Supervised Machine Learning Report

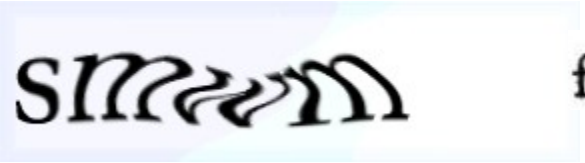
CS 7641 Machine Learning Assignment 1
Tao Peng

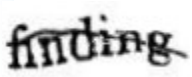
I. Datasets

Letter Recognition Data Set

This data set is from the UCI Machine Learning Repository [1]. Its purpose is to recognize distorted 26 English capital letters. Each letter is displayed by a large number of black-and-white rectangular pixels. Each letter has many different shapes, which are based on 20 different fonts and each font was randomly distorted. To identify those letters, 16 numerical attributes were used. Each attribute describes one feature of the shape of the letter. For examples, the overal horizontal position of the letter, the total width, the mean x of the pixels, the mean x * x * y of the pixels. The data has 20,000 instances and 16 attributes. There is no missing values. I used 70% of the data as training sample and the rest 30% for testing.

Why this data set is interesting for practical use? The most important use may be to recognize the Captcha, which is a combination of distorted letters used to test whether or not the user is human. The followings are examples of three different types of Capthas [2]:

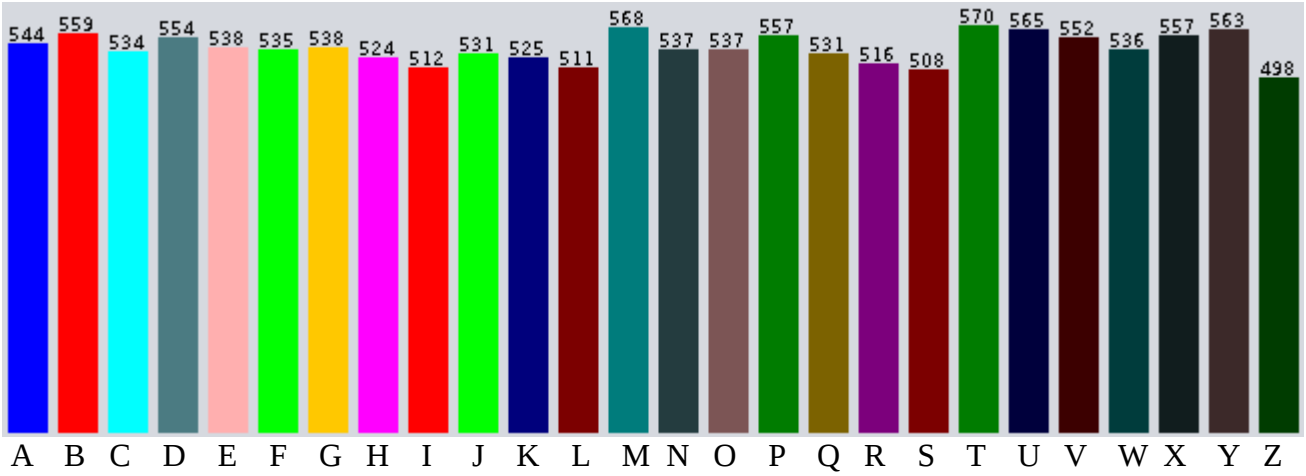




In order to recognize those letters, we first choose a large number of training sample and let the computer to learn from the 16 attributes of the training sample using different machine learning algorithms, and then we can test them and apply them to practical cases. However, this method of recognizing Captha may be limited if the letters are highly distorted, or if there is some extra curves crossing the letters (the middle example above), or if the letters are very crowded and enter each other's space (the right example above).

Why this data is also interesting for machine learning study? Because it works well under most of the algorithms. I think this is because of the large amount of sample available for the computer to learn, and relatively simple letter shapes (compared to the Captha shown above). However, there is still difference of the accuracy between different algorithms, making it possible to compare among them. Runing time can also be studied by comparing to small data set.

