Recommendation Systems

Andrea Araldo

March 5, 2020

Outline

- Recommender systems: applications and taxonomy
- Netflix competition
- Matrix Factorization: concepts and implementation
- Deep Recommender Systems: concepts and implementation
- Other improvements (stars vs. thumbs, blending, etc.)

Section 1

Introduction to Recommender Systems

Recommendation systems

Systems that aim to propose items that a particular user might be interested in.

- E-commerce: Amazon, E-bay
- Multimedia content: Netflix, Spotify, Youtube
- Social Networks: Facebook posts and groups
- Online advertisement: Google
- FMS

Final goal: **increase revenues**:

- 75% of Netflix's watched movies and 35% of Amazon's purchases come from recommenders [MMN13].

Types of recommendation systems

Source of information

- *Content-based*:
  Based on the description of the items and the users
  - The features of users (age, education, nationality) and of movies (genre, year, nationality) must be known.
- *Collaborative Filtering*
  Learns what to recommend based on the interaction between users and items
- *Hybrid*
  Important particularly for “cold start” (a user or movie are new - no interaction has been observed so far).

Types of feedback

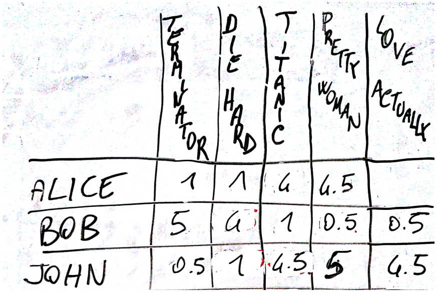
- *Explicit*: stars, thumb up / thumb down, etc.
- *Implicit*: Clicks, how much time user played a song.

Collaborative filtering

- Method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).[1]
- **Assumption**: I can guess what you will like if I observe:
  - What you have already liked
  - The preferences of the others that have liked the same things as you

[1] wikipedia

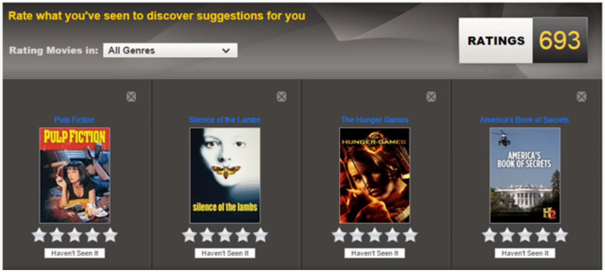
Intuition



	TERMINATOR	DIE HARD	TITANIC	PRETTY WOMAN	LOVE ACTUALLY
ALICE	1	1	4	4.5	
BOB	5	4	1	0.5	0.5
JOHN	0.5	1	4.5	5	4.5

- *User-Item matrix*
- Rating of Alice on Love Actually?

Netflix



- The user rates
- The system predicts future rates
- User may correct (the system can periodically re-train)

Netflix Competition Prize

- 2006-09
- Root Mean Squared Error 0.92525
- 1,000,000 $ to whoever was able to decrease it to 0.8572
- 20K teams

Netflix Competition method

- Training dataset: 1M
- During the competitions participants evaluated their algorithms:
  - Netflix gave 1.4M quiz set (without ratings)
  - Participants gave their predicted ratings
  - Netflix communicated the RMSQ on the quiz set (no actual predictions given)
- The “official” score (undiscovered until the end) was the RMSQ on a 1.4M test set

Netflix Competition dataset

- Each datapoint: [anonimized user id, title, rating, date of rating]
- 480K customers
- 17K movies

Contribution of Netflix competition?

- Many techniques were invented or improved during the competition
- Ex. Matrix Factorization, de facto standard model [Pen18].

Section 2

Matrix Factorization: concepts

Matrix Factorization: Goals

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1		0.9		
Bob		0.1		1	
John		1	0.1		
Ruby		1			0.2
Nick	0.1			1	
Paul		0.2	1		

- *Rating matrix R.*
- Observed/Missing values.
- Only 1% of user-movie ratings [Ste15].
- *Prediction* of missing values
  - Rating of Alice on Pretty Woman?
  - Rating of Ruby on Terminator?
- How is prediction related to recommendation?

What a machine sees

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	0.1		0.9		
User 2		0.1		1	
User 3		1	0.1		
User 4		1			0.2
User 5	0.1			1	
User 6		0.2	1		

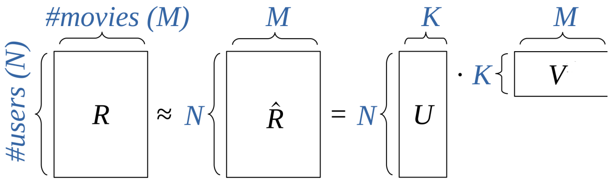
No semantics

- (Alice(1), Bob(2)) are no more similar than (Alice(1), Ruby(4))
- (Pretty(4), Love(5)) are no more similar than (Love(5), Terminator(1))

Matrix factorization

Find matrices U and V to approximate R:

$$R \approx \hat{R} = U \cdot V$$



Separation

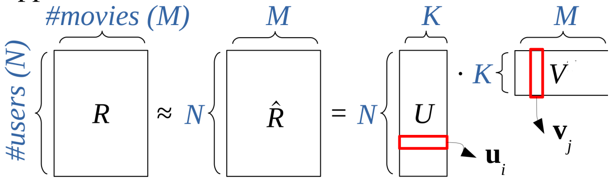
- In R user and movies information was mixed.
- Now, U (user info) and V (movie info).

