CS4641 Machine Learning Assignment1

Example1: Iris

Description:

This is the database that used in the pattern recognition literature, use different attributes find in different pictures try to predict the classification of different kind of iris. Which means that we can distinguish different kind of iris with different attributes.

Dataset is from the UCI Machine Learning Repository. Iris Data Set.

Method used:

scikit-learn for Decision tree, boosting, support vector machines, k-nearest neighbors
MATLAB for Neural Network.

Attribute:4

Sepal length in cm, Sepal width in cm,
Petal length in cm, Petal width in cm,

Classification:1(3 types)

Iris Setosa, Iris Versicolour, Iris Virginica

Data preprocessing:

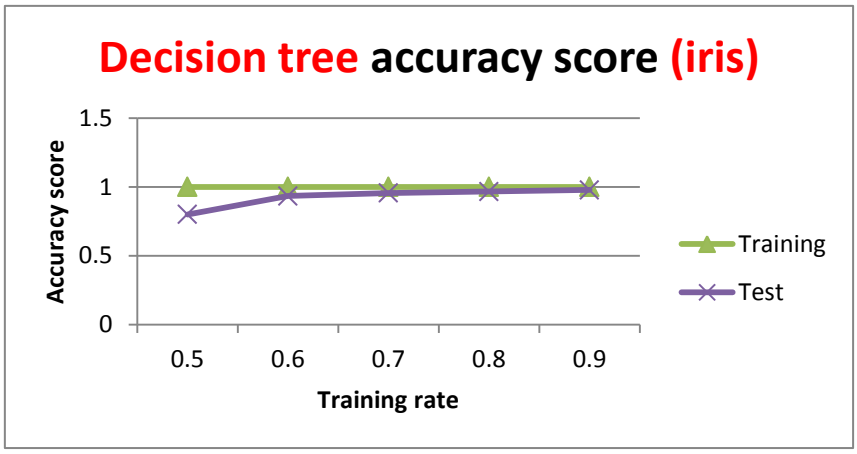
To be honest, according the dataset I obtained, and since it shows that it do not have missing values and with only four attributes, there's no need for me to do any data preprocessing work. And in order to show the result of data preprocessing, we can simply use the first two attributes to make the classifier better.

(training rate means the percentage of the dataset used as the training dataset)

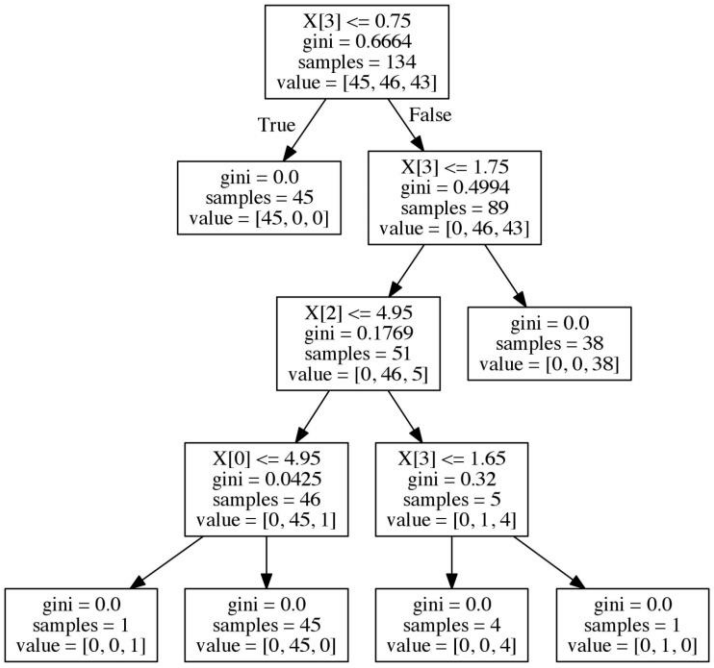
1. **Decision trees:**

   The Decision tree Classifier is kind of simple, we follow the best feature first criteria, and can obtain some useful conclusions.

   Pruning: in my code, I just use constrain the maximum leaf nodes to 10 and the maximum depth of the tree to 5 as the basic strategy for pruning.



