Locally Linear Factorization Machine

Anonymous Author(s)
Affiliation
email

Abstract

1 Introduction

2 Related work

3 Locally Linear Factorization Machine Model

Factorization machines (FMs) [Rendle, 2010; 2012] are a widely used method for efficiently using high-order feature interaction in classification and regression tasks even when the data is very high-dimensional. A standard 2-order factorization machine model takes the form:

$$f(\mathbf{x}) = \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f} v_{j',f}, \quad (1)$$

where p is the dimensionality of feature vector $\mathbf{x} \in \mathbb{R}^p$, $k \ll p$ is a hyper-parameter that denotes the dimensionality of latent factors, and $w_j, v_{j,f}$ are the model parameters to be estimated, i.e., $\Theta = \{w_1, \ldots, w_n, v_{1,1}, \ldots, v_{p,k}\} = \{\mathbf{w} \in \mathbb{R}^p, \mathbf{V} \in \mathbb{R}^{p \times k}\}$. It is equivalent to the following simple equation:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} (\mathbf{V}\mathbf{V}^\top)_{jj'} x_j x_{j'}, \quad (2)$$

The main advantage of FMs compared to the polynomial kernel in SVM [Vapnik, 2013] is the pairwise feature interaction weight matrix $\mathbf{Z} = \mathbf{V}\mathbf{V}^\top \in \mathbb{S}^{p \times p}$, where the number of parameters to estimate is reduced from p^2 to kp by utilizing the factorized form. In addition, this factorization form helps to drop the prediction cost to linear runtime by utilizing

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \frac{1}{2} \left(\|\mathbf{V}^\top \mathbf{x}\|^2 - \sum_{f=1}^{k} \|\mathbf{v}_f \circ \mathbf{x}\|^2 \right), \quad (3)$$

where $\mathbf{v}_f \in \mathbb{R}^p$ is the f^{th} column of $\mathbf{V}$ and $\circ$ denotes the element-wise product between two vectors by $\mathbf{v}_f \circ \mathbf{x} = [v_{f_1} x_1, \ldots, v_{fp} x_p]^\top$. Thus, the computation cost is in $O(kp)$ instead of $O(p^2)$. Moreover, under sparsity condition, the prediction cost reduces to $O(kN_z(\mathbf{x}))$, where $N_z(\mathbf{x})$ is the number of non-zero features in $\mathbf{x}$. Given a training set consisting of n feature vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times p}$ and corresponding targets $\mathbf{Y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$, model parameters Θ can be learned by using the principle of empirical risk minimization and solving the following non-convex problem

$$\min_{\mathbf{w} \in \mathbb{R}^p, \mathbf{V} \in \mathbb{R}^{p \times k}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x})) + \frac{\beta_1}{2} \|\mathbf{w}\|^2 + \frac{\beta_2}{2} \|\mathbf{V}\|_F^2, \quad (4)$$

where $\beta_1 > 0, \beta_2 > 0$ are hyper-parameters that avoid overfitting and ℓ is a convex loss function incurred. For regression task, we can adopt the squared loss $\ell(y, \hat{y}) = \frac{1}{2}(y - f(\mathbf{x}))^2$. This optimization problem can be efficiently solved by many off-the-shelf approaches, such as stochastic gradient descent or coordinate descent methods which has been implemented in the libfm library [Rendle, 2012]. Both methods have a runtime complexity of $O(kN_z(\mathbf{x}))$ under sparsity.

Factorization Machines haven been found successful in many prediction tasks, including classification, regression and ranking, due to its capability to model the pairwise feature interaction under low-rank constraint. However, Factorization Machines only consider the second order information of the input feature which limits their capacity in non-linear problems. Specifically, taking classification task into consideration, not all problems are approximately linearly separable after quadratic mapping. In most cases, real data naturally groups into clusters and lies on nearly disjoint lower dimensional manifolds and thus the original Factorization Machines are inapplicable. Although they have the capability of modeling the pairwise feature interaction, Factorization Machines fail to capture the underlying structures of complex data. One intuitive idea for addressing this limitation is to leverage the manifold geometric structure to learn a nonlinear function which can be effectively approximated by a linear function with an coding under appropriate localization conditions. In other words, we assume that in a sufficiently small region the decision boundary is approximately linear and each data point $\mathbf{x}$ can then be approximated with a linear combination of surrounding anchor points, which are usually called local codings scheme.

