Learning Piece-wise Linear Models from Large Scale Data for Ad Click Prediction

Kun Gai[1], Xiaoqiang Zhu[1], Han Li[1], Kai Liu[2†], Zhe Wang[3†]

[1] Alibaba Inc.

jingshi.gk@taobao.com, {xiaoqiang.zxq, lihan.lh}@alibaba-inc.com

[2] Beijing Particle Inc.

liukai@yidian-inc.com

[3] University of Cambridge.

zw267@cam.au.uk

[†] contribute to this paper while worked at Alibaba

April 19, 2017

Abstract

CTR prediction in real-world business is a difficult machine learning problem with large scale nonlinear sparse data. In this paper, we introduce an industrial strength solution with model named Large Scale Piece-wise Linear Model (LS-PLM). We formulate the learning problem with L_1 and $L_{2,1}$ regularizers, leading to a non-convex and non-smooth optimization problem. Then, we propose a novel algorithm to solve it efficiently, based on directional derivatives and quasi-Newton method. In addition, we design a distributed system which can run on hundreds of machines parallel and provides us with the industrial scalability. LS-PLM model can capture nonlinear patterns from massive sparse data, saving us from heavy feature engineering jobs. Since 2012, LS-PLM has become the main CTR prediction model in Alibaba's online display advertising system, serving hundreds of millions users every day.

1 Introduction

Click-through rate (CTR) prediction is a core problem in the multi-billion dollar online advertising industry. To improve the accuracy of CTR prediction, more and more data are involved, making CTR prediction a large scale learning problem, with massive samples and high dimension features.

Traditional solution is to apply a linear logistic regression (LR) model, trained in a parallel manner (Brendan et al. 2013, Andrew & Gao 2007). LR model with L_1 regularization can generate sparse solution, making it fast for online prediction. Unfortunately, CTR prediction problem is a highly nonlinear problem. In particular, user-click generation involves many complex factors, like ad quality, context information, user interests, and complex interactions of these factors. To help LR model catch the nonlinearity, feature engineering technique is explored, which is both time and humanity consuming.

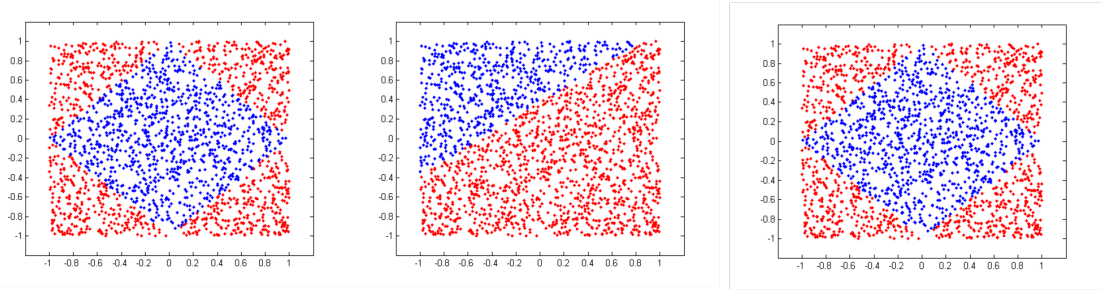
Another direction, is to capture the nonlinearity with well-designed models. Facebook (He et al. 2014) uses a hybrid model which combines decision trees with logistic regression. Decision tree plays a nonlinear feature transformation role, whose output is fed to LR model. However, tree-based method is not suitable for very sparse and high dimensional data (Safavian S. R. & Landgrebe D. 1990). (Rendle S. 2010) introduces Factorization Machines(FM), which involves interactions among features using 2nd order functions (or using other given-number-order functions). However, FM can not fit all general nonlinear patterns in data (like other higher order patterns).

In this paper, we present a piece-wise linear model and its training algorithm for large scale data. We name it Large Scale Piecewise Linear Model (LS-PLM). LS-PLM follows the divide-and-conquer strategy, that is, first divides the feature space into several local regions, then fits a linear model in each region, resulting in the output with combinations of weighted linear predictions. Note that, these two steps are learned simultaneously in a supervised manner, aiming to minimize the prediction loss. LS-PLM is superior for web-scale data mining in three aspects:

- **Nonlinearity**. With enough divided regions, LS-PLM can fit any complex nonlinear function.
- **Scalability**. Similar to LR model, LS-PLM is scalable both to massive samples and high dimensional features. We design a distributed system which can train the model on hundreds of machines parallel. In our online product systems, dozens of LS-PLM models with tens of million parameters are trained and deployed everyday.
- **Sparsity**. As pointed in (Brendan et al. 2013), model sparsity is a practical issue for online serving in industrial setting. We show LS-PLM with L_1 and $L_{2,1}$ regularizer can achieve good sparsity.

The learning of LS-PLM with sparsity regularizer can be transformed to be a non-convex and non-differential optimization problem, which is difficult to be solved. We propose an efficient optimization method for such problems, based on directional derivatives and quasi-Newton method. Due to the ability of capturing nonlinear patterns and scalability to massive data, LS-PLMs have become main CTR prediction models in the online display advertising system in alibaba, serving hundreds of millions users since 2012 early. It is also applied in recommendation systems, search engines and other product systems.

