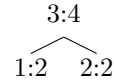


1. Decision Trees and ID3

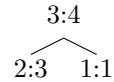
- (a) The result of splitting on A :



The associated remainder (weighted entropy):

$$-\left(\frac{1}{7} \ln \frac{1}{3} + \frac{2}{7} \ln \frac{2}{3} + \frac{2}{7} \ln \frac{2}{4} + \frac{2}{7} \ln \frac{2}{4}\right) \approx .669$$

And for splitting on B :



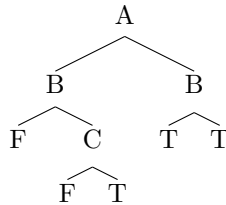
$$-\left(\frac{2}{7} \ln \frac{2}{5} + \frac{3}{7} \ln \frac{3}{5} + \frac{1}{7} \ln \frac{1}{2} + \frac{1}{7} \ln \frac{1}{2}\right) \approx .679$$

So splitting on A provides a result with a slightly lower remainder, and hence slightly higher information gain.

Splitting on A may be preferable because the entropy, i.e. information required after the split, is lower. Intuitively, splitting on A might be more useful because it provides a more even separation of the data into the true and false branches, and hence a shorter tree; splitting on B might also be useful because if B is true, then that branch is completely decided.

This shows that ID3 has inductive bias for shorter (more balanced) trees.

- (b) In the following tree, going down a left branch left indicates True, and right indicates False.



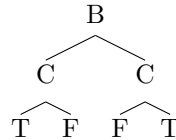
In the first stage, splitting on A generates the lowest remainder. Then considering the other three attributes, if we have A as True, then attributes B and C are tied for lowest remainder; if A is False, again B and C are tied for lowest remainder. We split on B in both cases.

If A is True and B is True, there is only one case, in which the classification is False. Otherwise, there are two cases that are exactly dependent on C .

If A is False and B is True, there is one case, in which the classification is True. Otherwise, there are two cases that cannot be separated, in which case we arbitrarily pick True.

6 of 7 are correctly identified by the tree.

- (c) Consider the following tree, that also correctly identifies 6 out of 7. Again, going left indicates True, and going right indicates False.



We see that although the ID3 algorithm is designed to greedily try to get the shortest / simplest tree by maximizing information gain at every split, it does not always end up choosing the shortest or simplest tree, probably due to the myopic nature of the greedy algorithm.

2. ID3 with Pruning

- (c) TODO: see resulting graphs.
- (d) i. TODO: how implementation makes use of instance weight in splitting decisions Our implementation uses instance weight throughout. It is implemented already to calculate entropy of a split, find the majority label in a branch, evaluate correct / incorrect. Then it uses this information ***

AdaBoost on ID3 would not work if splitting decisions were made by counting instances instead of summing weights, because AdaBoost is based on the idea that you update weights based on which data instances need better fitting.

- ii. Weighted entropy of $\{x_1, \dots, x_n\}$ where $y_1 = T$, $y_i = F$ else, $w_1 = 0.5$, and other weights are $0.5/(n-1)$: by definition $entropy(D) = -\sum_{y \in Y} \frac{W_y}{W} \log_2 \frac{W_y}{W}$ and $W_y = \sum_i w_k(i)I(y_i = y)$; we have $W_T = 0.5$, $W_F = 0.5(n-1)^{-1}(n-1) = 0.5$, $W = 1$, and so

$$entropy(D) = -\left[\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right] = 1$$

- i.
- ii.
- iii.
- iv. More rounds of boosting training (generally) increases accuracy in cross-validated test sets. It appears that sometimes an extra iteration of training decreases accuracy a bit, but the trend is overall positive until the plateau, although the variation in accuracy can be pretty large (i.e. boosting 4 times is the same as boosting 10 times, though the peaks in between go up). TODO: add some reasons why this is true

3.