Prediction of Click-through Rates for Ads

Abhishek Vijayan B110158CS
Aswathy Ajith B110759CS
Potturi JayaVenkata Sravani B110903CS
Guided by: Gopakumar G

November 24, 2014

Abstract

The online advertising industry is thriving on the learning problem of predicting click through rates for ads. We attempt to propose efficient machine learning techniques that could be used for predicting click through rates with a high degree of accuracy. So far, we have implemented basic machine learning algorithms in a practice data set to which such techniques had been applied and the results were known to us beforehand. This was required as we were relatively new to the topic.

1 Problem Definition

As part of an online machine learning competition held by KDD Cup, Tencent Corporation released a large data set obtained from the user logs of the search engine soso. The task is to predict the CTR (pCTR) of ads as accurately as possible.

2 Introduction

Click-through rate is a very useful metric for rankinf and pricing ads in the internet marketing world. It is one of the greatest applications of machine learning. Sponsored search advertising, contextual advertising, display advertising, and real-time bidding auctions have all relied heavily on the ability of learned models to predict ad clickthrough rates accurately, quickly, and reliably. A typical industrial model may provide predictions on billions of events per day, using a correspondingly large feature space, and then learn from the resulting mass of data. When a user does a search q, an initial set of candidate ads is matched to the query q based on advertiser-chosen keywords. An auction mechanism then determines whether these ads are shown to the user, what order they are shown in, and what prices the advertisers pay if their ad is clicked. In addition to the advertiser bids, an important input to the auction is, for each ad a, an estimate of P(click — q, a), the probability that the ad will be clicked if it is shown.

2.1 Literature Survey

The evaluation metric used in this contest was AUC(Area Under the ROC Curve). AUC gives the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. We will be using the same metric in our project.
The Area Under the ROC Curve is found to obtain a single scalar value from the ROC. The value ranges from 0 to 1 with AUC equal to 1 being a perfect classifier. A value of 0.5 indicates a random classifier.
AUC is favourable for the following reasons :

- No need to choose threshold, favourable when output is not a probability
- ROC - insensitive to changes in class-distribution
- For skewed class, a classifier that always outputs 0 or 1, will achieve high accuracy. But AUC value will be 0.5

We had done a literature survey on the papers published by the first four winners of KDD Cup 2012.

1. A Two-Stage Ensemble of Diverse Models for Advertisement Ranking This approach achieves 0.8069 AUC on the public test set and 0.8089 AUC on the private test set.
   According to their method, the system can be divided into three stages. They are generating individual models (Different models like Classification models, Regression models, Ranking models, Combined Regression and Ranking models, Matrix Factorization models were used to capture different concepts), blending with the validation set (For increasing the performance, different predictors that were used in first stage were combined) and ensemble learning with the test set (Using Ensemble methods, the blending models are further combined to improve the test set performance)

2. Ensemble of Collaborative Filtering and Feature Engineered Models for Click Through Rate Prediction This is a paper by the team Opera Solutions, who stood second on the leaderboard. Their technique had an AUC Score of 0.80824 on the private leaderboard. The methods adopted include Collaborative Filters, Probability Models, Feature Engineered Models. For directly optimizing the AUC, Collaborative Filters and ANN Models were used. Finally these models were blended using an AUC optimized ANN.

3. Click through rate Prediction for Sponsored Search Advertising with Hybrid Models This solution had secured the 3rd place in KDD cup 2012 track 2. An ensembling is done on a couple of individual methods which include Online Bayesian Probit Regression, Latent Factor Model, Support Vector Machine and Maximum Likelihood Estimation.

