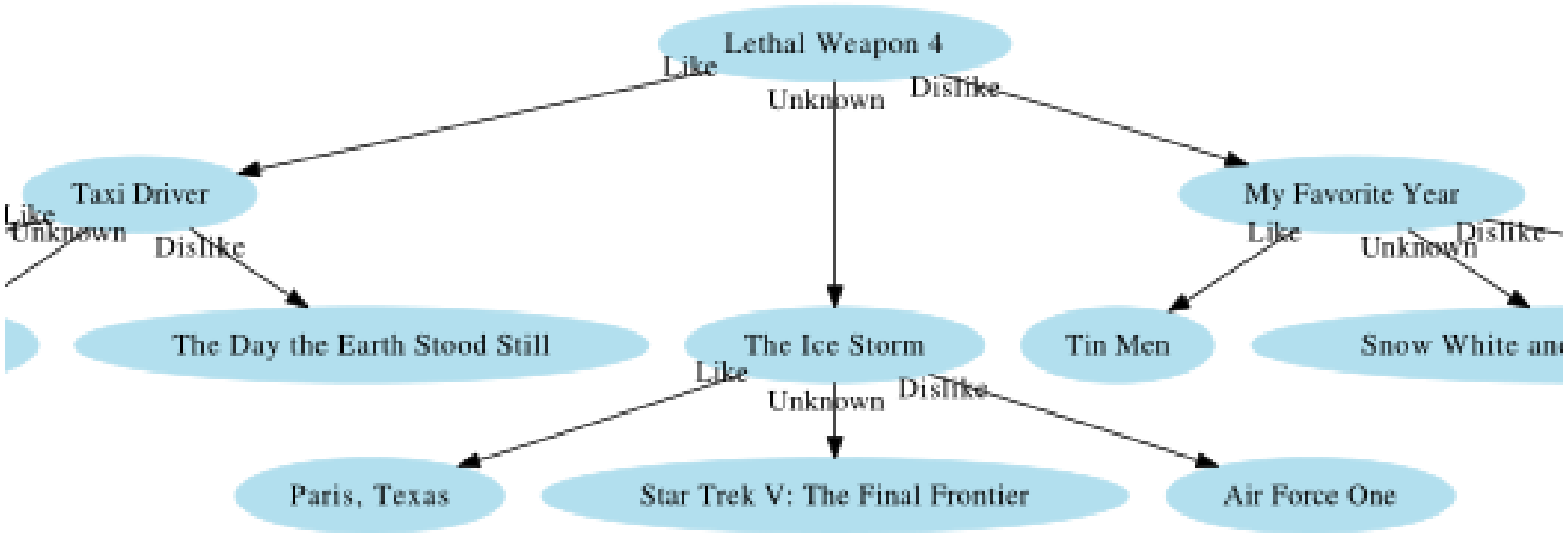
Una solución: Cuestionario de inicio

Características cuestionario

- No puede ser demasiado largo
- Debe entregar suficiente información
- Debe considerar distintos casos: el usuario podría desconocer los items preguntados
- El cuestionario debería ser dinámico: considerar respuestas anteriores para decidir la siguiente pregunta

