Decision Trees

Nishant Mehta

Lecture 2

Supervised Learning

In supervised learning (specifically, in classification), the goal is to find a rule, or hypothesis, which achieves low ***true error*** with respect to some unknown distribution *P*

Since *P* is unknown, this is only possible if we have some Experience, which will be given to us in the form of a ***training set***

A natural strategy then is to somehow find a hypothesis that has low ***training error***

Supervised Learning

First key question: “From what set of hypotheses, or ***hypothesis space***, will we select our hypothesis?

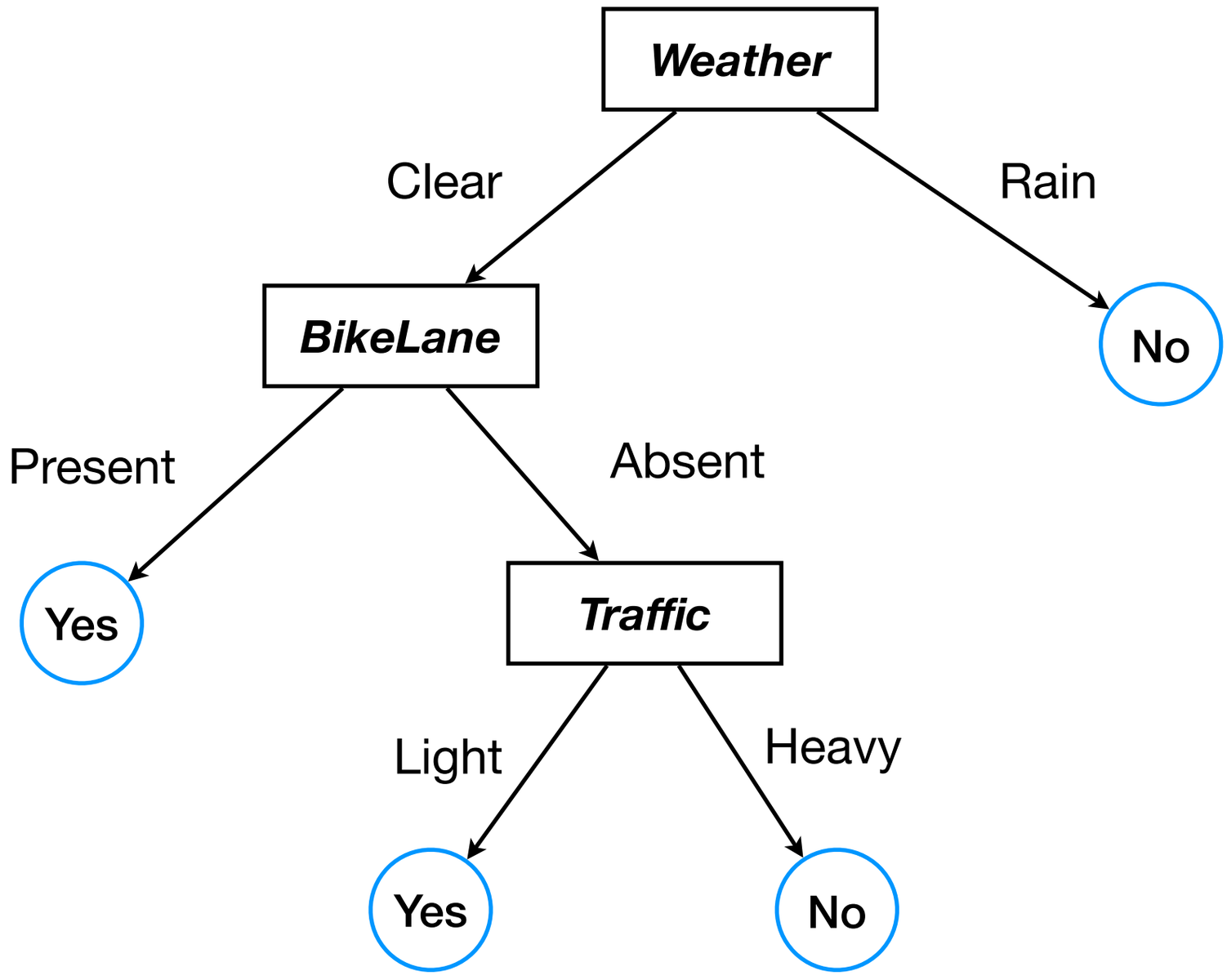
Hypothesis space - a set of hypotheses, where each hypothesis is a function that maps a feature vector (an input vector) to a predicted label

Different hypothesis spaces, along with methods for selecting a hypothesis given the training data, correspond to different machine learning methods.

Today, we’ll begin with an intuitive class of methods, called ***decision trees***