Specifically, let $\{\mathbf{z}_j\}_{i=1}^m$ denote the set of m anchor points, any point $\mathbf{x}$ is then approximated by a linear combination

of surrounding anchor points as $\mathbf{x} \approx \sum_{i=1}^{m} \gamma_{\mathbf{z}_i}(\mathbf{x})\mathbf{z}_i$, where $\gamma_{\mathbf{z}_i}(\mathbf{x})$ (to be later defined) is the local coordinates, depicting the degree of membership of $\mathbf{x}$ to the ith anchor point $\mathbf{z}_i$, constrained by $\sum_{i=1}^{m} \gamma_{\mathbf{z}_i}(\mathbf{x}) = 1$.

To encode this local linearity, the model parameters Θ of the Factorization Machines should vary according to the location of the point $\mathbf{x}$ in the feature space as:

$$f_{LLFM}(\mathbf{x}) = \mathbf{w}(\mathbf{x})^\top \mathbf{x} + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} (\mathbf{V}(\mathbf{x})\mathbf{V}(\mathbf{x})^\top)_{jj'} x_j x_{j'}. \tag{5}$$

According to [Yu *et al.*, 2009; Ladicky and Torr, 2011], for a local coding scheme defined by anchor points $\mathbf{v}$, we can approximate the weight function $\mathbf{w}(\mathbf{x}), \mathbf{V}(\mathbf{x})$ using local coding as:

$$\begin{aligned} & f_{LLFM}(\mathbf{x}) \\ &= \sum_{i=1}^{m} \gamma_{\mathbf{z}_i}(\mathbf{x}) \mathbf{w}_{\mathbf{z}_i}^\top \mathbf{x} + \sum_{i=1}^{m} \gamma_{\mathbf{z}_i}(\mathbf{x}) \sum_{j=1}^{p} \sum_{j'=j+1}^{p} (\mathbf{V}_{\mathbf{z}_i} \mathbf{V}_{\mathbf{z}_i}^\top)_{jj'} x_j x_{j'} \\ &= \sum_{i=1}^{m} \gamma_{\mathbf{z}_i}(\mathbf{x}) \left(\mathbf{w}_{\mathbf{z}_i}^\top \mathbf{x} + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} (\mathbf{V}_{\mathbf{z}_i} \mathbf{V}_{\mathbf{z}_i}^\top)_{jj'} x_j x_{j'} \right), \end{aligned} \tag{6}$$

where $\Theta_{LLFM} = \{\Theta_{\mathbf{z}_i}\}_{i=1}^{m} = \{\mathbf{w}_{\mathbf{z}_i}, \mathbf{V}_{\mathbf{z}_i}\}_{i=1}^{m}$ are the model parameters corresponding to the anchor point $\mathbf{z}_i$. This transformation can be seen as a finite kernel transforming a p-dimensional problem into a mp-dimensional one. It can also be interpreted as defining a locally linear Factorization Machines as the weighted average of m separate Factorization Machines with respect to each anchor point, where the weights are determined by the local coordinates $\gamma_{\mathbf{z}}(\mathbf{x})$.

To evaluate $f_{LLFM}(\mathbf{x})$ for each data point $\mathbf{x}$, we need to calculate the corresponding local coordinates $\gamma_{\mathbf{z}}(\mathbf{x})$, which further depend on the anchor points $\mathbf{z}$ being used and the local coding scheme. Note that the prediction function in Eq.(6) depends on both the model parameters $\mathbf{w}, \mathbf{V}$, the anchor point variable $\mathbf{z}$ and the local coding scheme. This leads to a natural two-step approach taken by existing methods [Ladicky and Torr, 2011; Yu *et al.*, 2009], which first estimate the anchor points for each data by adopting K-means clustering and evaluate the local codes with a predefined scheme, which usually utilizes exponential decay scheme or inversely-proportional decay scheme, and then feed the anchor points and the local codes to the downstream supervised model training. This two-step learning procedure is inconsistent with our objective and rather suboptimal as the prediction information is not used in discovering the anchor points and the local codes, which is clearly not an optimal local coding scheme for prediction task as the two methods are not tightly coupled to fully exploit their potential.

4 Optimization method for Joint learning Locally Linear Factorization Machines

In this section we first discuss assumptions, then formulate the optimal estimation problem. Finally, we present our optimization approach for joint learning Locally Linear Factorization Machines.

