Predicting student's learning performance with KDD Cup 2010

Binbin Xiong
Information Networking Institute
Carnegie Mellon University
Pittsburgh, PA 15213
bxiong@andrew.cmu.edu

Yujing Zhang
Information Networking Institute
Carnegie Mellon University
Pittsburgh, PA 15213
yujingz1@andrew.cmu.edu

Abstract

KDD Cup 2010 is an educational data mining problem of practical importance. It is to learn a model from students' past behavior and then predict their future performance. We extracted five features: *student ID*, *problem name*, *step*, *knowledge component*, and *opportunity count* from the dataset to predict the binary value of *correct first attempt*. Then, we implemented several classifiers including Naive Bayes, Logistic Regression, and SVM as our training models. In addition, we used AdaBoost to improve the weak learning algorithms, and tried some dimension reduction methods: tokenization, k-means and PCA, to optimize feature structure. At last, we compared each classifier with accuracy and Root Mean Squared Error to analyze the results. Experiment result shows that our method can achieve more than 80% accuracy with less than 0.44 RMSE[1].

1 Background

What is a student's learning generality and speed? What is the difference and similarity between problems How can we define the difficulty degree of a problem? These problems might be solved by analyzing the knowledge concepts and skills each problem needs.

1.1 Problem Description

In this report, we are tackling the problem of student learning process. We predict the student learning performance on mathematical problems given information regarding past performance, with the 2010 Knowledge Discovery and Data Mining competition dataset. This problem is not only a technical topic for researchers, but also of practical importance.

1.2 KDD 2010 Data Set Description

The dataset is divided into a training set and a test set. Four key terms form the building blocks of the dataset. These are problem, step, knowledge component, and opportunity. A problem is a task for a student to perform, and it typically involves multiple steps, which are observable parts of the solution to a problem. A knowledge component is a piece of information that is used to solve a step. It can be concept, principle, fact, skill, etc. Each step can be labeled with one or more hypothesized knowledge components. And every knowledge component is associated with one or more steps. Finally, an opportunity is a chance for a student to demonstrate whether he or she has learned a given knowledge component. In this report, we predict the binary value of "correct first attempt" as student-step performance.

2 Related Works

In[2], the authors points out that feature extraction is playing a vital role in this problem. In addition, as the student is learning during the tutoring process, temporal information is of great importance. Their work can be separated

[1] With Algebra I 2005-2006 dataset

into two parts: feature extraction and classification. For feature extraction, they use two techniques: sparse features generated by expanding a multi-category feature into a set of binary features, and condensed features using feature combinations to indicate some relationships. For classification part, they tried several machine learning techniques to train the learner. They use Logistic Regression for sparse feature set, and use Random Forest, AdaBoost for dense feature set. Finally, they combined all results from different classifiers by regularized linear regression, and won the first place of the competition.

The final prediction of the 2nd team[3] was a combination of Bayesian Hidden Markov Models (HMMs) and bagged decision tree.Their HMM model learns student specific parameters and then used them to train skill specific models. It was to predict the probability of knowledge for each student at each opportunity and the probability of correctness on each step. Then they used the prior skill data learned to generate features for the test sets, and implemented Random Forest to do the prediction.

The team of 3rd place[4] used Singular Value Decomposition (SVD) technique, and won the third place. Based on the idea of SVD model, they tried several factor models in order to correct the shortcomings of pure SVD. They introduced many biases into the model, like a step dependent bias, a student dependent bias, a KC dependent bias, a student and KC dependent bias, etc. In addition, they used KC dependent N dimensional feature vectors. Both of the bias and feature vector methods significantly lowered the value of RMSE.

3 Learning Model

According to implicit relationship among data fields, we extract several key features and build a feature model. Then we use both discriminative and generative models including Naive Bayes, Logistic Regression, and SVM to learn the *Algebra I 2005-2006*[2] data set. In order to improve learning performance, we further implemented AdaBoost and several feature reformation technologies. We also evaluated the performance of dimension reduction techniques like k-means and PCA. Experiment result shows that both AdaBoost and our feature extraction methods effectively improve the classification accuracy.

3.1 Feature Selection

There are 19 columns in total in the dataset, but some of them are irrelevant to our prediction, like row number. We extract some important features to build the learning model. Different problems and knowledge components have different difficulty degrees. Some are easy to solve, but others are hard. In addition, the more times a knowledge component the student meet, the easier the step for the student. And different students have different learning rate. So we decide to use *student ID*, *problem name*, *step*, *knowledge component*, and *opportunity count* as five features of our learning model. In addition, we use the value of *corret first attempt* as student-step performance predictor.

