Machine Learning CBC Assignment 2 Report

Aadil Khan, Gareth Jones, George Eracleous, Valdas Kriauciukas

Email: {aak08, gdj08, ge108, vk308}@imperial.ac.uk

Group Number: 23

Tutorial Helper: Brais Martinez

Table of Contents

1 Decision Trees

We create a decision tree for each of the basic emotions (anger, disgust, fear, happiness, sadness and surprise), where the method used to choose the best attribute to branch on came from the ID3 algorithm. The ID3 algorithm achieves its task by searching for an attribute with the largest information gain. With our implementations of the decision tree algorithm and its associated functions, the trees shown in Figure 1 were generated.

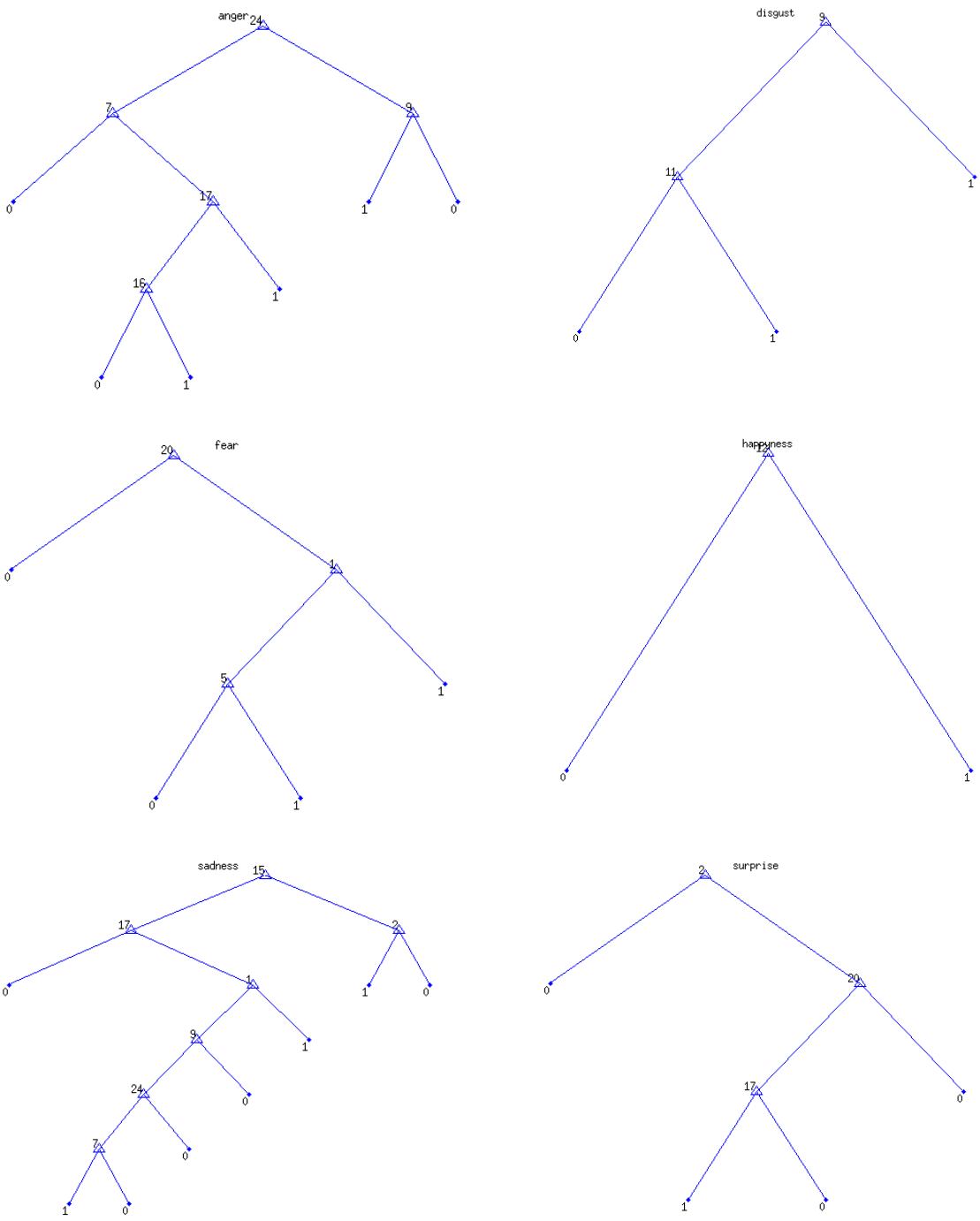

Figure 1: The decision trees for six basic emotions

2 Implementation Details

2.1 One-versus-all classification

In our implementation we have adapted the one-versus-all classification method which reduces the problem of k-class classification into k binary classification problems. More specifically, for each emotion we construct a decision tree which should output 1 if the example belongs to the emotion of this tree and 0 if the example belongs to any of the other emotions. In order to do this we wrote a small function which remaps targets according to a specific emotion and then constructs the decision tree based on ID3 algorithm. One-versus-all has the advantage that we end up with a smaller number of classifiers. However, the classifiers are trained on a small set of positive examples and a large set of negative ones. Therefore, this leads to an unbalanced training set. Additionally, the size of the trees is larger as opposed to the trees produced by the pairwise classification method. More important, one-versus-all classifiers may assign an example to more than one emotion or none at all which leads to ambiguities. The figure below depicts the problem for 2-dimensional feature space.

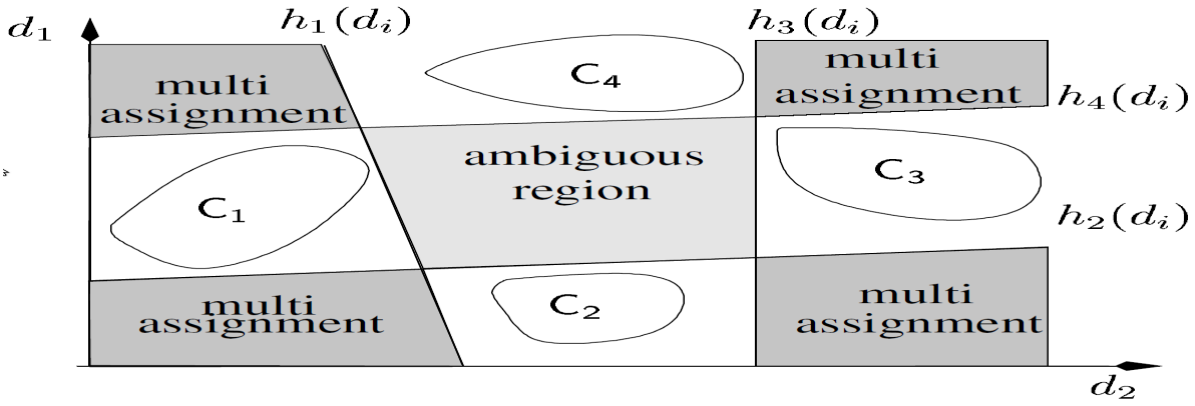

Figure 2: Source: Granitzer, M., 2003. *Hierarchical Text Classification using Methods from Machine Learning*. Available at: http://bit.ly/wGCygR.

2.2 Combining the binary decision trees and dealing with ambiguities

In order to evaluate our design we use 10-fold cross validation. More details about 10-fold validation can be found in the next section. First we discuss how we combine the six individual decision trees to get a single emotion using the *testTree* function. The function takes as input the set of decision trees constructed by the ID3 algorithm and the set of test examples. For each example in the test set, we use our decision trees to get the predictions from each one. In order to get the predictions, we recursively parse each tree until we end up with either a positive or a negative classification. After we get a vector of predictions we call the pickOneEmotion function which decides which prediction should be selected. In the case where there is only one positive prediction we just pick that one.

