Part 1. Classical Approaches To Recommendation

Olivier Koch - Ugo Tanielian - Zofia Trstanova

CAIL

July 25, 2019

Overview

Collaborative Filtering and the Netflix Challenge

Collaborative Filtering

The whole goal: personnalized recommendations !

- Assume you are a company selling items (clothes, movies, ...) and you want to recommend items that users are likely to want.
- You collect millions of ratings (explicit) /observations (implicit) from users on different items
- From these interactions between users and items, you have to:
  - predict the utility of a certain item for a particular user
  - suggest new items by finding the most relevant items for each user.

Matrix Factorization Techniques for Recommender Systems, Y. Kuren, R. Bell, C. Volinsky

Lessons from the Nextflix Prize Challenge

The Task

- Open competition to beat the Netflix baseline (CineMatch)
- Training set: 100,480,507 ratings (480,189 users, 17,770 movies)
- Qualifying set: 2,817,131 ratings (50% test set, 50% quiz set)
- Improve by 10% to win $1,000,000

The Lessons

- A trivial solution is almost as good as the baseline
- Use methods combining neighborhood-based methods and MF
- Incorporate side information (e.g. time of rating)

Lessons from the Netflix Prize Challenge, R. Bell and Y. Koren, SigKDD 2007

More formally

Given:

- $\mathbb{U} = (u_1, ..., u_n)$, a group of n users
- $\mathbb{I} = (i_1, ..., i_m)$, a group of m items
- A set of interactions $(u, i, R_{u,i}) \in \mathbb{U} \times \mathbb{I} \times \mathbb{R}^+$ between users and items split between a training set $\mathcal{T}_r$ and a test set $\mathcal{T}_e$

Task: Find the k most relevant items for each user

The Metrics: MPR

The Percentile Rank (PR) of the target item i is defined by:

$$\mathrm{PR}_i = \frac{\sum_{i\prime \in \mathbb{I}} 1_{(i \geq i\prime)}}{|\mathbb{I}|} \times 100\%$$

- 1_Z: indicator function of the event Z
- $|\mathbb{I}|$: number of items in the candidate set

  Mean Percentile Rank (MPR) is the average PR of all the instances in the test-set $\mathcal{T}_e$:

$$\mathrm{MPR} = \frac{\sum_{(u,i) \in \mathcal{T}_e} \mathrm{PR}_i}{|\mathcal{T}_e|}$$

The Metrics: Precision@k

$$\text{precision@}k = \frac{\sum_{(u,i)\in\mathcal{T}_e} 1_{(\text{rank}_i \le k)}}{|\mathcal{T}_e|}$$

- rank_i: predicted rank of the target item i
- Precision@k is the fraction of instances in the test set for which the predicted rank of the held-out item falls within the top k predictions

The Challenges

- **Data Sparsity:** a given user might have interacted with a very low % of items. Difficulty of the cold start problem.
- **Scalability:** the number of users, items might reachs billions...
- **Diversity:** difficulty of tackling *grey sheeps*...
- **Causality:** most recommender systems do not try to correct the bias coming from the *logging policy*

Neighborhood methods

Neighborhood methods

Neighborhood methods leverage on big datasets to find similarities between users or items to fill the missing values.

- User-oriented: find similar users. It suffers from scalability. User-oriented algorithms require computation that grows with both the number of users and the number of items.

- Item-oriented: find similar items and therefore avoid the bottleneck of the user-based approach.

Item-Based Collaborative Filtering Recommendation Algorithms, B. Sarwar, G. Karypis, J. Konstan and J. Riedl, 2001

Item-based recommendation

To predict the rating $R_{u,i}$ of an unobserved pair (user, item) (u, i):

- Among all items rated by u, find the k most similar items to i
- Weight each rating by the similarity between these items and i

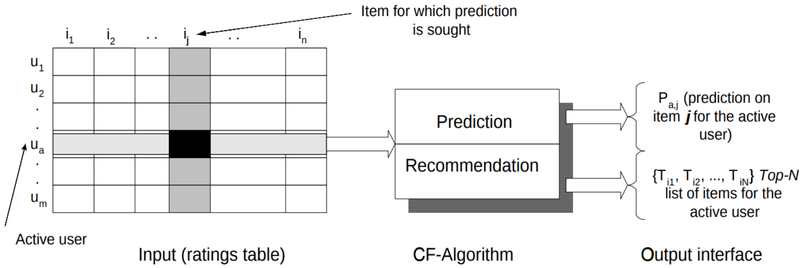
$$R_{u,i} = \frac{\sum_{\text{top similar items}} s_{i,j} \times R_{u,j}}{\sum_{\text{top similar items}} s_{i,j}}$$

- The similarity between two items can be cosine-based, correlation-based or with adjusted-correlation

$$s_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\|}$$

Item-based top-n recommendation algorithms, M.Deshpande and G. Karypis. Trans. Inf. Syst. 2004.

