Vinodh Kumar Sunkara (vsunkara6), Microsoft Seattle | Assignment 1 – Supervised learning

Classification Problems:

Wine Quality: This data set is related to red and white variants of the Portuguese "Vinho Verde" wine. The physiochemical (inputs) and sensory (output) variables are available as part of this dataset. It can be viewed as classification or regression task. The classes are ordered and not balanced – for eg, there are many normal wines other than excellent and poor ones. Outlier detection algorithms could be used to detect the few excellent and poor wines. It has 12 attributes in total and 4898 samples to compute across. Along with the practical advantages to detect excellent/poor wines, the dataset is interesting in terms of results obtained when supervised algorithms are applied. The output data based on sensory data could take a value from 0 to 10. This is a challenging task to be handled by any learning algorithm over a broad range of output. This dataset is also not known to have very good results with any supervised learning model, thus making the dataset useful for critiquing learning techniques.

Abalone: It's a dataset taken from a field survey of abalone, a shelled sea creature. The task is to predict the age of abalone given various physical statistics. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem. Abalone have gone through a period of significant decline in recent years due to both natural and unnatural environmental pressures (reintroduction of sea otters, loss of habitat). If techniques to determine age could be used that did not require killing the abalone in order to sample the population, this would be beneficial. The abalone dataset is of interest in many ways because it is so resistant to good performance under machine learning techniques. None of the supervised learning techniques that are presented here achieve particularly good results

Decision Trees

A decision tree is a set of rules that are used to classify data into categories. It looks at the variables in a data set, determines which are most important, and then comes up with a, well, tree of decisions which best partitions the data. The tree is created by splitting data up by variables and then counting to see how many are in each bucket after each split. The key idea is that the procedure to create decision trees is recursive.

For a set (S) of observations, the following algorithm is applied:

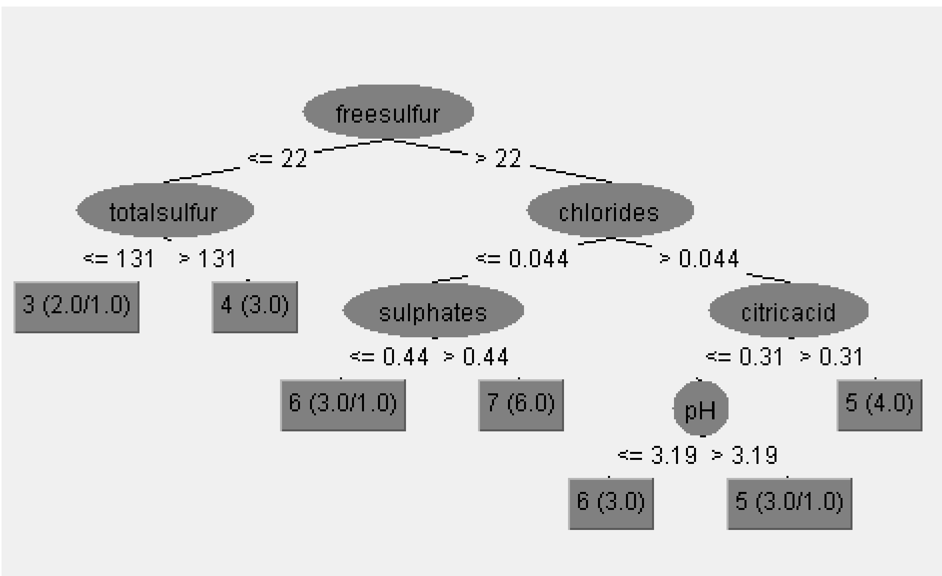
If every observation in S is the same class or if S is very small, the tree becomes an endpoint, labeled with the most frequent class.

If S is too large and it contains more than one class, find the best* rule based on one feature (e.g., "is weight > 150?") to split it into subsets, one for each class.

The best branching rule is the one that results in most information gain. Trees are pruned to avoid overfitting. The pruning algo removes the final nodes based on misclassification rates, so that the model is a little more general.

It’s hard to visualize complete picture of tree for huge dataset like WineQuality. I’ve taken a short sample out of training set and built the tree for it as shown below.

