Journal of Data Science 14(2016), 231-244

Predicting Students' Problem Solving Performance using Support Vector Machine

Young-Jin Lee

Educational Technology Program, University of Kansas

Abstract: This study investigates whether Support Vector Machine (SVM) can be used to predict the problem solving performance of students in the computer-based learning environment. The SVM models using RBF, linear, polynomial and sigmoid kernels were developed to estimate the probability for middle school students to get mathematics problems correct at their first attempt without using hints available in the computer-based learning environment based on their problem solving performance observed in the past. The SVM models showed better predictions than the standard Bayesian Knowledge Tracing (BKT) model, one of the most widely used prediction models in educational data mining research, in terms of Area Under the receiver operating characteristic Curve (AUC). Four SVM models got AUC values from 0.73 to 0.77, which is approximately 29% improvement, compared to the standard BKT model whose AUC was 0.58.

Key words: Bayesian Knowledge Tracing (BKT), Educational Data Mining, Log File Analysis, Support Vector Machine (SVM)

1. Introduction

As Koedinger and Aleven (2007) pointed out, it is critical to balance giving and withholding instructional supports in the computer-based learning environment in order to maximize student learning outcomes. Students may not exert enough cognitive effort and fail to acquire a schema from learning tasks if they receive instructional supports prematurely (Kapur, 2008; Schmidt and Bjork, 1992). On the other hand, academically weaker students are likely to fail to learn from learning tasks unless they are provided with appropriate instructional supports and guidance in time. In most of computer-based learning environments, simple heuristics (e.g., giving hints or feedback after students fail to resolve a learning task a certain number of times) or the learner's discretion is used to determine when instructional supports need to be provided. However, simple heuristics would not be able to find the right moment to provide instructional assistance that can maximize student learning outcomes. Similarly, providing instructional supports on the learner's demand may not lead to improved student learning because previous studies found that especially novice learners do not possess enough metacognitive abilities and

prior knowledge required to determine the right moment to ask for help (Clark and Mayer, 2003; Lawless and Brown, 1997).

In order to balance giving and withholding instructional supports in the computer-based learning environment, it is essential to quantify the ability or the level of understanding of students who are trying to learn from given learning tasks. For example, if we can estimate how likely students are to correctly solve a problem based on their performance on other (preferably related) problems they solved in the past, we should be able to make a better judgment on whether or not they need instructional supports.

One of the most popular approaches to quantifying the ability of students is Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1995). BKT is based on Hidden Markov Model (HMM) where the ability of students is assumed to be a binary variable (e.g., do vs. do not understand the Pythagorean theorem) that cannot be observed directly. BKT repeatedly estimates and updates this hidden variable as it encounters a series of successful or unsuccessful observable learning events (e.g., solve or fail to solve a problem requiring an understanding of the Pythagorean theorem). BKT has been used in many previous studies to model the ability of students in computer programming (Corbett and Anderson, 1995), mathematics (Pardos and Heffernan, 2011; Pardos, Gowda, Baker and Heffernan, 2012), reading (Beck and Chang, 2007) and physics (Pardos, Bergner, Seaton and Pritchard, 2013).

Although BKT has been a popular choice among researchers in educational data mining, there are other statistical learning algorithms, such as Support Vector Machine (SVM), that can estimate the ability of students. SVM frequently showed better performance than other data mining algorithms in many research projects ranging from text classification (Joachims, 2002) to bioinformatics (Ding and Dubchak, 2001; Furey, Duffy, Cristianini, Bednarski, Schummer and Hassler, 2000; Hua and Sun, 2001), handwritten digit identification (DeCoste and Sch¨olkopf, 2002) and face recognition (Maghaddam and Yang, 2002). Despite its success, SVM has been rarely utilized in educational data mining research. This study seeks to address this gap in educational data mining by developing SVM-based predictive models of problem solving performance of students, and comparing their predictive power to BKT.

The rest of this paper is organized as follows. Section 2 introduces the SVM classification method, the data set, and how the data set was pre-processed. Section 3 presents how various SVM models were fit to the pre-processed data, and compares the predictive power of the SVM models to the BKT model. Section 4 presents discussions and future directions. Finally, Appendix provides the Python source code snippet showing how an SVM model can be built.

