Recommender Systems

E. Le Pennec

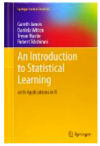
MAP541 - Machine Learning 2- Winter 2022-2023

Outline

Outline

Recommender Systems

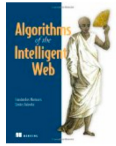
Recommended for You



An Introduction to ...
› Daniela Witten
★★★★★ (55)
~~$79.99~~ $73.58
Why recommended?

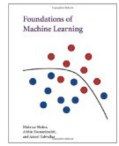
Interactive Data ...
› Scott Murray
★★★★½ (42)
~~$39.99~~ $26.85
Why recommended?

Data Smart: Using ...
› John W. Foreman
★★★★½ (66)
~~$45.00~~ $30.02
Why recommended?



Algorithms of the ...
› H. Marmanis
★★★★☆ (16)
~~$44.99~~ $29.13
Why recommended?

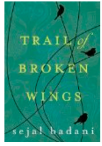
Scala for Machine ...
Patrick R. Nicolas
★★★★½ (6)
~~$59.99~~ $53.99
Why recommended?



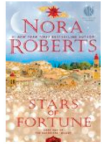
Foundations of Machine ...
› Mehryar Mohri
★★★★☆ (7)
~~$74.00~~ $66.60
Why recommended?

Hot New Releases in Kindle eBooks

New Release
The Stranger
Harlan Coben

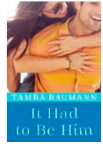


New Release
Trail of Broken Wings
Sejal Badani



Stars of Fortune: Book ...
Nora Roberts
$7.99

New Release
Boundary Crossed ...
Melissa F. Olson

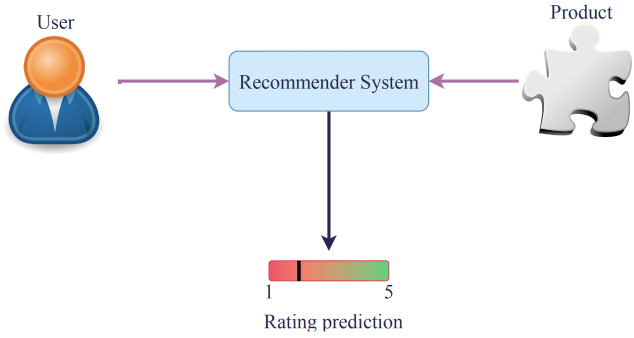


New Release
It Had to Be Him (An ...
Tamra Baumann

New Release
This Thing Called ...
Miranda Liasson

Source: Amazon

Recommender Systems



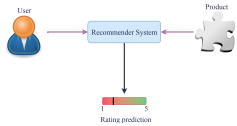
Recommender Systems

- Predict a rating for pairs of user/product,
- Use this to rank the products and suggest them to the user.

- May predict only a ranking...

Source: V. Guigue, LIP6

Data at Hands



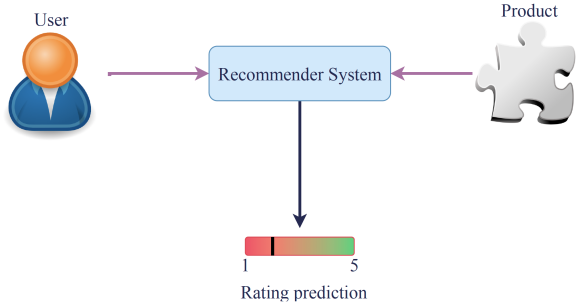
Basic observation: Triple or Pair

- Triple User/Item/Rating: (U, V, R)
- Natural interpretation as pair of User-Item/Rating: $((U, V), R)$
- Similar to the supervised setting!

Data at Hands

- Collection of pairs $((U_i, V_i), R_i)$
- User U may rate several items V and item V may be rated by several users U.
- Not in the classical i.i.d. setting because the item ratings by an user are not independent!

Goals



Goals

- Given a user U and an item V, predict the rating R.
- Rank the items V for a given user U.
- Suggest an item V to a given user U.

- We will focus on the first question!

Source: V. Guigue, LIP6

Some Issues

User

- What is a user? An id? A detailed profile?
- What about a new user?

Item

- What is an item? An id? A detailed description? A set of features?
- What about a new item?

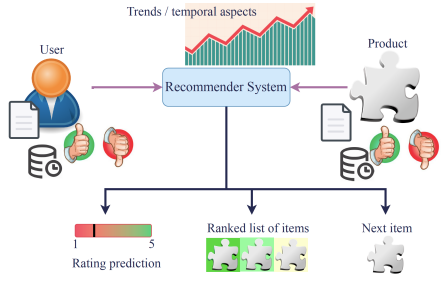
Rating

- Can we believe them?
- How to measure the error? Using the Euclidean norm?

- We will cover this...