J48 – Wine Quality Short sample - classification tree

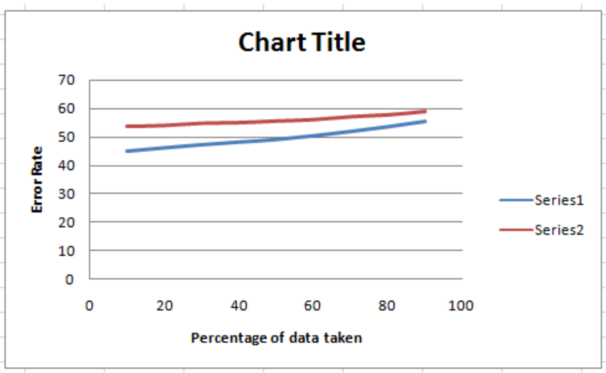


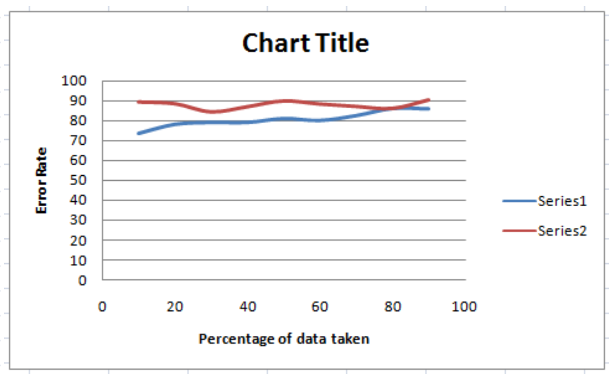
Below is the classifier output on Weka applying J48 classification algorithm on complete dataset. Even after pruning, this a large tree with 520 leaves and 1039 nodes overall. Different tree ensemble techniques like bagging or boosting can be applied to optimize it further, including the incorrectly classified instance rate. The learning curves for Abalone and Wine Quality data sets are shown below for training (series 1) and test (series 2) data sets.

```
Wine Quality - Classifier Output - J48

Number of Leaves  :   520, Size of the tree :    1039

Correctly Classified Instances         857               58.339  %
Incorrectly Classified Instances       612               41.661  %
Kappa statistic                          0.3746
Mean absolute error                      0.0819
Root mean squared error                  0.255
Relative absolute error                 66.5671 %
Root relative squared error            102.795  %
Coverage of cases (0.95 level)          73.9278 %
Mean rel. region size (0.95 level)      15.4156 %
```

```
Total Number of Instances            1469


=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   <-- classified as
   0   0   0   0   0   0   0   0   0   0   0 |   a = 0
   0   0   0   0   0   0   0   0   0   0   0 |   b = 1
   0   0   0   0   0   0   0   0   0   0   0 |   c = 2
   0   0   0   0   0   2   0   0   1   0   0 |   d = 3
   0   0   0   0  14  21  17   1   1   0   0 |   e = 4
   0   0   0   2  12 274 130  22   2   0   0 |   f = 5
   0   0   0   2   7 141 423  62  16   0   0 |   g = 6
   0   0   0   1   3  20 105 131  10   0   0 |   h = 7
   0   0   0   0   1   2  17  13  15   0   0 |   i = 8
   0   0   0   0   0   0   1   0   0   0   0 |   j = 9
   0   0   0   0   0   0   0   0   0   0   0 |   k = 10
```

