A Logistic Regression Approach to Ad Click Prediction

Gouthami Kondakindi
MS, Computer Science, USC
kondakin@usc.edu

Satakshi Rana
MS, Computer Science, USC
satakshr@usc.edu

Aswin Rajkumar
MS, Computer Science, USC
aswinraj@usc.edu

Sai Kaushik Ponnekanti
MS, Computer Science, USC
ponnekan@usc.edu

Vinit Parakh
MS, Computer Science, USC
vparakh@usc.edu

Abstract

This paper presents an empirical study of using different machine learning techniques to predict whether an ad will be clicked or not. We perform click prediction on a binary scale - 1 for click and 0 for no click. We use clicks data from Avazu provided as a part of Kaggle competition as our data set.
We perform feature selection to remove features that do not help improve classifier accuracy. We inspect data manually and also use feature selection capability of Vowpal Wabbit for this purpose. We experiment with several supervised classification algorithms and observe that logistic regression with L2 regularization gives us better classification accuracy.

1 Introduction

Search advertising [7] is a major source of income for Google, Bing and Yahoo. These companies employ a pay-per-click policy for charging the advertisers. Pay-per-click policy says that an advertiser has to pay for only those ads which have some clicks on them. As a result, click prediction systems are very essential [6]. They estimate the best possible location for placing ads so as to maximize revenue by using Click-through Rate(CTR). Hence, Click-through Rate prediction is very crucial to Sponsored Search [3].

In this project, we try to predict clicks for Avazu data provided as a part of Kaggle challenge. We are given 5.87 GB of training data and 673 MB of test data. The train data has 24 attributes including the class label.Figure 1 shows data distribution of class label in training dataset. It can be seen that there are very less number of clicks. This is expected because out of the large number of ads displayed on a webpage, people hardly click on any ads.

Initially, we adopted the approach of naive bayes. However, it did not give us very good results because it assumes feature independence. Also, as the number of unique values is each feature is high and the training dataset is significantly large in size, it is computationally expensive to compute all the probabilities. Later on, we experimented with Vowpal Wabbit which is an open source fast out-of-core learning system library. The results we obtained with Vowpal Wabbit's logistic loss classifier were good and significantly better than naive bayes.

Logistic Regression is usually considered a good supervised classification algorithm for most of the datasets. So, we tried to see if logistic regression with stochastic gradient descent will give us better logloss score for this dataset. It gave us our best score of 0.393862 which is a significant improvement over our previous approaches. In the rest of the paper we describe our methodology and results in more detail.

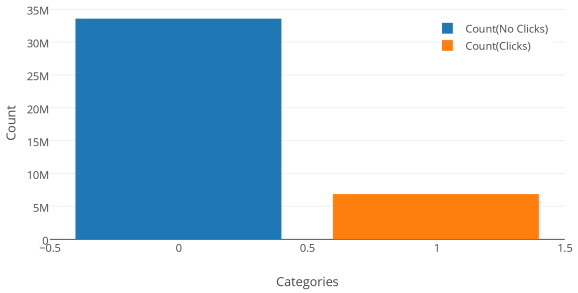

Figure 1: Class Label

2 Methodology

We first describe the data pre-processing and feature selection techniques that we adopted in our project. Thereafter, we explain our initial approaches namely naive bayes and vowpal wabbit. Finally, we explain the approach that gave us best results - logistic regression with stochastic gradient descent and weights regularization.

2.1 Data pre-processing

Feature Selection is a very important step in data pre-processing that determines the accuracy of a classifier. It helps remove unnecessary correlation in the data that might decrease accuracy.

We tried to implement Principal Component Analysis [4] to reduce the number of attributes in train data. PCA is often useful to measure data in terms of its Principal Components rather than on a normal x-y axis. Principal Components are the underlying structure in the data. They are the directions where there is most variance, the directions where the data is most spread out. PCA works best for discrete features. But upon observation, we found that all attributes were categorical attributes with number of unique values ranging from 3 to 6729487. So, we were unable to use PCA for data reduction.

As an alternative to PCA for dimensionality reduction, we decided to observe the data fields manually and remove fields that are not very helpful for classification task. For this purpose, we calculated the number of unique values for each data field. ID field has all unique values. So we removed it from the data. Also, there is a field named hour. It is given in YYMMDDHH format. Year and month are not necessary because the entire data belongs to the same month of the same year. So, we truncated them from the hour field. There is one important observation here. Click through rates usually follow some patterns: people click more ads on weekends as they are not busy, more number of clicks are observed during the day time when people are not sleeping. Taking these patterns into consideration, we further truncated the hour field to contain just the hour at which the ad was clicked and removed everything else.

