Factorization Machines

Marjolijn Cornelissen

Research Paper Business Analytics
Vrije Universiteit Amsterdam

Abstract. *Factorization Machines* are known to address many weaknesses of machine learning models. They can handle sparse datasets, they learn weights to interactions between variables which are factorized, and the obtained model equation does not depend on training examples. To show the effectiveness of *Factorization Machines*, an experiment is carried out to show that *Factorization Machines* outperform some other machine learning models. Results of the experiment do indeed show that *Factorization Machines* outperform the other models. The results also show differences in performance of *Factorization Machines* when changing some hyperparameters of the model. Using the learning approach Alternating Least-Squares and increasing the value of the number of dimensions of the latent parameter vector, gave the best performance for this experiment.

Introduction

Machine learning tries to recognize patterns in data and based on that, algorithms are created that predict and optimize. The use of machine learning continues to grow. Various models can be used for machine learning. Depending on the task to be performed, a suitable learning model is selected. Supervised learning is one of the tasks in machine learning, where a function is learned that maps an input to an output based on example input-output pairs. This paper focuses on a relatively new supervised machine learning model, called Factorization Machines (FMs). FMs were introduced by Steffen Rendle in [2]. FMs are an impactful model [13] and have shown excellent prediction capabilities.

FMs are especially popular in predicting the click-through rate (CTR) or for recommender systems. In this paper, an experiment is being conducted to determine how well FMs perform. In this experiment, three machine learning models, including FMs, will be used to perform the same task. To determine how well FMs perform, the results of the different models will be compared.

This research paper is divided into four sections. In the first section a study of relevant literature will be discussed. In Section 2 FMs are introduced. Next, Section 3 will describe the dataset that will be used for the experiment. Finally, in Section 4, the experimental setup and results are shown, together with some important conclusions.

1. Related work

Factorization Machines (FMs) are often used in tasks to predict whether an advertisement will be clicked (i.e. click-through rate) or to predict what relevant information is for a user of a webpage (i.e. recommendation systems). For those tasks, large datasets are usually used. In most cases the datasets are sparse, which means that only a few variables for each input vector of predictor variables are non-zero. Therefore, in most machine learning models, no importance is given to several variables for prediction. FMs are a suitable model class to use in such cases. FMs were first introduced in 2010 by S. Rendle [2]. FMs combine the advantages of Support Vector Machines (SVMs) with factorization models. Several published articles describe the comparison of SVMs and other factorization models to FMs. Therefore, this comparison will be explained first in this section. After that, some published articles are discussed about the performance of FMs.

1.1 FMs vs SVMs

SVMs, which is one of the most popular models in machine learning, uses the so-called kernel trick to find an optimal boundary between possible outputs. The optimal boundary, called hyperplane, is influenced by the support vectors. The support vectors are data points that are close to the hyperplane.

The similarity between FMs and SVMs is that both model nested interactions between variables (when using a polynomial kernel in SVMs). But, using SVMs, the weights given to those interactions are completely independent, and using FMs those weights are factorized. This means that using SVMs the weight given to the interaction between variable i and j ($w_{i,j}$) is independent of the weight given to the interaction between variable i and k ($w_{i,k}$). While using FMs, $w_{i,j}$ and $w_{i,k}$ depend on each other as they overlap and share parameters due to the factorization. Another benefit when using FMs is that making a prediction, using

the model equation obtained using training data, is independent of this training data. While using SVMs the prediction depends partly on the training data, because some data points in this training data (e.g. the support vectors) have some influence on the model equation. Both aforementioned advantages of FMs over SVMs are especially present when using very sparse data, this is because SVMs cannot learn reliable parameters in this case. Next to those benefits, FMs can learn the model parameters without transforming the data to the dual form. Using (non-linear) SVMs, the data (usually) needs to be transformed to the dual form.

1.2 FMs vs other factorization models

Before listing the advantage of using FMs over other factorization models, it is worth mentioning that several factorization models are often used for prediction tasks (especially click-through rate prediction and recommender systems) and they mostly outperform other machine learning models. For example, [1] describes the use of a factorization model, namely Matrix Factorization (MF), in the Netflix Challenge. The task in this challenge was to predict user ratings for movies offered by Netflix based on previous ratings. Using MF, the authors of [1] won the competition. Many more articles are published using MF, or other variants of factorization models, to win online competitions. Therefore, factorization models outperform other machine learning models quite often. Especially in settings like collaborative filtering [2].