J48:Wine Quality - Learning curve   J48:Abalone - Learning Curve



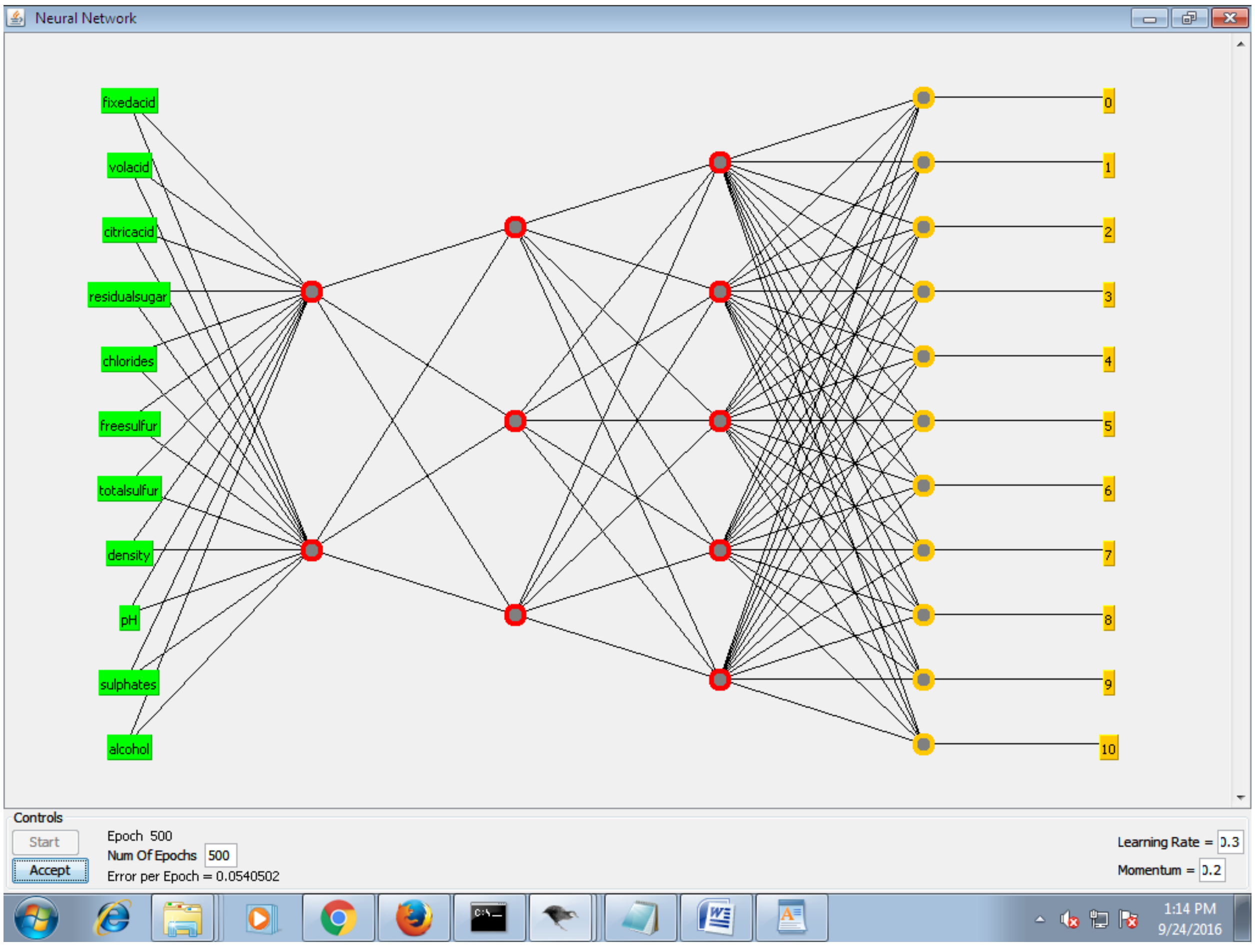


Neural Networks

Neural network is proficient to give the better classification by using non linear boundaries. In addition it is even easy to overcome overfitting by some regularizer settings. In addition, there are lots of things ML offers that Neural Networks might be a solution - Feature selection, density estimation, classification, anomaly detection, reinforcement learning. Neural networks can learn arbitrary boundaries and therefore more accurate provided enough training data. The downside is that the neural network is slower both on classification and training, and less interpretable compared to other algorithms like decision trees.

Below is a neural network for Wine Quality dataset with the hidden layers parameter as 2,3,5. We can see three vertical layers with 2,3,5 nodes in the neural network below. Weka offers MultiLayerPerceptron classifier function to simulate neural networks and the classifier output for algorithm run against wine quality data set is shown below. The accuracy is lesser in relative to the decision trees and also the time taken for complete run is larger. The accuracy also depends on the dataset. For the guess on wine quality score, neural networks work not very effective as per my observation. There is not significant difference in the learning curve trend from previous algorithm. Curve is farther from linearity compared to decision trees.

