CO395 Group 57

Dongxiao Huang, Zheng Xun Phang, Yufeng Zhang

Implementation

How do we select the best attribute at each node? For all attributes, we compute the information gain if the dataset is split on that attribute:

$$\text{Gain(Attribute)} = \mathcal{I}(p,n) - \left[\frac{p_0+n_0}{p+n}\mathcal{I}(p_0,n_0) + \frac{p_1+n_1}{p+n}\mathcal{I}(p_1,n_1)\right]$$

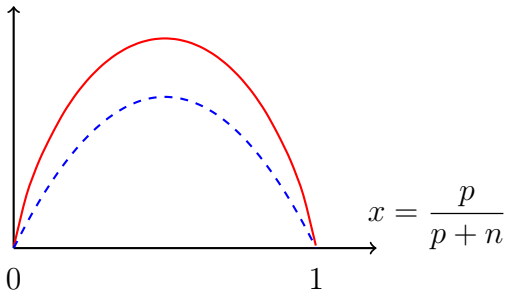
$$\begin{aligned} p_k &= \text{Number of positive examples with attribute} = k \\ n_k &= \text{Number of negative examples with attribute} = k \\ p = p_0 + p_1 &= \text{Number of positive examples before split} \\ n = n_0 + n_1 &= \text{Number of negative examples before split} \end{aligned}$$

There are two common ways to measure information.

Entropy:
$$\mathcal{I}(p,n) = -\frac{p}{p+n}\log\frac{p}{p+n} - \frac{n}{p+n}\log\frac{n}{p+n} \qquad \text{if } p,n \neq 0$$
$$\mathcal{I}(p,0) = \mathcal{I}(0,n) = 0$$

Gini impurity:
$$\mathcal{I}(p,n) = \frac{p}{p+n}\left(1-\frac{p}{p+n}\right) + \frac{n}{p+n}\left(1-\frac{n}{p+n}\right)$$

We used entropy since it is stated in the specification, but Gini impurity is faster to compute. Both metrics should give similar results since their graphs have a similar shape:



Entropy: $-x\log x - (1-x)\log(1-x)$

Gini: $2x(1-x)$

N.B. The *height* of those graphs is irrelevant because minimizing a function is equivalent to minimizing any positive multiple of that function. What matters is their *shape*.

The decision tree algorithm is given in the specification, so it's unnecessary to repeat it here. We try to use NumPy functions instead of Python loops for performance since the former run on vectorized C code.

Validation

To evaluate our decision tree, we performed K-fold cross validation as follows:

1. Shuffle the dataset and split it into $K = 10$ parts
2. For each $k \in \{1, \cdots, K\}$ we train the decision tree on the dataset *excluding* part k and then test the decision tree on part k. During testing, we aggregate the predictions from 6 trees (more details later) and increment the relevant cells in the confusion matrix.

Evaluation

Each cell of the confusion matrix is a total, not an average, over all folds of cross validation. There were $n = 1000$ test examples in total.

Note: It makes no sense to compute classification rate for each emotion separately. For example, if the actual emotion is Anger then (Fear, Surprise) counts as a true negative for Anger even though it's a misclassification!

Clean Data

Actual Class	Predicted Class					
	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Anger	106	11	3	5	5	1
Disgust	23	157	1	5	9	3
Fear	17	2	77	2	7	13
Happiness	22	6	1	179	6	1
Sadness	34	11	3	4	78	2
Surprise	27	3	7	4	4	161

We add up the relevant cells in the confusion matrix above to compute these summary statistics:

	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Precision	0.463	0.826	0.837	0.899	0.716	0.890
Recall	0.809	0.793	0.653	0.833	0.591	0.782
F_1 score	0.589	0.809	0.733	0.865	0.647	0.832

$$\text{Classification rate} = \frac{106 + 157 + \cdots + 161}{1000} = 0.758$$

Noisy Data

Actual Class	Predicted Class					
	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Anger	48	6	11	5	15	3
Disgust	38	127	9	7	3	3
Fear	42	10	108	13	7	7
Happiness	34	8	6	153	2	5
Sadness	46	2	6	4	48	4
Surprise	33	2	15	7	8	155

