eYSIP-2015

Auto-Tagging Online Selection Test

Intern
Shubham Gupta
suvam8694@gmail.com
+91 83 760 27800

Project Mentor
Mr. Amiraj Dhawan
amirajdhawan@gmail.com
+91 99 206 78775

Duration
05/06/2015 to 17/07/2015

Contents

Abstract

While Moore's Law makes machines faster, Machine Learning makes them smarter. Machine Learning is an active research area which has been successfully applied for solving a lot of challenging and practically significant problems. Here we have demonstrated two such applications of Machine Learning principles to solve two challenging problems that were presented to us. Our first problem, the *Auto-Tagging Problem* was concerned with automatically deducing the difficulty level of questions used in the selection test for eYRC 2014 based on performance of students. The second problem, the *Performance Prediction Problem* aimed at finding correlations between the profile of a student and his/her performance, and hence using the model obtained to predict the performance of students based on their profile.

1 Objectives

eYantra conducts an online test to select student teams for eYRC. The questions used for eYRC-2014 are manually assigned a difficulty level prior to the commencement of the test based on experience of question maker. Since there are a lot of questions and many different question makers, the manually assigned difficulty levels are not reliable due to the *relative nature* of difficulty of questions.
There are two main objectives:

1.1 Auto-Tagging

Using the data generated during eYRC-2014 Online Selection Test, the task is to determine the accuracy of manually assigned difficulty levels and also suggest corrections to them based on performance of students.

1.2 Performance Prediction

This task involves finding correlations between various aspects in the profile of a student and performance of the student and hence use the deduced machine learning model to predict the performance of a student in the future iteration of the same test given his/her profile.

2 Completion Status

Both the tasks have been completed successfully. Four different learning algorithms were implemented to solve task 1 and five different learning algorithms were implemented for solving task 2. We have also experimented with various features for both these tasks.

3 Results and Discussions

We have discussed the results obtained for both the tasks separately.

3.1 Auto-Tagging Problem

A variety of different machine learning algorithms were applied to solve the *Auto-Tagging* problem. Many combinations of features and algorithms were considered. Some of the main ones have been described here:

3.1.1 Neural Network

Both complete and sparse neural networks were implemented. This was the first approach that was used. Here the problem was viewed as a supervised learning problem where the assumption was that majority of the manually assigned tags are correct. Under this assumption it was predicted that the network will generalize well and automatically rectify the incorrect tags which are present in minority. It was discovered later that this assumption was wrong. This approach was tried with different combinations of features but the results were unsatisfactory.

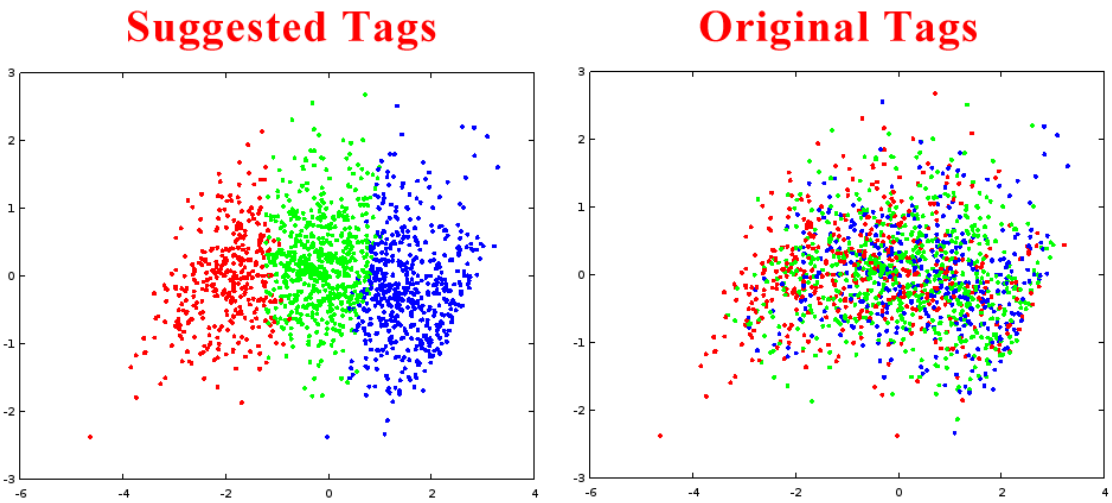
Estimated Manual Tagging Accuracy: 52.45%