Then divide the dataset into training and test dataset, we can see from the above plot, with the increasing rate of the training example (in my case, I set up the training rate instead of the number of training examples as the independent variable.), the accuracy score can also increase accordingly.

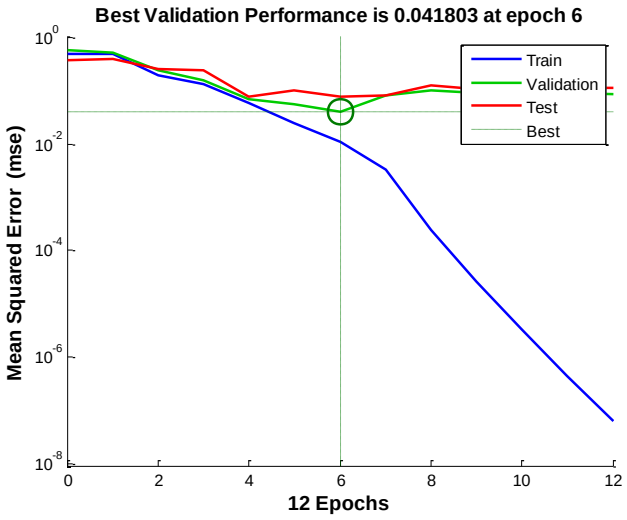


We can also generate some form of decision tree as the picture above, with some forms of pruning, the depth of the Decision tree can be as 5 layers and plus the root has 11 nodes in total.

2. Neural Network

For the Neural Network algorithms, instead of using scikit-learn in python, I turned to use MATLAB toolbox to implement the algorithms and evaluate the performance of the model, since the MATLAB neural network toolbox will use the cross validation automatically, so instead of compare the performance of training and test dataset, we can evaluate the performance of the model directly, and since neural network is a iteration method, so we can record the time needed

to realize the algorithms and do the evaluation.



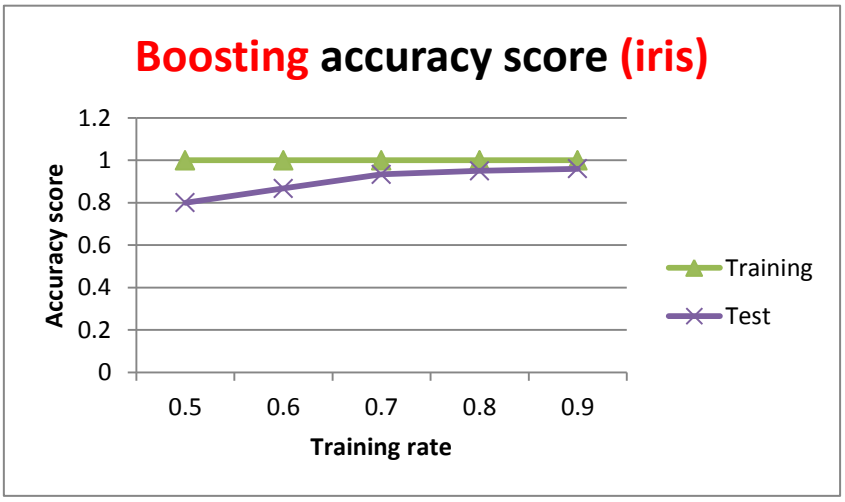
For the selection of the parameters, there are several key parameters that I want to evaluate, the number of hidden layers, proper training algorithms, epoch (number of iterations).

Below is the table used to evaluate the performance of different settings of the neural network, we can trace the table to find what is the best number of hidden layer to use and what is the most suitable Training algorithms and how many iterations (epoch) it may take to finish the experiment.

Index of the experiment	Performance	Number of hidden layer	Training algorithm	Epoch
1	0.041803	5	Levenberg-Marqudt	12
2	2.1172e-13	5	Bayesian Regularization	82
3	0.0261	5	Scaled Conjugate Gradient	55
4	0.016083	7	Levenberg-Marquardt	11
5	0.015002	9	Levenberg-Marquardt	11
6	0.056549	11	Levenberg-Marquardt	11
7	0.0072967	13	Levenberg-Marquardt	14
8	2.1286e-13	3	Levenberg-Marquardt	22
9	0.058206	1	Levenberg-Marquardt	59
10	0.12803	15	Levenberg-Marquardt	11

So based the table above, I can simply say in order to optimize the property of the neural network, I may use 7 hidden layers, using the Levenberg-Marquardt algorithms as the training algorithms and it will cause better results.