2. Method

2.1. Support Vector Machine (SVM)

SVM is a classification algorithm that tries to reduce the probability of misclassification by maximizing the distance between two class boundaries (positive vs. negative) in data. SVM tries to find a hyperplane, $\langle \vec{w}, \vec{x} \rangle = b$, that can separate positive data points from negative ones as much as possible in a high dimensional feature space. In the case of soft-margin

classification, which can deal with linearly non-separable or noisy data, the maxim margin hyperplane can be obtained by solving the following optimization problem (Cristianini and Shawe-Taylor, 2000):

$$\max_{\vec{\alpha}} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j \, y_i y_j k(\overrightarrow{x_i}, \overrightarrow{x_j}) \tag{1}$$

subject to the constraints

$$\sum_{i=1}^{l} \alpha_i \, y_i = 0$$

$$0 \leq \alpha_i \leq C$$

with $i = 1, \ldots, l$ (number of data points).

Here, α_i is a Lagrange multiplier of the Lagrangian dual of the soft-margin classifier optimization problem, and $k(\overrightarrow{x_i}, \overrightarrow{x_j})$ is a kernel function that allows us to efficiently compute the dot product in a high dimensional feature space without actually projecting data points $\overrightarrow{x_i}$ into the feature space. Table 1 provides a list of kernel functions used in this study, and their free parameter. The free parameter of these kernel functions and the cost constraint C can be determined by cross validation as explained below. The cost constant C regulates the amount of classification error the algorithm would accept while trying to solve the optimization problem. A larger (smaller) C value will result in a narrower (wider) margin classifier because the algorithm has to pay a high (small) price as it commits a classification error. Note that the cost constant allows us to use a linear kernel for the data that is not linearly separable because nonlinear data points can be considered noise.

Table 1: SVM kernel functions used in this study and their free parameter

Kernel name	Kernel function	Free parameter
RBF	$\exp(-\gamma\left\lvert\overrightarrow{x_i} - \overrightarrow{x_j}\right\rvert^2)$	$\gamma > 0$
Linear	$\langle\overrightarrow{x_i}, \overrightarrow{x_j}\rangle$	None
Polynomial	$\langle\overrightarrow{x_i}, \overrightarrow{x_j}\rangle^d$	$d \geq 2$
Sigmoid	$\tanh(\gamma\langle\overrightarrow{x_i}, \overrightarrow{x_j}\rangle)$	$\gamma > 0$

Once α_i values are determined, the normal vector $\vec{w}$ and the offset term b of the maximum margin hyperplane separating positive instances from negative ones in the data can be computed as follows:

$$\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \overrightarrow{x_i}$$

$$b = \sum_{i=1}^{l} \alpha_i y_i \langle \overrightarrow{x_i}, \overrightarrow{x_{xv+}} \rangle - 1 \quad (2)$$

where, $\overrightarrow{x_{xv+}}$ is any data point that lies on the maximum margin hyperplane supporting the positive class boundary. Then, the class membership of a new data point $\vec{x}$ can be determined by a decision function $\hat{f} = sign(\langle \vec{w}, \vec{x} \rangle - b)$. A new data point $\vec{x}$ will be predicted as a positive instance only when the decision function returns a positive value.

In order to estimate the probability of being in a positive class, rather than the class membership, the posterior probability can be approximated by a sigmoid function as suggested by Platt (2000):

$$\Pr(y = 1|\vec{x}) \approx P_{A,B}(\vec{x}) = \frac{1}{1 + \exp(A(\langle \vec{w}, \vec{x} \rangle - b) + B)} \quad (3)$$

A and B in Equation 3 can be solved by solving the following optimization problem:

$$\min_{z=(A,B)} F(z) = -\sum_{i=1}^{l} (t_i \log(p_i) + (1 - t_i)\log(1 - p_i)) \quad (4)$$

$$p_i = P_{A,B}(\vec{x})$$

$$t_i = \begin{cases} \dfrac{N_+ + 1}{N_+ + 2} \; if \; y_i = +1 \\ \dfrac{1}{N_- + 2} \; if \; y_i = -1 \end{cases}$$

$$N_+ = Number \; of \; positive \; y_i \; in \; the \; training \; set$$

$$N_- = Number \; of \; negative \; y_i \; in \; the \; training \; set$$

$$i = 1, \ldots, l$$

Building an SVM model typically requires the following steps:

