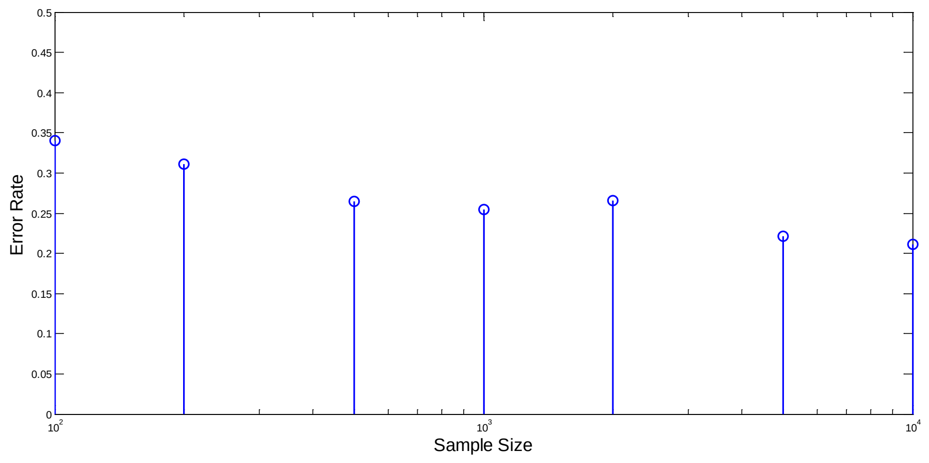
CS289 HW1: Support Vector Machine

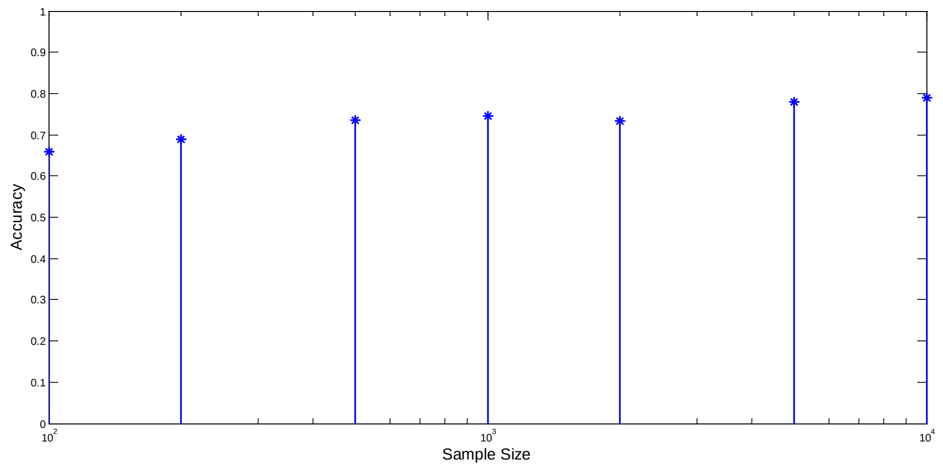
Jiaying Shi
shijy07@berkeley.edu

Data Partitioning

Problem 1

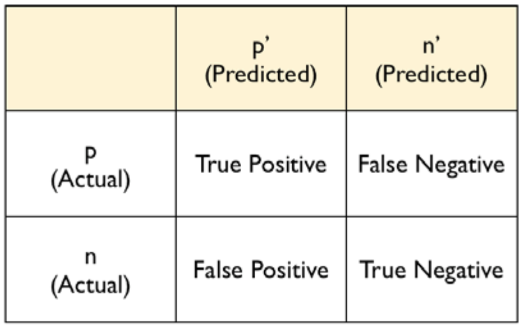
In problem 1, since the training data set is not arrange randomly, so the first thing to do is to shuffle the data. I randomly picked 20,000 training data and take 10,000 of the data points as the validation data. The error rate and accuracy rate of the data partitioning problem with training data size 100, 200, 500, 1000, 2000, 5000 and 10000 is plotted in the following graph. From the graph we can see that the error rate has a decreasing trend when the sample size increases. Similarly, the accuracy rate has an increasing trend when the sample size increases.