3.2 Naive Bayes

Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes Theorem with conditional independent given class label. Assume $X = \{x_1, x_2, ..., x_n\}$ is a collection of features, and $Y = \{1, 0\}$. Each x_i is a discrete feature, and $x_i \in X$ is independent given Y. Then:

$$P(Y = y \mid x_1, x_2, ..., x_n) = \frac{P(Y = y) \prod_{i=1}^{n} P(x_i \mid Y = y))}{\sum_{k=0}^{1} P(Y = k) \prod_{i=1}^{n} P(x_i \mid Y = k)}$$
$$Y \leftarrow \arg\max_{y_k} P(Y = y_k) \prod_{i=1}^{n} P(x_i \mid Y = y_k) \tag{1}$$

Naive Bayes classifier can be trained in linear time $O(nN)$, rather than by expensive iterative approximation as used for many other types of classifiers. Applying MAP estimation in Naive Bayes is necessary to avoid overfitting and extreme probability values.

3.3 Logistic Regression

Logistic Regression is also a probabilistic statistical classification model. For binary classification:

[2]Dataset can be downloaded at https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

$$P(Y=1 \mid X) = \frac{1}{1+exp(w_0+\sum_{i=1}^{n} w_i X_i)} \quad P(Y=0 \mid X) = \frac{exp(w_0+\sum_{i=1}^{n} w_i X_i)}{1+exp(w_0+\sum_{i=1}^{n} w_i X_i)} \tag{2}$$

This problem can be solved through gradient ascent, which is to update the vector of parameters iteratively, until it satisfies some termination condition. In addition, to reduce overfitting, Logistic Regression is allowed to add a regularization term $P(W)$. In our implementation, we plan to use both L1 and L2 norm, which can be represented as:

$$L1 : \ln P(W) \propto \sum_i \mid w_i \mid \quad L2 : \ln P(W) \propto \sum_i w_i^2 \tag{3}$$

Compared to Naive Bayes, Logistic Regression has no assumption that features are independent with each other conditional on label, but it can only produce a linear decision boundary.

3.4 Support Vector Machine

Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane that gives the maximum margin γ to the training examples.

$$\begin{aligned} \min_w \quad & ||w||_2^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(w^{T_i} x_i) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned} \tag{4}$$

Here the *C* is the slack variant that we need to assign before starting the training process.

The goal of SVM is to find a linear boundary with the maximum margin of the different labels. Therefore, if the data is linearly separable, the *linear* boundary that the SVM finds should be with a rather high accuracy. With the same cost function shown above, in addition to performing linear classification, SVMs can efficiently perform a non-linear classification using the *kernel trick*, which can implicitly mapping their inputs into high-dimensional feature spaces. A typical kernel function that can map the feature into a infinite space is the Gaussian kernel (which is also known as RBF), which can be represented as:

$$K(X_i, X_j) = \exp(-\gamma||X_i - X_j||^2), \text{for } \gamma > 0 \tag{5}$$

Thus if we train a non-linear classifier, even if the training data is not linearly separable, it should still give a rather decent accuracy, which means that by applying both linear and none-linear SVM classifiers, the change of the accuracy of them *should* tell us something of the distribution of training data. Therefore, we may implement both linear and none-linear with RBF kernel L1 SVM classifiers in our system.

3.5 AdaBoost

AdaBoost is adaptive boosting algorithm. It can turn weak algorithm into a strong learner by concentrating on examples misclassified by previous iteration and combining hypothesis of each iteration as the final hypothesis.

For dataset $S = \{(x_1, y_1), ..., (x_m, y_m)\}; x_i \in X, y_i \in Y = \{-1, 1\}$ and weak algorithm A, construct example distribution $D_t (t = 1, 2, ...T)$.

$$D_{t+1} = \frac{D_t(i)}{Z_t} \exp\{-\alpha_i y_i h_t(x_i)\} \quad (D_1 = \frac{1}{m}) \tag{6}$$

Run A on D_t producing h_t: $X \to \{-1, 1\}$, $\epsilon_t = P_{x_i}(h_t(x_i) \neq y_i)$, $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$

$$H_final(x) = sign(\sum_t \alpha_t h_t(x)) \tag{7}$$

AdaBoost can be used to improve the learning performance for any weak algorithms, like Naive bayes, decision stumps, etc. It converges vary fast, and has no parameters to tune. In addition, when training accuracy converges, the test accuracy could be further improved as the number of rounds increasing. In our project, we plan to applied AdaBoost to Naive Bayes classifier.