The following is the letter distribution of the training sample:



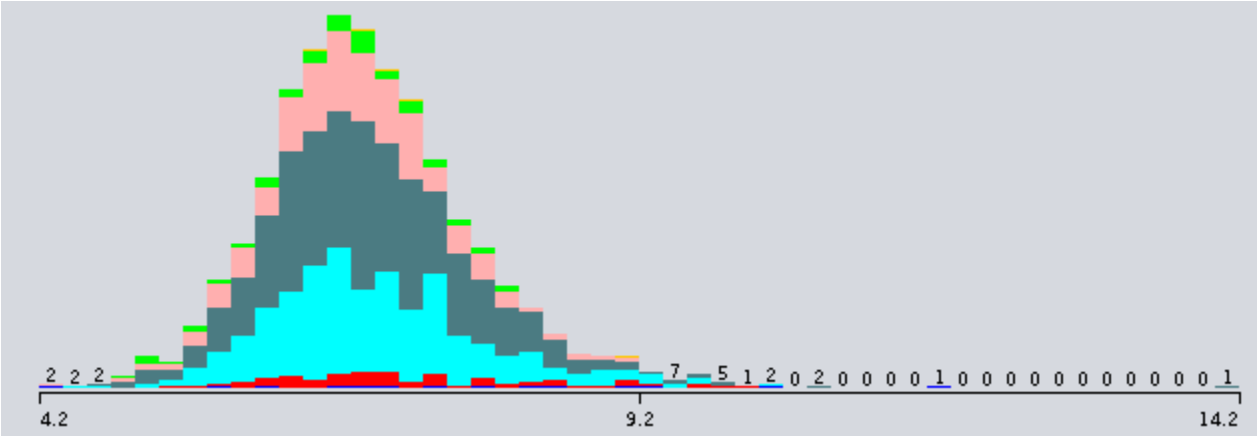
Wine Quality Data Set

This data set is also from the UCI Machine Learning Repository [3]. It is to determine the sensory quality of the Portuguese “Vinho Verde” wine. There are 11 classes of the wine quality (score between 0 and 10), which are determined from the 12 attributes. The attributes are the physicochemical feature of the wine, such as fixed acidity, residual sugar, pH. There are two data sets, one for red wine and the other for white wine. Because we already have a large data set of the letter recognition, we can use a small data set here to compare. So I only used the white wine data set, which contains 4898 instances and each instance has the 12 attributes mentioned above. I used 70% of the data as training sample and the rest 30% for testing.

This data set is interesting in practice. The quality of the wine is based on the sensory outcome, so if we do not use machine learning, we may have to destroy the wine by tasting it in order to determine its quality, which is not realistic in actual production process. Moreover, it may be quicker to determine the quality by running our models in computers than spending time tasting them one by one.

This data set is also interesting for machine learning. It does not work so well as the letter recognition data set, the running time is quicker than letter recognition, making it convenient to compare between these two data sets. It also behaves slightly differently under different algorithms, making it possible to compare among these algorithms.

The following is the quality distribution of the training sample:



Tools used

I used Weka for all the analysis below.