1. Preprocess data to create predictor variables conforming to the format of an SVM library (see Section 2.3). This study used an open source python data mining library called scikit-learn (Pedregosa et al., 2011) in building soft-margin SVM models that can predict the class membership and the posterior probability of unseen problem solving data.
2. Normalize all non-categorical predictor variables in order to keep them be- tween 0 and 1 (see Section 2.3).
3. Choose an appropriate kernel to be used (see Table 1).
4. Separate the data into training and test sets (see Section 2.3).
5. Find the best parameters for the selected SVM kernel by conducting cross validation on the training set (see Section 3.1)
6. Build the final SVM model using the best parameters identified in the previous step and the entire training set (see Section 3.2).
7. Evaluate the predictive power of the developed SVM model against the test set (see Section 3.2)

2.2. Data set

The data set analyzed in this study was obtained from the Pittsburgh Science of Learning Center (PSLC) (http://www.learnlab.org). Their DataShop Web service (http://pslcdatashop.org) provides log files of computer-based learning environments capturing the learning processes of students trying to solve various subject matters, from foreign language to mathematics and physics (Koedinger, Baker, Cunningham, Skogsholm, Leber and Stamper, 2010). This study used 'Assistment Math 2004-2005' data set that captured how 912 middle school students used a Web-based algebra learning environment for over 3,400 student hours. The original data set obtained from PSLC includes 580,786 database transactions where each transaction record contains information about students, and their problem solving activities such as problem/step name, problem/step solving time, and whether or not they were able to solve each problem solving step successfully (see Table 2).

Table 2: Problem solving information available in the PSLC data set

Column in PSLC data set	Description
Anonymized student ID	Anonymized student ID generated by DataShop
Problem name	Name of the problem associated with the transaction
Step name	Name of the problem solving step associated with the transaction
Problem time	Time at which the student started solving the problem
Step time	Time at which the student started working on a particular problem solving step
Number of problem views	Number of times the student tried to solve the same problem
Number of attempts at step	Number of times the student submitted an answer to the same problem solving step
KC	Knowledge Component associated with the transaction
Outcome	Result of the problem solving attempt (CORRECT, INCORRECT or HINT)

One thing to note is that each transaction is associated with a particular Knowledge Component (KC), which is defined as "an acquired unit of cognitive function or structure that can be inferred from performance on a set of related tasks1." KC allows for categorizing problem solving steps so that problem solving steps in the same category can be considered examining one particular mathematics concept (e.g., Pythagorean theorem).

2.3. Data pre-processing

[1] http://pact.cs.cmu.edu/pubs/PSLC-Theory-Framework-Tech-Rep.pdf

Considering the fact that building an SVM model is computationally expensive (see Table 3), compared to BKT and other standard statistical approaches such as Generalized Linear Model (GLM), the original data set obtained from PLSC was too big to be processed on a standard desktop computer. In order to make the computation tractable on a standard desktop computer, this study used randomly selected 10% of the transaction records.

Table 3: Best parameters and execution time of SVM models

Kernel	Parameter range	Best parameters	Execution time
RBF	$C = 10^{-3} - 10^{5}$ $\gamma = 10^{-3} - 10^{2}$	$C = 10$ $\gamma = 10^{-2}$	44h 27m 54s
Linear	$C = 10^{-3} - 10^{2}$	$C = 10^{-1}$	23h 45m 14s
Polynomial	$C = 10^{-3} - 10^{11}$ $d = 2, 3, 4$	$C = 10^{7}$ $d = 3$	36h 13m 22s
Sigmoid	$C = 10^{-3} - 10^{5}$ $\gamma = 10^{-3} - 10^{2}$	$C = 10^{2}$ $\gamma = 10^{-3}$	28h 38m 04s

Since the goal of this study was to build SVM models that can predict students' problem solving performance based on their problem solving history, the selected transaction records were pre-processed as follows. First, for each selected transaction record, its anonymized student ID, KC and step time were identified. This information is then used to compile all transaction records with the same anonymized student ID and KC, and earlier step time and problem time. From these records of past problem solving performance on the same KC, number of unique problems and steps the student solved, fraction of correct steps/problems, fraction of incorrect steps/problems, fraction of steps/problems with a hint request, and streaks of correct answers were computed. These predictors were then normalized in order to keep them in the same [0, 1] range. Otherwise, predictors with a broader range will have an unfair influence on the objective function an SVM algorithm tries to optimize (Equation 1). In addition, dummy variables were created to incorporate categorical variables, such as anonymized student ID, problem name, step name, and KC, into SVM prediction models. Finally, a new outcome variable (CORRECT or WRONG) was created by combining INCORRECT and HINT cases because the focus of this study was to predict whether students will be able to solve a problem without using hints available in the computer-based learning environment.

