Click Through Rate Prediction -- Project Report

CSE 5522 Advanced Artificial Intelligence

Ganga Reddy Tankasala, Javkhlan-Ochir Ganbat

Introduction

The rise in internet of technologies has made a significant growth in the per head consumption of internet data for various day to day activities . Online advertising picked up the pace and most of the internet companies are financially structured with revenue from advertisements(ad). But one of the challenge in this area is to determine users interest in the ad. Several online learning models have been implemented to predict click rate based on the history. We have explored , identified and implemented some of the interesting algorithms to address this challenge. Our data is part of the Kaggle.com's competition data on click through rate.

Data Description

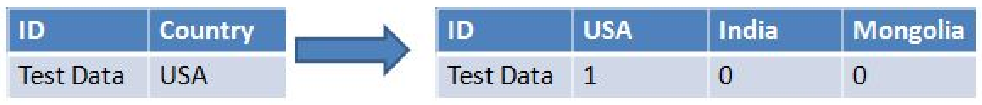
We have a 10 days of click through recording of mobile advertising data from Avazu.com. In total there are approximately 47 million training data each includes 24 features, all are considered categorical values. Some features identity is known for example: hours, banner_id, banner_pos and such but also includes anonymized features dubbed C14-C21. We checked number of categories of some known features.

Feature Name	# of category	Feature Name	# of category
banner_pos	7	site_category	26
site_id	4737	app_id	8552
site_domain	7745	app_domain	559

One-Hot encoding and Hashing Trick

One of the big problem that we had was how to convert the categorical values into numerical values since logistic regression and FTRL needs numerical values. We researched and found that creating additional dummy variables through the method called one-hot encoding. It creates binary variable for each category in the feature which means it will have value 1 in only one place where the actual category matches and 0 for the rest of the dummy variables.

For example: feature called *country* contains following county names; USA, India, Mongolia. Then we will split that *country* feature into 3 new feature each called *USA, India* and *Mongolia.*



ID	Country
Test Data	USA

ID	USA	India	Mongolia
Test Data	1	0	0

Unfortunately this method also has some drawbacks, it dramatically increases the dimensionality of the feature space. For example: categorical variable that takes 10 different values then we need to create 10 new binary dummy variable. Also we do not know how many category or when a new category would come in thus conversion to numeric becomes even more complicated but there is a clever trick for this. Hash function that maps string input into an integer value. We can map any input string into integer value even if it is a new value. However, there is an issue of collision but if the number of dimension is high enough this does not affect the performance that much.

There are two ways of applying hash function, use same hash function on all the feature; which is very simple and easy to implement or use different hash function for each feature; more complicated implementation and does not provide significant improvement in practice. We have implemented the one hash function over all the feature and set our maximum possible dimension to be 2^{20}.

Online and Offline methods

Our data size is very big and after conversion from categorical to numerical it has gotten even bigger therefore running an offline model on the data was not possible for us. Due to that we limited our approach to online learning algorithms (stochastic gradient descent). Therefore we did not explore more on Fisher’s Linear discriminant , SVM and k-NN, although there are variations of online algorithm that converges to SVM solution. SVM with an effective kernel is a good alternative to logistic regression. Our decision to not experiment SVM for this case was also supported by other competitive groups results who experimented with SVM kernels and got worse results than FTRL.

Random Forest

We have also explored the option of random forest. Since random forest should work well with categorical values. Unfortunately random forest is an offline model and needs to use all the data points to build the tree which unfortunately not feasible for us. Therefore, we subsampled 1 days of data and divide it into 10 parts and create 9 random forest parallely and used the rest to run the test on it and take the majority vote.

Online Logistic Regression using Stochastic Gradient Descent

Considering the scale and size of the data, online learning model best suits our needs. The application demand a continuous learning with respect to the experience. Online learning best suits high dimensional massive scale data which is growing over time. The advantage of sparsity can be exploited for efficiency in computation. Loss function which best suits {0,1} classification is log- loss. Sigmoid function can be used to evaluate the probability of prediction (belonging to {0,1} classes). Log loss is defined as follows (y = true value, p = predicted value = sigmoid(w*x)) log-loss = y*log(p) + (1-y)*log(1-p)

Despite of high prediction accuracy, stochastic gradient descent (SGD) fails in producing sparse models and modifications of the model is required to induce sparsity in the weights of the model. Regularization of the SGD has showed improved results but it could not handle sparsity effectively and the tradeoff of accuracy vs computational time in regularized logistic regression didn't seem reasonable.

One approach to overcome the limitation of above model is to develop a hybrid model which produces the highest prediction accuracy without compromising on sparsity. One such model is FTRL (Follow the proximally regularized leader) also known as FTRL-proximal.