II. Decision Tree

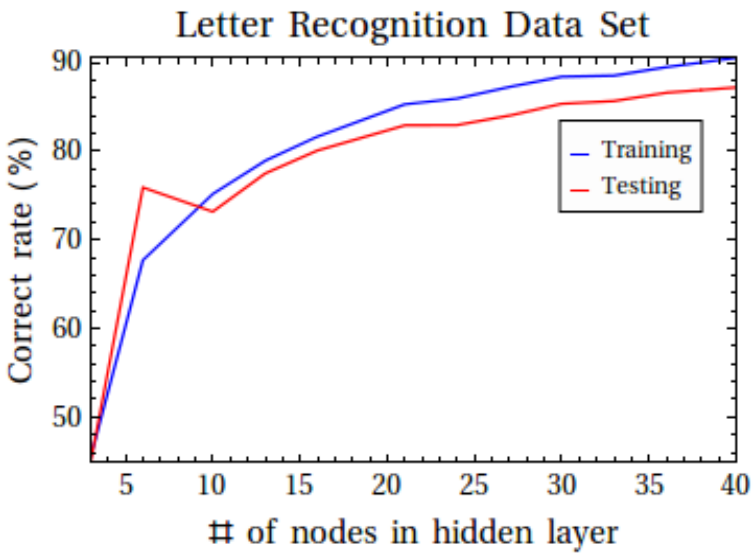
Letter Recognition Data Set						
Prune state	Confidence	Tree size	Training correct %	Testing correct %	Train time (seconds)	Test time (seconds)
pruned	0.1	1829	95.2857 %	86.0833 %	0.89	0.86
pruned	0.25	1953	95.7929 %	86.0667 %	0.97	0.92
pruned	0.4	1995	95.9357 %	86.0833 %	1.12	1.1
unpruned	--	2135	96.1286 %	86.1 %	0.91	0.92
5 fold cross-validation	0.25	1953	85.7857 %		1.00	
10 fold cross-validation	0.25	1953	86.3786 %		1.23	
20 fold cross-validation	0.25	1953	86.9571 %		1.00	

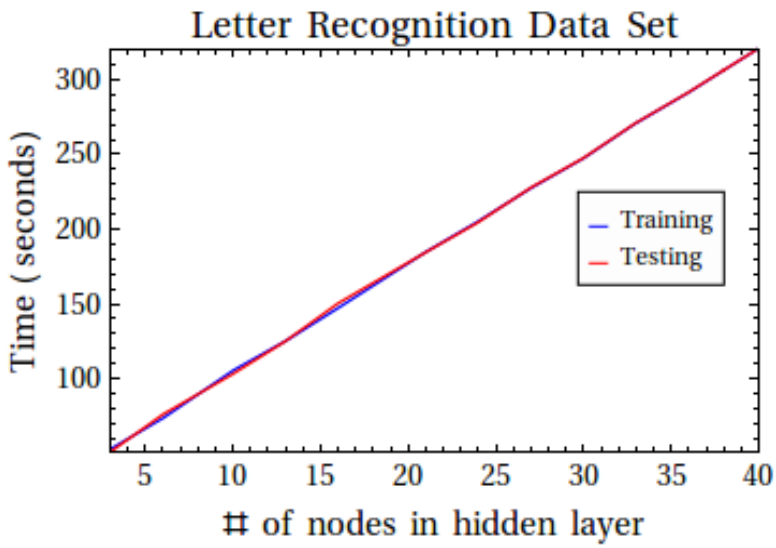
Wine Quality Data Set						
Prune state	Confidence	Tree size	Training correct %	Testing correct %	Train time (seconds)	Test time (seconds)
pruned	0.1	855	87.1391 %	44.7243 %	0.17	0.18
pruned	0.25	1131	91.7177 %	42.8863 %	0.16	0.18
pruned	0.4	1143	91.8927 %	42.9544 %	0.18	0.19
unpruned	--	1205	92.2718 %	43.0905 %	0.11	0.11
5 fold Cross-validation	0.25	1131	56.0805 %		0.17	
10 fold cross-validation	0.25	1131	57.6553 %		0.16	
20 fold cross-validation	0.25	1131	58.2386 %		0.21	

