Factorization Machines (FM)- novel but proved and promising approach

Evgeniy Marinov

July 2020

Introduction to FM

- In 2010, Steffen Rendle (currently a senior research scientist at Google), introduced a seminal paper [Rendle - FM].
- Many years have passed since such an impactful algorithm has been introduced in the world of ML /Recommender Systems/.
- FMs are a new model class which combines the advantages of Support Vector Machines(SVM)/polynomial regression and factorization models.

Kaggle Competitions

Kaggle Competitions won with Factorization Machines:

- Criteo's CTR on display ads contest - 2014
  `https://www.kaggle.com/c/criteo-display-ad-challenge`
  `http://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf`

Kaggle Competitions

Kaggle Competitions won with Factorization Machines:

- Criteo's CTR on display ads contest - 2014
  `https://www.kaggle.com/c/criteo-display-ad-challenge`
  `http://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf`
- Avazu's CTR prediction context
  `https://www.kaggle.com/c/avazu-ctr-prediction`
  `http://www.csie.ntu.edu.tw/~r01922136/slides/kaggle-avazu.pdf`

Kaggle Competitions

Kaggle Competitions won with Factorization Machines:

- Criteo's CTR on display ads contest - 2014
  https://www.kaggle.com/c/criteo-display-ad-challenge
  http://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf
- Avazu's CTR prediction context
  https://www.kaggle.com/c/avazu-ctr-prediction
  http://www.csie.ntu.edu.tw/~r01922136/slides/kaggle-avazu.pdf

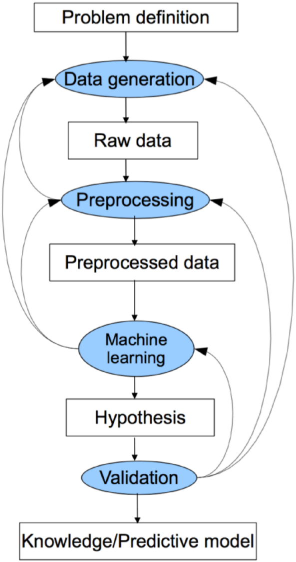
In both competitions the LogLoss function for evaluation of the classifiers has been used - the aim is to **minimize** the Logrithmic Loss.

- $LogLoss = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) - (1 - y_i)\log(1 - p_i)]$
  with binary classification

Task Description

- We are given (training and testing) dataset /images, documents,.../
  $D = \{(x^{(l)}, y^{(l)})\}_{l \in L} \subset \mathbb{R}^n \times T$
- The aim is to find a function $\hat{y} : \mathbb{R}^n \longrightarrow T$ (target space), which estimates the testing data /unknown to the trained model/.
- In recommender systems (Online advertising) we deal with sparse input vectors $x^{(l)} \in \mathbb{R}^n$.
- T can be $\{-1, 1\}$, $\{0, 1\}$ (classification), $\mathbb{R}$ (regression) or some categorical space (ranks for an item).

Task Description



Pipeline

- Each step generates many questions:
  - Data generation: data types, sample size, online/offline...
  - Preprocessing: normalization, missing values, feature selection/extraction...
  - Machine learning: hypothesis, choice of learning paradigm/algorithm...
  - Hypothesis validation: cross-validation, model deployment...

Ad Classification

- Classification from implicit feedback - the user does not rate explicitly..
- But we can "guess" items with which features he likes or does not care about.

Country	Day	Ad Type	Clicked ?
USA	3/3/15	MOVIE	1
China	1/7/14	GAME	0
China	3/3/15	GAME	1

Standard (dummy) encoding

USA	China	3/3/15	1/7/14	MOVIE	GAME	Clicked ?
1	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1

- Very large feature space
- Very sparse samples (feature values)

(De)motivation!

Often features are more important in pairs:

$$"Country == USA"\&"Day == Thanksgiving"$$

If we create a new feature (conjunction) for every pair of features..?!?

(De)motivation!

Often features are more important in pairs:

$$"Country == USA" \& "Day == Thanksgiving"$$

If we create a new feature (conjunction) for every pair of features..?!?

