CS 7641 Machine Learning
Assignment 1 - Supervised Learning

Liyue Hu (lhu81)

Abstract

A comparison of five different models for supervised learning on two sets of data. The different models include Decision Tree, K-Nearest Neighbors, Neural Networks, Support Vector Machine and Decision Tree with Boosting. The two datasets were predicting credit card default on next month's payment and handwriting recognition of digits.

Datasets:

For this assignment, I have selected 2 distinct classification problems: predicating default on credit card and classifying handwritten digits (0-9).

Dataset 1 – Credit Card Default

Description

This dataset contains information on 30000 clients in Taiwan from April 2005 to September 2005. There is a total of 23 features, including demographic features (Credit Limit, Sex, Education, Marriage, Age), repayment status over the five months, amount of bill statement over the five months and amount of previous payment in the five months. These features are used to predict whether these clients defaulted on their payment for the following month.

Why is it interesting?

The problem is very interesting as it has practical applications in the real world that is costing companies a lot of money each month. If banks are able to identify clients who are likely to default as early as possible, they can improve their profitability, or it could even lead to lower credit card rates for other (non-defaulting) clients. The features in this dataset will be interesting to compare and contrast as it includes several different types of data.

Dataset 2 – Pen Digits Recognition

Description

The second dataset contains 10992 instances of information. It contains a collection of samples of handwritten numbers from 44 writers. are 16 attributes consisting of 8 resampled (x_t, y_t) points, representing digits as constant length feature vectors. The final input attributes are integers in the range of 0 and 100. The classes (y) are the 10 digits between 0 and 9, each representing the corresponding digit.

Why is it interesting?

Pattern recognition is an important field of machine learning that has high applicability in our daily lives. We are increasingly interacting with pen-based digital devices such as Tablets, computers, smart-phones, etc. Theories of user-centric design advocates for "invisible" interfaces as they provide better

user experience. Handwriting recognition is an important aspect that will enable more “invisible” interfaces.

Models

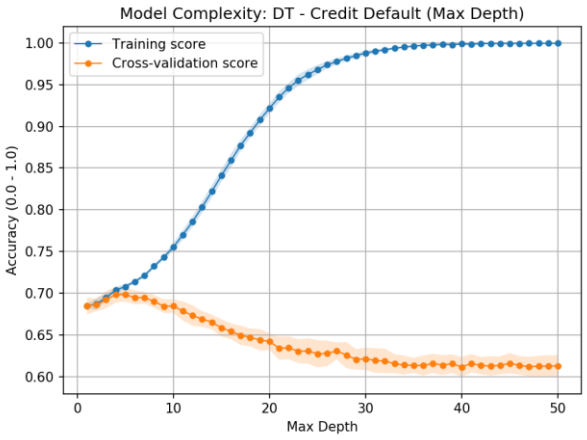
To analyze the training and testing error rates I obtained running the various learning algorithms, I created learning curve for all models.

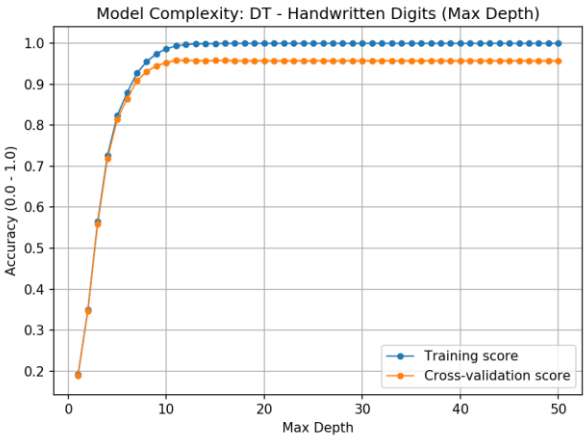
- the training and testing error rates you obtained running the various learning algorithms on your problems. At the very least you should include graphs that show performance on both training and test data as a function of training size (note that this implies that you need to design a classification problem that has more than a trivial amount of data) and—
- for the algorithms that are iterative--training times/iterations. Both of these kinds of graphs are referred to as learning curves, BTW.

<u>Decision Tree</u>

The optimal parameters are split with entropy, max depth of 5, minimum sample split of 2. Pruning was implemented according to “Reduced Error Pruning” as mentioned in the Machine Learning textbook by Mitchell. Pruning was done iteratively on the subtrees on all the nodes, checking to see if the removal of that subtree resulted in performance no worse than the original over the validation set.