How to represent decision trees

Problem: Predicting whether or not conditions are suitable for cycling



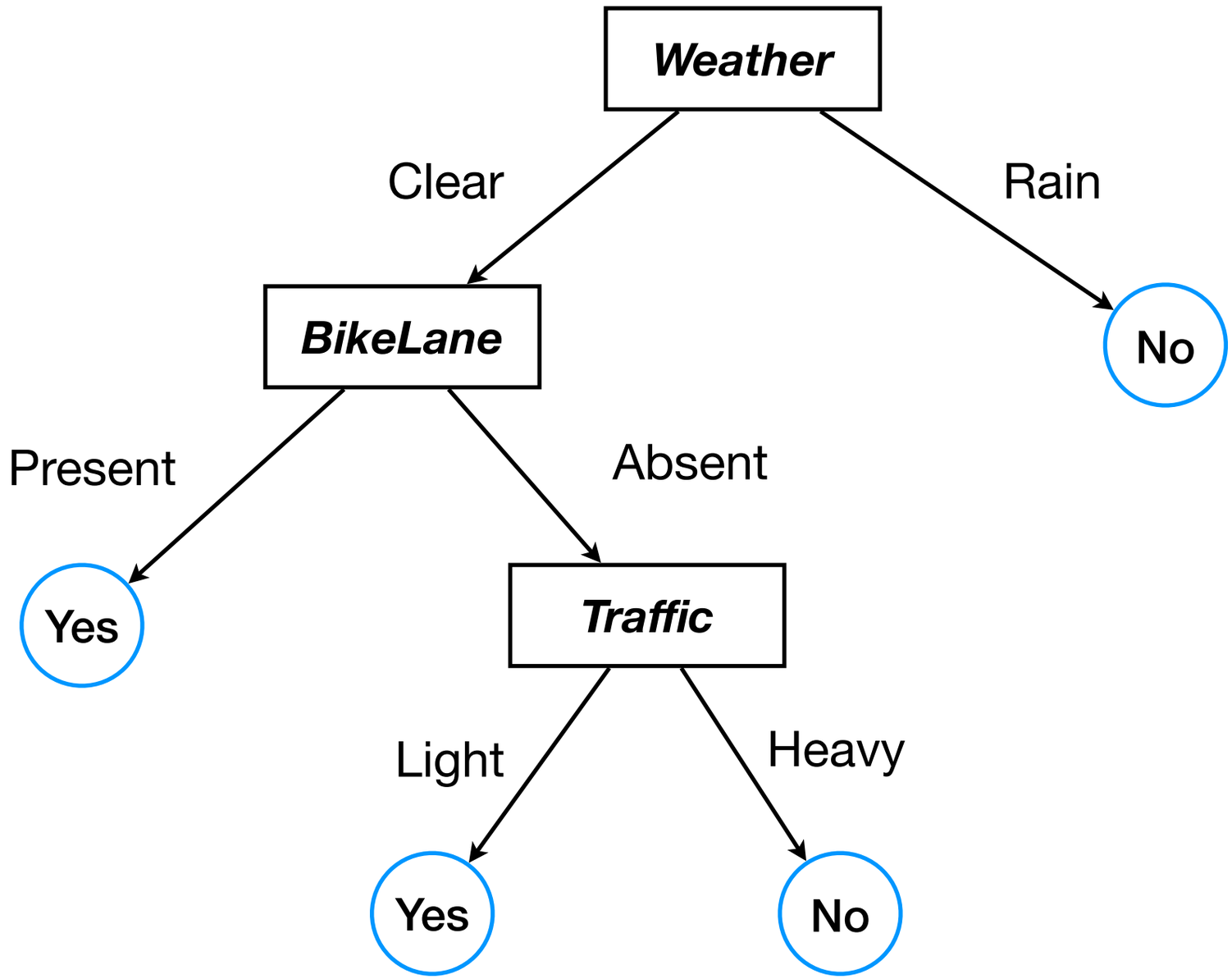
Another representation

In binary classification, any decision tree can be represented as a disjunction of conjunctions:

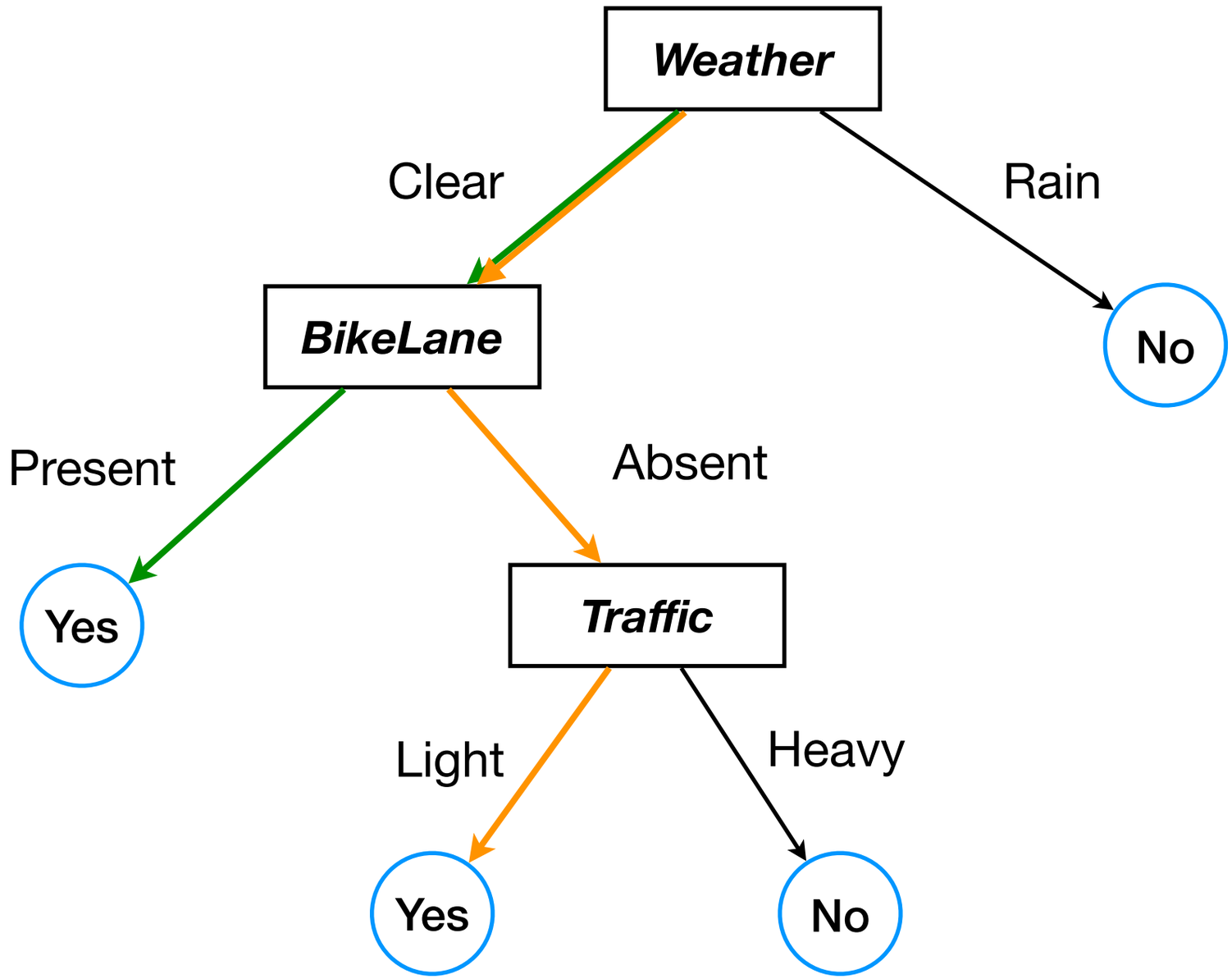
(1) For each root-to-leaf path ending in a positive label, form a conjunction.