Neural Network – Wine Quality

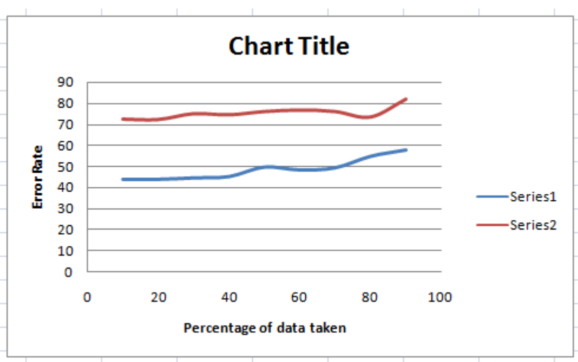


Classifier Output

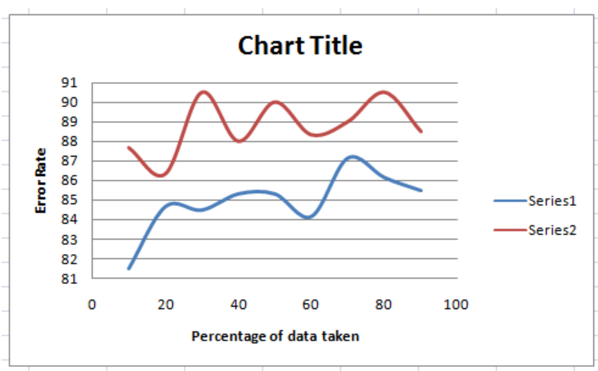
Correctly Classified Instances	764	52.0082 %
Incorrectly Classified Instances	705	47.9918 %
Kappa statistic	0.2258	
Mean absolute error	0.1086	
Root mean squared error	0.2377	
Relative absolute error	88.2803 %	
Root relative squared error	95.8145 %	
Coverage of cases (0.95 level)	97.4813 %	
Mean rel. region size (0.95 level)	32.1059 %	
Total Number of Instances	1469	