5 Experiments

In this section, to verify the effectiveness of our proposed Adaptive Locally Linear Factorization Machines algorithm, we will compare 3 different adaptive algorithm to the original Factorization Machines algorithm on real-world datasets.

5.1 Experimental Setup

Datasets

We conduct our experiments on 2 real-wrold datasets, including Banana and IJCNN. The statistics of the datasets after preprocesing are shown in Table 1.

Evaluation Metrics

To illustrate the recommendation prediction quality of different methods, we use RMSE metric to evaluate the overall error.

Methods

In our experiments, we compare our 3 different locally linear Factorization Machines to baseline method (FM).

- **LLFM**:FM with several fixed anchor points which initialized by K-means.

Algorithm 1 LLFM

Input: Training data S, the number of anchor points m, the number of nearest neighbors k, regularization parameter λ, learning rate η, parameter in Gaussian kernel β.
Output: Model parameters $\Theta = (\boldsymbol{W}, \boldsymbol{V})$ and anchor points.

```
Initialize anchor points v by K-means;Initialize Θ:W ← (0, ..., 0), V ~ N(0, 0.1)
repeat
    for (x, y) ∈ S do
        Compute the local coordinates γ_z(x)
        Compute ŷ(x|Θ)
        Update W
        Update V
    end for
until stopping criterion is met;
```

- **LLFMAAP**:LLFM with Adaptive Anchor Points, update anchor points during learning phase. The anchor points are also initialized by K-means.

Dataset	#Training	#Test	#feature	#class
Banana	3533	1767	2	2
IJCNN	49990	91701	22	2
Movielens 100k	90570	9430	2625(943 + 1682)	/
Netflix5K5K	152862	152862	10000(5000 + 5000)	/

Table 1: Basic statistics of datasets

Banana	Train loss	Test loss	Acc(%)	test time
FM	0.6792±0.0005	0.6763±0.004	53.77 ± 0.19	×1
LLFM-DO	0.2576±0.0018	0.2364±0.0020	89.87 ± 0.20	×12.06
LLFM-APL	0.2493±0.0018	0.2283±0.0047	89.96 ± 0.23	×11.95
LLFM-JO	**0.2235±0.0006**	**0.2170±0.0012**	**90.44 ± 0.29**	×11.05
IJCNN	Train loss	Test loss	Acc(%)	test time
FM	0.0911±0.0004	0.0927±0.0038	96.97 ± 0.41	×1
LLFM-DO	0.0797±0.0007	0.0708±0.0118	97.72 ± 0.21	×21.56
LLFM-APL	0.0799±0.0019	0.0525±0.0038	98.26 ± 0.12	×22.04
LLFM-JO	**0.0550±0.0015**	**0.0404±0.0014**	**98.73 ± 0.06**	×22.92
Movielens 100k	Train loss	Test loss	Acc(%)	test time
FM	0.8059±0.0012	0.9498±0.0044	90.50 ± 0.00	×1
LLFM	0.6801±0.0024	0.9464±0.0027	96.18 ± 0.25	×15.52
LLFM-APL	0.8630±0.0100	0.9384±0.0023	96.97 ± 0.68	×15.78
LLFM-JO	**0.8758±0.0153**	**0.9331±0.0016**	**97.12 ± 0.51**	×16.30
Netflix5K5K	Train loss	Test loss	Acc(%)	test time
FM	0.0409±0.0001	0.0415±0.0000	98.15 ± 0.02	×1
LLFM	0.0319±0.0002	0.0275±0.0004	98.82 ± 0.04	×7.08
LLFM-APL	0.0268±0.0002	0.0201±0.0003	99.03 ± 0.03	×7.17
LLFM-JO	**0.0258±0.0007**	**0.0194±0.0008**	**99.08 ± 0.02**	×7.28

Table 2: Comparison of different algorithms in terms of train loss, test loss, classification accuracy and test time (normalized to test time of FM)

Algorithm 2 LLFMAAP

Input: Training data S, the number of anchor points m, the number of nearest neighbors k, regularization parameter λ, learning rate η, parameter in Gaussian kernel β.
Output: Model parameters $\Theta = (\boldsymbol{W}, \boldsymbol{V})$ and anchor points.