However, as we mentioned above, in some cases multiple classifiers (or none of them) classify an example positively. In order to alleviate this problem, we decided to compare two different

strategies. The first one selects an emotion randomly when a tie or no classification occurs. The second method assigns the prediction of the classifier with the largest depth in the case of no classification or a tie. The former method is the simplest one and it can resolve the ambiguities but relying on chance it is not the best way to fix the problem. Therefore, the second technique aims to solve the problem in a smarter way. The rationale behind selecting the matched emotion with the highest depth is that the higher the depth the greater the similarity between the example and the class represented by the classifier. An alternative strategy which we have tried is to always assign the first positively predicted emotion in the case of ties and a random one in the case of no prediction at all. However, this method was discarded immediately since it introduces a bias in our predictions since it always favors certain emotions.

We have validated our classifiers with the two aforementioned methods and we calculated the precision, recall and F metric in each case. The results of the second method resulted in generally higher F measures but in order to assert the effectiveness of the technique we must perform significance tests. The results are presented in the next section.

3 Average Cross Validation Classification Results

3.1 Results of the cross validation experiment

Our classifiers are evaluated using 10-fold cross validation. We have created a function that splits the data-set into 10 different pairs of training and test examples. 90% of the examples are used for training and the rest are used for testing. For each pair of training and test examples we first construct the six decision trees based on the training set and then by calling the *testTree* function we calculate the predictions for the test set. The prediction labels of all 10 runs are then fed to a function which constructs the confusion matrix. Based on this matrix we calculate the precision, recall and F metrics. The results are presented below comparing the performance of our classifiers using the two different methods used to resolve the ambiguities.