(2) Take the disjunction (OR) of all of the conjunctions

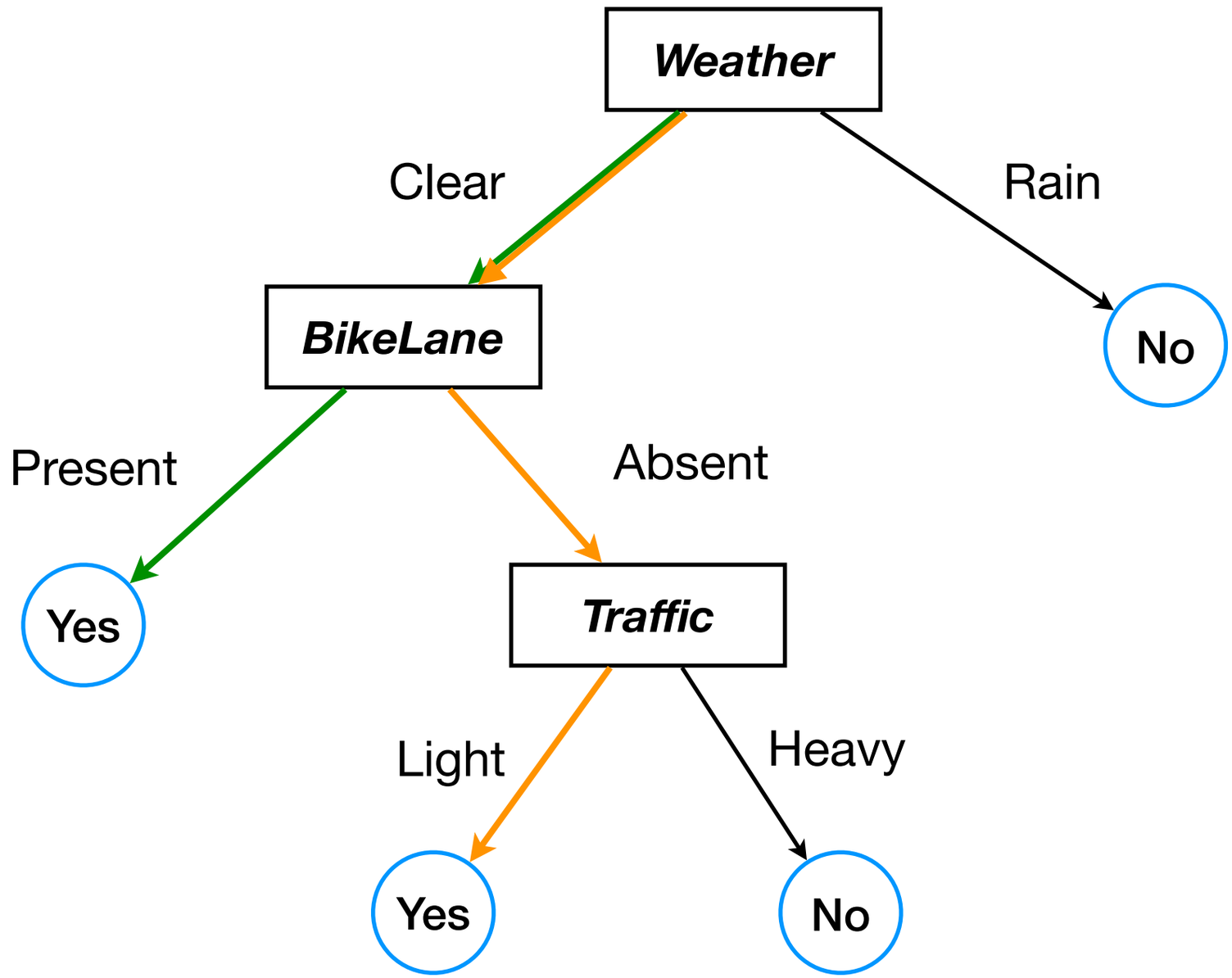
Another representation



Another representation



Another representation



$$(Weather = \text{Clear} \wedge BikeLane = \text{Present})$$

$$\vee\ (Weather = \text{Clear} \wedge BikeLane = \text{Absent} \wedge Traffic = \text{Light})$$

What are decision trees most suited for?