Se puede modelar con un árbol de decisión

Ejemplo: Árbol de decisión



Construcción del árbol

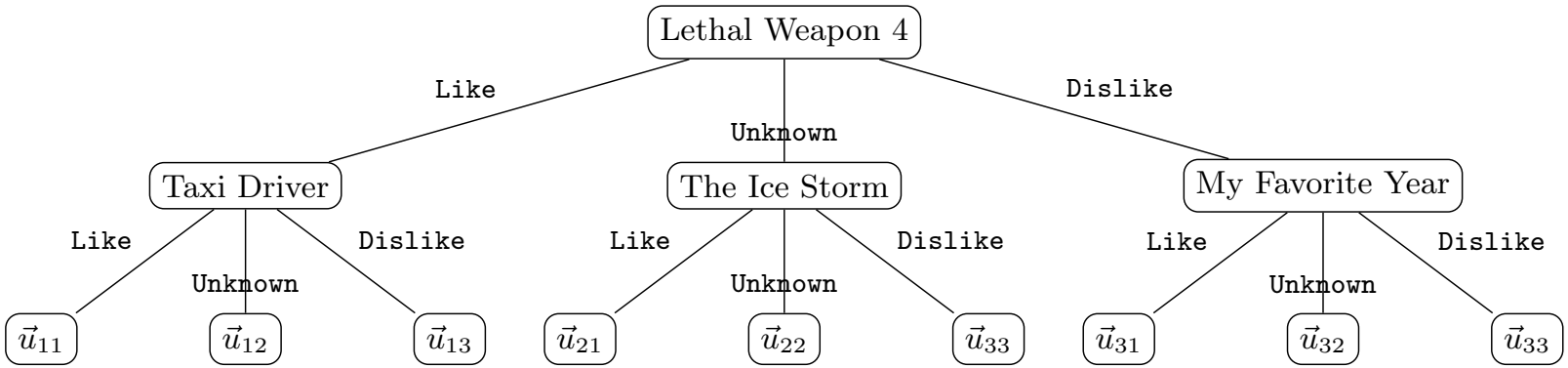
- Dos posibilidades principales: IGCN, optimizar directamente el error
- Se necesita estimar el error tras cada pregunta: Factorización matricial

Idea general

Usar factorización matricial para estimar el árbol de decisión y la matriz de factores latentes de los items.
El árbol de decisión transforma las respuestas del cuestionario a los perfiles de usuario.

Árboles de decisión como funciones

- Un árbol de decisión se puede entender como una función
- Procesa el input desde la raíz, y cada hoja tiene un resultado



Factorización matricial funcional

Factorización matricial estándar

$$U, V = \underset{U,V}{\text{argmín}} \sum_{(i,j)\in O} (r_{ij} - v_j^T u_i)^2 + \lambda \mathcal{R}$$

Factorización matricial funcional

$$T, V = \underset{T\in\mathcal{H},V}{\text{argmín}} \sum_{(i,j)\in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \| V \|^2$$

donde $\mathcal{H}$ es el espacio de cuestionarios, y a_i son las respuestas del usuario i.

Estimación de ratings

- En factorización matricial estándar:

$$r_{ij} = v_j^T u_i$$

- En factorización matricial funcional:

$$r_{ij} = v_j^T T(a_i)$$

El valor de $T(a_i)$ es el vector de la hoja respectiva.

Optimización: Alternating least squares

Alternar

1. Dado $T(\cdot)$, se encuentra v_j como:

$$v_j = \underset{v_j}{\text{argmín}} \sum_{(i,j)\in O} (r_{ij} - v_j^T T(a_i))^2 + \lambda \|v_j\|^2$$

2. Dado V, se busca $T(\cdot)$ tal que:

$$T = \underset{T\in\mathcal{H}}{\text{argmín}} \sum_{(i,j)\in O} (r_{ij} - T(a_i)^T v_j)^2$$

Optimización: Solución

El primer problema tiene una solución cerrada:

$$v_j = \left(\sum_{(i,j) \in O} T(a_i) T(a_i)^T + \lambda I \right)^{-1} \left(\sum_{(i,j) \in O} r_{ij} T(a_i) \right)$$

Pero el segundo no.

Estimación heurística: Construcción greedy

Construcción

Similar a greedy extend.

Idea

Comenzar desde la raíz, y en cada nodo elegir una pregunta (item), cuya respuesta genere el menor error estimado.

