Report on Assignment 2 - Decision Trees

Group 23: David Toth, Vinea Au Yeung, Tina Liu

Introduction

Detection of an emotion of a human face is based on its extracted action units or attributes. The classifier function needs to learn the target function - a linear combination of these attributes. Throughout this project, we keep consistency with the CBC manual, so the emotions which are labelled 1 to 6, correspond to anger, disgust, fear, happiness, sadness and surprise respectively.

Summary on Implementation Details

We implemented the **decision_tree_learning.m** function following the given pseudo-code, which recursively builds a binary decision tree for a specified emotion label. We have used this function to construct six trees, by training them on the entire provided example data set (see Figures 3-8 in Appendix). Each tree classifies examples according to whether an emotion is present or not. Our final goal was to produce a classifier which combines the decision trees to classify examples to exactly one of the six emotions.

majority_value.m takes an array of binary values (zero's and one's) and returns the more frequent value.

choose_best_decision_attribute.m chooses the attribute that gives the highest information gain. It takes as argument the array of remaining attributes and for each attribute, calculates the information gain by executing *find_entropy.m* and subtracting from this the result of *find_remainder.m*. It then returns the index of the attributes vector which corresponds to the specific attribute with the highest information gain.

Cross validation

We used 10-fold cross validation which involves dividing the available data into 10 parts: 9 parts are used for training and 1 part for testing. The process is repeated 10 times, in each fold a different part is selected for testing. We first randomise the given example set, and for each fold, split them into training examples and testing examples based on the current fold number. For example, after randomising the data set, if fold_number = 3, we will set aside the 3rd (approximate) tenth of the original data set to be our test set, by rounding up our computed starting index and ending index of the test data, and the remaining 9 tenths will be used to train our trees.

Computing the average results

Since our cross validation consists of 10 folds, we loop *cross_validate.m* 10 times in our main function *statistics.m* and each set of results is computed and returned in the form of a struct from cross_validate.m. Then we access the results through the structs, and their arithmetic means are calculated. In the case of the confusion matrices, we sum up the 10 confusion matrices and divide their entry values by 10. The average classification rate was calculated by 1 - (average error rate over all 10 folds).

Classifiers

We had several ideas as to how we would combine the six trees into one classifier and in the end, we decided to combine these ideas into one classification algorithm. This proved to be the most effective as it produced the best average classification results. The algorithm is based on Tree Height, Entropy and Number of Examples. More details are provided in the Ambiguity Question section.