3.6 Feature Engineering

Feature engineering plays an important role in machine learning. It transforms raw data into features with better structures to feed the learning models, improving learning performance. Features you extracted will directly influence the predictive models you use and the results you can achieve. Better feature structures will allow you to use less complex models and get results faster. Feature extraction is to automatically reduce the dimensionality of observed data into a much smaller set. High dimensional features not only make the computation complex, but also may cause the overfitting problem. We plan to reduce the dimensions of some data fields and reconstruct the features. In our project, we will use both K-means and PCA techniques.

3.6.1 PCA

Principle Component Analysis(PCA) is an unsupervised machine learning technique for extracting variance structure from high dimensional datasets. It transforms the original data into a lower dimensional space so that the variance of the projected data is maximized. The lower dimensional space consists of principle components v with large eigenvalues.

$$arg\max_{v} \frac{1}{n}\sum_{i=1}^{n} x_i - (v^T x_i)v \tag{8}$$

PCA as one of dimensional reduction techniques, helps to get a better generalization of data.

3.6.2 K-Means

K-Means is an unsupervised learning algorithm. It partitions data into k clusters with the nearest distance to the cluster center(mean). Assume that $d(x, y)$ represents the distance between point x and y. The algorithm aims to find center $c_1, c_2, ..., c_k$ of k clusters, and

$$minimize \sum_{i=1}^{n} min_{j\in\{1,2,...,k\}} d^2(x_i, c_j) \tag{9}$$

At the beginning, initialize k clusters with centers c_i, and partition all data into these k clusters. Then, according the data distribution, update the center of each cluster. Finally, we can find an optimal local solution through iterations. K-means can be used to reduce feature dimension, which will be compressed from n-dimensional space to k-dimensional space.

4 Experiment Results

4.1 Data Preprocessing

We first extract *student ID*, *problem name*, *step*, *knowledge component*, and *opportunity count* from the dataset to predict the value of *correct first attempt* from the training set. For Logistic Regression and SVM, we map the string fields into positive integers before feeding them to the learning machine. To be specific, we build a dictionary for each string field. The key is a string, and the corresponding value is a positive integer. As a result, an training sample can be represented as five integers.

4.2 Naive Bayes

We use both MLE and MAP to compute the conditional probabilities of each feature. For MLE, the conditional probability is represented as:

$$P(x_i = m \mid Y = y_k) = \frac{\text{count of } x_i = m \text{ among } Y = y_k \text{ cases}}{\text{total count of } Y = y_k \text{ cases}} \tag{10}$$

For MAP, we use Laplace Smoothing. The conditional probability is represented as:

$$P(x_i = m \mid Y = y_k) = \frac{\text{count of } x_i = m \text{ among } Y = y_k \text{ cases } + 1}{\text{total count of } Y = y_k \text{ cases} + \mid x_i \mid} \tag{11}$$

The experiment results are shown in Table 1. We set the Naive Bayes with Maximum Likelihood as our benchmark in the following experiment.

Table 1: Classification results of Naive Bayes

CLASSIFIER	TRAIN ACCURACY	TEST ACCURACY	TRAIN RSME	TEST RSME
NB-MLE	67.77%	63.04%	0.5676	0.6079
NB-MAP	67.53%	64.40%	0.5698	0.5966

Where the Root Mean Squared Error is computed by:

$$RMSE(\hat{y} = \sqrt{E((\hat{y} - y)^2)}) = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y_t} - y)^2}{n}} \tag{12}$$

The prediction accuracy of Naive Bayes is not ideal. The deviation of the differences between predicted values and real values are large. And the result of MAP does not make a great improvement compared to the result of MLE. We assume that the reason of low accuracy may be the conditional independent assumption of Naive Bayes. In our case, if the features are not conditional independent, the result may be bad. For example, if a knowledge component is difficult to understand, the corresponding steps would be hard to solve. In this case, knowledge component and step are not independent.

4.3 Logistic Regression

For the logistic regression learning, we first map each feature's value into $[0, 1]$ space mentioned in previous section, and then implement the Liblinear[3] open source software to train our model. To reduce overfitting, we apply L1 and L2 regularization term($\lambda = 1$) in the training process. The resulting accuracy and the RSME of both training set and the test set are given as follow.

Table 2: Classification results of Logistic Regression

CLASSIFIER	TRAIN ACCURACY	TEST ACCURACY	TRAIN RSME	TEST RSME
L1-Logistic Regression	73.03%	74.38%	0.51932	0.50612
L2-Logistic Regression	73.04%	74.38%	0.51891	0.50612

Compared to Naive Bayes, we can see that the classification result of Logistic Regression is much better, which in turn proves our reasoning for the inaccuracy of NB classifier. However, the accuracy of our LR classifier is still not high. In this case, the reason should be either overfitting or that samples are *not* linearly separatable. Though Logistic Regression doesn't assume conditional independence during training, it can only produce linear decision surfaces.