The paper is structured as follows. In Section 2, we present LS-PLM model in detail, including formulation, regularization and optimization issues. In Section 3 we introduce our parallel implementation structure. in Section 4, we evaluate the model carefully and demonstrate the advantage of LS-PLM compared with LR. Finally in Section 5, we close with conclusions.



A) Training dataset B) LR model C) LS-PLM model

Figure 1: A demo illustration of LS-PLM model. Figure A is the demo dataset. It is a binary classification problem, with red points belong to positive class and blue points belong to negative class. Figure B shows the classification result using LR model. Figure C shows the classification result using LS-PLM model. It's clear that LS-PLM can capture the nonlinear distribution of data.

2 Method

We focus on the large scale CTR prediction application. It is a binary classification problems, with dataset $\{x_t, y_t\}|_{t=1}^{n}.\ y_t \in \{0, 1\}$ and $x_t \in R^d$ is usually high dimensional and sparse.

2.1 Formulation

To model the nonlinearity of massive scale data, we employ a divide-and-conquer strategy, similar with (Jordan & Jacobs 1994). We divide the whole feature space into some local regions. For each region we

employ an individual generalized linear-classification model. In this way, we tackle the nonlinearity with a piece-wise linear model. We give our model as follows:

$$p(y=1|x) = g\Big(\sum_{j=1}^{m} \sigma(u_j^T x)\eta(w_j^T x)\Big) \tag{1}$$

Here $\Theta = \{u_1, \cdots, u_m, w_1, \cdots, w_m\} \in \mathbb{R}^{d\times 2m}$ denote the model parameters. $\{u_1, \cdots, u_m\}$ is the parameters for dividing function $\sigma(\cdot)$, and $\{w_1, \cdots, w_m\}$ for fitting function $\eta(\cdot)$. Given instance x, our predicating model $p(y|x)$ consists of two parts: the first part $\sigma(u_j^T x)$ divides feature space into m (hyper-parameter) different regions, the second part $\eta(w_j^T x)$ gives prediction in each region. Function $g(\cdot)$ ensures our model to satisfy the definition of probability function.

Special Case. Take softmax (Kivinen & Warmuth 1998) as dividing function $\sigma(x)$ and sigmoid (Hilbe 2009) as fitting function $\eta(x)$ and $g(x) = x$, we get a specific formulation:

$$p(y=1|x) = \sum_{i=1}^{m} \frac{\exp(u_i^T x)}{\sum_{j=1}^{m}\exp(u_j^T x)} \cdot \frac{1}{1+\exp\left(-w_i^T x\right)} \tag{2}$$

In this case, our mixture model can be seen as a FOE model (Jordan & Jacobs 1994, Wang & Puterman 1998) as follows:

$$p(y=1|x) = \sum_{i=1}^{m} p(z=i|x)p(y|z=i,x) \tag{3}$$

Eq.(2) is the most common used formulation in our real applications. In the reminder of the paper, without special declaration, we take Eq.(2) as our prediction model. Figure 1 illustrates the model compared with LR in a demo dataset, which shows clearly LS-PLM can capture the nonlinear pattern of data.

The objective function of LS-PLM model is formalized as Eq.(4):

$$arg\min_{\Theta} f(\Theta) = \text{loss}(\Theta) + \lambda\|\Theta\|_{2,1} + \beta\|\Theta\|_1 \tag{4}$$

$$\text{loss}(\Theta) = -\sum_{t=1}^{n}\Big[y_t \log\left(p(y_t=1|x_t,\Theta)\right) + (1-y_t)\log(p(y_t=0|x_t,\Theta))\Big] \tag{5}$$

Here loss(Θ) defined in Eq.(5) is the neg-likelihood loss function and $\|\Theta_{2,1}\|$ and $\|\Theta_1\|$ are two regularization terms providing different properties. First, $L_{2,1}$ regularization ($\|\Theta\|_{2,1} = \sum_{i=1}^{d}\sqrt{\sum_{j=1}^{2m}\theta_{ij}^2}$) is employed for feature selection. As in our model, each dimension of feature is associated with $2m$ parameters. $L_{2,1}$ regularization are expected to push all the $2m$ parameters of one dimension of feature to be zero, that is, to suppress those less important features. Second, L_1 regularization ($\|\Theta\|_1 = \sum_{ij}|\theta_{ij}|$) is employed for sparsity. Except with the feature selection property, L_1 regularization can also force those parameters of left features to be zero as much as possible, which can help improve the interpretation ability as well as generalization performance of the model.

However, both L_1 norm and $L_{2,1}$ norm are non-smooth functions. This causes the objective function of Eq.(4) to be non-convex and non-smooth, making it difficult to employ those traditional gradient-descent optimization methods (Andrew & Gao 2007, Zhang 2004, Bertsekas 2003) or EM method (Wang & Puterman 1998).

