CS5011: INTRODUCTION TO MACHINE LEARNING

PROGRAMMING ASSIGNMENT - 2

Adarsh B
MM14B001

1. SUPPORT VECTOR MACHINES

Problem Statement

Use the training instances from DS2, carry out feature extraction and use it to train an SVM to classify the test images into either of the following four categories: coast, forest, inside-city, mountain. Use the training data to build classification models using the following kernels.

1. Linear kernel
2. Polynomial kernel
3. Gaussian kernel
4. Sigmoid kernel

Come up with the kernel parameters for the various models. You can use a fraction of data supplied to do a n-fold cross validation to find the best model parameters.

Approach

The svm utility from sklearn package has been used. We have imported the images from the dataset DS2 one by one, extracted the features, and then imported it for training SVMs of various kernels. The data has been standardized using a Standard scaler (removing the mean and scaling to unit variance). Five-fold cross validation has been used while running each kernel, to pick the optimal values of the kernel parameters and the regularization parameter. Basically it maximizes the margin that separates the data points of the classes using a hyperplane.

To perform nonlinear classification, linear SVMs are exploited using the kernel-trick. It is an optimized way of basis expansion using a kernel function, K(x,x') for computing inner products between vectors x and x'. The various kernel functions are summarized below:

Kernel type	Kernel function $K(x, x')$
Linear	$x'^T x$
Polynomial	$(x'^T x + coef0)^{degree}$
Gaussian/RBF	$e^{-(\frac{(x'-x)^T(x'-x)}{2\gamma^2})}$
Sigmoid	$tanh(\gamma x'^T x + coef0)$

Output / Results

The parameters that yield the best fits are tabulated:

Kernel type	**Best fit parameters**	**Accuracy**
Linear	C=0.35938136638	79.8073986503
Polynomial	coef0=0.4; degree=2; C=21.5443469003	87.5661297473
RBF	C=10000.0; gamma=0.00599484250319	87.6641544751
Sigmoid	C = 2.78255940221; coef0 = -0.5	72.0555637653

2. NEURAL NETWORKS

Problem Statement

Implement original back-propagation algorithm. Use DS2 for training your neural network. Report per-class precision, recall and F-measure on the test data used in Question-1. Now consider the following alternate error function for training neural networks.

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

where N is the number of training instances, K is the number of output features, $f_k(x)$ is the predicted output vector, y is the original output vector, α and β are the weights and γ is a regularization parameter. Derive the gradient descent update rule for this definition of R. Now train your neural network with this new error function. Report per-class precision, recall and F-measure on the same test data. What will happen when you vary the value of γ? Vary the value of γ from 10^{-2} to 10^{2} in multiples of 10 and repeat the experiment and report the results. Can you figure out the effect of γ in the results? Look at the weights learnt using the new error function. What do you infer from them?

Approach

We have used the following in the approach:

- Gradient descent approach for training the Neural Network
- Backpropagation algorithm, with the squared error loss function
- A sigmoid activation function for the hidden layer
- A softmax activation function for the output layer.

Here, we have implemented a 3-layer neural network with 96 input neurons, 40 hidden neurons and 4 output neurons. The output is softmaxed and is taken as the probability of the prediction of each class. The number of hidden neurons is fixed after a number of trials.

Since Artificial Neural Networks has a large number of weights to estimate complicated functions, the training process is computationally heavy and time consuming. But once learnt, the forwarding process to predict the output is quick. Also, if the number of parameters to be estimated is large, we might encounter overfitting issues. The problem is addressed using regularization. The L2 regularization keeps the problem mathematically tractable, whilst shrinking the parameter values to near zero. Weights close to zero will not contribute significantly to the total cost.

ANNs are configured using a variety of parameters, the important ones being:

- Learning rate α: It controls the step taken in the direction opposite to the gradient. Larger the learning rate, quicker the optimization but might run into issues of divergence or skipping minimas.
- Regularization parameter λ: It controls the importance given to obtaining small weights while solving for weights. Larger the regularization parameter, more emphasis is on obtaining lesser magnitude of weights at the cost of performance. It controls overfitting and enables models to generalize well on the test-set.

