1 Extending Decision Trees

- Real-valued features
- Non-binary class variable
- Noise and overfitting

1.1 Non-binary class variable

So far, the class variable is binary (Tennis is Yes or No). What if there are more than two classes? Suppose class is in $c_1, \ldots, c_l$.

The modified ID3 algorithm:

Algorithm 1 ID3 Algorithm (Features, Examples)

```
1: If all examples belong to the same class $i$, return a leaf node with decision $i$.
2: If no features left, return a leaf node with the majority decision of the examples.
3: If no examples left, return a leaf node with the majority decision of the examples in the parent node.
4: else
5:     choose feature $f$ with the maximum information gain
6:     for each value $v$ of feature $f$ do
7:         add arc with label $v$
8:         add subtree $ID3(F - f, s \in S | f(s) = v)$
9:     end for
```

Calculation of information gain:

Consider feature A with c_i examples in class i, $i = 1, \ldots, l$. For $j = 1, \ldots, k$, if A takes value v_j, then there are c_i^j examples in class i.

$$
\begin{aligned}
& Gain(A) && (1) \\
&= I\left(\frac{c_1}{c_1 + \cdots + c_l}, \ldots, \frac{c_l}{c_1 + \cdots + c_l}\right) && (2) \\
&\quad - \sum_{j=1}^{k} \frac{c_1^j + \cdots + c_l^j}{c_1^j + \cdots + c_l^j} I\left(\frac{c_1^j}{c_1^j + \cdots + c_l^j}, \ldots, \frac{c_l^j}{c_1^j + \cdots + c_l^j}\right) && (3)
\end{aligned}
$$

(CQ)

Suppose that we are classifying examples into three classes. Before testing feature X, there are 3 examples in class c_1, 5 examples in class c_2, and 2 examples in class c_3. Feature X has two values a and b. When $X = a$, there are 1 examples in class c_1, 5 examples in class c_2, and 0 examples in class c_3. When $X = b$, there are 2 examples in class c_1, 0 examples in class c_2, and 2 examples in class c_3.

What is the information gain for testing feature X at this node?

$I(3/10, 5/10, 2/10) = 1.485$

$6/10 * I(1/6, 5/6, 0/6) + 4/10 * I(2/4, 0/4, 2/4) = 6/10 * 0.65 + 4/10 * 1 = 0.79$

Information gain is $1.485 - 0.79 = 0.695$.

1.2 Real-valued features

Real-world problems often have real-valued features.

For example, Jeeves could have recorded the temperature as a real-valued feature.

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	29.4	High	Weak	No
2	Sunny	26.6	High	Strong	No
3	Overcast	28.3	High	Weak	Yes
4	Rain	21.1	High	Weak	Yes
5	Rain	20.0	Normal	Weak	Yes
6	Rain	18.3	Normal	Strong	No
7	Overcast	17.7	Normal	Strong	Yes
8	Sunny	22.2	High	Weak	No
9	Sunny	20.6	Normal	Weak	Yes
10	Rain	23.9	Normal	Weak	Yes
11	Sunny	23.9	Normal	Strong	Yes
12	Overcast	22.2	High	Strong	Yes
13	Overcast	27.2	Normal	Weak	Yes
14	Rain	21.7	High	Strong	No

How should we deal with real-valued features?

Solution 1: Discretize the values.

- Temp < 20.8 –> Cool

- $20.8 \leq$ Temp < 25.0 –> Mild
- $25.0 \leq$ Temp –> Hot

Advantage is that this is easy to do. Disadvantage is that we lose valuation information. What if the discretization we pick makes the decision tree much more complicated?

Solution 2: Dynamically choose a split point c for a real-valued feature

Split a feature f into $f < c$ and $f \geq c$.

How should we choose the split point c?

1. Sort the instances according to the real-valued feature
2. Possible split points are the values that are midway between two values that differ in their classification.
   (a) Suppose that the feature changes its value from X and Y. Follow the steps below to determine whether we should consider $(X + Y)/2$ as a possible split point.
   (b) For all the data points where the feature takes the value X, gather all the labels into the set L_X.
   For all the data points where the feature takes the value Y, gather all the labels into the set L_Y.
   (c) If there exists a label $a \in L_X$ and a label $b \in L_Y$ such that $a \neq b$, then we will consider $(X + Y)/2$ as a possible split point.
3. Determine the information gain for each possible split point and choose the split point with the largest gain.

First, let's sort the data set according to the values of Temp. The sorted data set is below.

