10-315 Notes
Decision Trees

Carnegie Mellon University
Machine Learning Department

Contents

1 Decision Trees **1**

1.1 Reminder: ML Training and Inference 1

1.2 Using Decision Trees 2

1.3 Evaluating Decision Trees 3

1.4 Building Decision Trees from Training Data 4

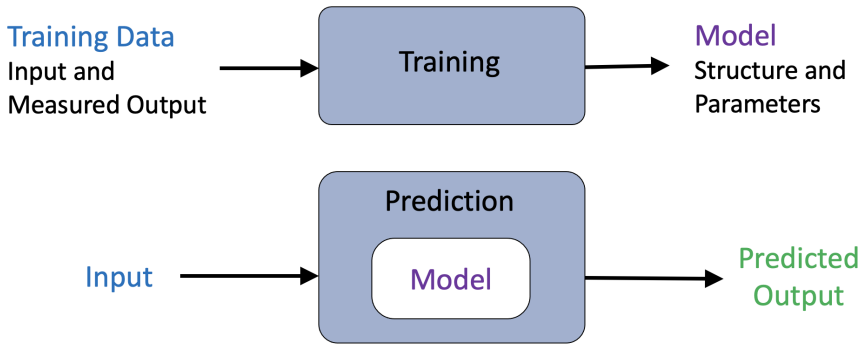
1.4.1 Decision Stumps and Majority Vote 4

1.4.2 From Stumps to Trees 4

1 Decision Trees

1.1 Reminder: ML Training and Inference

A simple definition of machine learning is using (training) data to learn a model that we'll later use for inference (predicting the output for a new input).



The **prediction** step relies on a **model** that maps **input** to **predicted output**. The **training** step uses pairs of input and **measured output** to construct the specific **structure** and **parameters** of the model.

When using machine learning decision trees, the **model** is a specific decision tree.

1.2 Using Decision Trees

A **decision tree** is model for a prediction function, also called a **hypothesis function**, $\hat{y} = h(\mathbf{x}$. It takes in a vector of features, also called attributes, $\mathbf{x}$, and follows the tree from root to a leaf based on the decisions at each node.

- Each non-leaf node uses one input feature (aka attribute), x_j, to determine which branch follow
- Nodes often split into two branches but can also split into multiple branches
- Leaf nodes return the predicted output value, $\hat{y}$

Example: Medical Classification

An overly simple decision tree and associated data to predict whether or not an expectant mother should have a cesarean section:

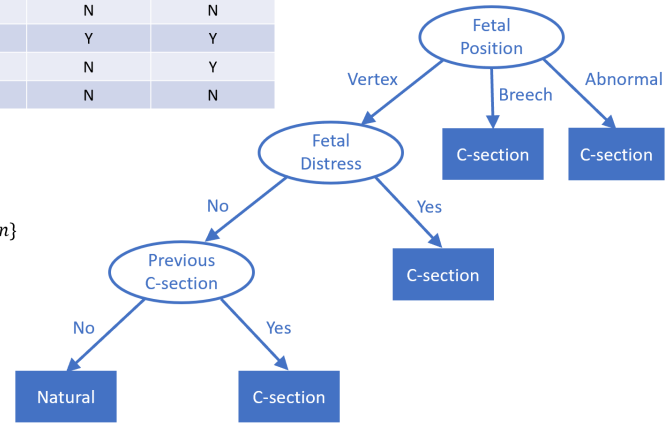
	y	x_1	x_2	x_3
i	Outcome	Fetal Position	Fetal Distress	Previous C-sec
1	Natural	Vertex	N	N
2	C-section	Breech	N	N
3	Natural	Vertex	Y	Y
4	C-section	Vertex	N	Y
5	Natural	Abnormal	N	N

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [x_1, x_2, x_3]^T$$

$x_1 \in \{Vertex, Breech, Abn\}$
$x_2 \in \{Y, N\}$
$x_3 \in \{Y, N\}$

$y \in \{C\text{-}section, Natural\}$

$\hat{y} = h(\mathbf{x})$



Example: Species Classification

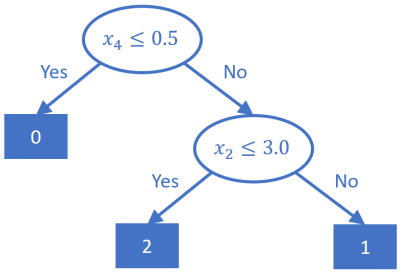
A decision tree to predict the species of an iris flower given the input feature consisting of four-petal/sepal measurements:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = [x_1, x_2, x_3, x_4]^\top$$