Source: V. Guigue, LIP6

More Issues



More Issues

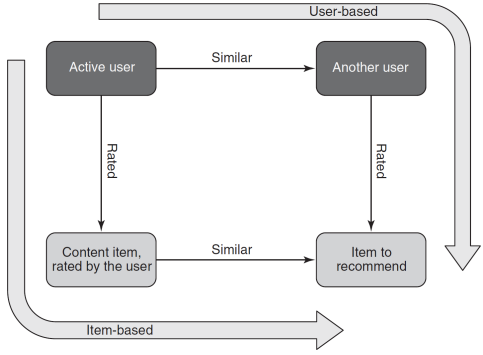
- How to take into account the temporality?
- How to take into account indirect feedbacks?
- How to propose directly a ranking?

- We won't cover that...

Source: V. Guigue, LIP6

Outline

Collaborative Filtering



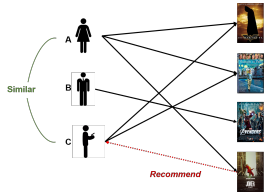
Collaborative Filtering

- Use similarity between users or items to predict ratings.

- Similar idea than in supervised learning.

Source: K. Falk

User-based Filtering



User-based Filtering

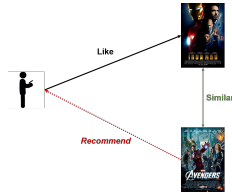
- Given a target pair of user/item (U, V).
- Choose a similarity measure $w(U, U')$ between users.
- Define a neighborhood $\mathcal{N}(U)$ of *similar* users U_i having rated V, i.e. $V_i = V$.
- Compute a predicted rating by

$$\widehat{R} = \frac{\sum_{U_i \in \mathcal{N}(U)} w(U, U_i) R_i}{\sum_{U_i \in \mathcal{N}(U)} w(U, U_i)}$$

- Choice of similarity and neighborhood will be discussed later.

Source: B. Kim

Item-based Filtering



Item-based Filtering

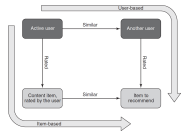
- Given a target pair of user/item (U, V).
- Choose a similarity measure $w'(V, V')$ between items.
- Define a neighborhood $\mathcal{N}(V)$ of *similar* items V_i rated by U, i.e. $U_i = U$.
- Compute a predicted rating by

$$\widehat{R} = \frac{\sum_{V_i \in \mathcal{N}'(V)} w'(V, V_i) R_i}{\sum_{V_i \in \mathcal{N}'(V)} w'(V, V_i)}$$

- Choice of similarity and neighborhood will be discussed later.

Source: B. Kim

Similarities and Neighborhood?



Similarities Based on Known Features

- Same setting than kernel density technique in supervised/unsupervised learning.

Similarities Based on Ratings

- Similarity based on (common) rated items/users.

Neighborhood

- Same setting than kernel density technique in supervised/unsupervised learning.
- Most classical approaches:
  - local – k closest neighbors or neighbors whose similarity is larger than a threshold...
  - non-local – based on a prior clustering of the users (items).

Source: K. Falk

Reminder on Similarity Measures

L^p Distance

- Formula:

$$d_p(X, X') = \left(\sum_{j=1}^{d} (X^{(i)} - X'^{(i)})^p \right)^{1/p}$$

- Renormalized version:

$$d_p(X, X') = \left(\frac{1}{d} \sum_{j=1}^{d} (X^{(i)} - X'^{(i)})^p \right)^{1/p}$$

Inverse Distance and Exponential Minus Distance

- Inverse Distance: $1/d(X, X')$
- Exponential Minus Distance: $\exp(-d(X, X'))$
- Distance may be raised to a certain power.

Reminder on Similarity Measures

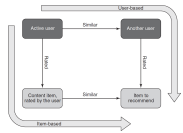
Cosine Similarity

- Formula:

$$\cos(X, X') = \frac{\sum_{j=1}^{d} X^{(i)} X'^{(i)}}{\left(\sum_{j=1}^{d} \left(X^{(i)}\right) 2\right)^{1/2} \left(\sum_{j=1}^{d} \left(X'^{(i)}\right) 2\right)^{1/2}}$$

- All those formulas require a coding of categorical variables.
- Other similarities exist!

Similarities Based on Features



Classical Features

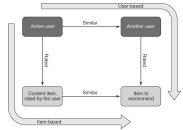
- Usual (difficult) supervised/unsupervised setting!
- (Inverse/Exponential Minus) Distance,...

Content Based Approach

- User/Item described by a text.
- NLP setting.
- Often based on a bag-of-word / keywords approach.
- (Inverse/Exponential Minus) Distance, Cosine,...

Source: K. Falk

Similarities Based on Ratings



- **Not necessarily the same number of ratings for different users or items!**

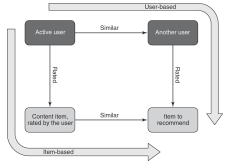
Similarity Based on Ratings

- Similarity based on the vector of rating of common rated items/rating users.
- Renormalization needed.
- (Inverse/Exponential Minus) Renormalized Distance, Cosine,...

