Programming Assignment 2

Ashutosh Kumar Jha, ME14B148

7th October 2016

1 Support Vector Machines

The LIBSVM package was used to train the SVM models.

The features were extracted using the python code supplied along with the dataset.

The features were then normalized to 0 mean and a standard deviation of 1.

10 fold cross validation was used to train the models and the model parameters were selected by changing them in magnitudes of 10and seeing which parameters performed the best.

The various parameters and the performance of the models are given below:

Kernel	Parameters	Accuracy
Linear	$c = 0.1$	60.835%
Polynomial	$c = 3, d = 1$	61.829%
Gaussian	$c = 1, g = 0.01$	66.004%
Sigmoidal	$c = 0.1, g = 0.01, r = 0$	53.082%

Here,
c is the regularization parameter
d is the degree in the polynomial kernel function
g is the gamma in the RBF and sigmoidal kernels
r is the **coeff0** in the sigmoidal kernel

P.S. For saving the model into a mat file, cross validation was removed from the **svmtrain** command as in the presence of cross validation, the model was not being obtained as struct.

2 Backpropogation Algorithm

The same features as those obtained in the previous question were used.

The features were again standardized.

2.1 Without Regularization

First, the standard backpropogation algorithm was implemented using a cross-entropy loss and a sigmoid activation function.

The number of hidden layer neurons was set to $M = 50$.

The learning rate used was $\eta = 0.1$

The per class precision, recall and F-scores are as follows:

Performance Measure	Class City	Class Coast	Class Forest	Class Mountain
Precision	0.6875	0.7391	0.5714	0.5500
Recall	0.5500	0.8500	0.6000	0.5500
F-Measure	0.6111	0.7907	0.5854	0.5500

2.2 With Regularization

In the case above, there are a lot of parameters which are involved. As a result of this there is a very high chance that overfitting might occur.

In the given new loss function, we are penalizing the parameters using the L2-loss:

$$R(\theta) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2 + \gamma(\sum_k\sum_m \beta_{km}^2 + \sum_m\sum_l \alpha_{ml}^2)$$

We now differentiate this with respect to all the different parameters.

The first term in the loss function has already been done in great detail in class. As for the regularization term, the derivative of the square term will become the term itself multiplied by 2. Thus, we get:

$$\frac{\partial R}{\partial \beta_{km}} = \begin{cases} \sum_{i=1}^{N} \delta_{ki} z_{mi} + 2\gamma\beta_{km} & k \neq 0 \\ \sum_{i=1}^{N} \delta_{ki} z_{mi} & k = 0 \end{cases}$$

$$\frac{\partial R}{\partial \alpha_{ml}} = \begin{cases} \sum_{i=1}^{N} x_{il} S_{mi} + 2\gamma\alpha_{ml} & m \neq 0 \\ \sum_{i=1}^{N} x_{il} S_{mi} & m = 0 \end{cases}$$

where,

$$\delta_{ki} = -2(y_{ik} - f_k(x_i))g_k^{'}(\beta_k^T z_i)$$

$$S_{mi} = \sigma^{'}(\alpha_m x_i) \cdot \sum_k \beta_{km}\delta_{ki}$$

Thus, the update rules are:

$$\beta_{km}^{(r+1)} = \begin{cases} \beta_{km}^{(r)} - \eta \sum_{i=1}^{N} \delta_{ki} z_{mi} - 2\eta\gamma\beta_{km}^{(r)} & k \neq 0 \\ \beta_{km}^{(r)} - \eta \sum_{i=1}^{N} \delta_{ki} z_{mi} & k = 0 \end{cases}$$

and

$$\alpha_{ml}^{(r+1)} = \begin{cases} \alpha_{ml}^{(r)} - \eta \sum_{i=1}^{N} S_{mi} x_{il} - 2\eta\gamma\alpha_{ml}^{(r)} & m \neq 0 \\ \alpha_{ml}^{(r)} - \eta \sum_{i=1}^{N} S_{mi} x_{il} & m = 0 \end{cases}$$

In this assignment we have chosen g_k to be the sigmoid function instead of the softmax function for ease of writing the program.

The value of γ was varied from $10^{-2}$to 10^2 in order of 10's and the value of $\gamma = 1$ was chosen.

The performance measures are given as follows:

Performance Measure	Class City	Class Coast	Class Forest	Class Mountain
Precision	0.7271	0.7627	0.5326	0.5734
Recall	0.5324	0.8215	0.6275	0.5485
F-Measure	0.6147	0.7910	0.5761	0.5606

It was observed that as the value of γ increased, the F-measures improved but after a certain point, they start dropping. This is because the bias error becomes too high and we under fit if γ is very high.

The presence of γ did exactly what was expected. It reduced the values of the parameters and tried to drive them to zero. Thus, it tried to reduce the variance of the model.