Day	Outlook	Temp	Humidity	Wind	Tennis?
7	Overcast	17.7	Normal	Strong	Yes
6	Rain	18.3	Normal	Strong	No
5	Rain	20.0	Normal	Weak	Yes
9	Sunny	20.6	Normal	Weak	Yes
4	Rain	21.1	High	Weak	Yes
14	Rain	21.7	High	Strong	No
8	Sunny	22.2	High	Weak	No
12	Overcast	22.2	High	Strong	Yes
10	Rain	23.9	Normal	Weak	Yes
11	Sunny	23.9	Normal	Strong	Yes
2	Sunny	26.6	High	Strong	No
13	Overcast	27.2	Normal	Weak	Yes
3	Overcast	28.3	High	Weak	Yes
1	Sunny	29.4	High	Weak	No

Whenever the value of Temp changes in its sorted order, we have a possible split point. For example, here are a few possible split points.

- (17.7 + 18.3) / 2 = 18
- (22.2 + 23.9) / 2 = 23.05

We could choose to test all such midway values as the split points. However, the procedure described above tries to go through fewer split point values by only choosing split points for which the two values differ in their classification. Here are a few examples:

- The classification for 17.7 is Yes, whereas the classification for 18.3 is No. Thus, we will consider (17.7 + 18.3) / 2 = 18 as a possible split point.
- The classification for 20.6 and the classification for 21.1 are both Yes. Therefore, we will NOT consider the midway value between these two as a possible split point.
- The classification for 21.7 is No, whereas the classification for the 2 data points with 22.2 are Yes, and No. In this case, we will consider (21.7 + 22.2) / 2 = 21.95 as a possible split point (because No for 21.7 is different from Yes for 22.2.)
- The classification for 2 data points with 22.2 are No and Yes, whereas the classification for the two data points with 23.9 are both Yes. We will consider (22.2 + 23.9) / 2 = 23.05 as a possible split point (because No for 22.2 is different from Yes for 23.9.)

Following the procedure described above, you should derive 8 possible split points.

Here is an example for calculating the information gain for the split point $c = \frac{21.7+22.2}{2} = 21.95$.

- Temp: +: 3, 4, 5, 7, 9, 10, 11, 12, 13. -: 1, 2, 6, 8, 14.
- Temp < 21.95: +: 4, 5, 7, 9. -: 6, 14.
- Temp ≥ 21.95: +: 3, 10, 11, 12, 13. -: 1, 2, 8.

Gain (Temp < 21.95) = $I(9/14, 5/14) - [6/14\, I(4/6, 2/6) + 8/14\, I(5/8, 3/8)] = 0.00134$

Repeat this for all real-valued features and all split points.

Additional complication: On any path from the root to a leaf

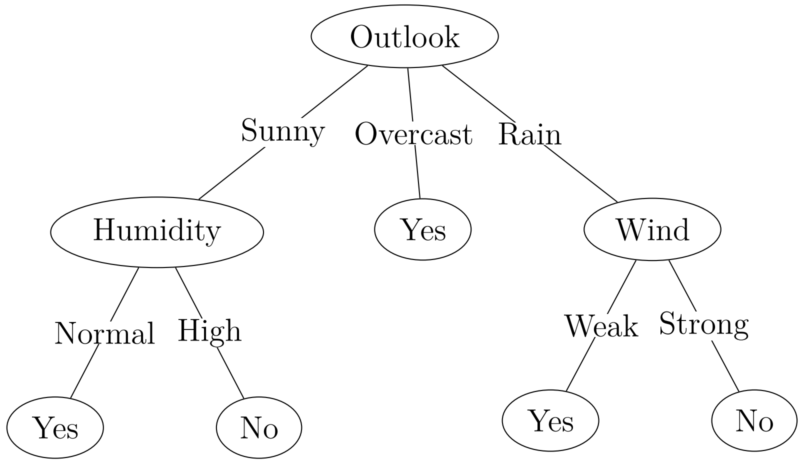
- Any discrete feature is tested at most once.
- Any real-valued feature can be tested many times.

This means that we will have larger trees and the trees may be difficult to understand.