FTRL

$w_{(t+1)} = (w_t - n_t g_t)$ -- update equation in SGD

The update model for the FTRL is modified as below

$$w_{(t+1)} = \frac{argmin}{w} (g_{1:t} * w + \sum_{s=1}^{t} \sigma_s \|w - w_s\|^2 + \lambda_1 \| w \|_1$$

update equation of FTRL

The model incorporates the previous history of gradients and the regularization parameter (λ_1) to induce sparsity. The implementation proves to be harder to implement as the size of model parameters increase with training data. However, mathematical adjustment of the above equation shows that the model coefficients can be updated in a recursive manner as described in the equation below.

lets define z_t as follows

$$z_{t-1} = (g_{1:t-1} - \sum_{s=1}^{t-1} \sigma_s w_s)$$

$$z_{t+1} = (g_{1:t} - \sum_{s=1}^{t} \sigma_s w_s)$$

z_t can be written recursively as follows

$$z_t - z_{t-1} = g_t - \sigma_t w_t$$

$$\sigma_s = \left(\frac{1}{n_t} - \frac{1}{n_{t-1}}\right)$$

$$w_{t+1} = \frac{argmin}{w} \ (g_{1:t} - \sum_{s=1}^{t} \sigma_s w_s)\, w + \lambda_1 \| w\|_1 + \frac{0.5}{n_t} \| w\|_2^2 + \frac{0.5}{n_t} \|w_s\|_2^2$$

$$\frac{1}{n_t} \|w_s\|_2^2 = constant$$

$$w_{t+1} = \frac{argmin}{w} \ (z_t * w + \lambda_1 \| w\|_1 + \frac{1}{n_t} \| w\|_2^2$$

Solving the above equation in closed form on a per coordinate(i) basis gives us

$$w_{t+1,i} = -n_t \ (z_{t,i} - sgn(z_{t,i}) * \lambda_1)$$

$$= 0 \quad \text{if} \quad z_{t,i} \leq \lambda_1$$

substituting $\lambda_1 = 0$ gives an approximation of the SGD without regularization.

FTRL Algorithm for per coordinate FTRL -proximal with L1 and L2 regularization for logistic regression

```
Input :

Parameters $\alpha$, $\beta$, $\lambda_1$, $\lambda_2$

# Initialization

$\forall\ i\ \varepsilon \{ 1,2,3......d(\ dimensions)\},\ initialize\ z_i = 0$ and $n_i = 0$

for t = 1 to T do

        x = vector Data sample and I = { i | $x_i \neq 0$ }

        for $i\ \varepsilon\ I$ compute
        $\nu = -\left(\lambda_2 + \frac{\beta + \sqrt{n_i}}{\alpha}\right)^{-1}$

        $w_{t+1,i} = -\nu\ (z_{t,i} - sgn(z_{t,i}) * \lambda_1)$

                  $= 0 \quad \text{if} \quad z_{t,i} \le \lambda_1$

    Predict p = $\sigma\ (w * x)$ using the $w_{t+1}$ computed from the above equation.

    observed value $y_t = \{0, 1\}$

    for all i $\varepsilon\, I$ do

            $g_i = (p_t - y_t) * x_t$

            $\sigma_i = \frac{1}{\alpha}(\sqrt{n_i + g_i^2} - \sqrt{n_i})$

            $z_i = z_i + (g_i - \sigma_i x_{t,i})$

            $n_i = n_i + g_i^2$

    end for

end for
```

Logistic regression

We have also implemented logistic regression with L1 regularization and constant learning rate. Logistic regression also run in online.

Result

FTRL result

We have tested the algorithm on various values of learning and regularization parameters. The usual benchmark was based on 10 fold cross validation and showed 82% accuracy. We also calculate holdout validation in every 100 data point we observe and calculate logloss. .

$\alpha = \textit{learning rate} \quad \beta = \textit{smoothing parameter}$

$L1 = L1 \text{ regularization} \quad L2 = L2 \text{ regularization}$

α	β	**L1**	**L2**	Accuracy	Logloss
0.05	1	1	1	0.829538963787	0.391904
0.1	1	1	1	0.833283	0.397111
0.5	0.5	1	1.5	0.826692246777	0.389618
0.5	1	1	1.5	0.827095836886	0.389722
1	1	1	1.5	0.824734689225	0.390685
2	1	1	1.5	0.821053384736	0.393591
2	2	1	1.5	0.822187269206	0.393293

Logistic regression

Simple logistic regression applied on our test data yields a decently good efficiency of 81.6 %. The log loss error of the data is 0.475 which is roughly 10 % more than FTRL algorithm. But the computational time of simple logistic regression is much less than FTRL which asserts confidence in SGD.