3.1.2 k-Means Clustering

k-Means Clustering is an unsupervised learning algorithm used for clustering the data into semantically sound clusters. This algorithm was used with many different combination of features. The most satisfactory output was obtained by using the following three features:

1. Fraction of people who solved a question correctly.
2. Fraction of people in top five percentile who solved the question correctly.
3. Average marks of people who did not attempt the given question or solved it incorrectly.

This output has been shown below:



Feature	Level	Min	Q1	Median	Q3	Max	Mean
Feature 1	**Easy**	-0.515	0.842	1.370	1.818	2.934	1.318
	Medium	-1.359	-0.470	-0.106	0.373	1.813	-0.031
	Hard	-1.686	-1.213	-0.946	-0.605	1.106	-0.860
Feature 2	**Easy**	-0.342	0.962	1.159	1.159	1.159	1.018
	Medium	-0.842	-0.018	0.409	0.730	1.159	0.377
	Hard	-1.843	-1.843	-1.092	-0.642	0.159	-1.139
Feature 3	**Easy**	-4.498	-1.631	-1.091	-0.623	1.104	-1.134
	Medium	-1.667	-0.339	0.028	0.479	2.620	0.093
	Hard	-2.050	0.159	0.589	1.132	3.380	0.662

Table 1: Statistics for k-Means Output

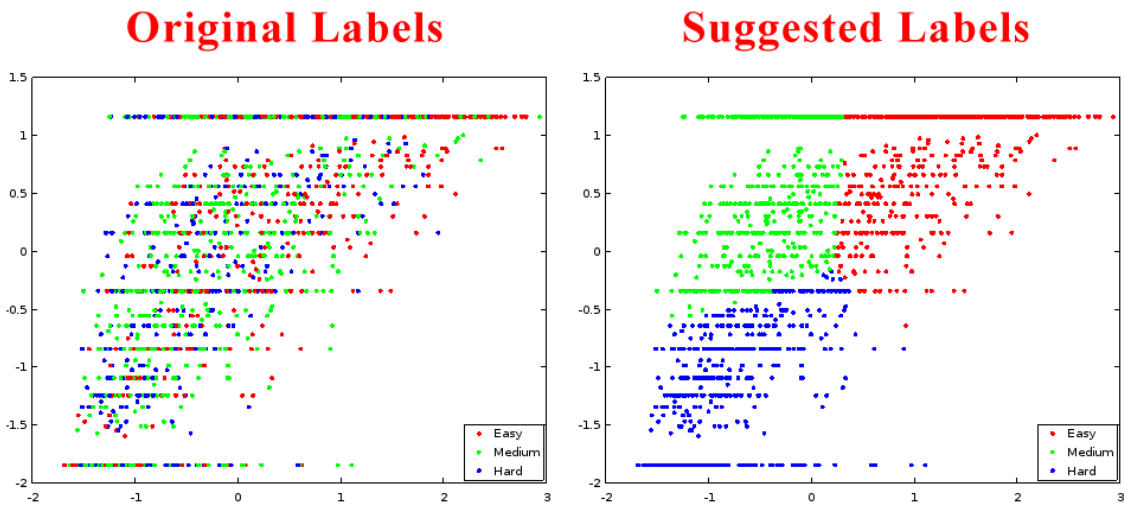
Estimated Manual Tagging Accuracy: 43.73%

3.1.3 Competitive Learning

Competitive Learning is another unsupervised learning algorithm which can be used for clustering of data. This algorithm can be implemented as a neural network. Autoencoder was used to encode features to feed input to the algorithm. Following two features were fed to autoencoder:

1. Fraction of people who solved a question correctly.
2. Fraction of people in top five percentile who solved the question correctly.