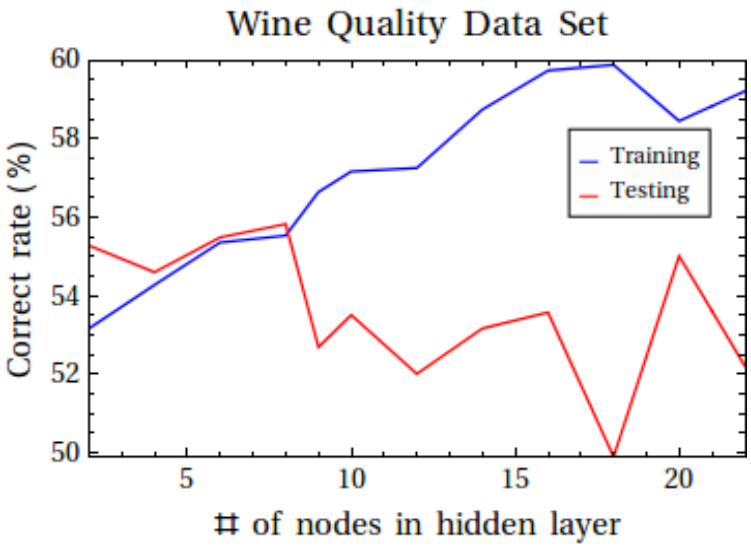
We first look at the correctly classified rate. The wine quality data has lower correctly classified rate, I believe this is because it has fewer data samples for training. For the wine quality data sets, the cross-validation has better correct rate than non-cross-validation cases, I would say this is because cross-validation helps to limit overfitting. The wine quality dataset does not have many data points, so we may rely on some error data points, making overfitting possible. And we see the more folds there are, only slightly better work cross-validation does. I think this may be because more subsamples can help better cross check among each other, so the higher correct rate. But since as long as we can limit overfitting we can do much better, so the number of folds is not so important. For the letter recognition data, cross-validation does not quite help, this is because we have so many data in this case, the error data points become less important so there is much less overfitting. And therefore cross-validatoin does not quite help to improve the accuracy in this case.

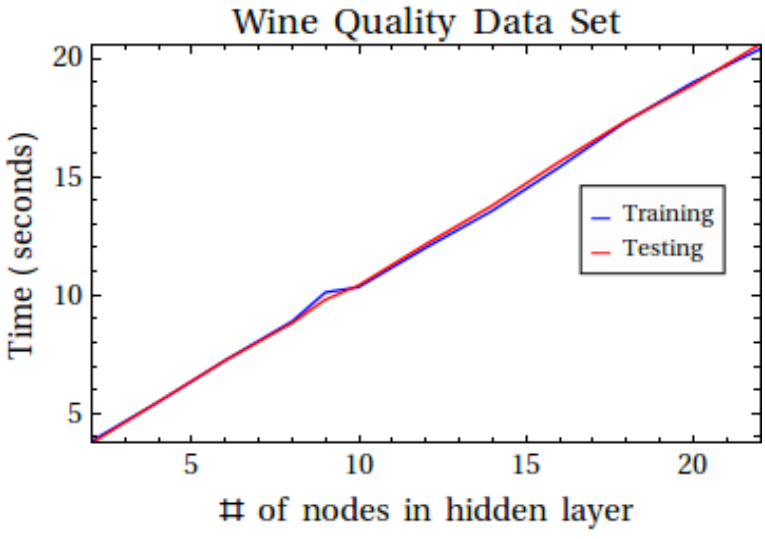
And then for the running time. For the letter recognition data, less pruning leads to longer training and testing time. I think this is because less pruning allows bigger tress and it needs longer time to process bigger tress. However, unpruned tree is the biggest but it takes shorter time than most of the pruned trees, this may be because it does not need to spend time on the pruning process. In contrast, for the wine quality data, there isn't much difference between the training and testing time of different levels of pruning, and I think is is because the trees are smaller than the letter recognition case, so it can be done very quickly and thus hides the differences in actual running time and makes fluctuaton in computing time important. But the unpruned tree still takes shorter time, and this again may be because there no need to spend time on pruning. The cross-validation running time is not longer or shorter than other cases, because in cross-validatoin we partioned the samples but we still need to analyse all of them as in other cases.

III. Neural Networks









For neural networks, I choosed to do one hidden layer but with different number of nodes in the hidden layer. Our first observation is that for the Letter Recognition Data Set, the correct rate increases as the number of nodes in the hidden layer increases. I think this makes sense because the more nodes in the hidden layer, the more weights we have, so the more degree of freedom we have. In the training phase, we can adjust those weights to better fit the training data. So we have a higher correct rate.