Output / Results

The performance measures for non-regularized approach are:

Measure	Coast	Forest	Inside-city	Mountain
Accuracy	58.75%			
Precision	0.8	0.8	0.65	0.1
Recall	0.44444444	0.84210526	0.59090909	0.66666667
F-score	0.57142857	0.82051282	0.61904762	0.17391304

The performance measures for L2-regularized approach are (best model is obtained for λ=1.0):

Measure	Coast	Forest	Inside-city	Mountain
Accuracy	63.75%			
Precision	0.75	0.95	0.65	0.2
Recall	0.53571429	0.7037037	0.68421053	0.66666667
F-score	0.625	0.80851064	0.66666667	0.30769231

The regularized model clearly performs better than the non-regularized model. We infer that the regularization has indeed shrunk a majority of the weights close to zero, by its inclusion in the gradient descent. This ensures that overfitting is avoided, and also leads to a stable classifier, i.e. small changes in the input will not affect the output much.

3. DECISION TREES

Problem Statement

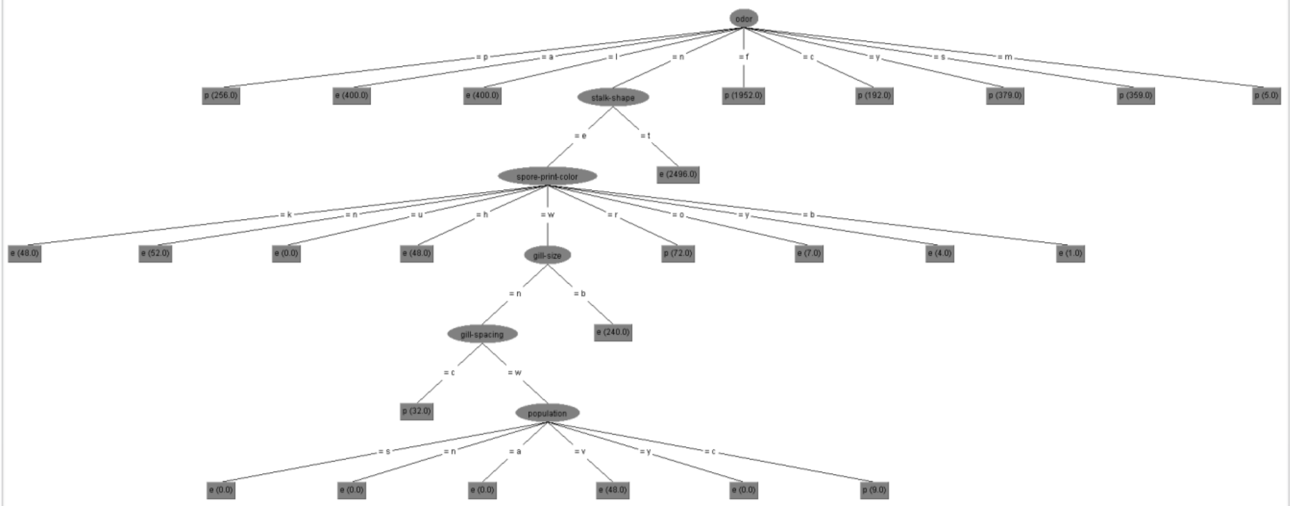
You need to use Weka for this question. We will use Mushroom dataset from UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets/Mushroom). This is a 2-class problem with 8124 instances. Use the last 1124 instances as test data and the rest as training data.

1. Convert the data into ARFF format.
2. Run J48 Decision Tree algorithm from Weka. Report precision, recall and f1- measure. What is the effect of MinNumObj on the performance? What happens when you do reduced Error Pruning?
3. What are the important features in deciding whether a mushroom is edible or not?
4. Turn in the Decision Tree learnt by the model (the decision tree with the best performance).

Approach

The dataset contains 22 features and a binary class variable that says whether the mushroom is edible or not, based on observation. The given .data and .names files have been converted into .arff files, with 1124 datapoints in the test file and the rest in the training file. We run the J48 algorithm in Weka for the above set up.

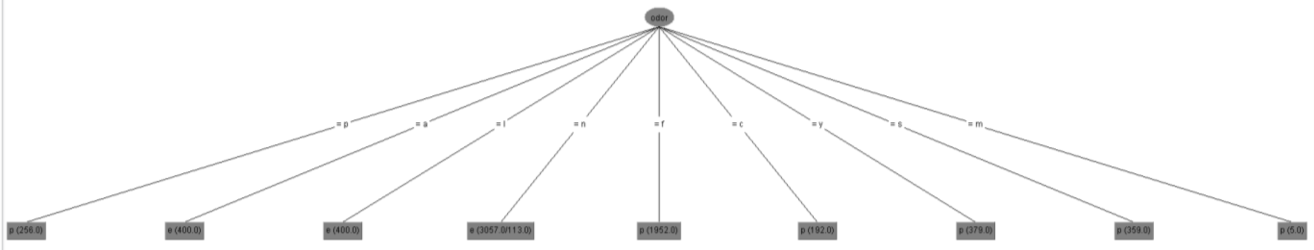
From the outputted tree, we can infer that the important features learnt are odor, stalk-shape, sporeprint-color, gill-size, gill-spacing and population.