Note that, while (Wang & Puterman 1998) gives the same mixture model formulation as Eq.(3), our model is more general for employing different kinds of prediction functions. Besides, we propose a different objective function for large scale industry data, taking the feature sparsity into consideration explicitly. This is crucial for real-world applications, as prediction speed and memory usage are two key indicators for online model serving. Furthermore, we give a more efficient optimization method to solve the large-scale non-convex problem, which is described in the following section.

2.2 Optimization

Before we present our optimization method, we establish some notations and definitions that will be used in the reminder of the paper. Let $\partial_{ij}^{+} f(\Theta)$ denote the right partial derivative of f at Θ with respect to Θ_{ij}:

$$\partial_{ij}^{+} f(\Theta) = \lim_{\alpha \downarrow 0} \frac{f(\Theta + \alpha e_{ij}) - f(\Theta)}{\alpha} \tag{6}$$

where e_{ij} is the ij^{th} standard basis vector. The directional derivative of f as Θ in direction d is denoted as $f'(\Theta; d)$, which is defined as:

$$f'(\Theta; d) = \lim_{\alpha \downarrow 0} \frac{f(\Theta + \alpha d) - f(\Theta)}{\alpha} \tag{7}$$

A vector d is regarded as a descent direction if $f'(\Theta; d) < 0$. $\text{sign}(\cdot)$ is the sign function takes value $\{-1, 0, 1\}$. The projection function

$$\pi_{ij}(\Theta; \Omega) = \begin{cases} \Theta_{ij} & , \quad \text{sign}(\Theta_{ij}) = \text{sign}(\Omega_{ij}) \\ 0 & , \quad \text{otherwise} \end{cases} \tag{8}$$

means projecting Θ onto the orthant defined by Ω.

2.2.1 Choose descent direction

As discussed above, our objective function for large scale CTR prediction problem is both non-convex and non-smooth. Here we propose a general and efficient optimization method to solve such kind of non-convex problems. Since the negative-gradients of our objective function do not exists for all Θ, we take the direction d which minimizes the directional derivative of f with Θ as a replace. The directional derivative $f'(\Theta; d)$ exists for any Θ and direction d, whic is declared as Lemma 1.

Lemma 1. *When an objective function $f(\Theta)$ is composed by a smooth loss function with L_1 and $L_{2,1}$ norm, for example the objective function given in Eq. (4), the directional derivative $f'(\Theta; d)$ exists for any Θ and direction d.*

We leave the proof in Appendix A. Since the directional derivative $f'(\Theta; d)$ always exists, we choose the direction as the descent direction which minimizes the directional derivative $f'(\Theta; d)$ when the negative gradient of $f(\Theta)$ does not exist. The following proposition 2 explicitly gives the direction.

Proposition 2. *Given a smooth loss function $loss(\Theta)$ and an objective function $f(\Theta) = loss(\Theta) + \lambda \|\Theta\|_{2,1} + \beta \|\Theta\|_1$, the bounded direction d which minimizes the directional derivative $f'(\Theta; d)$ is denoted as follows:*

$$d_{ij} = \begin{cases} s - \beta sign(\Theta_{ij}), & \Theta_{ij} \neq 0 \\ \max\{|s| - \beta, 0\} sign(s), & \Theta_{ij} = 0, \|\Theta_{i\cdot}\|_{2,1} \neq 0 \\ \frac{\max\{\|v\|_{2,1} - \lambda, 0\}}{\|v\|_{2,1}} v, & \|\Theta_{i\cdot}\|_{2,1} = 0, \end{cases} \tag{9}$$

where $s = -\nabla loss(\Theta)_{ij} - \lambda \frac{\Theta_{ij}}{\|\Theta_{i\cdot}\|_{2,1}}$ and $v = \max\{|-\nabla loss(\Theta)_{ij}| - \beta, 0\} sign(-\nabla loss(\Theta)_{ij})$.

More details about the proof can be found in Appendix B. According to the proof, we can see that the negative pseudo-gradient defined in Gao's work (Andrew & Gao 2007) is a special case of our descent direction. Our proposed method is more general to find the descent direction for those non-smooth and non-convex objective functions.

Based on the direction d^k in Eq.(9), we update the model parameters along a descent direction calculated by limited-memory quasi-newton method (LBFGS) (Wang & Puterman 1998), which approximates the inverse Hessian matrix of Eq.(4) on the given orthant. Motivated by the OWLQN method (Andrew &

Algorithm 1 Optimize problem Eq.(4)

Input:Choose initial point $\Theta^{(0)}$
$S \leftarrow \{\}, Y \leftarrow \{\}$
for $k = 0$ to **MaxIters** do