In order to estimate the predictive power of SVM models without bias, the preprocessed PLSC data set was divided into training and test sets. When creating a test set, which consists of 20% of the pre-processed data, stratified random sampling was used to ensure that the ratio of positive to negative instances in the training and the test sets are similar.

3. **Results**

3.1. Tuning SVM models

In this study, five-fold cross validation was used to find the best values for tuning parameters (cost constant in Equation 1 and free parameter of kernel functions in Table 1) of SVM models that can maximize the predictive power on unseen future data. First, the values of tuning parameters were selected from a grid spanning an appropriate parameter space. For example, an SVM model using an RBF kernel chooses two values for C and γ from a two dimensional parameter space. Then, the training set was randomly divided into five sets of roughly equal size with similar proportions of positive and negative instances, and an SVM model with the selected tuning parameter values was fit using all samples in the training set except for one subset. The samples in the held-out set, which played a role of future data because they were not used in the model building process, were then used to estimate the performance of the SVM model with these particular parameter values. These processes were repeated five times with a different subset of the training set being used as a held-out set. The average of the five estimates of predictive power was used to represent how well an SVM model with specific tuning parameters would work when it is given new problem solving data in the future.

In this study, the predictive power of an SVM model is measured by Area Under the Curve (AUC) obtained from a Receiver Operating Characteristic (ROC) curve analysis. When applied to a binary classification problem, an ROC curve is a plot of false positive rate (1 − specificity) vs. true positive rate (sensitivity) derived from the posterior probability of each data point estimated by the learning algorithm (Equation 3). Since a good classification model will report a small false positive rate and a large true positive rate, the area under the ROC curve of a good classification model will has a large AUC value. AUC can vary from 0.5 (predictive power not better than simple guessing) to 1.0 (perfect predictive power), and it is known to be equal to the probability that a learning algorithm ranks a randomly chosen positive instance higher than a randomly chosen negative one (Fawcett, 2006).

Figure 1 shows the AUC values of an SVM model using an RBF kernel which has two tuning parameters, C and γ. The γ parameter of an RBF kernel determines how far the influence of data points in the training set selected as support vectors can reach. A large γ value means that the influence of support vectors will be limited to the data points close to them. As a result, when γ is large (γ = 1, 10, 102 in Figure 1), AUC does not change much. When γ is small, on the other hand, AUC shows significant changes and the largest AUC value (0.755) was obtained at C = 10, $\gamma = 10^{-2}$.

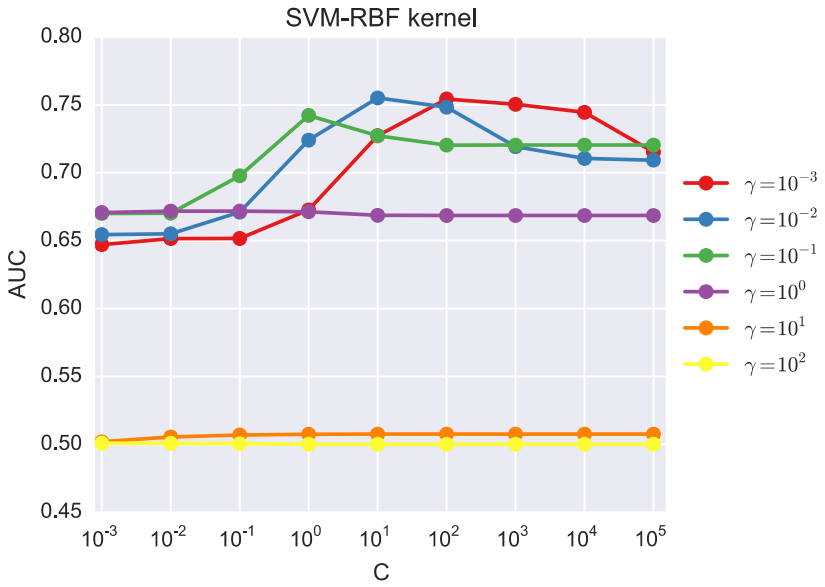

Figure 1: AUC values of an RBF kernel SVM model computed over different C and γ values