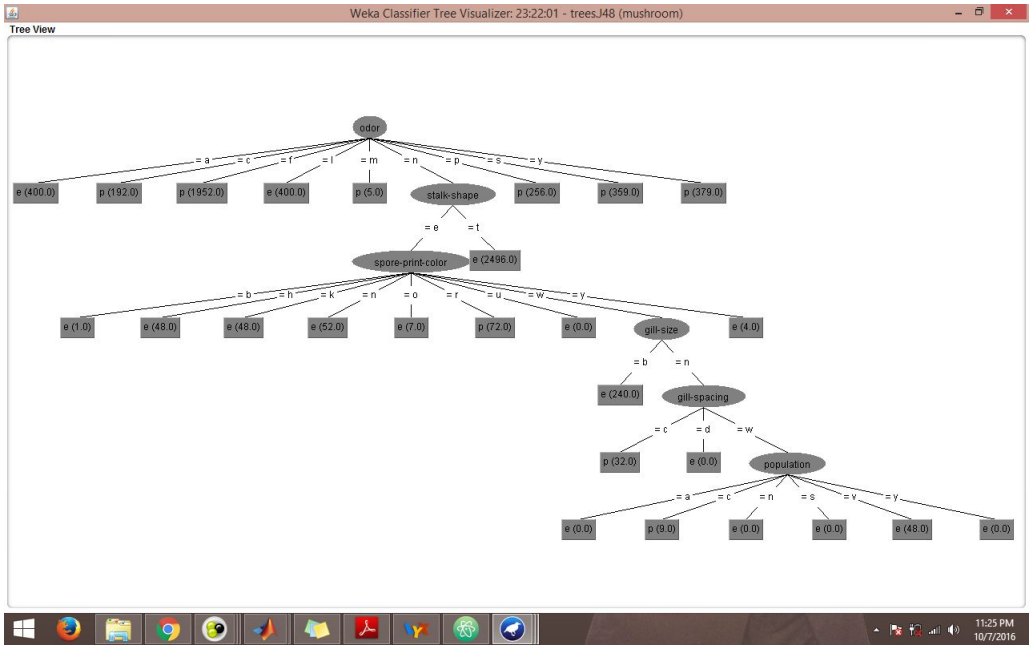
3 Decision Trees

The dataset was downloaded from the UCI ML Repository.

The dataset was then converted to an appropriate .arff format and the set was split into training and test data as asked.

We first run the J48 tree algorithm in Weka with the default parameters. The performance measures obtained are:

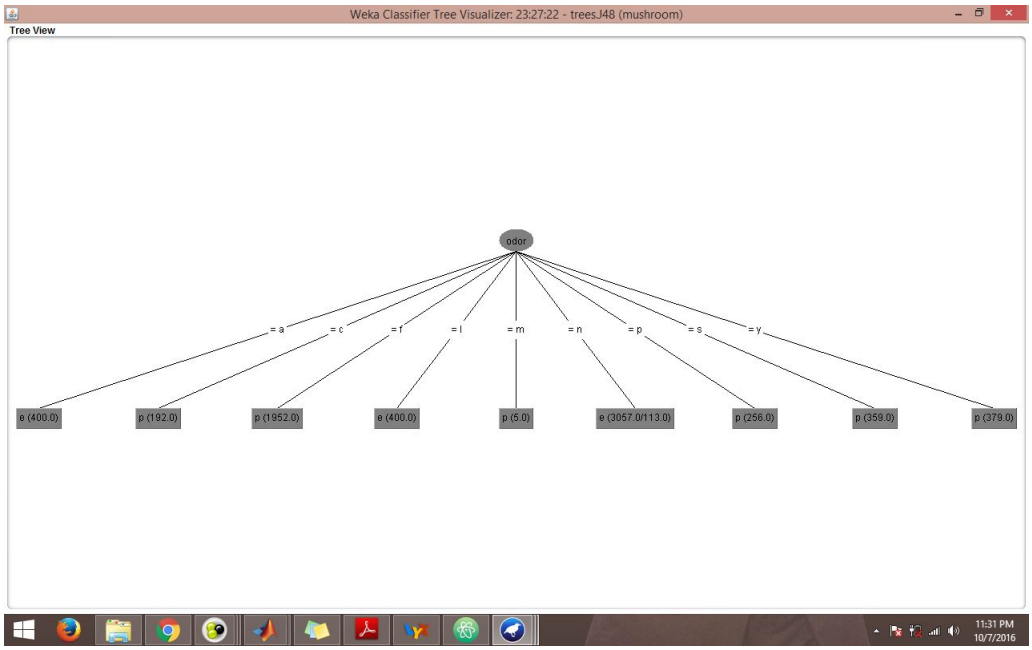
Measure	Class Edible	Class Poisonous
Precision	1	1
Recall	1	1
F-Measure	1	1



The minNumObj is increased in the orders of magnitude of 10. We notice that no effect is seen on the performance measures till $M = 200$.