1.3 Noise and overfitting

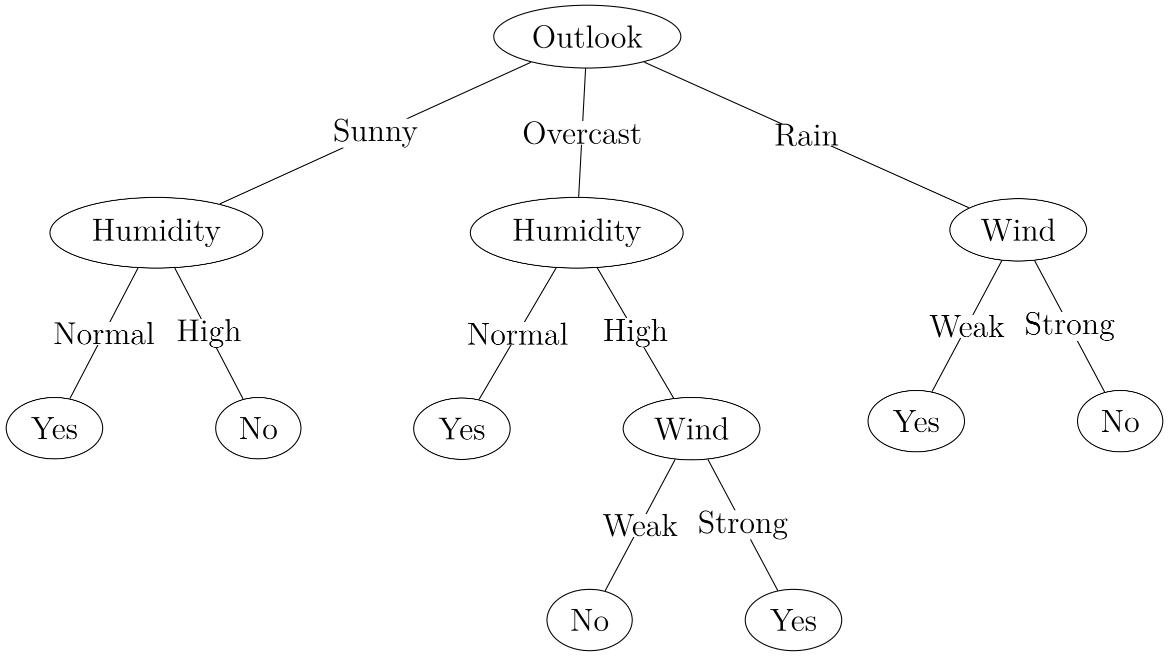
Training examples may be misclassified. For example, suppose that the class of Day 3 is corrupted to No.

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



We need to expand the sub-tree under Outlook = Overcast.

Result: The tree becomes more complicated just to handle one outlier. –> Overfitting.



What are the test errors for these two trees?

- With subtree replaced by a leaf node with Yes: 0 errors. (Recall that the first tree classifies all examples in the test set perfectly.)
- With subtree: 2 errors – This is overfitting!

Problem: The ID3 algorithm is a perfectionist. It grows the tree until the tree perfectly classifies all the training examples.

When there is noise in the data, the ID3 algorithm works really hard to capture all the noise. Over-fitting occurs.

When the number of examples at a leaf is too small, the ID3 algorithm works diligently to capture all of the classification even if it's not worth it. Over-fitting occurs again.

Solutions: We want to force the tree to be simple.

1. Grow the tree to a pre-specified maximum depth.
2. Enforce a minimum number of examples at a leaf node.
3. Post-prune the tree using a validation set. (Most successful in practice.)

For all the solutions, we will have leaf nodes that are a mixture of positive and negative examples. How should we make a decision at such a node?

- Label leaf with the majority class.
- Label leaf with a probability for each class proportional to the number of examples in the class. $\frac{p}{p+n}$ and $\frac{n}{p+n}$.

Suppose that we want to grow the tree to a maximum depth. What maximum depth should we choose? This is a parameter of our model and we need to use a validation set to choose this parameter.

Process for solution 1 (growing a tree to a maximum depth)

- Randomly split the entire data into a training set and a validation set. (For example, 2/3 is the training set and 1/3 is the validation set.)
- For each pre-specified maximum depth, generate a tree with the maximum depth on the training set.
- Calculate the prediction accuracy of the generated tree on the validation set.
- Choose the maximum depth which results in the tree with the highest prediction accuracy.

This fails to use all the available data for training. If the training set is too small, we may get a poor hypothesis. If the validation set is too small, we may get a poor estimate of the prediction accuracy.

Could we use the data more efficiently?

Answer: use k-fold cross-validation. Each example serves double duty—as training data and validation data. Let $k = 10$.

1. For each pre-specified maximum depth, do steps 2 to 6.
2. First we split the data into 10 equal subsets.
3. Perform 10 rounds of learning.
4. On each round, 1/10 of the data is held out as a validation set and the remaining examples are used as training data. (Draw a picture of this.)
5. Over the 10 rounds, we generate 10 different trees and determine their prediction accuracies on 10 different validation sets.
6. Calculate the average prediction accuracy on the validation sets.
7. Choose the maximum depth that results in the highest prediction accuracy.