- All the similarities can be combined...

Source: K. Falk

Local Neighborhood

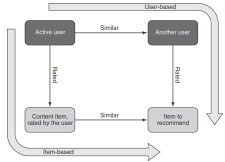


Top k / Threshold on Similarity

- Precompute the similarity for each pair of users (items) sharing an item (user)
- For any user U and item V, define the user (item) neighborhood as the k most similar users (items) sharing item V (user U) or the ones with similarity above the threshold.
- Localized neighborhood as in nearest neighbors in supervised learning.

Source: K. Falk

Non-local Neighborhood



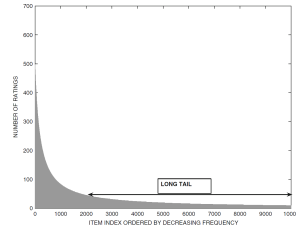
Prior Clustering

- Precompute a clustering of the users (items).
- Use the group to which user U (item V) belongs as initial neighborhood.
- Restrict it to the users (items) sharing the item V (user U)
- Non-local neighborhood as in partition based method in supervised learning.

- Strong connection with classical marketing approach!

Source: K. Falk

Ratings Issues



Ratings Issues

- User rating bias: different users may have different rating scale.
- Long tail phenomena: different users (items) may have very different number of ratings (and most users (items) have few)

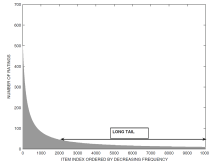
Sources: Unknown/C. Aggarwal

User Bias

User Bias

- Different users may have different rating scale.
- Possible solution:
  - Find a formula to obtain debiased ratings $D_U(R(U,V))$
  - Predict debiased rating $D_U(\widehat{R(U,V)})$ using only debiased ratings
  - Compute the biased rating using the inverse formula $D_U^{-1}\left(D_U(\widehat{R(U,V)})\right)$
- Classical formulas:
  - Mean corrected: $D_U(R(U,V)) = R(U,V) - \overline{R(U)}$ with $\overline{R(U)}$ the mean rating for user U. so that $D_U^{-1}\left(D_U(\widehat{R(U,V)})\right) = D(\widehat{R(U,V)}) + \overline{R(U)}$
  - Standardize: $D_U(R(U,V)) = (R(U,V) - \overline{R(U)})/\sigma(R(U))$ with $\sigma(R(U))$ the standard deviation of the ratings of user U so that $D_U^{-1}\left(D_U(\widehat{R(U,V)})\right) = \sigma(R(U))D(\widehat{R(U,V)}) + \overline{R(U)}$

Long-tail Phenomena

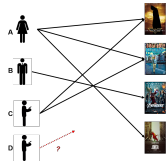


Long-tail Phenomena

- Different users/items may have very different number of ratings (and most users/items have few)
- Similarity may be biased by few items/users having a lot of ratings
- Possible solution:
  - Use a weighted similarity with a weight $-\log(N(U)/(\sum_{U'} N(U'))$ $(-\log(N(V)/(\sum_{V'} N(V'))))$ where $N(U)$ $(N(V))$ is the number of ratings of user U (item V)
- Information theory approach similar to tf-idf in NLP.

Source: C. Aggarwal

Cold Start Issue



Cold Start Issue

- Many users (items) have very few ratings.
- Some users (items) are new...

- **Not an issue for feature based or content based approaches!**

Possible Solutions

- Population approach: average based recommendation.
- Demographic approach: simple feature based recommendation.
- Scarce information approach: seeded recommendation.

Source: B. Kim

Top Items

Population Approach

- For a new user, one can use the population average to estimate $R(U, V)$
- Amount to use a constant similarity and a neighborhood equal to the whole population.
- No equivalent approach for a new item!

Demographic Approach

- If one has a *demographic* group information on the user, one may compute the average on the group.
- Amount to use a constant similarity and a neighborhood equal to the *demographic* group.
- Similar idea for a new item!

Source: B. Kim

Seeded Recommendations and Blending

Seeded Recommendations

- Compute the average on a group depending on the user behavior
- Most classical choice: compute an average on the users having given a good rating to the current viewed item
- Amount to use a constant similarity and a neighborhood equal to the group of users having given a good rating to the current viewed item.

Blending

- For user (item) with few ratings, it is often better to blend a collaborative solution with a cold start one.