Dimensionality reduction

- In R each user was represented by M numbers.
- Now, only by K numbers.
- (also for the movies)

Training

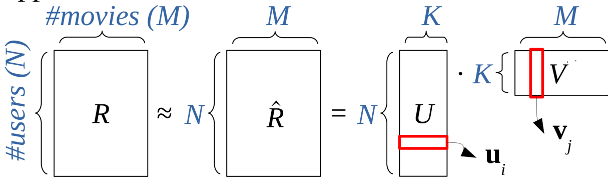
For a value K (fixed by design), determine U and V that best approximates R.



- $\mathbf{u}_i$: related to user i; $\mathbf{v}_j$: related to movie j. $\hat{r}_{ij} =$???

Training

For a value K (fixed by design), determine U and V that best approximates R.

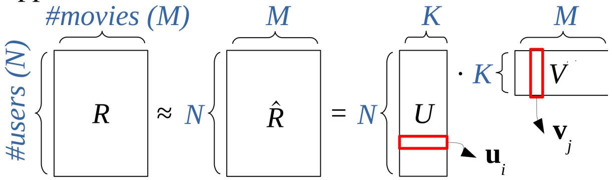


- $\mathbf{u}_i$: related to user i; $\mathbf{v}_j$: related to movie j. $\hat{r}_{ij} =$???
- **Optimization problem**: Find $\mathbf{u}_i$ and $\mathbf{v}_j$, for any user i and movie j, such that *Loss Function* (error):

$$\min \sum_{(i,j)\text{ observed}} ||r_{ij} - \hat{r}_{ij}||^2$$

Training

For a value K (fixed by design), determine U and V that best approximates R.



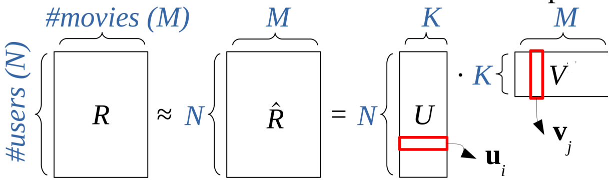
- $\mathbf{u}_i$: related to user i; $\mathbf{v}_j$: related to movie j. $\hat{r}_{ij} = ???$
- **Optimization problem**: Find $\mathbf{u}_i$ and $\mathbf{v}_j$, for any user i and movie j, such that *Loss Function* (error):

$$\min \sum_{(i,j) \text{ observed}} ||r_{ij} - \hat{r}_{ij}||^2 = \min \sum_{(i,j) \text{ observed}} ||r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j||^2$$

- Contraint: each element in $[0,1]$ (§3.6.6 of [Agg16])
- Equivalent to finding matrices U and V.
- Several methods to solve it (out of scope).

Training: Remarks

Determine U and V such that $\hat{R}$ is "as close as possible" to R.



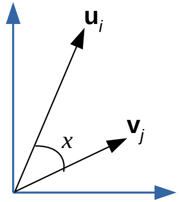
- **Optimization problem**:
  Find U and V such that

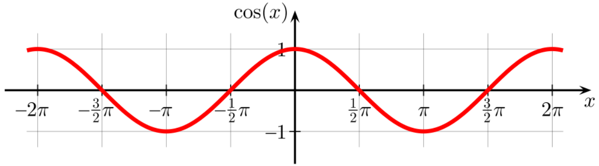
$$\min \sum_{(i,j)\ \text{observed}} ||r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j||^2$$

- The computation of U and V only depends on the *observed values*.
- But we get all the ratings $\hat{r}_{ij}$, also for non observed user-movie ratings.
- **We are able to predict non-observed ratings!**

Similarity

$$\mathbf{u}_i \cdot \mathbf{v}_j = ||\mathbf{u}_i|| \cdot ||\mathbf{v}_j|| \cdot \cos x$$

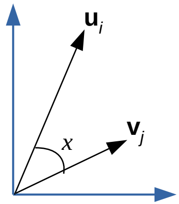


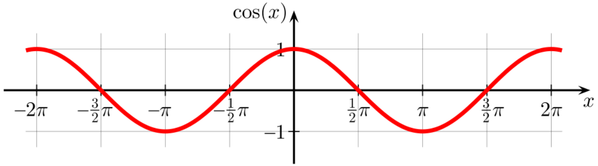


Which angle x maximizes $\mathbf{u}_i \cdot \mathbf{v}_j$?