Table 3 summarizes the best tuning parameters of each SVM model and the execution time that was needed to find them. Since each SVM model converged differently in the parameter space, different parameter ranges had to be explored. In the case of a polynomial kernel SVM model, for example, a much wider range of C values had to be examined because its AUC value changed rather slowly, compared to other kernels such as a linear kernel whose AUC value was peaked at $C = 10^{-1}$ and leveled off quickly. On a standard desktop computer with an Intel Core i5 processor (3.40 GHz) and 8 GB of memory, the parameter tuning of SVM models took about one or two days.

3.2. Comparing SVM models to standard BKT model

The final SVM prediction models were built by fitting the entire training set with the best tuning parameters determined from the five-fold cross validation procedure explained above (see Table 3). For each student in the test set, their class membership, CORRECT (get the problem solving step correct at their first attempt without using any hints available in the computer-based learning environment) vs. WRONG (either get the problem solving step wrong or requested a hint), and the posterior probability of being in the CORRECT class were computed using Equation 2 and 3.

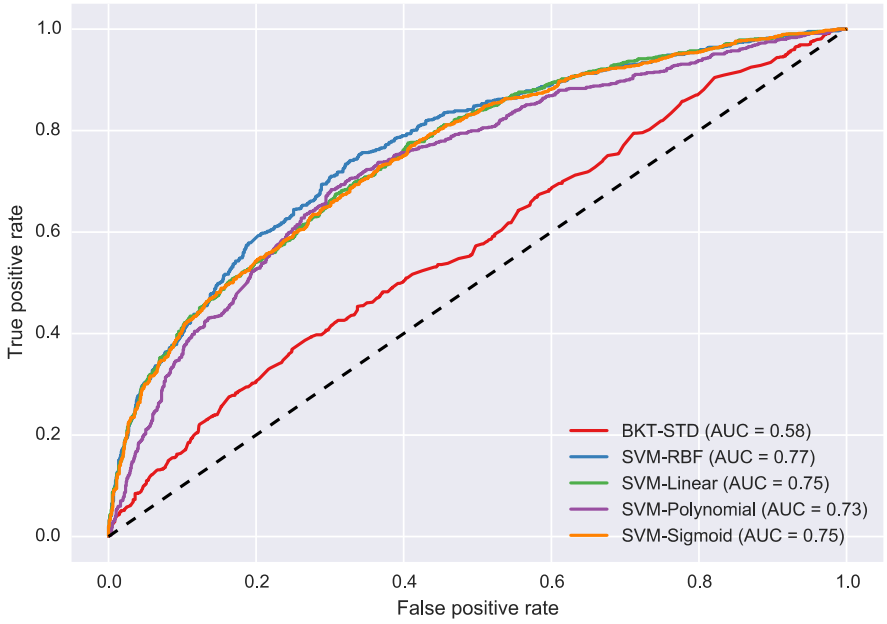

Figure 2: Comparison of ROC curves and their AUC values of prediction models

Figure 2 compares the ROC curves of four SVM models and the standard BKT model built from a publicly available C++ code (https://github.com/IEDMS/ standard-bkt). Four SVM models yielded much larger AUC values, ranging from 0.73 to 0.77, than the standard BKT model (AUC = 0.58), indicating that these SVM models would make more accurate predictions than the standard BKT model.

The confusion matrices of the standard BKT model and the RBF kernel SVM model reveal that the RBF kernel SVM model performed better across the board (see Figure 3). The RBF kernel SVM model showed higher recalls (0.78 vs. 0.72 for Wrong; 0.61 vs. 0.40 for Correct) and precisions (0.73 vs. 0.62 for Wrong; 0.66 vs. 0.50 for Correct) when students were predicted to be able to solve the problem by themselves if the posterior probability, $\Pr(y = 1|\vec{x})$, is greater than 0.5.

