Report One: Decision Trees

Mikhail Khrypach (mk207@doc.ic.ac.uk),
Martin Pavlovsky (mp107@doc.ic.ac.uk),
John Psomopoulos (jp07@doc.ic.ac.uk),
Oliver Rogers (or107@doc.ic.ac.uk)

February 11, 2010

1 Creating the Decision Trees

Decision trees are one of the simplest and yet most successfull forms of learning algorithms.

After geting all the all the data from the *cleandata_students.txt* we used it to train our decision trees. We created one tree for each emotion and we used the ID3 algotithm to make the best choise for the attributes.

Here are the results:

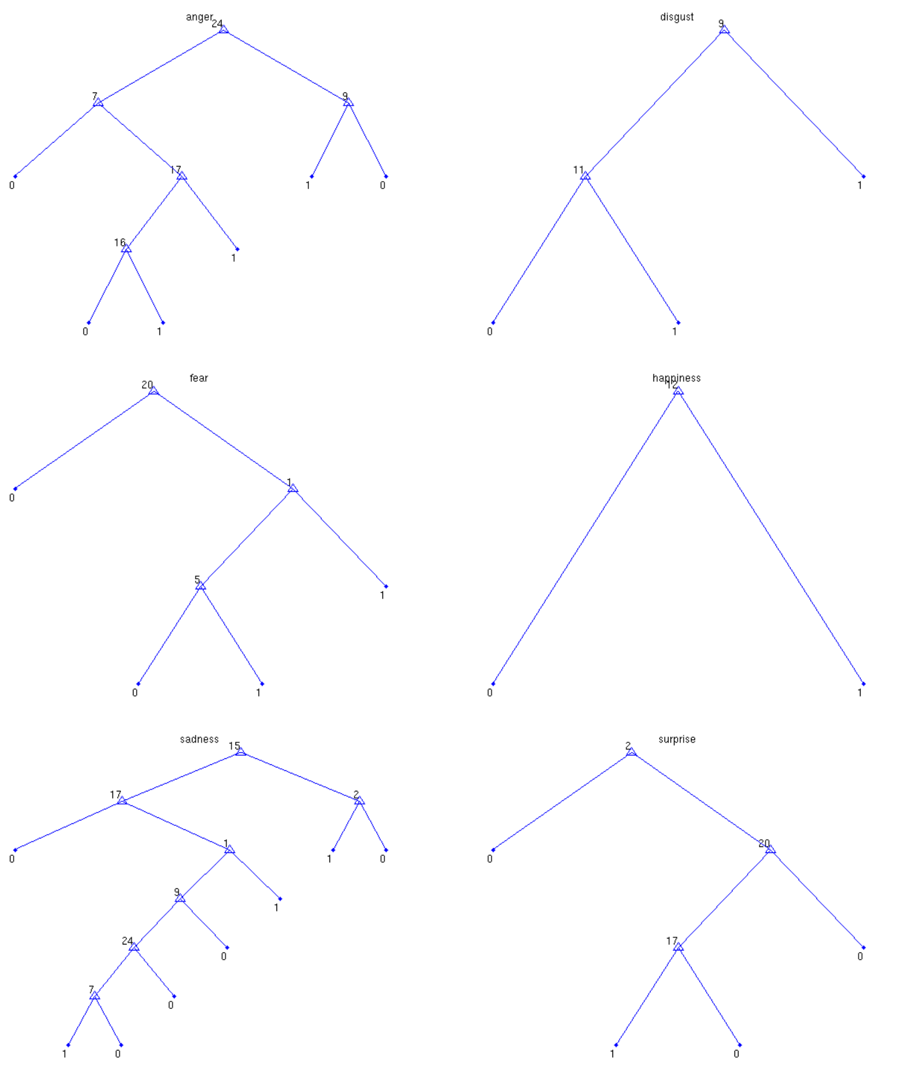

Figure 1: The trees

2 Average Cross Validation Classification Results

2.1 Confusion Matrix

A confusion matrix is a visualization tool typically used to present the results attained by a learner.

When we tested the created trees, we got the classificaitons for each set of AU's. Using this result we created the confusion matrices.

Here are the results:

	anger	disgust	fear	happyness	sadness	surprise
anger	12	0	0	0	0	0
disgust	0	22	0	0	0	0
fear	0	0	7	0	0	0
happyness	0	0	0	24	0	0
sadness	0	0	0	0	12	0
surprise	0	0	0	0	0	23

Table 1: Confusion Matrix after the initial training

	anger	disgust	fear	happyness	sadness	surprise
anger	8	0	3	0	1	0
disgust	0	22	0	0	0	0
fear	2	0	5	0	0	0
happyness	0	0	1	23	0	0
sadness	2	0	0	0	10	0
surprise	0	0	0	0	0	23

Table 2: Confusion Matrix after the 10-Fold validation

We can see that for the first case there are no wrong classifications since the data used to test the trees is the same with the data used to train them.