Source: B. Kim

Pros and Cons

Pros

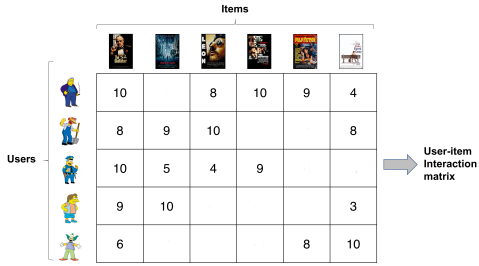
- Intuitive idea
- Easy to explain
- Can handle features and text
- Can be degraded to handle cold start

Cons

- Require an (expensive) neighborhood search!
- Require a lot of ratings to use them in similarities

Outline

Recommendation as Matrix Completion



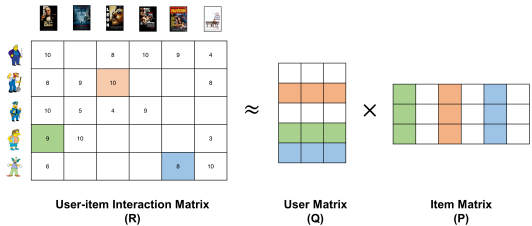
User-Item Interaction Matrix

- Matrix of ratings!
- Often most of the ratings are unknown
- Predicting the missing recommendation can be seen as completing the whole user-item interaction matrix.

- Approach based only on the ratings...

Source: B. Kim

Matrix Factorization Principle

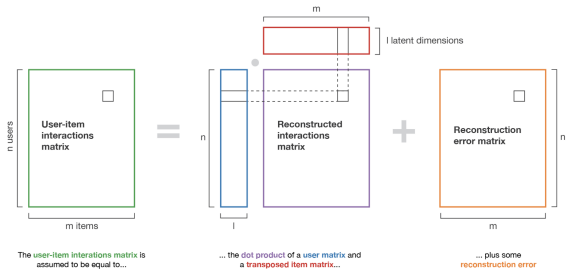


Matrix Factorization Principle

- To fill the voids, we need to add some regularity assumption.
- Simplest assumption: the $n \times p$ matrix R is (approximately) low rank, i.e $R \simeq UV^\top$ with U a $n \times k$ matrix and V a $p \times k$ matrix.

Source: B. Kim

Matrix Factorization Principle



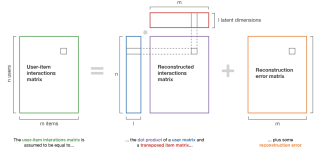
Strong Link with SVD

- Any $n \times p$ matrix $R.$ can be written $UDV^\top$ where U and V are orthogonal matrices and D is diagonal
- The best low rank approximation is obtain by restricting those matrix to the singular values with the largest eigenvalues in D.

- **Here R is not fully known so that we can't use the raw SVD!**

Source: S. Canu

Practical Factorization with SVD



SVD

- Formulation:

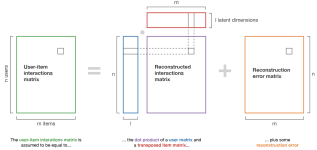
$$\underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \|R - UV^\top\|_2^2$$

$$\Leftrightarrow \underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} (R_{i,j} - U_{i,\cdot} {V_{j,\cdot}}^\top)^2$$

- Explicit solution through the SVD of the unknown R.
- May be used to obtain a baseline factorization by applying SVD to a completed R with simple replacement of the missing ratings by the mean(s).

Source: S. Canu

Practical Factorization with Weighted SVD



Weighted SVD

- Idea: Use a weight to mask the missing values in the fit
- Formulation:

$$\underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \|W \odot (R - UV^\top)\|_2^2$$

$$\Leftrightarrow \underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2 (R_{i,j} - U_{i,\cdot} V_{j,\cdot}{}^\top)^2$$

- No explicit solution!
- Non convex optimization problem!

Source: S. Canu

Practical Factorization with Iterative Masked SVD

Iterative Masked SVD

- When W is a mask, i.e. $W_{i,j} \in \{0, 1\}$, there exists a simple descent algorithm!
- Algorithm:
  - Start by an initial factorization $U_0 V_0^\top$.
  - Iterate T time:
    - Compute the completed matrix $R_t = W \odot R + (1 - W) \odot (U_t V_t^\top)$
    - Use the SVD to obtain a factorization of R_t by $U_{t+1} V_{t+1}^\top$
  - Use the last factorization $U_T V_T^\top$.
- Instance of a MM algorithm without any global optimality result.
- Previous use of the SVD on the completed ratings corresponds to one step of this algorithm.

- **Computing the SVD can be very expensive!**

Source: S. Canu

Practical Factorization with Alternate Least Square

Alternate Least Square

- Weighted SVD formulation:

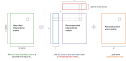
$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \|W\odot(R-UV^\top)\|_2^2 \Leftrightarrow \underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2(R_{i,j}-U_{i,\cdot}.V_{j,\cdot}{}^\top)^2$$

- Optimization on U (V) corresponds to n (p) classical least-squares optimizations.
- Lead to an alternate least-squares descent algorithm without any global optimality result:
  - Start by an initial factorization $U_0 V_0{}^\top$
  - Iterate T times
    - Solve $U_{k+1} = \operatorname{argmin}_{U\in\mathcal{M}_{n,k}} \|W\odot(R-UV_k{}^\top)\|_2^2$
    - Solve $V_{k+1} = \operatorname{argmin}_{V\in\mathcal{M}_{p,k}} \|W\odot(R-U_{k+1}V^\top)\|_2^2$
  - Use $U_T V_T{}^\top$ as final factorization.