Program Flow

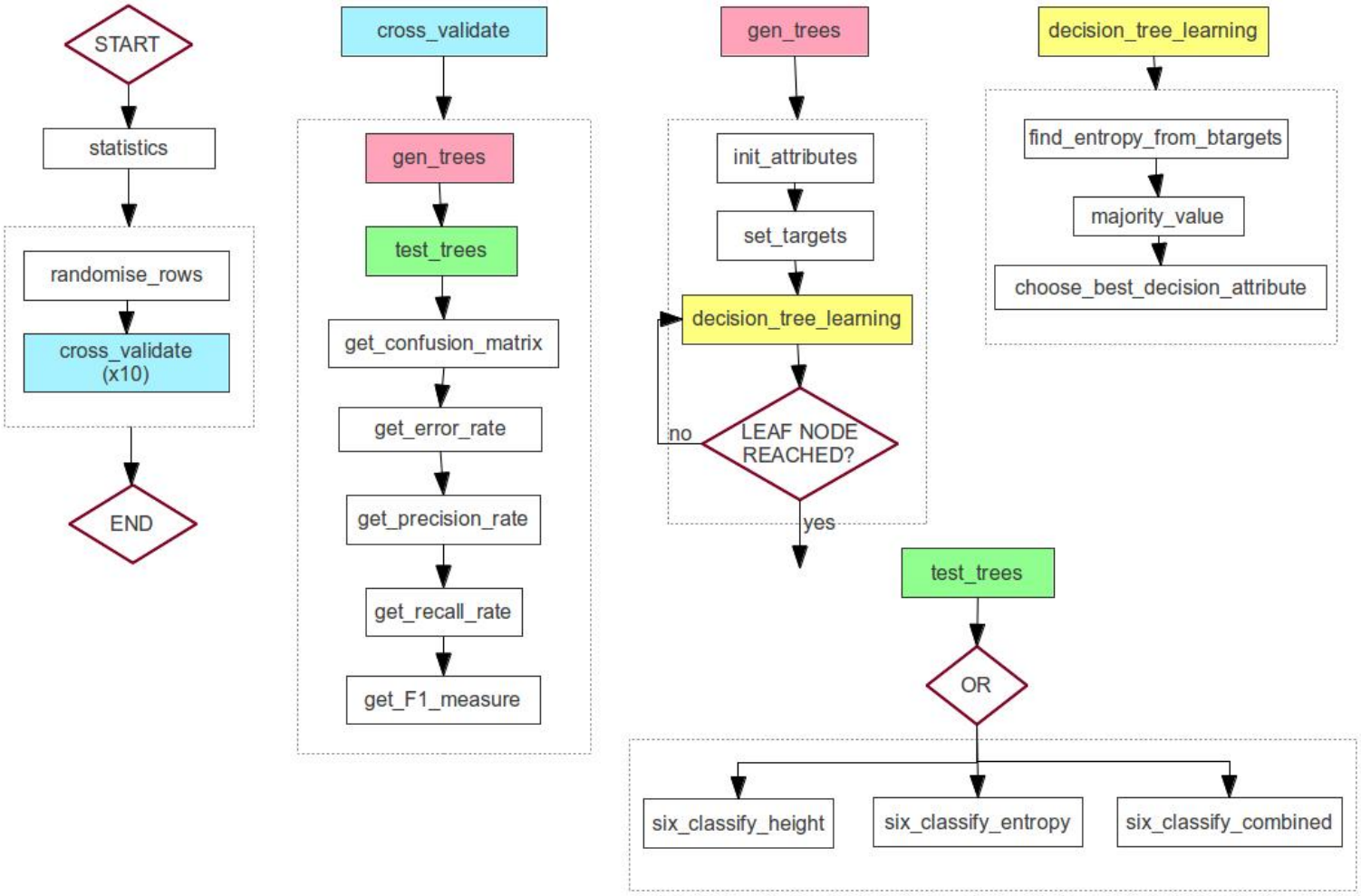

Figure 1: Flow chart showing the sequence of invoked functions

The above flowchart (Figure 1) describes the stages of our running program. The sequences of functions enclosed in the dotted lines allows us to represent our function hierarchy. For instance, statistics.m calls randomise_rows.m and cross_validate.m. Then cross_validate.m (highlighted in blue), in turn, calls gen_trees.m followed by test_tree.m, etc.

Our main function is statistics.m which takes as inputs: the matrix of examples and the target vector. The cross-validate.m function is invoked 10 times by statistics.m. Its purpose is to carry out one fold of the cross validation so we obtain a training set (9/10th of the original data set) and a test set (1/10th of the original data set). It then generates six trees (one corresponding to each emotion), combines the six trees into one classifier and tests the classifier on the allocated test set. Based on the results of this test, it computes the following statistics for each fold: a confusion matrix, the error rate, and for each class the precision and recall rates, and the F1 measures. Once this process is completed for all 10 folds, we obtain the average of these results in statistics.m. The methods which are used for combining the six trees into one classifier are

implemented in six_classify_height.m, six_classify_entropy.m and six_classify_combined.m.

Results of the Evaluation

Average Classification Rate: 0.6950

The Confusion Matrix:

		Predicted Emotions					
		1	2	3	4	5	6
Actual Emotions	1	8.6	1.3	0.8	0.4	2.0	0.2
	2	2.1	14.5	0.3	1.1	1.2	0.7
	3	1.6	0.3	7.3	0.1	0.5	2.2
	4	1.2	1.3	0.3	18.3	0.5	0.2
	5	2.8	1.3	1.0	1.0	5.6	1.6
	6	1.1	0.5	1.7	0.1	1.5	16.1

Table 1: Confusion Matrix

From the confusion matrix (Table 1) we can see that the elements corresponding to same predicted emotions and actual emotions (in yellow) have the highest values in both the columns and rows, indicating that given an example with its actual label, the probability of computing the right predicted emotion is higher than any other emotions for all of our trees. Given an example with its predicted emotion, the probability of the person having that emotion is highest compared to the rest.

	Predicted Emotions					
	1	2	3	4	5	6
Average Recall Rate	64.5	72.3	60.8	84.4	42.6	76.0
Average Precision Rate	49.5	75.6	63.7	86.9	50.6	76.2
Average F1 Measure	55.4	73.4	61.9	85.1	45.4	75.9

Table 2: Average Cross Validation Classification Results

Table 2 shows the classification results for each class (1 to 6), averaged over all the 10 folds. We can see

that for emotion 1, the average recall rate is noticeably higher than the average precision rate. This means that when a person has emotion 1, the tree has a fair chance of recognising its presence, while it only has less than half the accuracy in recognising the absence of the emotion: it often returns a positive result even when the emotion is not actually present. From the decision tree for emotion 1 (see Figure 3 in Appendix) we can see that there are some overfitted branches while others are quite short. This gives some evidence that the overfitted branches have evaluated the supposedly negative examples to positives.

In general, the precision and recall rates seem to be similar within each class. For emotion 5, the corresponding tree is not reliable at all as both rates are low. This can be seen in the 5th decision tree diagram (see Figure 7 in Appendix) since there are many obvious overfitting branches. We also see that emotions 2, 4 and 6 have relatively the highest F1 measures. We therefore conclude that when we are looking for the presence of these three emotions, our trained classifier is rather reliable in finding them, but not so for the rest.