A distinction can be made between standard factorization models (like PARAFAC or MF), which factorize a relation between categorical variables and specialized factorization models (like SVD++, PITF or FPMC), which are used for specific data and tasks. The benefit of FMs over standard factorization models is that they can work with any real-valued feature vector. This in contrast to the standard factorization models, which require feature vectors that are divided in m groups (when having m categorical variables) and in each group exactly one value has to be 1 and the rest 0 [3]. Therefore, FMs are a general predictor (just

like SVMs for example). Specialized factorization models are designed for a single task. This results in many different published articles, like [7] [8] [9] [10] [11] [12]. All these articles define a new model and a suitable learning algorithm. [2] showed that FMs can mimic many of the most successful factorization models just by feature extraction.

1.3 Performance of FMs

Thus, FMs bring together the advantages of SVMs (using a polynomial kernel) and factorization models: they are general applicable and predict accurately (even in sparse settings).
Because of [2], which says: “FMs combine the advantages of Support Vector Machines (SVMs) with factorization models”, many articles pay attention to the comparison in performance between FMs and SVMs, and between FMs and other factorization models. Not much published work compares the performance of FMs to other well-known and often used machine learning models. [5] does compare FMs to other models. They compare, among other things, the performance of a linear model and polynomial regression to the performance of FMs. Most of the time FMs outperform the linear model and the polynomial regression model. [6] does not compare FMs to other machine learning models directly, but the authors of the article have won the Expedia Hotel Recommendation challenge using FMs. This indirectly implies that it has better performance than other machine learning models.

2. Factorization Machines

In this section, *Factorization Machines* (FMs) will be introduced. FMs are an increasingly popular model that can be used in both classification and regression tasks. FMs can even be used when the data is very high-dimensional. To explain how FMs work, assume having a dataset $\boldsymbol{X} \in \mathbb{R}^{nxp}$. For every row i, the dataset can be described as follows, $(\boldsymbol{x_i}, y_i)$ where $\boldsymbol{x_i} \in \mathbb{R}^p$ is a vector of the p predictor variables (feature vector), and y_i its corresponding target. FMs model all nested interactions up to order d between the p predictor variables, using factorized interaction parameters. In this research, only the model with degree two (d = 2) will be considered, where all single and pairwise interactions between variables will be captured. FMs predict the output associated with an input vector, using equation (1). The first part of this equation of the model corresponds to standard linear regression, which associates a weight to each predictor variable. w_0 is the global bias, and w_{i} the weight of the i[th] variable. Linear regression only, does not take the interaction between variables into account. Therefore, the second part gives a weight to the interaction between the variables. It does not only give a weight to the product of variables ($x_i x_j$), but it models all possible interactions between the values in the feature vector, $\boldsymbol{x_i}$. Therefore, FMs break the independence of the interaction parameters by factorizing them (i.e. the data for one interaction helps to estimate the parameters for related interactions). So, $\widehat{w}_{i,j} = < v_{\mathrm{i}}, v_{\mathrm{j}} >$ is the factorized weight for the interaction between variables i and j. When having datasets with high sparsity, FMs are a good model to use because it can also estimate a weight for interactions between variables that are not observed.

$$\hat{y}_{FM}(x) := \underbrace{w_0 + \sum_{i=1}^{p} w_i x_i}_{linear\ regression} + \underbrace{\sum_{i=1}^{p} \sum_{j=i+1}^{p} x_i x_j \widehat{w}_{i,j}}_{\substack{breaking\ the\ independence\ of \\ interaction\ parameters}} \qquad (1)$$

$$where\ \widehat{w}_{i,j} = < v_{\mathrm{i}}, v_{\mathrm{j}} > = \sum_{f=1}^{k} v_{\mathrm{i,f}} v_{\mathrm{j,f}}$$

The parameters that need to be estimated in this model are $w_0 \epsilon \mathbb{R}$, $\boldsymbol{w} = \left[w_1, \ldots, w_p\right] \in \mathbb{R}^p$ and $\boldsymbol{V} = \left[v_{1,f}, \ldots, v_{p,f}\right]|_{f=1}^{k} \in \mathbb{R}^{pxk}$ which denotes a k-dimensional latent parameter vector.