Max Depth

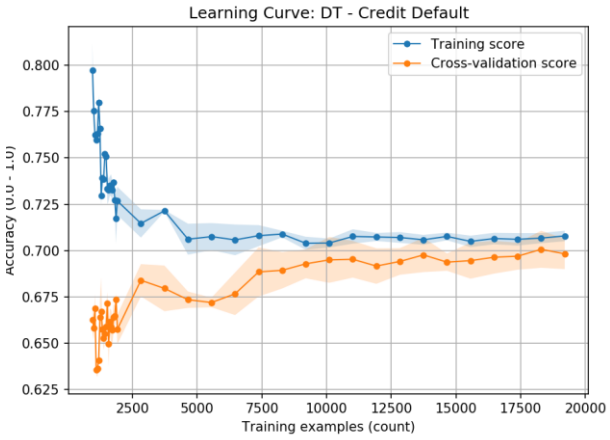


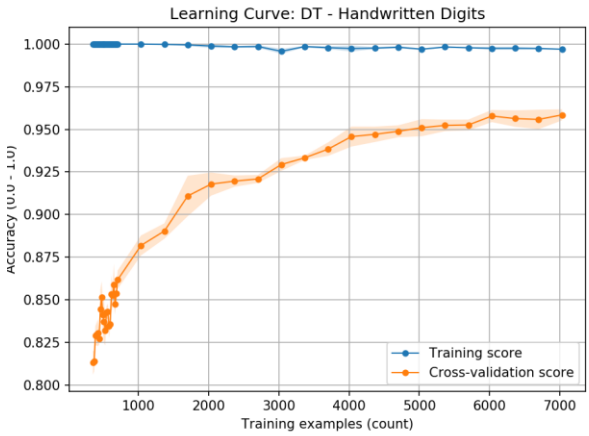


Credit Card Default – the optimal max depth is 5, and as max depth grow above that, meaningful overfitting happens, seen as training accuracy becomes increasingly greater than cross-validation score. This could mean that the model has a lot of noise and that by increasing the depth of the tree, the model become worse at predicting the actual labels. We have a limited amount of data on each client and a lot of other factors can cause a client to default on next month’s payment.

Pen Digits Recognition – the optimal max depth is 8 for this dataset, and the model complexity graph looks very different from that of the first dataset.

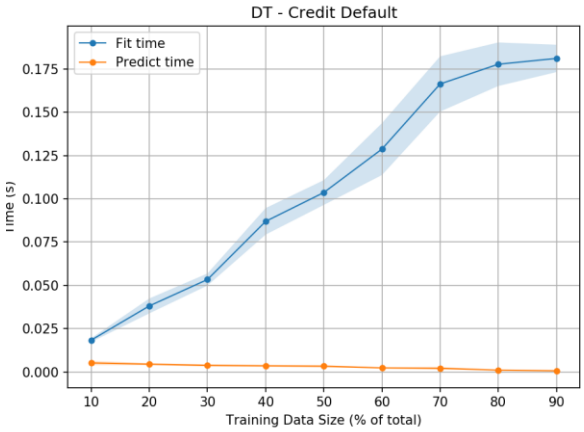
Learning Curve

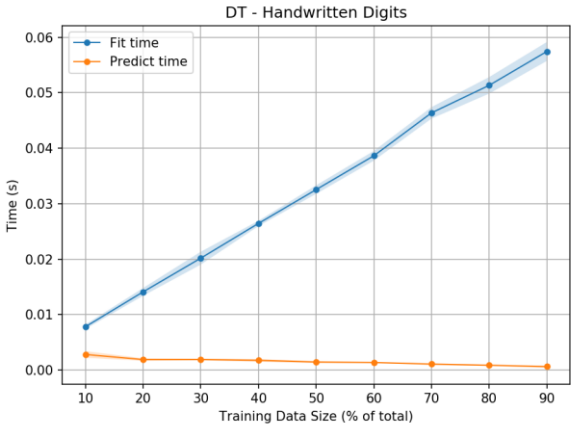




Both datasets show convergence on the learning curve as more training examples are used.

Time:

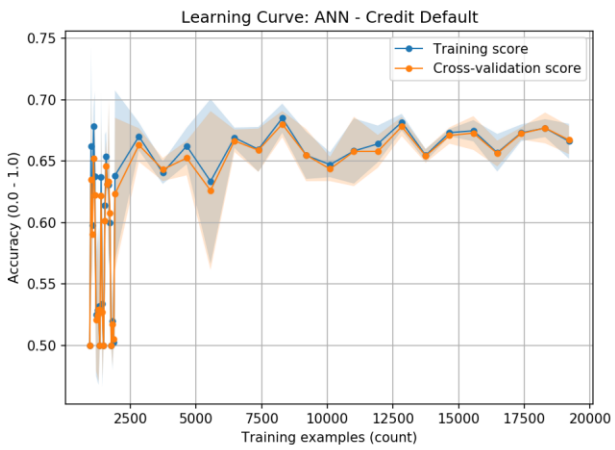


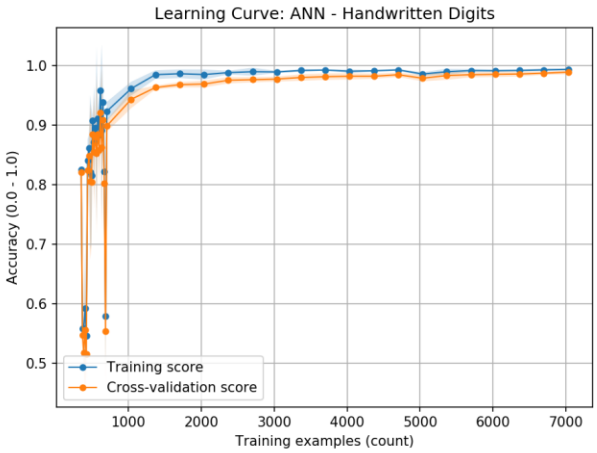


For both models, predict time was constant and close to 0 while fit time increased almost linearly with increases in training data. This is because Decision Tree is an eager learner, it front loads the computing by constructing a classification model first, which decreases the time needed on querying.

Neural Networks

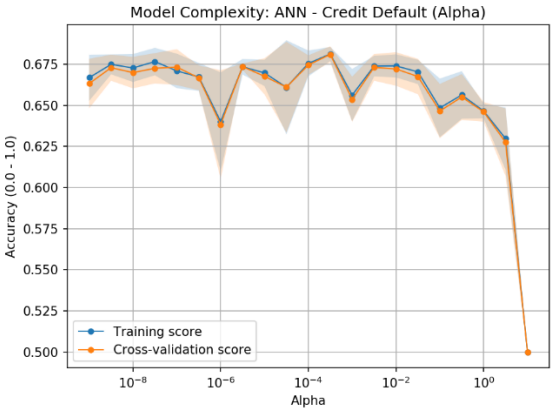
Learning Curve

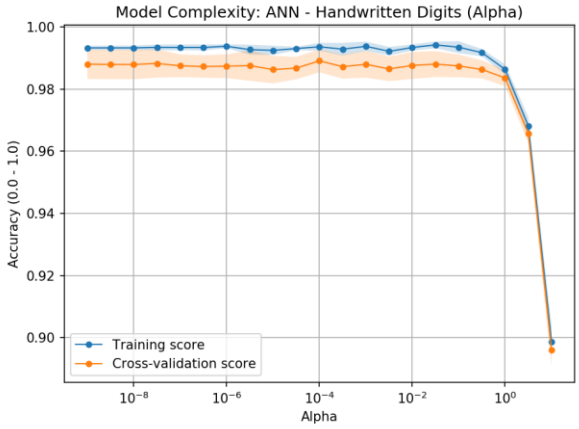




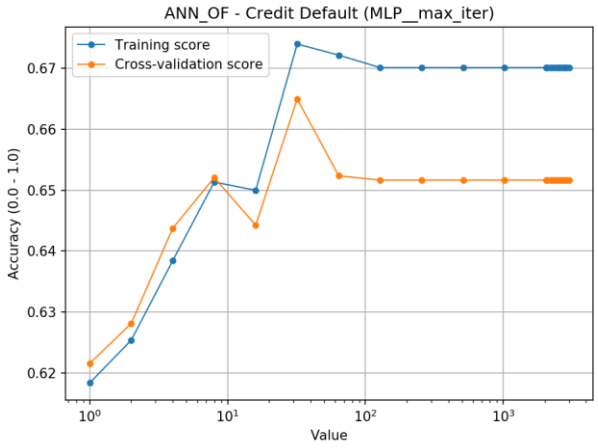
Both datasets show close together training and cross-validation learning curves.