Similarity

$$\mathbf{u}_i \cdot \mathbf{v}_j = ||\mathbf{u}_i|| \cdot ||\mathbf{v}_j|| \cdot \cos x$$



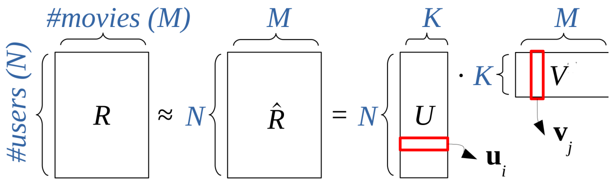


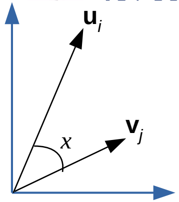
Which angle x maximizes $\mathbf{u}_i \cdot \mathbf{v}_j$?

If $\mathbf{u}_i \cdot \mathbf{v}_j$ high $\Rightarrow$ $\cos x$ high $\Rightarrow$ x small $\Rightarrow$ the vectors are similar.

$\mathbf{u}_i \cdot \mathbf{v}_j$ is the **affinity** of user i toward movie j.

Insights on training





- **Optimization problem**:
  Find U and V such that

$$\min \sum_{(i,j) \text{ observed}} ||r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j||^2$$

- If $r_{ij} \approx 1$, the training algorithm should construct $\mathbf{u}_i$ and $\mathbf{v}_j$ such that their angle is . . . ?
- What if $r_{ij} \approx 0$?

Example of predicted rating matrix

Original rating matrix R

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1		0.9		
Bob		0.1		1	
John		1	0.1		
Ruby		1			0.2
Nick	0.1			1	
Paul		0.2	1		

Predicted rating matrix $\hat{R} = U \cdot V$

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1	0.04	0.86	0.94	0.86
Bob	0.04	0.1	0.84	1	0.78
John	0.88	1	0.1	0.2	0.1
Ruby	0.78	1	0.4	0.6	0.2
Nick	0.1	0.12	0.76	9.6	9.6
Paul	0.4	0.24	1	0.8	0.9

- The predicted matrix has also values for unobserved ratings.
- Would you agree with the values?

Example of predicted rating matrix

Original rating matrix R

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1		0.9		
Bob		0.1		1	
John		1	0.1		
Ruby		1			0.2
Nick	0.1			1	
Paul		0.2	1		

Predicted rating matrix $\hat{R} = U \cdot V$

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1	0.04	0.86	0.94	0.86
Bob	0.04	0.1	0.84	1	0.78
John	0.88	1	0.1	0.2	0.1
Ruby	0.78	1	0.4	0.6	0.2
Nick	0.1	0.12	0.76	9.6	9.6
Paul	0.4	0.24	1	0.8	0.9

- The predicted matrix has also values for unobserved ratings.
- Would you agree with the values?
- For observed (i,j), $\hat{r}_{ij} \approx r_{ij}$, but they are not always the same. Can we do better?

Example of predicted rating matrix

Original rating matrix R

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1		0.9		
Bob		0.1		1	
John		1	0.1		
Ruby		1			0.2
Nick	0.1			1	
Paul		0.2	1		

Predicted rating matrix $\hat{R} = U \cdot V$

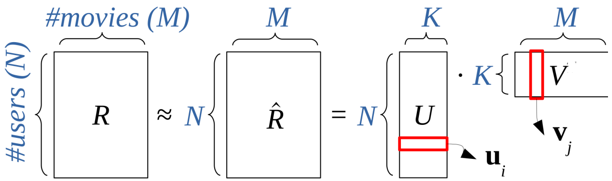
	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1	0.04	0.86	0.94	0.86
Bob	0.04	0.1	0.84	1	0.78
John	0.88	1	0.1	0.2	0.1
Ruby	0.78	1	0.4	0.6	0.2
Nick	0.1	0.12	0.76	9.6	9.6
Paul	0.4	0.24	1	0.8	0.9

- The predicted matrix has also values for unobserved ratings.
- Would you agree with the values?
- For observed (i,j), $\hat{r}_{ij} \approx r_{ij}$, but they are not always the same. Can we do better?
- No, to minimize the overall error

$$\sum_{(i,j)\text{ observed}} ||r_{ij} - \hat{r}_{ij}||^2 = \sum_{(i,j)\text{ observed}} ||r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j||^2$$

we need to sacrifice the precision on the single observed (i,j).

Latent factors



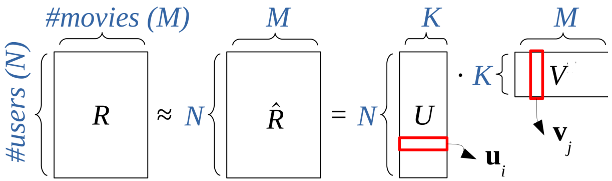
Fixing $K = 2$, V=

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Factor 1	0.98	1	0.56	0.2	0.06
Factor 2	0.06	0	1	0.98	0.99

$\mathbf{v}_j$

- Can you give a meaning to Factor 1 and 2 by looking at V?