1. Compute $d^{(k)}$ with Eq. (9).
2. Compute p_k with Eq. (11) using S and Y.
3. Find $\Theta^{(k+1)}$ with constrained line search (12).
4. If termination condition satisfied then
   stop and return $\Theta^{(k+1)}$
   End if
5. Update S with $s^{(k)} = \Theta^{(k)} - \Theta^{(k-1)}$
6. Update Y with $y^{(k)} = -d^{(k)} - (-d^{(k-1)})$

end for

Gao 2007), we also restrict the signs of model parameters not changing in each iteration. Given the chosen direction d^k and the old $\Theta^{(k)}$, we constrain the orthant of current iteration as follows:

$$\xi_{ij}^{(k)} = \begin{cases} \text{sign}(\Theta_{ij}^{(k)}), & \Theta_{ij}^{(k)} \neq 0 \\ \text{sign}(d_{ij}^{(k)}), & \Theta_{ij}^{(k)} = 0 \end{cases} . \tag{10}$$

When $\Theta_{ij}^{(k)} \neq 0$, the new Θ_{ij} would not change sign in current iteration. When $\Theta_{ij}^{(k)} = 0$, we choose the orthant decided by the selected direction $d_{ij}^{(k)}$ for the new $\Theta_{ij}^{(k)}$.

2.2.2 Update direction constraint and line search

Given the descent direction d^k, we approximate the inverse-Hessian matrix H_k using LBFGS method with a sequence of $y^{(k)}, s^{(k)}$. Then the final update direction is $H_k d^{(k)}$. Here we give two tricks to adjust the update direction. First, we constrain the update direction in the orthant with respect to $d^{(k)}$. Second, as our objective function is non-convex, we cannot guarantee that H_k is positive-definite. We use $y^{(k)T} s^{(k)} > 0$ as a condition to ensure H_k is a positive-definite matrix. If $y^{(k)T} s^{(k)} \leq 0$, we switch to $d^{(k)}$ as the update direction. The final update direction p_k is defined as follows:

$$p_k = \begin{cases} \pi(H_k d^{(k)}; d^{(k)}), & y^{(k)T} s^{(k)} > 0 \\ d^{(k)}, & \text{otherwise} \end{cases} \tag{11}$$

Given the update direction, we use backtracking line search to find the proper step size α. Same as OWLQN, we project the new $\Theta^{(k+1)}$ onto the given orthant decided by the Eq. (10).

$$\Theta^{(k+1)} = \pi(\Theta^{(k)} + \alpha p_k; \xi^{(k)}) \tag{12}$$

2.3 Algorithm

A pseudo-code description of optimization is given in Algorithm 1. In fact, only a few steps of the standard LBFGS algorithm need to change. These modifications are:

1. The direction $d^{(k)}$ which minimizes the direction derivative of the non-convex objective is used in replace of negative gradient.

2. The update direction is constrained to the given orthant defined by the chosen direction $d^{(k)}$. Switch to $d^{(k)}$ when the H_k is not positive-definite.

3. During the line search, each search point is projected onto the orthant of the previous point.

3 Implementation

In this section, we first provide our parallel implementation of LS-PLM model for large-scale data, then introduce an important trick which helps to accelerate the training procedue greatly.

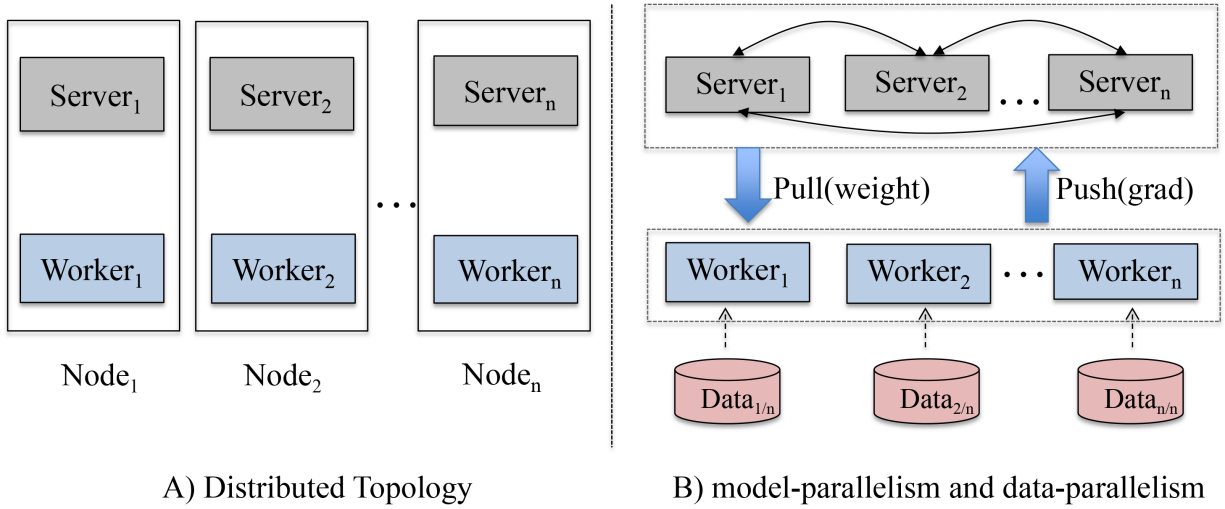


Figure 2: The architecture of parallel implementation. Figure A illustrates the physical distributed topology. It's a variant of parameter server, where each computation node runs with both a server and a worker, aiming to maximize the utility of computation power as well as memory usuage. Figure B illustrates the parameter server structure in model-parallelism and data-parallelism manner.

