CS725 Data Mining Contest Report
Team *Bast*

09005050	09005056	09005057	09005068
Bhanu Prakash	Avinash Teetla	Sai Teja Pratap	Tarun Gujjala

November 9, 2012

1 Abstract

The data set being very big it would take really long time to train with state-of-art classifiers like SVM. Moreover as most of the attributes are categorical, we need to create one dimension for each possible value of the category for using classifiers which learn decision boundaries . We did not find any ML libraries which processes sparse data effectively.

The first implementation was using the naive approach, hacking with the probabilities. Next we tried the most natural way to train categorical data, Tree Regression . As the results were not as expected we then implemented Linear regression. As number of dimensions are huge to try SVMs we made an attempt to reduce the dimensions using Principal component analysis(PCA) which eventually ended up in vain.

2 Data Preprocessing

The training samples with result 'skipped' and 'time outputs' were ignored as they would contribute negatively to the classifier decision. Only certain features were considered which are considered to be important. These are:

1. user_id
2. group_name
3. question_type
4. track_name
5. subtrack_name
6. game_type
7. num_players

Apparently, the other features seemed to add noise to the classifier and increase the training time and hence were ignored.

3 Feature Engineering

We attempted the following feature engineering steps:

1. After some experiments we realized that track_name name to group_name mapping is unique and removed the track_name from the list of features to learnt with. Similarly we found out that subtrack_name implies a single track_name and removed it from the features if subtrack_name is used.

2. For boundary based classifiers we replaced the user ids with the correct to wrong answer ratio of the user.
   This did not yield good results so we added two features which represent the proficiency of the user in English and math/science respectively in similar manner.

3. We also replaced the category values for features like track name with a sparse set of features Ex: track name = 3 corresponds to a features [0 0 0 1 0 0 0 0] (assuming track name can take values from 0 to 7).

   Reason for doing this is that the boundary based classifiers learn a boundary which can be represented as a function (f:X-¿y) and based on the value of f(x) the class id is determined. High values of f(x) implies positive example with great confidence and the reverse applies for negative examples. This clearly does not work for categorical attributes.

4 ML Algorithms

4.1 Probability based approaches

We maintained the number of correct and wrong answers for each user and for each sub track, track, game type (three different dictionaries). Thus we have the probability of the user getting a question correct given (user,track), (user,sub-track)(user,question type), (user,game type) individually. Each of them can be considered as a separate classifier.

For a given test input, the model gives the mean of all the probabilities. Inspired by Naive Bayes Classifier we tried the product of probabilities also which failed miserably implying that the data features are not independent.

4.2 Decision Tree based approaches

We ran decision tree with features mentioned in the data preprocessing section (point 1) and after some experimenting, removed the redundant features in the data. We tried both regression and classification.

4.3 Linear Regression

We tried Linear Regression with features as mentioned in data preprocessing section(points 2,3). Replacing the user id with proficiency in English and Math performed better than replacing the user id with the corresponding success probability.

5 Out-of-Class Efforts

The results for the methods above are not great. So we thought of using the tags. But learning any model with tags seems impossible on our laptops. The program led to auto shutdown of our laptops because of heat due to overload on the laptop. So we thought of reducing the number of tags from 300 to 20. This seemed reasonable because all the questions can be more or less classified to 20 categories. But we were never able to test our idea. (details in Unfinished efforts)

6 Code Structure

For each approach we tried , we have a python program which implements it. We used sklearn library (python) for linear regression and Tree regression,classifier.Each of them would take around 2 minutes to complete training and produce the test outputs.All these programs need two arguments the train and test data locations respectively.

7 Unfinished Efforts

We made an attempt to reduce the number of tags using Principal Component Analysis and implemented that part. But we did not test this as the other bits and pieces to complete the learning with the reduced dimensions is not ready.Also we did not experiment to find out number of dimensions to reduce.