Ambiguity Question

The decision_tree_learning.m function can train a tree for only one emotion, but since we need our classifier to classify an example to six possible classes we need a way to combine the six results of the six binary classifications. In a case when more than one tree returns a positive classification we need to decide which tree is more likely to be correct, and therefore resulting in classification to exactly one of the six emotions.

Tree Height from the root node to the leaf node which we used for classifying our example is a good indicator of the classification correctness. A smaller tree height suggests that the attributes used to classify the leaf node are more indicative of the correct classification. This is because fewer attributes produce a shorter hypothesis, thus a more probable one and less susceptible to the problem of overfitting. However in the case where the two compared heights are the same or similar, this method will only be as good as picking a random label.

Entropy measures the uncertainty of information in a system. 45 attributes might not be sufficient to classify an example correctly. Moreover, in the real world the training data is often noisy. The majority_value.m function chooses the most likely classification, however preserving the actual entropy as well can provide us with a measure of uncertainty of our classification. Having six entropies for all six binary classifications, we can solve the issue of having two positive classifications by choosing the one with the lowest entropy - with the highest certainty. We have created a new field of the tree struct called tree.entropy to store this information for each of the leaf nodes.

Multiple Trees for each pairwise emotion combination could provide more information needed to resolve the ambiguity. When faced with a decision between anger and happiness, a tree trained on examples representing only these two target emotions would be used. However, the prespecified function test_trees.m takes as input arguments only the six trees so we could not apply this technique.

Number of examples classified at the leaf node corresponds to having more examples testifying the correctness of the classification. We have created yet another field in the tree struct called tree.size which stores the number of training examples which were classified at each leaf node. Entropy alone cannot express this information, for instance, it does not tell us whether four training examples were classified as positive at a given leaf node or just one. But we might also have examples classified incorrectly at a node

due to noise or the lack of features in a training set. Hence this method needs to be combined with the entropy to be effective.

Combination of the approaches chooses the best of every world. We combined Tree Height, Entropy and Number of Examples into one classifier: six_classify_combined.m. However, we were unable to analyse our approach rigorously in a theoretical manner. Our decisions on exact combinations were based on the empirical statistical results calculated during the cross-validation process. The combined classifier uses a linear combination of the measures expressed by the 3 classifiers.

Pruning Example

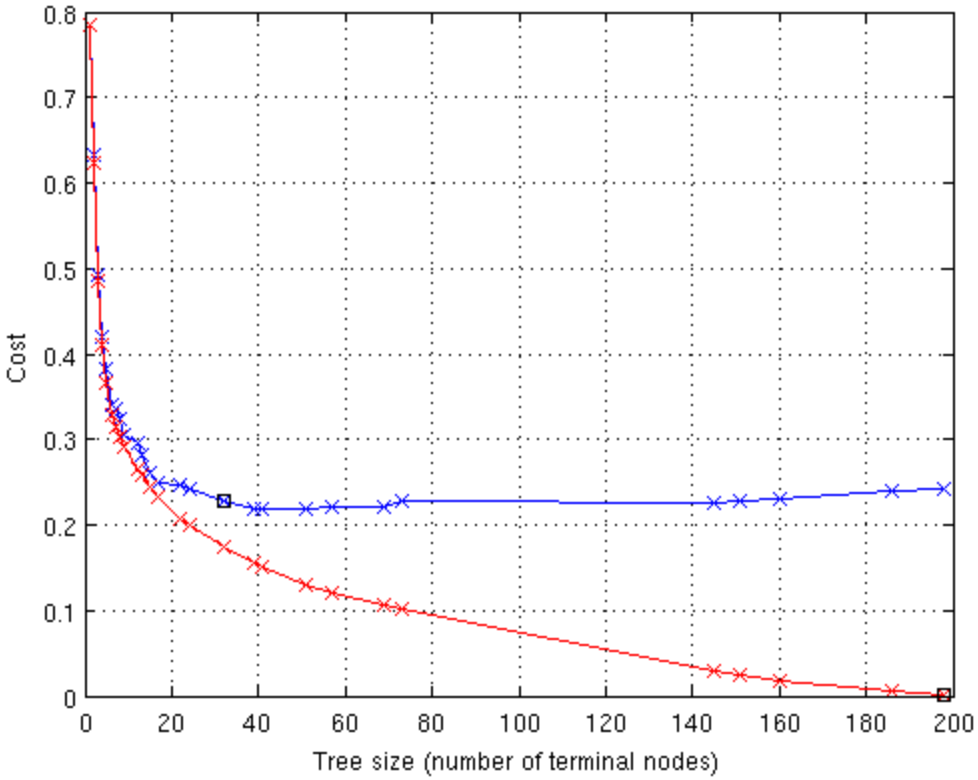

Figure 2: Graph produced from pruning_example.m

The pruning_example.m function creates a tree based on the training sample (x, y), then it performs two statistical methods measuring the error rate of a classifier. These can be seen in Figure 2 as cross-validation (blue) and resubstitution (red). Resubstitution usually underestimates the error, whereas cross-validation produces an unbiased estimate, however it is more variable. According to cross-validation, the minimal error rate is for a tree with 16 leaf nodes. According to resubstitution, the optimal tree size is 200 leaf nodes. Thus the optimal tree size is between 16 and 200 leaf nodes.