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   <-- classified as
   0   0   0   0   0   0   0   0   0   0   0 |   a = 0
   0   0   0   0   0   0   0   0   0   0   0 |   b = 1
   0   0   0   0   0   0   0   0   0   0   0 |   c = 2
   0   0   0   0   0   2   1   0   0   0   0 |   d = 3
   0   0   0   0   0  25  26   3   0   0   0 |   e = 4
   0   0   0   0   0 173 262   7   0   0   0 |   f = 5
   0   0   0   0   0  86 495  70   0   0   0 |   g = 6
   0   0   0   0   0   6 168  96   0   0   0 |   h = 7
   0   0   0   0   0   0  28  20   0   0   0 |   i = 8
   0   0   0   0   0   0   0   1   0   0   0 |   j = 9
   0   0   0   0   0   0   0   0   0   0   0 |   k = 10
```

NN – Wine Quality - Learning curve



NN – Abalone – Learning Curve



Boosting

Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. Modern boosting methods build on AdaBoost, most notably stochastic gradient boosting machines. AdaBoost is best used to boost the performance of decision trees on binary classification problems. I've used J48 as the base classification model and applied boosting on top. As expected, there is an improvement in the accuracy of correctly classified instances. The error rate is also less compared to decision trees plotted from learning curves plotted on wine quality and abalone data sets. The classifier output from Weka is shown below:

=== Run information ===

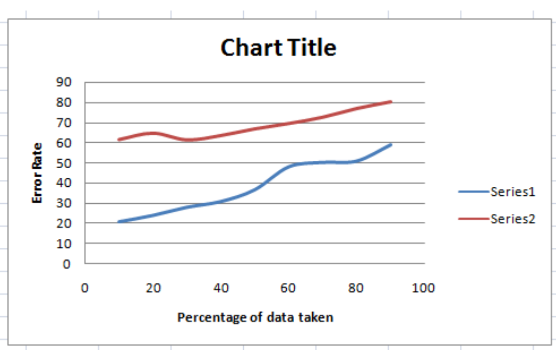
Number of Leaves : 430; Size of the tree : 859
Weight: 2.24; Number of performed Iterations: 10

Correctly Classified Instances	942	64.1253 %
Incorrectly Classified Instances	527	35.8747 %
Kappa statistic	0.455	
Mean absolute error	0.0652	
Root mean squared error	0.2413	
Relative absolute error	52.975 %	
Root relative squared error	97.2613 %	
Coverage of cases (0.95 level)	73.3833 %	
Mean rel. region size (0.95 level)	11.2445 %	
Total Number of Instances	1469	

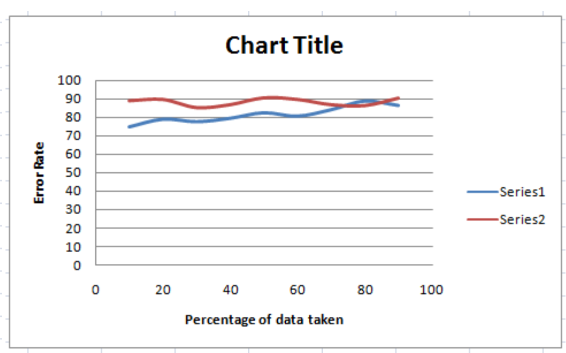
=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	k	<-- classified as
0	0	0	0	0	0	0	0	0	0	0	a = 0
0	0	0	0	0	0	0	0	0	0	0	b = 1
0	0	0	0	0	0	0	0	0	0	0	c = 2
0	0	0	0	0	2	1	0	0	0	0	d = 3
0	0	0	0	15	24	12	2	1	0	0	e = 4
0	0	0	1	4	285	136	16	0	0	0	f = 5
0	0	0	1	0	100	471	73	6	0	0	g = 6
0	0	0	0	0	22	89	153	6	0	0	h = 7
0	0	0	0	0	1	19	10	18	0	0	i = 8
0	0	0	0	0	0	1	0	0	0	0	j = 9
0	0	0	0	0	0	0	0	0	0	0	k = 10

Wine Quality - Learning Curve



Abalone - Learning curve



KNN

K Nearest Neighbors - Classification K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. For the wine quality dataset, k values after 3 showed a stable correctness with less variance with k. k value 3 and 6 behaved almost the same and this dataset has maximum correctness for knn1. Weka has IBK classifier that sets the KNN and it has parameter to change k value. Below is the classifier outputs for different k values and learning curve plotted for k=1,3,6 on abalone and wine quality datasets.

Weka Knn - 1

Correctly Classified Instances	921	62.6957 %
Incorrectly Classified Instances	548	37.3043 %
Kappa statistic	0.4464	
Mean absolute error	0.0681	