- **Computing those solutions may remain expensive!**

Source: S. Canu

Practical Factorization with SGD



Stochastic Gradient Descent

- Weighted SVD formulation:

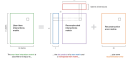
$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \|W\odot(R-UV^\top)\|_2^2 \Leftrightarrow \underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2(R_{i,j}-U_{i,\cdot}V_{j,\cdot}{}^\top)^2$$

- Look at this problem as an optimization on $U_{i,\cdot}$ and $V_{j,\cdot}$ and use a stochastic gradient scheme without any global optimality result:
  - Start by some initial $U_{i,\cdot}$ and $V_{j,\cdot}$
  - Iterate
    - Pick uniformly a pair (i,j)
    - Update $U_{i,\cdot}$ by $U_{i,\cdot}+W_{i,j}\gamma(R_{i,j}-U_{i,\cdot}V_{j,\cdot}{}^\top)V_{j,\cdot}$
    - Update $V_{j,\cdot}$ by $V_{j,\cdot}+W_{i,j}\gamma(R_{i,j}-U_{i,\cdot}V_{j,\cdot}{}^\top)U_{i,\cdot}$
  - Use $UV^\top$ as final factorization.

- **As in any SGD scheme, the choice of the stepsize γ is very important.**

Source: S. Canu

Extension of Practical Factorization



Unbiased Rating

- Better results if one replace R with an unbiased version:
  - by subtracting the global mean (and adding it afterward)
  - by subtracting the user means (and adding them afterward)

Regularization

- Regularized Weighted SVD formulation:

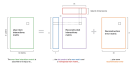
$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \|W\odot(R-UV^\top)\|_2^2+\lambda\|U\|_2^2+\lambda\|V\|^2$$

$$\Leftrightarrow \underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2(R_{i,j}-U_{i,\cdot}{V_{j,\cdot}}^\top)^2+\lambda\left(\sum_{i=1}^{n}\|U_{i,\cdot}\|_2^2+\sum_{j=1}^{p}\|V_{j,\cdot}\|_2^2\right)$$

- Alternate Least-Squares and SGD can be extended to this setting.

Source: S. Canu

Practical Factorization and Funk's Algorithm



Funk's Algorithm

- Funk's formulation:

$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k},\mu\in\mathbb{R},u\in\mathbb{R}^n,v\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2 (R_{i,j} - (\mu + u_i + v_j + U_{i,.} V_{j,.}^\top))^2 + \lambda \left(\mu^2 + \sum_{i=1}^{n} (u_i^2 + \| U_{i,\cdot} \|_2^2) + \sum_{j=1}^{p} (v_j^2 + \| V_{j,\cdot} \|_2^2) \right)$$

- Explicit formula including the user and item bias!
- SGD can be used in this setting!

- **Lead to state of the art results!**

Source: S. Canu

Pros and Cons

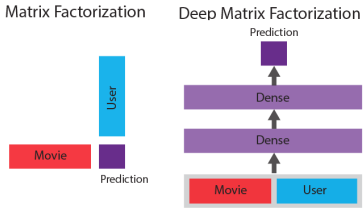
Pros

- Quite efficient even if the rating matrix is sparse.
- Lead to an explicit formula for any pair of user/item.
- Efficient numerical algorithm.

Cons

- No straightforward explanation of the prediction.
- Do not use features or text.
- No way to handle cold start.

Recommendation as Prediction



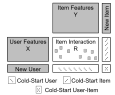
Factorization as a Prediction Algorithm

- Optimization of a formula
$$R(U_i, V_j) = \mu + u_i + v_j + U_{i,\cdot} V_{j,\cdot}^{\top}$$
with a least-squares criterion.
- Other formulas are probably possible...
- Key: representation learning ? Can we use Deep Learning?

- **Not easy to do better than matrix factorization with a classical DNN!**
- **Explicit scalar product seems required!**

Source: J. Schreiber

Model Based Recommendation

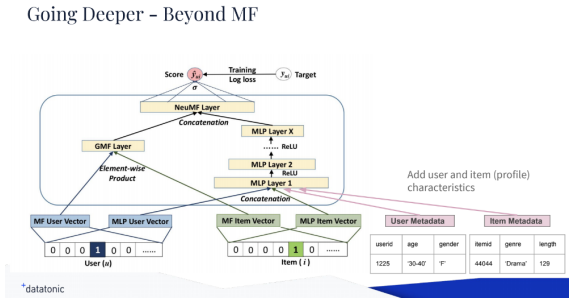


Model Based Recommandation

- Optimization of a formula:
$$R(U_i, V_j) = f(U_i, V_j)$$
where U_i and V_i can be a combination of an id (one hot encoding) and features.
- Simple additive models:
$$R(U_i, V_i) = f_U(U_i) + f_V(V_j)$$
- Models with explicit interactions:
$$R(U_i, V_i) = f_U(U_i) + f_V(V_j) + F_{UV}(U_i, V_j)$$
- Possible extension of factorization based on
$$F_{UV}(U_i, V_i) = M_U U_i (M_v V_j)^\top$$
- Link with transformers...