Construcción greedy del árbol

Se comienza con los valores v_j fijos. Por cada pregunta p, se particionan los usuarios en tres conjuntos:

$$R_L(p) = \{i | a_{ip} = \text{"}Like\text{"}\}$$
$$R_D(p) = \{i | a_{ip} = \text{"}Dislike\text{"}\}$$
$$R_U(p) = \{i | a_{ip} = \text{"}Unknown\text{"}\}$$

Construcción greedy

Se elige la pregunta siguiente según:

$$\min_{p} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u_L^T v_j)^2 +$$

$$\sum_{i \in R_D(p)} \sum_{(i,j) \in O} (r_{ij} - u_D^T v_j)^2 + \sum_{i \in R_U(p)} \sum_{(i,j) \in O} (r_{ij} - u_U^T v_j)^2$$

donde u_L, u_D, u_U son perfiles de usuario óptimos dentro de cada conjunto.

Construcción greedy: Perfiles de usuario

En el caso de u_L, se busca minimizar:

$$u_L = \underset{u}{\text{argmín}} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2$$

donde la solución cerrada es:

$$u_L = \left(\sum_{i \in R_L(p)} \sum_{(i,j) \in O} v_j v_j^T \right)^{-1} \left(\sum_{i \in R_L(p)} \sum_{(i,j) \in O} r_{ij} v_j \right)$$

u_D y u_U son análogos.

Resumen de la técnica

Inicializar V aleatoriamente, y luego:

1. Por cada item p, particionar los usuarios
2. Calcular el error dentro de cada partición
3. Elegir la pregunta con menor error total
4. Continuar recursivamente hasta alcanzar una profundidad fija
5. Fijar el árbol, y calcular V usando la forma cerrada
6. Iterar desde 1, hasta alcanzar una condición de término

Regularización jerárquica

- El árbol particiona los usuarios
- Al aumentar la profundidad, los nodos tienen cada vez menos usuarios
- Aumenta el impacto del overfitting

Solución propuesta

Regularizar en relación al nodo padre. Si u_C es el valor de u en el nodo padre, entonces:

$$u_L = \underset{u}{\text{argmín}} \sum_{i \in R_L(p)} \sum_{(i,j) \in O} (r_{ij} - u^T v_j)^2 + \lambda_h \|u - u_C\|^2$$

Evaluación experimental

Problemas estudiados

- Desempeño con usuarios nuevos, y comparación con baselines
- Impacto de valores desconocidos
- Desempeño *warm-start* en comparación a MF tradicional
- Sensibilidad de parámetros

Evaluación experimental: Datos

Se evaluó offline, usando RMSE.
Datos: MovieLens, EachMovie y Netflix.

