Click Through Rate Prediction with Hybrid Models

Lin Li, Zhenyu Zhou, Yi Duan, Jinshu Cui, Sahil Garg
University of Southern California
Los Angeles, CA 90007
{lli874, zhenyuz, yiduan, jinshucu, sahilgar}@usc.edu

Abstract

Click-through rate (CTR) has been identified very important for both ads platforms and advertisers. As a result, click prediction systems are more and more widely used for sponsored search and real-time bidding. Meanwhile, fueled by the global proliferation of mobile phones and devices, mobile advertising has recently seen dramatic growth. Avazu data on Kaggle brings us the challenge to predict whether a mobile ad is clicked. To tackle this challenge, We first implemented three existing models including logistic regression, factorization machines and gradient boosting regression trees. Each of the models was trained independently to get the optimal parameters. We then proposed an ensemble method which greatly improves the results. Our solution obtained a logarithmic loss of 0.39704 using only 10% of the training data.

1 Introduction

Predicting ad click-through rates (CTR) is a massive-scale learning problem that is central to the multi-billion dollar online advertising industry. The revenue in advertising is usually determined by the number of clicks, which has been utilized by large search engines such as Google, Yahoo and Microsoft [1]. In turn, CTR is important for both the advertisers to compete with each other and the ads platforms to maximize their revenue and attract more customers. Therefore, predicting and further increasing CTR is a major task for ads platforms to accomplish. Also, as the global proliferation of mobile phones and devices, it becomes a challenging job to make ads compatible and efficient on cross-device tasks,which is a strength of emerging companies like Avazu.

Based on historical data, researchers have established ways to estimate and predict CTR. A widely used approach is logistic regression.For instance, Richardson, Dominowska and Ragno [1] proved a logistic regression model using featurs of ads, terms, and advertisers can accurately predict CTR for new ads. In addition, Graepel et al. proposed an algorithm using Bayesian probit regression to predict CTR for Bing search [2]. Other methods include clustering [3], factorization machines [4], and ensembling [5]. It was found multiple features of ads, including position [5], content and design of banners [6], keywords [3], etc., can impact CTR. The historical data we received from Avazu consists of a lot of features related to mobile devices, which is not evaluated extensively in previous studies. Therefore, it is valuable to investigate CTR prediction on this fresh data by applying existing approaches and exploring on new approaches.

In this paper, we report our approach to tackle the click-through rate prdiction challenge of Avazu data with hybrid models. We first trained three models individually and get the optimal parameters. Then we proposed an approach to combine each individual together to get a better solution. The results show that we can reach a good performance utilizing only 10% of the training data.

The remainder of the paper is organized as follows. In Section 2, we describe feature selection process. In Section 3, we introduce three models tested individually: logistic regression, factorization machines and gradient boosting regression trees. Results of each individual model and the ensemble are discussed in Section 4. We conclude our work and point out some future directions in Section 5.

2 Feature Selection

Due to limited computing resources, we only chose 10 percent of data to train our model. Observing that the training data is given according to the time-stamp, we draw randomly instead of using the first 10 percent. We also build a validation set with a size equal to the size of test data in a similar way. There are 22 features in total, including 9 anonymized categorical variables. Based on the selected training set, we first calculated conditional entropy of each feature with respect to the prediction label. Furthermore, we examined pair-wise conditional entropy. Results indicated site_id, app_id, device_id, device_ip, C14, C17 are the features which have the stronger connections to the training label. Surprisingly, when we applied these fixed features to the test set, device_id, device_ip turned out to be relatively useless as they're too scattered (sparse). Thus, we dropped these two features. The other four features from the six are assigned a twice weight compared with the rest of features when they feed to the model.

Since all the features are categorical, we expanded every feature into a binary vector so that each dimension indicates whether the instance belongs to the corresponding category. By concatenating binary vectors of all the features, it yields a overall feature vector of 18673 dimension. Obviously it is a high dimensional space. We tried to apply dimension reduction techniques to it, including PCA and merging small categories into one. However, both didn't work quite well in terms of predicting test labels. Therefeore we end up with using original dimensionality in all of our experiments.