Source: DeepAI

Deep Recommendation



Deep Recommendation

- Combine an explicit dot product structure with a classical DNN.
- Allow to learn a representation and to add features / text content directly.
- Large flexibility in the architecture.

Source: Altoros

Pros and Cons

Pros

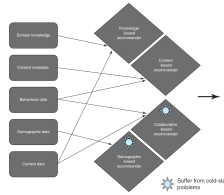
- Combine the strength of the factorization based and the feature based methods
- Best performances...

Cons

- Not so easy to construct a good formula/architecture...
- Not so easy to train...
- Not easy to beat raw matrix factorization (when using only user/item interactions)!

Outline

Hybrid Recommender



Hybrid Recommender

- Combine the scores of several recommendation algorithms.
- Can be casted as an ensemble method where the number of interactions is used.

Pros

- Lots of flexibility

Cons

- Lots of flexibility!

Source: K. Falk

Performance Measure

CASE 1: Evenly distributed errors

ID	Error	|Error|	Error^2
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4

MAE	RMSE
2.000	2.000

CASE 2: Small variance in errors

ID	Error	|Error|	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

CASE 3: Large error outlier

ID	Error	|Error|	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

MAE	RMSE
2.000	6.325

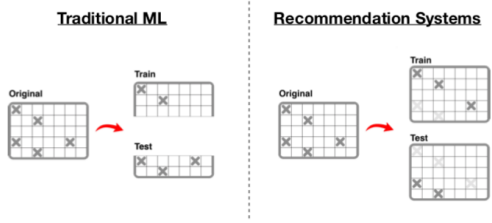
- **Need of a metric to measure the performance!**

Metric on the ratings

- RMSE:
  - Most classical choice
  - Implicitly used in collaborative filtering and explicitly in matrix factorization.
  - Easy to use.
- MAE: more robust to outliers...

Source: JJ

Validation



- **Need of validation technique!**

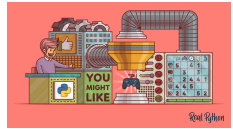
Validation Scheme

- Much more complicated that the usual supervised setting.
- Lack of independence of the observations.
- Most classical choice: random partition of the ratings!

- No strong theoretical support!

Source: J. Cates

Metric vs Goals



- Are those metrics really the right thing to optimize?

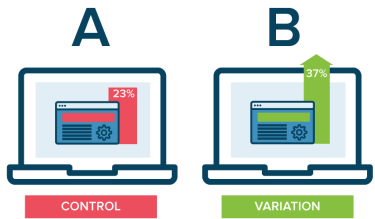
Better Goals

- Diversity : do not always suggest the same items.
- Coverage: suggest most of the items to at least some users.
- Serendipity: suggest **surprising** items.
- Business Goal: Sell more! Earn more money!

- Explain why there is a lot of post-processing to go from the ratings to the suggested item list!
- For instance: use of lift instead of ratings, use of localization, use of randomization...

Source: Real Python

A/B Testing



A/B Testing

- No direct way to estimate the performance according to non trivial metric.
- Solution: perform experiment to test whether a method is good or not!
- A/B Testing: classical hypothesis testing on the means (or the proportions).
- Bandit approach: real-time optimization of the allocation (not much used in practice).

Source: Optimizely

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

References

K. Falk.
Practical Recommender Systems.
Manning, 2019

F. Ricci, L. Rokach, and B. Shapira.
Recommender Systems Handbook (3rd ed.)
Springer, 2022

Ch. Aggarwal.
Recommender Systems, The Textbook.
Springer, 2016

Outline

Outline

Text and Bag of Words

- How to transform a **text** into a vector of **numerical features**?

Bag of Words strategy

- Make a **list** of words.
- Compute a **weight** for each word.

List building

- Make the list of all used words with their number of occurrence.
- Gather the words in the list having the same stem (stemming).
- Compute the histogram $h_w(d)$.

Weight computation

- Apply a renormalization:
  - tf transform (word profile):
    $$\mathrm{tf}_w(d) = \frac{h_w(d)}{\sum_w h_w(d)}$$
    so that $\mathrm{tf}_w(d)$ is the frequency within the document d.
  - tf-idf transform (word profile weighted by rarity):
    $$\mathrm{tf-idf}_w(d) = \mathrm{idf}_w \times \mathrm{tf}_w(d)$$
    with idf a corpus dependent weight
    $$\mathrm{idf}_w = \log \frac{n}{\sum_{i=1}^{n} \mathbf{1}_{h_w(d_i)\neq 0}}$$

- Use the vector $\mathrm{tf}(d)$ (or $\mathrm{tf-idf}(d)$) to describe a document.
- Most classical text preprocessing!
- Latent Semantic Analysis: PCA of this representation.
- Hashing can be used to reduce the number of words.