Root mean squared error	0.2601	
Relative absolute error	55.3629 %	
Root relative squared error	104.8203 %	
Coverage of cases (0.95 level)	62.6957 %	
Mean rel. region size (0.95 level)	9.0909 %	
Total Number of Instances	1469	

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	k	<-- classified as
0	0	0	0	0	0	0	0	0	0	0	a = 0
0	0	0	0	0	0	0	0	0	0	0	b = 1
0	0	0	0	0	0	0	0	0	0	0	c = 2
0	0	0	0	0	1	2	0	0	0	0	d = 3
0	0	0	0	13	19	19	2	1	0	0	e = 4
0	0	0	0	9	304	113	13	3	0	0	f = 5
0	0	0	1	5	121	424	85	14	1	0	g = 6
0	0	0	1	1	14	76	160	17	1	0	h = 7
0	0	0	0	0	1	16	11	20	0	0	i = 8
0	0	0	0	0	0	0	0	1	0	0	j = 9
0	0	0	0	0	0	0	0	0	0	0	k = 10

Weka knn - 3

Correctly Classified Instances	784	53.3696 %
Incorrectly Classified Instances	685	46.6304 %
Kappa statistic	0.2994	
Mean absolute error	0.0883	
Root mean squared error	0.2437	
Relative absolute error	71.7368 %	
Root relative squared error	98.2466 %	
Coverage of cases (0.95 level)	81.2798 %	
Mean rel. region size (0.95 level)	16.0777 %	
Total Number of Instances	1469	

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	k	<-- classified as
0	0	0	0	0	0	0	0	0	0	0	a = 0
0	0	0	0	0	0	0	0	0	0	0	b = 1
0	0	0	0	0	0	0	0	0	0	0	c = 2
0	0	0	0	0	1	2	0	0	0	0	d = 3
0	0	0	0	7	23	17	7	0	0	0	e = 4

0	0	0	0	10	277	135	19	1	0	0	f = 5
0	0	0	3	11	176	371	86	4	0	0	g = 6
0	0	0	1	1	29	111	122	6	0	0	h = 7
0	0	0	1	1	2	18	19	7	0	0	i = 8
0	0	0	0	0	0	1	0	0	0	0	j = 9
0	0	0	0	0	0	0	0	0	0	0	k = 10

Weka knn - 6

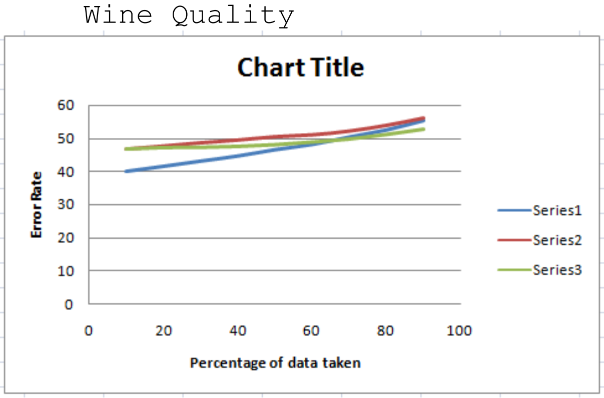
Correctly Classified Instances 769 52.3485 %
Incorrectly Classified Instances 700 47.6515 %
Kappa statistic 0.2738
Mean absolute error 0.0958
Root mean squared error 0.2378
Relative absolute error 77.8447 %
Root relative squared error 9569

=== Confusion Matrix ===

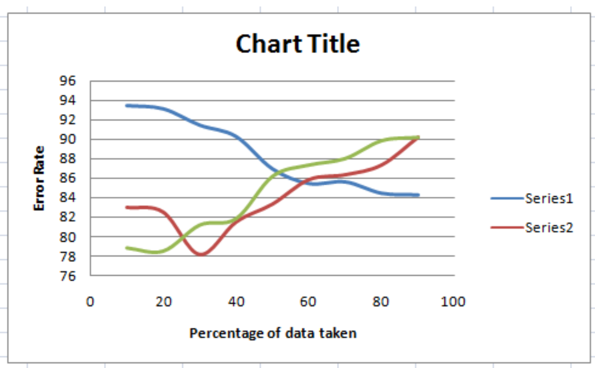
a	b	c	d	e	f	g	h	i	j	k	<-- classified as
0	0	0	0	0	0	0	0	0	0	0	a = 0
0	0	0	0	0	0	0	0	0	0	0	b = 1
0	0	0	0	0	0	0	0	0	0	0	c = 2
0	0	0	0	0	1	2	0	0	0	0	d = 3
0	0	0	0	3	31	16	4	0	0	0	e = 4
0	0	0	0	10	279	129	22	2	0	0	f = 5
0	0	0	0	2	174	387	82	6	0	0	g = 6
0	0	0	0	0	29	137	98	6	0	0	h = 7
0	0	0	0	0	0	20	26	2	0	0	i = 8
0	0	0	0	0	0	1	0	0	0	0	j = 9
0	0	0	0	0	0	0	0	0	0	0	k = 10

Weka KNN - 1,3,6 (b,r,g) Learning Curve

Wine Quality



Abalone



SVM

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features er have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line). These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs we define. Weka has SMO classifier that works as SVM, and the kernel parameter is available to change the kernel function used for the transformation. Below are the classifier outputs for SMO with three different kernel functions – PolyKernel, NormalizedPolyKernel, RBFKernel. Normalized Polykernel has worked better in correctness compared to polykernel and RBFKernel. The error rates are dataset dependent. Learning curves for both abalone and wine quality data sets are shown below. Wine Quality showed a regular trend while abalone has a huge drop in error with increase in the percentage of sample.

Weka SVM – SMO – Polykernel