- Feature space: insanely large
  If originally n features $\longrightarrow$ additional $\binom{n}{2} = \frac{n(n-1)}{2!}$
- Samples: still sparse

Problems with feature types

- Big number of features -> Dimensionality reduction -> SVD, PCA
  - **Dimensionality reduction**: "compress" the data from a high-dimensional representation into a lower-dimensional one (useful for visualization or as an internal transformation for other ML algorithms)
- Sparse features -> Hashing

SVD with ALS

Singular Value Decomposition (Matrix Factorization) with Alternate Least Squares.
The most basic matrix factorization model for recommender systems models the rating $\hat{r}$ a user u would give to an item i by:

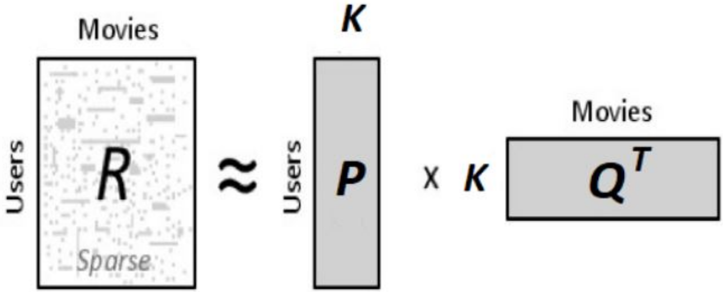
$$\hat{r}_{ui} = x_u^T y_i,$$

where

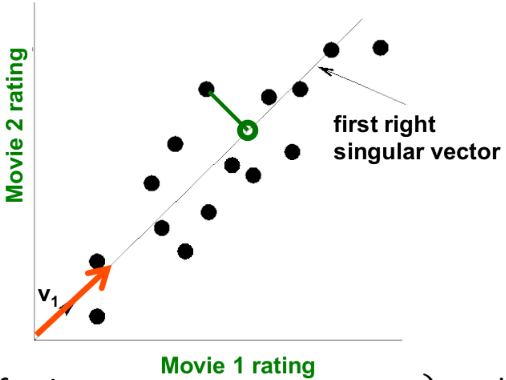
- $x_u^T = (x_u^1, \cdots, x_u^N)$ - factor vector associated to the user
- $y_i^T = (x_y^1, \cdots, y_i^N)$ - factor vector associated to the item
- The dimension N of the **factors** x_u^T, y_i^T is the **rank** of the model (factorization).

Factorization Models

Collaborative Filtering:
- Neighborhood Methods
- Latent Factor Methods



SVD – Dimensionality Reduction



- Instead of using two coordinates (x, y) to describe point locations, let's use only one coordinate (z)
- Point's position is its location along vector v_1

SVD - Dimensionality Reduction

More details

- Q: **How exactly is dim. reduction done?**
- **A: Set smallest singular values to zero**

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\approx
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

SVD - Dimensionality Reduction

More details

- **Q: How exactly is dim. reduction done?**
- **A: Set smallest singular values to zero**

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\approx
\begin{bmatrix}
\mathbf{0.13} & 0.02 & \cancel{-0.01} \\
\mathbf{0.41} & 0.07 & \cancel{-0.03} \\
\mathbf{0.55} & 0.09 & \cancel{-0.04} \\
\mathbf{0.68} & 0.11 & \cancel{-0.05} \\
0.15 & \mathbf{-0.59} & \cancel{0.65} \\
0.07 & \mathbf{-0.73} & \cancel{-0.67} \\
0.07 & \mathbf{-0.29} & \cancel{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
\cancel{0.40} & \cancel{\mathbf{-0.80}} & \cancel{0.40} & \cancel{0.09} & \cancel{0.09}
\end{bmatrix}
$$

Advantages of Factorization Models

Factorization models:

- Factorization of higher order interactions
- Efficient parameter estimation and superior performance
- Even for sparse data where just a few or no observations for those higher order effects are available

Main applications:

- Collaborative Filtering (in Recommender Systems)
- Link Prediction (in Social Networks)

Disadvantages of Factorization Models

BUT are not general prediction models. There are many factorization models designed for specific tasks.

- Standard models:
  - Matrix Factorization (MF)
  - Tensor Factorization (TF)
- Specific tasks:
  - SVD++ [Koren, Bell - Advances in CF]
  - timeSVD++ [Koren, Bell - Advances in CF]
  - PIFT (Pairwise Interaction Tensor Factorization)
  - FPMC (Factorizing Personalized Markov Chains)
  - And others...

Advantages of FMs

- FMs allow parameter estimation under very sparse data where SVMs fail.
- FMs have linear (and even "almost constant"!) complexity and don't rely on support vectors like SVMs.
- FMs are a general predictor that can work with any real valued feature vector. In contrast, other state-of-the-art factorization models work only on very restricted input data.
- Through suitable feature engineering, FMs can mimic state-of-the-art models like MF, SVD++, timeSVD++, PARAFAC, PITF, FPMC and others.