3. Boosting

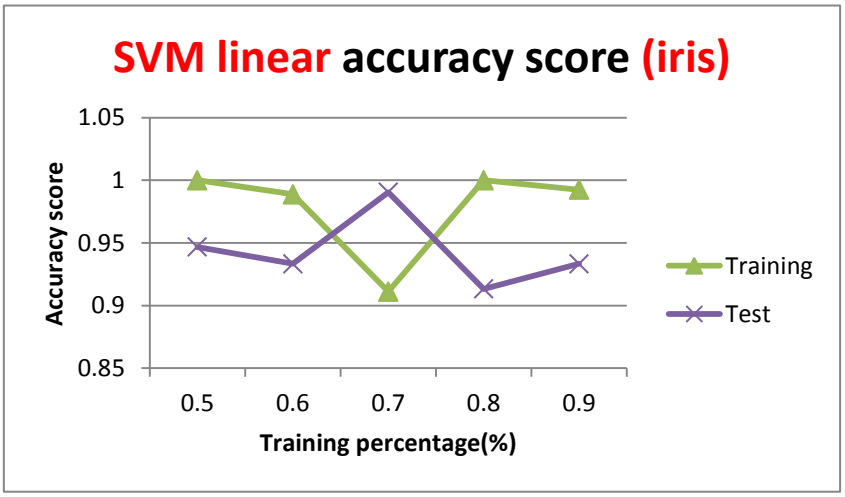


In my experiment for boosting, I simply use the Gradient boosting, which is an ensemble method combine several decision trees. We can see from the above plot that with the increase of the training rate, the accuracy score also increase and in the end converge to the 100% accuracy , which means that the Gradient boosting can really increase the effect of the Decision tree.

4. Support Vector Machines with kernels

For the Support Vector Machines, we can implement different kernels to make the algorithms more strong. Like below, I use the linear kernel, and try to test the accuracy score with different size of dataset. It is easy to see that with the increase of the training size, the accuracy score for the training set and the test set show some really wired pattern. Maybe it is because when we split the whole dataset into training and test dataset, the random seed we implement just accidently generate some abnormal data for the training set when the training rate is 0.7, so the accuracy score become kind of abnormal.

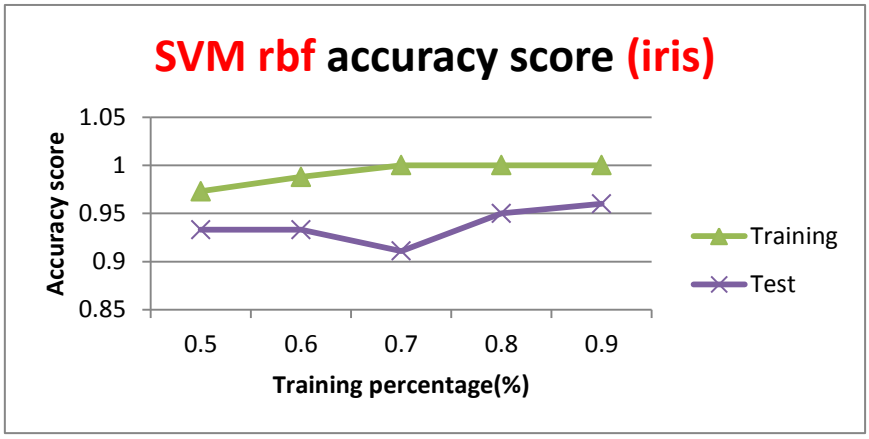
And theoretically, the accuracy score will increase with the increase of the training rate, but the whole trend for the curve is not increase, maybe that because the accuracy score for this example is already high enough, so can not increase again.