Best results were obtained when only these two features were used, although other feature combinations were also tried. This algorithm is also faster than k-Means clustering. Output has been shown below:



Feature	Level	Min	Q1	Median	Q3	Max	Mean
Feature 1	**Easy**	0.234	0.652	1.051	1.580	2.934	1.169
	Medium	-1.503	-0.695	-0.419	-0.096	0.324	-0.412
	Hard	-1.686	-1.211	-0.886	-0.414	1.106	-0.781
Feature 2	**Easy**	-0.642	0.492	1.159	1.159	1.159	0.804
	Medium	-0.556	-0.042	0.409	0.859	1.159	0.392
	Hard	-1.843	-1.843	-1.092	-0.717	-0.175	-1.185

Table 2: Statistics for Competitive Learning Output

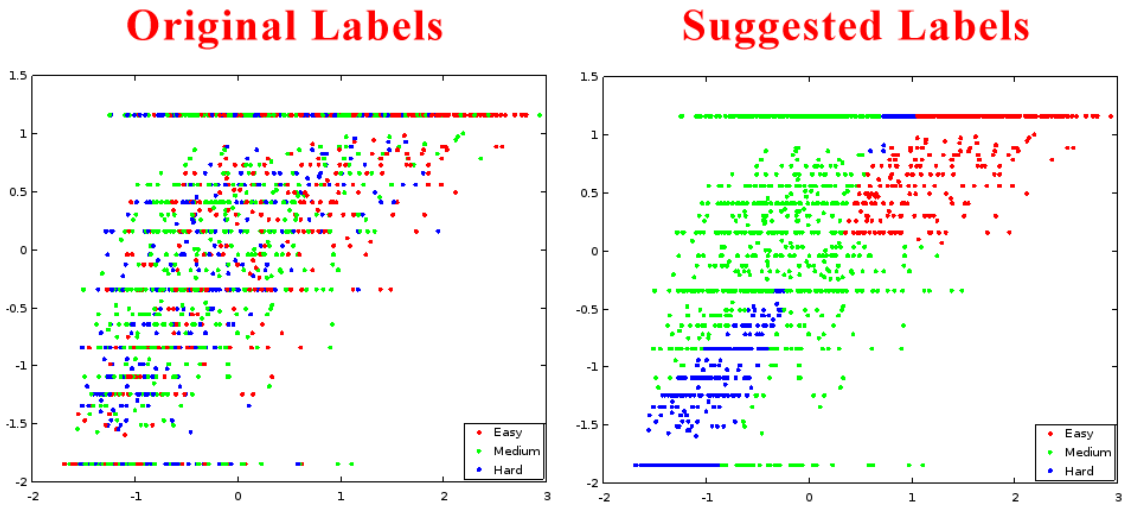
Estimated Manual Tagging Accuracy: 43.05%

3.1.4 Expectation Maximization Algorithm

Expectation Maximization algorithm can be used for *soft clustering.* of the data. For this algorithm, a mixture of gaussians model was assumed. From the obtained soft clustering output, question labels were predicted by taking most likely tag for each question. The following features were used:

1. Fraction of people who solved a question correctly.
2. Fraction of people in top five percentile who solved the question correctly.