Summary statistics:

	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Precision	0.199	0.819	0.697	0.810	0.578	0.876
Recall	0.545	0.679	0.577	0.736	0.436	0.705
F_1 score	0.292	0.743	0.631	0.771	0.497	0.781

$$\text{Classification rate} = \frac{48 + 127 + \cdots + 155}{1000} = 0.639$$

The F_α score by van Rijsbergen (1975) is a harmonic mean of precision P and recall R:

$$\frac{1+\alpha^2}{F_\alpha} = \frac{1}{P} + \frac{\alpha^2}{R} \qquad \Rightarrow \qquad F_\alpha = \frac{(1+\alpha^2)PR}{\alpha^2 P + R}$$

which allows us to adjust the weight given to precision or recall. This is useful if, for example, it's essential to detect Anger in an image when it's present. This averaging also enables F_α to deal with imbalanced classes.

On both noisy and clean data, our 6 trees classify the emotions *Disgust*, *Happiness* and *Surprise* with high accuracy, while *Anger* and *Sadness* are harder to recognize.

We also find that all emotions except *Anger* are most often misclassified as Anger. This is because if none of our 6 trees detect an emotion in an image, then we classify it as *Anger*.

Moreover, the Facial Action Coding System in the specification has more rules that map to *Anger* than to other emotions. This suggests that there are more ways for a face to express anger than other emotions, so images are likely to be classified as *Anger*. However, this explanation isn't satisfactory since *Fear* also has many rules, but not many images are misclassified as such.

We could have created another label "No emotion" for images that cannot be classified as any of the 6 emotions. However the specification requires our testTrees(...) function generate predictions in $\{1, \cdots, 6\}$.

Miscellaneous

Noisy-Clean Datasets Question

Not surprisingly, the noisy dataset has lower overall performance. Its classification rate is 0.639 compared to 0.758 for clean data. The conclusion is the same when we analyse each emotion separately: precision, recall and F_1 score are lower for noisy data.

Noise degrades the classification accuracy of any machine learning model because it won't be learning the true underlying function that maps features to the target variable. For decision trees, this means taking many wrong branches to fit the noise.

Another intuition is that as noise in the data increases, the features that predict each emotion will become random, and so a decision tree will effectively be guessing randomly.

Among the 6 emotions, *Anger* is most susceptible to noise because a fair number of noisy test examples will not be classified as having any of the 6 emotions, so it is labelled as *Anger* by default. As a result, its F_1 score is 0.29 only.

Ambiguity Question

In case our 6 trees predict that a given image depicts more than one emotion, we considered several methods of selecting only one emotion:

1. Select the first detected emotion in alphabetical order
Examples: If Fear and Surprise are detected, then we predict Fear. If no emotion is detected, then we predict Anger.

2. Select by the depth of the tree

If selected by the shorter tree:
If selected by the larger tree:

3. Disable each active unit in turn, and take a majority vote
In this method, for example, if one collected data which contains 45 features in this project, and its actual result is one emotion. It can be said that if we remove one of the features (just as hiding part of the face)then the data should also be concluded to the actual emotion. In order to avoid removing the most important feature for the emotion, this method will remove one feather each time and find the most possible emotion. Example:

Pruning Question

The function $pruning\_example(x, y)$ takes two inputs: x is an $n \times p$ matrix of features and y is a n-dimensional vector of target variables.

It builds a classification tree and returns a figure with two curves, showing how classification cost changes with tree size (measured by number of leaves instead of depth). The figure also marks the smallest tree size whose cost is within 1 standard error of the minimum cost.

The two curves correspond to two methods of computing classification costs.

- Resubstitution: Tests a tree using the same data that was used to fit the tree (i.e. computes in-sample error)
- 10-fold cross validation: See our explanation in the Implementation section earlier. Validation error is an out-of-sample error if we don't use it to tune a hyperparameter such as maximum depth, otherwise it's only an estimate of out-of-sample error.

