CS 217: Artificial Intelligence and Machine Learning **Jan-Apr 2024**

Lecture 10: Decision Trees

Lecturer: Swaprava Nath *Scribe(s): SG19 & SG20*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

10.1 Decision Trees

Following from the previous lecture discussion, we'll continue with some more examples of Decision Trees.

Example 1: Exam Results

Exam Result	Online Course Taken	Background	Mock Test
P	Y	Maths	N
F	N	Maths	Y
F	Y	Maths	Y
P	Y	CSE	N
F	N	Other	Y
F	Y	Other	Y
P	Y	Maths	N
P	Y	CSE	N
P	N	Maths	Y
P	N	CSE	Y
P	Y	CSE	Y
P	N	Maths	N
F	Y	Other	Y
F	N	Other	N
P	Y	Maths	Y

Goal: Create a classifier to determine whether a given student passes or not based on the data.

We'll use a trial-and-error approach[1]:

Try 1: Using only Mock Test

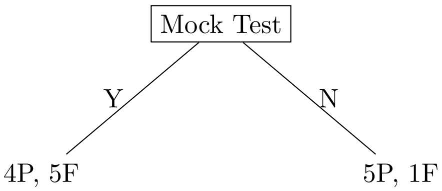

Figure 10.1: Decision tree for Try 1

[1] Note that P/F numbers in decision trees are slightly different from what done in class, corrected according to original dataset

We need to make more divisions/classifications based on other features further for Figure : 10.1 decision tree.

Try 2:

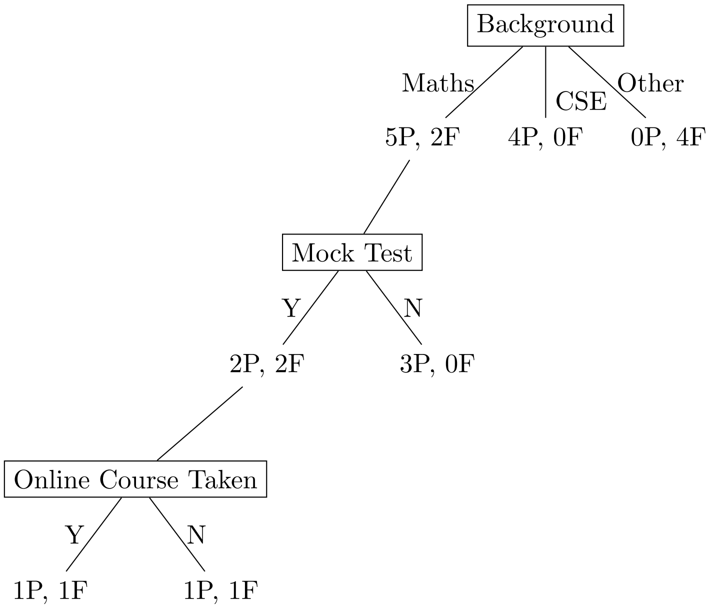

Figure 10.2: Decision tree for Try 2

No further divisions are possible in Figure 10.2 decision tree. We'll stop here as there are no more features left for classification. Not much useful information can be gained at the last step of Figure 10.2 decision tree.

# At the first stage of classification, Try 2 seems to be a better decision tree because in it, the later two divisions are certain (CSE → Pass, Others → Fail).

Question: How can we make such a splitting scheme more systematic?

Answer: Naive approach - check for all possible features one by one in every possible order. However, this can lead to an exponential increase in size[2]. We need to look for a better systematic approach.

Example 2: Iris Dataset

Below is a plot of petal-width vs. petal-length for several classes in the IRIS dataset. We can see that some boundaries can be marked in order to separate out the classes. We also see the associated Decision Tree structure in Figure 10.3.

Again, we are faced with the following questions:

1. *How to build the tree systematically?*
2. *Where do we exactly stop building a tree?*

[2] The increase in size mentioned here refers to the increase in the number of combinations or configurations that the algorithm needs to consider, which can become impractical or impossible to explore exhaustively.

