Factorization Machine and Recommendation Systems

Tzviel Frostig

November 30, 2019

Table of contents

Factorization Machines

Recommendation systems and Matrix Factorization methods

- Matrix Factorization Techniques for Recommender Systems - Yehuda Koren, Robert Bell and Chris Volinsky (2009)
- Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model- Yehuda Koren (2008)

Factorization Machines

- Factorization Machines - Steffen Rendle (2010)
- Factorization Machines with libfm - Steffen Rendle (2012)

Part I

Introduction

Papers for Talk

- A recommendation system goal is to predict a user response to various options
- Common machine learning task, used in: music and video playlist generators, product recommendations, online dating, restaurants and more
- Data of typical problem is characterized by large amount of missing values
- A hard problem

Netflix Prize

- In 2006, Netflix offered a prize of 1m$ to whoever could improve their user prediction algorithm by 10%
- The competing teams were given a data set of 480k users and 18K movie, most of it empty
- The prize was finally won at 2009 using methods based on Matrix Factorization

Recommendation Systems - Data Example

The example that will follow us throughout the lecture, movie preferences, inspired by the Netflix competition.

Users / Movies	Titanic	Ghost	Terminator	Hot Fuzz	...
Guy	5	?	?	2	...
Roei	5	?	1	?	...
Ayala	?	?	4	1	...
...	5	2	?	5	...
Miriam	?	3	?	?	...

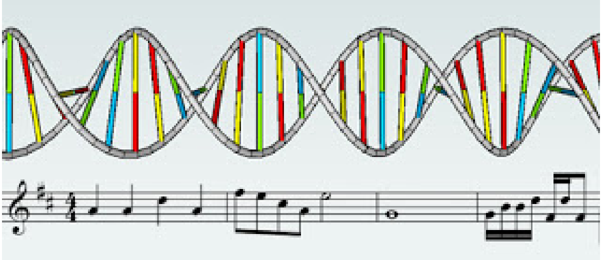
Recommendation Systems

Recommendation systems are based on one of two strategies:

- Content Based - Create a profile for each user or product to characterize it. Recommend a product according to the attributes.
- Collaborative Filtering - A method of making prediction (filtering) about interests of the user by collection preference from many users (collaborating).
    1. Neighborhood based approach (find similar users, K-NN for example)
    2. Latent factor methods, (find some hidden structure in the data)

Example - Content Based

- Content Based - Pandora recommendation system, each song is scored using the Music Genome Project by a specialist. As the user likes and dislikes songs, the model 'learns' the preferred features and suggests similar songs.



Example - Neighbor Based Method

- Joe likes 'Angel Has Fallen' 'Troy' and 'Predator' which is liked by Trump, Putin and James, which all like 'Crank' therefore that would be the recommended movie.

Latent Factor Method

- Find $k,\ k << p$ latent factors which characterize items and users, the factors correspond to the music Genes.
- The latent factors can be for example comedy vs drama, amount of action and the like.
- For users, each factor measures how much the user likes movies that score high on the corresponding movie factor.

Example - Latent Factor Method

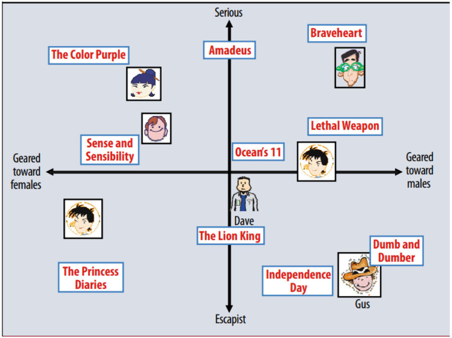


Figure: Taken from Matrix Factorization Techniques For Recommender Systems -Yehuda Koren, Robert Bell Bell and Chris Volinsky

- We would expect Gus to love Dumb and Dumber, to hate The Color Purple, and to rate Braveheart about average.
- How can we factor a matrix in such a way?

SVD

If M is a $n \times p$ matrix, its Singular Value Decomposition (SVD) is

$$M = U\lambda V'.$$

U is a $n \times n$ orthonormal matrix (unit eigenvectors of MM'), λ a $n \times n$ rectangular diagonal matrix, and V' is a $p \times p$ orthogonal matrix (eigenvectors of $M'M$).

Reminder

Define V as the eigenvectors of $M'M$

$$M'MV = V\lambda \rightarrow MM'MV = MV\lambda. \tag{1}$$

$U = MV\lambda^{-1}$ are the unit eigenvectors of MM',

$$U = MV\lambda^{-1} \rightarrow U\lambda V' = M \tag{2}$$

SVD & PCA

- As statistician we are more familiar with Principle Component Analysis (PCA)
- Note that PCA is SVD for M' and M

$$M'M = V\lambda U'U\lambda V' \rightarrow M'M = V\lambda^2 V$$

SVD - Example For Dimensionality Reduction

- Can SVD be used to for dimensionality reduction?
- A lower rank matrix factorization cant be found, by removing vectors which contribute little to the explained variance

SVD - Dimensionality Reduction

Using SVD we can take the only the 200 first eigenvectors, below are the first 10

Almost perfect reconstruction

Figure: Images are taken Nicolas Hug blog

SVD for Recommendation Systems

- U columns can be viewed as the various typical user
- V columns can be viewed as the various typical movies
- λ can be viewed as how prevalent is the combination of specific typical movie and user
- Seems suitable for our problem, can we use it for our recommendation problem?