In figures 1 and 2, we find that classification costs have similar behaviour on both clean and noisy data. Resubstitution error keeps falling as the tree gets larger, while validation error falls initially and then rises.

This is expected because a deeper tree will always fit a dataset better than a shallower tree. However if a tree is too deep, it will overfit the training data but fail to generalize to unseen data, which is manifested in the increasing validation error.

Differences between clean and noisy data:

- For a fixed tree size, resubstitution and validation errors are higher on noisy data than clean data, since noise by definition makes it harder for a tree to fit the data.
- For the same reason, noisy data require more branches, and hence more leaves, to classify all training examples. It took about 275 vs 200 leaves to achieve zero in-sample error.
- The gap between resubstitution and validation errors rises faster for noisy data than clean data, which shows that overfitting is more severe when the data have noise.

Optimal tree size for clean data is about 22 leaves while that for noisy data is 25 leaves.

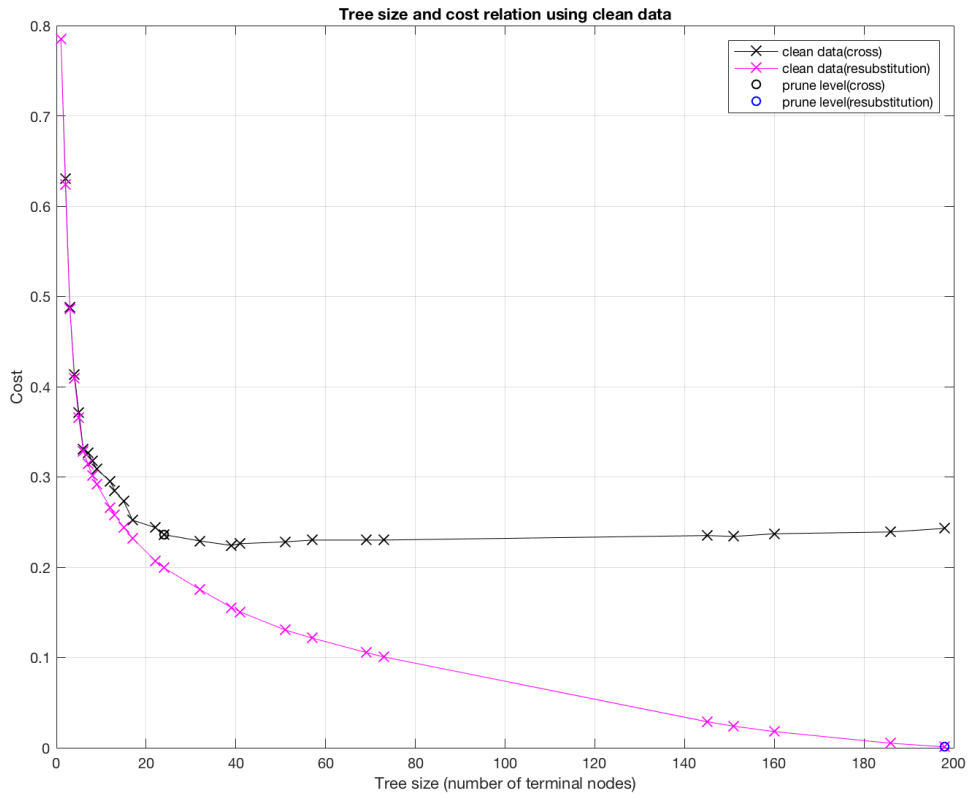


Figure 1: Clean Data

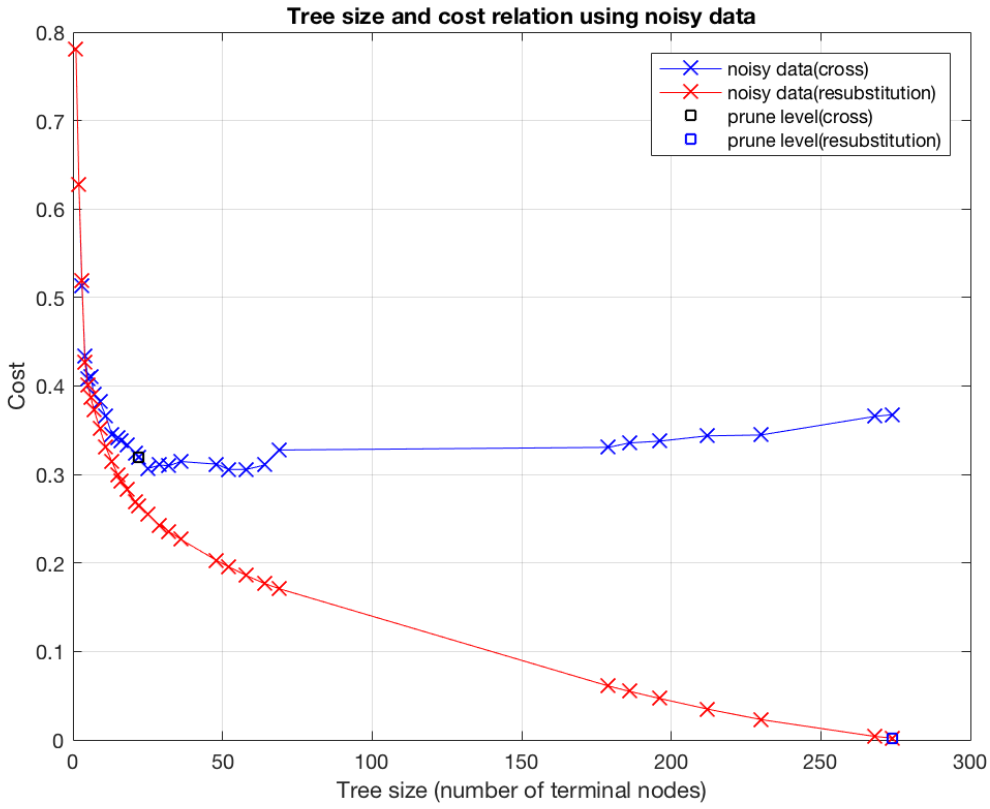


Figure 2: Noisy Data

Tree Diagrams

Figure 3 to 8 are the trees trained on the entire clean dataset (1004 examples).