Summary

Decision trees are good for simple binary classification problems, however the presence of multiple classes poses several problems. Having multiple branches (one per class) at the node makes the measure of the information gain less indicative. Creating one tree per class and training each for binary classification creates the Ambiguity Problem on the other hand. The creation of an effective classifier is often hard and impossible. Based on our intuition and measurements we believe that other methods (like ANN) are more appropriate for these types of problems.

Appendix – Decision Trees

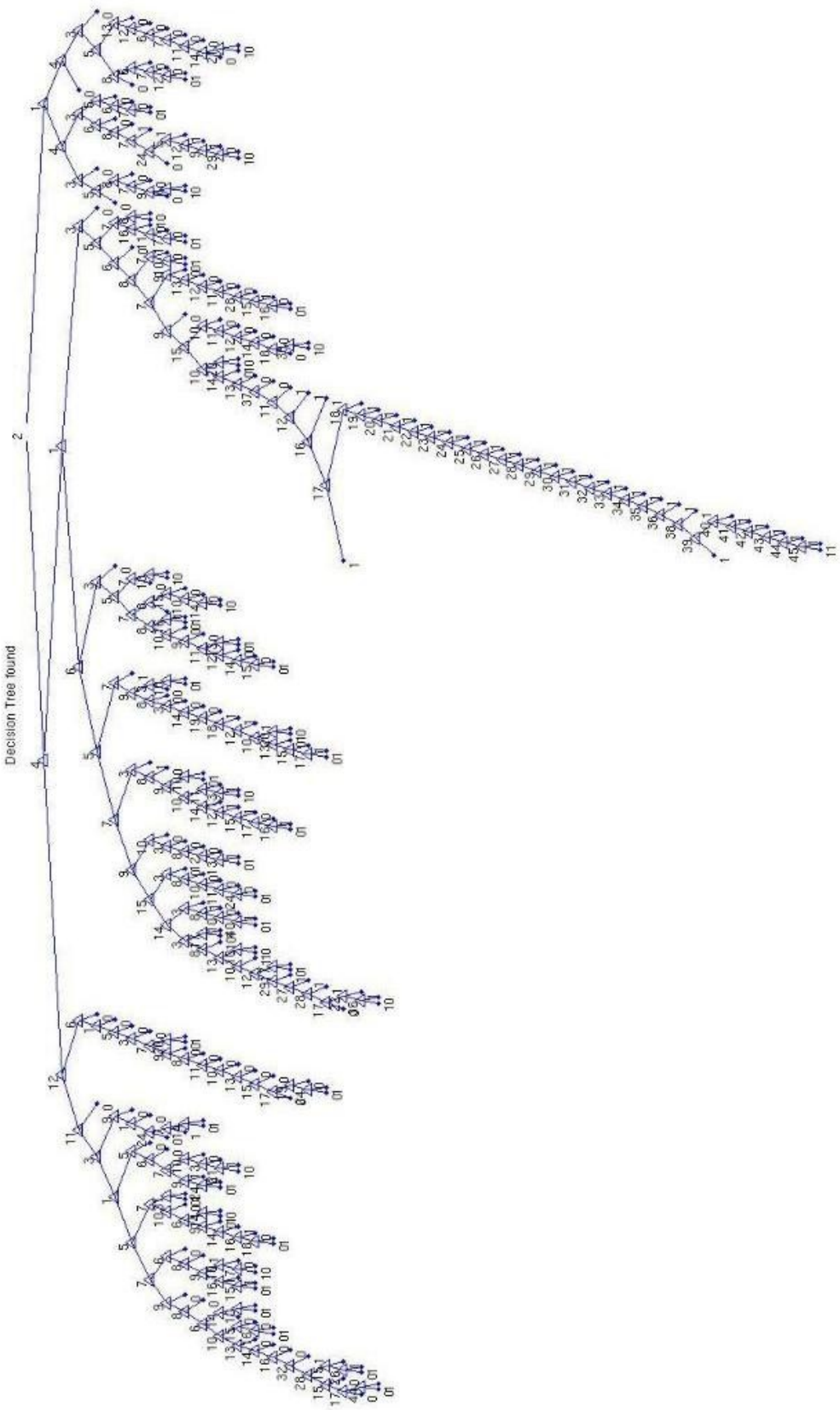

Figure 3: Decision tree for emotion 1 (anger) trained on the entire data set

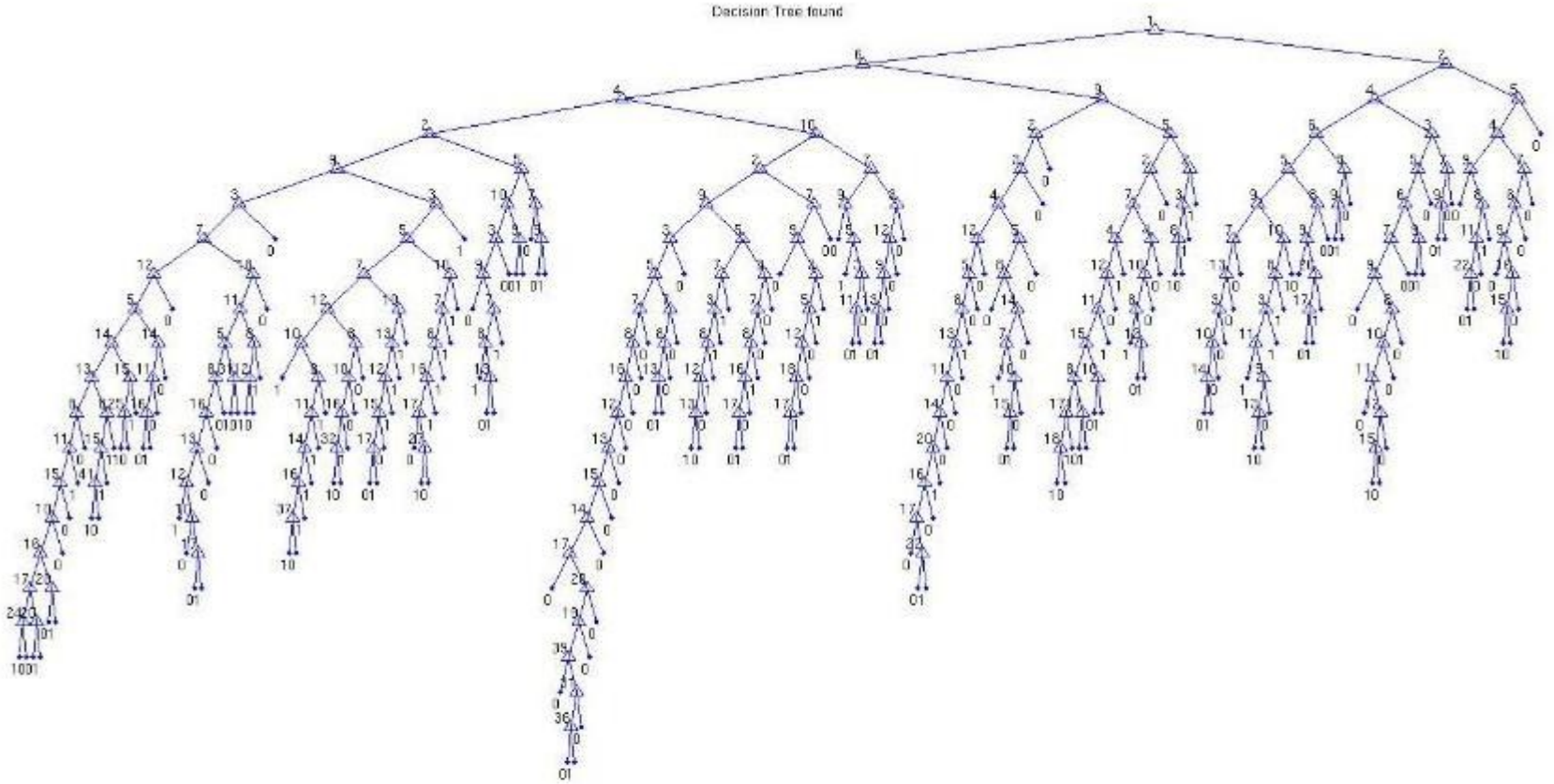