Problem I

- SVD assumes that M is a dense matrix, i.e no missing values

Users / Movies	Titanic	Ghost	Terminator	Hot Fuzz	...
Guy	5	?	?	2	...
Roei	5	?	1	?	...
Ayala	?	?	4	1	...
...	5	2	?	5	...
Miriam	?	3	?	?	...

- Data is characterized by mostly missing value, less than 99% of the table is full

Solution I - Imputation

What wrong with imputation?

- Computationally expensive, as the most of the entries of the matrix are missing
- It can distort the data

Solution II - SVD MF

- Suggested by Simon Funk, played a crucial role in the winning teams algorithms for the Netflix prize
- Inspired by SVD, but allows for sparse matrices

Define $r_{u,i} \in R$, the rating user u gave to item i, where the rows represent users and the columns represent items. The optimization task is

$$\min_{q_i^*, p_u^*} \sum_{r_{i,u} \text{ exists}} (r_{u,i} - q_i' p_u)^2$$

where q_i measure the score of item i for the k latent factors and p_u measures the 'interest' of the user in those latent factors.

Solution II - SVD MF

It is clear that such a direct approach will lead to over fitting, and so to deal with it we add regularization

$$\min_{q_i^*, p_u^*} \sum_{r_{i,u} \ exists} (r_{u,i} - q_i' p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2). \tag{3}$$

Should be familiar to us as ridge regression, λ controls the 'strength' of the regularization.

Solution II - SVD MF

Since the model only includes interactions, it makes sense to add the bias (intercept), involving the rating, obtaining

$$\min_{q_i^*,p_u^*,\mu^*,b_u^*,b_i^*} \sum_{r_{i,u}\ exists} (r_{u,i}-\mu-b_u-b_i-q_i'p_u)^2+\lambda(||q_i||^2+||p_u||^2+b_u^2+b_i^2)$$

where b_i is the user effect i, b_u is the item effect u and μ is the global mean.

How To Optimize?

- The original method for optimizing suggested by Simon Funk was using Stochastic Gradient Descent (SGD)
- An alternative method is Alternating Least Squares (ALS), or Coordinate Descent

The error of the predictive model $\hat{r}_{u,i}$ is defined as

$$\epsilon_{u,i} = r_{u,i} - \hat{r}_{u,i}$$

SGD

Obtaining the gradients for Eq. 3 is simple

$$
\begin{aligned}
q_i &\leftarrow q_i + \gamma(\epsilon_{u,i} - \lambda q_i) \\
p_u &\leftarrow p_u + \gamma(\epsilon_{u,i} - \lambda p_u)
\end{aligned}
$$

where γ is the learning rate. Cycle through all pairs of User / Items interactions until convergence.

ALS

- If q_i or p_u were known, Eq. 3 was convex and could be solved using least squares
- Alternating between p_u and q_i, at each iteration solving using least square ensures a decrease of the loss and eventually convergence
- Since each q_i and p_u are obtained individually, it is easy to parallelize.

Summary

- SVD- MF offers a reasonable method to model User - Item interactions
- Two simple algorithms for optimization
- Whats missing?

Cold Start Problem

From Wikipidea

"Cold start is a potential problem in computer-based information systems which involve a degree of automated data modeling. Specifically, it concerns the issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information."

- What happens when a user supply very few ratings, how can we predict his/her preference?

Implicit Preferences

- In many cases we have additional information regarding the users, in the Netflix data, we could have not only the rated movies but also the history of rented videos
- The data is not limited to implicit preferences, but also to information such as demographics
- Define $x_i \in N(u), |x_i| = k$ the set of items for which user u showed implicit preference, $p_u \in R(u)$ the explicit preferences of the user and $y_a \in A(u), |y_a| = k$ is the set of attributes that describe a user

SVD++

SVD++ allows us to incorporate the implicit data, notice we also find factors of the implicit data

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i' \left(p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_\alpha \right) \quad (4)$$

We are now minimize across p_u, q_i, x_i and y_a.

- The factorization model with the best results for the Netflix data
- But what do we do when a new user arrives?

Asymmetric SVD

The Asymmetric SVD prediction is

$$\hat{r}_{u,i} = b_{u,i} + q_i'(|R(u)|^{-0.5} \sum_{j \in R(u)} (r_{u,j} - b_{u,j}) z_j + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i)$$

where $b_{u,i} = \mu + b_u + b_i$. p_u is replaced, no longer is each user parametrized by vector p_u.

