Introduction to Machine learning (CS5011)
Programming assignment #2

Aravind S
EE14B013

7th Oct. 2016

Contents

1 Support Vector Machine

1.1 Goal

We are given the training instances for an image classification problem DS2. We have to train an SVM to classify it into either of the following four categories: coast, forest, inside-city and mountain.

Also, we need to find the best kernel parameters for the SVM, for the following kernels:

- Linear kernel
- Polynomial kernel
- Gaussian kernel
- Sigmoid kernel

1.2 Approach

Support Vector Machines are supervised learning models that can analyze data and can be used for classification and regression analysis. It maximises the margin that separates the data points of 2 classes (consider a binary classification problem) using a linear hyperplane. The magnitude of slack variables are controlled using a parameter, C.

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

The kernel functions for popular kernels 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	$tanh(\gamma x'^T x + coef0)$

Now, depending on the parameters of the kernel (c, d in a polynomial kernel etc) the basis obtained are different and the SVMs learnt are different.

So, the first step is to estimate the kernel parameters for the different kernels. I have used a 5-fold cross validation on the train set and trained SVMs for different kernels and different parameters. Based on the average performance in the cross validation set, the best parameters are chosen for each kernel. Then the model is trained on the complete train-set with these parameters and performance is reported on the test-set.

1.3 Results

The best-fit parameters obtained for each kernel is shown below. Also, the accuracy of the best-fit model on the test-set is shown.

Kernel type	Best-fit parameters	Accuracy
Linear	C = 2.7825594022071258	66.25 %
Polynomial	C = 10000.0	67.50 %
	coef0 = 0.0	
	degree = 2	
Gaussian	C = 2.7825594022071258	70.00 %
	gamma (γ) = 0.35938136638046259	
Sigmoid	C = 1291.5496650148827	25.0 %
	coef0 = 0.30000000000000004	

2 Neural Networks

2.1 Goal

We are given the training instances for an image classification problem DS2. We have to train a neural network to classify it into either of the following four categories: coast, forest, inside-city and mountain.

2.2 Approach

Artificial neural networks (commonly referred to as neural networks) is a network inspired by biological neural networks which are used to estimate or approximate functions that depend on large number of inputs. Since it has a large number of weights to estimate complicated functions, the training process is computationally heavy and time consuming. But once learnt, the feed-forward process to predict the output is quick.

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

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 generalise well on the test-set.

2.3 Results

By trial the number of hidden neurons are fixed at 40.

Without regularization ($\lambda = 0$), the performance is reported below.

Measure	Class coast	Class forest	Class insidecity	Class mountain
Accuracy	25.00 %			
Precision	0.0	0.0	0.0	1.0
Recall	0.0	0.0	0.0	0.25
F1-score	0.0	0.0	0.0	0.40

With regularization, and with λ varied between 10^{-2} to 10^{2} in 5 steps on log scale. The weights obtained are smaller in magnitude. This ensures the following:

- The new model obtained is less overfitted when compared to the older one and it will generalise better than that. This is evident from the performance of the second model when compared to the first.
- Small changes in the input will not affect the output much, i.e the new model is more stable when compared to the old model.

The optimal value of λ for best performance turns out to be $\lambda = 1.0$. The performance of this best model is reported below.

Measure	Class coast	Class forest	Class insidecity	Class mountain
Accuracy	70.00 %			
Precision	0.60	0.80	0.85	0.55
Recall	0.60	0.84	0.68	0.69
F1-score	0.60	0.82	0.76	0.61

Thus, we can see how the performance of the second model is better than that of the first.

3 J48 Decision tree classification

3.1 Goal

We are given the Mushroom dataset from the UCI machine learning repository. This is a 2-class problem with 8124 instances. We will use the last 1124 instances as test-data and the remaining as train-data. For training, we will use the J48 Decision tree algorithm (C4.5).

3.2 Approach

C4.5 decision tree algorithm builds decision trees using information gain. At each node, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain i.e the attribute with the highest information gain is chosen to make a decision. Then, it recurs and splits the tree at each node.

The base cases followed by C4.5 are:

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

It has a number of parameters like minNumObj, reducedErrorPruning for controlling the structure/properties of the tree learnt.

- minNumObj specifies the minimum number of instances per leaf. The default value for this is 2.
- reducedErrorPruning is a boolean variable which specifies whether to prune the tree using reduced error pruning method. The default value for this is false.

We need to report the performance of the algorithm and study the variance of performance with minNumObj (M) and reducedErrorPruning (r).