Since the training accuracy and the test accuracy are very similar, our regularized model is free from overfitting. Then given the previous result, the bottleneck of the LR classifier might lie in the data distribution rather than overfitting (In fact, one of the experience we learnt from class is that overfitting usually happens when the feature is too complex or the number of training examples are very limited, our LR result gives support to it), that is the data we extracted is not linearly separable.

4.4 Support Vector Machine

For the SVM training, we implement the Liblinear[3] for fast linear SVM learning and LibSVM[4] for the none-linear SVM learning. Parameter $c = 1$

In order to increase the accuracy as well as to boost the training speed, we apply a scaling operation to each of the features, mapping the value to [0, 1], where the 0 means the missing feature. In addition, we add the regularization term L1 during the training process.

4.4.1 Cross Validation

During our implementation, in order to choose the best parameter set for both the linear and the none-linear SVM, we implement the k-fold cross validation method. In k-fold, the original sample is randomly partitioned into k equal size

Table 3: Classification results of SVM

CLASSIFIER	TRAIN ACCURACY	TEST ACCURACY	TRAIN RSME	TEST RSME
Linear SVM	73.02%	74.38%	0.51921	0.50612
Non-linear SVM	73.56%	74.86%	0.50962	0.50311

subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k 1 subsamples are used as training data. The cross-validation process is then repeated k times with each of the k subsamples used exactly once as the validation data. The k results are averaged to produce a single estimation. In our implementation, we set $k = 5$. The potential parameter sets we choose for C in both linear and none-linear SVM is $C \in \{2^{-5}, 2^4, 2^3, ..., 2^3, 2^4, 2^5\}$ while the variable g in RBF kernel is $g \in \{2^{-5}, 2^4, 2^3, ..., 2^3, 2^4, 2^5\}$. After the grid search, the best values we get are: for linear SVM, $C = 1.0$, while for none-linear SVM, $C = 1.0$ and $g = 0.25$, while the cross validation accuracy of linear SVM is 73.85%, and for none-linear SVM is 73.96%

It is worth mentioning that during our cross validation process, we find that when we change the parameters, the accuracies of different validation processes are almost the same.

4.4.2 Classification Results

With the validated best parameters, we then train the corresponding classifiers with the processed dataset. In addition to the accuracy, we also compare the RMSE value of these two classifiers both on the training set and on the testing set. the results are shown in Table 3.

From the results shown above, we can see that both kinds of classifiers outperforms the NB classifier. In addition, the result of the linear SVM is very close to the result of Logistic Regression. However, the results we get from the none-linear classifier is surprisingly close to the linear version, which means that given the number of training samples, even when we map the data into a infinite space, the classification still can't be perfectly solved. Given this observation, we deduce that the bottleneck of our solution should lies on the feature selection, rather than the classifier models.

4.5 AdaBoost

We applied AdaBoost to improve the learning performance of Naive Bayes. The training process is shown in the Table 4.

Table 4: Classification results of AdaBoost

Classifier	Train Accuracy	Test Accuracy	Train RSME	Test RSME
Naive Bayes	67.53%	64.40%	0.5698	0.5966
AdaBosst	73.03%	74.38%	0.5192	0.5061

AdaBoost greatly enhance the accuracy of Naive Bayes, which is almost the same as the results of LR and SVM. And the convergence rate of AdaBoost is very fast. After 4 iterations, the error rates of both the training dataset and test dataset converge. In addition, the test error rate deceases as the training error rate increases. There is no overfitting problem, because as the training error rate increases, the classifier is more and more confident.

4.6 Feature Extraction

Instead of using integer to represent string fields, we use a categorical vector to represent the string field feature. The length of the vector is the dimension of the field. Values in the categorical vector are 0 or 1 for feature "Student ID", "Problem name", and "Step". The ith position of the vector represents a possible value x_i. When "1" occurs at the ith position, it means the vector represents value x_i. There is only one "1" in the vector. Inspired by[2], based on the categorical vector, we implement feature combination. Moreover, "knowledge component" and "opportunity" always appear in pairs and the value of "opportunity" is a positive integer, so we combine these two fields into one feature. To be specific, the ith position of the vector represents a possible "opportunity" value x_i. When "0" occurs at the ith position, it means the vector does not contain "knowledge component" x_i. The positive integer at it position represents the "opportunity" value of "knowledge component" x_i.