Then, we added a new field day of week. This field contains the day of week equivalent for date given in the original dataset. Introducing this new field will make sure that the classifier learns the pattern similarities in ad clicks depending on the day of week. For example, if train data contains data about ads clicked on some Monday and if test data also contains some data about ads displayed on Monday, it might be possible that there are some similar trends in ad click rates on these days. Addition of the extra data field ensures that this relationship is introduced into the data.

We then converted this new data from categorical to binary form. We used one-hot encoding for this purpose. one-hot refers to a group of bits among which the legal combinations of values are only those with a single high - 1 bit and all the others low - 0 . Suppose we have a feature that comprises of 3 categories. We convert it into three binary features such that if a training instance belongs to category I in the original sample space, it will be mapped to [1,0,0] in transformed sample space. Category 2 will be represented as [0,1,0] and so on. This is called one-hot encoding for categorical features. This drastically increases the number of data fields but it is a convenient format

of having data because it can be easily input into any classification algorithm without any further modifications.

2.2 Classification

2.2.1 Naive Bayes

We started with a simple approach of Naive Bayes [5], to get a sense of how the data looks like. Naive Bayes, has an underlying assumption of independence between the features. This is a strong assumption and might not work for clicks data because certain features like location where ad is displayed, device on which user sees the ad are clearly related to each other. But, we wanted to implement Naive Bayes to see if our understanding about the data given to us is right.

For this approach we assumed the entire data to be categorical and calculated the counts for all the unique values in each feature with respect to whether the ad is clicked or not. Once we had the counts, the probabilities for the test data were calculated as follows:

$$P(Y = 1|X) = \frac{P(X|Y = 1)P(Y = 1)}{P(X|Y = 1) + P(X|Y = 0)} \tag{1}$$

As expected, Naive Bayes gave results which were bad. We got a log loss of 0.6013. However, this served as a good starting point.

2.2.2 Vowpal Wabbit

Our second approach was to use the machine learning tool Vowpal Wabbit [1] from Yahoo/Microsoft Research. It is a fast and efficient online machine learning algorithm. It supports different loss functions and optimizations. We first converted the labels of the training data to $\{$-1,1$\}$ and converted the comma-separated training file to vowpal-wabbit vw format. Except for stripping off the year, month and day information from the timestamp, the rest of the fields were left untouched. All the features were treated as categorical features.

We then chose logarithmic loss function from Vowpal Wabbit. After training using our train data, we tested the model on our test data converted into vw format and obtained predictions. These predictions were sent through a sigmoid function to get the final probabilistic output, which we submitted and achieved a reasonable logloss score of 0.47.

Next, we did feature selection and ignored the fields which had too many unique values like device id and device ip. We trained Vowpal Wabbit with this new data and achieved a better logloss score of 0.4007. Next, we randomized the rows of the training data using the command line. By shuffling the data, we were able to lower the test error to 0.4001. This is due to the fact that Vowpal Wabbit is an online learning algorithm and since the training data was initially sorted according to time, the model was skewed. After randomization, Vowpal Wabbit was able to predict the outcome better.

We tried various optimization parameters of Vowpal Wabbit like quadratic and cubic features, eliminated the constant feature and automatic feature selection. We also increased the bit precision from the default value of 18 using the -b option. We used Vowpal Wabbit utilities like $vw-varinfo$ and $vw - hypersearch$ to fine tune the model. We also randomized the initial weights and varied the learning rate and tested the output. Since clicks are of lower distribution in the data, we assigned higher weights for clicks than no clicks. But in spite of all these optimizations and experimentation, we were unable to cross the logloss barrier of 0.39901 that we achieved with Vowpal Wabbit.

This gave us a useful insight into how a classifier can only increase accuracy upto a certain point beyond which no matter how hard we try to fine tune the hyperparameters, it will not give any significant increase in accuracy. Figure 2 shows the optimizations that we tried to do and the logloss value we achieved with Vowpal Wabbit. Logloss value does not decrease after a certain point.

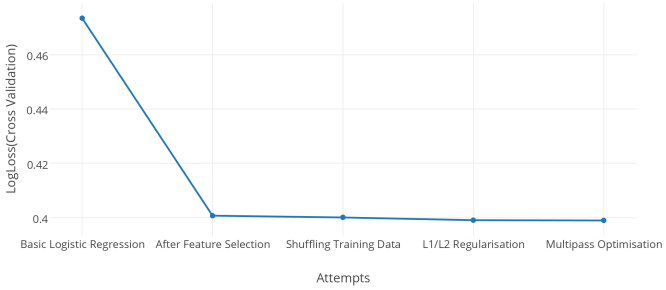

Figure 2: Vowpal Wabbit Logloss

2.2.3 Final Approach: Logistic Regression

Logistic Regression states that

$$P(y = 1|x; w) = \sigma(g(x)) \tag{2}$$

Where

$$g(x) = \sum_d w_d x_d = w^T x \tag{3}$$

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{4}$$

We used L2 Regularized Logistic Regression [2]. The regularization parameter (λ) is estimated using cross validation. Figure 3 shows all the values of λ that we had experimented with and the corresponding values of Log Loss that we got. It can be seen that the value of 0.00025 has given us the least Log Loss and thus it is chosen as the optimum λ value for this data. It could be possible that this value of $lambda$ is a local optimum and there might be some global optimum that could give us better results. But, this is the best value we could get by using greedy approach.