4. A Feature Engineering Approach for Click-Through Rate Prediction The three main algorithms used were: Vowpal Wabbit: Used mainly for Bag of words related models, Linear Mixed Effects(LME) : Used for Statistical features creation, Generalized Boosted Models (GBM) : Used for Model training

3 Data Set

3.1 Actual Data Set

The types of data provided include training data, additional data and testing data. There are 149,639,105 training instances and 20,297,594 testing instances.
Each instance contains at least the following information:
UserID, AdID, Query, Depth [number of ads impressed in a session], Position [order of an ad in the impression list], Impression [number of search sessions in which the ad(AdID) was impressed by the user (UserID) who issued the query (Query)], Click [number of times, among the above impressions, the user (UserID) clicked the ad (AdID)] Fields in the training instance:
Click, Impression, DisplayURL [The URL is hashed for anonymity], AdID, AdvertiserID, Depth, Position, QueryID [It is the key of the data file queryid_tokensid.txt], KeywordID, TitleID [This is the key of titleid_tokensid.txt], DescriptionID [This is the key of descriptionid_tokensid.txt], UserID [This is the key of userid_profile.txt]
Besides the details given above, some further details are also provided with additional data files and there are five such files queryid_tokensid.txt, purchasedkeywordid_tokensid.txt, titleid_tokensid.txt, descriptionid_tokensid.txt, userid_profile.txt [composed of UserID, Gender, and Age, delimited by the TAB character. Not every UserID in the training and the testing set will be present in 'userid_profile.txt'.] These data files include information such as gender, age ('1' for (0, 12], '2' for (12, 18], '3' for (18, 24], '4' for (24, 30], '5' for (30, 40], and '6' for greater than 40.)
Along with the training data and additional data, the test set is also given which has the same format as the training dataset , except for the counts of ad impressions and ad clicks that are needed for computing the empirical CTR.

3.2 Practice Data Set

YearPrediction MSD DataSet from the UCI Machine Learning Repository was used to get a hands-on experience working with a relatively large dataset. This dataset was chosen as it was suggested in the Coursera ML course's discussion forum and we could compare it with the accuracy of others.

The task is to predict the release year of western songs from its audio features. The dataset contains 515345 tuples with 90 features. As specified in the dataset description, first 463715 tuples were chosen as training instances and remaining ones as test set.

The first column gives the output value : the year in the range 1922-2011. The 90 attributes are 12 timbre average and 78 timbre covariance taken over all segments, with each segment represented by a 12-dimensional timbre vector.

4 Work Done

4.1 Algorithms Implemented

We have implemented basic algorithms on a practice dataset.
The code is available in our github repository https://github.com/ABHIVIJ/MLPractice.

4.1.1 Linear regression using Gradient Descent method

Linear regression involves predicting a real-valued output y, by modelling it with a linear function with the independent variables as the input features (represented by the vector X) . So the algorithm is essentially determining the coefficients of the linear function.

Suppose the regression coefficients are $a_0, a_1, a_2, ...a_n$ and the independent variables be $x_1, x_2, ...x_n$, then the linear function also called as hypothesis(h) that is used to predict the dependent variable y is of the form

$h(x) = a_0 + a_1x_1 + a_2x_2 + ... + a_nx_n$

The cost function (J) used here is the squared error function which is:

$J(a_0, a_1x_1, a_2x_2, ..., a_nx_n) = (\frac{1}{2m})\sum_{i=1}^{m}(h(x^{(i)})\text{-}y^{(i)})^2$
where m is the number of training examples.

Suppose, $a = (a_0, a_1, a_2, ...a_n)$ be the vector representation of the regression coefficients.
To get a good hypothesis, the cost function should be minimized. Two ways to obtain the values of the regression coefficients that minimize the Cost function are:

1. Normal Equation Method
2. Gradient Descent Method

Normal Equation Method Normal equation method is an approach for calculating the regression coefficients that minimize the Cost function using Linear Algebra. This method is not efficient when the number of independent variables (or features) is large. Here, in the practice data set, the number of features is 90. So, the Normal equation method is not efficient for this problem.

Gradient Descent Method Gradient Descent is used to find the local minimum of a function.

Algorithm Repeat (until change in the cost function in successive repetitions is very less){
$a_j = a_j - \alpha\frac{\partial J(a)}{\partial a_j}$; (simultaneous update of all $a_0, a_1, a_2, ...a_n$)
}

where α is the learning rate , $\frac{\partial J(a)}{\partial a_j} = (\frac{1}{m})\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})x_j^{(i)}$ and $x_0^{(i)} = 1 \; \forall$ i = 0, 1,2.....m.
If α is too small, Gradient Descent algorithm will be too slow to converge. If α is too large, cost function J(a) may not decrease with each iteration and Gradient Descent may not converge.
For Gradient Descent to converge faster, Feature Scaling can be done. Feature Scaling is the approach used to get every feature into an approximate range of -1 to 1.