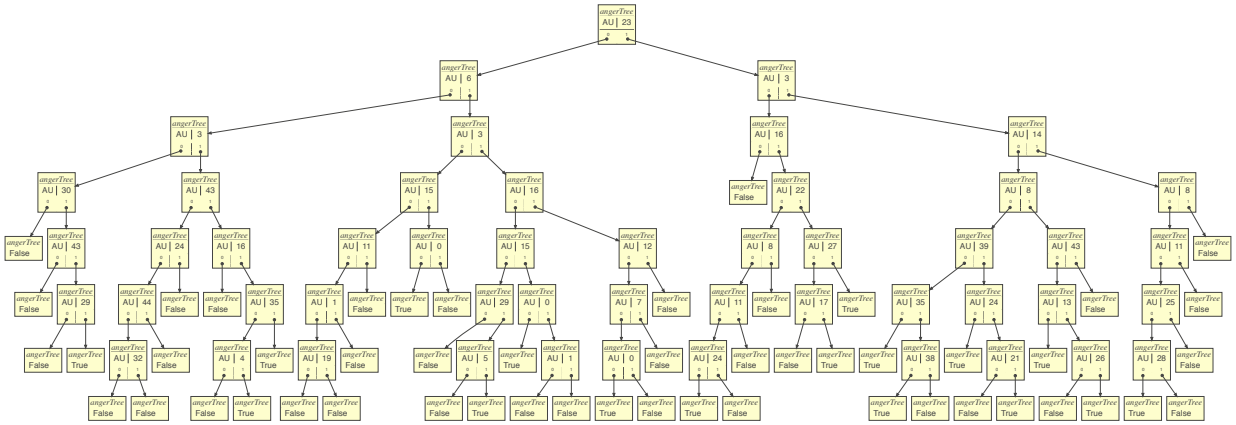


Figure 3: Anger Tree

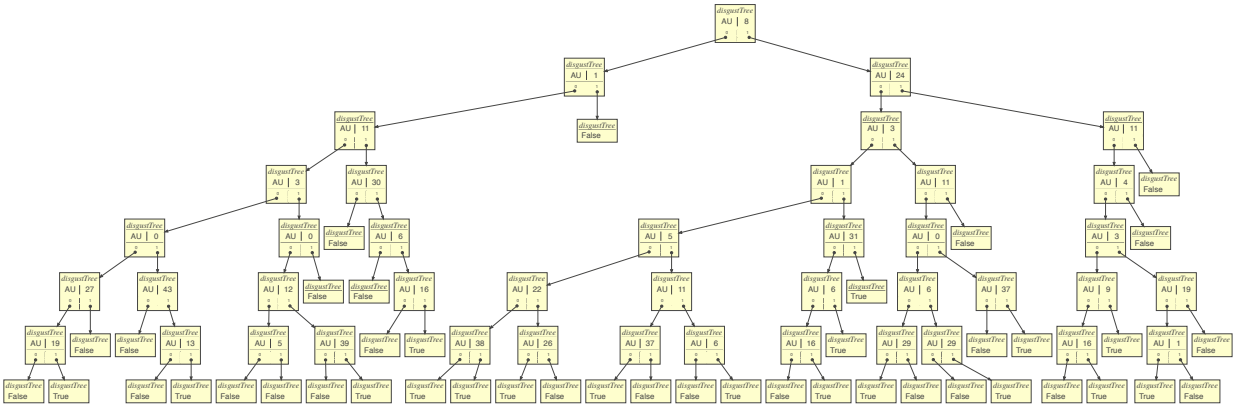


Figure 4: Disgust Tree

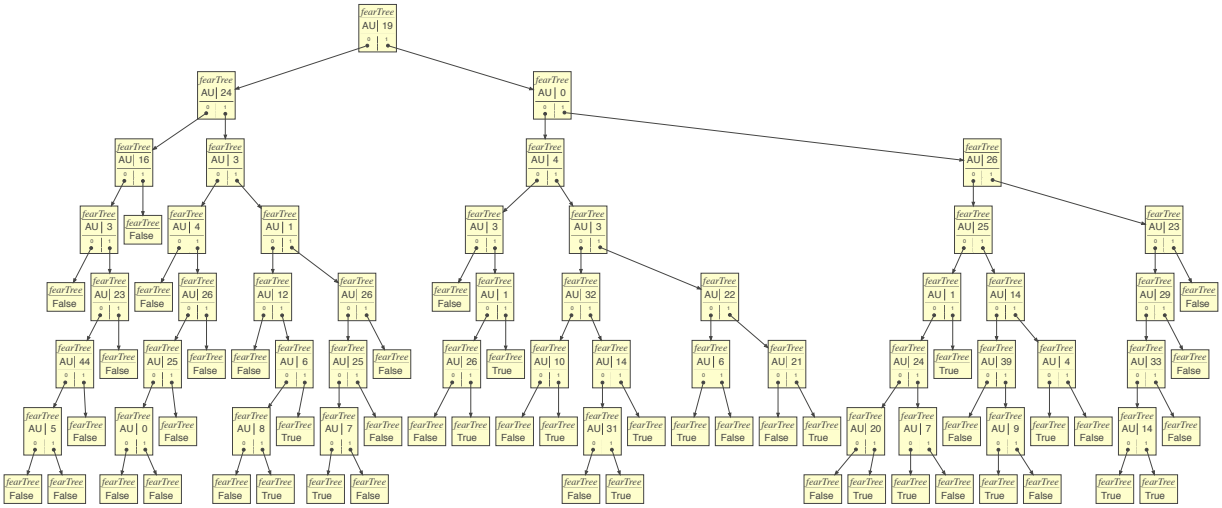


Figure 5: Fear Tree