To learn these model parameters, the following approaches can be used for FMs: Stochastic Gradient Descent (SGD), Alternating Least-Squares (ALS) or Markov Chain Monte Carlo (MCMC). Each of these approaches uses the loss function $L(\{y_i(x), \hat{y}_i(x)\}|_{i=1}^{n})$, which needs to be minimized. y_i denotes the target, and $\hat{y}_i$ the predicted value corresponding to the input feature vector $\boldsymbol{x_i}$. FMs can be applied to regression or classification prediction tasks. The loss function is determined for each of these prediction tasks differently. For a regression task, the squared loss is mostly used, and for a classification task, the log loss can be used. Most of the time the number of model parameters that need to be estimated is large. Because of this, FMs makes them prone to overfitting. Therefore, regularization term(s) may be included in the loss function.

3. Data

The dataset used in this research originates from an open machine learning competition on the website of *Kaggle*. The goal of this competition is to predict whether a mobile advertisement will be clicked. Data contains ten days worth of data from the company Avazu. The ten days are from 21 October 2014 – 30 October 2014 and has a total of 40,428,868 user sessions. The following assumption is made: the day of the week will not influence the click-through rate. This assumption is made because the available dataset only contains data of a few days in October, which is too few data to see if there is any pattern concerning the day of the week and whether an advertisement is clicked. Using this assumption, only the data of the first day will be used instead of using all ten days, to make computations easier due to the relatively small size of the data[1].

The data of the first day has a total of 4,122,995 user sessions. Every user session in the dataset is described by 24 variables. All variables are categorical, and are shown in Table 1a, together with the corresponding number of categories. An overview of how the dataset looks like is given in Table 1b.

[1] To make the models predict more accurately, it would be desirable to use all data. But since this research only focusses on the comparison of performance, and not to obtain the overall best performance, data of one day is enough.

Variables			*Number of categories*
Identified variables (all categorical)	Id		4,122,995
	Hour		24
	Position		7
	Site metadata	ID	2865
		Domain	3394
		Category	22
	Applica-tion metadata	ID	4154
		Domain	287
		Category	31
	Devices metadata	ID	368,962
		IP	1,048,575
		Model	6098
		Type	4
		Connection type	4
Anonymized variables (all categorical)	C1, C14 – C21		Resp.: 7; 785; 8; 9; 193; 4; 44; 168; 38
Target variable (binary)	Click		2

Table 1a: The variables of the Avazu dataset.

Overview dataset

click	*hour*	*banner_pos*	*site_id*	*site_domain*	. . .	*C21*
1	14102100	0	1fbe01fe	f3845767	. . .	79
0	14102100	1	fe8cc448	9166c161	. . .	157
			⋮			
0	14103023	1	f61eaaae	25d4cfcd	. . .	117

Table 1b: Overview of the dataset.
(Note: the variable 'hour' is given in the following format: YYMMDDHH).

There are two major classes of categorical variables –where the different categories of the variable have an order (ordinal), or without an order (nominal). Some of the variables, like site id, are not numeric. Those must be transformed into a numeric value before one can work with most machine learning models. The variables where an order can be found (ordinal) are easy to give a numeric interpretation. For nominal data, this transformation can be done using one-hot encoding. However, this way of transforming is not useful if there are many different

unique categories, because the number of columns in the data will increase immensely, which leads to the need for more memory space and computational power. For example, the variable device id has 368,962 unique categories, which results in 368,962 columns (i.e. new variables) instead of one. Because of limitations in memory, it is not possible to one-hot encode all variables. There are other techniques that can be used to transform categorical variables with many unique categories (for example, hashing). But since the aim of this research is to compare the performance of different models, and not to get the overall best performance, the variables with a large number of unique categories are removed. The remaining variables have been transformed using one-hot encoding.

After doing this, the data can be used to predict whether an ad will be clicked, using different machine learning models. The type of model that needs to be used is a supervised learning model. Because based on the obtained data, where examples of input and associated output (click or no click) are given, the algorithm learns how the properties of the input are decisive for the output. After this learning phase, supervised learning models can produce the right output for new input with a certain accuracy. The experimental setup and results are presented in the next section.

4. Experiment

Using the dataset described in the previous section, an experiment is conducted to see how well FMs perform. First, the setup of this experiment is described, followed by the results.

4.1 Experimental setup

To demonstrate the effectiveness of FMs, the performance is compared to the performance of *Logistic Regression* and *Decision Trees*. These models were chosen because they are both useful when having a binary target variable, in this research click or no click (1/0), and because they both belong to the most frequently used machine learning models at the moment.