3 Models

3.1 Logistic Regression

Logistic regression (LR) is a reasonable approach in evaluating and predicting CTR, as proposed in the literature [1]. Given a set of features, probability of clicking an ad, i.e., CTR, can be predicted as:

$$\hat{y} = \frac{1}{1+e^{-z}}, \quad z = \sum_k w_k f_k \tag{1}$$

where f_k represents the value of kth feature and w_k represents the weight of that feature learned by the model. The implementation of logistic regression can be accessed from class LogisticRegression in scikit-learn [7]. This implementation can fit a multiclass (one-vs-rest) logistic regression with an option of L2 or L1 regularization. As an optimization problem, binary class L2 penalized logistic regression minimizes the following cost function:

$$\min_{\mathbf{w},c} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \log\exp\Big(-y_i(\mathbf{x}_i^T\mathbf{w}+c)+1\Big) \tag{2}$$

Similarly, L1 regularized logistic regression solves the following optimization problem:

$$\min_{\mathbf{w},c} \quad ||\mathbf{w}||_1 + C\sum_{i=1}^{n} \log\exp\Big(-y_i(\mathbf{x}_i^T\mathbf{w}+c)+1\Big) \tag{3}$$

L1 penalization yields sparse predicting weights. The class LogisticRegression in scikit-learn implements L1 and L2 regularized logistic regression using the LIBLINEAR [8] library. To convert the train data and test data into sparse input, we also use numpy and scipy [9].

3.2 Factorization Machines

Factorization machines (FM) [10] are useful approaches that provide high accuracy in several important prediction problems, for example, recommender systems. They combine the generality of feature engineering with the superiority of factorization models in estimating interactions between categorical variables of large domain. Factorization Machines have proven to be successful in predicting click-through rate of ads in search engines in KDD Cup 2012 [4].

Let us suppose the data of our prediction task is described by a multiset S of cases $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^p$ is p-dimensional feature vector and y is the corresponding target. Factorization Machines

model all nested interactions up to order d between the p input variables in feature vector $\mathbf{x}$ using factorized interaction parameters. Factorization machine (FM) model of order $d = 2$ is defined as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=j+1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{j,f} v_{j',f} \tag{4}$$

where k is the dimensionality of the factorization. The model parameters involved are $\Theta = \{w_0, w_1, ..., w_p, v_{1,1}, ..., v_{p,k}\}$ where $w_0 \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^p, \mathbf{v} \in \mathbb{R}^{p \times k}$.

The first part of the FM model contains the unary interactions of each input variable x_j with the target, which is exactly as the linear regression model mentioned above. The second part with the two nested sums contains all pairware interactions of input variables. To make inference over model parameters Θ, several learning algorithms have been proposed from point estimators using coordinate descent (CD) and stochastic gradient descent (SGD) to Bayesian inference based on Markov Chain Monte Carlo (MCMC) sampling. In our experiments, we chose MCMC, which makes Bayesian inference over the predictive distribution by marginalizing over the model parameters and hyperparameters

$$p(y|\mathbf{x}, S) = \int p(y|\mathbf{x}, \Theta, \lambda) p(\Theta, \lambda|S) d\{\Theta, \lambda\} \tag{5}$$

This distribution is approximated by sampling from the posterior distribution $p(\Theta, \lambda|S)$. The reason why we chose MCMC over SGD and ALS is that MCMC is much easier. The regularization is integrated and there is no learning rate in MCMC. Except the dimension number of latent vectors, The only hyperparameter we have to tune is the standard derivation of the Gaussian distribution that is used to initialize model parameters. This is due to the reason that our optimization problem is not convex. A proper choice of starting position can accelerate the convergence of the sampler. The performance of standard deviation is quite stable among different choices of features and number of latent dimensions. In CD and SGD, however, we need to tune the model by enumerating all the parameters from a candidate set, which is impractical due to our limited time and hardware. What is more, the performance of Bayesian models is usually better than non-Bayesian models.

3.3 Gradient Boosting Regression Trees

Tree model is a widely used machine learning approach for mainly three reasons. First, it is easy to construct without careful treatment of the features. Second, unlike linear models, it explicitly learns high-order interactions among features. Third, the complexity of training a tree model is relatively low, which makes it easy to be applied to large-scale datasets. Since over half of features in this task are anonymous, different features can interplay with each other, and the dataset is fairly big, we think tree model is a good choice. More precisely, we choose Gradient Boosting Regression Tree (GBRT) in the family of tree models as it can specify the loss function that the model is learning. We choose Logarithmic Loss to keep consistent with the evaluation method of the competition.