3.1 Parallel implementation

To apply Algorithm 1 in large-scale settings, we implement it with a distributed learning framework, as illustrated in Figure 2. It's a variant of parameter server. In our implementation, each computation node runs with both a server node and a worker node, aiming to

- Maximize the utility of CPU's computation power. In traditional parameter server setting, server nodes work as a distributed KV storer with interfaces of push and pull operations, which are low computation costly. Running with worker nodes can make full use of the computation power.

- Maximize the utility of memory. Machines today usually have big memory, for example 128GB. Running on the same computation node, server node and worker node can share and utilize the big memory better.

In brief, there are two roles in the framework. The first role is the work node. Each node stores a part of training data and a local model, which only saves the model parameters used for the local training data. The second role is the server node. Each node stores a part of global model, which is mutually-exclusive. In each iteration, all of the worker nodes first calculate the loss and the descent direction with local model and local data in parallel(data parallelism). Then server nodes aggregate the loss and the direction $d^{(k)}$ as

well as the corresponding entries of Θ needed to calculate the revised gradient (model parallelism). After finishing calculating the steepest descent direction in Step 1, workers synchronize the corresponding entries of Θ, and then, perform Step 2–6 locally.

3.2 Common Feature Trick

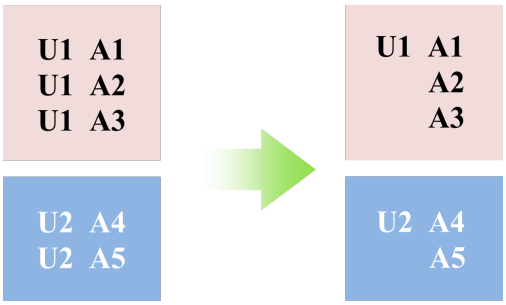


Figure 3: Common feature pattern in display advertising. Usually in each page view, a user will see several different ads at the same time. In this situation, user features can be shared across these samples.

In addition to the general-purpose parallel implementation, we also optimized the implementation in online advertising context. Training samples in CTR prediction tasks usually have similar common feature pattern. Take display advertising as an example, as illustrated in Figure 3, during each page view, a user will see several different ads at the same time. For example, user **U1** in Figure 3 sees three ads in one visit session, thus generates three samples. In this situation, features of user **U1** can be shared across these three samples. These features include user profiles (sex, age, etc.) and user behavior histories during visits of Alibabas e-commerce websites, for example, his/her shopping item IDs, preferred brands or favorite shop IDs.

Remember the model defined in Eq. 2, most computation cost focus on $\mu_i^T x$ and $w_i^T x$. By employing the common feature trick, we can split the calculation into common and non-common parts and rewrite them as follows:

$$\begin{aligned} \mu_i^T x &= \mu_{i,c}^T x_c + \mu_{i,nc}^T x_{nc} \\ w_i^T x &= w_{i,c}^T x_c + w_{i,nc}^T x_{nc} \end{aligned} \tag{13}$$

Hence, for the common feature part, we need just calculate once and then index the result, which will be utilized by the following samples.

Specifically, we optimize the parallel implementation with common features trick in the following three aspects:

- Group training samples with common features and make sure these samples are stored in the same worker.
- Save memory by storing common features shared by multiple samples only once.
- Speed up iteration by updating loss and gradient w.r.t. common features only once.

Due to the common feature pattern of our production data, employing the common feature trick improves the performance of training procedure greatly, which will be shown in the following section 4.3.

4 Experiments

In this session, we evaluate the performance of LS-PLM. Our datasets are generated from Alibaba's mobile display advertising product system. As shown in Table 1, we collect seven datasets in continuous sequential

Table 1: Alibaba's mobile display advertising CTR prediction datasets

Dataset	#features	#samples (training/validation/testing)
1	3.04×10^6	$1.34/0.25/0.26 \times 10^9$
2	3.27×10^6	$1.44/0.26/0.26 \times 10^9$
3	3.49×10^6	$1.56/0.26/0.25 \times 10^9$
4	3.67×10^6	$1.62/0.25/0.26 \times 10^9$
5	3.82×10^6	$1.69/0.26/0.26 \times 10^9$
6	3.95×10^6	$1.74/0.26/0.26 \times 10^9$
7	4.07×10^6	$1.78/0.26/0.26 \times 10^9$

periods, aiming to evaluate the consist performance of the proposed model, which is important for online product serving. In each dataset, training/validation/testing samples are disjointly collected from different days, with proportion about 7:1:1. AUC (Fawcett 2006) metric is used to evaluate the model performance.

4.1 Effectiveness of division number

LS-PLM is a piece-wise linear model, with division number **m** controlling the model capacity. We evaluate the division effectiveness on model's performance. It is executed on dataset 1 and results are shown in Figure 4.

Generally speaking, larger m means more parameters and thus leads to larger model capacity. But the training cost will also increase, both time and memory. Hence, in real applications we have to balance the model performance with the training cost.