On the other hand, the second confution matrix tells us that our system is confusing some classes. For example, out of the total 12 anger emotions the system predicted that 3 are fear and one is sadness.

2.2 Average Recall and Precision Rates per Class

Recall and Precision measure the quality of an information retrieval process.

Recall describes the completeness of the retrieval.

$$recall = \begin{array}{|c|}\hline 0.6667 \\ \hline 1.0000 \\ \hline 0.7143 \\ \hline 0.9583 \\ \hline 0.8333 \\ \hline 1.0000 \\ \hline \end{array}$$

Precision describes the actual accuracy of the retrieval, and is defined as the portion of the positive examples that exist in the total number of examples retrieved.

$$precision = \begin{array}{|c|}\hline 0.6667 \\ \hline 1.0000 \\ \hline 0.5556 \\ \hline 1.0000 \\ \hline 0.9091 \\ \hline 1.0000 \\ \hline \end{array}$$

2.3 F_1 Measure

The F_α measure combines the recall and precision rates in a single equation:

$$F_\alpha = (1+\alpha)\frac{precision*recall}{\alpha*precision+recall}$$

where α defines how recall and precision will be weighted. In case recall and precision are evenly weighted then the F_1 measure is defined as follows:

$$F_1 = 2*\frac{precision*recall}{precision+recall}$$

From the results we obtain:

$$F_1 = \begin{array}{|c|}\hline 0.6667 \\ \hline 1.0000 \\ \hline 0.6250 \\ \hline 0.9787 \\ \hline 0.8696 \\ \hline 1.0000 \\ \hline \end{array}$$

3 Pruning

3.1 Pruning is an important issue in trees

Tree pruning is important in decision tree learning. We prefer shorter trees to classify new data as the time taken to classify this instance is shorter that using a deeper tree. The *pruning_example* method creates a single tree to classify the set of AU's to an emotion using the built-in matlab functions. It then performs several methods on the tree to find the optimal pruning point.

The simple idea of these methods is to plot the error in classification against the number of leaves (i.e. terminal nodes) that the tree contains. A tree with more leaves would be bigger because we are only dealing with binary attributes, whether the emotion has a specific action unit or not.

First we use all the data to train a tree and then we use a simple algorithm to prune this tree. The pruning algorithm works by looking at the node we want to be a leaf and set the value of this leaf to the class with the most instances in the data we still have to classify. We can then attempt to classify the data using the tree to obtain an error giving the amount of data that we ignored to get to this level of pruning. This does not give us the full picture however, as the data we are classifying now was used to train the tree in the first place.

An improvement on our pruning technique would be to apply the same technique to an 10-folded method, to create a tree based on 90% of the data and classify the remaining 10% and then average the error produced over all of the folds. This method is better than the previous method at giving us a reliable estimate at which level we should prune the tree because it is likely that there will be some data encountered that will not be in the training data.

The methods produce a vector of costs or errors for each of the pruning levels. We can simply choose the lowest error from this vector and prune the tree to this level. This data is then displayed in two graphs, one for each method, with the lowest error value identified.

3.2 Running the Code

After running the *pruning_example* function we got this two plots:

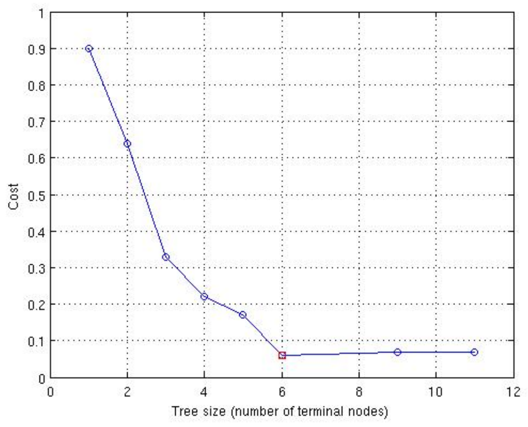

Figure 2: cross-validation

Figure 2 shows the plot of Cost vs Tree size when a tree is tested using 10-fold cross-validation to compute the cost vector. We can see that the cost decreases as the tree size increases until the size of the tree size is equal to 6. Also we have to mention that this method choose random the training and testing test and that results to different results each time.

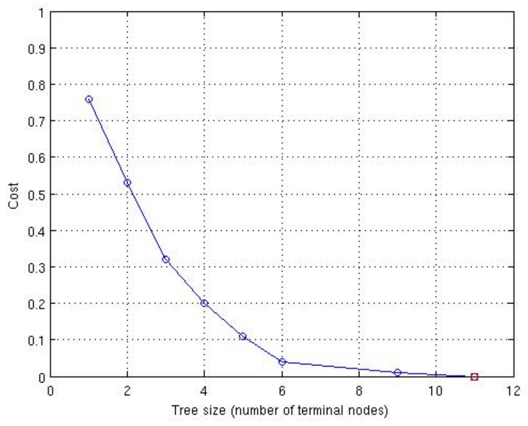

Figure 3: resubstitution

Figure 3 shows the plot of Cost vs Tree size when a tree is tested using using a resubstitution method to compute the cost vector. We can see that the cost decreases as the tree size increases until it reaches zero when teh tree size is equal to 11.