Instances described by feature-value pairs

- Each feature has small number of possible values

    … *but we can also handle continuous features*

Classification (the label is discrete)

… but regression is also possible

Optimal decision tree

Idea: Let's select the optimal decision tree

What do we mean by "optimal"?

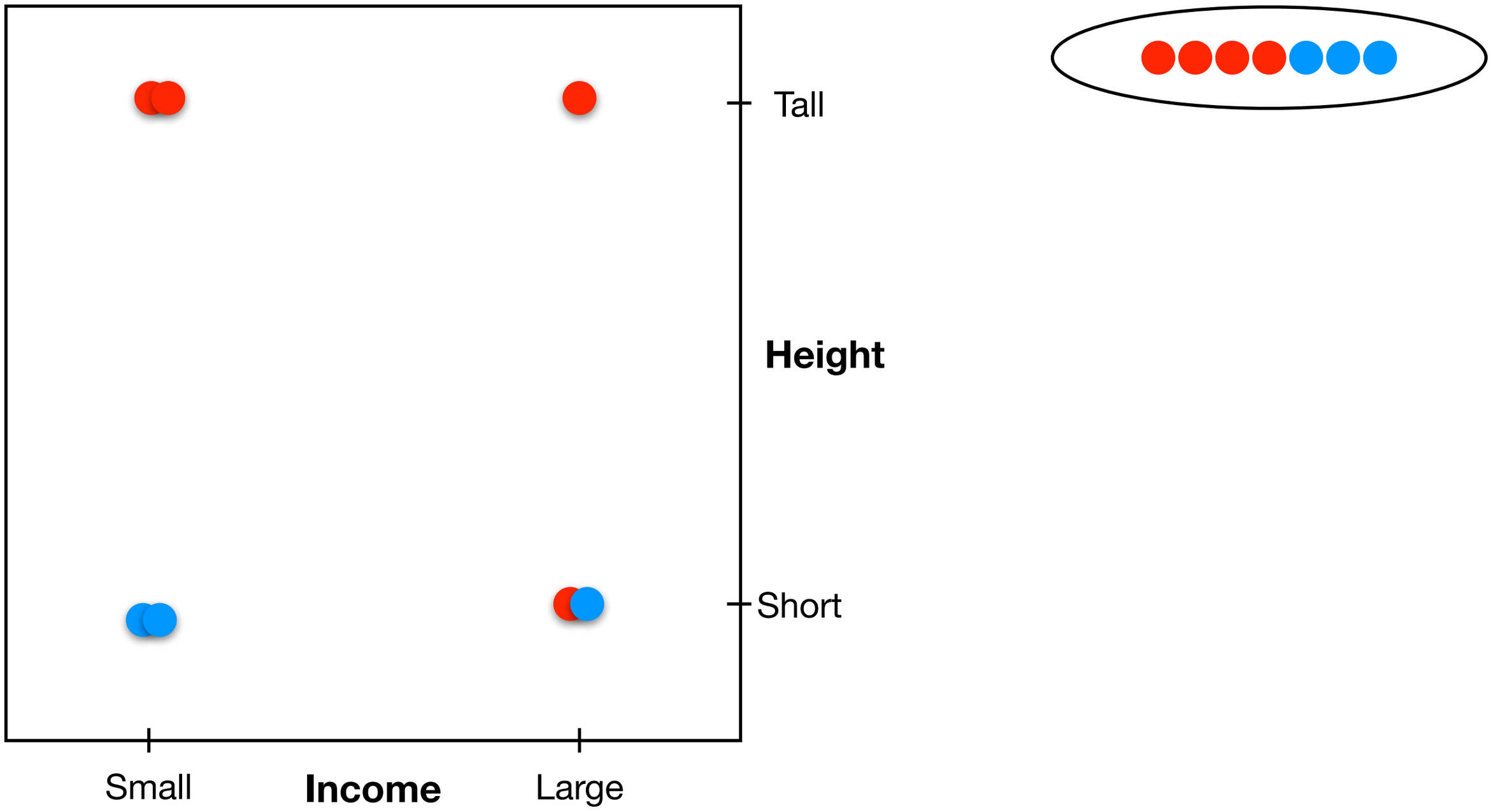
When a decision tree classifies an example, the example follows a path from the root to some leaf

"Optimal" decision tree: minimizes the average path length, where the average is over the training set