Figure 4: Decision tree for emotion 2 (disgust) trained on the entire data set

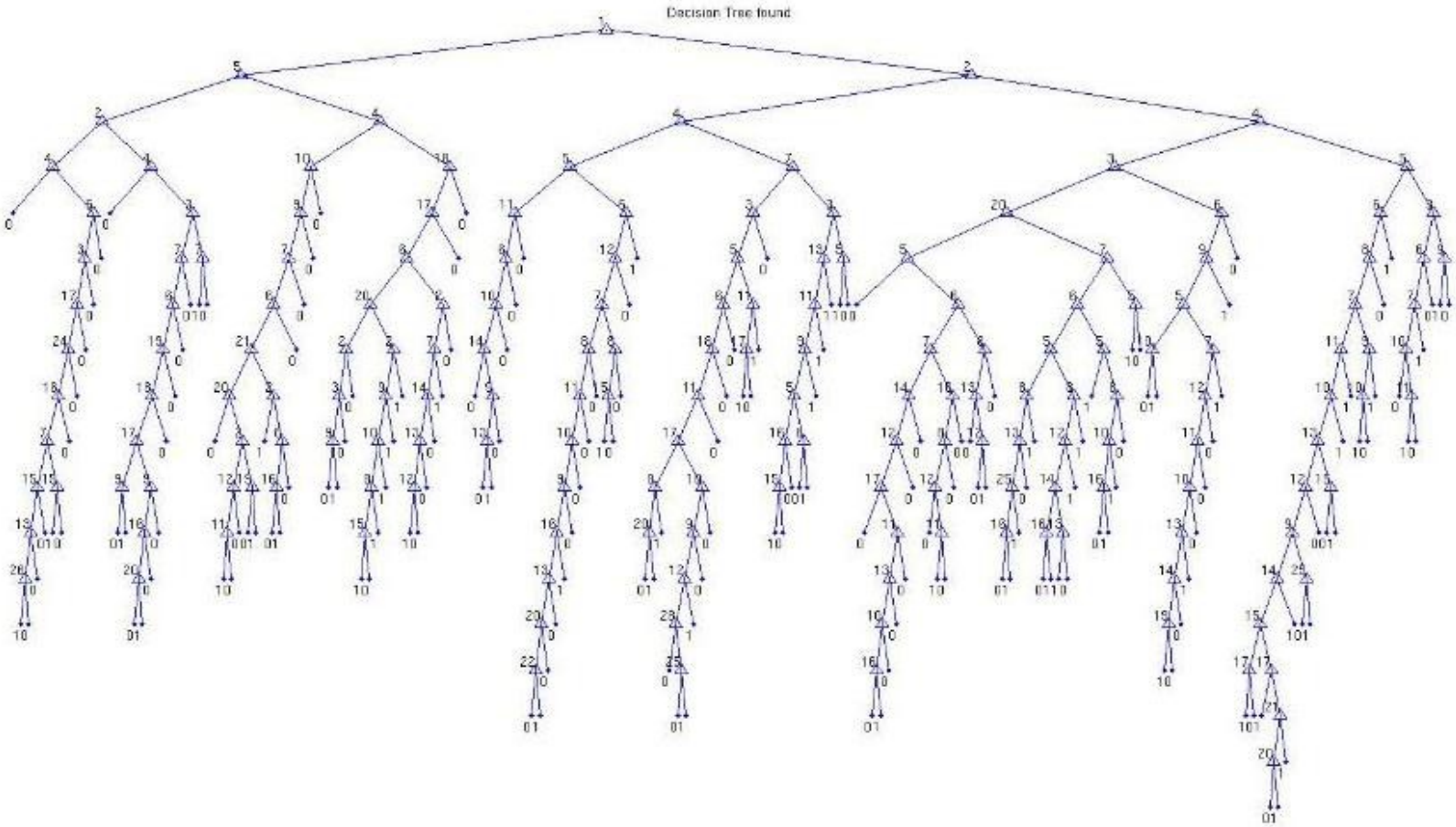

Figure 5: Decision tree for emotion 3 (fear) trained on the entire data set

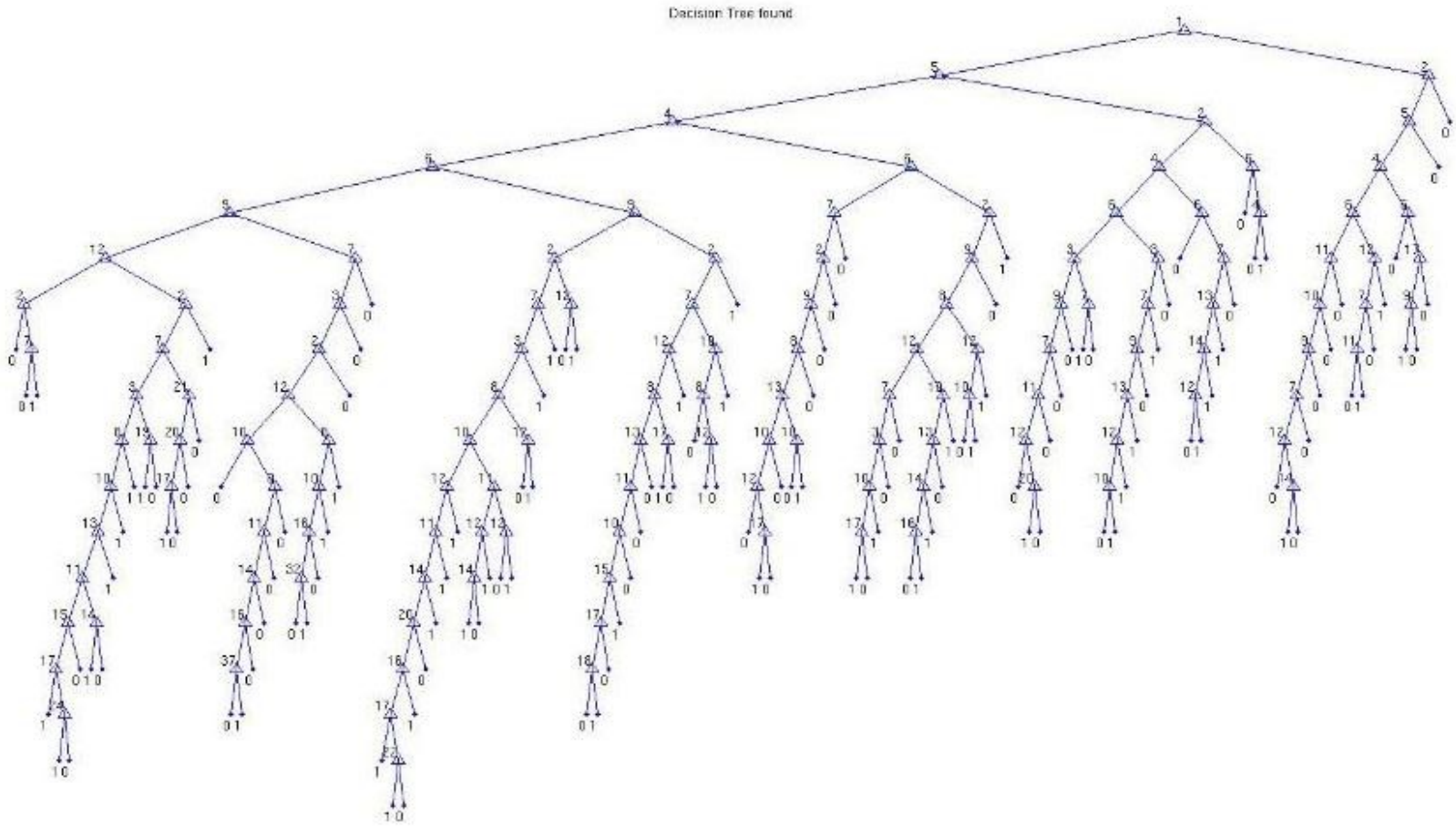

Figure 6: Decision tree for emotion 4 (happiness) trained on the entire data set