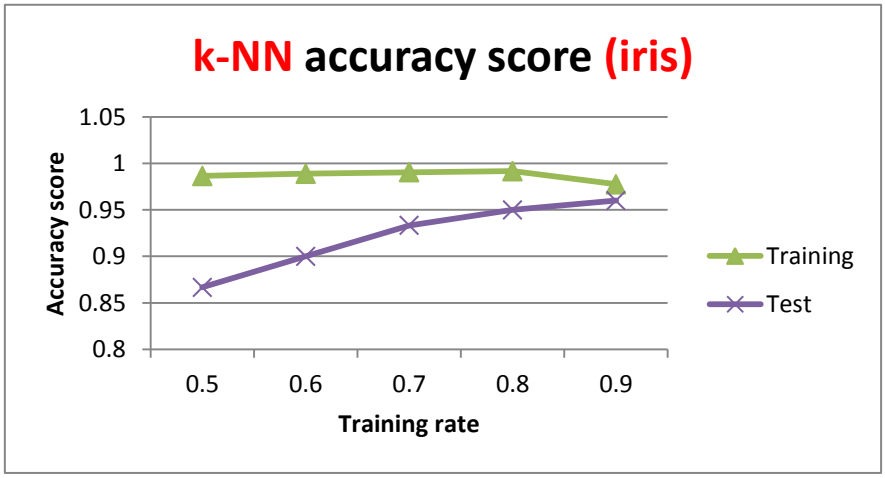
Same as the linear kernel Support Vector Machines, the rbf (radial basis function) Support Vector Machine function show an abnormal point when the training rate is 0.7, it maybe the

same reason as we discussed above, the random seed split generate some special case function for our dataset and give us some strange results.

As for the whole trend of the curve, it increase with the increase of the training rate, which make sense and should be like that.



5. K-nearest neighbors



Basically, the k-nearest neighbors algorithm can be regard as some kind of classification algorithms, the accuracy score increase with the training rate gradually as showed for the above plot. The training dataset always has high accuracy score while the test dataset has the score that shows a trend for increase.

Example2:Wine

Description:

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of the wines.

Attributes: 13

(1) Alcohol (2) Malic acid (3) Ash (4) Alcalinity of ash (5) Magnesium (6) Total phenols (7) Flavanoids (8) Nonflavanoid phenols (9) Proanthocyanins (10) Color intensity (11)Hue (12)OD280/OD315 of diluted wines (13)Proline

Classification:1 (3 types)

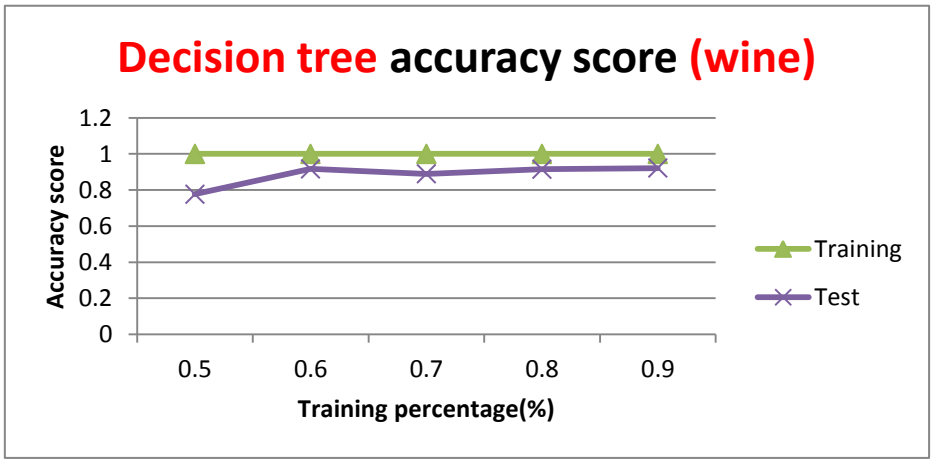
1, 2, 3

Data preprocessing:

Since according to the original literature, the dataset should has around 30 variables, so the 13 attributes are already the attributes that after some kind of reduce dimensions operation.