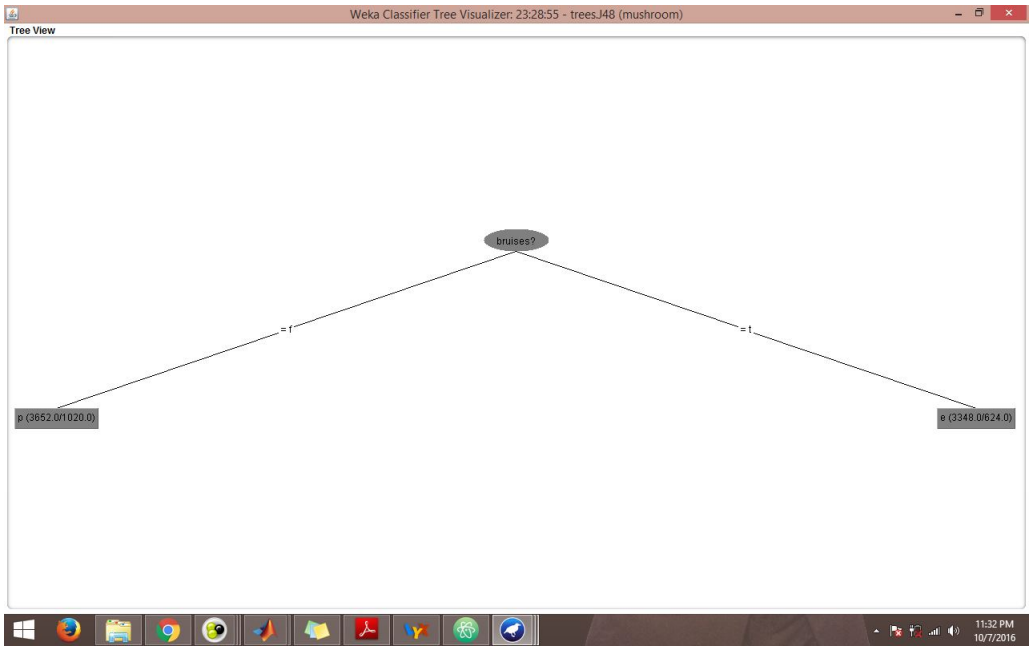
At $M = 200$:

Measure	Class Edible	Class Poisonous
Precision	0.985	1
Recall	1	0.989
F-Measure	0.993	0.995



We further increase M to see at $M = 2000$:

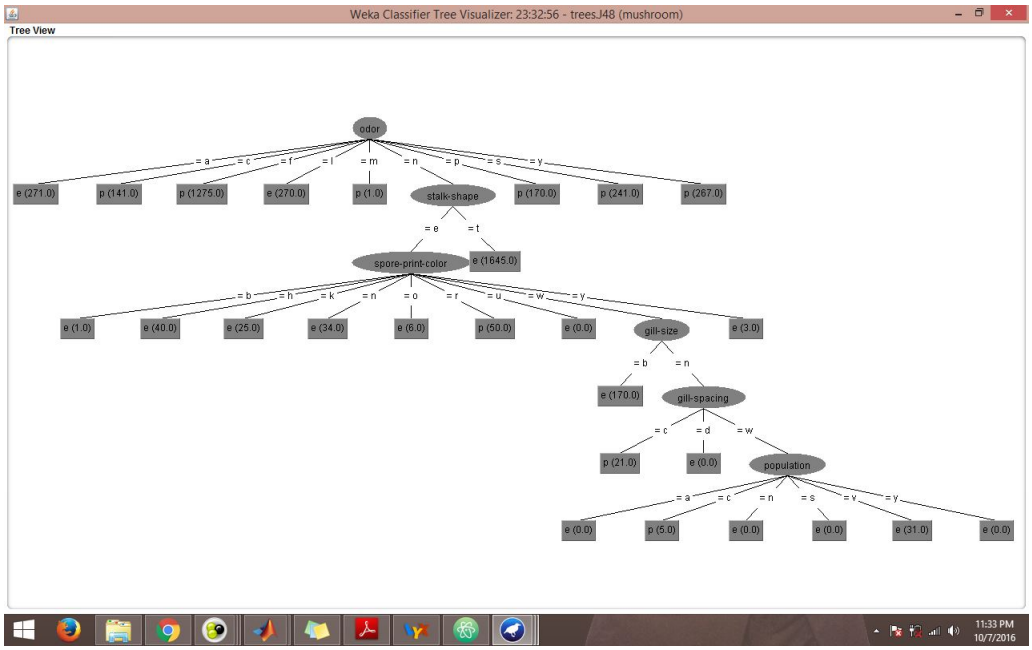
Measure	Class Edible	Class Poisonous
Precision	1	0.602
Recall	0.060	1
F-Measure	0.114	0.752



We thus observe that as minNumObj increases, the performance falls.

We then do a reduced error pruning and observe the following performance measures:

Measure	Class Edible	Class Poisonous
Precision	1	1
Recall	1	1
F-Measure	1	1



Note that at every stage, the tree itself has changed by a lot even though the performance measures might be the same.

As seen from the pruned tree, **odor, spore-print-colour, gill-size, stalk-shape, gill-spacing and population** are the important factors in deciding if the mushroom is edible or not.