```
Correctly Classified Instances         736               50.1021 %
Incorrectly Classified Instances       733               49.8979 %
Kappa statistic                          0.1596
Mean absolute error                      0.1474
Root mean squared error                  0.2656
Relative absolute error                119.8248 %
Root relative squared error            107.0685 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)      63.6364 %
Total Number of Instances             1469

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   <-- classified as
   0   0   0   0   0   0   0   0   0   0   0 |   a = 0
   0   0   0   0   0   0   0   0   0   0   0 |   b = 1
   0   0   0   0   0   0   0   0   0   0   0 |   c = 2
   0   0   0   0   0   2   1   0   0   0   0 |   d = 3
   0   0   0   0   0  34  20   0   0   0   0 |   e = 4
   0   0   0   0   0 208 234   0   0   0   0 |   f = 5
   0   0   0   0   0 123 528   0   0   0   0 |   g = 6
   0   0   0   0   0  14 256   0   0   0   0 |   h = 7
   0   0   0   0   0   0  48   0   0   0   0 |   i = 8
   0   0   0   0   0   0   1   0   0   0   0 |   j = 9
   0   0   0   0   0   0   0   0   0   0   0 |   k = 10
```

Weka – SVM – SMO – RBFKernel

```
Correctly Classified Instances         651               44.3159 %
Incorrectly Classified Instances       818               55.6841 %
Kappa statistic                          0
Mean absolute error                      0.1479
```

```
Root mean squared error                  0.2666
Relative absolute error                120.2375 %
Root relative squared error            107.4531 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)      63.6364 %
Total Number of Instances             1469

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   <-- classified as
   0   0   0   0   0   0   0   0   0   0   0 |   a = 0
   0   0   0   0   0   0   0   0   0   0   0 |   b = 1
   0   0   0   0   0   0   0   0   0   0   0 |   c = 2
   0   0   0   0   0   0   3   0   0   0   0 |   d = 3
   0   0   0   0   0   0  54   0   0   0   0 |   e = 4
   0   0   0   0   0   0 442   0   0   0   0 |   f = 5
   0   0   0   0   0   0 651   0   0   0   0 |   g = 6
   0   0   0   0   0   0 270   0   0   0   0 |   h = 7
   0   0   0   0   0   0  48   0   0   0   0 |   i = 8
   0   0   0   0   0   0   1   0   0   0   0 |   j = 9
   0   0   0   0   0   0   0   0   0   0   0 |   k = 10
```

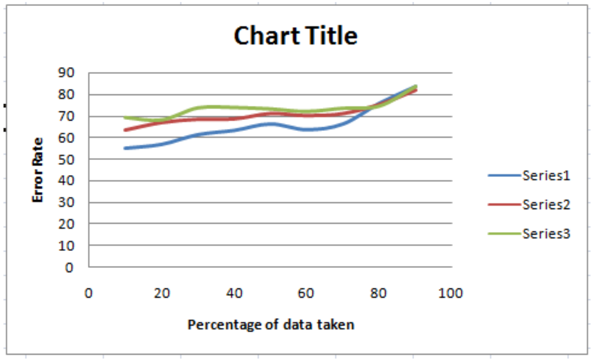
Weka – SVM – SMO – NormalizedPolyKernel

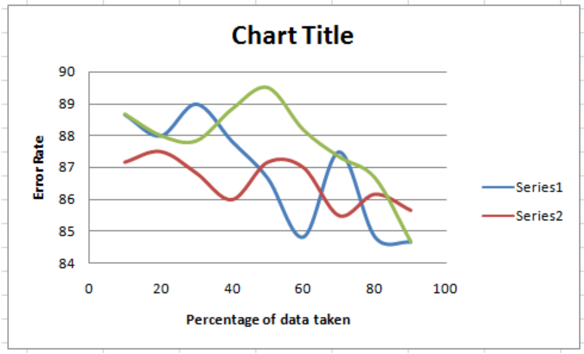
```
Correctly Classified Instances         739               50.3063 %
Incorrectly Classified Instances       730               49.6937 %
Kappa statistic                          0.161
Mean absolute error                      0.1475
Root mean squared error                  0.2657
Relative absolute error                119.8494 %
Root relative squared error            107.0916 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)      63.6364 %
Total Number of Instances             1469

=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   <-- classified as
   0   0   0   0   0   0   0   0   0   0   0 |   a = 0
   0   0   0   0   0   0   0   0   0   0   0 |   b = 1
   0   0   0   0   0   0   0   0   0   0   0 |   c = 2
   0   0   0   0   0   2   1   0   0   0   0 |   d = 3
   0   0   0   0   0  31  23   0   0   0   0 |   e = 4
   0   0   0   0   0 205 237   0   0   0   0 |   f = 5
   0   0   0   0   0 117 534   0   0   0   0 |   g = 6
   0   0   0   0   0  11 259   0   0   0   0 |   h = 7
   0   0   0   0   0   0  48   0   0   0   0 |   i = 8
   0   0   0   0   0   0   1   0   0   0   0 |   j = 9
   0   0   0   0   0   0   0   0   0   0   0 |   k = 10
```

SMO Learning curve – PolyKernel, normalizedPolyKernel, RBFKernel (b,r,g)

Wine Quality

Abalone





THANK YOU