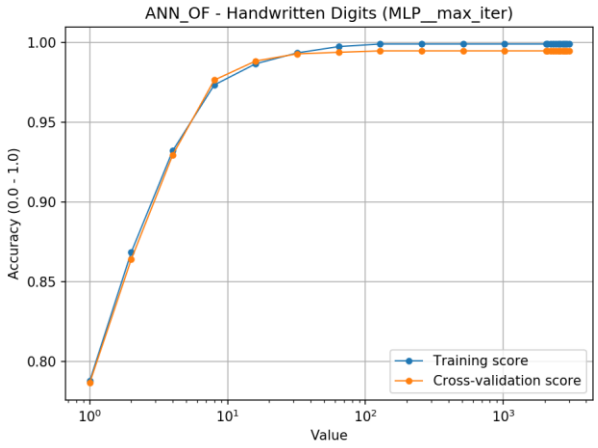
Alpha

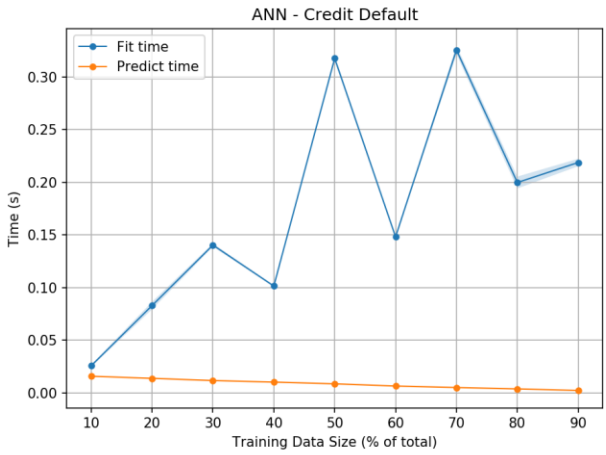


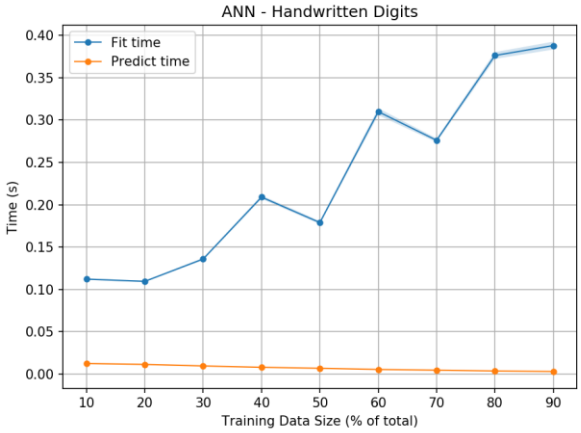


For both datasets, the optimal alpha (learning rate) is fairly small – 0.003 for credit default and 0.001 for pen digits recognition. As alpha increases towards 1, accuracy drops, more dramatically towards the end as larger weight modifications will decrease model accuracy with increasing likelihood of overstepping. However smaller alpha would be more computational expensive and by looking at the two graphs, it seems that as long as the alpha is not too close to 1, accuracy will not suffer greatly.





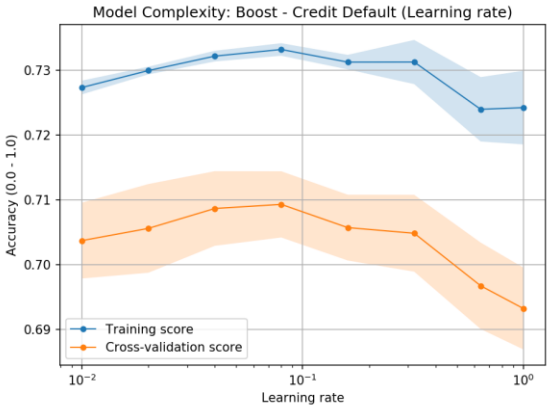


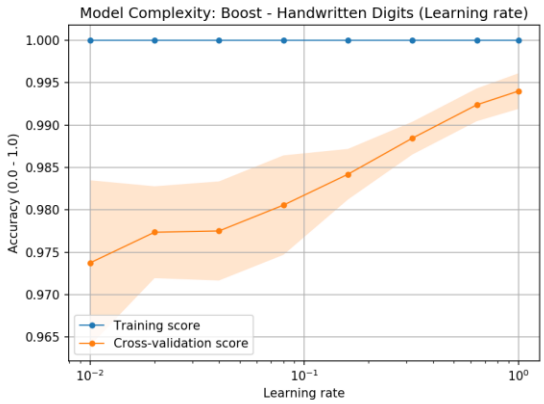


- You may use networks of nodes with as many layers as you like and any activation function you see fit.