Latent factors



Fixing $K = 2$, $V=$

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Factor 1	0.98	1	0.56	0.2	0.06
Factor 2	0.06	0	1	0.98	0.99

$\mathbf{v}_j$

- Can you give a meaning to Factor 1 and 2 by looking at V?
- **Latent factors**: they are hidden in the data and show up "magically" after matrix factorization.
- They may have a meaning, but not necessarily.
- We can now compute "distance" between movies.
- Our human knowledge about movies is now better embedded in the matrix.

Training: other insights

$R =$

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1		0.9		
Bob		0.1		1	
John		1	0.1		
Ruby		1			0.2
Nick	0.1			1	
Paul		0.2	1		

$V =$

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Factor 1	0.98	1	0.56	0.2	0.06
Factor 2	0.06	0	1	0.98	0.99

$\rightarrow \mathbf{v}_j$

$$\min \sum_{(i,j) \text{ observed}} ||r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j||^2$$

- Build a possible $\mathbf{u}_{\text{Alice}} = (a, b)$.
- It must be such that $\mathbf{u}_{\text{Alice}} \cdot \mathbf{v}_{\text{terminator}} \approx 0.1$ while $\mathbf{u}_{\text{Alice}} \cdot \mathbf{v}_{\text{titanic}} \approx 0.9$
- $\dots$

Remarks:

- The value of $\mathbf{u}_{\text{Alice}}$ is determined by the affinity of Alice on the movies she has ranked.
- We never construct $\mathbf{u}_i$ and $\mathbf{v}_j$ by hand. We use algorithms instead.

Latent Factors and User-Movie affinity

$U =$

	Factor 1	Factor 2
Alice	0.1	1
Bob	0.04	0.95
John	1	0.04
Ruby	1	1
Nick	0.2	0.95
Paul	0.06	1

$\rightarrow \mathbf{u}_i$

$V =$

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Factor 1	0.98	1	0.56	0.2	0.06
Factor 2	0.06	0	1	0.98	0.99

$\rightarrow \mathbf{v}_j$

- Factor 1 is at the same time
  - How much action is in a movie
  - How much action a user likes
- Factors have the **same semantic** on users and movies.

Recap on latent factos

- They can have some semantics.
- The semantic is common to both users and movies.
- They allow to evaluate similarity between users and between movies:
  - If $\mathbf{u}_i$ is similar to $\mathbf{u}_{i'}$, then users i and i' have the same tastes, even if they have no common movie ranked.
  - If $\mathbf{v}_j$ is similar to $\mathbf{v}_{j'}$, then movie j and j' are of the same kind, even if no common user have ranked them.
  - Note that this information is not present in the original rating matrix R.
- They allow to evaluate the affinity of user i and movie j by computing $\mathbf{u}_i \cdot \mathbf{v}_j$.

Section 3

Matrix Factorization in practice

Embedding

- Motivation:
  - We had users Alice, Bob, etc. and movies Titanic, Die Hard, etc.
  - To predict the rating, we transformed them in vectors $\mathbf{u}_{\text{Alice}}, \mathbf{u}_{\text{Bob}}, \dots$ and $\mathbf{v}_{\text{Titanic}}, \mathbf{v}_{\text{Die Hard}}, \dots$.
- **Definition**:
  Embedding is the process of mapping objects into numerical vectors, so that we can apply mathematical operations.
- Can you construct a user-to-vector embedding arbitrarily?

Embedding

- Motivation:
  - We had users Alice, Bob, etc. and movies Titanic, Die Hard, etc.
  - To predict the rating, we transformed them in vectors $\mathbf{u}_{\text{Alice}}, \mathbf{u}_{\text{Bob}}, \ldots$ and $\mathbf{v}_{\text{Titanic}}, \mathbf{v}_{\text{Die Hard}}, \ldots$.
- **Definition**:
  Embedding is the process of mapping objects into numerical vectors, so that we can apply mathematical operations.
- Can you construct a user-to-vector embedding arbitrarily?
- Matrix Factorization: does it construct an arbitrary embedding or a good? In what sense?

Embedding

- Motivation:
  - We had users Alice, Bob, etc. and movies Titanic, Die Hard, etc.
  - To predict the rating, we transformed them in vectors $\mathbf{u}_{\text{Alice}}, \mathbf{u}_{\text{Bob}}, \ldots$ and $\mathbf{v}_{\text{Titanic}}, \mathbf{v}_{\text{Die Hard}}, \ldots$.
- **Definition**:
  Embedding is the process of mapping objects into numerical vectors, so that we can apply mathematical operations.
- Can you construct a user-to-vector embedding arbitrarily?
- Matrix Factorization: does it construct an arbitrary embedding or a good? In what sense?
- Matrix Factorization finds a good embedding, i.e., the one that

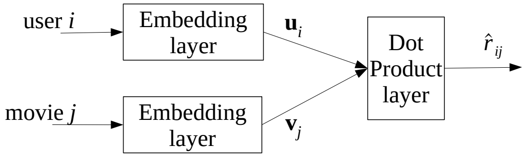
$$\min \sum_{(i,j)\text{ observed}} ||r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j||^2$$

- Other uses of embedding: Natural Language Processing (NLP).