Model

Feature vector **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	$y^{(6)}$

Notations

- For any $x \in \mathbb{R}^n$ let us define:
  - $m(x)$ to be the number of the nonzero elements of the feature vector;
  - $\bar{m}_D$ to be the average of $m(x^{(l)})$ for all $l \in L$;
- Since we deal with huge sparsity, we have that $\bar{m}_D \ll n$ - the dimensionality of the feature space.
- One reason for huge sparsity is that the underlying problem deals with large categorical variable domains.

Definition of Factorization Machine Model of degree 2

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j \qquad (1)$$

where the model parameters that have to be estimated are:

- $w_0 \in \mathbb{R} -$ global baias (mean rating over all items/users)
- $w = (w_1, \ldots, w_n) \in \mathbb{R}^n -$ weights of the corresponding features
- $v_i := (v_{i,1}, \ldots, v_{i,k})$ within $V \in \mathbb{R}^{n \times k}$ describes the i-th variable(feature) with k factors.

Complexity of the model equation

- The model equation $\hat{y}(x)$ for every $x \in D$ can be computed in linear time $\mathcal{O}(kn)$ and under sparsity $\mathcal{O}(km(x))$

Proof:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j =$$
$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle v_i, v_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^{n} \langle v_i, v_i \rangle x_i x_i =$$
$$\frac{1}{2} [\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i] =$$
$$\frac{1}{2} \sum_{f=1}^{k} [(\sum_{i=1}^{n} v_{i,f} x_i)(\sum_{j=1}^{n} v_{j,f} x_j) - \sum_{i=1}^{n} v_{i,f}^2 x_i^2] =$$
$$\frac{1}{2} \sum_{f=1}^{k} [(\sum_{i=1}^{n} v_{i,f} x_i)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2]$$

Complexity of parameter updates

The needed computation to get the partial derivatives of $\hat{y}(x)$ is done while computing the computation of $\hat{y}(x)$. Therefore,

- All parameter updates for a case (x, y) can be done in $\mathcal{O}(kn)$ and under sparsity $\mathcal{O}(km(x))$.

$$\frac{\partial \hat{y}(x)}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2 & \text{if } \theta \text{ is } v_{i,f} \end{cases} \tag{2}$$

General applications of FM

FMs can be applied to a variety of prediction tasks.

- **Regression:** $\hat{y}(x)$ can be directly applied as a regression predictor
- **Binary classification:** The sign of $\hat{y}(x)$ is used and the parameters are optimized for hinge loss or logit loss.
- **Ranking:** Input vectors x are ordered by the score of $\hat{y}(x)$ and optimization is done over pairs of instance vectors with a pairwise classification loss ([Joachims - Ranking CTR]).

Inference

Inference algorithms:

- SGD (Stochastic Gradient Descent)
- ALS (Alternate Least Squares)
- MCMC (Monte Carlo Markov Chains)

Library: libFM (C++)

- Author: Steffen Rendle
- Web site: `http://www.libfm.org/`
  (very poor description of the library)
- Source code (C++): `https://github.com/srendle/libfm`
  Very professionaly written source code! (Linux and MacOS X)
- libFM features
  - stochastic gradient descent (SGD)
  - alternating least squares (ALS)
  - Bayesian inference using Markov Chain Monte Carlo (MCMC)

Library: fastFM (Python)

- Author: Immanuel Bayer
- Web site: `http://ibayer.github.io/fastFM/`
  Very detailed tutorial for the usage of the library!
- Source code (Python):
  `https://github.com/ibayer/fastFM`
  Python (2.7 and 3.5) with the well known scikit-learn API.
  Performence critical code in C and wrapped with Cython.

Task	Solver	Loss
Regression	ALS, SGD, MCMC	Square Loss
Classification	ALS, SGD, MCMC	Probit(MAP), Probit, Sigmuid
Ranking	SGD	[Rendle - BPR]

Library: LIBFFM (C++)

A library for Field-aware Factorization Machines from the Machine Learning group at National Taiwan University

- Author: YuChin Juan and colleagues
- Web site: `https://www.csie.ntu.edu.tw/~cjlin/libffm/` (very poor description of the library)
- Source code (C++): `https://github.com/guestwalk/libffm`
- FFM is prone to overfitting and not very stable for the moment.

Table of contents

Thank you for your attention!

References

- Steffen Rendle *Factorization Machines.* 2010. `http://www.ismll.uni-hildesheim.de/pub/pdfs/Rendle2010FM.pdf`
- Steffen Rendle, *Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. UAI '09, pages 452* 2009
- Koren Y. and Bell R. *Advances in Collaborative Filtering.*
- Blondel M. *Convex Factorization Machines.* http://www.mblondel.org/publications/mblondel-ecmlpkdd2015.pdf
- Thorsten Joachims, *Optimizing Search Engines using Clickthrough Data* 2002.