Boosting

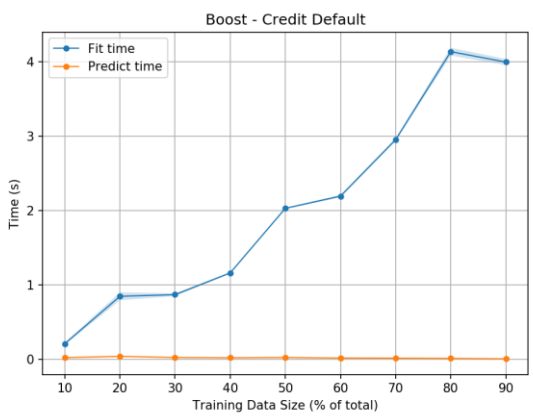
Model Complexity – Learning Rate

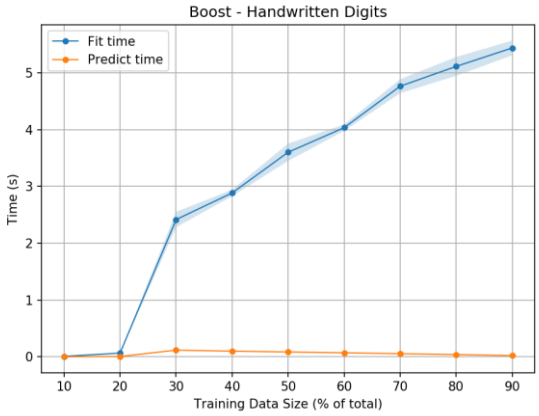




The optimal learning rate is 0.08 for credit card default dataset as it produces the highest accuracy for both training and cross-validation data. For pen digits recognition, learning rate is actually optimal when it is 1.

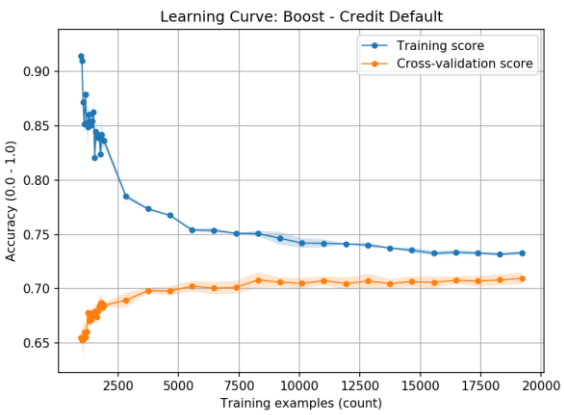
Time:

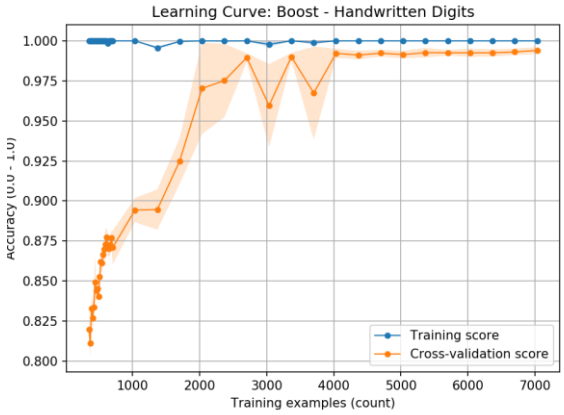




Boost is similar to Decision Tree, the time curve looks similar to that of Decision Tree as well. Since it is an eager learner, it spends increasing time on increasing amount of training data while predict time is constant/close to 0.

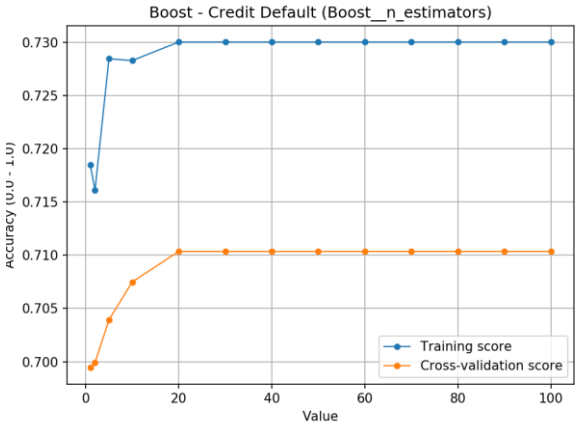
Learning Curve:

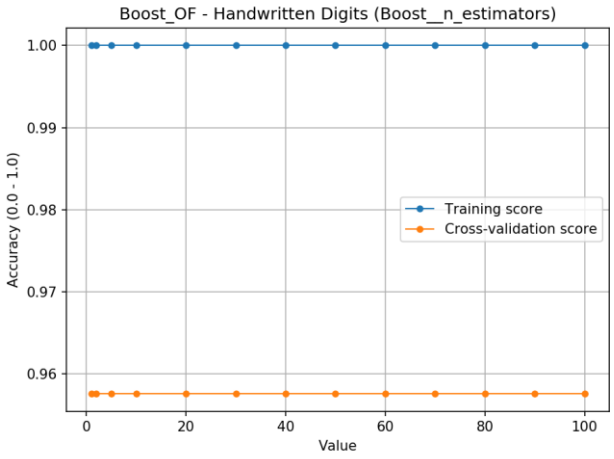




The learning curve for credit card default data show similar result to that of Decision Tree, boosting is prone to noise and that could be the reason why it did not have significant effect. For the pen digits recognition dataset, it is interesting that the result improved greatly towards the high end of training data used (an improvement from ~90% to ~100%).