Keras

- Python library used for Neural Networks and Deep Learning
- ... but not only. We will use it for Neural Networks
- Some facts
  - Based on Python
  - Open source
  - Primary author and maintainer is Franois Chollet, a French Google engineer.

Matrix Factorization in Keras



- Keras performs iterations
- First, the Embedding Layers initialize U and V arbitrarily
- Then, at each iteration
  - Measure the error between the prediction and the real ratings $\sum_{(i,j)\text{ observed}} ||r_{ij} - \hat{r}_{ij}||^2$.
  - The Embedding Layers adjust U and V in order to decrease the error, leveraging Stochastic Gradient Descent.

Supervised? Unsupervised? Semi-supervised?

It can be considered both:

- Unsupervised:
  - Dimensionality Reduction: We represent the rating matrix R with two smaller matrices U and V
  - From $O(NM)$ to $O((N+M)K)$
- Semisupervised:
  - Missing labels are useful to learning
  - **Ex.**: Suppose DieHard has been rated high by few users $\mathscr{U}_1$.
  - Other users $\mathscr{U}_2$ like similar movies to $\mathscr{U}_1$.
  - Bob likes similar movies to $\mathscr{U}_2$
  - Will Bob like DieHard?

Supervised? Unsupervised? Semi-supervised?

It can be considered both:

- Unsupervised:
  - Dimensionality Reduction: We represent the rating matrix R with two smaller matrices U and V
  - From $O(NM)$ to $O((N+M)K)$
- Semisupervised:
  - Missing labels are useful to learning
  - **Ex.**: Suppose DieHard has been rated high by few users $\mathscr{U}_1$.
  - Other users $\mathscr{U}_2$ like similar movies to $\mathscr{U}_1$.
  - Bob likes similar movies to $\mathscr{U}_2$
  - Will Bob like DieHard?
  - Yes. To infer (Bob,DieHard) rating we are using $\mathscr{U}_2$, although we do not have the rate label of $\mathscr{U}_2$ on DieHard.

Matrix Factorization

Let us code …

Section 4

Deep Recommender Systems

Motivation

So far, we have approximated the rating as:

$$\hat{r}_{ij} = \mathbf{u}_i \cdot \mathbf{v}_j$$

To be more flexible, we may want to approximate it as

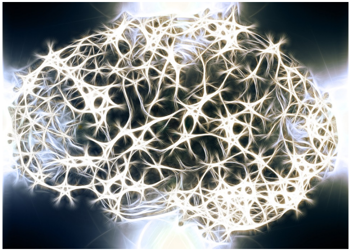
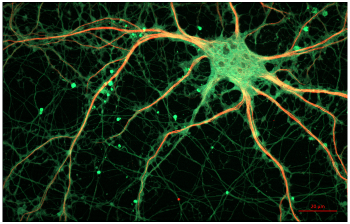
$$\hat{r}_{ij} = f(\mathbf{u}_i, \mathbf{v}_j)$$

and find the “best” f, i.e., the one that

$$\min \sum_{(i,j)\text{observed}} ||r_{ij} - f(\mathbf{u}_i, \mathbf{v}_j)||^2$$

The “best” f can be obtained with a Neural Network!

Neural Network - Human brain

- By ZEISS Microscopy from Germany (Cultured Rat Hippocampal Neuron) [CC BY 2.0 (http://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons

- https://pixabay.com/en/neurons-brain-cells-brain-structure-1739997/

- *Walter Pitts*: logician

  CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], via Wikimedia Commons

- *Warren McCulloc*: neurophysiologist

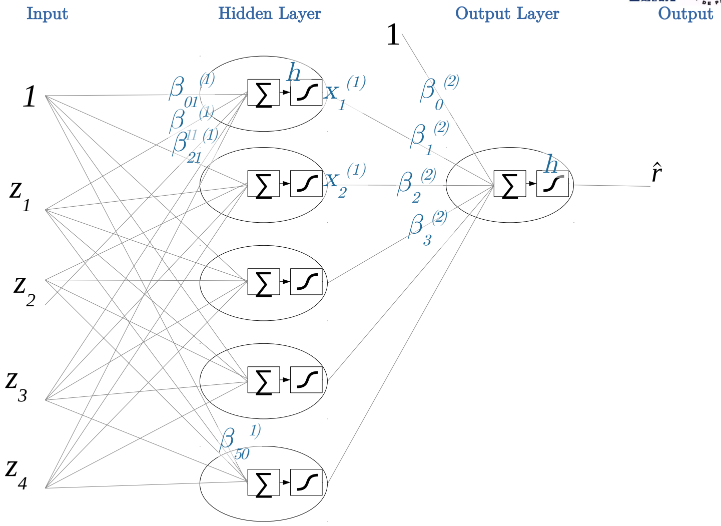
The life of a genius

Walter Pitts:

- At 12 read Principia Mathematica from Bertrand Russel.
- He wrote to Russel about some problems of his book.
- Russel invited him to Cambridge University and Pitts refused.

source: Wikipedia

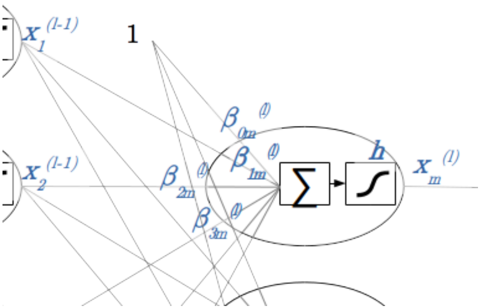
Neural Network - Model



In our case, the $z_1, z_2, \dots$ are the elements of the vectors $\mathbf{u}_i$ and $\mathbf{v}_j$.
$\hat{r} = f_\beta(\mathbf{z})$. f is *parametrized* by the weights.

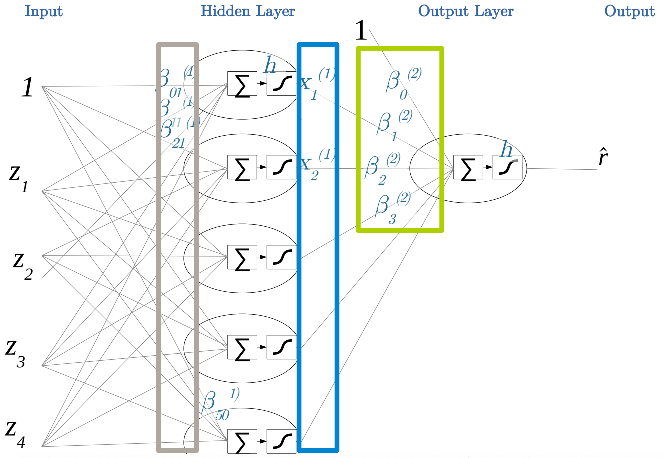
Neural Network - Single neuron

Let us look at the m-th neuron in the l-layer.



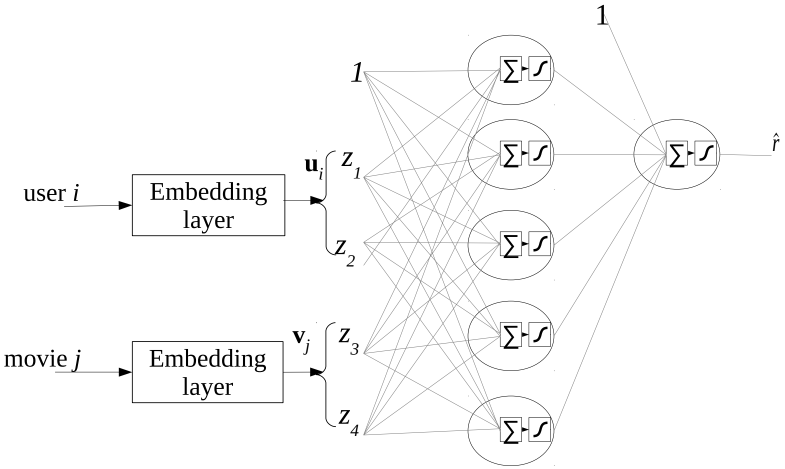
- Output from the previous level: $\mathbf{x}^{(l-1)}$
- Weights: $\boldsymbol{\beta}_m^{(l)}$
- Activation function $h(\cdot)$, e.g. sigmoid, linear, etc.
- Output: $\mathbf{x}_m^{(l)} = h\left(\boldsymbol{\beta}_m^{(l)\prime}\mathbf{x}^{(l-1)}\right)$. This can be feed to further neurons.

Approximating function



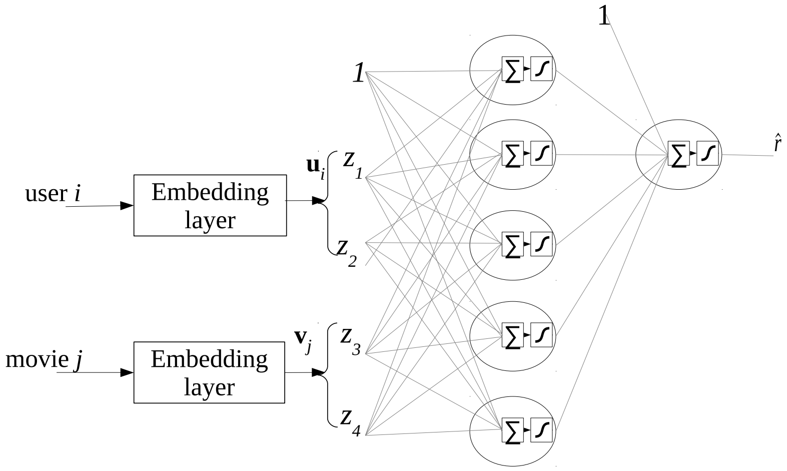
$$\hat{r} = f_\beta(\vec{z}) = h\left(\vec{\beta}^{(2)\,t} \cdot \vec{x}^{(1)}\right) = h\left(\sum_m \boxed{\beta_m^{(2)}} \; \boxed{h\left(\boxed{\vec{\beta}_m^{(1)}} \cdot \vec{z}\right)}\right)$$