Table 5: Dimension of Each Feature

Feature	Dimension	Tokenized Dimension
Student ID	575	575
Problem	1084	1084
Step	187539	24674
KC	112	112
Total	189448	26583

Compared to others features, the dimension of "step" itself has more than $180k$ dimensions, which makes it a highly sparse vector. However, many of them are very similar, like "-4x+2 = 7" and "-4x+2-2 = 7-2", "x-500000 = 2" and "x-5000000 = 40000". We remove all the numbers and spaces in the step names, and then result in only 24674 distinguished step names. Then we use the reconstructed feature as well as the tokenized "step" to train the Naive Bayes, Logistic Regression, and SVM classifiers. The results can be seen in Figure 1. In our experiments, we implement *numpy* to represent sparse data and use *sklearn* in python to train our classifiers.

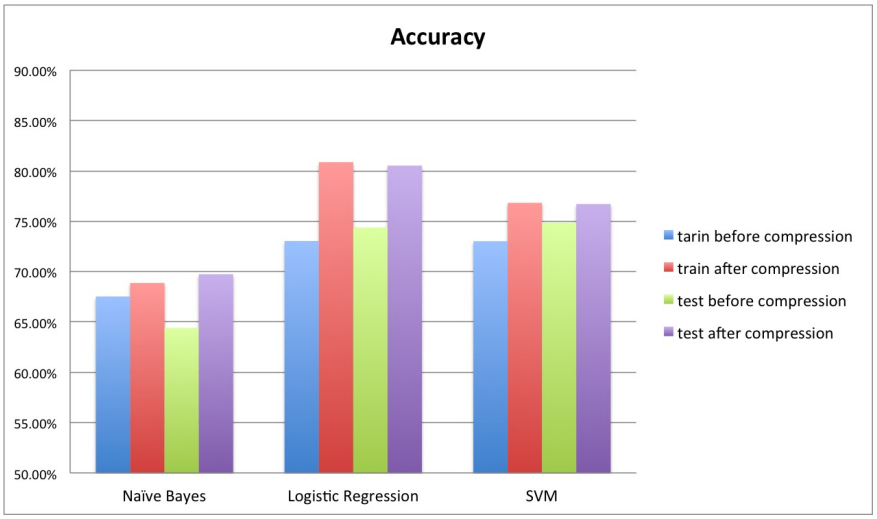

Figure 1: Ccomparison of "step" compression

We get higher accuracy and lower RSME after feeding re-extracted features to all the three classifiers: NB, LR and SVM. The reason is that hashing strings into integers may lose some information. For example, when computing distance in SVM, integer 50 is closer to 100 than 1 to 100, but there is no such assumption. In addition, the tokenization would also help increase prediction accuracy. Lots of steps only have several corresponding examples, which are not enough to make the learner fit the training data. Dimension reduction can make a better generalization of the underlying date structure. The resulting simple feature is easier to feed the classifier and much less likely to cause the overfitting problem.

The prediction accuracy improvement of linear SVM is less than the improvement of LR. The reason might be that SVM produce a linear decision boundary, however, the reduced "step" space makes the whole feature distribution not linear separable any more. In our experiment, even if we train the classifier for 5000 iterations, the value of the loss function would still be more than 10000 while the default suggested convergence criteria is 0.001. Besides, the accuracy of the classifier didn't improve much or even get worse when we increase the iteration number from 1000 to 5000. On the other hand, for Logistic Regression, the learning process can get converged within 10 iterations. Thus, the learning performance of SVM wouldn't improve as much as the performance of LR.

4.6.1 Principal Component Analysis

In this part, we would also like to test on the performance of the PCA reduction. Since the "step name","student id", and "problem name" are unit vectors, it would be less efficient to apply PCA on them (the PCA would result

in picking the most frequent unit vectors as the principal components),. Thus we choose to evaluate PCA on the combined feature representing "knowledge component" and "opportunity". The original dimension of the "knowledge component" feature is 112. The resulting variance as against to the number of most principal components is shown in Figure 2.

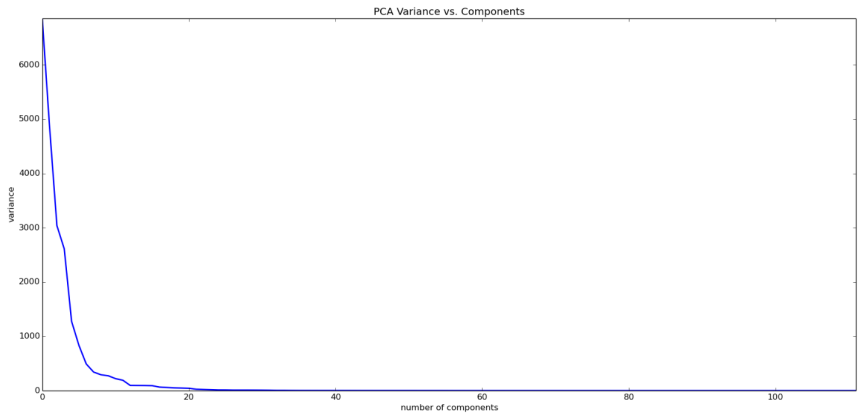

Figure 2: Variance on each components.