Best results were obtained when only these two features were used, although other feature combinations were also tried. This algorithm provides a skewed output forcing many questions to fall in the medium difficulty level category. Output has been shown below:



Feature	Level	Min	Q1	Median	Q3	Max	Mean
Feature 1	**Easy**	0.355	0.986	1.412	1.824	2.934	1.415
	Medium	-1.519	-0.671	-0.265	0.137	1.485	-0.271
	Hard	-1.686	-1.250	-0.979	-0.576	1.031	-0.762
Feature 2	**Easy**	0.068	0.559	0.971	1.159	1.159	0.846
	Medium	-1.843	-0.342	0.159	0.659	1.159	0.081
	Hard	-1.843	-1.843	-1.242	-0.717	1.159	-0.983

Table 3: Statistics for EM Output

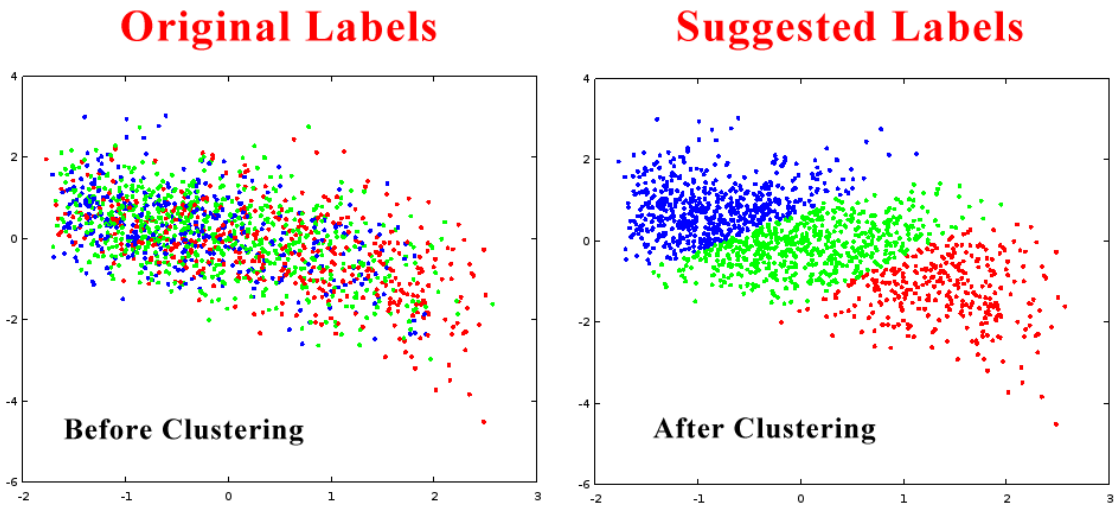
Estimated Manual Tagging Accuracy: 45.40%

3.1.5 Weighted Clustering

The core idea behind using weighted clustering is to allow students who have got higher scores to contribute more towards clustering as compared to those students who have got lower scores. The rationale behind using such a strategy is that students with higher scores give more authentic data containing fewer random guesses. The following weighted features were used with k-Means Clustering to achieve weighted clustering:

1. Fraction of people who solved a question correctly adjusted by weight.
2. Average weight adjusted marks of students who did not attempt a given question or solved it incorrectly.

Weighing of features is used by assigning higher weights to students who have got higher scores and vice versa. This in effect gives rise to a weighted clustering algorithm without changing the core algorithm. One difficulty is estimating the accuracy of suggested tags since now we are no longer dealing with original features, hence result analysis has been done using original features from k-Means Clustering.



Feature	Level	Min	Q1	Median	Q3	Max	Mean
Feature 1	**Easy**	-0.515	0.907	1.424	1.851	2.934	1.380
	Medium	-1.504	-0.447	-0.013	0.483	2.165	0.040
	Hard	-1.686	-1.188	-0.883	-0.506	1.485	-0.802
Feature 2	**Easy**	-1.843	1.159	1.159	1.159	1.159	1.020
	Medium	-1.843	-0.183	0.409	0.826	1.159	0.238
	Hard	-1.843	-1.468	-0.842	-0.342	1.159	-0.818
Feature 3	**Easy**	-4.498	-1.733	-1.127	-0.731	0.481	-1.227
	Medium	-1.850	-0.494	0.136	0.198	1.979	-0.137
	Hard	-1.121	0.379	0.741	1.238	3.380	0.822

