Factorization Machines

1.Introduction

Various models can be used for machine learning depending on the task to be performed. Supervised learning is one of the tasks, where a function is learned that maps an input to an output based on example input-output pairs. Support Vector Machines (SVMs) are a set of supervised learning techniques designed to solve discrimination and regression problems. They are one of the most popular predictors in Machine Learning. However in some cases, SVMs play no important role because they can't learn reliable parameters (hyperplanes) in non-linear kernel spaces under very sparse data (data where only a few variables for each input vector of predictor variables are non-zero). Thus, the best models are either direct applications of standard matrix / tensor factorization models or specialized models using factorized parameters. However, these models also have the disadvantage that they are not applicable to standard prediction data.

In this report, a new supervised machine learning model, called Factorization Machines (FMs) will be introduced. It's a new class of models that combines the advantages of SVMs and factorization models. FMs are especially popular in predicting the click-through rate (CTR) or for recommender systems. FMs are an impactful model and have shown excellent prediction capabilities. They have many advantages:

- FMs are a general predictor that can work with any real valued feature vector.
- FMs model all interactions between variables using factorized parameters.
- FMs are capable of estimating interactions in problems with a high degree of sparsity where SVMs fail.
- FMs have linear complexity, can be optimized in the primal and don't rely on support vectors like SVMs.

This report will introduce FMs, their propreties, comparison to other models and finally some important conclusions.

2.Factorization Machine

2.1. Factorization Machine Model

a) Model Equation:

The most common prediction task is to estimate a function $y : \mathbb{R}^n \to T$ from a real valued feature vector $x \in \mathbb{R}^n$ to a target domain T. Let $m(x)$ be the number of non-zero elements in the feature vector x and $\bar{m}_D$ be the average number of non-zero elements $m(x)$ of all vectors $x \in D$, where D=$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\}$ is the training dataset.

For a factorization machine of degree d = 2 (also called a 2-way FM), the model equation is defined as:

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

where the model parameters that have to be estimated are:

$$w_0 \in \mathbb{R}, w \in \mathbb{R}^n, V \in \mathbb{R}^{nxk} \qquad (2) \qquad \text{With} \quad < v_i, v_j > \; = \sum_{f=1}^{k} v_{i,f} \, v_{j,f} \qquad (3)$$

A row v_i within V describes the i-th variable with k factors, $k \in \mathbb{N}_0^+$ is a hyperparameter that defines the dimensionality of the factorization.

A 2-way FM captures all single and pairwise interactions between variables:

- w_0 is the global bias.
- w_i models the strength of the i-th variable.
- $\hat{w}_{i,j} =< v_i, v_j >$ models the interaction between the ith and j-th variable.

The FM models the interaction by factorizing it instead of using an own model parameter $w_{i,j} \in \mathbb{R}$ for each interaction.

b) Expressiveness :

We know that if a matrix W is positive definite , there exists a matrix V such that $W = V \cdot V^t$ provided that k is sufficiently large, under this condition, a FM can express any interaction matrix W. However in sparse settings, when there is not enough data to estimate complex interactions W, one should chose a small k. In this case, restricting k (and thus the expressiveness of the FM) improves interaction matrices under sparsity.

c) Parameter Estimation Under Sparsity:

In sparse settings, there is not enough data to estimate directly and independently the interactions between variables , but even in this case, FMs can estimate these interactions well by breaking the independence of the interaction parameters by factorizing them.

d) Computation

The complexity of straight forward computation of eq.(1) is in $O(kn^2)$ because all pairwise interactions have to be computed. By reformulating pairwise interactions, it drops to linear runtime as follows :

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \\
=&\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle \, x_i \, x_i \\
=&\frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{j,f} \, x_i \, x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} \, v_{i,f} \, x_i \, x_i \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right) \left(\sum_{j=1}^{n} v_{j,f} \, x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} \, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 \, x_i^2 \right)
\end{aligned}$$

This equation has only linear complexity, its computation is in $O(kn)$. In the case of sparsity, most of the elements in x are 0, so the sums have only to be computed over the non-zero elements. Thus, the computation of the factorization machine is in $O(k\bar{m}_D)$.

2.2 Factorization Machines as Predictors

FM can be applied to a variety of prediction tasks :

- Regression
- Binary classification
- Ranking

In order to prevent overfitting, regularization terms like $L2$ are usually added to optimization objective.

2.3 Learning Factorization Machines

As said before, FMs have a model equation with a linear complexity. Thus, the model parameters (w_0, w, V) can be learned efficiently by gradient descent methods for a variety of losses, among them are square, logit or hinge loss. The gradient of the FM model is calculated as follows:

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

$\sum_{j=1}^{n} v_{j,f} \, x_j$ can be precomputed since it's independent of i .

2.4 d-way Factorization Machine

We can easily generalize the 2-way FM to a d-way FM :

$$\begin{aligned}
&\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i \, x_i \\
&+ \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right) \qquad (5)
\end{aligned}$$

The straight-forward complexity for computing eq. (5) is $O(k_d n^d)$, but as done before for eq.(1), one can show that it can be computed in linear time.

2.5 FMS vs. SVMS

SVMs are one of the most popular models in machine learning, which use the kernel trick to find an optimal boundary between possible outputs. The optimal boundary, called hyperplane , is influenced by the support vectors which are data points close to the hyperplane. When using a polynomial kernel in SVMs, both FMs and SVMs model nested interactions between variables. But using SVMs, the weights given to those interactions 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, those weights are factorized, ie $w_{i,j}$ and $w_{i,k}$ depend on each other as they overlap and share parameters due to the factorization. Another advantage is that making a prediction using FMs with its 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 have some influence on the model equation.The two mentioned advantages of FMs over SVMs are particularly present when using very sparse data, as SVMs cannot learn reliable parameters in this case. Besides, using (nonlinear) SVMs, the data usually needs to be transformed to the dual form unlike FMs which can learn the model parameters without transforming the data to the dual form.

2.6 Other Types of Factorization Models vs. Factorization Machines

Several factorization models are often used for prediction tasks (particularly click-through rate (CTR) prediction and Recommender Systems). For instance, 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.

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 n groups (when having n categorical variables) and in each group exactly one value has to be 1 and the rest 0. Therefore, FMs are a general predictor. Specialized factorization models are designed for a single task. This results in many different published articles, and it's also showed that FMs can mimic many of the most successful factorization models just by feature extraction.

3. Conclusion

In this report, we introduced Factorization Machine, how it works and its different properties. Below are some of the conclusions that we have been able to draw about the different models we have studied:

- FMs model all possible interactions between values in the feature vector x using factorized interactions instead of full parametrized ones. This has 2 main advantages:
  - The interactions between values can be estimated even under high sparsity. Especially, it is possible to generalize to unobserved interactions.
  - The number of parameters as well as the time for prediction and learning is linear.
- The dense parametrization of SVMs requires direct observations for the interactions which is often not given in sparse settings.
- Parameters of FMs can be estimated well even under sparsity.
- FMs can be directly learned in the primal. Non-linear SVMs are usually learned in the dual.
- The model equation of FMs is independent of the training data. Prediction with SVMs depends on parts of the training data.
- Standard factorization models like PARAFAC or MF are not general prediction models like factorization machines. Instead they require that the feature vector is partitioned in m parts and that in each part exactly one element is 1 and the rest 0.
- FM can mimic other specialized models(such as SVD++, PITF, FPMC) well.

$\text{Factorization Machine is a must} - \text{have algorithm in any data scientist's pocket!}$

References:

- link
- link