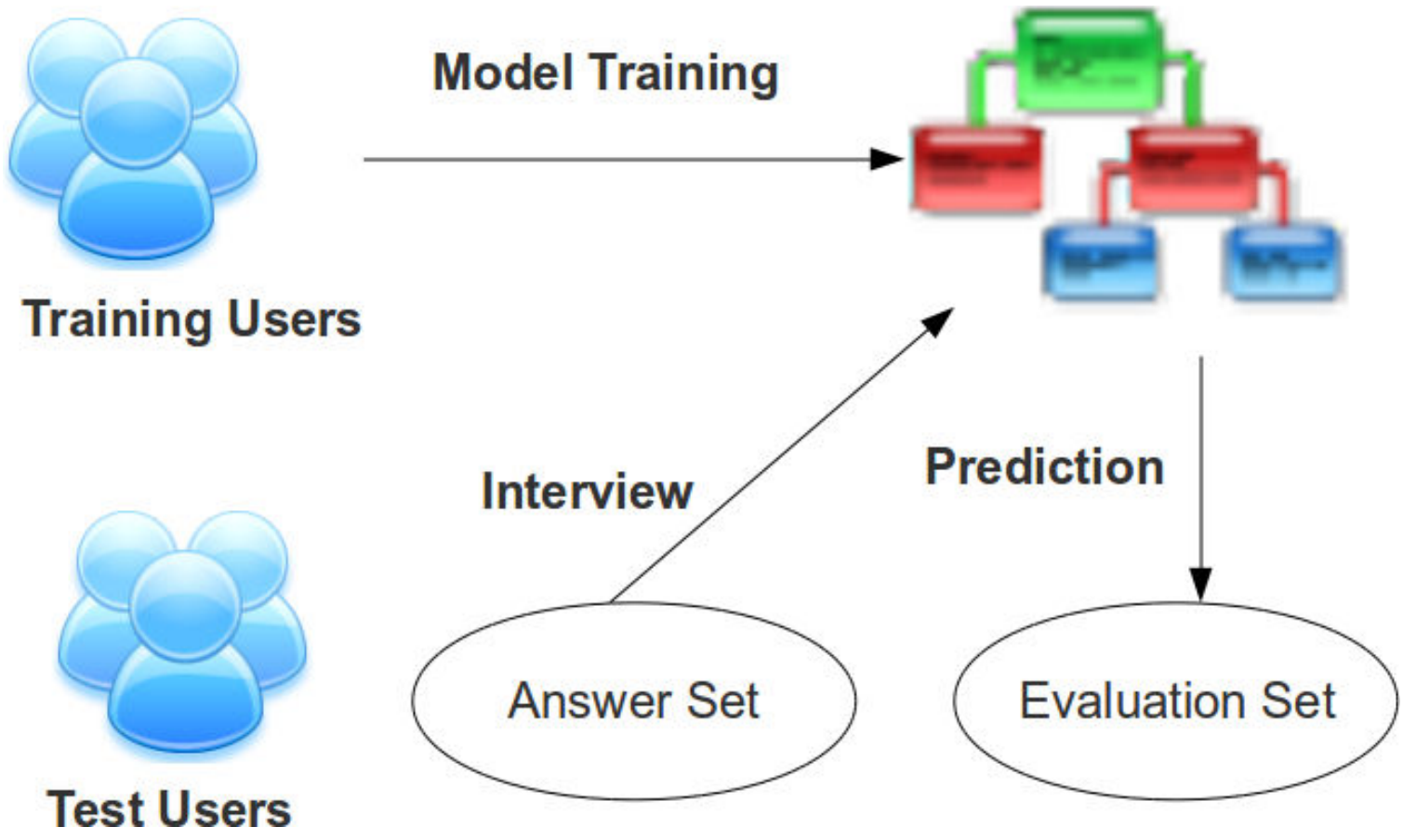
Respuestas desde ratings

$$a_{ij} = \begin{cases} 0 & (i,j) \in O \text{ y } r_{ij} \leq 3 \\ 1 & (i,j) \in O \text{ y } r_{ij} > 3 \\ \text{“}Unknown\text{”} & (i,j) \notin O \end{cases}$$

Evaluación experimental: Metodología

- Dos conjuntos de usuarios: training (75 %) y testing (25 %).
- Se usa Training para construir los modelos: árbol y V.
- Por cada usuario de Test se dividen los ratings en dos conjuntos: *answer set* (75 %) y *evaluation set* (25 %)
- Se simula el cuestionario con el *answer set*, y se evalúa contra el *evaluation set*.

Evaluación experimental: Metodología



Evaluación experimental: Baselines

Se utilizaron dos técnicas anteriores: **Tree**, **TreeU**.

Baselines: Tree

Idea principal

Construcción greedy. Se elige en cada nodo el item que minimice el error cuadrado.

$$Err_t(i) = e^2(tL) + e^2(tH) + e^2(tU)$$
$$splitter(t) \doteq \underset{i}{\text{argmín}}\ Err_t(i)$$

Aprovecha que tL y tH tienen menos elementos.

Referencia

Golbandi, Nadav, Yehuda Koren, and Ronny Lempel. "Adaptive bootstrapping of recommender systems using decision trees." Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011.