Stemming and Lemmatization

Stemming

adjustable → adjust

formality → formaliti

formaliti → formal

airliner → airlin ⚠

Lemmatization

was → (to) be

better → good

meeting → meeting

Text Preprocessing

- Very important step in text processing.
- Art of obtaining good tokens.
- Spelling correction, Stemming, Lemmatization...

Source: Quora

Okapi BM25 for Text Retrieval

Okapi BM25

- Representation (smoothed tf-idf):

$$\mathrm{bm25}_w(d) = \mathrm{idf}_w \times \frac{(k_1+1)\mathrm{tf}_w(d)}{k_1 + \mathrm{tf}_w(d)}$$

- Match quality for a set of words Q measured by a simple scalar product:

$$\mathrm{BM25}(d, Q) = \sum_{w \in Q} \mathrm{bm25}_w(d)$$

- Extensively used in text retrieval.
- Can be traced back to 1976!

Unsupervised Text Clustering

Probabilistic latent semantic analysis (PLSA)

- Model:

$$\mathbb{P}(\mathrm{tf}) = \sum_{k=1}^{K} \mathbb{P}(k)\,\mathbb{P}(\mathrm{tf}|k)$$

with k the (hidden) topic, $\mathbb{P}(k)$ a topic probability and $\mathbb{P}(\mathrm{tf}|k)$ a multinomial law for a given topic.

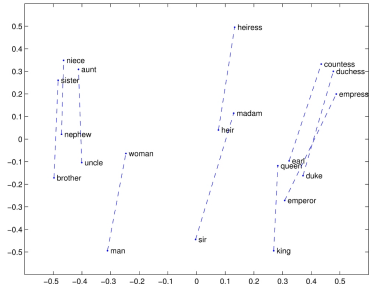
- Clustering according to a mixture model

$$\mathbb{P}(k|\mathrm{tf}) = \frac{\widehat{\mathbb{P}(k)}\widehat{\mathbb{P}(\mathrm{tf}|k)}}{\sum_{k'} \widehat{\mathbb{P}(k')}\widehat{\mathbb{P}(\mathrm{tf}|k')}}$$

- Same idea than GMM!
- Bayesian variant called LDA.

Outline

Word Vectors



Word Embedding

- Map from the set of words to $\mathbb{R}^d$.
- Each word is associated to a vector.
- Hope that the relationship between two vectors is related to the relationship between the corresponding words!

Source: nlp.stanford.edu

Word And Context

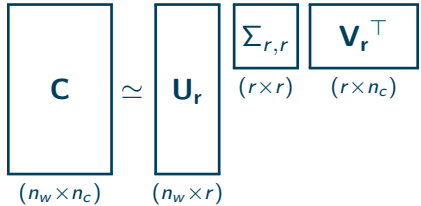
Look ! A single **word** and its context

Word And Context

- **Idea:** characterize a word w through its relation with words c appearing in its context...
- **Probabilistic description**:
  - Joint distribution: $f(w,c) = \mathbb{P}(w,c)$
  - Conditional distribution(s): $f(w,c) = \mathbb{P}(w|c)$ or $f(w,c) = \mathbb{P}(c|w)$.
  - Pointwise mutual information: $f(w,c) = \mathbb{P}(w,c)/(\mathbb{P}(w)\mathbb{P}(c))$
- Word w characterized by the vector $C_w = (f(w,c))_c$ or $C_w = (\log f(w,c))_c$.

- In practice, C is replaced by an estimate on large corpus.
- Very high dimensional model!

A (Naïve) SVD Approach



Truncated SVD Approach

- Approximate the embedding matrix C using the truncated SVD decomposition (best low rank approximation).
- Use as a code

$$C'_w = U_{r,w}\Sigma^{\alpha}_{r,r}$$

with $\alpha \in [0, 1]$.

- Variation possible on C.
- State of the art results but computationally intensive...

A Least-Squares Approach

- All the previous models correspond to
$$-log\mathbb{P}(w,c) \sim C'^t_w C''_c + \alpha_w + \beta_c$$

GloVe (Global Vectors)

- Enforce such a fit through a (weighted) least-squares formulation:
$$\sum_{w,c} h(\mathbb{P}(w,c)) \left\| -log\mathbb{P}(w,c) - (C'^t_w C''_c + \alpha_w + \beta_c) \right\|^2$$
with h a increasing weight.
- Minimization by alternating least square or stochastic gradient descent...

- Much more efficient than SVD.
- Similar idea in recommendation system.

A Learning Approach

Supervised Learning Formulation

- True pairs (w, c) are positive examples.
- Artificially generate negative examples (w', c') (for instance by drawing c' and w' independently in the same corpus.)
- Model the probability of being a true pair (w, c) as a (simple) function of the codes C'_w and C''_c.