3.2 Confusion Matrix, Recall, Precision and F_1-Measure

The matrix below combines all 10 runs in the data-set and from that matrix, statistics in Figure 3 are calculated. The error measure is calculated by taking TP+TN/(All matrix). It shows percentage examples incorrect, but is not a representative metric. Recall is calculated by taking TP/(TP+FN), where denominator is whole row for selected emotion. Precision = TP/(TP+FP), where denominator is whole column for selected emotion. Alpha values of 1 were used in calculating F-Measure.

3.2.1 Random ambiguity resolution method

	anger	disgust	fear	happyness	sadness	surprise
anger	7	1	1	0	3	0
disgust	0	22	0	0	0	0
fear	2	0	4	0	1	0
happyness	0	0	1	23	0	0
sadness	1	0	0	0	11	0
surprise	0	0	0	0	0	23

	Error Measure	Recall	Precision	F-Measure
anger	8	58.3333	70	63.6364
disgust	1	100	95.6522	97.7778
fear	5	57.1429	66.6667	61.5385
happyness	1	95.8333	100	97.8723
sadness	5	91.6667	73.3333	81.4815
surprise	0	100	100	100

Figure 3: The confusion matrix (top) and statistics table (bottom) from cross validation that used a simple *testTree* function, that randomly assigns labels for ambiguous examples

3.2.2 Tree depth ambiguity resolution method

	anger	disgust	fear	happyness	sadness	surprise
anger	6	0	1	0	5	0
disgust	1	21	0	0	0	0
fear	3	0	4	0	0	0
happyness	1	0	0	23	0	0
sadness	2	1	0	0	9	0
surprise	0	0	0	0	0	23

	Error Measure	Recall	Precision	F-Measure
anger	13	50	46.1538	48.0000
disgust	2	95.4545	95.4545	95.4545
fear	4	57.1429	80	66.6667
happyness	1	95.8333	100	97.8723
sadness	8	75	64.2857	69.2308
surprise	0	100	100	100

Figure 4: The confusion matrix (top) and statistics table (bottom) from cross validation that used a tree depth algorithm in *testTree* function, that uses classifier tree depth for assigning labels to ambiguous examples

The performance of the more complex classifier method is comparable to random assignment method. It would probably be significantly better than random in a more numerous classifier environment.

In both cases we can easily observe that the surprise classifier is extremely accurate classifying correctly all the instances of this emotion. Disgust and happiness exhibit similar performance close to that of the surprise classifier. The other three emotions anger, fear and sadness are classified less accurately. A possible explanation for these results is the fact that in the given data-set; anger, fear and sadness are represented by far less examples (12, 7 and 12 examples respectively) than the other emotions. Therefore, during the training phase the algorithm was biased towards the other emotions.

4 Pruning

4.1 To Prune a Tree

The size of decision trees are able to be reduced through a method called *pruning.* The motivation behind a size reduction is to generate a decision tree capable of classifying new data quickly with minimal effort. Traversing a shorter tree would take less time than a tree with great

depth. The accuracy of a decision tree classifier increases as pruning reduces the chances of over-fitting. With a large decision tree, over-fitting of the training data has a higher chance of occurring and, consequently, being unable to generalize to new data. Though a large tree is useful to potentially provide a predictor that is accurate 100% of the time, it is preferable to generate a tree with optimal size, capable of generalizing to new samples. There are two particular techniques in which to go about pruning; Reduced-Error Pruning and Rule Post-Pruning.

After splitting the data into training and validation sets, Reduced-Error Pruning removes a sub-tree rooted at node *X* (thus making *X* a child node), then assigning *X* with the most common classification of those nodes that were below it. This is done repetitively until pruning becomes harmful. To make such a judgment, one should evaluate the affect on the validation set of pruning each possible node or when removing a node that had initially improved the validation set accuracy.

Rule Post-Pruning converts the decision tree into an equivalent set of rules to help distinguish the various contexts where a decision node may have been used. Each rule is then pruned independently of each other by removing any preconditions that result in an improved accuracy of a classification. The pruned rules are then sorted by the derived accuracy or by any order one would consider for further classifications.

4.2 Running the 'pruning_example'

The *pruning _example* method creates a single tree to classify the set of AU's to an emotion using built-in Matlab functions. All of the data is used to train a tree before it is pruned. The technique used is similar to that of Reduced-Error Pruning. Data is classified using a particular tree and an error is obtained, reflecting the amount of data that was ignored at a particular level of pruning. However, any information on the cost of pruning shouldn't be taken lightly as the data used for classification is the same as the one used to train the decision tree.