1. **Decision trees:**

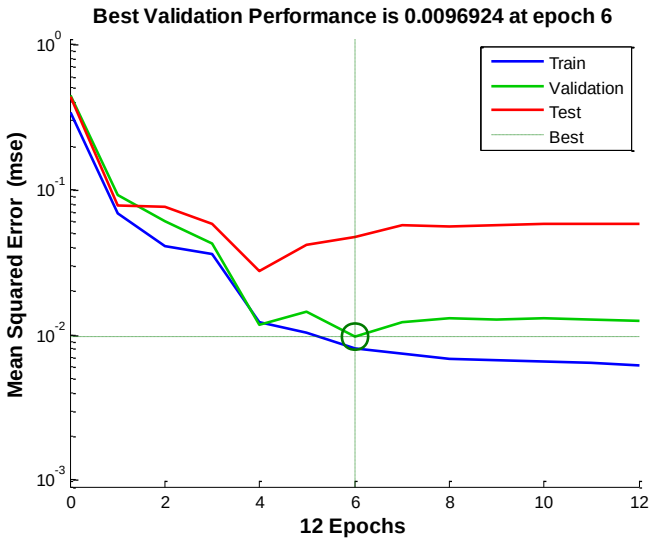
   The decision tree classifier is kind of simple, as the method used in the iris dataset, same pruning technics is used in this decision tree. We set the maximum leaf nodes as 10 and the maximum depths as 5, which can show some pattern for the decision tree just as the iris dataset, due to the space problem, just save the plot.



According to the plot above, it is easy to see that with the increase of the training size(which use training percentage to represent), the accuracy score also increase, while in the same time, the accuracy score for the training set remain in 1. For this experiment, I think it shows that the Decision tree as a classifier can work pretty good when we have only limited types of wine we need to make classification.

2. **Neural Netowrk**

   As the experiment did before, MATLAB will automatically do the cross validation for us the test the performance of the dataset and algorithms. Below is when I select the best parameters (I will discuss about it later) for the neural network, how it will perform in our dataset. As the plot below shows that after 12 times of iteration, we can conclude that the best performance will happen when the number of iteration is 6.

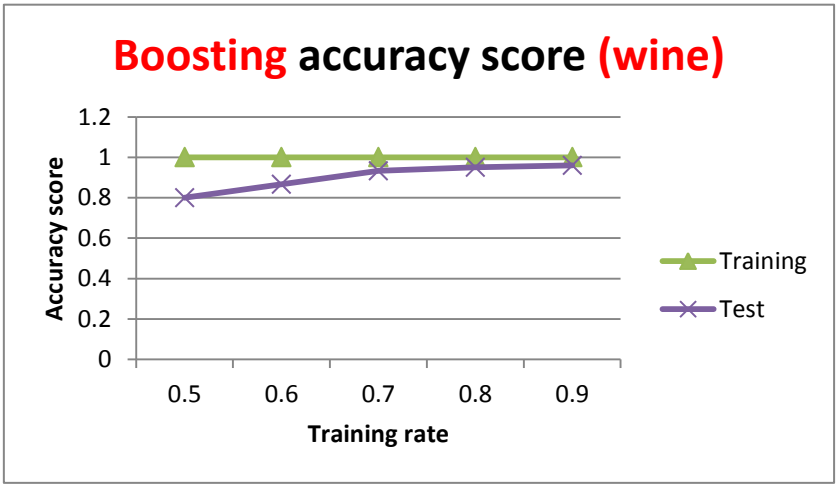


As for the best parameters we need to choose, the table below give us some clues about it. We can say that based on the evaluation of the performance, the best number of the hidden layers is 7, the best training algorithm is Levenberg-Marqudt for our dataset.

No	Performance	Number of hidden layer	Training algorithm	Epoch
1	0.041803	5	Levenberg-Marqudt	12
2	2.1172e-13	5	Bayesian Regularization	82
3	0.0261	5	Scaled Conjugate Gradient	55
4	0.016083	7	Levenberg-Marquardt	11
5	0.015002	9	Levenberg-Marquardt	11
6	0.056549	11	Levenberg-Marquardt	11
7	0.0072967	13	Levenberg-Marquardt	14
8	2.1286e-13	3	Levenberg-Marquardt	22
9	0.058206	1	Levenberg-Marquardt	59
10	0.12803	15	Levenberg-Marquardt	11