Initialize anchor points v by K-means;Initialize Θ:$\boldsymbol{W} \leftarrow (0, ..., 0), \boldsymbol{V} \sim N(0, 0.1)$
repeat
  for $(\boldsymbol{x}, y) \in S$ **do**
    Compute the local coordinates $\gamma_{\boldsymbol{z}}(\boldsymbol{x})$
    Compute $\hat{y}(\boldsymbol{x}|\Theta)$
    Update $\boldsymbol{W}$
    Update $\boldsymbol{V}$
    for $j = 1$ **to** k **do**
      Update the jth nearest anchor point of $\boldsymbol{x}$.
    end for
  end for
until stopping criterion is met;

- **LLFMAAPK**:LLFMAAP with Adaptive KNN, choose the optimal number of nearest neighbors of KNN for each sample. Also using adaptive anchor points like the former algorithm.

Algorithm 3 LLFMAAPK

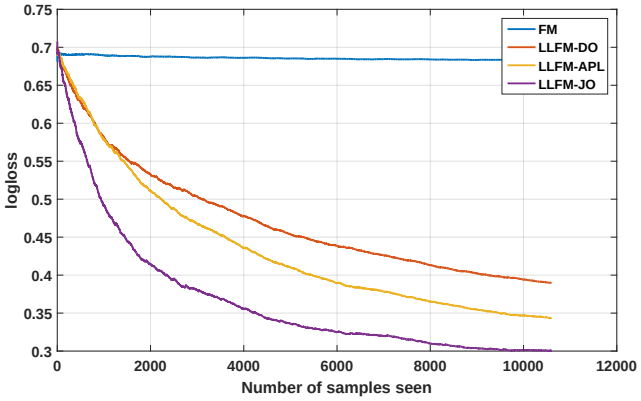
Input: Training data S, the number of anchor points m, regularization parameter λ, learning rate η, Lipschitz to noise ratio L/C.
Output: Model parameters $\Theta = (\boldsymbol{W}, \boldsymbol{V})$ and anchor points.

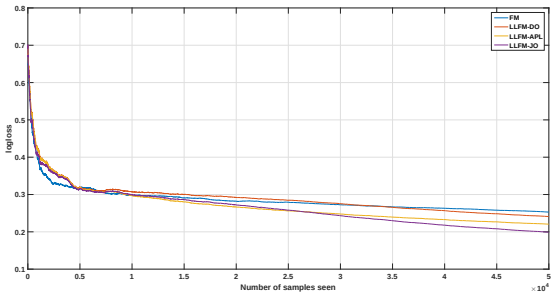
Initialize anchor points v by K-means;Initialize Θ:$\boldsymbol{W} \leftarrow (0, ..., 0), \boldsymbol{V} \sim N(0, 0.1)$
repeat
  for $(\boldsymbol{x}, y) \in S$ **do**
    Compute the local coordinates $\gamma_{\boldsymbol{z}}(\boldsymbol{x})$ and the number of nearest neighbor $k_{\boldsymbol{z}}(\boldsymbol{x})$
    Compute $\hat{y}(\boldsymbol{x}|\Theta)$
    Update $\boldsymbol{W}$
    Update $\boldsymbol{V}$
    for $j = 1$ **to** $k_{\boldsymbol{z}}(\boldsymbol{x})$ **do**
      Update the jth nearest anchor point of $\boldsymbol{x}$.
    end for
  end for
until stopping criterion is met;

Hyper-parameter Settings

We apply 5-fold cross validation to find the hyper-parameters for the four different algorithm independently. For parameter settings, we adopt the same parameter tuning schemes for all the compared algorithm to enable fair comparisons. We perform grid search to choose the best parameters for each algorithm on the training set.

- **Learning rate** η: We do grid search in $\{0.001, 0.005, 0.01, 0.05, 0.1\}$.
- **Latent dimension** d:For comparison purposes,we set a fixed value($d = 10$ in our experiment)for all methods based on factorization models.
- **Regularlization** λ:We do grid search in $\{0, 0.001, 0.01, 0.1, 1\}$.
- **parameter in Gaussian kernel** β:We do grid search in $\{0.1, 1, 10\}$.
- **Lipschitz to noise ratio** L/C:We do grid search in $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$.





6 Conclusion

References

[Ladicky and Torr, 2011] Lubor Ladicky and Philip Torr. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 985–992, 2011.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[Rendle, 2012] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[Vapnik, 2013] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[Yu *et al.*, 2009] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *Advances in neural information processing systems*, pages 2223–2231, 2009.