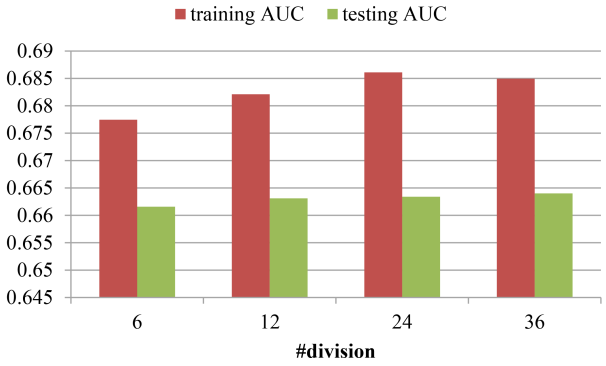


Figure 4: Model performance with different divisions.

Figure 4 shows the training and testing AUC with different division number m. We try $m = 6, 12, 24, 36$, the testing AUC for $m = 12$ is markedly better than $m = 6$, and improvement for $m = 24, 36$ is relatively gentle. Thus in all the following experiments , the parameter m is set as 12 for LS-PLM model.

4.2 Effectiveness of regularization

As stated in Session 2, in order to keep our model simpler and more generalized, we prefer to constrain the model parameters sparse by both L_1 and $L_{2,1}$ norms. Here we evaluate the strength of both the regularization terms.

Table 2 gives the results. As expected, both L_1 and $L_{2,1}$ norms can push our model to be sparse. Model trained with $L_{2,1}$ norm has only 9.4% non-zero parameters left and 18.7% features are kept back. While in L_1 norm case, there are only 1.9% non-zero parameters left. Combining them together, we get the sparsest

Table 2: Regularization effects on model sparsity and performance

$\beta/\lambda(L_1/L_{2,1})$	#features	#non-zero parameters	testing auc
0/0	3.04×10^6	7.30×10^7	0.6489
0/1	5.68×10^5	6.64×10^6	0.6570
1/0	3.87×10^5	1.33×10^6	0.6617
1/1	2.55×10^5	1.15×10^6	0.6629

Table 3: Training cost comparision with/without common feature trick

Cost	Without CF.	With CF.	Cost Saving
Memory cost/node	89.2 GB	31 GB	65.2%
Time cost/iteration	121s	10s	91.7%

result. Meanwhile, models trained with different norm get different AUC performance. Again adding the two norms together the model reaches the best AUC performance.

In this experiment, the hyper-parameter m is set to be 12. Parameters β and λ are selected by grid search. $\{0.01, 0.1, 1, 10\}$ are tried for both norms in the all cases. The model with $\beta = 1$ and $\lambda = 1$ preforms best.

4.3 Effectiveness of common feature trick

We prove the effectiveness of common features trick. Specifically, we set up the experiments with 100 workers, each of which uses 12 CPU cores, with up to 110 GB memory totally. As shown in Table 3, compressing instances with common feature trick does not affect the actual dimensions of feature space. However, in practice we can significantly reduce memory usage (reduce to about 1/3) and speed up the calculation (around 12 times faster) compared to the training without common feature trick.

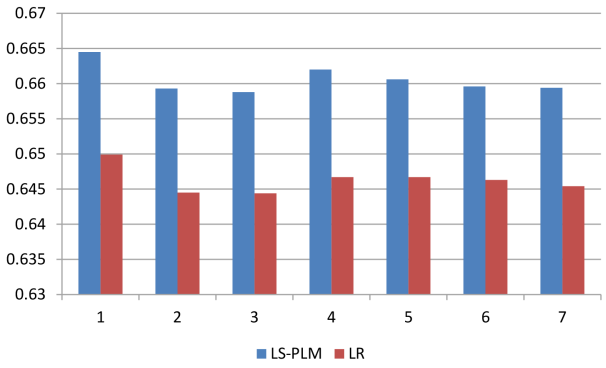


Figure 5: Model performance comparison on 7 different test datasets. LS-PLM owns consistent and markable promotion compared with LR.

4.4 Comparison with LR

We now compare LS-PLM with LR, the widely used CTR prediction model in product setting. The two models are trained using our distributed implementation architecture, running with hundreds of machines for speed-up. The choice of the L_1 and $L_{2,1}$ parameters for LS-PLM and the L_1 parameter for LR are based on grid search. $\beta = 0.01, 0.1, 1, 10$ and $\lambda = 0.01, 0.1, 1, 10$ are tried. The best parameters are $\beta = 1$ and $\lambda = 1$ for LS-PLM, and $\beta = 1$ for LR.

As shown in Figure 5, LS-PLM outperforms LR clearly. The average improvement of AUC for LR is 1.44%, which has significant impact to the overall online ad system performance. Besides, the improvement is stable. This ensures LS-PLM can be safely deployed for daily online production system.

5 Conclusions

In this paper, a piece-wise linear model, LS-PLM for CTR prediction problem is proposed. It can capture the nonlinear pattern from sparse data and save us from heavy feature engineering jobs, which is crucial for real industry applications. Additionally, powered by our distributed and optimized implementation, our algorithm can handle problems of billion samples with ten million parameters, which is the typical industrial data volume. Regularization terms of L_1 and $L_{2,1}$ are utilized to keep the model sparse. Since 2012, LS-PLM has become the main CTR prediction model in alibaba's online display advertising system, serving hundreds of millions users every day.