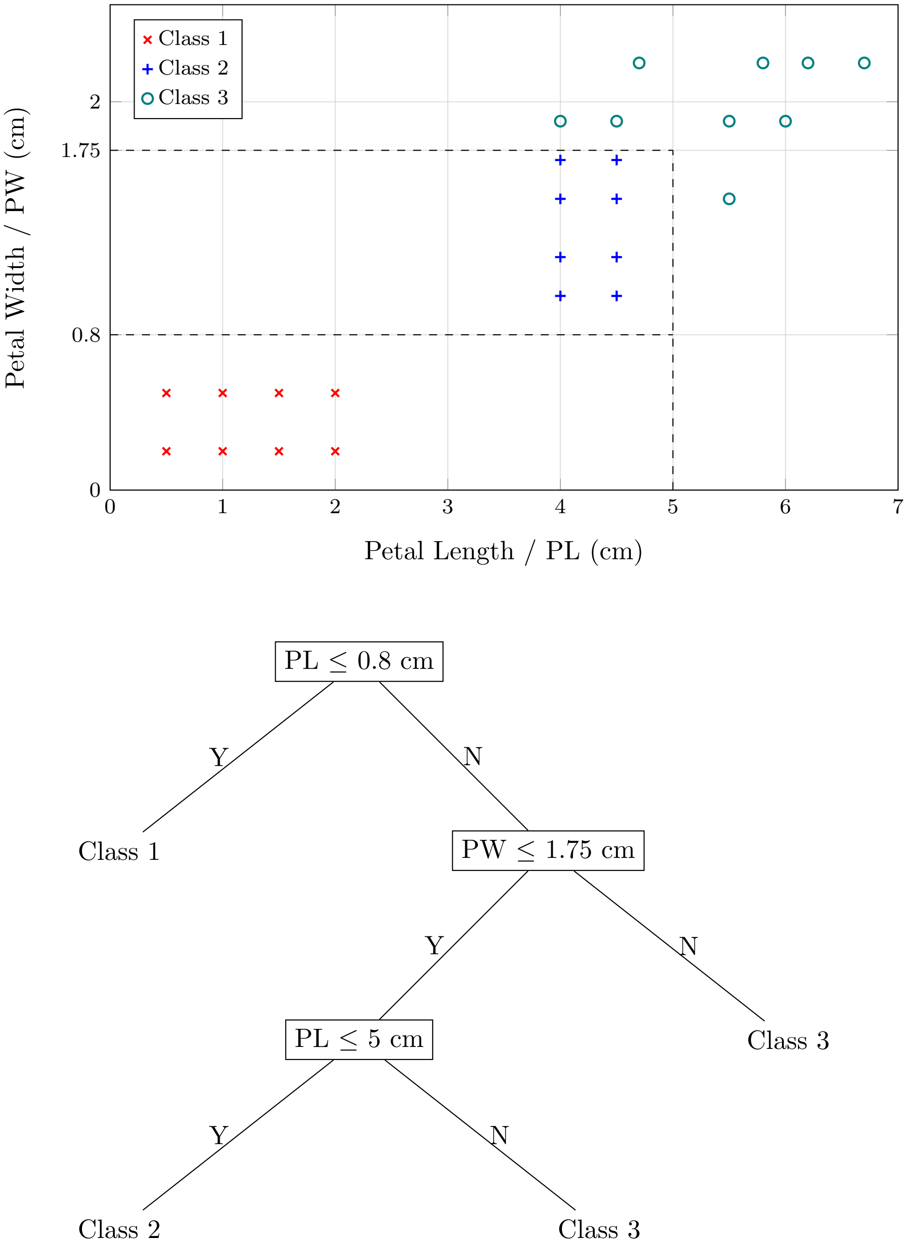

Figure 10.3: Iris Dataset Decision Tree

Answering these will help create an automated algorithm to build a Decision Tree given the dataset. Let's look at another example now.

Example 3: Boolean Truth Table

We show a dummy dataset in Table 10.1 and the associated Decision Tree in 10.4.

If we divide the above data wrt X_1 or X_2 what can we say about the classification and with what certainty? To answer this question we will introduce entropy.

