Learning Models for Click-Through-Rate Prediction

Sampoorna Biswas
Department of Computer Science
The University of British Columbia
sambis@cs.ubc.ca

Sam Creed
Department of Computer Science
The University of British Columbia
samcreed@cs.ubc.ca

Abstract

CTR prediction is an important metric for evaluating ad performance in online advertising, and is commonly used in sponsored search and real-time bid applications. In this paper we explore combinations of basic learning algorithms to predict CTR, and compare them against Google's state-of-the-art Proximal algorithm. using logistic loss as our evaluation measure, we find certain results hold over the Kaggle competition (state how algorithms measure up).

1 Introduction

Click-through rate (CTR) prediction is a well studied problem in the online advertisment world [source]. Under common advertisment models, companies only pay for ad placement when a user clicks on the ad. By learning the likelihood of an ad being clicked, companies are more able to target interested users more frequently, and advertisers directly improve their expected profit. Considering that online marketplaces such as Amazon generate billions of dollars in revenue each year, even a 1% improvement to targeting will have a large impact.

One major challenge of CTR prediction is the massive scale of the data involved. In a given day, a large online advertiser may have billions of sample events (e.g. customer views ad, and then does not click on it) to consider [source]. If we are to take advantage of the large amounts of training data available, we are restricted to online algorithms that do not require the entire training dataset to be loaded into memory. Another issue is the unbounded space of the cateogorical features. Any algorithms we choose must be able to handle unknown classes in features, as they occassionally arise in this problem.

The goal of this paper is to explore the performance of common machine learning algorithms (Naive Bayes, Random Forests, SGD) and techniques (Bagging, AdaBoost, Ensemble) to tackle the CTR prediction problem. We also compare these simple algorithms to the state-of-the-art FTRL-Proximal online learning algorithm created by Google for their CTR prediction. Time permitting, we will attempt to use ensemble methods to combine these learning algorithms together to acheive a more accuracte predictor.

To test our results, we will compete in the ongoing Kaggle CTR prediction contest, which provides 11 days worth of Avazu ad click metadata to train our models.

2 Related Work

In this section we will

- identify existing research in this area (at least three sources)
- discuss how our approach differs from existing research or is incomplete (perhaps does not use ensemble methods, or not for this particular application)
-

3 Ensemble Learning

In this section we will give an overview of the different learning stubs used in our ensemble learning algorithm.

3.1 Ensemble Learning

We should also perhaps explain what ensemble learning is.

3.1.1 Stub 1: Naive Bayes?

Description.

3.1.2 Stub 2: Random Forest?

Description.

3.1.3 Stub 3: Neural Network?

Description.

3.1.4 Stub 4: SVM?

Description.

3.1.5 Google Fast Solution Approach

Description. Discuss the fast python solution, maybe use as part of an ensemble, and compare our implementation against it.

3.2 Dataset and Feature Selection

Description. For kaggle competition, discuss train/test data given.

Discuss analysis of features and how we attempt to limit or ignore certain features in the learning process. Maybe something else to compare against with time/quality performance.

4 Experimental Results

In this section we should talk about

- the kaggle competition and the train and test datasets
- comparison in quality of ensemble, fast solution, stubs, and using feature selection
- could possibly compare different loss functions for fun - logistic vs hinge loss.

5 Discussion and Future Work

State main conclusion obtained from the course project. List one strength and one weakness of our contribution. State what we would do with more time.

References

References follow the acknowledgments. Use unnumbered third level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to 'small' (9-point) when listing the references. **Remember that this year you can use a ninth page as long as it contains *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer-Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.