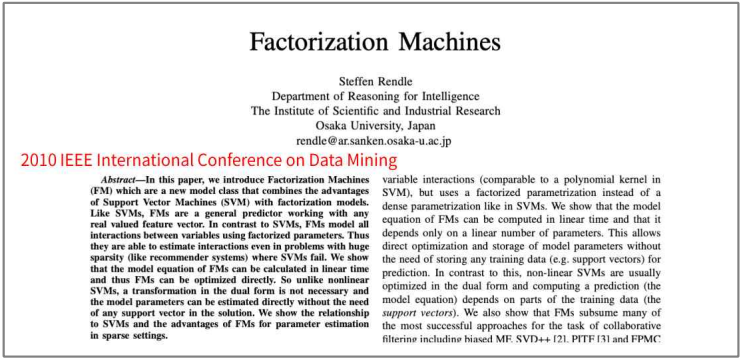
2021.05.17

Factorization Machines



Factorization Machines

Steffen Rendle
Department of Reasoning for Intelligence
The Institute of Scientific and Industrial Research
Osaka University, Japan
rendle@ar.sanken.osaka-u.ac.jp

2010 IEEE International Conference on Data Mining

***Abstract*—In this paper, we introduce Factorization Machines (FM) which are a new model class that combines the advantages of Support Vector Machines (SVM) with factorization models. Like SVMs, FMs are a general predictor working with any real valued feature vector. In contrast to SVMs, FMs model all interactions between variables using factorized parameters. Thus they are able to estimate interactions even in problems with huge sparsity (like recommender systems) where SVMs fail. We show that the model equation of FMs can be calculated in linear time and thus FMs can be optimized directly. So unlike nonlinear SVMs, a transformation in the dual form is not necessary and the model parameters can be estimated directly without the need of any support vector in the solution. We show the relationship to SVMs and the advantages of FMs for parameter estimation in sparse settings.**

variable interactions (comparable to a polynomial kernel in SVM), but uses a factorized parametrization instead of a dense parametrization like in SVMs. We show that the model equation of FMs can be computed in linear time and that it depends only on a linear number of parameters. This allows direct optimization and storage of model parameters without the need of storing any training data (e.g. support vectors) for prediction. In contrast to this, non-linear SVMs are usually optimized in the dual form and computing a prediction (the model equation) depends on parts of the training data (the *support vectors*). We also show that FMs subsume many of the most successful approaches for the task of collaborative filtering including biased MF, SVD++ [2], PITF [3] and FPMC

Abstract

Abstract—In this paper, we introduce Factorization Machines (FM) which are a new model class that combines the advantages of Support Vector Machines (SVM) with factorization models. Like SVMs, FMs are a general predictor working with any real valued feature vector. In contrast to SVMs, FMs model all interactions between variables using factorized parameters. Thus they are able to estimate interactions even in problems with huge sparsity (like recommender systems) where SVMs fail. We show that the model equation of FMs can be calculated in linear time and thus FMs can be optimized directly. So unlike nonlinear SVMs, a transformation in the dual form is not necessary and the model parameters can be estimated directly without the need of any support vector in the solution. We show the relationship to SVMs and the advantages of FMs for parameter estimation in sparse settings.

On the other hand there are many different factorization models like matrix factorization, parallel factor analysis or specialized models like SVD++, PITF or FPMC. The drawback of these models is that they are not applicable for general prediction tasks but work only with special input data. Furthermore their model equations and optimization algorithms are derived individually for each task. We show that FMs can mimic these models just by specifying the input data (i.e. the feature vectors). This makes FMs easily applicable even for users without expert knowledge in factorization models.

1. Factorizaion Machine은 SVM과 Factorization Model 의 장점을 합친 모델이다.
2. Real Value(실수) Feature Vector를 활용한 General Predictior이다.
3. FM의 방정식은 linear Time이다.
4. 일반적인 추천시스템은 special input data, optimization algorithm이 필요하지만, FM은 어느 곳에서든 쉽게 적용 가능하다.

I. Introduction

In total, the advantages of our proposed FM are:

1) FMs allow parameter estimation under very sparse data where SVMs fail.
2) FMs have linear complexity, can be optimized in the primal and do not rely on support vectors like SVMs. We show that FMs scale to large datasets like Netflix with 100 millions of training instances.
3) FMs are a general predictor that can work with any real valued feature vector. In contrast to this, other state-of-the-art factorization models work only on very restricted input data. We will show that just by defining the feature vectors of the input data, FMs can mimic state-of-the-art models like biased MF, SVD++, PITF or FPMC.