Item-based recommendation


Item for which prediction is sought
i_1 i_2 . . i_j . . i_n
u_1 u_2 . . u_a . . u_m
Active user
Input (ratings table)
Prediction
Recommendation
CF-Algorithm
$P_{a,j}$ (prediction on item *j* for the active user)
$\{T_{i1}, T_{i2}, ..., T_{iN}\}$ *Top-N* list of items for the active user
Output interface

Item-based collaborative filtering recommendation algorithms, B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl. WWW'2001.

Matrix Factorization methods

Principles of Matrix Factorization

Matrix factorization methods have received great exposure and can be used as an unsupervised learning method for latent variable decomposition.

- Learn a latent factor model from the data and then use the discovered factors to find items with high expected ratings.
- Low-Rank factorization
- Dimensionality Reduction

Neighborhood vs Matrix Factorization

- Neighborhood models detect localized relationships
- MF captures the totality of weak signals contained in the matrix
- Both methods can be fused (e.g. SVD++)
- Both methods have trouble making predictions for users with few ratings

Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, Y. Koren, KDD'08

Singular Value Decomposition

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

where **U** and **V** contain the left- and right-singular vectors of M.

- SVD can be used to compute optimal low-rank approximations (best possible approximation w.r.t. Frobenius norm).
- SVD (Singular Value Decomposition) is closed form, requires a full-filled matrix (biased towards unseen observations)
- Classic SVD cannot handle missing values
- On the opposite, Matrix Factorization uses optimization and has been shown to work better (e.g. on known entries only)

Unconstrained Matrix factorization

- Find a low rank approximation to a sparse ratings matrix by minimizing a loss function on the known ratings
- Popular choice for applications in industry
- Simple, highly scalable to large datasets (thanks to sparsity)
- Captures global signals (but fails to capture strong local signals)

Steps for matrix factorization

- Build user-item matrix
- Define factorization model (cost function)
- Remove global mean
- Remove item bias and user bias
- Add regularization to prevent overfitting
- Minimize cost function (e.g. stochastic gradient descent, alternating least squares)

Definition of the objective in MF

- Objective: Find U and V such that:

$$\min_{U,V} \sum_{(u,i)\in k} (R_{u,i} - \mu - b_i - b_u - U_i^T V_u)^2$$

- Convexification of the objective using regularization:

$$\min_{U,V} \sum_{(u,i)\in k} (R_{u,i} - \mu - b_i - b_u - U_i^T V_u)^2 + \lambda(\|U_i\|^2 + \|V_u\|^2 + b_i^2 + b_u^2)$$

where λ is an hyper-parameter.

Recap: hyper-parameters in Matrix Factorization

- cost function
- vector size
- learning rate
- regularization
- optimization method

Other variants of MF

Non-Negative Matrix Factorization

NMF takes into the fact that most real-world data (images, videos, recommender systems) are non negative.

Idea: Maintain such non negativity constraints in the factorization process.

We factorize the rating matrix with two low-rank matrices

$$R_{u,i} = U^T_{n \times k} * V_{k \times m}$$

where $U_{n \times k}$ is the user matrix and $V_{k \times m}$ the item matrix.

All elements in the three matrices R, U and V are therefore positive.

Algorithms for Non-negative Matrix Factorization, D.D. Lee and H.S Seung, NIPS 2001

Matrix Factorization for Implicit Feedback

- Model the probability that a user will interact with a given item with a logistic function
- Weigh zero-entries
- Optimize log likelihood with Alternating gradient descent
- Outperforms regular MF on small vectors (Spotify dataset)

Collaborative Filtering for Implicit Feedback Datasets, Y. Hu and Y. Koren and C. Volinsky

Probabilistic Matrix Factorization

- Scales linearly with number of observations
- Can incorporate the fact that users who rate similar sets of movies have similar preferences
- Evaluation on the Netflix dataset

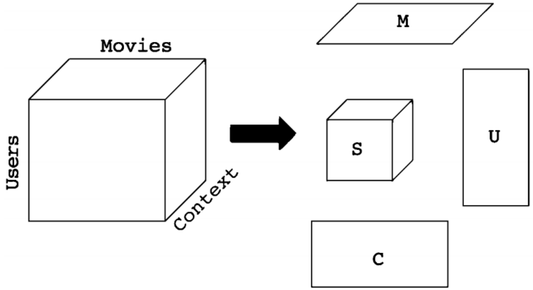
$$p(R \mid q, p, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij} \mid U_i^T V_j, \sigma^2)\right]^{I_{ij}}$$