Because the data is so huge, it is not possible to load the entire data into memory. So, batch gradient descent and newton method were not an option for us. Hence, we used Stochastic Gradient Descent approach for updating the weights. The learning rate η for the Stochastic Gradient Descent has also been experimented upon. We have tried a few values for the learning rate and chose the optimum value as 0.01.

We implemented the above approach by taking initial weights to be 0. We also tried initializing the weights with values other than 0. The reason why we tried to do this is, there might be some values in the features from test data which were never seen in the training data. Initializing all weights to 0 means that we are giving a 0 weight to these unseen instances. That might lead to classification error. But giving a small initial weight to features did not seem to improve the accuracy in anyway. So, we went ahead with initializing the weights to 0.

3 Results

Figure 1 is a visualization of the clicks in training data. It can be seen that the number of training instances for class Click=0 are a lot more than the number of training instance for class Click=1. This is a useful insight into the distribution of data and the properties of clicks in real world. Click through rate is usually expected to be low

In Figure 2, we plot a graph for different log-loss scores we achieved by adopting different approaches in Vowpal Wabbit.

Figure 3 plots the different log-loss scores for different value of regularization parameter λ for logistic regression. We chose the value of λ that gave us the best results i.e. 0.00025.

In Table 1, we summarize the scores we achieved for the different techniques that we implemented in this project. It can be seen that logistic regression with proper data pre-processing gave us the best score and good ranking in the Kaggle competition.

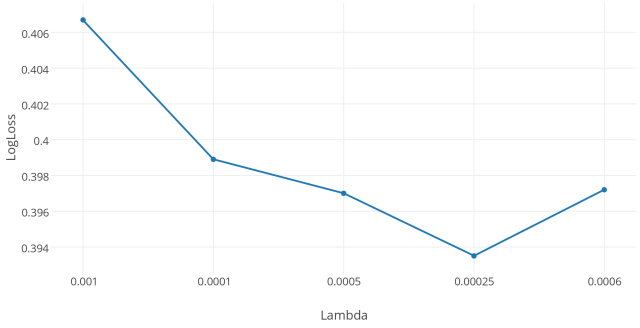

Figure 3: L2 Regularisation Parameter Estimation

Approach	Score
Naive Bayes	0.6013
Vowpal Wabbit	0.47
Vowpal Wabbit(Feature Selection)	0.4007
Vowpal Wabbit(Shuffled Data)	0.4001
Vowpal Wabbit(Optimization)	0.39901
Logistic Regression	0.396
Logistic Regression (Feature Selection)	0.393862

Table 1: Different Approaches Implemented and Scores

4 Conclusion

We investigated methods for data reduction using feature selection. We intelligently removed few fields from the data which improved our ranking in Kaggle competition by a decent range. It is evident that unnecessary features act as noise and degrade the performance of the system. Therefore, it is essential to remove these fields. We tried several supervised classification algorithms to see which one will work best for our dataset. Logistic regression with regularization gave us a logloss that took us to top 100 in Kaggle rankings.

The main challenge that we faced in the project was to find ways to efficiently handle huge data with limited resources. The main advantage of using Logistic Regression with stochastic gradient was that we could do computation by reading the data one line at a time.

We could not implement Support Vector Machines because the output that SVM gives is not a probability and Kaggle needs output in that form. So, we leave the investigation of converting SVM output into probability for future work. We could not implement Random Forest due to infrastructure limitations. We plan to try implementing it in the future.

References

[1] Vowpal Wabbit open tool. `http://hunch.net/~vw/`. Accessed: 2010-09-30.

[2] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[3] Zhipeng Fang, Kun Yue, Jixian Zhang, Dehai Zhang, and Weiyi Liu. Predicting click-through rates of new advertisements based on the bayesian network. *Mathematical Problems in Engineering*, 2014, 2014.

[4] Dejian Lai. Principal component analysis on human development indicators of china. *Social Indicators Research*, 61(3):319–330, 2003.

[5] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer, 1998.

[6] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[7] Steffen Rendle. Scaling factorization machines to relational data. In *Proceedings of the VLDB Endowment*, volume 6, pages 337–348. VLDB Endowment, 2013.