$\mathbf{x} \in \mathbb{R}^4$

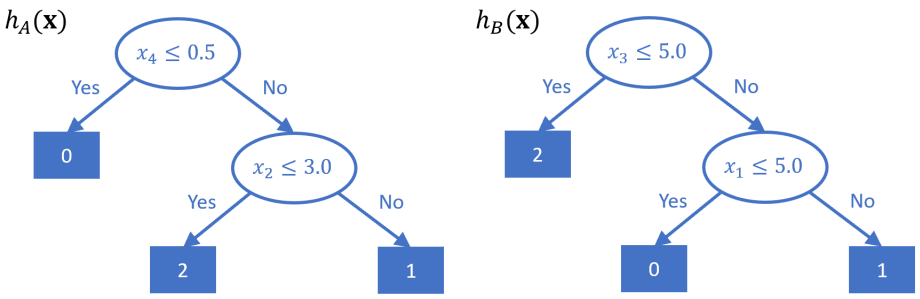
$y \in \{0, 1, 2\}$

$\hat{y} = h(\mathbf{x})$



1.3 Evaluating Decision Trees

Consider the following two decision trees for the Iris classification task. Which is better?



We can't tell which is better. We need data! And, we need a **performance measure**.

For the performance measure, we'll use **classification error rate**,

$$\frac{1}{N}\sum_i \mathbb{I}\left(y^{(i)} \neq \hat{y}^{(i)}\right)$$

And, we'll use the following table of labeled Iris data to evaluate our two decision tree models:

	y	x_1	x_2	x_3	x_4
i	Species	Sepal Length	Sepal Width	Petal Length	Petal Width
1	0	4.3	3.0	1.1	0.1
2	0	4.9	3.6	1.4	0.1
3	0	5.3	3.7	1.5	0.2
4	1	4.9	2.4	3.3	1.0
5	1	5.7	2.8	4.1	1.3
6	1	6.3	3.3	4.7	1.6
7	2	5.9	3.0	5.1	1.8

Now, we can see which Iris tree is better by passing each data point through the tree to determine its predicted output, $\hat{y}$. Then, comparing the predicted values to the output values from the dataset, we can compute the classification error rate for each decision tree model.

$\hat{y}_A$	$\mathbb{I}\left(y^{(i)} \neq \hat{y}_A{}^{(i)}\right)$
0	0
0	0
0	0
1	0
1	0
2	1
2	0

i	y
1	0
2	0
3	0
4	1
5	1
6	1
7	2

$\hat{y}_B$	$\mathbb{I}\left(y^{(i)} \neq \hat{y}_B{}^{(i)}\right)$
0	0
0	0
1	1
0	1
1	0
1	0
2	0

Model A (left) has an error rate of $1/7$ and model B (right) has an error rate of $2/7$, so model A performs better on this dataset.

1.4 Building Decision Trees from Training Data

1.4.1 Decision Stumps and Majority Vote

First, we'll start with a decision stump. A **decision stump** is a decision tree with only one split, i.e., one node that branches into leaf nodes.

Construction:

To construct a decision stump:

1. Select attribute and split criteria
2. Split training dataset into subsets by passing each training point through our decision node. Each leaf will then have a subset of the training dataset.
3. The leaves use a **majority vote** among the output values, $y^{(i)}$ for all data points in that leaf. Note: a tie-breaking scheme needs to be specified.

Majority vote: Given a leaf's subset of training data points, each with discrete label value $y^{(i)} \in \mathcal{Y}$, the majority vote is:

$$\hat{y} = \underset{k \in \mathcal{Y}}{argmax} \sum_i \mathbb{I}\left(y^{(i)} = k\right)$$

Or, put more plainly, the majority vote is the output class that appears most often in the training data subset in that leaf.

Choosing the best split:

For each possible split, use that split and the training data to construct a decision stump with a majority vote at each leaf.

We then evaluate which stump is best according to some **splitting criteria**. Just as we used (classification) error rate for the performance measure on the decision trees above, we can use error rate as the splitting criteria to choose the best decision stump.

Note: Later we'll learn that error rate as a splitting criterion is problematic and will replace it with other splitting criteria such as mutual information.

1.4.2 From Stumps to Trees

We already have the key pieces. To grow a decision tree from a stump, we just recursively repeat the decision stump process on each leaf, using the subset of the training data that was in that leaf.

We will also need to select the **stopping criteria** to end the recursion. We'll learn more about these later, but for now, a simple stopping criterion is to stop splitting a leaf when its training data subset contains homogeneous output values (output values that are all the same value).