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Table 10.1: A dummy Boolean Truth-Table dataset

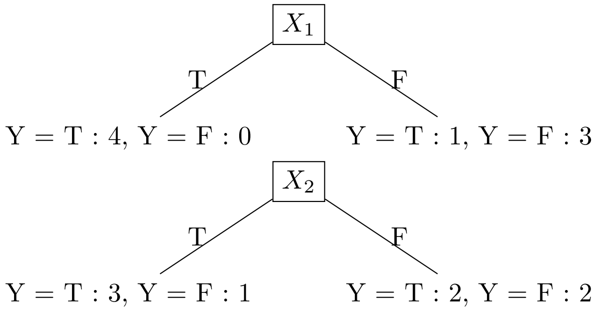

Figure 10.4: Decision tree with root node X_1 and X_2

10.2 Entropy

Entropy is the measurement of the randomness of a Random Variable.
Let X be a categorical Random Variable with a state-space $\mathcal{X}$, then $\forall x \in \mathcal{X}$, We have $p(x) = P(X = x)$
We define,

$$H(X) = -\sum_{x \in X} p(x) \cdot \log_{|\mathcal{X}|} p(x)$$

Here $|\mathcal{X}|$ is the number of elements in X. For a Binary random variable, its value will be 2.
We can observe that H(X) can be seen as negative of $\mathbb{E}\left[\log_{|X|}(p(x))\right]$.

We can show that $H(X)$ lies in between 0 and 1 both inclusive, i.e., $H(X) \geq 0$ and $H(X) \leq 1$. The first one is obvious as $0 \leq p(x) \leq 1$. To prove the second, we will use Jensen's Inequality.
We know that `log` is a concave function. So,

$$\mathbb{E}[\log(X_1)] \leq \log(\mathbb{E}[X_1])$$

If we replace random variable X_1 with another random variable $1/p(X)$, LHS will become $H(X)$ for `log` to the base $|X| = n$, So

$$\begin{aligned}
\mathbb{E}\left[\log_n(1/p(X))\right] &\leq \log_n\left(\mathbb{E}(1/p(X))\right)\\
\mathbb{E}\left[-\log_n(p(X))\right] &\leq \log_n\left[\sum p(x) \cdot 1/p(x)\right]\\
-\mathbb{E}\left[\log_n(p(X))\right] &\leq \log_n(n) = 1\\
H(X) &\leq 1
\end{aligned}$$

10.2.1 Observations of Entropy Function

Consolidating our discussion from above, we note two important bounds on Entropy, as follows:

1. $H(X) \geq 0$
   Since $0 \leq p(x) \leq 1$, the `log` part of each term is always negative, hence entropy is always positive. The equality is attained for random variables which are certain, so probability for each category except the one which is certain (say x^*) is 0, so all those terms cancel out (assuming $0 \cdot \log 0 = 0$) and $\log p(x^*) = 0$. Hence total entropy is 0.

2. $H(X) \leq 1$
   This can be proved using Jensen's inequality which states that, for any function $f(x)$ which is convex in R_X, and $\mathbb{E}[f(X)]$ and $f(\mathbb{E}[X])$ are finite, then

   $$\mathbb{E}[f(X)] = f(\mathbb{E}[X])$$

   Notice that $H(X)$ is a concave function, so the inequality just reverses.
   Proving this is left as an exercise to the reader.

10.2.2 Conditional Entropy

Conditional Entropy denotes the quantity of information needed to describe one random variable X, when another random variable Y is already observed.