Acknowledgments

We would like to thank Xingya Dai and Yanghui Yan for their help for this work.

Appendix

A Proof of Lemma 2.1

Proof. the definition of $f'(\Theta; d)$ is given as follows:

$$\begin{aligned} f'(\Theta; d) =& \lim_{\alpha \downarrow 0} \frac{f(\Theta + \alpha d) - f(\Theta)}{\alpha} \\ =& \lim_{\alpha \downarrow 0} \frac{\text{loss}(\Theta + \alpha d) - \text{loss}(\Theta)}{\alpha} \\ &+ \lim_{\alpha \downarrow 0} \lambda \frac{\|\Theta + \alpha d\|_{2,1} - \|\Theta\|_{2,1}}{\alpha} \\ &+ \lim_{\alpha \downarrow 0} \beta \frac{\|\Theta + \alpha d\|_1 - \|\Theta\|_1}{\alpha}. \end{aligned} \tag{14}$$

As the gradient of loss function exists for any Θ, the directional derivative for the first part is

$$\lim_{\alpha \downarrow 0} \frac{\text{loss}(\Theta + \alpha d) - \text{loss}(\Theta)}{\alpha} = \nabla \text{loss}(\Theta)^T d \tag{15}$$

For the second part, we know if $\|\Theta_{i\cdot}\|_{2,1} \neq 0$, the $L_{2,1}$ norm's partial derivative exists. So the directional derivative is

$$\lim_{\alpha \downarrow 0} \lambda \frac{\|\Theta_{i\cdot} + \alpha d_{i\cdot}\|_{2,1} - \|\Theta_{i\cdot}\|_{2,1}}{\alpha} = \lambda \frac{\Theta_{i\cdot}^T d_{i\cdot}}{\|\Theta_{i\cdot}\|_{2,1}}. \tag{16}$$

However, when $\|\Theta_{i\cdot}\|_{2,1} = 0$, it means $\Theta_{ij} = 0, 1 \leq j \leq 2m$. Then its directional derivative can be denoted as follows:

$$\begin{aligned} &\lim_{\alpha \downarrow 0} \lambda \frac{\|\Theta_{i\cdot} + \alpha d_{i\cdot}\|_{2,1} - \|\Theta_{i\cdot}\|_{2,1}}{\alpha} \\ &= \lim_{\alpha \downarrow 0} \lambda \frac{\|\alpha d_{i\cdot}\|_{2,1}}{\alpha} \\ &= \lambda \|d_{i\cdot}\|_{2,1} \end{aligned} \tag{17}$$

So combine the above cases in Eq. (16) and Eq. (17), we get the directional derivative for the second part:

$$\lim_{\alpha\downarrow 0}\lambda\frac{\|\Theta+\alpha d\|_{2,1}-\|\Theta\|_{2,1}}{\alpha} \tag{18}$$
$$=\sum_{\|\Theta_{i\cdot}\|_{2,1}\neq 0}\lambda\frac{\Theta_{i\cdot}^T d_{i\cdot}}{\|\Theta_{i\cdot}\|_{2,1}}+\sum_{\|\Theta_{i\cdot}\|_{2,1}=0}\lambda\|d_{i\cdot}\|_{2,1}$$

Same as the second part, the direction derivative for the third part is:

$$\lim_{\alpha\downarrow 0}\beta\frac{\|\Theta+\alpha d\|_1-\|\Theta\|_1}{\alpha} \tag{19}$$
$$=\sum_{\|\Theta_{ij}\|_1\neq 0}\beta\text{sign}(\Theta_{ij})d_{ij}+\sum_{\|\Theta_{ij}\|_1=0}\beta|d_{ij}|$$

Based on Eq. (15), Eq. (18) and Eq. (19), we get that for any Θ and direction d, $f'(\Theta;d)$ exists.

□

B Proof of Proposition 2.2

Proof. Finding the expected direction turns to an optimization problem, which is formulated as follows:

$$\min_d f'(\Theta;d)\quad \text{s.t. } \|d\|^2\leq C. \tag{20}$$

Here the direction d is bounded by a constant scalar C. To solve this problem, we employ lagrange function to combine the objective function and inequality function together:

$$L(d,\mu)=f'(\Theta;d)+\mu(\|d\|^2-C). \tag{21}$$

Here $\mu\geq 0$ is the lagrange multiplier. Setting the partial derivative of d with respect to $L(d,\mu)$ to zero has three cases.