To evaluate the performance of the three models, the classification loss function - logarithmic loss (log loss) - is used. This evaluation metric is appropriate when the output is the probability of a binary outcome, which is the case in this research. It defines how well the prediction is. It takes the confidence of the probability into account when assessing how to penalize incorrect classifications.

The formula for the log-loss (binary classification) equals:

$$L = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)),$$

where y_i and $\hat{y}_i$ represent the target and prediction respectively for every user.

Different hyperparameter settings are possible using FMs.

- The learning approach, which can be Stochastic gradient descent (SGD), Alternating least-squares (ALS) or Markov Chain Monte Carlo (MCMC).
- The k-dimensional latent parameter vector. To evaluate the impact of this hyperparameter, k=8 and k=100 are used.

For the *Decision Trees* the hyperparameter for the metric for measuring the best split in the tree can be tuned. This is measured using either Gini Impurity or Information Gain (IG). For *Logistic Regression* the hyperparameter for the optimization method can be tuned. Three optimization methods are investigated, a Newton method to solve the primal problem and Coordinate Descent (CD) to solve the dual problem.

4.2 Experimental results

As mentioned before, different hyperparameter settings are possible. The performance of the three machine learning models, measured in log loss, are presented in Table 2. In this table, the model is given – *Decision Trees, Logistic Regression* or *Factorization Machines* (FMs) – together with certain settings for the hyperparameters. In case of non-convexity, multiple iterations are performed to prevent to get stuck in a local optimum.

Machine learning model	*Log loss value*
Decision Tree (Gini)	0.468637
Decision Tree (IG)	0.469266
Logistic regression (CD)	0.436450
Logistic regression (Newton)	0.436744
Logistic regression (SAG)	0.438423
FMs (MCMC, k=8)	0.423139
FMs (MCMC, k=100)	0.420696
FMs (ALS, k=8)	0.418806
FMs (ALS, k=100)	0.416772
FMs (SGD, k=8)	0.423534
FMs (SGD, k=100)	0.423678

Table 2: performance of different machine learning models with different hyperparameter settings, measured using log loss (best performance = green, worst performance = red).

No matter what machine learning model is used, FMs outperform other models in terms of log loss. However, it is worth mentioning that the training time is longer than *Decision Trees* and *Logistic Regression*. For FMs, the learning approach ALS is significantly better that the other two in terms of log loss. Next to this, it can be obtained that increasing the number of dimensions in the latent parameter vector (k) results in better performance of FMs.

References

[1] Koren, Y., Bell, R.M., Volinsky, C., (2009): Matrix Factorization techniques for recommender systems. IEEE Computer 42(8), p.30-37.

[2] Rendle, S., (2010): Factorization Machines. IEEE International Conference on Data Mining, p.995-1000.

[3] Freudenthaler, C., Schmidt, L., Rendle, S., (2011): Factorization Machines, Factorized Polynomial Regression Models.

[4] He, X., Chua, T., (2017): Neural Factorization Machines for Sparse Predictive Analytics.

[5] Juan, Y., Zhuang, Y., Chin, W., (2016): Field-aware Factorization Machines for CTR Prediction.

[6] Liu, X., et al., (2013): Combination of Diverse Ranking Models for Personalized Expedia Hotel Searches.

[7] Koren, Y., (2008): Factorization meets the neighborhood, a multifaceted collaborative filtering model. ACM SIGKDD International Conference on Knowledge discovery and data mining, p.426-434.

[8] Koren, Y., (2009): Collaborative filtering with temporal dynamics. ACM SIGKDD International Conference on Knowledge discovery and data mining, p.447-456.

[9] Rendle, S., Freudenthaler, C., Schmidt-Thieme, L., (2010): Factorizing Personalized Markov Chains for Next-Basket Recommendation. International Conference on World Wide Web, p.811-820.

[10] Rendle, S., Schmidt-Thieme, L., (2010): Pairwise interaction tensor factorization for personalized tag recommendation. ACM International Conference on Web search and data mining, p.81-90.

[11] Salakhutdinov, R., Mnih, A., (2008): Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. International Conference on Machine Learning.

[12] Salakhutdinov, R., Mnih, A., (2008): Probabilistic Matrix Factorization.

[13] Harris, B., (2015): Factorization Machines, a new way of looking at machine learning. Retrieved from https://securityintelligence.com/factorization-machines-a-new-way-of-looking-at-machine-learning/