The performance measures obtained for the above tree are:

Measure	**Poisonous**	**Edible**
Precision	1.0	1.0
Recall	1.0	1.0
F-score	1.0	1.0

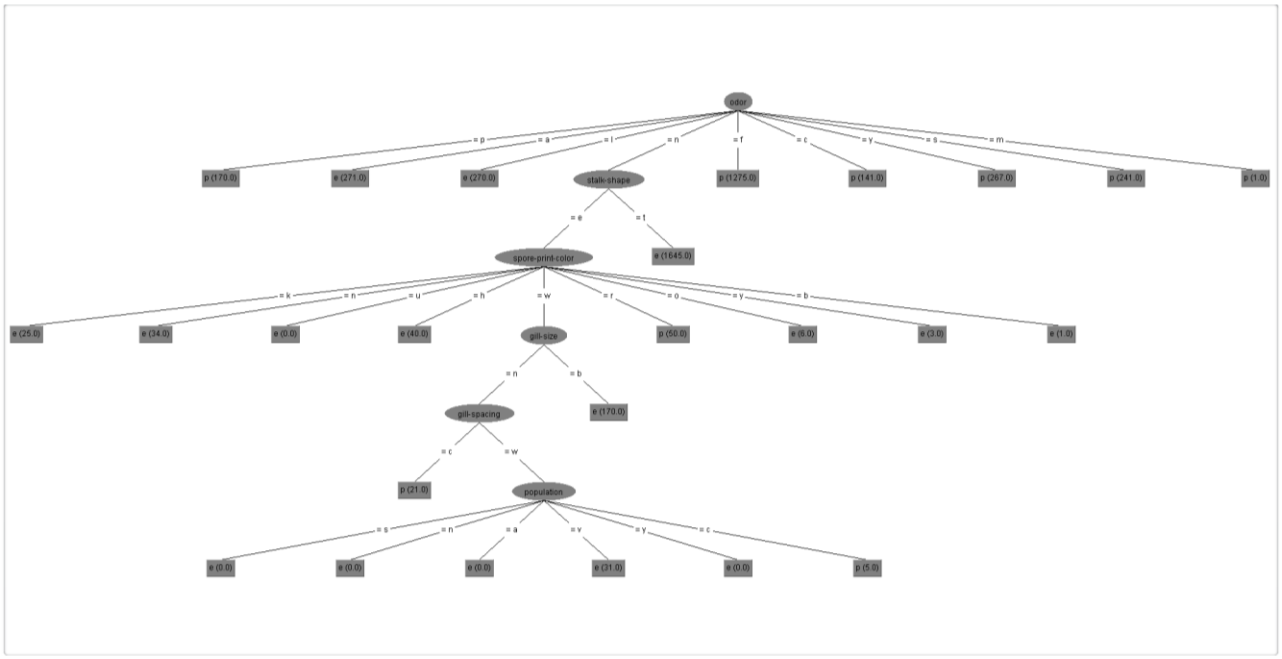
We see that we get a perfect classification on the test set by using the default attributes in the J48 algorithm. On increasing the minNumObj to 200, we get the following tree:



We see that the tree has drastically changed, and the only parameter learnt now is odor. Basically minNumObj specifies the minimum number of instances per leaf. The default value for this is 2. On increasing minNumObj, the minimum number of instances in the leaves has been increased, hence finer splitting doesn't occur and the leaves will no longer be pure! This leads to a decrease in performance, as illustrated below:

Measure	Poisonous	Edible
Precision	1.0	0.971
Recall	0.965	1.0
F-score	0.982	0.985

On setting reduced error pruning to true, we get the following outputs:



Reduced error pruning does not affect the output adversely. Basically it just tried to combine nodes if it sees an improvement in performance. The decision variables are still unchanged, but the count of classified data points is different in each case. Performance is as equally good as reported in the default case:

Measure	Poisonous	Edible
Precision	1.0	1.0
Recall	1.0	1.0
F-score	1.0	1.0

Conclusively, the default case and the reduced pruning case show optimal performance characteristics, and the important features learnt for dciding whether a mushroom is edible or not are **odor, stalk-shape, sporeprint-color, gill-size, gill-spacing and population**.