장점 요약

1. Sparse data
2. Linear Complexity
3. General Predictor로써 추천시스템 이외 다른 머신러닝에서 사용할 수 있다.

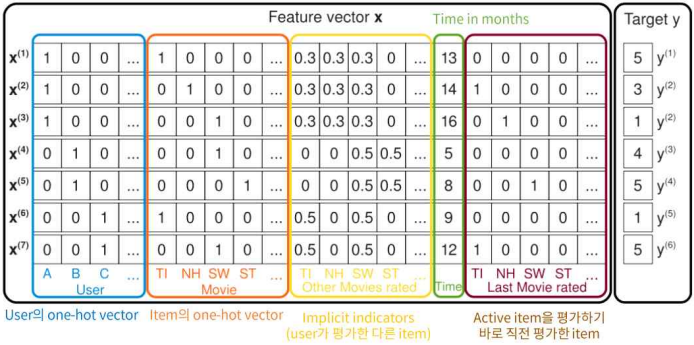
II. Prediction Under Sparsity

$$U = \{\text{Alice (A)}, \text{Bob (B)}, \text{Charlie (C)}, \ldots\}$$
$$I = \{\text{Titanic (TI)}, \text{Notting Hill (NH)}, \text{Star Wars (SW)}, \text{Star Trek (ST)}, \ldots\}$$

Let the observed data S be:

$$S = \{(\text{A}, \text{TI}, 2010\text{-}1, 5), (\text{A}, \text{NH}, 2010\text{-}2, 3), (\text{A}, \text{SW}, 2010\text{-}4, 1), (\text{B}, \text{SW}, 2009\text{-}5, 4), (\text{B}, \text{ST}, 2009\text{-}8, 5), (\text{C}, \text{TI}, 2009\text{-}9, 1), (\text{C}, \text{SW}, 2009\text{-}12, 5)\}$$

1. Matrix Factorization은 user, movie에 대한 rating만을 사용하는데, FM은 시간을 포함해 더 다양한 feature를 포함한다.
2. Feature engineering을 한다는 면에서 SVM과 비슷하다.
3. Sparse한 데이터도 이런식으로 펼쳐놓고보면 충분히 Prediction을 할 수 있다고 강조



A. Factorization Machine Model

1) ~ 2) equation

2-way FM (degree d=2) 방정식

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

w_0: Global Bias

$w_i x_i$: i번째 weight를 모델링한다

$$< v_i, v_j > := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

Size k의 두 벡터를 내적한다

{ 개별 변수 또는 변수 간 interaction모두 모델링함. }

$< v_i, v_j >$

{ Vi와 Vj는 결국 Xi와 Xj간 Interaction을 구할때 Latent Vector를 조합하는 것 -> Pairwise Feature Interaction을 고려할 수 있게 됨 }

{ Pairwise feature interaction을 고려하기 때문에 sparse한 상황에 적합 }

A. Factorization Machine Model

1) ~ 2) equation

```python
def predict(X, w0, w, v, n_factors, i):
    data = X.data
    indptr = X.indptr
    indices = X.indices
    """predicting a single instance"""
    summed = np.zeros(n_factors)
    summed_squared = np.zeros(n_factors)

    # linear output w * x
    pred = w0
    for index in range(indptr[i], indptr[i + 1]):
        feature = indices[index]
        pred += w[feature] * data[index]

    # factor output
    for factor in range(n_factors):
        for index in range(indptr[i], indptr[i + 1]):
            feature = indices[index]
            term = v[factor, feature] * data[index]
            summed[factor] += term
            summed_squared[factor] += term * term

        pred += 0.5 * (summed[factor] * summed[factor] - summed_squared[factor])

    # gradient update할 때, summed는 독립이므로 re-use 가능
    return pred, summed
```

각 feature에 대한 weight.

[0.2, 0.5, 0.7, 0.1]

Example

latent factors

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

(w_0: 실수값)

$$= \frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f} x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right)$$

① 예측을 위해 필요한 파라미터 ⇒ X, w_0, w, V, n_factors

X =

feature 1	feature2	feature3	feature4
1	0	1	0

V =

	feature1	feature2	feature3	feature4
1	0.1	0.2	0.3	0.4
2	0.5	0.6	0.7	0.8

n_factors를 2라고 예시해보자

② 마지막 식이 어떻게 계산되는지

factor 1에 대해서 $\left[(0.1 \times 1) + (0.3 \times 1) \right]^2 - \left[(0.1)^2(1)^2 + (0.3)^2(1)^2 \right]$

factor 2에 대해서 $\left[(0.5 \times 1) + (0.7 \times 1) \right]^2 - \left[(0.5)^2(1)^2 + (0.7)^2(1)^2 \right]$

factor 1 부분과 factor 2 부분의 합이 $\sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$ 부분이 된다.