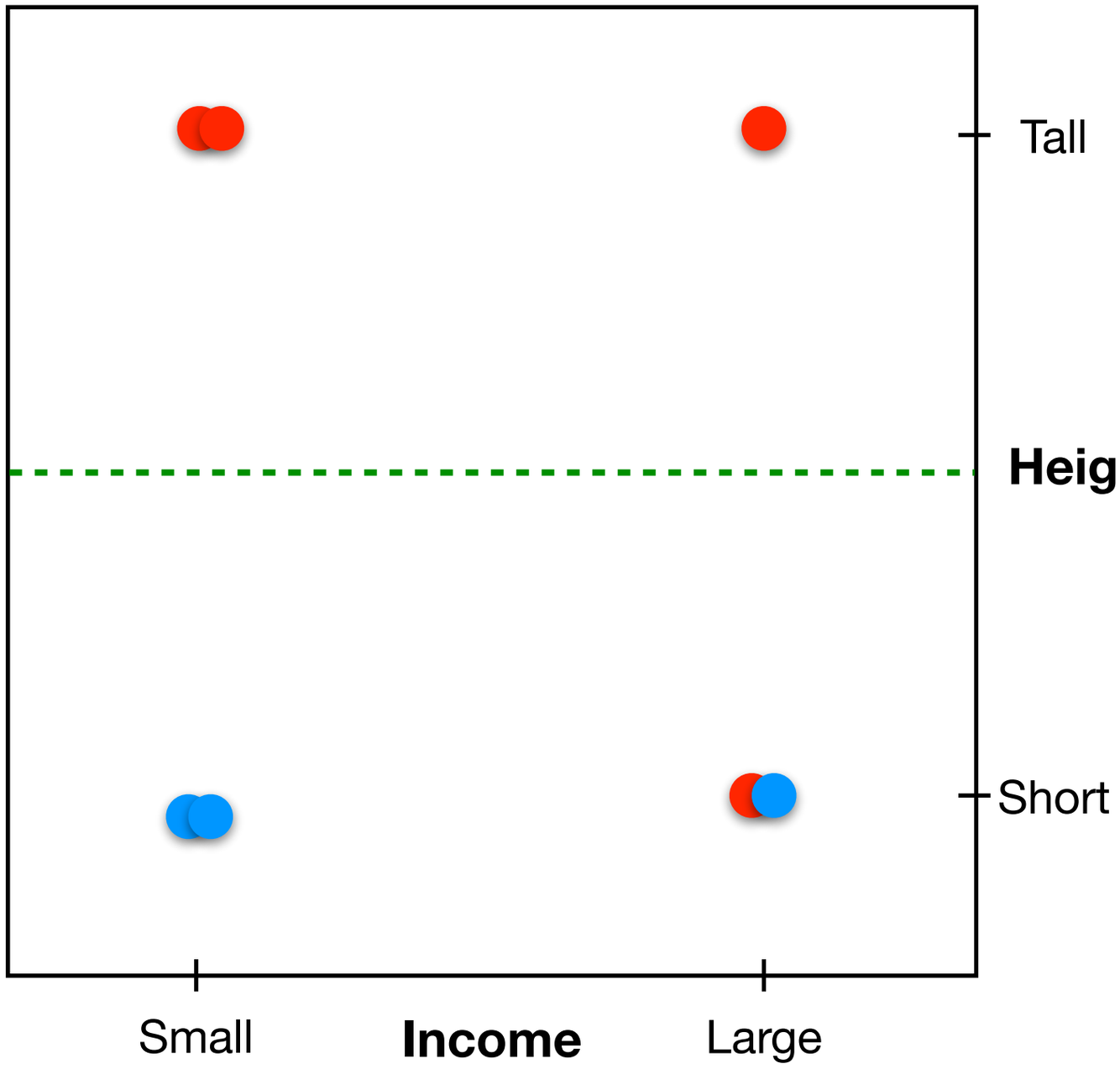
Bad news: The problem of constructing optimal decision tree is NP-complete