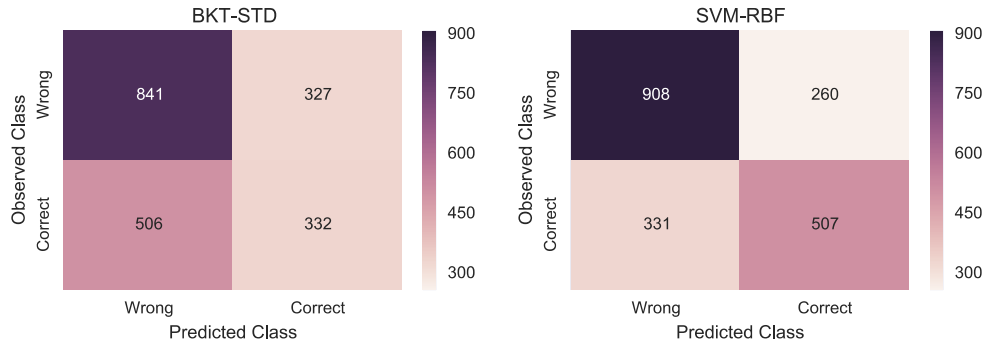

Figure 3: Comparison of confusion matrices of prediction models (Cut-off probability = 0.5)

4. Discussion

The results of this study suggest that SVM may be able to make better predictions on the problem solving performance of students, compared to the standard BKT model, the most widely used computational method in educational data mining research. The possible reason for this result is that SVM models used more information about how students solved relevant problems in the past, compared to the standard BKT model; SVM models have 13 predictors (number of unique problems and steps students solved, fraction of correct steps/problems, fraction of incorrect steps/problems, fraction of steps/problems with a hint re- quest, streaks of correct answers, anonymized student ID, problem name, step name, and KC) whereas the standard BKT model has only one predictor variable (sequence of correct or wrong responses on each KC). This interpretation is in line with Pardos and Heffernan (2011)'s study where they were able to achieve performance gains by including the difficulty of problems in the standard BKT model. It would be important to further investigate whether other predictors or some combinations of predictors can be used to improve the performance of SVM models because adding non-relevant predictors can hurt the performance of a data mining algorithm.

Another reason for the poor performance of the standard BKT model is that it does not take into account the ability of students. Obviously, academically stronger students would show better problem solving performance than academically weaker students. However, the standard BKT model does not incorporate the ability of students, resulting in a prediction model for average students. Recently Yudelson, Koedinger and Gordon (2013) proposed an individualized BKT model to allow the standard BKT model to include the ability of students. It would be interesting to compare the individualized BKT model to SVM models when the individualized BKT model's code becomes available for researchers to use in the future.

Finally, recent research has shown that ensemble methods can help build a better predictive model of how students solve problems in the computer-based learning environment (Pardos, Gowda, Baker and Heffernan, 2012). As a future work, various ensemble models, such as Random Forest (Breiman, 2001), Ada Boost (Freund and Schapire, 1999) and Gradient Boosting Machine (Friedman, 2001), will be developed to compare their predictive power to that of the SVM models reported in this study.

References

[1] Beck, J. E. and Chang, K. M. (2007). Identifiability: A fundamental problem of student modeling. In Proceedings of the International Conference on User Modeling, 137–146. Corfu, Greece.

[2] Breiman, L. (2001). Random forests. Machine Learning 45, 5–32.

[3] Clark, R. C. and Mayer, E. (2003). E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning. Pfeiffer, San Francisco.

[4] Corbett, A. and Anderson, J. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction 4, 253–278.

[5] Cristianini, N. and Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, New York.

[6] DeCoste, D. and Sch¨olkopf, B. (2002). Training invariant Support Vector Machines. Machine Learning 46, 161–190.

[7] Ding, C. and Dubchak, I. (2001). Multi-class protein fold recognition using Support Vector Machines and neural networks. Bioinformatics 16, 349– 358.

[8] Fawcett, T. (2006). An introduction to roc analysis. Pattern Recognition Letters 27, 861–874.

[9] Freund, Y. and Schapire, R. E. (1999). A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence 45, 771–780.

[10] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29, 1189–1232.

[11] Furey, T. S., Duffy, N. Cristianini, N., Bednarski, D., Schummer, M. and Hassler, D. (2000). Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data. Bioinformatics 16, 906–914.

[12] Hua, S. and Sun, Z. (2001). Support Vector Machine approach for protein subcellular localization prediction. Bioinformatics 17, 721–728.

[13] Joachims, T. (2002). Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Kluwer Academic Publishers.