A. Factorization Machine Model

3) Parameter Estimation Under Sparsity

Factorization machines can estimate interactions even in sparse settings well because they break the independence of the interaction parameters by factorizing them. In general this means that the data for one interaction helps also to estimate the parameters for related interactions.

Example

$$S = \{(\mathrm{A}, \mathrm{TI}, 2010\text{-}1, 5), (\mathrm{A}, \mathrm{NH}, 2010\text{-}2, 3), (\mathrm{A}, \mathrm{SW}, 2010\text{-}4, 1), (\mathrm{B}, \mathrm{SW}, 2009\text{-}5, 4), (\mathrm{B}, \mathrm{ST}, 2009\text{-}8, 5), (\mathrm{C}, \mathrm{TI}, 2009\text{-}9, 1), (\mathrm{C}, \mathrm{SW}, 2009\text{-}12, 5)\}$$

Rating Y를 예측하기 우해

Alice와 Star Track의 Interaction을 추정하고 싶다!

1. Bob과 Charlie는 둘다 star wars와 유사한 상호작용을 가지고 있기 때문에 유사한 factor vector인 Vb, Vc를 가진다.
   <Vb, Vsw>와 <Vc, Vsw> 는 서로 유사
2. Alice(Va)는 Titanic과 Star wars와 다른 상호작용을 가지므로 Charlie(Vc)와는 다른 factor vector이다. (Alice (vA) will have a different factor vector from Charlie (vC) because she has different interactions with the factors of Titanic and Star Wars for predicting ratings)
3. Startrek의 factor vector는 Star Wars와 유사할것이다. Bob이 두 영화에 대해 유사한 상호작용을 가지고 있기 때문이다.
4. 전체적으로 이것은 Alice와 Star Trek의 factor vector 내적이 Alice와 Star Wars의 내적과 유사하다는걸 의미한다.

A. Factorization Machine Model

4) Computation-어떻게 FM이 Linear Complexity를 갖는가

$O(kn^2)$

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle x_i x_j$$

$$=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle x_i x_j - \frac{1}{2}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_i\rangle x_i x_i$$

$$=\frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{j,f}\, x_i x_j - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{i,f}\, x_i x_i\right)$$

$$=\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)\left(\sum_{j=1}^{n} v_{j,f}\, x_j\right) - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right)$$

$$=\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right)$$

$O(kn)$

변수2개에 직접적으로 연관된 model parameter 가 없기때문에 pairwise interaction을 정리하면 Linear Complexity를 가질 수 있게된다.

III. Factorization Machine Model

B. Factorization Machines as Predictors

FM can be applied to a variety of prediction tasks. Among them are:

Least Square error활용

- **Regression**: $\hat{y}(\mathbf{x})$ can be used directly as the predictor and the optimization criterion is e.g. the minimal least square error on D.

0 또는 1(부호)을 예측

- **Binary classification**: the sign of $\hat{y}(\mathbf{x})$ is used and the parameters are optimized for hinge loss or logit loss.
- **Ranking**: the vectors $\mathbf{x}$ are ordered by the score of $\hat{y}(\mathbf{x})$ and optimization is done over pairs of instance vectors $(\mathbf{x}^{(\mathbf{a})}, \mathbf{x}^{(\mathbf{b})}) \in D$ with a pairwise classification loss (e.g. like in [5]).

점수로 order를 설정

FM은 Generalized Task를 해결할 수 있다.

C. Learning Factorization Machines

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

- FM의 Model parameters는 경사하강법에 의해 효과적으로 학습될 수 있다. - stochastic gradient descent(SGD)
  - Y햇을 미분했을때 세타값에 따라 다음 값을 가지며 이때 Vjf*Xj는 i에 독립적이어서 미리 계산될수 있다.
    - 그래서 time complexity가 더 떨어진다.

Conclusion

1. Factorized Interaction을 사용하여 feature vector x의 모든 가능한 interaction을 모델링한다.
2. 매우 sparse한 상황에서 interaction을 추정할 수 있다. Unobserved interaction에 대해서도 일반화할 수 있다.
3. 파라미터 수, 학습과 예측시간 모두 linear하다. Linear complexity
4. 이는 SGD를 활용한 최적화를 진행할 수 있고 다양한 loss funtion을 사용할 수 있다.
5. SVM, specialized model보다 더 나은 성능을 증명한다.