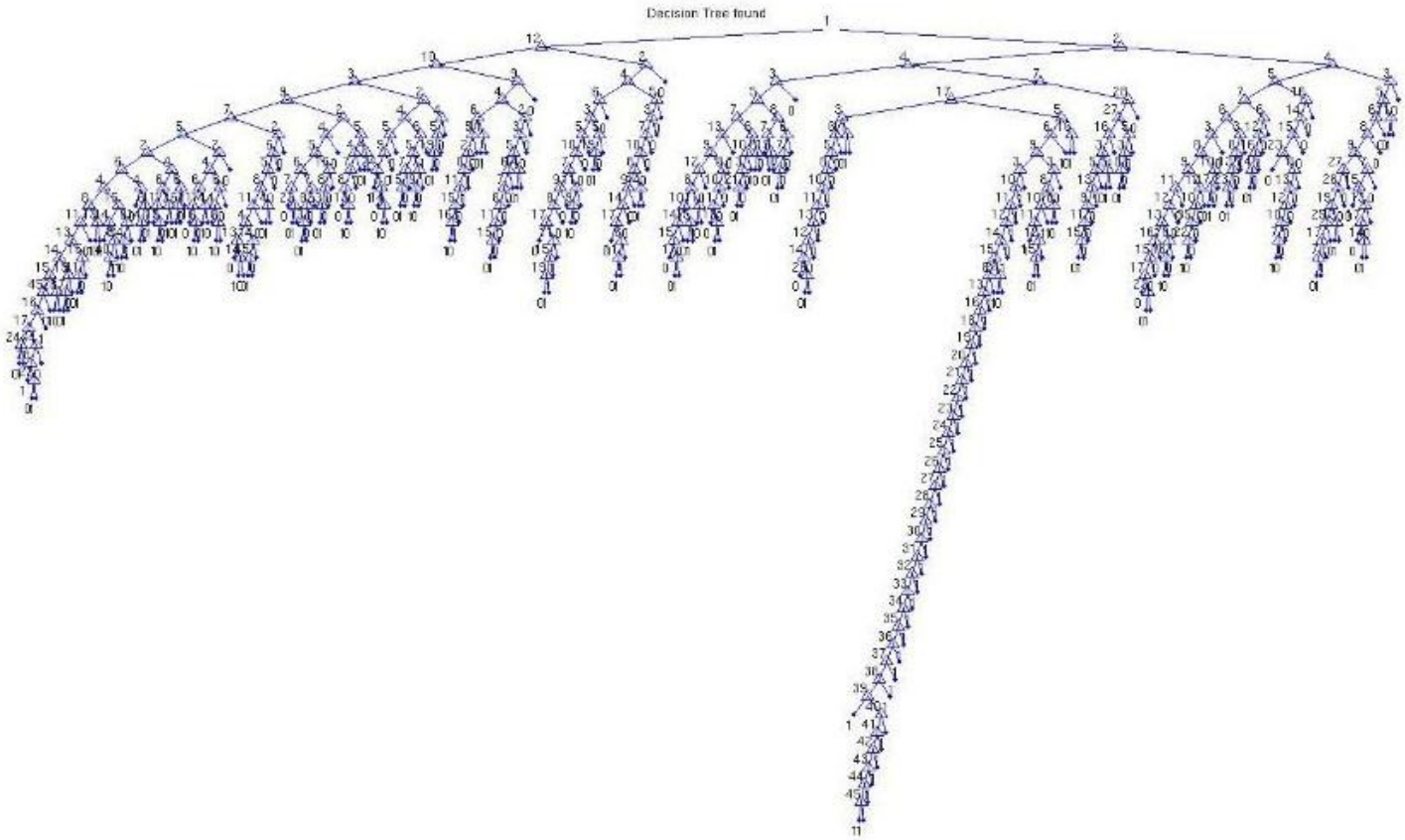

Figure 7: Decision tree for emotion 5 (sadness) trained on the entire data set

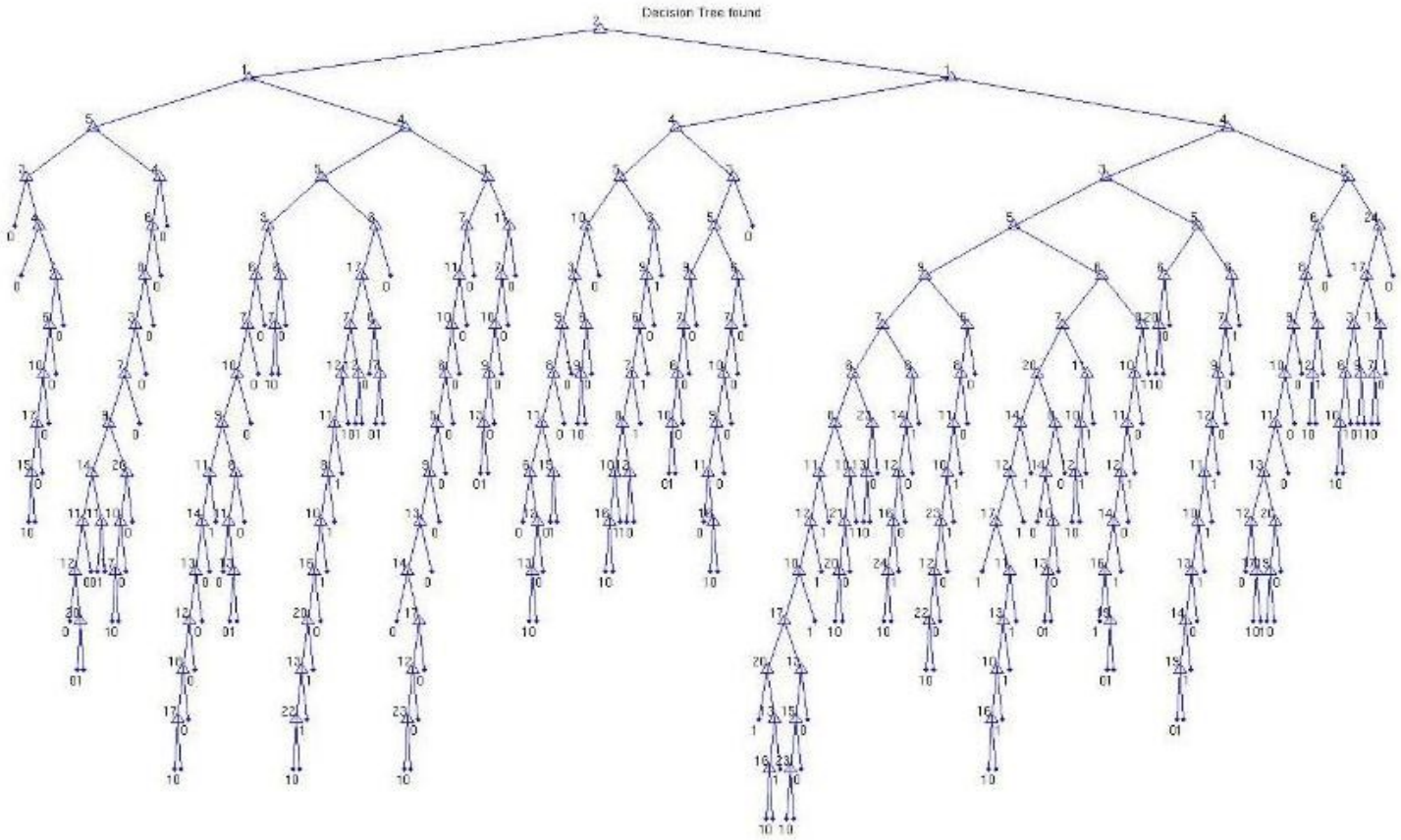

Figure 8: Decision tree for emotion 6 (surprise) trained on the entire data set