Formally, it's defined as:

$$H(X|Y) = -\sum_y \sum_x p(x,y) \cdot logp(x|y)$$

where, $p(x|y) = P(X = x|Y = y)$

We can rewrite the conditional entropy expression as –

$$H(X|Y) = -\sum_y \sum_x p(x,y) \cdot \log(p(x|y))$$
$$H(X|Y) = \sum_y p(y) \cdot \left(-\sum_x p(x|y) \cdot \log(p(x|y)) \right)$$
$$H(X|Y) = \sum_y p(y) \cdot H(X|Y = y)$$

Some observations:

- If $X \perp\!\!\!\perp Y$ (notation for denoting X and Y are independent random variables), then $H(X|Y)$ is just $H(X)$ which is intuitive because knowing about Y doesn't provide us any information about X.
- For a specific y,
  $$H(X|Y = y) = -\sum_x p(x|y) \cdot \log p(x|y)$$

10.2.3 Mutual Information

It's the measure of how much information is gained by observing X given that you've already observed Y. Formally, it's written as –

$$\mathrm{I}(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

We can prove second equality by Bayes Theorem as follows

$$\begin{aligned}
H(X) - H(X|Y) &= \sum_x P(X=x) \cdot \log(1/P(X=x)) \\
&\quad - \sum_y \sum_x P(X=x, Y=y) \cdot \log(1/P(X=x|Y=y)) \\
&= \sum_x (\sum_y P(X=x, Y=y) \cdot \log(1/P(X=x))) \\
&\quad - \sum_x (\sum_y P(X=x, Y=y) \cdot \log(1/P(X=x|Y=y))) \\
&= \sum_x \sum_y P(X=x, Y=y)) \cdot \log(P(X=x|Y=y)/P(X=x)) \\
&= \sum_x \sum_y P(X=x, Y=y)) \cdot \log(P(Y=y|X=x)/P(Y=y)) \\
&= \sum_y (\sum_x P(Y=y, X=x) \cdot \log(1/P(Y=y))) \\
&\quad - \sum_y (\sum_x P(Y=y, X=x) \cdot \log(1/P(Y=y|X=x))) \\
&= H(Y) - H(Y|X)
\end{aligned}$$

Now getting back to Example 3,

We will calculate $I(Y, X_1)$ and $I(Y, X_2)$, which are equal to $H(Y) - H(Y|X_1)$ and $H(Y) - H(Y|X_2)$ respectively. From these two values whichever will be the maximum will give us the feature which will be used to split the data. For that, we have to find the feature that gives us minimum conditional entropy,

$$\begin{aligned}
H(Y|X_1) &= \sum_{(x_1 \in \{T,F\})} p(X_1 = x_1) \cdot H(Y|X_1 = x_1) \\
&= p(X_1 = T) \cdot H(Y|X_1 = T) + p(X_1 = F) \cdot H(Y|X_1 = F) \\
p(y|X_1 = T) &= P(Y = y|X_1 = T) = \{1 \text{ if } y = T, \quad 0 \text{ if } y = F\} \\
H(Y|X_1 = T) &= 0 \\
p(y|X_1 = F) &= P(Y = y|X_1 = F) = \{1/4 \text{ if } y = T, \quad 3/4 \text{ if } y = F\} \\
H(Y|X_1) &= -1/2 \cdot (1/4 * \log_2 1/4 + 3/4 * \log_2 3/4) \\
H(Y|X_1) &= 0.4056
\end{aligned}$$

Similarly,

$$H(Y|X_2) = 0.9056$$

10.3 Algorithm for decison tree building

Keep finding the feature that yields the maximum information gain (minimum conditional entropy) until stopping criteria is met.

Specifically,

1. Find features that yield maximum information gain (minimum conditional entropy).
2. Repeat until stopping criteria (recursive calls) not met.

Remark: **Gini index** can be used as another metric for building the decision tree.
Gini index: probability for a random instance being misclassified when chosen randomly

Where to stop?

- **Base case 1:** (Figure 10.5) Reaching nodes with atomic distributions i.e., $H(Y|node) = 0$.
  After reaching a node, if there is no randomness and we know that all data points reaching this node belong to a unique class then it makes sense to stop there as the data has been classified.
- **Base case 2:** (Example 4 below) Information gain is same for all remaining variables.
  It is not always a good base case as it can lead to no splitting at all. This is illustrated in example 4.