4.1.2 Polynomial Regression Analysis using Gradient Descent

In Polynomial Regression, a nonlinear function with the independent variables as input features (represented by the vector X) is used to model the dependent variable say y. Here, we added polynomial functions of the original features and performed linear regression on the new set of features.

4.2 Majority Prediction

For setting a benchmark to the accuracy, a simple model which predicts the majority year as the output for all tuples was implemented. This gave better results than the others.

4.3 Logistic Regression

Logistic Regression is a classification problem. By using Logistic Regression, a discrete valued output is predicted. Given a data set containing the training examples $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ...(x^{(m)}, y^{(m)}))$, then model or hypothesis (h) trained with Logistic Regression algorithm predicts the class to which the test example belongs to.

In Logistic Regression, $0 \leq h(x) \leq 1$
Here, the hypothesis used is $h_\theta(x) = g(\theta^T x)$ where x is the feature vector, θ is the vector representation of the logistic regression coefficients and g(b) $= \frac{1}{1+e^{-b}}$

In simple terms, when there are only two classes then h(x) is the probability that y(dependent variable or outcome) = 1 given the value of x. And (1-h(x)) is the probability that y = 0 given the value of x.

When there are more than two classes, the hypothesis that is to be used is $h_\theta(x^{(i)})$ = p (y = i : x, θ)

and the class predicted is the class that maximizes the output of the hypothesis function i.e., $max_i\ h_\theta(x^{(i)})$.

Regularization is a technique generally used when there is an overfitting of data. Suppose $J_R(\theta)$ represents the Logistic Regression Cost function when Regularization is used, $J(\theta)$ represents the Logistic Regression Cost function, then

$J_R(\theta) = J(\theta) + (\frac{\lambda}{2m})\sum_{j=1}^{n}\theta_j^2$
Where λ is the Regularization Parameter

4.4 Artificial neural networks

Artificial neural networks try to mimic the functioning of biological neural systems in order to train the machine to predict the behaviour of data. Like people, ANNs learn by example. It is composed of simple processing units called nodes which are capable of processing the input data and classifying them into categories.
Advantages of using neural networks include :

- Adaptive Learning: They can learn how to do the task of classifying using the training data.
- Self-Organization: It can create its own representation of the information given during training
- Real time operation: ANN computations can work in parallel.
- Fault tolerance via redundant information coding: Partial destruction of the network will degrade the performance but there won't be major network damage.

Architecture The most commonly used types of neural networks are feed-forward, feedback, radial basis function network, Kohonen self-organizing networks, etc. We have used the classic feed-forward architecture to implement the ANN model. The characteristics of feed-forward networks are as follows:

1. Nodes are grouped into a specific number of layers with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external environment.
2. Each node in one layer is connected to every node on the next layer. The information is constantly "fed forward" from one layer to the next.
3. There is no connection among nodes in the same layer.

The neural network is formed in three layers: the input layer, hidden layer and output layer. The input layer consists of 90 nodes as there are 90 features given in the data set and 90 output layer nodes, each representing 90 possible years that could be possibly predicted.

The hidden layer is free to create its own representation of the input. The number of units in the hidden layer should neither be too low as it would result in underfitting nor too high as that may lead to overfitting and will require a long time to train.

As there is no hard and fast rule of thumb to determine the exact number of hidden layer units that will give the most accuracy, we have tried testing with different number of units greater than the number of input layer units.

We have used the sigmoid function as the activation function.

Algorithm

1. Parameters θ_1 and θ_2 are randomly initialized with values in the range -ϵ to +ϵ
   where $\epsilon = \sqrt{\frac{6}{No.of input layer units + No.of output layer units}}$.
2. Obtain the prediction $h_\theta(x^{(i)})$ for each tuple x(i) in the training set using forward propagation.
3. Compute the cost function $J(\theta)$.
4. Compute the partial derivatives
   $$\frac{\partial J(\theta)}{\partial \theta_{jk}^{(l)}}$$
   where l is the layer and j, k are the indices of θ^L matrix using backpropagation.
5. Minimize $J(\theta)$ using advanced optimization techniques which are directly taken from the exercises of the machine learning course in Coursera.