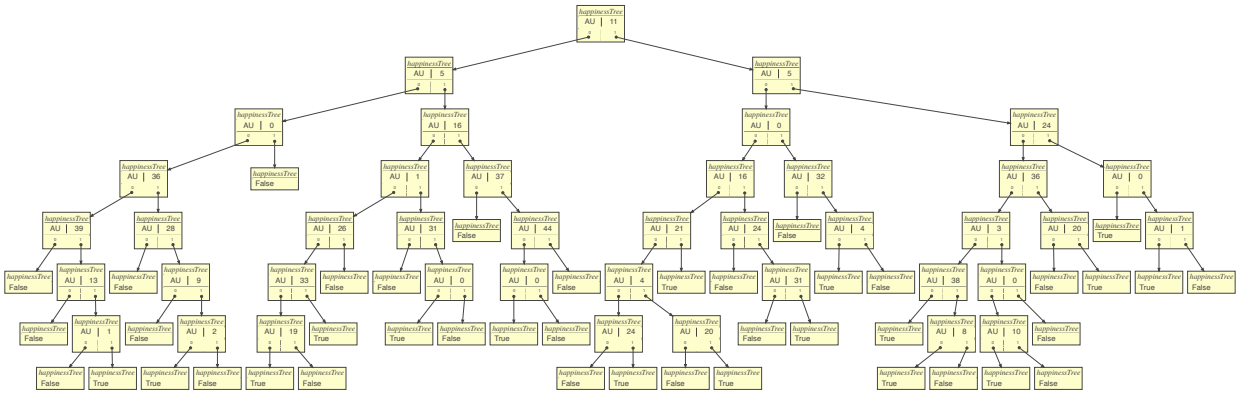


Figure 6: Happiness Tree

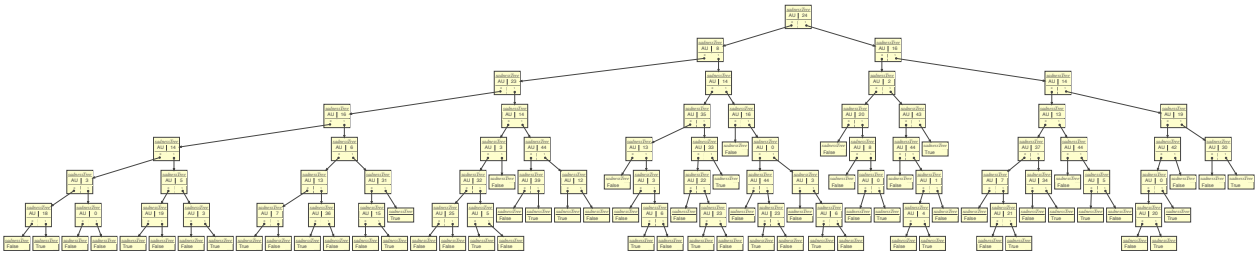


Figure 7: Sadness Tree

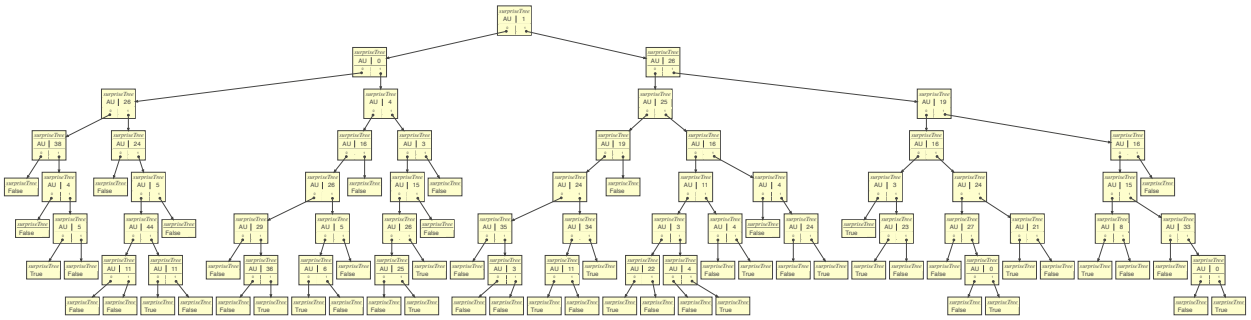


Figure 8: Surprise Tree