Probabilistic Matrix Factorization, R. Salakhutdinov and A. Mnih, NIPS 2007

Hybrid Methods

Hybrid methods: use of contextual side information

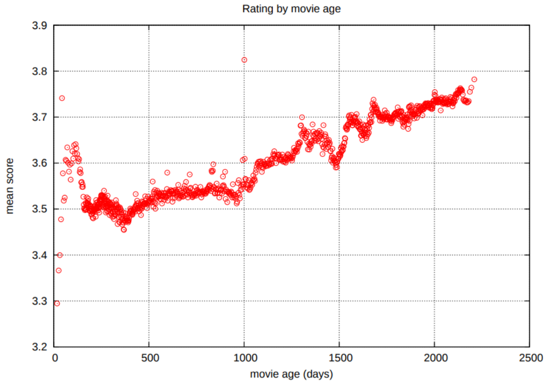
- High-order SVD optimized on observed entries only
- Batch sub-space descent $\rightarrow$ SGD for larger datasets



Multiverse Recommendation: N-dimensional Tensor Factorization for Context Aware Collaborative Filtering, Karatzoglou A. Amatriain X. Baltrunas L. Oliver N. RecSys 2010

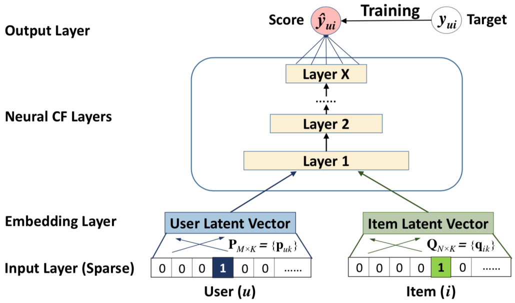
Hybrid methods: Temporal dynamics

- Including temporal dynamics in user and item behaviors
- Applicable to neighborhood-based and factorization methods



Collaborative Filtering with Temporal Dynamics, Koren Y., SigKDD 2009

Hybrid methods: Neural Matrix Factorization



- Instead of the dot product of the user and the item embedding, concatenate the embeddings and use them as features for the neural network

Neural Collaborative Filtering, X. He et al, 2017

Lessons from the Real World

- What problem are you trying to solve? Predicting DVD rentals is different from predicting instant streaming selection.
- Pick a single solution (one that fits your problem)
- There is often a deceptively simple baseline solution
- Improve on it incrementally
- Evaluate online early

The Word2Vec model

Context

- Two-layer neural network trained to reconstruct a word context
- Motivation: explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.

Distributed Representations of Words and Phrases and their Compositionality, T. Mikolov et al, NIPS 2013

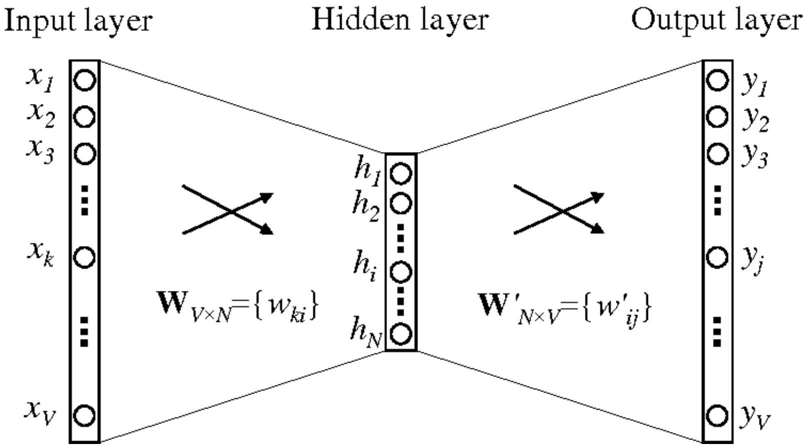
The word2vec model

Given a sequence of training words $w_1, w_2, \cdots, w_T$, maximize:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} \mid w_t)$$

where c is the size of the training context

W2V



Input layer
Hidden layer
Output layer
x_1
x_2
x_3
x_k
x_V
h_1
h_2
h_i
h_N
y_1
y_2
y_3
y_j
y_V
$\mathbf{W}_{V\times N}=\{w_{ki}\}$
$\mathbf{W}'_{N\times V}=\{w'_{ij}\}$

The word2vec model

We note E the embedding matrix and O the output layer. Intuitively, E codes the user and O the item.