From the above figure we can see that the 112 dimension data can be reformed to 11 components. After the 11^{th} components, the data is very sparsely distributed, thus with very low variance. Further, we use Logistic Regression to train on the transformed principal components to see how well the transformed data can represent original model. The accuracy as against to the number of components used on the training set is shown in Figure 3.

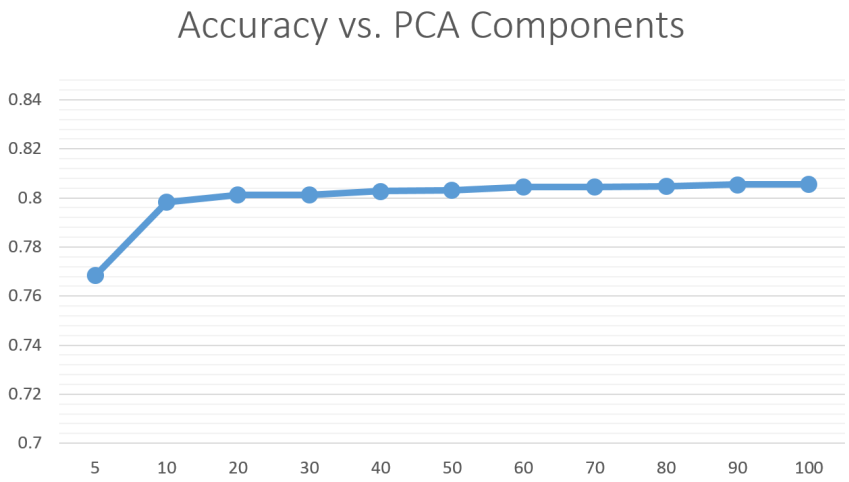

Figure 3: Accuracy of different classifiers

We can see that the training result is consistent with the variance in the previous figure. When we use only 5 principal components, the accuracy is only 76.8%. However, when we use 10 principal components, the accuracy is largely increased. And after 10, even when we increase the number of components, the accuracy only increase a little bit. The reason is that the variance for components from 11 to 112 are very small, which indicates that the rest of the dimensions carry only little information. Also, it is worth mentioning that the accuracy increases when we increase the number of components.

Before we can implement PCA to process the data, we need to densify the data matrix. This becomes a nightmare when the data is of high dimension and the memory is limited since the PCA matrix is related to the number of training examples as well as the feature dimension. Actually, in our implementation, we have to use AWS EC2 instances

(c4.4xlarge) rather than personal PC even for the only 112 dimensional knowledge component feature. (Remember we have more than 800k training examples!)

4.6.2 K-Means Clustering

Known as a famous unsupervised clustering method, k-means can also be used for dimension reduction. As against to PCA which may encounter horrible memory disaster, K-means clustering method doesn't require the data matrix to be dense, thus it can be implemented for high-dimensional sparse data under limited memory. However, K-means has several weaknesses. One biggest problem is the local optimum, and the initialization have a great influence on the performance. In our experiments, we choose to use K-Means++ algorithm with random initial data points to cluster the 112 knowledge component values. The maximum iteration is set to 60. The objective function as against to the number of centers are shown in Figure 4.

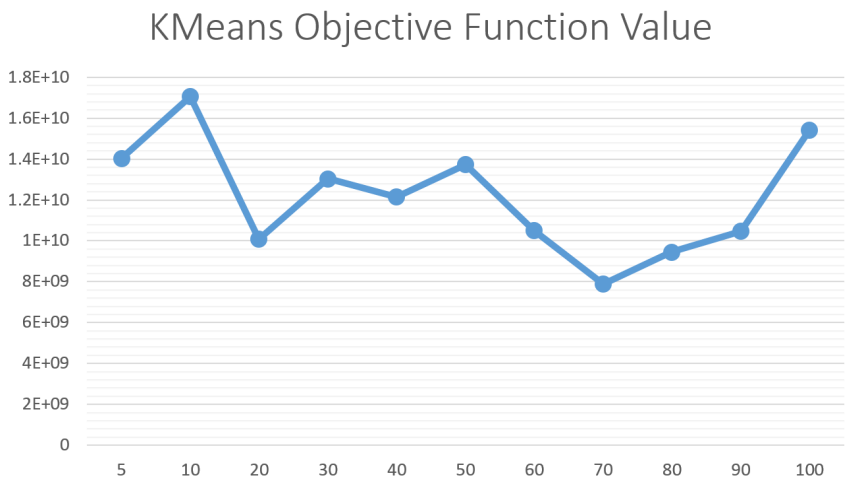

Figure 4: Variance on each components.