- Word2vec: logistic modeling

$$\mathbb{P}(1|w, c) = \frac{e^{C'^t_w C''_c}}{1 + e^{C'^t_w C''_c}}$$

- State of the art and efficient computation.
- Similar to a factorization of $-\log(\mathbb{P}(w, c)/(\mathbb{P}(w)\mathbb{P}(c)))$ but without requiring the estimation of the probabilities!

Outline

Text as Sequences

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition.

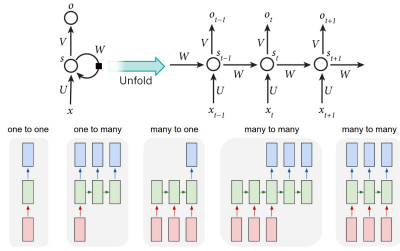
Sequences

- Word = sequence of letters.
- Text = sequence of letters/words.

- Capitalize on this structure.

Source: Wikipedia

Recurrent Neural Networks



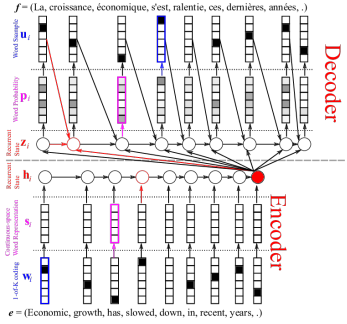
Recurrent Neural Network Unit

- Input seen as a sequence.
- **Simple** computational units with shared weights.
- Information transfer through a context!

- Several architectures!

Sources: Y. LeCun / Unknown

Automatic Translation

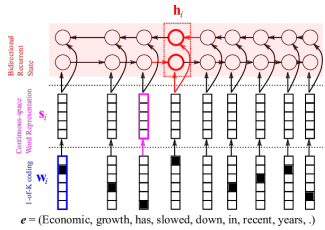


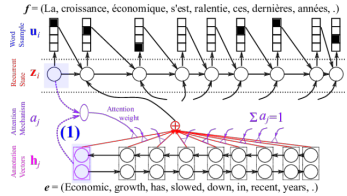
Encoder/Decoder structure

- Word vectors, RNN, stacked structure.

Source: Nvidia

Automatic Translation



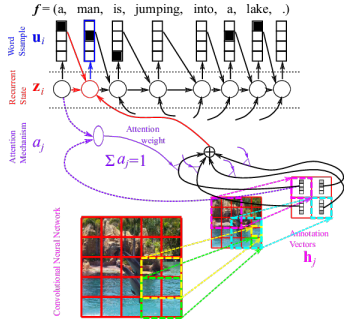


Encoder/Decoder structure

- Much more complex structure: asymmetric, attention order...

Source: Nvidia

Automatic Captioning

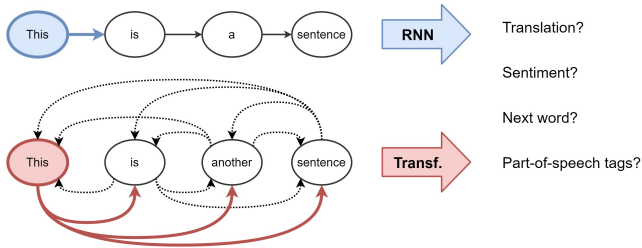


Encoder/Decoder structure

- Much more complex structure: asymmetric, attention order...

Source: Nvidia

Text as Graph

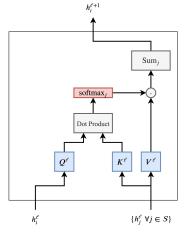


Text as Graph

- More than just sequential dependency.
- Each word is related to (all the) other words.
- Graph structure with words and directed relations between words.

Source: Chaitanya Joshi

Attention



Attention between words

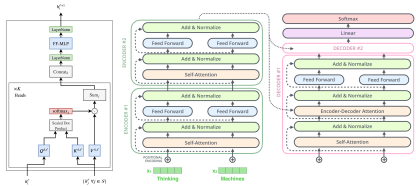
- Words encoded by h_i at layer l.
- Compute individual value for each word: $v_i = V^l h_i$
- Compute combined value for each word: $h_i' = \sum_j w_{i,j} v_j$
- **(Self) Attention:** weight $w_{i,j}$ defined by

$$w_{i,j} = \text{SoftMax}\left(\left\langle Q^l h_i, K^l h_j \right\rangle\right)$$

- $Q^l h_i$ is called a query and $K^l h_j$ a key.

Source: Chaitanya Joshi

Transformer



Transformer

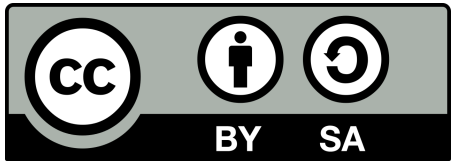
- Block combining several attention heads and a classical MLP.

Encoder/Decoder Architecture

- Combine several transformers and more MLP in a task-adapted architecture.
- End-to-end training is not easy (initialization, optimization...).
- Initial embedding at token level rather than word level to cope with new words!

Sources: Chaitanya Joshi / Jay Alamar

Licence and Contributors



Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.