For a given pair (u, i), we use softmax to model the conditional probabilities $p(u \mid i)$. Therefore, we want to maximize:

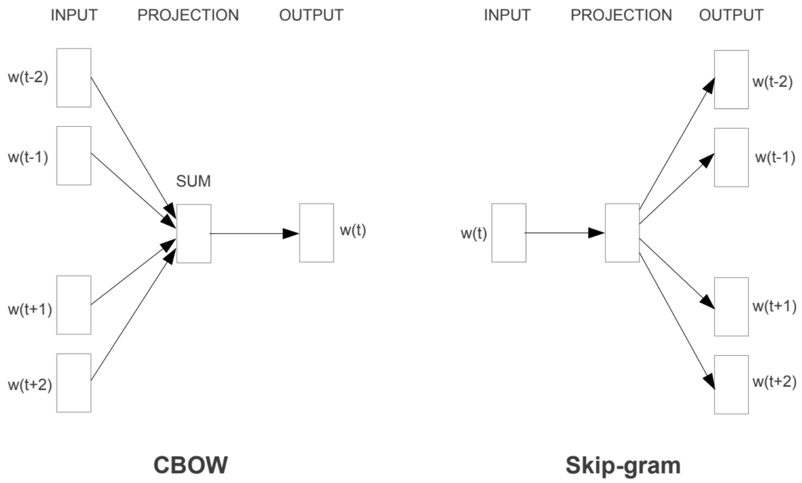
$$p(u \mid i) = \frac{exp(E_u^T O_i)}{\sum_{i' \in \mathbb{I}} exp(E_u^T O_{i'})}$$

Skip-gram and Continuous Bag-of-Word (CBOW)

The word2vec architecture comes in two flavors:

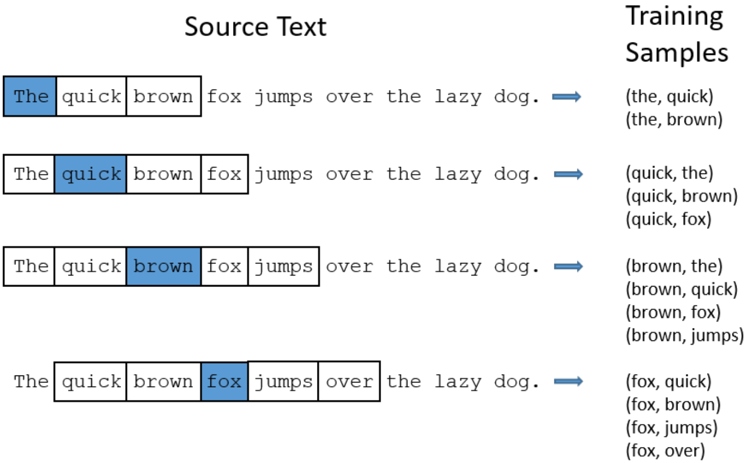
- CBOW: predict the current word given the context
- Skip-gram: predict the context given the current word

Skip-gram vs CBOW



Efficient Estimation of Word Representations in Vector Space, ICLR 2013

Word2vec skip-gram training example



Drawbacks of softmax and candidate methods

- Computing the softmax is quite expensive:
  - Needs computing scores for $(u, i), \ \forall i \in \mathbb{I}$
  - Implies back-propagating on many parameters θ
- To avoid this problem, we can restrict the number of negatives we keep. Usually, we do this by sampling in a *Negative Sampling distribution*.
- *Negative Sampling loss*, it is a classification/discriminative loss :

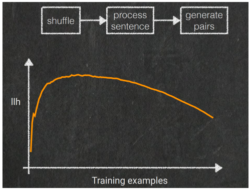
$$\underset{\theta \in \Theta}{\text{argmax}} \ \mathbb{E}_{(u,i) \sim \mathcal{T}_r} \quad \log \sigma(E_u^T O_i) + \mathbb{E}_{i_1,\dots,i_n \sim Q} \ \log -\sigma(E_u^T O_{i_k}))$$

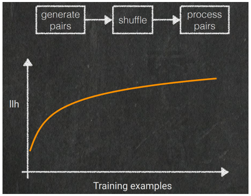
Many other applications

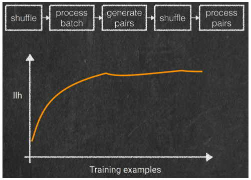
- NLP: word representation (word2vec), text representation (doc2vec)
- Recommendation: prod2vec or user2vec: consider products as words and sequences of interactions as sentences (skip-gram word2vec). Can bring significant uplift on CTR and YR (yield rate) compared to popular products
- Graph Theory: graph2vec and node2vec
- Others: author2vec, tweet2vec, sense2vec ...

M. Grbovic et al, E-commerce in your inbox: Product recommendations at scale. SigKDD 2015

Word2vec at scale: preprocessing matters...







Feeding Word2vec with tens of billions of items, what could possibly go wrong?, S. Dolle, Berlin Buzzwords, 2015