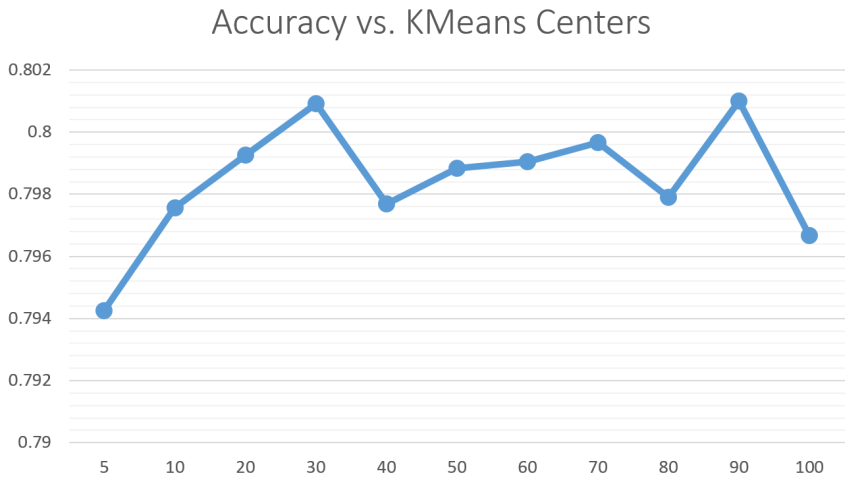

Figure 5: Variance on each components.

From the figure above, the objective function for K-Means in our implementation divergences as we increase the number of centers. This scenario might result partly from the random initialization and partly from the insufficient number of iterations. To better illustrate the performance of k-means in our experiment, we also implement Logistic Regression on the clustered data. The prediction accuracy as against to the number of centers are shown in Figure 5.

According to Figure 5, unlike PCA, the training accuracy doesn't necessarily increase when we increase the number of clustering numbers. We infer that this might due to the fact that even as a enhanced version of K-Means algorithm (Lloyd's heuristic), K-Means++ still doesn't ensure global optimum. And, the randomness in the initial points also bring uncertainty for the performance for the clustering result. On the other hand, unlike PCA, K-Means doesn't require the orthogonality between clustered dimensions. This might also be one reason for the low accuracy in our data set.

Besides, in our experiments, we observe that the clustering time of K-Means is usually much higher than the PCA solving time. In conclusion, though the fact that K-Means can be done with sparsity on large scale data matrix, the performance couldn't be guaranteed. Thus if without memory limitation, it might be better to use PCA rather than K-Means for dimension reduction.

5 Conclusion

As a conclusion, we plot all the results from previous sections in the same figure as shown in Figure 6 and Figure 7.