However, if we look at the correct rate of the Wine Quality Data Set, we see something different. The training sample does better with more nodes in the hidden layer, while the testing sample does worse. So this is overfitting. Why do we have overfitting in this case? Because as we have more nodes in the hiddern layer, we can have arbitrarily complicated networks so that we can fit anything including the noise. That's a bad thing and the more hidden nodes we have, the “better” we fit the noise, and the worse we do in our classifying work. In the Letter Recognition Data Set, we did not have overfitting, that's again because we have so many data points in this case that the noise is diluted by the normal points. So in practice, if we want to avoid this overfitting, as we learned in lecture, we may have to set some bound number for the hidden nodes and the number of layers, so that we have a constrained network and be safe from overfitting.

Then for the running time, we see that more hidden nodes leads to longer running time. This is expected because we have more complicated network. One interesting thing we notice is that the amount of time grows linearly as the number of hidden nodes increases. I think this is because each of the hidden nodes are doing the same thing: when building the model, find the same number of optimized weights, and when calculating the outputs, take the same number of weights and multiplying by the factors and get the output. So the amount of work in each hidden node is the same, therefore the total running time grows linearly.

IV. Boosting

Letter Recognition Data Set						
Prune state	# of iterations	Confidence	Training correct %	Testing correct %	Train time	Test time
pruned	10	0.1	100	94.9	9.59	9.75
pruned	10	0.25	100	94.7	9.55	9.44
pruned	10	0.4	100	94.6667	9.66	9.28
unpruned	10	--	100	94.6667	7.86	8.34
pruned	30	0.1	100	96.45	31.71	30.69
pruned	30	0.25	100	96.35	30.13	30.3
pruned	30	0.4	100	96.05	30.33	30.27
unpruned	30	--	100	96.5833	26.81	25.79

Wine Quality Data Set						
Prune state	# of iterations	Confidence	Training correct %	Testing correct %	Train time	Test time
pruned	10	0.1	100	48.128	2.06	2.31
pruned	10	0.25	100	47.9918	2.03	2.02
pruned	10	0.4	100	45.405	2.1	2.06
unpruned	10	--	100	44.1116	1.48	1.98
pruned	30	0.1	100	48.7406	7.37	6.32
pruned	30	0.25	100	51.1913	6.46	6.3
pruned	30	0.4	100	49.6937	6.24	6.4
unpruned	30	--	100	48.128	4.46	4.94

Compared to decision tree without boosting, we see that the correction rates for both Letter Recognition and Wine Quality Data Set are improved. This is expected since boosting put more emphasis on harder problems and has more than one iterations. What else we notice is that the improvement of correction rate by using boosting is bigger in the Letter Recognition Data Set is bigger than the Wine Quality Data Set. I think this is because the Letter Recognition Data Set has more data, so there may be more mistakes in the first few iterations, so that the boosting algorithm can put emphasis on more mistakes and thus boosting becomes more effective than the case of Wine Quality Data Set when there may be less mistakes.

There are also two strange points (both underlined). The first one is the testing correct rate of the unpruned tree for the Letter Recognition Data Set. This rate is bigger than those of pruned trees. I double checked it by running it again and got the same result. The second strange point is the correction reate underlined for the Wine Quality Data Set. It is smaller than the rates of trees with higher confidence (less pruned). While explaining these two points is beyond what I could accomplish here, I would guess they may be somewhat related to overfitting, or just because of fluctuation of the correct rate because they do not differ very much from others.

For the running time, we can see that boosting takes much longer time than non-boosting trees. This is expected because boosting takes more iterations. Another thing we noticed is the running time is proportional to the number of iterations. I think this is because each iteration almost does the same thing.