3.3 Expalining the results

3.3.1 The tree

Below we have the tree created:

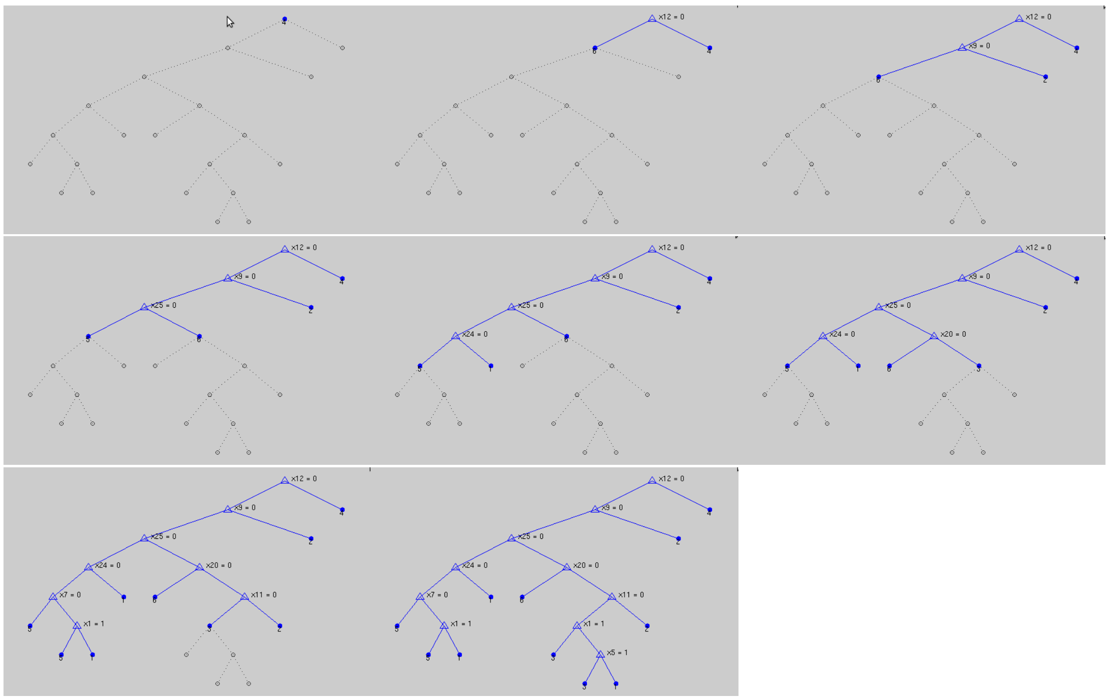

Figure 4: the tree

In Figure 4 we can see the 8 different pruning levels of the tree. In the first level we have the value 4 (happyness) since that is the most probable case and makes the error minimoum. Further done, at the sixth pruning level, we observe that all 6 emotions can be resulted. Finally the tree goes until the eightth pruning level were the cost for the resubstitution method is zero (All examples are classified correctly).

3.3.2 Cross VS Resubstitution

Below we have the the two plots generated:

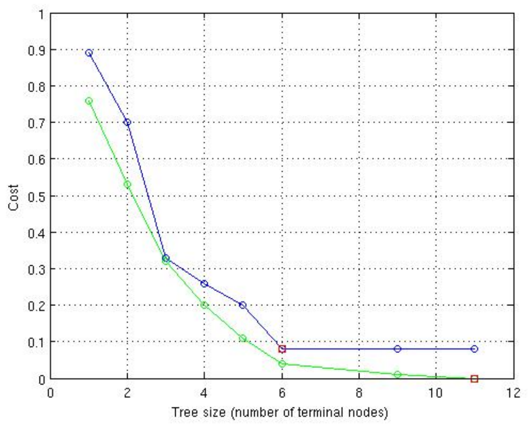

Figure 5: Cross VS Resubstitution

In Figure 5 the green plot is from the resubstitution method and the blue plot from the 10-fold cross-validation method.

The main differences deteween the two plots is that the resubstitution one smoothly decreases to zero as the number of terminal nodes increases but the cross-validation decreases unsmoothly until it reaches a minimum and goes on with out improving. That happens because the is tested with examples that were not used to train the tree.

The two red squares repsresent the minimun value of the cost. That point is also the best point to prune the tree since the cost wont improve in greater depths. So we can prube the cross-validation tree when the tree size is equal to six which will increase the speed of searching the tree.