Using the notations in the previous sections, assume we aim at training a ensemble model with K individual trees, the GBRT model is defined as

$$\hat{y}(\mathbf{x}) = \sum_{k=1}^{K} f_k(\mathbf{x}) \tag{6}$$

in which each $f_k(\mathbf{x})$ maps part of the final score. Typically, we construct the trees in an additive way (boosting), namely iteratively we add a single decision tree to the model, by defining staged prediction and objection function as follows,

$$\hat{y}^{(t)}(\mathbf{x}) = \hat{y}^{(t-1)}(\mathbf{x}) + f_t(\mathbf{x}) \tag{7}$$

$$\begin{aligned} L &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}(\mathbf{x}_i)) + \sum_{i=1}^{t} \Omega(f_i(\cdot)) \\ &\approx \sum_{i=1}^{n} \big[l(y_i, \hat{y}_i^{(t-1)}) + \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} f_t(\mathbf{x}) + \frac{1}{2} \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial^2 \hat{y}_i^{(t-1)}} f_t(\mathbf{x})^2 \big] + \sum_{i=1}^{t} \Omega(f_i(\cdot)) \end{aligned} \tag{8}$$

in which $\Omega(f_i(\cdot))$ is a regularizer that models the complexity of each single decision tree. The depth of the tree and the smoothness of the leaf nodes are two key factors.

Thus the construction of each individual tree can be totally separated. For each decision tree, the solution still varies. We use package XGBoost [11] which applies a greedy approach to scan the whole features and choose the best split at every depth. Afterwards, it takes the regularize component into account and prunes the tree. In addition, Xgboost wins "HEP meets ML Award" in Higgs Boson Challenge, which is a fully optimized emerging machine learning toolkit.

4 Results

In this section, we present the results of each individual model and the ensemble model. There are two questions we need to answer: (1) What is the performance of each individual model? (2) How much can we improve the performance through ensemble of models?

4.1 Experiment Setting

The experiments are conducted on a machine with 2.4 GHz Intel Core i5 CPU and 8 GB 1600 MHz DDR3 memory. The training data provided in the competition contains 10 days of click-through data with $40,428,968$ records. The test data contains 1 day of click-through data with $4,577,465$ records. Due to the limitation of hardware, it is impractical for us to load all the training data into memory to train our models. Online algorithms can be an option for us because of its low requirement of memory, but the performance of online algorithms is quite limited. Instead, we randomly sampled 10% of the training data to train our models. To obtain the validation data, we randomly sampled a subset of the same size with test data from the remaining 90% of the training data. We train local models on the training data and test them on the validation data before submission. We use the Logarithmic Loss (LogLoss) as the evaluation metric, which is in the following form:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right] \tag{9}$$

where $\hat{y}_i$ is our predicted probability. To prevent adding infinite to the loss and making every other entry pointless, predictions are bounded away from the extremes by a small value ϵ. The results reported in this paper are all submission results.

4.2 Result of Logistic Regression

In logistic regression, we mainly experimented using L1 or L2 regularizer and solving in prime or dual form. Different combinations lead to four different configurations. We found that using L2 regularizer and solving in dual form performed the best.

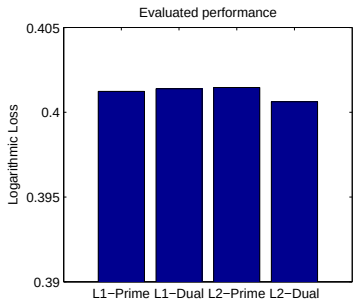

Figure 1: Average performance for different configurations

4.3 Result of Factorization Machines

We used Markov Chain Monte Carlo (MCMC) sampling to make inference for our model parameters. In each round, we sample a prediction from the posterior distribution $p(\Theta, \lambda|S)$. The final

result is obtained by averaging the predictions of each round. The only parameters we need to tune are dimension number of latent vectors and the standard derivation of the Gaussian distribution that is used to initialize model parameters. In our experiments, we set the number of sampling round to be 200. We discarded the results of first 10 samplings to get a better performance. Figure 2 shows the average performance for different values of dimension number and standard derivation. When we tuned dimension number, we set standard derivation to be 0.005, and when we tuned standard derivation, we set dimension number to be 16. We can see that as dimension number or standard

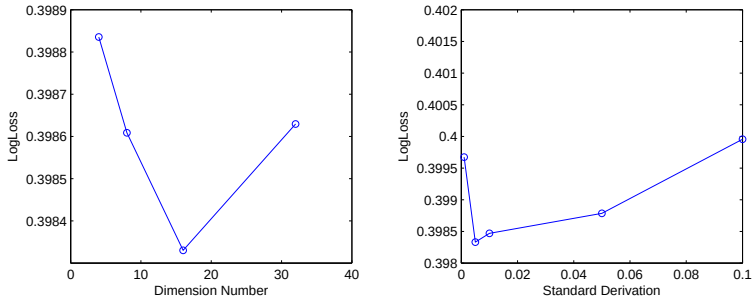

Figure 2: Average performance for dimension number and standard derivation

derivation increases, LogLoss first decreases and then increases. By setting dimension number as 16 and standard derivation as 0.005, which are their optimal values, we get the LogLoss 0.39833.

