Clase 217 — Factorización de matrices: SVD, ALS

Parte: 6 — Sistemas de Recomendación · Fuente: Koren, Bell, Volinsky Matrix Factorization Techniques for Recommender Systems (IEEE Computer, 2009) + Hu, Koren, Volinsky Collaborative Filtering for Implicit Feedback Datasets (ICDM 2008). Duración estimada: 80 min.

Clase 217 — Factorización de matrices: SVD, ALS

> *Parte: 6 — Sistemas de Recomendación · Fuente: Koren, Bell, Volinsky Matrix Factorization Techniques for Recommender Systems (IEEE Computer, 2009) + Hu, Koren, Volinsky Collaborative Filtering for Implicit Feedback Datasets (ICDM 2008). Duración estimada: 80 min.*

Objetivo

Reemplazar la matriz usuario-item dispersa por dos matrices densas de baja dimensión: R ≈ P × Q^T donde P (n_users × k) y Q (n_items × k). Aprender los embeddings k-dimensionales que capturan los factores latentes (género de película, gusto del usuario). Usar SVD (cuando hay datos completos) y ALS (Alternating Least Squares — robusto a sparse). Implicit-feedback ALS (Hu et al. 2008) es el algoritmo que ganó la mayoría de los Netflix Prize spin-offs en producción.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar el modelo r̂(u, i) = p_u · q_i + b_u + b_i + μ (con biases).
- Aplicar SVD truncado sobre matriz densa (poco realista, pero base teórica).
- Implementar ALS explicit (rating predicho) y ALS implicit (con confidence weighting c_ui = 1 + α × r_ui).
- Elegir hiperparámetros: factors (típicamente 20-200), regularization (λ para evitar overfitting), iterations (10-30).
- Usar implicit.AlternatingLeastSquares (Cython, multi-thread, rápido).

Temas

#	Tema	Por qué importa				
1	Modelo latente: fac	"Acción", "drama",				
2	SVD vs SVD trunca	SVD asume matriz				
3	Biases: μ, b_u, b_i	"Usuarios duros" y				
4	Implicit feedback +	0 no es "dislike"; p				
5	Regularización L2	`λ × (	p	² +	q	²) para evitar overfitting con k` alt
6	Cold-start parcial:	Para user nuevo: r̂				

Definiciones y características

- Factor latente: dimensión del embedding que captura una variable no observada (género, mood, demographic).
- SVD truncado: descomposición R = U Σ V^T quedándose con top-k singular values. Asume R completa (no es nuestro caso).
- ALS (Alternating Least Squares): fija Q, resuelve P por LSQ → fija P, resuelve Q → iterá. Cada subproblema es lineal y paralelizable.
- Modelo con biases: r̂(u, i) = μ + b_u + b_i + p_u · q_i. μ es la media global; b_u/b_i capturan el sesgo del usuario/item.

- Implicit feedback ALS: en vez de minimizar error sobre ratings observados, minimiza sobre TODA la matriz, ponderando por confidence c_ui = 1 + α × r_ui. Items con interacción tienen alta confianza de "1" (preferencia); sin interacción tienen confianza baja de "0".
- Regularización: L = Σ (r_ui - p_u · q_i)² + λ × (|p_u|² + |q_i|²). Sin esto, embeddings explotan a memorizar el train.
- Factors k: típicamente 20-200. Más → más capacidad y más overfitting. Tunear con validación.
- Funk SVD (Simon Funk, Netflix Prize 2006): el "SVD" popular del Netflix Prize NO es SVD sensu stricto — es factorización por SGD con regularización. El nombre quedó.

Dataset / recursos

- Explicit: MovieLens 1M con ratings 1-5.
- Implicit: Last.fm "lastfm-360K" o convertir MovieLens (rating ≥ 4 = "le gustó" = 1).
- Librerías: implicit>=0.7 (ALS rápido en Cython), scipy.sparse.linalg.svds, opcional surprise.

Ejercicios

1. SVD truncado: tomar matriz R densa imputando 0. U, sigma, Vt = svds(R, k=20). Reconstruir R̂ = U × diag(sigma) × Vt. Verificar que R̂ predice valores no-cero similares y "rellena" los ceros con guesses.
2. ALS explicit con Surprise: from surprise import SVD; algo = SVD(n_factors=50, n_epochs=20). algo.fit(trainset). Predict algo.predict(user, item). RMSE sobre test split.
3. ALS implicit con implicit: model = implicit.als.AlternatingLeastSquares(factors=64, regularization=0.05, iterations=15). model.fit(R_train * 40) (multiplicar por α=40). model.recommend(user_id, R_train[user_id], N=10).
4. Inspeccionar embeddings: PCA de model.item_factors a 2D y plot. Items "parecidos" deben caer cerca.
5. Biases analysis: imprimir top 10 movies por b_i (las que todos aman / odian) y top 10 usuarios por b_u.