V. Support Vector Machine

Letter Recognition Data Set					
Kernel type	Parameter	Training correct %	Testing correct %	Training time (seconds)	Testing time (seconds)
polynomial	degree=1	86.1	84.4167	8.99	8.68
polynomial	degree=2	98.9643	94.35	11.47	10.91
polynomial	degree=3	100	94.6667	11.94	10.52
RBF	gamma=0.1	99.7571	96.95	38.17	37.84
RBF	gamma=1	100	48.65	70.06	71.89
RBF	gamma=10	100	12.15	127.42	128.31
Sigmoid	gamma=0.1	4.0714	3.7667	35.15	36.38
Sigmoid	gamma=1	4.0714	3.7667	36.42	36.13
Sigmoid	gamma=10	4.0714	3.7667	36.91	37.81

Wine Quality Data Set					
Kernel type	Parameter	Training correct %	Testing correct %	Training time (seconds)	Testing time (seconds)
polynomial	degree=1	49.4313	55.548	170.39	168.71
polynomial	degree=2	50.802	47.5834	538.42	533.59
polynomial	degree=3	45.5235	42.5459	990.55	1000.02
RBF	gamma=0.1	85.0977	48.128	5.67	5.23
RBF	gamma=1	99.3876	50.5786	8.78	8.57
RBF	gamma=10	99.9708	49.8979	10.14	10.4
Sigmoid	gamma=0.1	42.753	49.8298	3.52	1.76
Sigmoid	gamma=1	42.753	49.8298	1.38	1.69
Sigmoid	gamma=10	42.753	49.8298	2.04	1.29

For SVM, I chose the following three kernels:

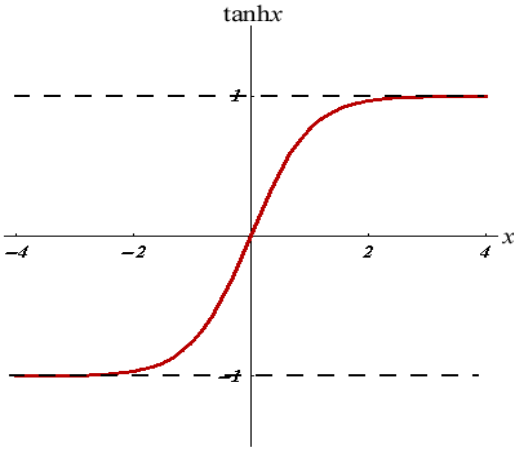
Polynomial: (gamma*u*v+coef0)^degree
Radial Basis Function (RBF): exp(-gamma*|u-v|^2)
Sigmoid: tanh(gamma*u*v+coef0)

First, for the polynomial kernel, we see that as the degree is higher, the Letter Recognition Data Set has better correct rate. This is because we have more freedom to fit the data with higher degrees. But the Wine Quality Data Set has worse correct rate with higher degree. I think this is again because of

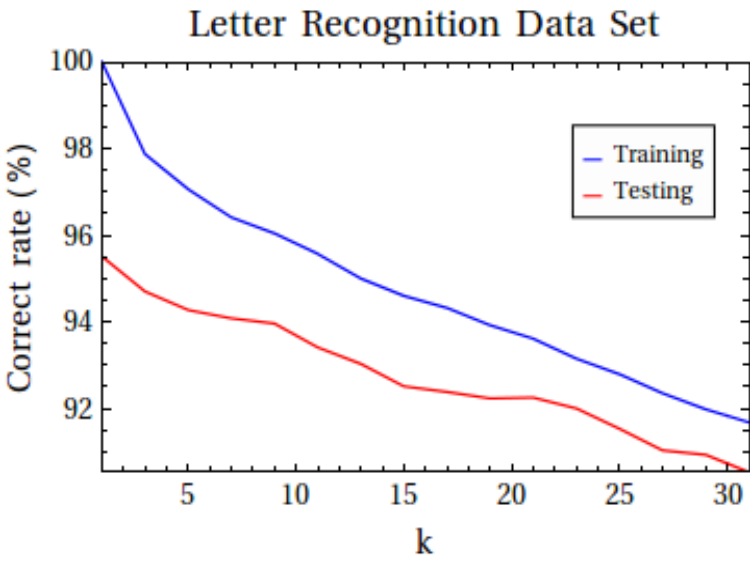
overfitting. We have less data in Wine Quality Data Set. So noise becomes more important, making overfitting more likely than the Letter Recognition Data Set. Higher degrees also need to run longer times, especially for the Wine Quality Data Set. We notice that the running time for Wine Quality Data Set is really long (1000 seconds for degree equals 3), and it is much longer than the running time of all other algorithms. Perhaps this is because of overfitting, the computer was struggling to fit with both the normal and noise data as accurately as possible, which seems to be a hard work. And the high degree (high power) itself requires a lot of computing.