Baselines: TreeU

Idea general

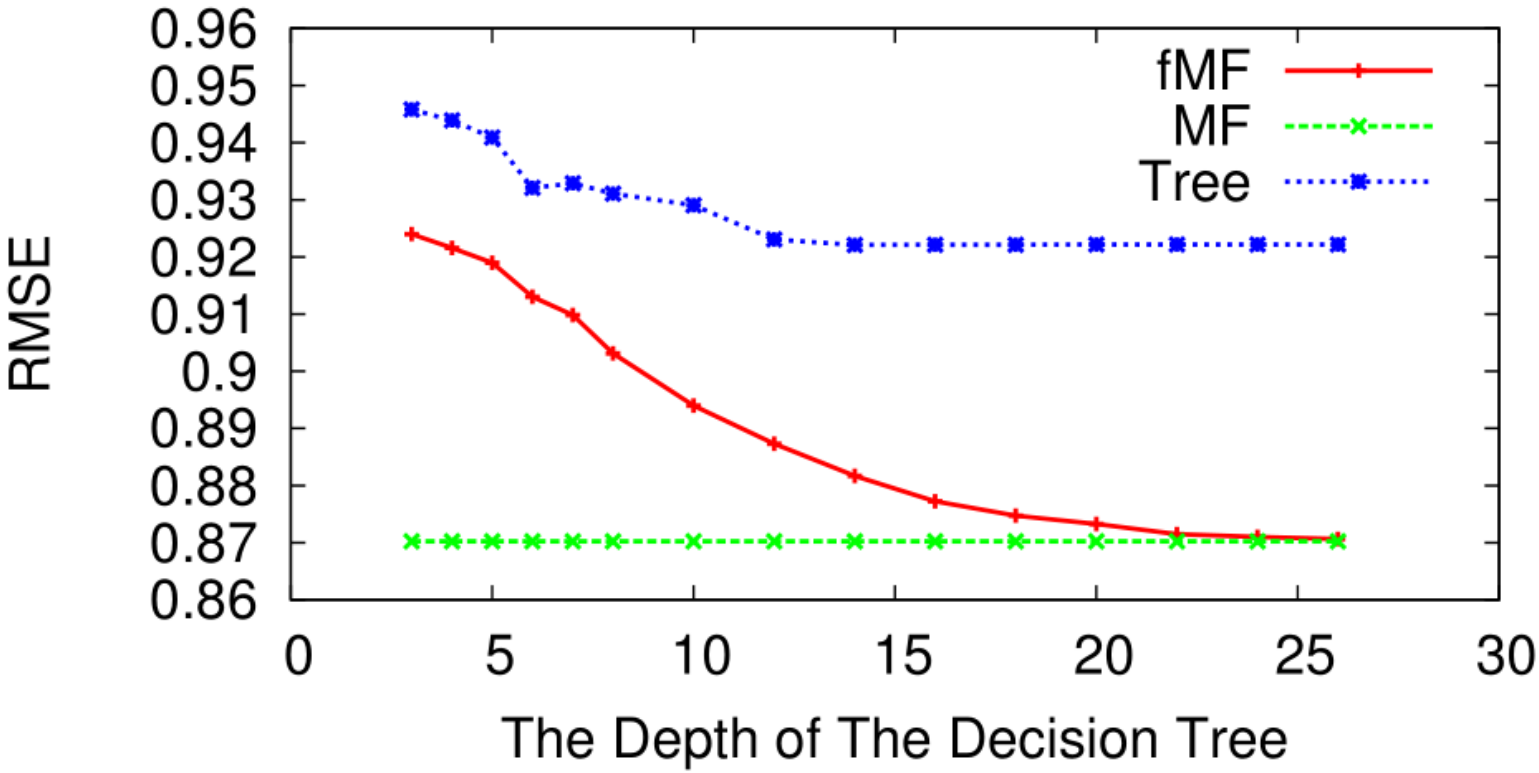
- Estimar perfiles de usuario y de item usando MF estándar
- Ajustar un árbol de decisión con el mismo algoritmo de Tree, pero sobre los factores latentes.