Table 4: Statistics for EM Output (Using Features from k-Means Clustering)

Estimated Manual Tagging Accuracy: 43.18%

3.2 Performance Prediction Problem

Different algorithms were implemented for solving the *Performance Prediction Problem*. Four of these were classification algorithms and one was a regression algorithm. The results obtained have been described below, beginning with a description of features that have been used:

3.2.1 Features

Following features were used for training various algorithms:

1. Year in college. For example, 3^{rd} year student
2. Branch. For example, Computer Science and Engineering
3. Gender
4. State to which student belongs
5. Region to which student belongs
6. College to which student belongs
7. Role of Student. Team Leader or Team Member

It is to be noted that all these are categorical features Further there are different number of categories possible for each of these features. In order to deal with categorical features we have adopted the widely popular technique of **feature binarization**.

In feature binarization each categorical feature is turned into a vector of binary features. One binary feature is present per category and this binary feature tests whether the given example belongs to that category or not.

In order to deal with missing category information and unseen category values in the test set, a separate category called *other* is included with each feature.

3.2.2 Linear Regression

Linear Regression is often viewed as the simplest machine learning algorithm. Despite being very simple, this is one of the most powerful learning algorithm. If we want to predict the exact marks that a student might get instead of assigning a class to each student, linear regression is the perfect candidate.

All the features described in [Features section 3.2.1] were used. The mean square error per test example was found to be 68.53 when 5% of the data was used as test data. This translates to an error of 8.28 marks per test example. It was also found that the average standard deviation from mean is 8.405 even when all the features except gender and role are same. Hence the results of linear regression are satisfactory given the data that we had.

3.2.3 Support Vector Machine

Support Vector Machines or SVMs are considered to be the most effective off-the-shelf classifiers. To use a classification algorithm like SVM, we divided the marks obtained by various students into various class intervals and assigned the corresponding class to each student. Thus the goal was to predict the class to which a student belongs.

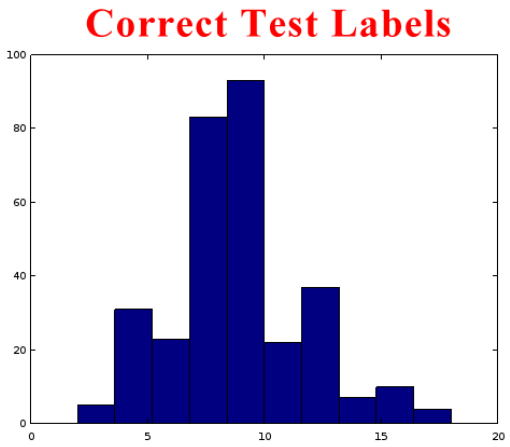
A highly optimized software library for training and using SVMs called **libsvm**[2] was used. The training time of this algorithm was greater than that of any other algorithm that has been implemented for *Performance Prediction Problem*.

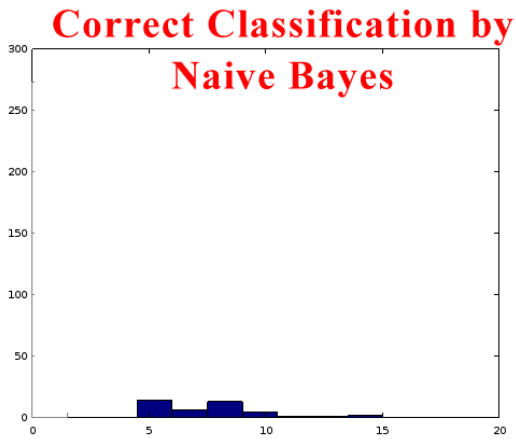
On the test set, an accuracy in the range of 10% to 13% was obtained when 5% of the data was used as the test set. The output was strange in the sense that SVM predicted that all the students in the test set belong to class 9, which is the mean class. Even with such an extreme output, we obtained better accuracy than what we will get if we randomly assign classes.