Number of Estimators



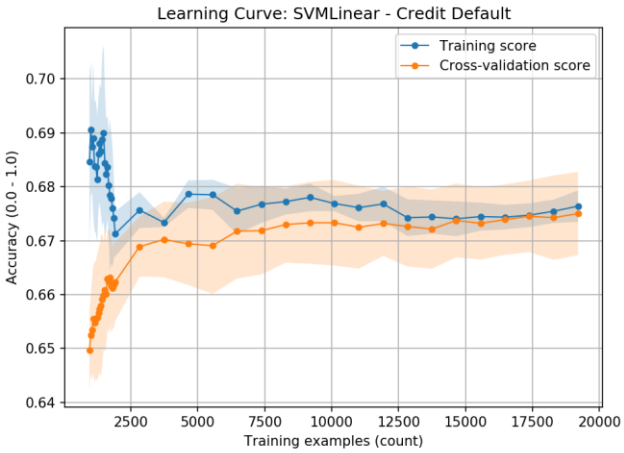


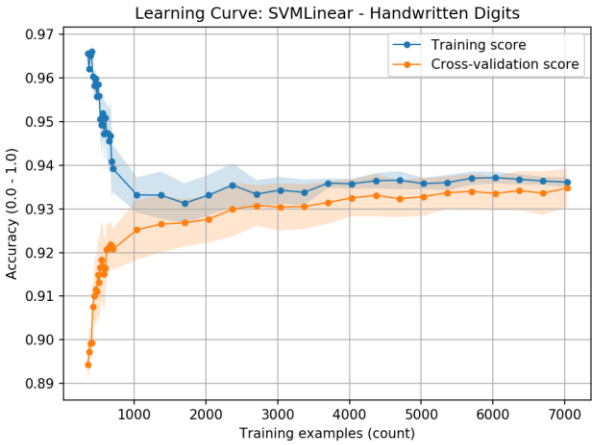
The optimal number of estimators for credit card default data is 30, above which training or cross validation accuracy stops improving, this means that having less variance in data improves performance (slightly). Interestingly, the number of estimators (trees/weak learners) does not change the accuracy of the training or validation data.

Support Vector Machine

For SVM I did both Linear as well as RGB

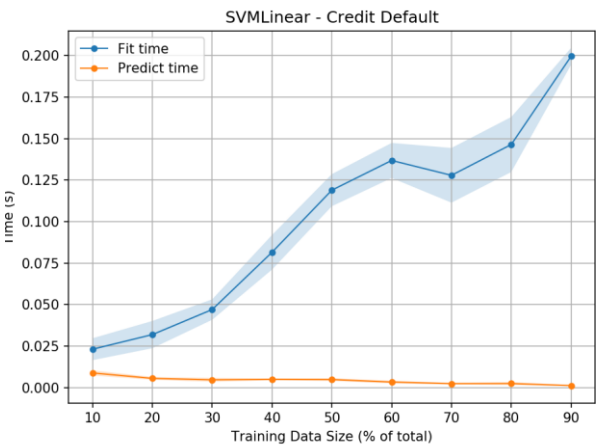
Learning Curve

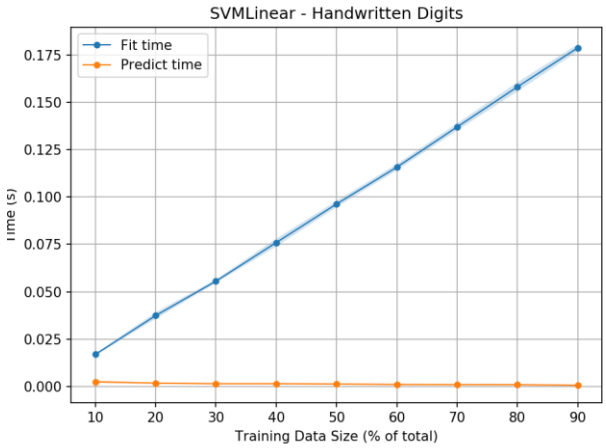




For both datasets, the learning curve converged fairly early on with a low number of training examples.