Resultados en caso Cold-Start

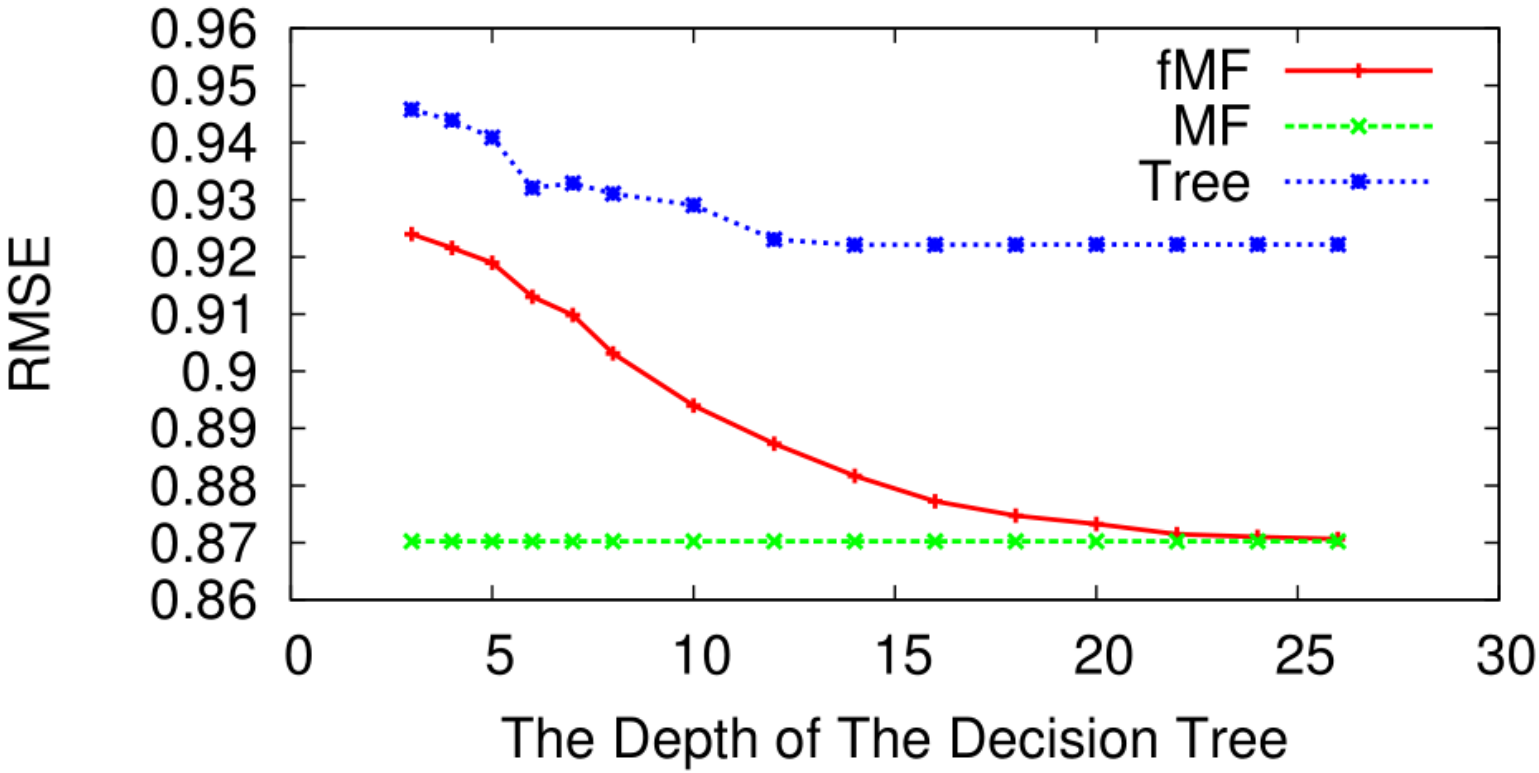
En MovieLens:

N. preguntas	3	4	5	6	7
fMF	0.9509	0.9480	0.9437	0.9429	0.9410
Tree	0.9767	0.9683	0.9615	0.9512	0.9523
TreeU	0.9913	0.9887	0.9789	0.9690	0.9557

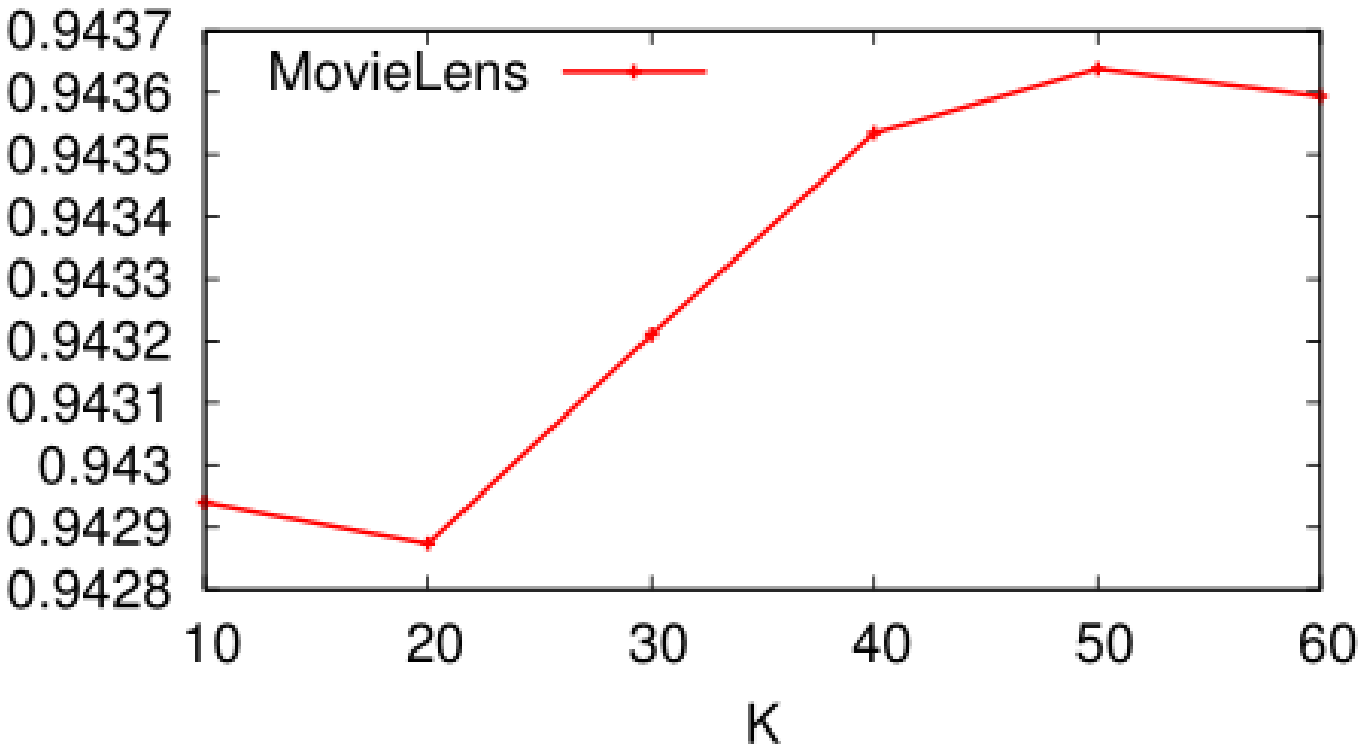
Resultados Warm-Start



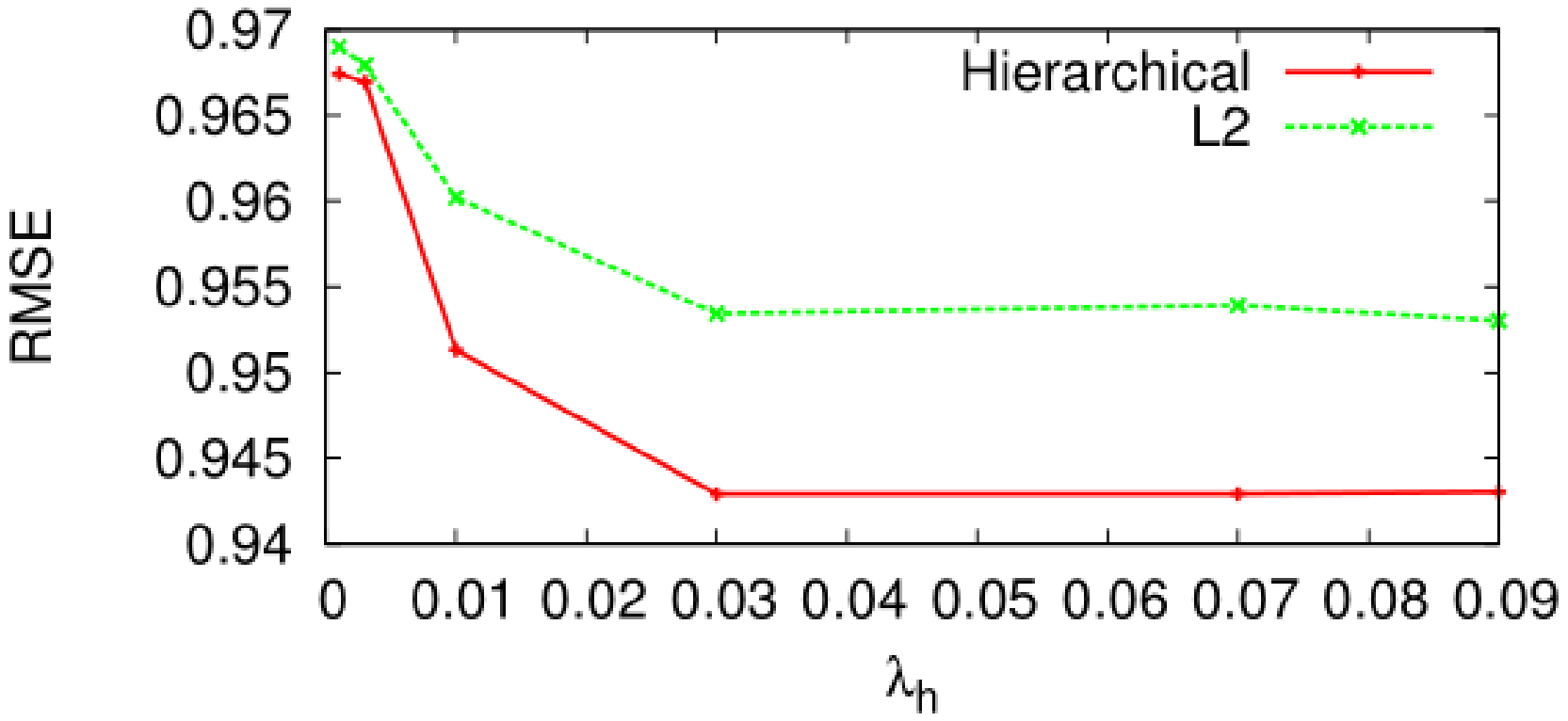
Sensibilidad parámetros: Profundidad del árbol



Sensibilidad parámetros: K



Sensibilidad parámetros: λ



IIC3633 — Sistemas Recomendadores

Functional Matrix Factorizations for Cold-Start Recommendation

Juan Navarro