Instead of using the whole data set, results of pruning can become more reliable with 10-fold cross validation. A tree is trained on the grounds of 90% of the given data while the remaining 10% is used in validation of the tree's ability in classification. While a newly created tree has not encountered the test data, we can retrieve a more reliable estimate on the level of pruning.

The *pruning_example* method generates a vector of errors (*costs*) for each pruning level. Determining the level to prune the tree is done by choosing the lowest error from the vector. The two graphs that are displayed highlight the lowest error value that we desire.

4.3 Resubstitution vs Cross-Validation

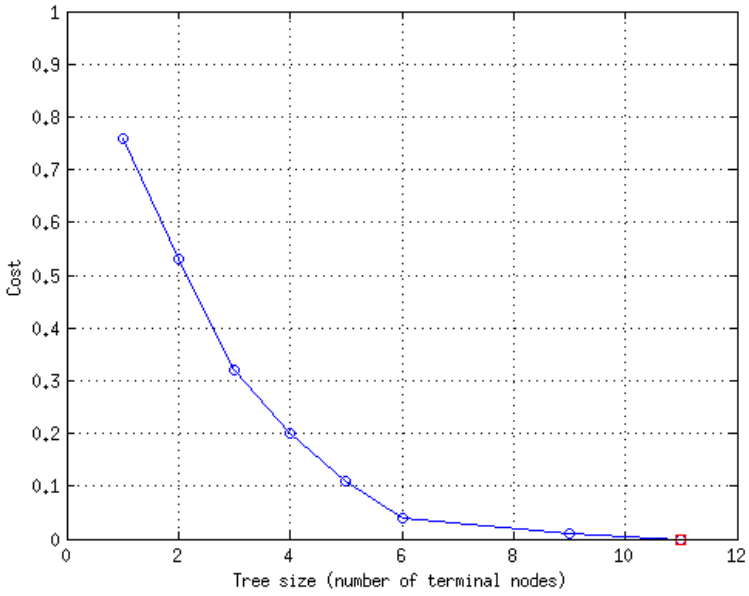

Figure 5: Plot for Resubstitution

Figure 5 shows the plot of Cost vs Tree Size where the cost (error) vector is computed when a tree is tested through the use of a resubstitution method (i.e. 100% of the data is used to train the decision tree). While the graph makes a steady decline, we deduce the tree size to be eleven as the level in which a decision tree is at it's optimum.

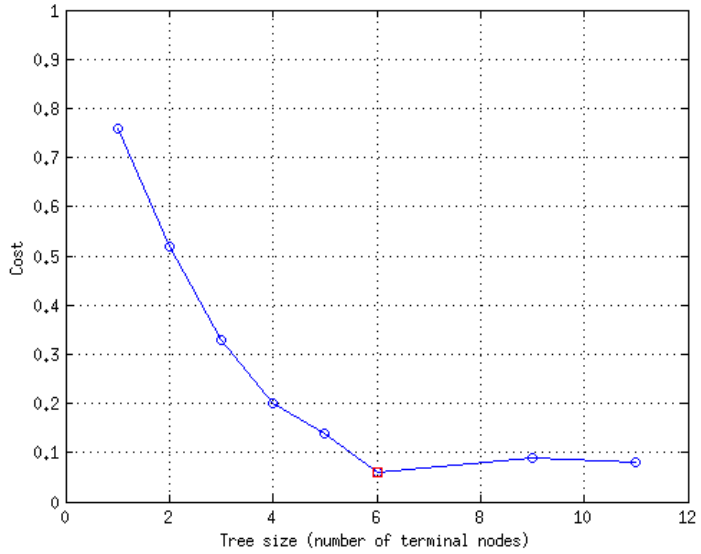

Figure 6: 10-fold Cross Validation

Figure 6 shows the plot of Cost vs Tree Size where the cost (error) vector is computed when a tree is tested through the use of 10-fold cross validation. It can be seen here that where the cost decreases as the tree size increases, a level of six is suggested to overcome over-fitting while remaining as a tree capable of accurate classifications. In addition, the plot reveals that any further increase of the size of the tree would regress the tree's ability in a credible classification.

Beyond six terminal nodes, over-fitting of data becomes, yet again, a problem.

A primary difference to note is that resubstitution decreases steadily to a minimum cost of zero while the number of terminal nodes increases. Whereas, the method of cross validation does not make such a smooth descent, however, does reach a minimum much sooner than resubstitution before discovering that a larger tree size does not help in improving the accuracy of the tree. This distinction, the shape of the plot, exists because the tested examples used for cross validation were not used during the training of the decision tree.

The red squares on the graphs denote the minimum value of the cost. This point has been suggested the optimal level in which to prune the tree. For cross validation, a level of six will decrease the time taken to search the tree (while retaining an accuracy in classification) while resubstitution would prefer a depth of eleven terminal nodes.