Time

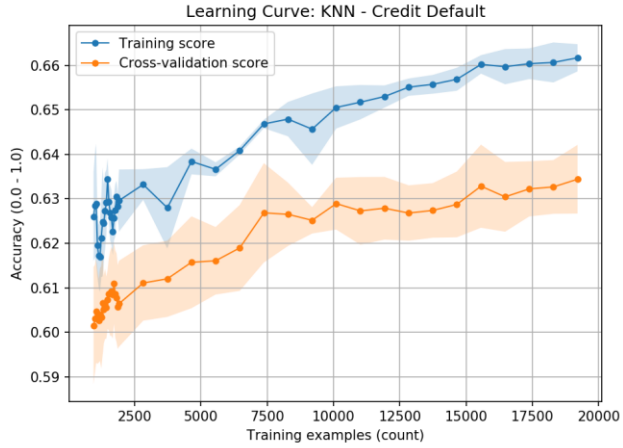


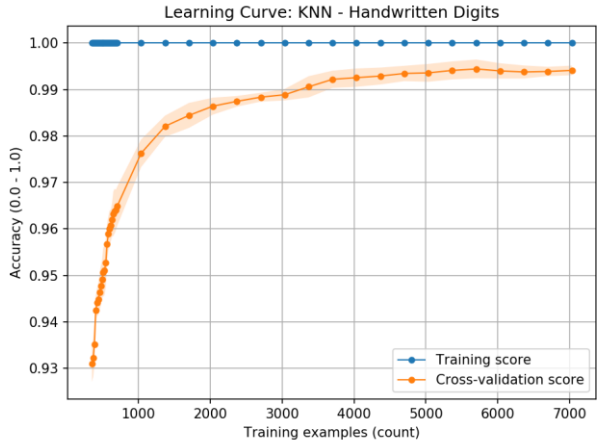


Since SVM is an eager learner, its fit time increases with the amount of training data used while the predict time stays fairly constant/close to 0.

K-Nearest Neighbor

Learning Curve:

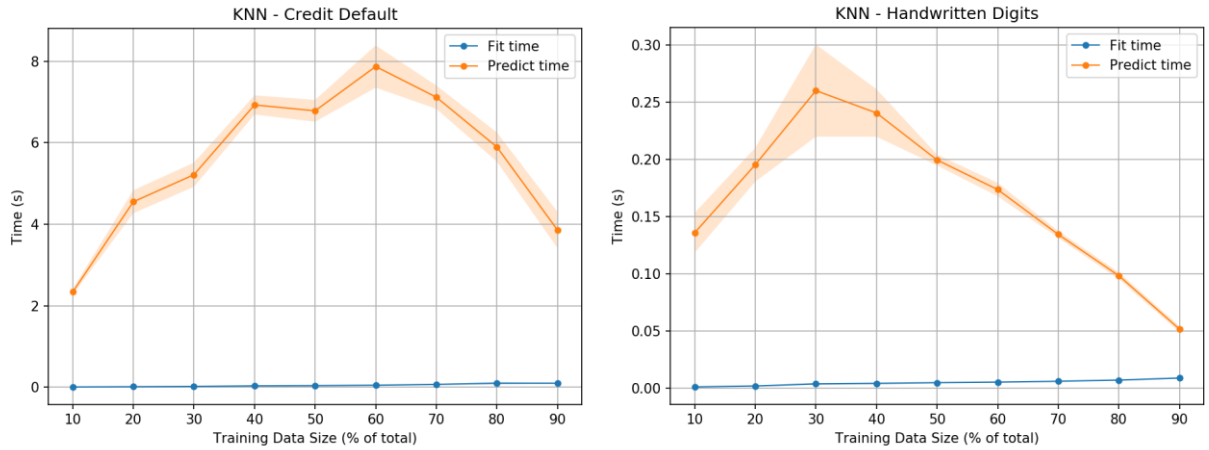




Credit Card Default – KNN model improved in accuracy for both training and cross validation as training examples increased. Training accuracy remained higher than cross-validation accuracy. This is because KNN tends to overfit because it does not generalize and there are (almost) infinite possibilities on what the features can be. Therefore, increasing the training size does increase accuracy but there is still a gap between training and cross validation because of the noise in the credit card default data.