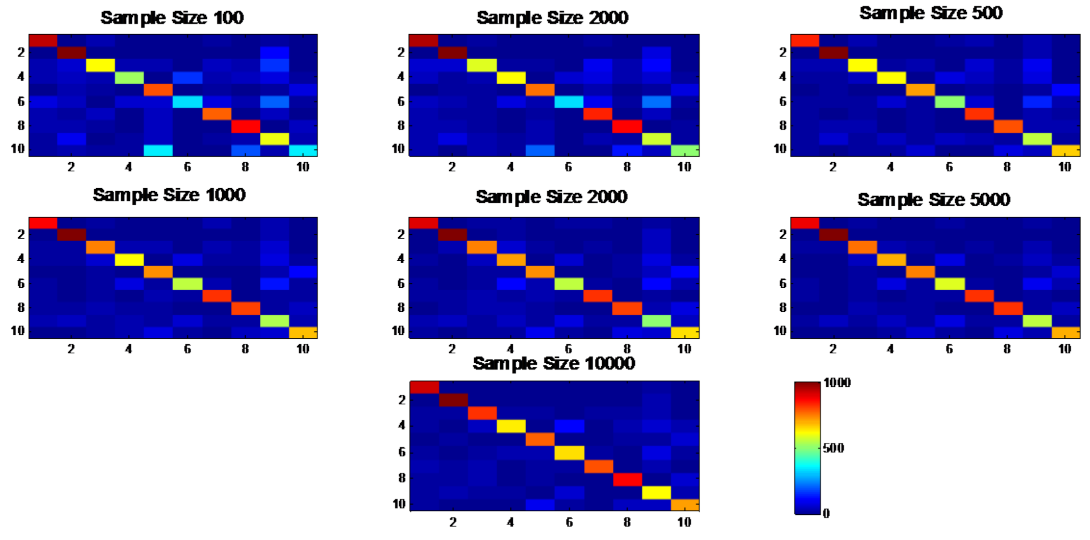
Problem 2

The confusion matrix is defined as the following graph:



	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

The color coded confusion matrices for the seven tests are plotted in the following graph:



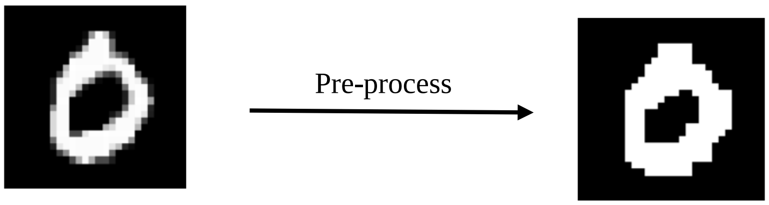
From the color coded confusion matrix we can see that the diagonal elements of the confusion matrices are much larger than the off-diagonal elements. The diagonal elements represent accurately predicted digits. So it means that most digits are predicted accurately. The off-diagonal elements get darker when the sample size is larger. That means the sample size would affect the error rate of the classifier.

Problem 3:

Cross-validation is a model validation technique that can assess how the results of a statistical analysis will generalize to an independent data set. It can help us estimate how accurately a predictive model will perform in practice. There may be over-fitting if we do the validation on a single set or on the training set. Cross-validation can help us avoid over-fitting. Moreover, cross-validation can also help us pick better hyper-parameters since it's a way to assess the accuracy of the model.

Using cross-validation, the best parameter C is 0.35 in my program. But since the training set is randomly picked each time, the best value of C may vary. **The accuracy rate of the SVM with the best C=0.35 is 79.56% on the validation set. The accuracy rate on Kaggle is 79.46%.**

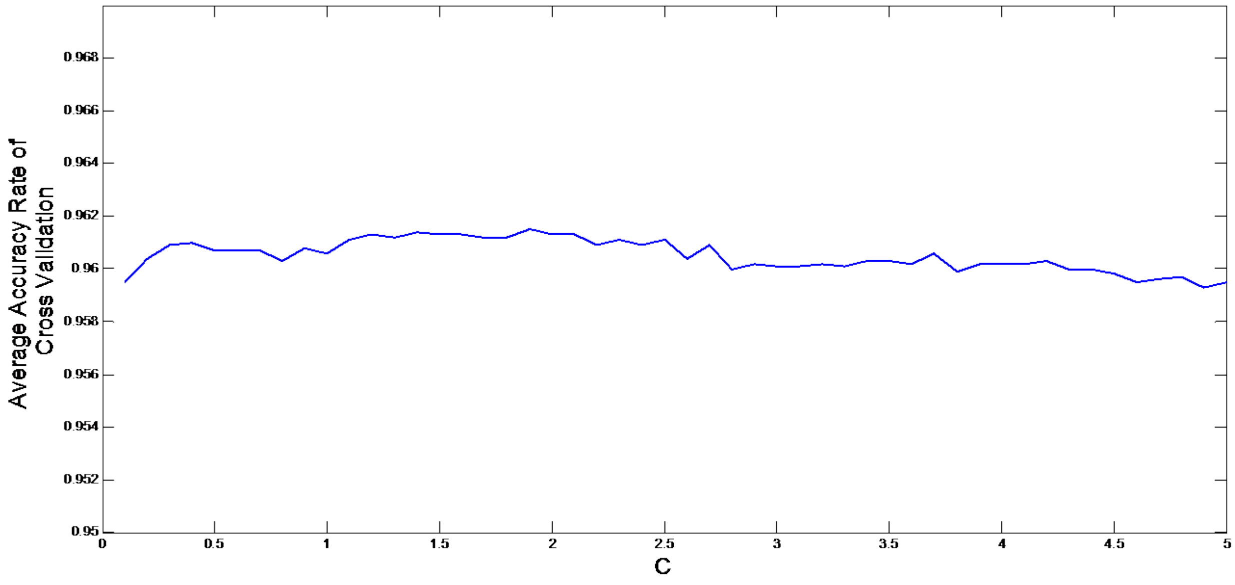
Other than using the raw data, I also tried to filter the data to remove the noise. The intensity image is converted to a binary image. The data before and after the pre-process is shown in the following picture:



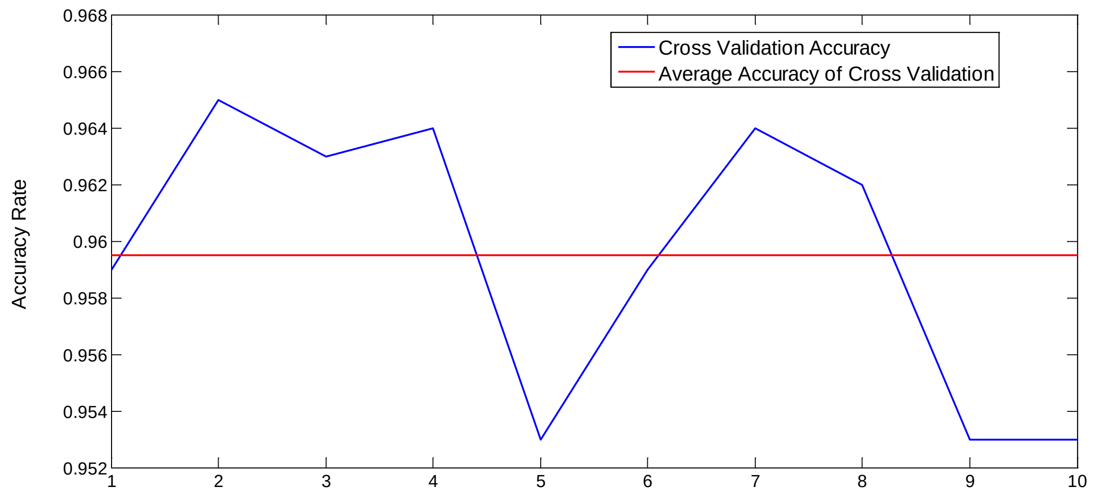
The converted data is treated as feature. Using the processed data, the performance of the digit partitioning is improved. **The accuracy rate of the SVM with the best C=0.10 is 84.29% on the validation set. The accuracy rate on Kaggle is 83.76%.** The code for the cross-validation digit partitioning on the processed data is *corssValidationDigit_BW.m*.

Moreover, I also tried to use HOG of the processed data as features for the data partitioning. A hog feature generation function is downloaded here: http://www.mathworks.com/matlabcentral/fileexchange/46408-histogram-of-oriented-gradients--hog--code-using-matlab. Using the new features, the performance is improved significantly. The code the updated digit partitioning using HOG as features is *crossValidationDigitHOG.m*. The feature generation code is *hog_feature_vector.m*. **The accuracy rate of the SVM with the best C=4 is 89.11% on the validation set. The accuracy rate on Kaggle is 88.96%.**

Another version of HOG feature is also tried in this homework. The window size of the hog generation is tuned. The function is downloaded here: http://www.mathworks.com/matlabcentral/fileexchange/28689-hog-descriptor-for-matlab. In the new HOG feature generation function, the gray-scale images are first filtered by a [-1 0 1] filter (using matlab function imfilter). Then the features are generated using the processed data. The code for this version is *crossValidationDigitHOG_v2.m*. The feature generation code is *HOG.m*. **The accuracy rate of the SVM with the best C=1.9 is 96.35% on the validation set. The accuracy rate on Kaggle is 96.28%.** The accuracy rate of different C value is ploted in the following graph:



I did not store all the accuracy rates for the cross validation but the last C value it tries. The last C that the program tries is 5. With C=5, the accuracy rate for 10-fold cross validation is shown in the following graph:



Spam Detection

Problem 4 (Code for Problem 4: crossValidationSpam.m)**:**

In Problem 4, the features that I added are listed in the following table:

http	www	link	discount	cost	specials	earn
free	luxury	guaranteed	website	edu	investment	snoring
loss	investment	valium	viagra	vicodin	Dear	price
url	re :	meeting	dollar	fyi	hello	deal
sales	cash	bonus	cheap	rate	click	insurance
preview	order	membership	resume	conference	thanks	investment
$	cc:					

In this problem, we have 5172 training data points, and 12 is a factor of 5172. I found the best C value using a 12-fold cross-validation. **The best C is 3.75. My Kaggle accuracy rate is 84.597%. The average cross-validation accuracy for the best C is 87.82%**