This output serves as a control. By using a widely used off-the-shelf library we ensured that there is no bug in the implementation of SVM. Thus the strange output signifies that classification is very difficult given the current data. The correlation between the profile of student and his/her performance is very low, not enough to predict anything else except the mean class.

3.2.4 Naive Bayes

Naive Bayes algorithm is an effective algorithm that is used for probabilistic classification. It is based on Bayes' rule. This algorithm makes an underlying assumption of conditional independence which is known as the Naive Bayes assumption.





The problem was turned into a classification problem by dividing marks obtained into class intervals as it was done for Support Vector Machine. One important detail that separates this algorithm from other classification algorithms that have been implemented for *Performance Prediction* problem is that we have not used feature binarization. Since fea-

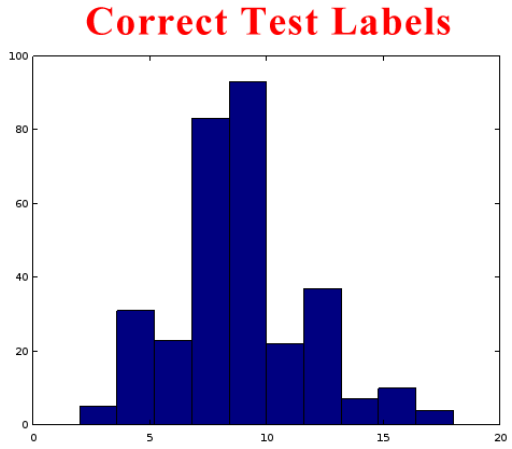
ture binarization was the bottleneck of performance for other algorithms, this algorithm runs much faster.

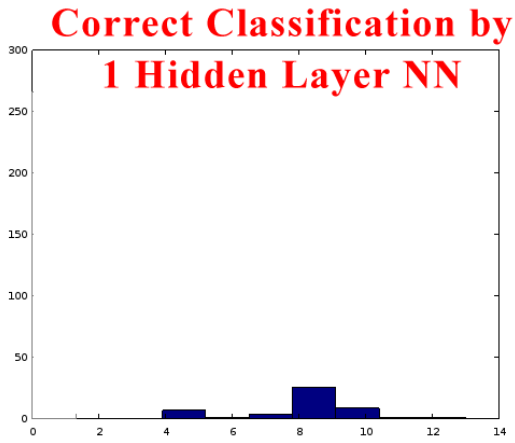
Accuracy on Test Set: 13% - 18%

It was also observed that while other classification algorithms tend to predict only those class labels which are closer to mean, this algorithm gave a more distributed and realistic class label output. It was not selected as the final algorithm because Neural Networks were providing a higher accuracy in general.

3.2.5 Neural Network with Single Hidden Layer

Neural Networks can be used as very powerful non-linear classifiers. Two different neural networks were implemented with different number of hidden layers. This section describes the neural network with a single hidden layer.





After formulating the problem as a classification problem as it was done in case of SVMs, the single layer neural network was made. It had 148-151 input units depending on the output of PCA, 35 hidden units and 20 output units, one for each class.