Architecture for recommenders



$$\hat{r}_{ij} = f_\beta(\mathbf{u}_i, \mathbf{v}_j)$$

Training



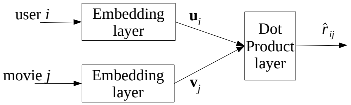
Training: Find β and the embedding $i \to \mathbf{u}_i$ and $j \to \mathbf{v}_j$ such that

$$\min \sum_{(i,j)\text{ observed}} ||r_{ij} - \hat{r}_{ij}||^2 = \min \sum_{(i,j)\text{ observed}} ||r_{ij} - f_\beta(\mathbf{u}_i, \mathbf{v}_j)||^2$$

Comparison

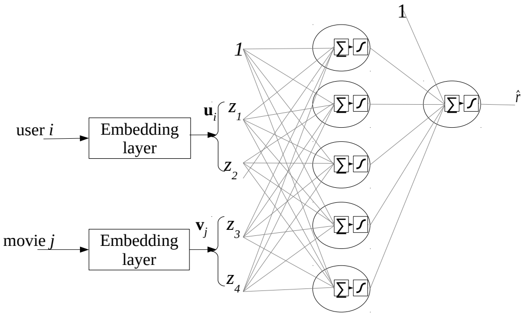
Matrix Factorization

$$\hat{r}_{ij} = \mathbf{u}_i \cdot \mathbf{v}_i$$



Neural Network

$$\hat{r}_{ij} = f_\beta(\mathbf{u}_i, \mathbf{v}_i)$$

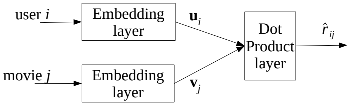


Advantages? Disadvantages?

Comparison

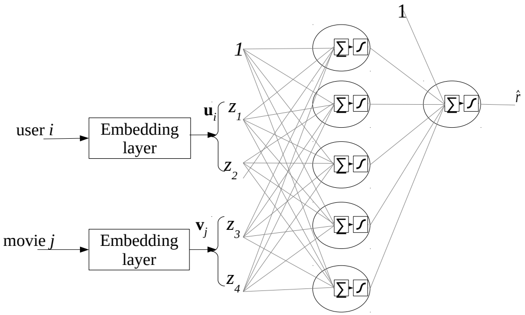
Matrix Factorization

$$\hat{r}_{ij} = \mathbf{u}_i \cdot \mathbf{v}_i$$



Neural Network

$$\hat{r}_{ij} = f_\beta(\mathbf{u}_i, \mathbf{v}_i)$$



Advantages? Disadvantages?

- Neural Networks are more general? But ...
- More difficult to train (Embedding + β)

Universality

- **Universal approximation theorem**:
  For any continuous function $g(\cdot)$, there exist a neural network with one hidden layer that approximates it, i.e., $f_\beta(\cdot) \approx g(\cdot)$.
- Therefore, we can find a neural network that can approximate $g(\mathbf{u}_i, \mathbf{v}_j) = \mathbf{u}_i \cdot \mathbf{v}_j$
- So, with Neural Networks we can approximate Matrix Factorization but also do many more ... $\Rightarrow$ **generality**.
- But the theorem does not tell how many neurons we need! We may need a huge number.

Recommender Systems with Neural Networks

Let us code ....

Section 5

Other improvements

Blending

- The best teams of Netflix Prize used > 100 models
- Blend: the prediction is a linear model of the result of other models

$$\hat{r}_{ij} = \sum_l w_l \cdot g_l(i,j)$$

- $\hat{r}_{ij}$: final prediction of rating of user i on movie j
- $g_l(i,j)$: prediction by the l-th model
- w_l weight, computed by linear regression
- We use models as predictors of a bigger model.

Blending

$$\hat{r}_{ij} = \sum_l w_l \cdot g_l(i,j)$$

- Model l may be better than the others depending on regions of (i,j) (e.g., amount of ratings of i or on j)

$$\hat{r}_{ij} = \sum_l w_l(i,j) \cdot g_l(i,j)$$

- How do we get $w_l(i,j)$? We need to train the blended model.

Stars vs. Thumb up / Thumb down

- Stars were not ratings from other users, but a prediction of how much you will like the movie.
- They have been removed now. Why? [McA17]

Stars vs. Thumb up / Thumb down

- Stars were not ratings from other users, but a prediction of how much you will like the movie.
- They have been removed now. Why? [McA17]
- Not good for recommendations
  - Users were objective: “I do not like the movie but, to be honest, it is a good movie. I rate it 4 stars.” What is the consequence?