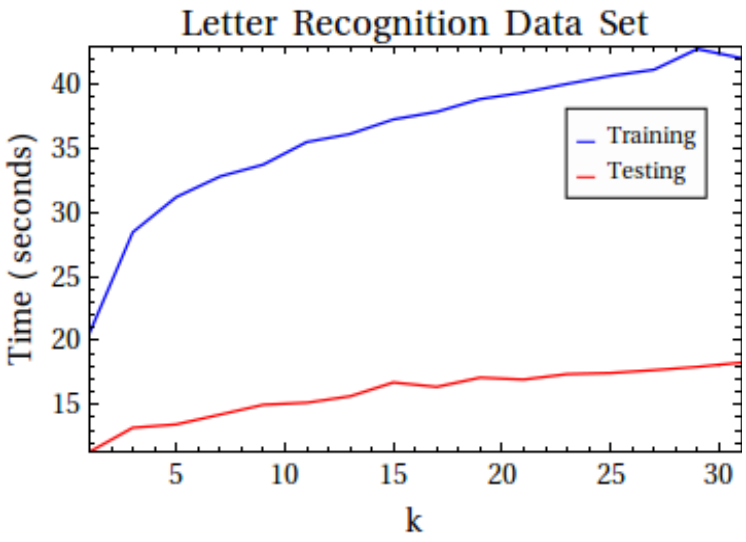
Second, for the Radial Basis Function (RBF), we see that for the Letter Recognition Data Set, the correct rate decreases very quickly as gamma grows. I think this is because the parameter gamma is inside the exponential function, so as gama grows, the exponential decreases quickly to zero. A constant zero kernel function is obviously a bad choice. So the correct rate drops quickly. However, the Wine Quality Data Set behaves differently: correct rate does not quite change as gamma increases. While it is not clear to me why it is like this, I would guess that this is because the Wine Quality Data Set has much fewer data, so it is much easier to fit those data, even with an almost constant zero kernel function.

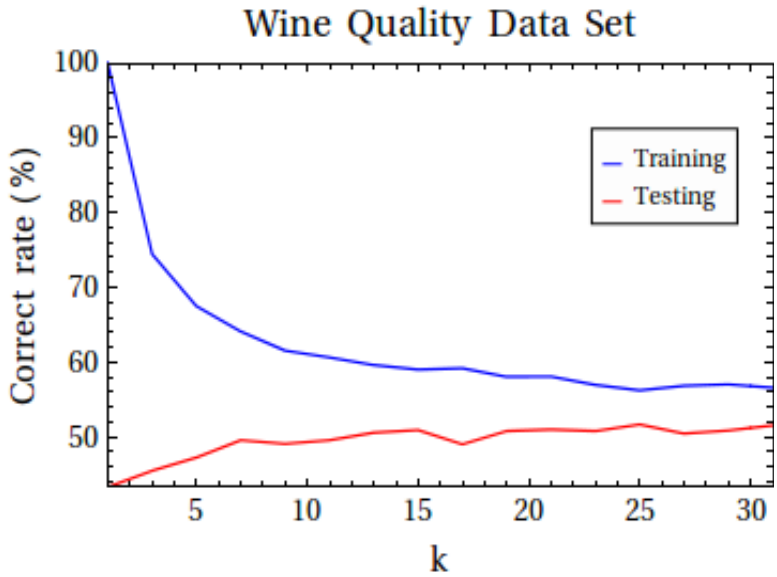
Finally, for the Sigmoid kernel, the correct rates for the two data sets do not change as the parameter gamma changes. I think this is because the Sigmoid kernel is almost a constant as the parameter becomes large. The following is the curve of the tanh(x) function in the Sigmoid kernel. So a change in the gamma does not bring any big difference to everything, including the correct rate. And again, as we concluded above, a constant kernel is a bad kernel, and this may be the reason why we have very small correct rate (~3.7%) for the Letter Recognition Data Set.