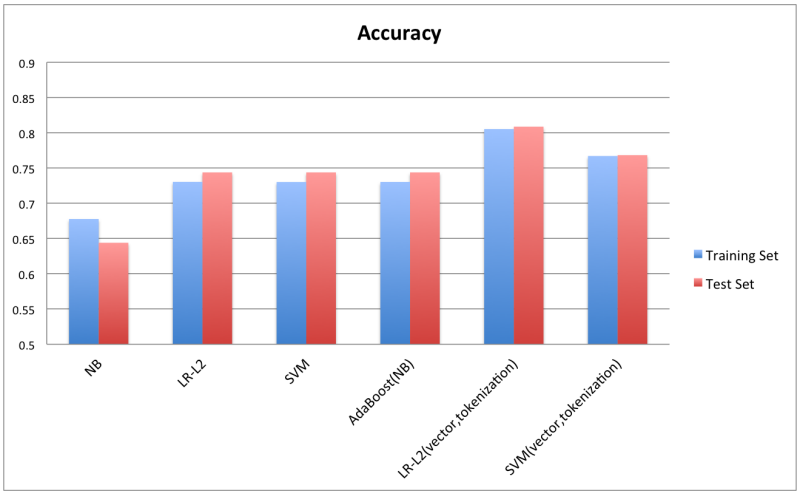

Figure 6: Accuracy of different classifiers

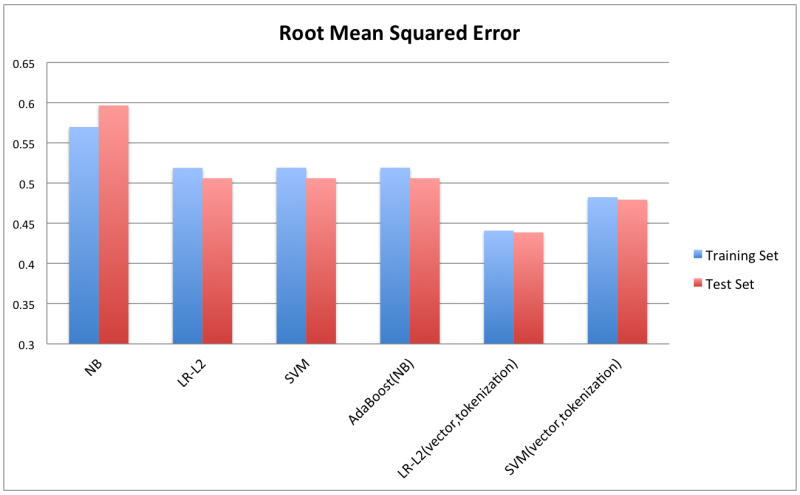

Figure 7: RMSE of different classifiers

Naive Bayes classifier can be done in linear time, but it generates very high RMSE. The features in this problem are not conditional independent, however, Naive Bayes has a strong requirement that each feature is conditional dependent with each other. To improve the prediction accuracy, we apply AdaBoost to Naive Bayes. The result is greatly improved, but still not high enough. So Naive Bayes may not be an ideal classifier for this problem.

For Logistic Regression and linear SVM, the running time complexity is also relatively low. Compared to Naive Bayes, the classification result of Logistic Regression is much better. However, the accuracy of our LR and linear SVM classifier is still not high.

For non-linear SVM, the time complexity is very high. The result is similar to the result of linear SVM. Expending our features to a higher dimension can still not improve the performance.

To further improve the learning performance, we transfer our focus from selecting different classifiers to extracting better features. We implement several methods to reduce the dimension of different fields. For high-dimensional "step" field, we reduce its dimension by tokenization, removing some unimportant characters. The much lower-dimensional space greatly improve our prediction accuracy. For low-dimensional "Knowledge Component" field, we apply both K-means and PCA to reduce dimension. The accuracy has no improvement. When applying dimension reduction to low-dimensional features, it would result in losing some information. So the resulting accuracy even decrease 1%.

In our experiments, we also observe that as a way of dimension reduction, PCA would generate better results than K-Means, because K-Means algorithm suffers from local optimum. As for clustering time, PCA is also more efficient than K-Means in that K-Means need to be performed in multiple iterations. However, the PCA needs to be solved with dense matrix format, which becomes a nightmare when the dimension is high and memory is limited, while the K-Means can be solved with sparse matrix, so it is more scalable.

In conclusion, the best result we get, with highest prediction accuracy and lowest RSME, is to use Logistic Regression classifier with L2 generalization term and reduce dimension of "step" field by tokenization.

6 Future Work

In the future step, we will keep focus on feature engineering. We can reduce the dimension of "step" using some other machine learning techniques. And we can try to add temporal feature into our feature space. For example, for each step, add step name and Knowledge Component from the previous few steps. In addition, we can also record whether a student has met a problem or knowledge component before. In addition, we can try to use linear regularization method to combine results from different classifiers. Our target will still be to improve the predicting accuracy as well as decreasing the RSME.

Acknowledgments

We would like to give special thanks to Prof. Tom M. Mitchell and Prof. Maria-Florina Balcan, it is their outstanding and inspiring lectures that help us open doors to the now trending machine learning field.

We would also appreciate the TAs, especially Ahmed Hefny, for their warm help. Your advices and elaborated explanations helped us turn the knowledge into real world projects.

References

[1] Pardos Z A, Gowda S M, Baker R S J, et al. The sum is greater than the parts: ensembling models of student knowledge in educational software[J]. ACM SIGKDD explorations newsletter, 2012, 13(2): 37-44.

[2] Yu H F, Lo H Y, Hsieh H P, et al. Feature Engineering and Classifier Ensemble for KDD Cup 2010[J].

[3] Zach A. Pardos, neil T.Heffernan. Using HMMs and bagged decision trees to leverage rich features of user and schoolroom an intelligent tutoring system dataset.

[4] Andreas Töscher, Michael Jahrer. Collaborative Filtering Applied to Educational Data Mining.

[5] Fan, Rong-En, et al. "LIBLINEAR: A library for large linear classification." The Journal of Machine Learning Research 9 (2008): 1871-1874.

[6] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology (TIST) 2.3 (2011): 27.