Define $s=-\nabla loss(\Theta)_{ij}-\lambda\frac{\Theta_{ij}}{\|\Theta_{i\cdot}\|_{2,1}}$

a. when $\Theta_{ij}\neq 0$, it implies that

$$2\mu d_{ij}=s-\beta\text{sign}(\Theta_{ij})$$

b. when $\Theta_{ij}=0$ and $\|\Theta_{i\cdot}\|_{2,1}>0$, it is easy to have

$$2\mu d_{ij}=\max\{|s|-\beta,0\}\text{sign}(s)$$

c. We give more details when $\Theta_{ij}=0$ and $\|\Theta_{i\cdot}\|_{2,1}=0$. For $d_{i\cdot}$ we have

$$\frac{\partial L(d,\mu)}{\partial d_{i\cdot}}=\nabla\text{loss}(\Theta)_{i\cdot}+\beta\text{sign}(d_{i\cdot})+\lambda\frac{d_{i\cdot}}{\|d_{i\cdot}\|_{2,1}}+2\mu d_{i\cdot}=0.$$

Here we use $\text{sign}(d_{i\cdot})=[\text{sign}(d_{i1}),\ldots,\text{sign}(d_{i2m})]^T$ for simplicity. Then we get

$$(2\mu+\frac{\lambda}{\|d_{i\cdot}\|_{2,1}})d_{i\cdot}=-\nabla\text{loss}(\Theta)_{i\cdot}-\beta\text{sign}(d_{i\cdot})$$

which implies that $\text{sign}(d_{i\cdot})=\text{sign}(-\nabla loss(\Theta)_{i\cdot}-\beta\text{sign}(d_{i\cdot}))$. When $d_{ij}\geq 0$, it implies $-\nabla\text{loss}(\Theta)_{ij}-\beta\text{sign}(d_{ij})\geq 0$. Inversely, we have $-\nabla\text{loss}(\Theta)_{ij}-\beta\text{sign}(d_{ij})\leq 0$ when $d_{ij}\leq 0$. So we define $v\doteq-\nabla\text{loss}(\Theta)_{i\cdot}-\beta\text{sign}(d_{i\cdot})$ and $v_j=\max\{|-\nabla\text{loss}(\Theta)_{ij}|-\beta,0\}\text{sign}(-\nabla\text{loss}(\Theta)_{ij})$. So

$$\begin{aligned}&(2\mu+\tfrac{\lambda}{\|d_{i\cdot}\|_{2,1}})d_{i\cdot}=v\\ \Rightarrow\quad&(2u\|d_{i\cdot}\|+\lambda)\|d_{i\cdot}\|=\|v\|\|d_{i\cdot}\|\\ \Rightarrow\quad&(2u\|d_{i\cdot}\|+\lambda)=\|v\|.\end{aligned}$$

As $\|d_{i\cdot}\| \geq 0$, we have $2\mu\|d_{i\cdot}\| = \max(\|v\| - \lambda, 0)$. Thus $2\mu d_{ij} = \frac{\max(\|v\|-\lambda,0)}{\|v\|}v$

The lagrange multiplier u is a scalar, and it has equivalent influence for all d_{ij}. We can see that the optimal direction which is bounded by C is the same direction as we defined in Eq. (9) without considering the constant scalar μ. Here we finish our proof.

□

References

[1] Andrew G. and Gao J. (2007) Scalable Training of L_1-Regularized Log-Linear Models. *Proceedings of the 24-th International Conference on Machine Learning*.

[2] Bertsekas, D. (2003) *Nonlinear Programming*. Springer US, 51–88.

[3] Brendan H., Holt G., Sculley D., Young M., Ebner D., Grady J., Nie L., Phillips. T, Davydov E., Golovin D., Chikkerur S., Liu D., Wattenberg M., Hrafnkelsson A., Boulos T., Kubica J. (2013) Ad Click Prediction: a View from the Trenches. *Proceedings of the 19-th KDD*.

[4] Fawcett T. (2006) An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 861–874.

[5] Friedman J. (1999) Greedy Function Approximation: A Gradient Boosting Machine. *Technical Report, Dept. of Statistics*, Stanford University.

[6] Hilbe M. (2009) Logistic regression models. CRC Press.

[7] He X., Pan J., Jin O., Xu T., Liu B., Xu T, Shi Y., Atallah A, Herbrich R., Bowers S., Candela J. (2014) Practical Lessons from Predicting Clicks on Ads at Facebook. *Proceedings of the 20-th KDD*.

[8] Jordan I., Jacobs A (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2): 181-214.

[9] Kivinen J., Warmuth M K. (1998) Relative Loss Bounds for Multidimensional Regression Problems. *Machine Learning*, 45(3):301-329.

[10] Rendle S. (2010) Factorization Machines. *Proceedings of the 10th IEEE International Conference on Data Mining*.

[11] Roth S, Black M J. (2009) Fields of experts. *International Journal of Computer Vision*, 82(2): 205–229.

[12] Safavian S. R., Landgrebe D. (1990) A survey of decision tree classifier methodology[J].

[13] Wang P.-M and Puterman M. (1998) Mixed Logistic Regression Models. *Journal of Agricultural, Biological, and Environmental Statistics*, 3(2), 175–200.

[14] Zhang T. (2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Proceedings of the twenty-first international conference on Machine learning*. ACM, 116.

[15] Gai K. `http://club.alibabatech.org/resource_detail.htm?topicId=106`