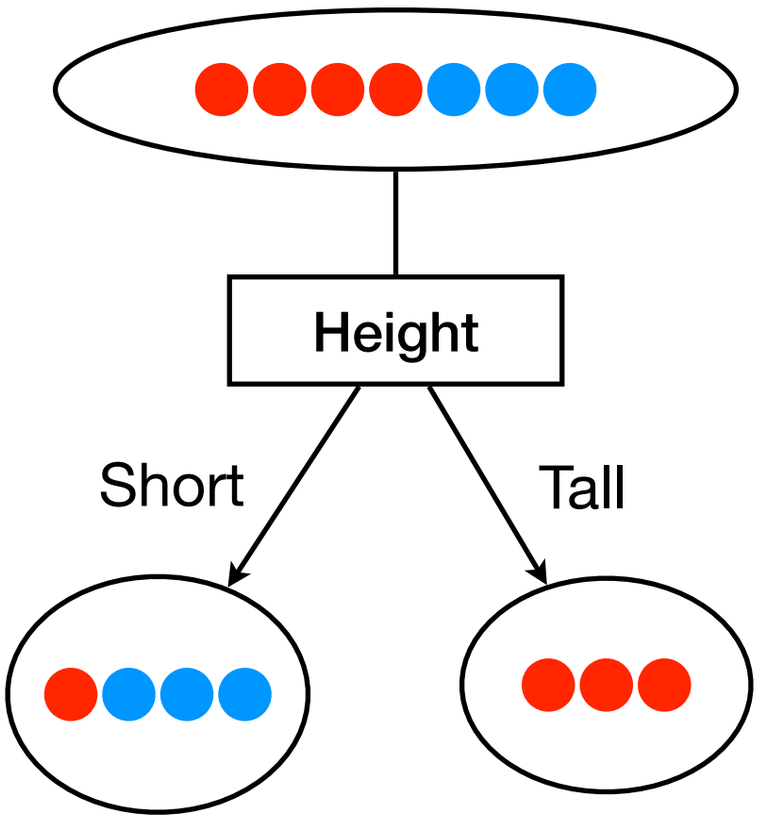
more information

The geometric view

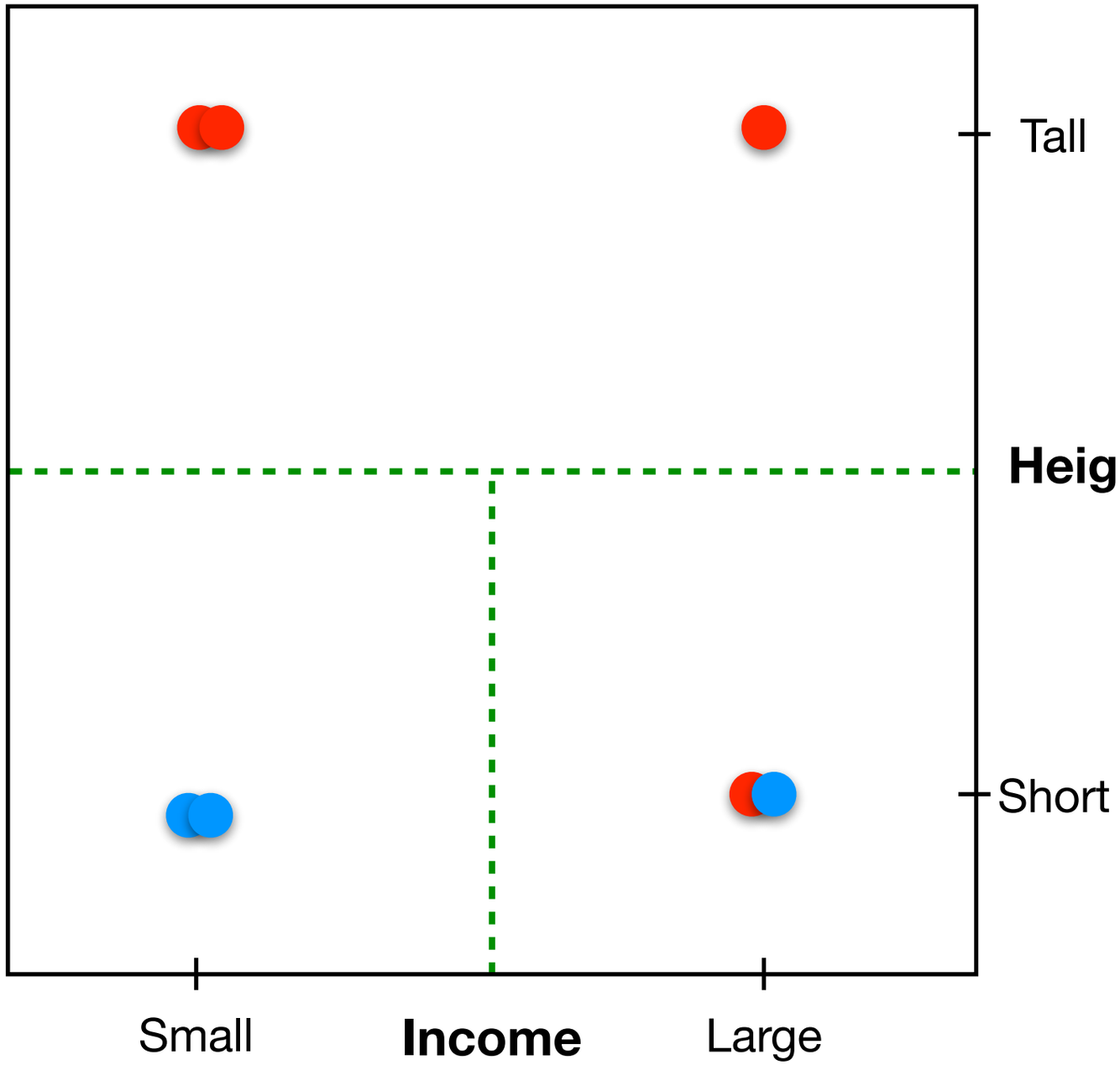


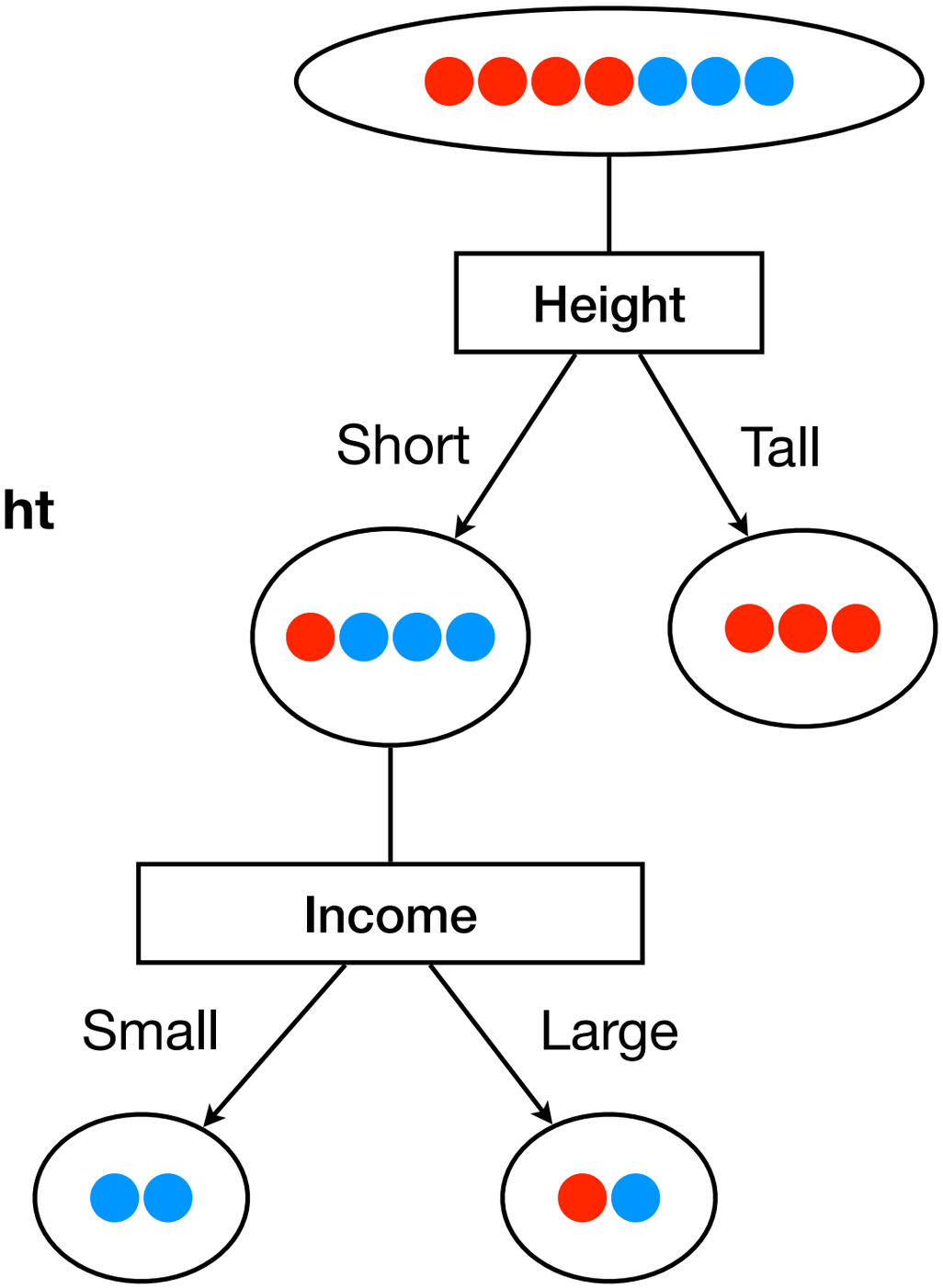
The geometric view



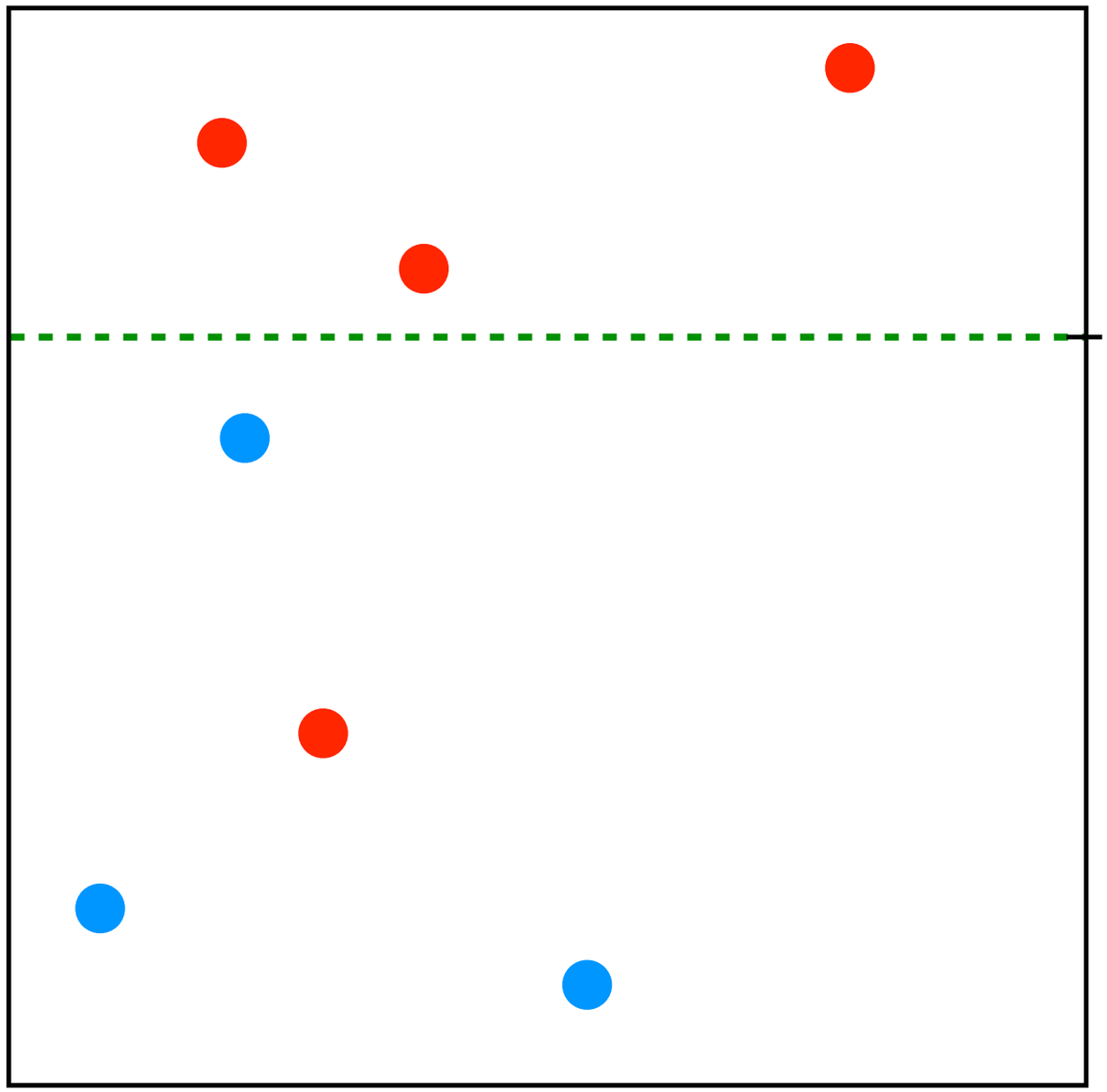


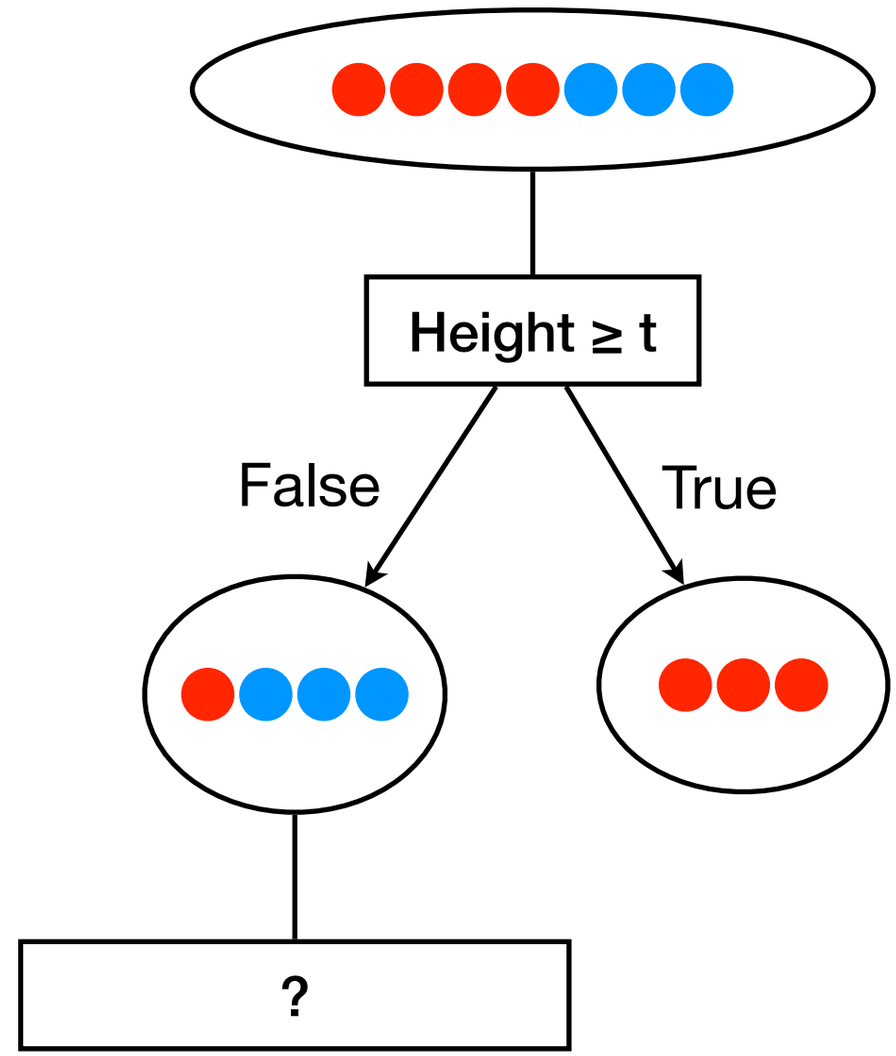
The geometric view





Easily extends to continuous features





Main Question

How to select a feature to split on?

Selecting the split feature: Impurity measures

Intuition:

For a low-depth tree, we want paths to end in leaves as soon as possible