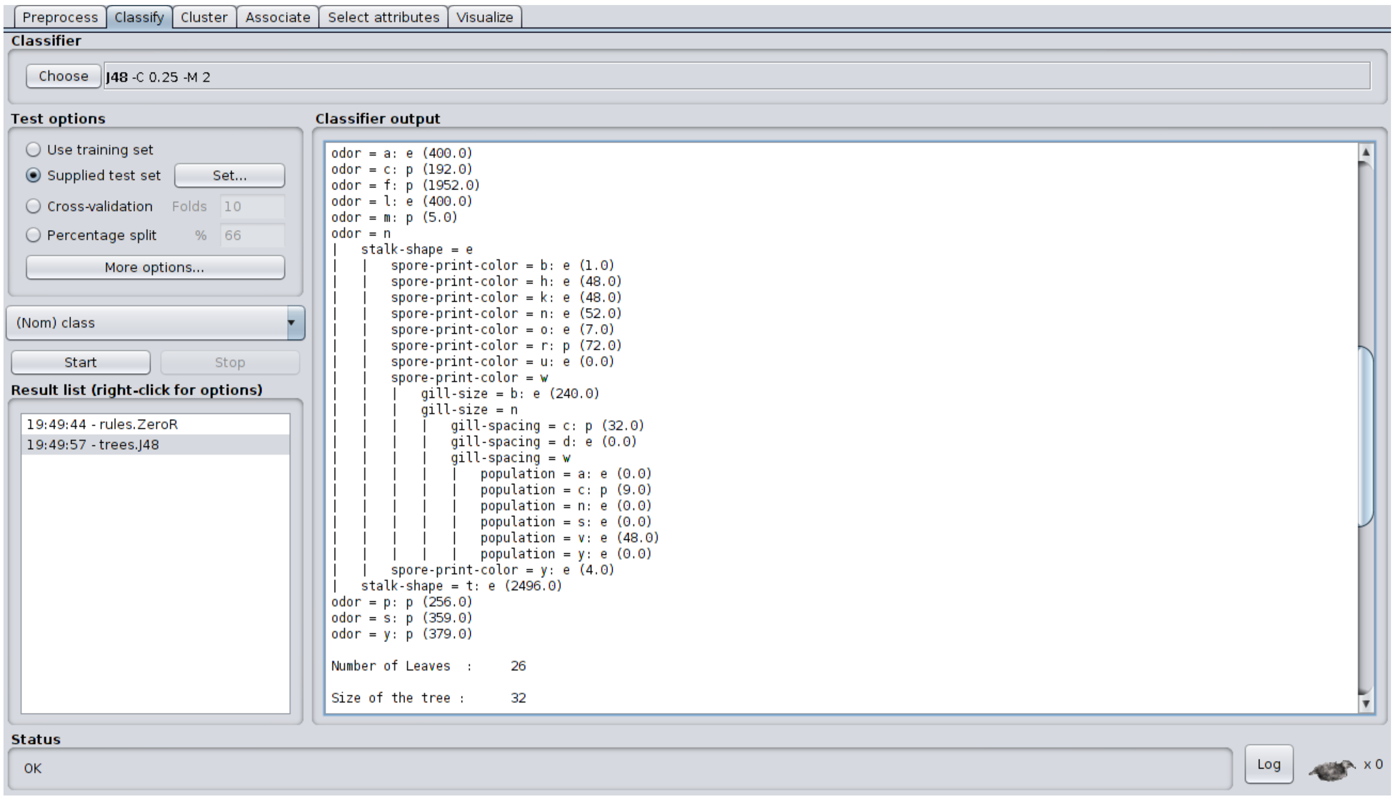
3.3 Results

The performance of the best-model trained is summarized below.

Measure	Class - e (edible)	Class - p (poisonous)
Accuracy	100.00 %	
Precision	100.00 %	100.00 %
Recall	100.00 %	100.00 %
F-measure	1.00	1.00

As we all know, C4.5 is a very popular algorithm and works really well for a variety of practical problems. And our trial has shown its performance.

Let us analyze the best-model learnt. The splits are summarized below.



J48 -C 0.25 -M 2

Use training set
[✓ Supplied test set] Set...
Cross-validation Folds 10
Percentage split % 66

(Nom) class

19:49:44 - rules.ZeroR
19:49:57 - trees.J48

```
odor = a: e (400.0)
odor = c: p (192.0)
odor = f: p (1952.0)
odor = l: e (400.0)
odor = m: p (5.0)
odor = n
|   stalk-shape = e
|   |   spore-print-color = b: e (1.0)
|   |   spore-print-color = h: e (48.0)
|   |   spore-print-color = k: e (48.0)
|   |   spore-print-color = n: e (52.0)
|   |   spore-print-color = o: e (7.0)
|   |   spore-print-color = r: p (72.0)
|   |   spore-print-color = u: e (0.0)
|   |   spore-print-color = w
|   |   |   gill-size = b: e (240.0)
|   |   |   gill-size = n
|   |   |   |   gill-spacing = c: p (32.0)
|   |   |   |   gill-spacing = d: e (0.0)
|   |   |   |   gill-spacing = w
|   |   |   |   |   population = a: e (0.0)
|   |   |   |   |   population = c: p (9.0)
|   |   |   |   |   population = n: e (0.0)
|   |   |   |   |   population = s: e (0.0)
|   |   |   |   |   population = v: e (48.0)
|   |   |   |   |   population = y: e (0.0)
|   |   spore-print-color = y: e (4.0)
|   stalk-shape = t: e (2496.0)
odor = p: p (256.0)
odor = s: p (359.0)
odor = y: p (379.0)

Number of Leaves  :     26

Size of the tree :      32
```

OK

Based on the above splits, we can infer that the important features learnt are: **odor, stalk-shape, spore-print-color, gill-size, gill-spacing** and **population**.

When we increase M, the minimum number of instances per leaf increases. As a result, the leaves of the tree do not split if the number of instances in its children tend to go below M. As a result, finer splitting doesn't occur i.e the initial layers (along depth) will be split fine, whereas the layers near the leaves tend to stop splitting further. This can also be compared to pruning the tree after it has been built.

Reduced error pruning on the other hand combines nodes if there is an improvement in performance. This indirectly makes sure the constraint on M is more rigorously satisfied.

The best decision tree model is submitted along with the source code.