4.4 Result of Gradient Boosting Regression Tree

Since GBRT is quite robust against the number of trees in the model, we tried to learn as many rounds as possible in our experiment. By contrast, we constrained the depth of each tree as it would potentially cause overfitting. In Figure 3, the number of trees of the left plot is fixed by 500, while the

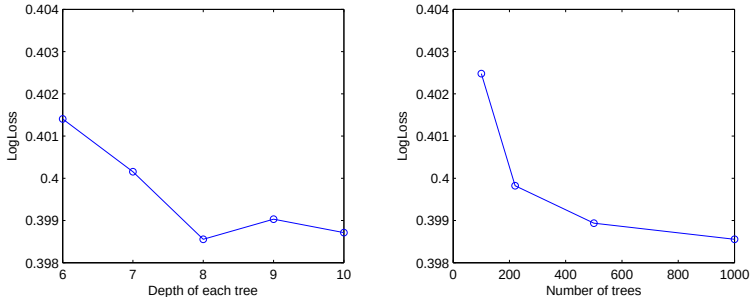

Figure 3: Average performance for depth of each tree and number of trees

depth of each tree of the right plot is fixed by 8. As we stated above, we found that the configuration with $depth = 8, \#trees = 1000$ would achieve the best result.

4.5 Result of Ensemble

We ensembled the three models by weighting them using the best result of each individual model. The final output is defined as

$$\widetilde{y}(\mathbf{x}; \alpha) = \sum_{i=1}^{3} \alpha_i y_i(\mathbf{x}) \tag{10}$$

where y_1, y_2, y_3 are the results from the three individual approaches. We applied grid search with a 0.05 stepsize on our validation set to get the best weights. The best weight was 0.40 for FM, 0.45 for GBRT and 0.15 for LR. Our ensemble method received a 0.39704 performance on the test set.

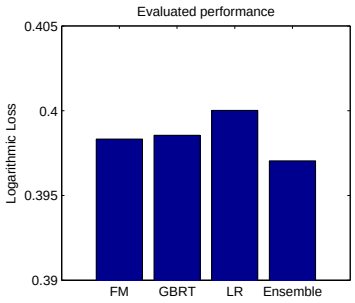

Figure 4: Performance comparison of FM, GBRT, LR and ensemble model

5 Conclusion and Future Work

Our method combines three models, namely Factorization Machines, Gradient Boosting Leearning Tree and Logistic Regression. These three models cover different aspect of the task, i.e., how features are independently/interactively contribute to the prediction label and how user, app, site are related. Combining the three models gives us a comprehensive result. All the three models are trained based on 10 percent data. Our model achieved a 0.39704 submission performance, and it is robust to different test sets as we did not trick the particular dataset in the competition. Additionally, our model can be trained in a relatively short time (less than one day). The weakness of our model majorly lies in the following two points. We only trained on a small fraction of training data, which is a great loss. This can be easily solved if we have a more powerful machine for computation. The other one is we did not do a lot of processing to the raw feature data. Our model can be further improved if we can pre-process the data further and find a better representation of the data.

References

[1] Richardson, M., Ewa D. & Robert R. (2007) Predicting clicks: estimating the click-through rate for new ads. *Proceedings of the 16th international conference on World Wide Web.* ACM.

[2] Graepel, T., Candela, J.Q., Borchert, T. & Herbrich, R. (2010) Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. *In Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. pp. 13-20.

[3] Regelson, M. & Fain, D. (2006) Predicting click-through rate using keyword clusters. *In Proceedings of the Second Workshop on Sponsored Search Auctions.* vol. 9623.

[4] Rendle, S. (2012) Social network and click-through prediction with factorization machines. *KDD Cup.*

[5] Dembczynski, K., Kotlowski, W. & Weiss, D. (2008) Predicting ads click-through rate with decision rules. *In Workshop on Targeting and Ranking in Online Advertising*. vol. 2008.

[6] Lohtia, R., Naveen D. & Edmund K.H. (2003) The impact of content and design elements on banner advertising click-through rates. *Journal of advertising Research.* **43** (4):410-418.

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R. Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research* 12, 2825-2830.

[8] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008) LIBLINEAR: A library for large linear classification *Journal of Machine Learning Research* 9, 1871-1874.

[9] van der Walt, S., Colbert, S. C., & Varoquaux., G. (2011) The NumPy Array: A Structure for Efficient Numerical Computation, *Computing in Science and Engineering.* 13, 22-30

[10] Rendle, S. (2012) Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology.* **3**(3), 57.

[11] Chen, T. XGBoost: eXtreme Gradient Boosting (Tree) Library.