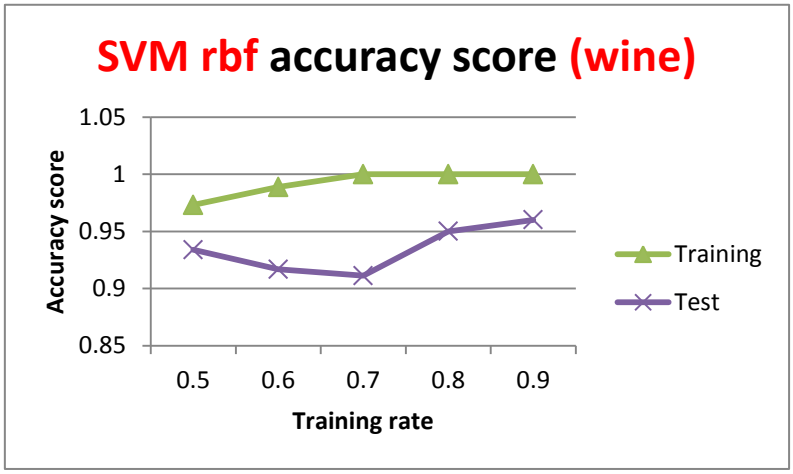
3. Boosting

Same discuss as the iris example, the boosting method increase the accuracy of the classification algorithm, and the accuracy score increase with the training rate for the test dataset, but the accuracy is always below the accuracy of the training dataset.

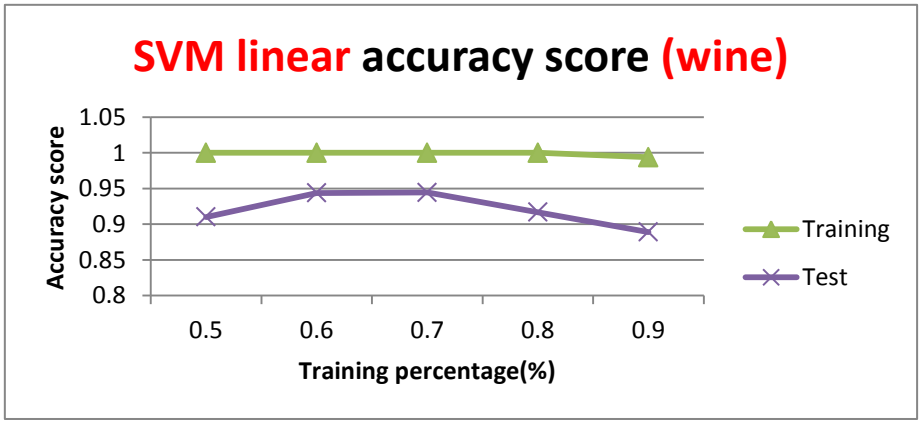


4. Support Vector Machines with kernel.

We can see from the curves that with the increase of the training rate, the accuracy score also increase, the accuracy score for the training dataset is always higher than the test dataset, and the abnormal performance of the test dataset may resulted from the way we split the dataset, randomly split it cause some unexpected trouble.

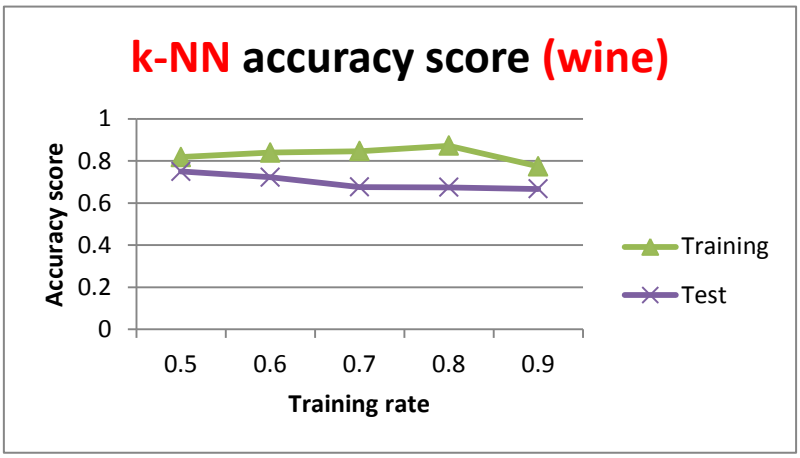


For the SVM linear kernel example, it show show some strange shape of the test curve, maybe caused by the overfitting of the training dataset when the training rate is aroung 0.6 or 0.7, and same as the above graph, the training accuracy score is always higher than the test dataset.