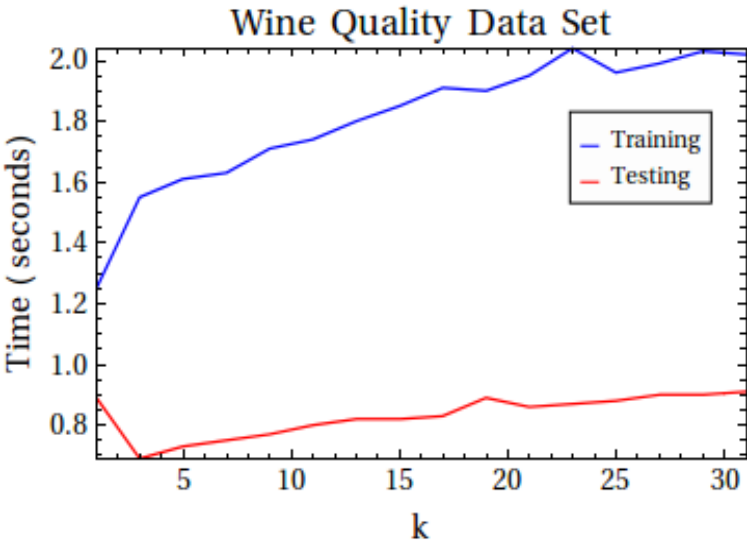


VI. K-Nearest Neighbors









First, for the correct rate. If we look at the curves for the Letter Recognition Data Set, we see that the correct rate drops as *k* grows, where *k* is the number of neighbors. This can be understood by using an interesting theoretical result [4]: If training set size $n \to \infty$ and $k \to \infty$ and $k/n \to 0$, error approaches minimum. So in the Letter Recognition Data Set, we have a large amount of data, which satisfys the condition of $n \to \infty$, and as k is smaller, $k/n \to 0$, so we have smaller error, which means bigger correct rate. However, for the Wine Quality Data Set, what we see is that the correct rate (of testing) is almost a constant. The way to understand this, is to look at the training curve. We see that as *k* grows, the correct rate of training drops, which is consistent with the above theoretical result. But we also know that the correct rate of testing can not be bigger than that of the training. So when *k* is large (> 20), there is an upper bound for the correct rate of testing (~60%). So the testing curve is compressed below the training curve, which is not much space, making the testing curve looks like a constant.

Then we look at the running time. We see that the running time grows as k grows, which makes sense because larger *k* requires more calculation. But what is more interesting is the components of time.

Take the training process when *k* equals 1 as an example, the running time consists of the time taken to build model and the time taken to test model on training data. For the Letter Recognition Data Set, time taken to build model is shorter or aroud 0.02 seconds, while the time taken to test model on training data: 20.54 seconds. Similar things happen to the Wine Quality Data Set, time taken to build model is aroud 0.02 seconds, while the time taken to test model on training data: 1.25 seconds. That means for both data sets, the time taken to test model on training data is much longer than the time taken to build model. This is a sign of lazy learner: the kNN algorithm takes less time learning than query, which is consistent with what we learned in the lecture.

Summary

From the two different data sets, we see that they behave differently. And different algorithms also have different behaviors, in terms of different correct rate and running time. So there is no best algorithm. For different data sets, we need to consider different algorithms. Also, most of the behaviors can be explained or understood using the theories we learned from lecture, which is a good exercises to help us understand the lecture materials better.

References

[1] https://archive.ics.uci.edu/ml/datasets/Letter+Recognition
[2] https://zh.wikipedia.org/wiki/%E9%AA%8C%E8%AF%81%E7%A0%81
[3] https://archive.ics.uci.edu/ml/datasets/Wine+Quality
[4] Data Mining with Weka, Ian H. Witten. Youtube video: https://www.youtube.com/watch?v=zjYUYJ2b4r8&list=PLm4W7_iX_v4NqPUjceOGd-OKNVO4c_cPD&index=19