[14] Kapur, M. (2008). Productive failure. Cognition and Instruction 26, 379–424.

[15] Koedinger, K. R. and Aleven, V. (2007). Exploring the assistance dilemma in experiments with cognitive tutors. Educational Psychology Review 19, 239–264.

[16] Koedinger, K. R, Baker, R. S. J. d., Cunningham, K., Skogsholm, A., Leber, B. and Stamper, J. (2010). A Data Repository for the EDM community: The PSLC DataShop. In Handbook of Educational Data Mining (Edited by C. Romero, S. Ventura, M. Pechenizkiy, and R. S. J. d. Baker), 43–55, International Educational Data Mining Society.

[17] Lawless, K. A. and Brown, S. W. (1997). Multimedia learning environments: Issues of learner control and navigation. Instructional Science 25, 117–131.

[18] Maghaddam, B. and Yang, M. H. (2002). Learning gender with support faces. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 707– 711.

[19] Pardos, Z. A., Bergner, Y., Seaton, D. and Pritchard, D. E. (2013). Adapting Bayesian Knowledge Tracing to a Massive Open Online Course in edX. In Proceedings of the 6th International Conference on Educational Data Mining, 137–144. Memphis, TN.

[20] Pardos, Z. A., Gowda, S. M., Baker, R. S. J. d. and Heffernan, N. T. (2012). The sum is greater than the parts: Ensembling models of student knowledge in educational software. SIGKDD Explorations Newsletter 13, 37–44.

[21] Pardos, Z. A. and Heffernan, N. T. (2011). KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model. In Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization, 243–254. Girona, Spain.

[22] Pedregosa, F. Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[23] Platt, J. C. (2000). Probabilities in SV Machines. In Advances in Large Margin Classifiers (Edited by Smola J. A.), MIT Press, 61–74.

[24] Schmidt, R. A. and Bjork, R. A. (1992). New conceptualizations of practice: Common principles in three paradigms suggest new concepts for training. Psychological Science 3, 207–217.

[25] Yudelson, M., Koedinger, K. R. and Gordon, G. J. (2013). Individualized bayesian knowledge tracing models. In Proceedings of the 16th International Conference on Artificial Intelligence in Education, 171–180. Memphis, TN.

Received December 12, 2013; accepted July 26, 2014.

Young-Jin Lee
Educational Technology Program
University of Kansas
1122 W. Campus Road
Lawrence, KS 66045, USA

Appendix

```
import pandas as pd
import numpy as np
from sklearn.cross validation import train test split
from sklearn.preprocessing import StandardScaler
from sklearn.grid search import GridSearchCV
from sklearn.svm import SVC

# Read the preprocessed data file
data = pd.read_csv('../ dataFile.txt' ,sep='\t ')
y = data['CFA']
X = data.drop('CFA', axis=1)

# Create dummy variables for categorical predictors
X_student_dummy = pd.core.reshape.get_dummies(X['Student'])
X_problem_dummy = pd.core.reshape.get_dummies(X['Problem'])
X_step_dummy = pd.core.reshape.get_dummies(X['Step'])
X_kc_dummy = pd.core.reshape.get_dummies(X['KC'])
X_num = X.drop(['Student', 'Problem', 'Step', 'KC'], axis=1)

# Create training and test sets
X = np.concatenate((X_num, X_student_dummy, X_problem_dummy,
    X_step_dummy, X_kc_dummy) , axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=1128)

# Normalize non-categorical predictors
scaler = StandardScaler().fit(X_train_num)
X_train_num_transformed = scaler.transform(X_train_num)
X_test_num_transformed = scaler.transform(X_test_num)
X_train_transformed = np.concatenate((X_train_num_transformed,
    X_train_cat), axis=1)
X_test_transformed = np.concatenate((X_test_num_transformed,
    X_test_cat), axis=1)

# Find the best parameters for an SVM model using training set
param_grid = {'C': 10.0**np.arange(-3, 3), 'gamma': 10.0**np.arange(-3,
    3), 'kernel': ['rbf']}
clf = GridSearchCV(SVC(probability=True) ,param_grid , cv=5,
    scoring='roc_auc')
clf.fit(X_train_transformed, y_train)

# Predict class membership of test set
predicted_class = clf.predict(X_test_transformed)

# Predict posterior probability of test set
predicted_prob = clf.predict_proba(X_test_transformed)
```