Asymmetric SVD - Benefits

- Since Asymmetric-SVD does not parameterize users, we can handle new users as soon as they provide feedback to the system, without needing to re-train the model and estimate new parameters. For new items we do need to re-train the model.
- Integration of implicit data, similar to SVD++ takes into account additional information regarding the user

Results on Netflix Data

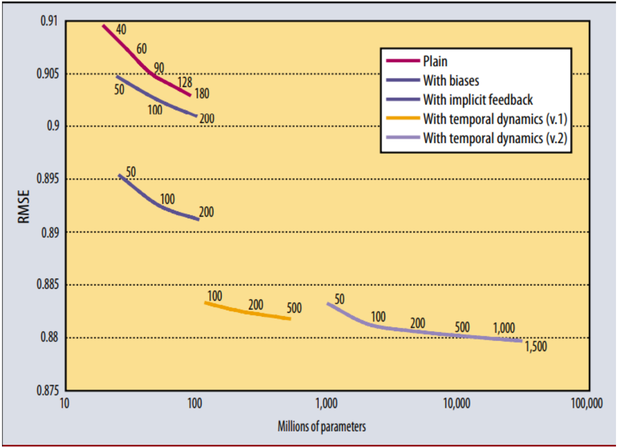


Figure: The RMSE for winning the prize is 0.08563, Netflix system achieved 0.9514

The First Two Factors

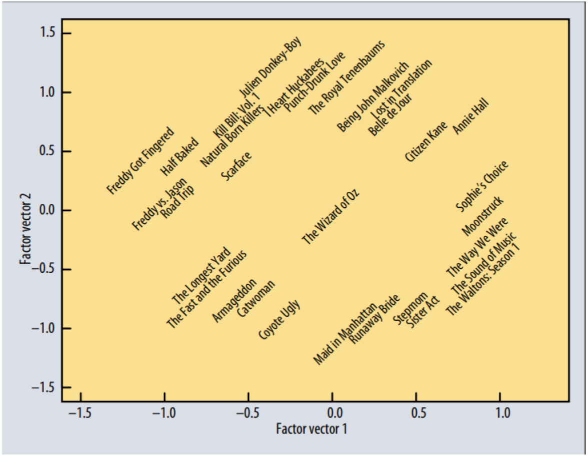


Figure: Taken from Matrix Factorization Techniques For Recommender Systems -Yehuda Koren, Robert Bell Bell and Chris Volinsky

Are we done?

- There are many other models each incorporate additional information, such as confidence level, temporal dynamics and more, each requires to create the data matrix in a specific way
- It seems that for each scenario one should use a specialized algorithm

Part II

Factorization Machines

Factorization Machines

- Suggested by Steffen Rendle (2010)
- Successful in tasks such as collaborative filtering and click rate prediction
- Implemented in a few python packages, tesnorFlow and libFM

Factorization Machines

- A general predictor, can be used for classification, regression and ranking
- Generalizes Factorization Models, mostly used in recommendation systems
- Model based
- Linear in time

Factorization Machines

- A general predictor, can be used for classification, regression and ranking
- Generalizes Factorization Models, mostly used in recommendation systems
- Linear in time
- In the words of the author ".. combines the advantages of Support Vector Machines (SVM) with factorization models"

Data

This is not necessary, as the model will work on any data set.

Feature vector $\mathbf{x}$ | Target y

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

Figure: Taken from Factorization Machines - Steffen Rendle

Model

Lets begin from the basic regression model

$$\hat{y}(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{p} x_i \beta_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} w_{i,j} x_i x_j \tag{5}$$

Or we can write it in matrix form as

$$\hat{y}(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}' \boldsymbol{\beta} + W \boldsymbol{x} \boldsymbol{x}' \tag{6}$$

$$\beta_0 \in R \quad \boldsymbol{\beta} \in R^p \quad W \in R^{p \times p}$$

.

where $W_{i,j} = 0 \; \forall j < i$.

Model

$$\hat{y}(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}'\boldsymbol{\beta} + W\boldsymbol{x}\boldsymbol{x}' \tag{7}$$

- Nothing new, linear regression is dated to 1877 (Galton)
- Cant actually be evaluated in our case, since the data is extremely sparse
- Let's use Matrix Factorization

$$W = VV'$$

where $V \in R^{p \times k}$. V represent our latent factors.

Model

$$\hat{y}(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}'\boldsymbol{\beta} + W\boldsymbol{x}\boldsymbol{x}' \tag{8}$$

- Nothing new, linear regression is dated to 1877 (Galton)
- Cant actually be evaluated in our case, since the data is extremely sparse
- Let's use Matrix Factorization

$$W = VV'$$

where $V \in R^{p \times k}$. V represent our latent factors.