A simple timestamp report with respect to the iterations shows the following results

```
 Epoch 0 finished       logistic logloss: 0.475172 , elapsed time: 0:16:47.450486
```

Time per sample point = 0.00153 seconds (8 GB RAM , 2 GHz processor)

The accuracy of the test is 81.6383 %

Regularized logistic Regression:

Epoch 0 finished logistic logloss: 0.454551 , elapsed time: 0:23:45.453781

Time per sample point = 0.00207 , accuracy = 82.0465 %

FTRL

Regularized logistic Regression:

Epoch 0 finished logistic logloss: 0.388260 , elapsed time: 0:32:52.46923

Time per sample point = 0.00288 , accuracy = 83.2347 %

Log Loss comparison

Our goal is to minimize the log-loss values; therefore comparison between log-loss values show good indication of performance. Below is the comparison of log-loss value of FTRL and logistic regression that ran on full data set.

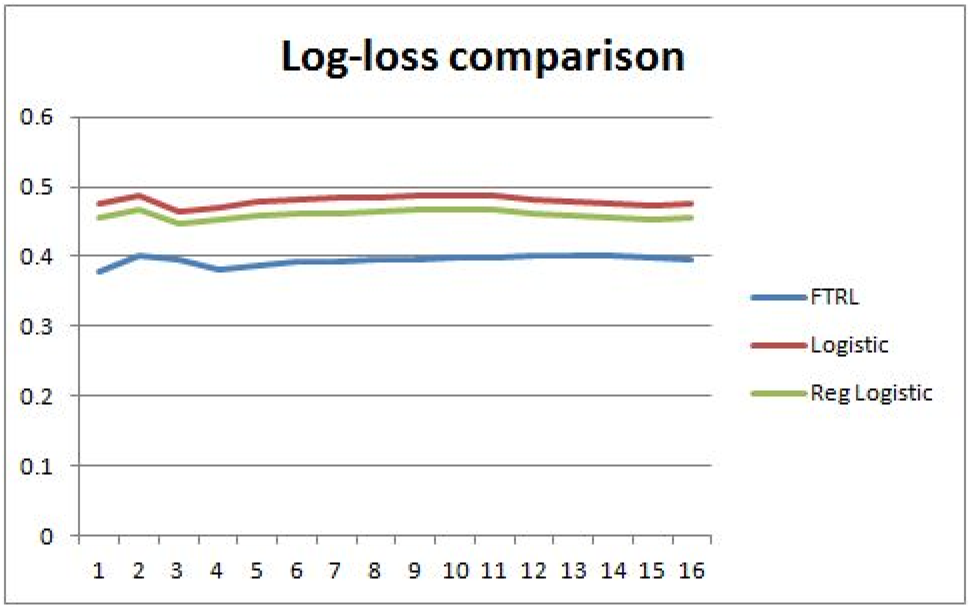


Table: Log-loss comparison

Random Forest

Since random forest requires all training data therefore we had to use supercomputing environment to calculate sample run. Our sample training data point contained 3.6 million data points each having 24 features and test data contained 0.4 million data points. Average runtime is 50 minute and average memory usage is 16 GB. Average accuracy of one run was 78.4280%.

Competition Results

Since the project is aligned with an kaggle challenge, we had the opportunity to look at other competitors results who are using various other methods, the current best results corresponds to a log loss value of 0.3874684. Our best result is 0.389722 which is applied on the cross validation set from training data set. (There are no results available for the test data set.)

Challenges encountered

The biggest challenge we faced was to efficiently convert the categorical values into numerical. Also running the whole simulation on our own laptop took very long time and as well as running into memory issues. Therefore, we have also run few heavy load jobs on OSC (Ohio SuperComputer Center) such as random forest and 10 fold cross validations. Except for random forest, no external Machine learning libraries are used.

Future Work

In the future, we want to explore more online algorithms especially the algorithm introduced in [3]. Also we plan to explore the relationship between two features. Currently we are assuming the all the features are independent of each other but that is almost often not the case in real life data. Conjunction of features are explored in [5] for categorical variables simply took the cartesian product which further increases the feature dimension therefore good hashing trick is crucial.

References

[1] Ad Click Prediction: a View from the Trenches

http://www.eecs.tufts.edu/~dsculley/papers/ad-click-prediction.pdf

[2] Predicting Clicks: Estimating the Click-Through Rate for New Ads

http://www2007.org/papers/paper784.pdf

[3] Web-scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoft Bing Search Engine

http://www.icml2010.org/papers/901.pdf

[4] Random Forests

https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
[5] Simple and scalable response prediction for display advertising
http://people.csail.mit.edu/romer/papers/TISTRespPredAds.pdf