Homework verificable

Notebook con:

1. Comparar 3 modelos sobre MovieLens 1M con leave-one-out por user:

- kNN item-based (Clase 216).
- Funk SVD (Surprise).
- ALS implicit (binarizando rating ≥ 4).

1. Reportar NDCG@10, recall@10, tiempo de entrenamiento.
2. Plot 2D (UMAP/t-SNE) de item_factors. Colorear por género — los géneros deberían formar clusters visibles.
3. Estudio de sensibilidad: k ∈ {10, 50, 100, 200} × λ ∈ {0.001, 0.01, 0.1} con cross-validation.
4. Recomendar para 3 users diferentes (un cinéfilo, un casual, un nuevo) — mostrar diferencia cualitativa.

Criterio de aceptación: ALS implicit gana en NDCG@10 sobre kNN; el grid de (k, λ) revela el sweet spot; los embeddings de items muestran estructura por género al PCAear.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
svds da componentes con signo arbitrario	SVD no fija el signo de los singular vecto
ALS converge a embeddings constantes	λ muy alto. Fix: bajar regularización (0.0
MemoryError con ALS sobre 10M users × 1M i	Implementación naive. Fix: implicit librar
Implicit-ALS overfit: recomienda solo lo q	Sin confidence weighting o con α muy alto.
Embeddings no muestran estructura semántic	k muy chico (k=5 no capta nada) o train se
Predicciones constantes para todos los use	Modelo bajó a solo predecir biases. Fix: c
Cold-start total: nuevo user → score=0	p_u no existe para user nuevo. Fix: fallba

Preguntas frecuentes

¿SVD del Netflix Prize es SVD matemático?

No. Lo que ganó Netflix era factorización por SGD con regularización ("Funk SVD"), no SVD-decomposition real. El nombre pegó por marketing.

¿implicit o lightfm o tensorflow recommenders?

- implicit: ALS implicit + BPR. Rápido (Cython), implicit-only. Default para CF moderno.
- lightfm: hybrid (CF + content features). Bueno si tenés metadata (Clase 219).
- tensorflow recommenders / recsim: deep learning, two-towers, sequential. Para escala grande + features ricas.

¿Qué k (factors) elijo?

Típicamente 50-200. Más k requiere más data + más regularización. Si tu dataset es chico (<100K interactions): k=20-50. Si tenés millones de interactions: k=100-200. Cross-validation es la única respuesta correcta.

¿ALS vs SGD para optimizar?

- ALS: cada subproblema es LSQ lineal → paralelizable, converge en pocas iteraciones. Bueno cuando matrix cabe en memoria y querés multi-core.
- SGD: por minibatch, on-line, escala a streaming. Más sensible a learning rate.

implicit usa ALS; PyTorch implementations típicamente usan SGD.

¿Embeddings se pueden usar para más cosas?

Sí. Aplicaciones: (1) similar items (cosine(q_i, q_j)), (2) clustering de items, (3) features para modelos downstream (CTR prediction), (4) two-tower retrieval (precomputar q_i, buscar nearest neighbor de p_u en producción con FAISS).

¿Y deep learning recommenders (NCF, deep CF)?

Two-towers + transformers (BERT4Rec, SASRec) son SOTA para sequential recommendation. Pero: ALS implicit sigue compitiendo en producción por simplicidad, latencia y costo. Empezá con ALS; pasá a DL si la métrica lo justifica.

Referencias

- Koren, Y., Bell, R., Volinsky, C. Matrix Factorization Techniques for Recommender Systems (IEEE Computer, 2009) — el clásico Netflix Prize.

- Hu, Y., Koren, Y., Volinsky, C. Collaborative Filtering for Implicit Feedback Datasets (ICDM 2008) — implicit ALS.
- implicit library — ALS + BPR en Cython.
- Funk SVD blog (Simon Funk, 2006) — el blog que cambió la historia.
- Recommender Systems: An Introduction (Jannach et al., 2010).

Siguiente clase

Clase 218 — Content-based filtering

Apéndice: notebook (primer bloque)

Requiere: pip install scipy scikit-learn implicit. Sobre dataset sintético similar a MovieLens 100K.

```
import numpy as np
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import svds

rng = np.random.default_rng(42)
n_users, n_items, K_TRUE = 1000, 500, 8

# Generamos datos con estructura latente conocida (k=8) para validar que MF recupera estructura
P_true = rng.normal(0, 1, (n_users, K_TRUE))
Q_true = rng.normal(0, 1, (n_items, K_TRUE))
R_full = P_true @ Q_true.T   # rating ideal sin ruido

# Solo observamos 8% de las interacciones (sparse)
mask = rng.random((n_users, n_items)) < 0.08
R_obs = np.where(mask, np.clip(R_full + rng.normal(0, 0.3, R_full.shape), -3, 3), 0)
print(f'observed: {mask.sum():,} / {n_users * n_items:,} ({mask.mean():.2%})')
```

Archivos complementarios

- notebook.ipynb