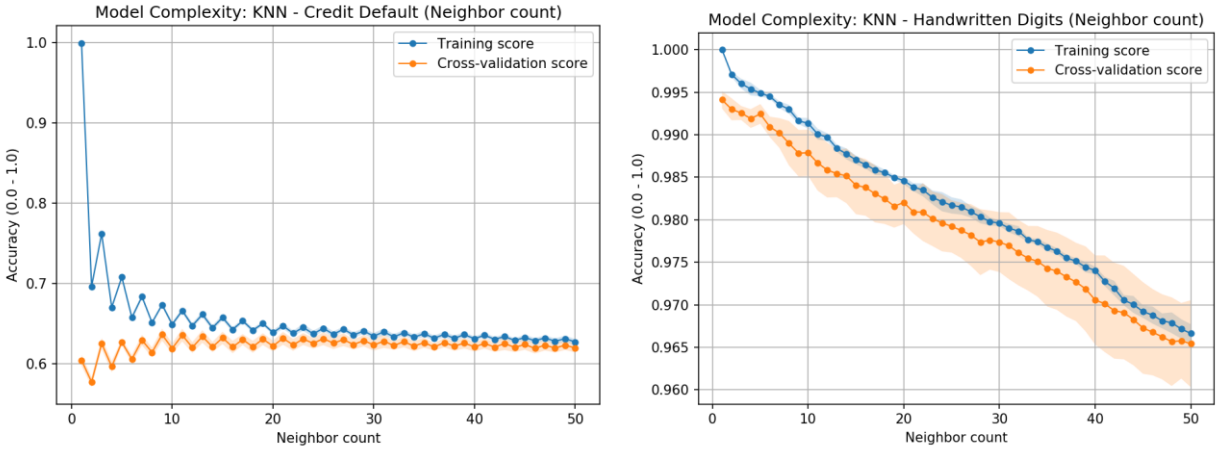
Pen Digits Recognition – Learning curve showed convergence with higher training examples for this dataset. Digit recognition is well suited for KNN model since there aren't as many unaccountable noises in handwritings. The closest looking digits will tend to be the correct digit.

Training Time:



Since KNN is a "lazy" learner (instance-based learner), fit time was 0 across all training sizes, for both sets of data. The predict time however, shows a parabolic shape. The predict time for KNN should be a function of ln of N + K, therefore, the shape of the predict time roughly follows a log normal distribution. Increasing initially as training data size increases as a percentage of total, because of the incremental effort needed to query more datasets; but then decreases towards the end as more data makes it easier for the model to find the best value.

Varying Neighbor Count:



Credit Default – Increasing the neighbor count had a large effect on model accuracy towards the beginning. However, the effect started to wane off as neighbor count increased over 13, and training/testing accuracy started to converge and remain fairly constant at ~60%. This shows that adding neighbor count reduces overfitting toward the beginning and as the neighbor count approaches the optimal number, the incremental effect decreases.

Pen Digit Recognition – Interestingly, best neighbor count parameter is actually 1. As shown in the model complexity chart Figure X, model accuracy decreases for both training and validation as neighbor count increases. This means that the best way to classify a digit is by looking at the digit that is most similar, which makes sense since there are not a lot of noise (especially in the way of unknown factors).

Summary

	Test Accuracy		Best Parameters		
	Credit Card	Pen Digits	Parameter	Credit Card	Pen Digits
Decision Tree	0.70	0.95	Class Weight: Criterion: Max Depth:	Balanced Entropy 5	None Gini 12
K-Nearest Neighbors	0.64	0.99	Metric: # Neighbors: Weights:	Euclidean 13 Uniform	Manhattan 1 Uniform
Boosting	0.71	0.99	Max Depth Learning Rate # Estimators	7 0.08 30	10 1 100
SVM	0.67	0.93	C Class Weight Max Iter Tol	0.251 Balanced 42 0.01000001	2.251 Balanced 114 1e-08
Neural networks	0.66	0.99	Activation Alpha Hidden Layer Sizes Learning Rate	Relu 0.0003162 (11, 11, 11) 0.128	Relu 0.0001 (32, 32, 32) 0.004

In summary, the two different datasets represent two distinct classification problems. The credit card default dataset seems to be a much "harder" problem. The test accuracy across all five models are in the range of 64 – 71%, implying that there are a lot of noise in the data. The model at the end of the day is only as good as the data, and further effort to improve accuracy could mean obtaining additional datapoints, feature engineering or trying an ensemble method. However, it is more important to look at the problem from a more practical stand point – perhaps a ~70% accuracy rate is more than good enough for credit card companies to flag potential clients in danger of default and have further communications with them to prevent default.

On the other hand, pen digits recognition seems to be an "easier" problem to classify. The accuracy across the five different models are in the range of 93 – 99%, signaling that the data has very little noise and the features are well defined. In particular, K-Nearest Neighbors, Decision Tree with Boosting and Neural Networks all achieved 99% accuracy. In real life, this problem might require a high degree of accuracy – the whole point of electronic handwriting recognition is to improve usability of interfaces, which requires minimizing the amount of time a user has to re-write in a more recognizable way.

Looking at the different models, it is clear that they perform differently on different datasets/problems. One consistency is that boosting achieve better accuracy over Decision Tree for both datasets (although

the improvement is slight for Credit Card Default), re-affirming the usefulness of sub-sampling and averaging. In fact, boosting has the highest accuracy for Credit Card Default problem, due to its benefit of reducing overfitting, especially for more complex problems.

Reference:

Data Source: https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
Data Source: https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset
Code Source: https://github.com/cmaron/CS-7641-assignments/tree/12655c7fc3d26b6bddb6b4494863e8a927e842dd/assignment1