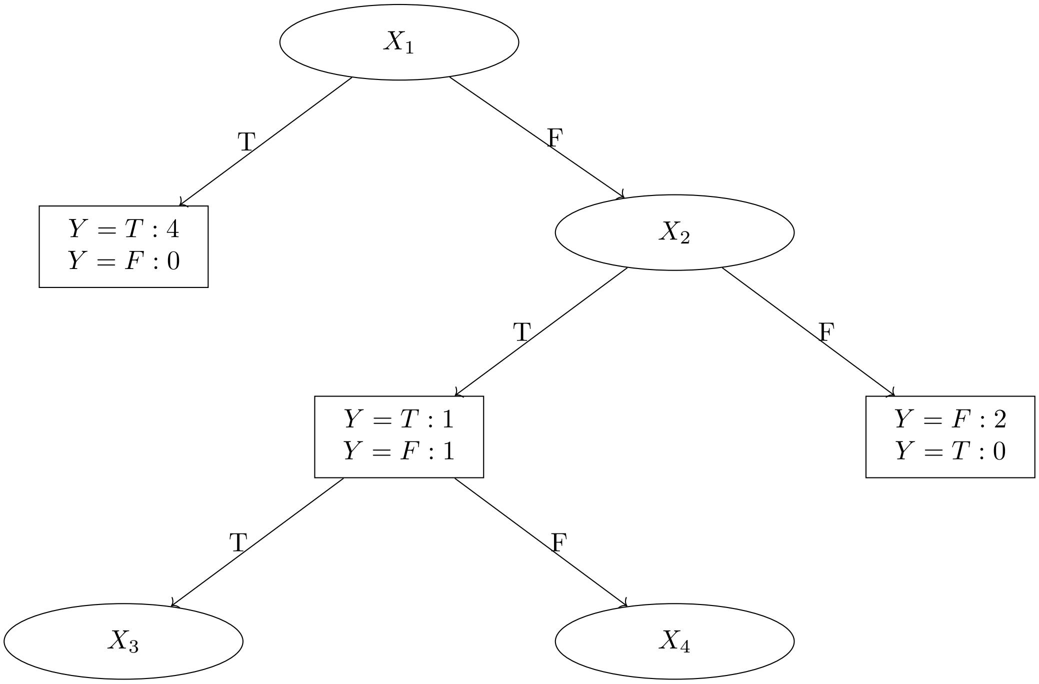

Figure 10.5: Decision Tree

The algorithm should stop when conditional entropy is zero at the point, i.e., the node with atomic distributions, $H(Y|\text{node}) = 0$.

Example 4:

Z_1	Z_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$H(Y) = 1$

$H(Y|Z_1) = 1 = H(Y|Z_2)$

According to Base Case 2, this should not be split.

10.3.1 Overfitting in Decision Trees

Shallow decision trees lack the power to effectively distinguish between different classes or patterns in the data due to their limited depth and simplicity.
On the other hand, deep decision trees can become overly specific to the training examples, capturing noise or outliers that may not generalize well to new data. This is called overfitting.

Dealing with overfitting

- **Pre-pruning or Early stopping:** We use a validation set to check how well the model works. The depth of the model is expanded until the validation set error starts to rise, indicating potential overfitting. Further depth increase is then halted to prevent overfitting. Figure 10.6 shows a way to do it. Model Complexity refers to the depth of Decision Tree in this context.

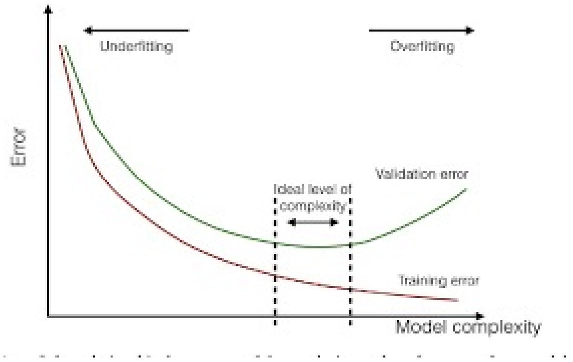

Figure 10.6: Pre-pruning

- **Post-pruning:** We let the tree grow fully and then reduce some branches, reducing its depth to prevent overfitting.
- **Ensemble method:** It involves training multiple decision trees and then averaging their outputs to produce a final prediction.