Stars vs. Thumb up / Thumb down

- Stars were not ratings from other users, but a prediction of how much you will like the movie.
- They have been removed now. Why? [McA17]
- Not good for recommendations
  - Users were objective: “I do not like the movie but, to be honest, it is a good movie. I rate it 4 stars.” What is the consequence?
  - The system was proposing movies “objectively good”, even if the user may not like it
  - No incentive in rating (the user do not see the improvement of suggestions)
- 200% more ratings now!

RMSQ is not enough

- Just looking at the RMSQ may be misleading
- Final Goal: suggest items that a user will like
- The absolute value of the RMSQ is not important
- Heterogeneity of propositions is important
- Context
- RMSQ weights equally low and high rate, while only high rate items matter.
  - Evaluate error only on high rate data points.

Open dataset: watch out for lawyers!

From wikipedia:[2]

- Although the data sets were anonymized ... in 2007 two researchers from the University of Texas were able to identify individual users by matching the data sets with film ratings on the Internet Movie Database(IMD) ...
- In 2009, an anonymous Netflix user sued Netflix...
- This, as well as concerns from the Federal Trade Commission, led to the cancellation of a second Netflix Prize competition in 2010.

[2] `https://en.wikipedia.org/wiki/Recommender_system`

Summary

- Recommendation Systems and Collaborative Filtering (CF)
- User based CF
- Item based CF
- Singular Value Decomposition
- Blending
- Other tricks for recommendation systems

If you want to know more

- Bell, R. M., Park, F., Volinsky, C., & Park, F. (2008). The BellKor 2008 Solution to the Netflix Prize, (12), 121.
- Grisel O., Neural Networks for Recommender Systems
- Embedding applied to graphs [Fra19].

References I

- Charu C. Aggarwal, *Recommender Systems*, 2016.
- Alicia Frame, *Graph Embedding*, https://youtu.be/oQPCxwmBiWo, 2019.
- Nathan McAlone, *The exec who replaced Netflix's 5-star rating system with 'thumbs up, thumbs down' explains why*, 2017.
- Ian MacKenzie, Chris Meyer, and Steve Noble, *How Retailers can keep up with Consumers*, Tech. report, 2013.
- Nick Pentreath, *Deep Learning for Recommender Systems*, 2018.
- Harald Steck, *Matrix Factorization for Movie Recommendations*, 2015.

Section 6

Additional material

User-based collaborative filtering

- To predict the rating of a user u on an item i, I observe what similar users rated.
- Similarity between users u, v: correlation:

$$\rho_{uv} = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u) \cdot (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}$$

- I_{uv}: items rated by both u, v
- $r_{u,i}$: rating of user u on item i
- $\bar{r}_u$: average rate of user u

User-based collaborative filtering

- Training: computing pairwise ρ_{uv}
- Prediction with K Nearest Neighbors (KNN) method
- We take the K users v most similar to u

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in K} \rho_{uv} \cdot (r_{vi} - \bar{r}_v)}{\sum_{v \in K} |\rho_{uv}|}$$

- Problems
  - Information is **sparse**: Difficult to find other users that rated the same things as u

	Terminator	Die Hard	Titanic	Pretty Woman	Love Actually
Alice	0.1		0.9		
Bob		0.1		1	
John		1	0.1		
Ruby		1			0.2
Nick	0.1			1	
Paul		0.2	1		

User-based collaborative filtering

- Another[3] similarity measure between users u, v: cosine:

$$\rho_{uv} = \frac{(\mathbf{r}_u - \bar{r}_u\mathbf{1})' \cdot (\mathbf{r}_v - \bar{r}_v\mathbf{1})}{||\mathbf{r}_u - \bar{r}_u\mathbf{1}|| \cdot ||\mathbf{r}_v - \bar{r}_v\mathbf{1}||}$$

- $\mathbf{r}_u = (r_{u1}, r_{u2}, \ldots)'$
- Cosine of the angle between the vectors
- Self-damping effect

[3] M. Ekstrand. Similarity Functions for User-User Collaborative Filtering

Item-based collaborative filtering

- 1998: Amazon Patent US6266649B1
- Training: compute similarity ρ_{ij} between items i, j

$$\rho_{ij} = \frac{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i) \cdot (r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{u,j} - \bar{r}_j)^2}}$$

- U_{ij}: users who rated both items i, j
- $\bar{r}_i$: average rate for item i
- Predict

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in K} \rho_{ij} \cdot (r_{uj} - \bar{r}_j)}{\sum_{j \in K} |\rho_{ij}|}$$

- It overcomes the problems of user-based

Item-based collaborative filtering

- Ex. positive correlation: Harry Potter and Lord of Rings
  If you like one, you will like the other
- Ex. negative correlation: Saving Private Ryan vs. Godzilla
  If you like one, you will hate the other