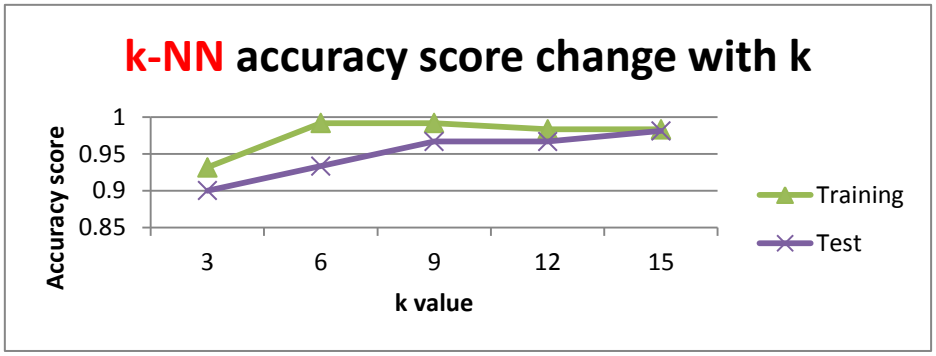


5. k-nearest-neighbors

According to the accuracy score, the k-nearest-neighbors algorithm maybe the last algorithm we want to use since it is around 0.8, but it is pretty stable even with the increase of the training rate, the accuracy score do no change that much. What's more, same as the other results, the test accuracy score is always lower than the training accuracy score.



Below graph shows the increase with k can influence the performance of the model, for my dataset, with the increase of k, the accuracy score also increase. I think maybe it is because my dataset for wine is kind of spread out, or more widely distributed, it is kind of the properties of the dataset.



Discussion:

Comparison of different types of algorithms.

According to different experiment of different dataset, we can come up with some useful conclusions to use maybe in the future.

1. The decision tree is more general, for two different datasets, it behave almost same the accuracy score for the test dataset is from 0.8 to 1.
2. For the Neural Network, since maybe because it is a relative complex method, so the accuracy is pretty high, but we need to implement more parameters to make it more efficient.
3. For the boosting method, it behave kind of the same as the Decision tree, because it is originally the ensemble of the decision tree.

4. For the SVM method, it has relative high accuracy(almost 0.94 accuracy score), but shows some really wired properties and really unstable , different kernels can cause big different of the results.
5. For the k-nearest-neighbors method, it highly related to the property of the dataset, especially highly rely on the number of attributes of the dataset.

Cross Validation

I did the cross validation for all the algorithms, but since we only have limited pages for the report, so save the tables in other files.

But the basic cross validation shows that with the increase of CV, the accuracy will also increase.

Support Vector Machine, accuracy for the cross validation test (irisdata)

CV	Accuracy(1 is 100%)									
3	1	0.96	1							
4	0.9744	0.9474	0.9722	1						
5	0.9667	1	0.9667	0.9667	1					
6	0.9630	1	0.9167	0.9583	1	1				
7	0.9565	1	0.9524	1	0.9048	1	1			
8	0.9524	1	1	0.8889	1	0.9444	1	1		
9	1	1	1	0.8889	1	0.9333	1	1	1	
10	1	1	1	1	0.8666	1	0.9333	1	1	1

Time consumption

According to our dataset and the after running all of the codes, it is really easy to say that except the neural network, the other algorithms finished nearly within one second, so there is no need to count the time consumption of that, only for the neural network using matlab, we need to record the time and make comparison.

We already tested that the best parameters for a neural network is 7 hidden layers, Levenberg-Marqudt algorithm as the training algorithm and to avoid the overfitting, the training rate can use 0.8, to make the time more significant, I use the wine dataset, since it has more attributes.

Reference

Scikit learn cross validation
http://scikit-learn.org/stable/modules/cross_validation.html