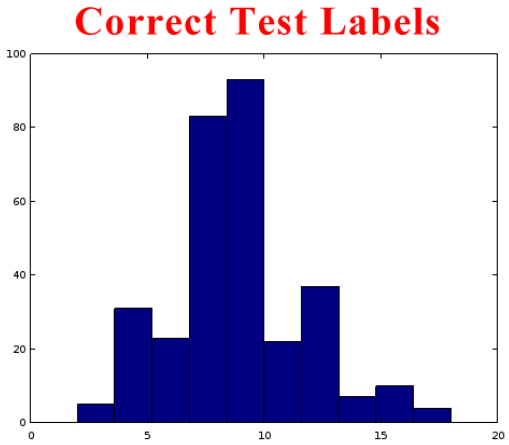
Accuracy on Test Set: 15% - 20%

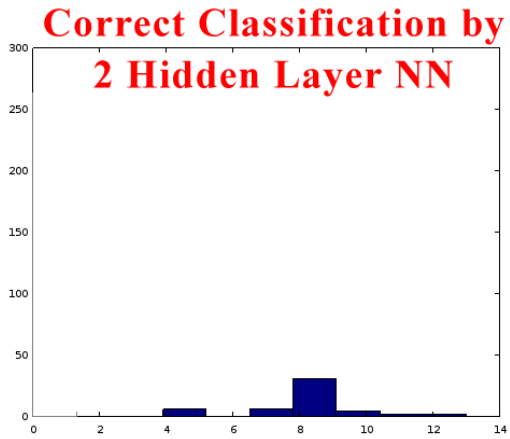
This output was obtained when 5% of the data was used as test set and 99% of variance was retained by PCA. Accuracy increases up to 48% if we allow an error of one neighboring class during classification. This implementation gives a balance between accuracy and complexity, hence this was chosen as the final algorithm.

3.2.6 Neural Network with Two Hidden Layers

Another neural network with two hidden layers was implemented to as an attempt to increase the accuracy by making the model more complex. This two hidden layer neural network has been described in this section.

After formulating the problem as a classification problem as it was done in case of SVMs, the two layer neural network was made. It had 148-151 input units depending on





the output of PCA, and two hidden layers containing 25 hidden units each. The output layer consisted of 20 units, one for each class. The total number of free parameters in this model, i.e. the total number of weights that can be changed is about 4900, hence this network is more prone to overfitting.

Accuracy on Test Set: 15% - 20%

This output was obtained when 5% of the data was used as test set and 99% of variance was retained by PCA. In general, the accuracy of this network is more than the accuracy of the neural network with a single hidden layer but the difference is not much (approx. 1%). Since this model increases the complexity of the model but not lead to any significant increase in accuracy, it is advisable to use the single hidden layer neural network.

4 Known Bugs

1. plotData function in all algorithms crashes if only 1 feature is chosen.
2. Only 1489 questions out of 1500 have been classified. For the remaining questions:
   (a) Two questions were solved by all people and hence are definitely easy. These 2 questions are:
      i. 199
      ii. 1350
   (b) Eight questions were solved by none and hence are definitely hard. These 8 questions are:
      i. 206
      ii. 377
      iii. 378
      iv. 1029
      v. 1038
      vi. 1143
      vii. 1257

viii. 1386

(c) One question was never presented to any person in top 5 percentile. (This question has been classified by weighted clustering)

3. The regression algorithm using Neural Network has a very poor convergence rate. The gradients generated are very small which cause the algorithm to stop before convergence.

5 Future Work

Some of the proposed projects that can be built using the results of this projects are as follows:

5.1 Suggesting Study Material

If every question is assigned to one of the categories, then we can reverse the process to classify students based on the type of questions that they have solved. This will help in suggesting appropriate study material to students. For example some students did very well in calculus and performed poorly in complex numbers, then they should be suggested the study material focused on complex numbers.

5.2 Noise Removal

Once correct difficulty tags are known, this information can be used to detect random guesses made by the students which will reduce noise in data obtained from subsequent iterations of the test. Now this filtered data can be used to further improve the accuracy of assigned difficulty levels. This is just like Expectation Maximization algorithm spread in time. By detecting random guesses an improved performance estimate can be made for each student which can be used for selecting students instead of using the raw score.

5.3 More Intelligent Performance Prediction

One major problem with performance prediction was non-availability of relevant data. The correlation between available data and performance of students was very poor. By gathering data pertaining to the academic performance of students in the past, a more intelligent Performance Prediction system can be designed.

References

[1] GNU. *Octave*. URL: http://www.gnu.org/software/octave/.

[2] *libsvm*. URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[3] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*.