5 Results

Results for Linear regression Training data accuracy: 5.218507, Test data accuracy: 5.272129

Results for Polynomial regression

Features added	Training Cost	Training Accuracy	Testing Cost	Testing Accuracy
Squared features	44.690954	5.413454	44.676964	5.413519
Squared, cubic	44.012999	5.540256	44.336964	5.636258
Cubic features	44.50686	5.491304	44.587195	5.489057
Cubic, product of adjacent pairs, product of adjacent triples	44.115562	5.570879	44.343716	5.502615
Cubic, fifth power, product of adjacent pairs, product of adjacent triples	43.753159	5.672018	45.873208	5.655627

Results for Majority Prediction Training Set Accuracy:7.628608
Test Set Accuracy: 7.803603

Results for Logistic Regression

λ	Iterations	Training Accuracy	Testing Accuracy
0	50	8.837325	8.037962
0	100	8.842285	8.018594
0.01	50	8.839481	8.028278
0.01	100	8.842716	8.018594
0.03	50	8.835384	8.020531
0.1	50	8.840128	8.03215
0.3	50	8.843363	8.028278
1	50	8.833012	8.057331
3	50	8.834521	8.036026
10	50	8.817269	8.068952
30	50	8.792901	8.039899
30	100	8.814466	8.045710
100	50	8.759044	8.059268
1000	50	8.552451	7.921751

Results for ANN

hidden_layer_size	Iterations	λ	Cost	Training Accuracy	Testing Accuracy
90	20	1	4.403	7.457	7.521
90	100	1	4.187	8.772	8.014
100	20	1	4.499	6.922	6.918
100	100	1	4.189	8.716	8.106
100	120	1	4.183	8.798	8.022
100	100	5	4.174	8.846	8.223
100	100	10	4.169	9.003	8.419
120	90	5	4.535	7.119	7.042
120	100	5	4.468	7.254	7.193

6 Proposed Method

We are proposing to improve the newly found AUC by different means by varying the parameter values, deriving new features by combining existing features and trying out alternate combinations of basic algorithms for blending.

7 Future Work

We will try other basic ML techniques in a random subset of the actual KDD Cup dataset with about 10,00,000 tuples. As part of this, tools like Octave, R, Python, Vowpal Wabbit will be used and effectiveness of each tool will be compared. An efficient method to find Area Under ROC Curve should be found. We have to make the algorithm more scalable after gaining a clear understanding about Hadoop, MapReduce. We also need to set up a cluster as it has been found that adding new features to the currently used practice dataset itself goes beyond the 4GB RAM in our machines. We need to divide the whole training data into training, validation and test data efficiently, by following some suitable metric and by using cross validation, random sampling, K-validation. In this divided dataset, we plan on implementing the algorithms in the papers by KDD Cup winners and calculate the Area Under ROC with each of these approaches.

8 Conclusion

We have implemented the basic algorithms like Linear Regression, Polynomial Regression, Logistic Regression and Artificial Neural Network on a practice data set. And ANN seems to the best on this practice data set (test set). Presently, we are working on getting the subset of the original data set of our problem that represents the original data set so that first we can work on that subset in our local machine. In the next semester, we are planning to work on the complete data set using some parallel processing techniques.

9 Bibliography

1. Machine Learning by Andrew Ng, https://class.coursera.org/ml-006
2. NEURAL NETWORKS by Christos Stergiou and Dimitrios Siganos, http://www.doc.ic.ac.uk/ nd/surprise_96/journal/vol4/cs11/report.html
3. Types of artificial neural networks, http://en.wikipedia.org/wiki/Types_of_artificial_neural_networks
4. Neural Networks, http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html
5. KDD Cup 2012, https://www.kddcup2012.org/c/kddcup2012-track2
6. YearPredictionMSD Data Set, https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD
7. Million Song Dataset, http://labrosa.ee.columbia.edu/millionsong/