We create a leaf whenever all the examples in a node have the same label

So, let's select a split such that the child nodes are as "pure" as possible (each child node has examples' labels agree as much as possible)

How to measure purity? We need an *impurity measure*

Entropy

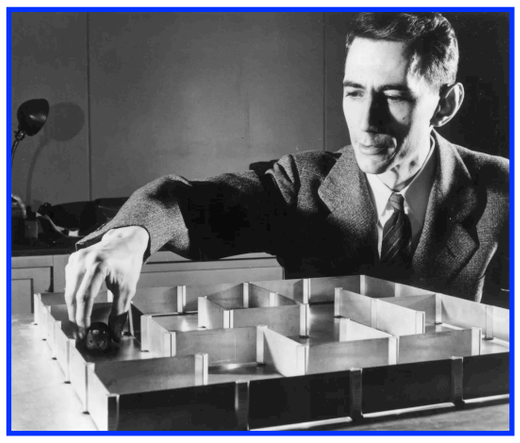
Let Y be a Bernoulli random variable (so, two outcomes: 1 and 0)

Assume that $P(Y = 1) = p_1$ and $P(Y = 0) = p_0$

Then the ***entropy*** of Y is $H(Y) = -p_1 \log p_1 - p_0 \log p_0$

Note that this is the same as $\mathrm{E}_{Y \sim P}\left[\log \frac{1}{P(Y)}\right]$

The “self-information”, or “surprisal”



Entropy was introduced by Claude Shannon

Properties of Entropy

If Y is "pure" (all of its probability mass is on a single outcome), then $H(Y) = 0$

Q: For what Y is $H(Y)$ maximized,
and what is the maximum value?

A:

Properties of Entropy

If Y is "pure" (all of its probability mass is on a single outcome), then $H(Y) = 0$

Q: For what Y is $H(Y)$ maximized,
and what is the maximum value?

A:

Fact: Entropy is concave

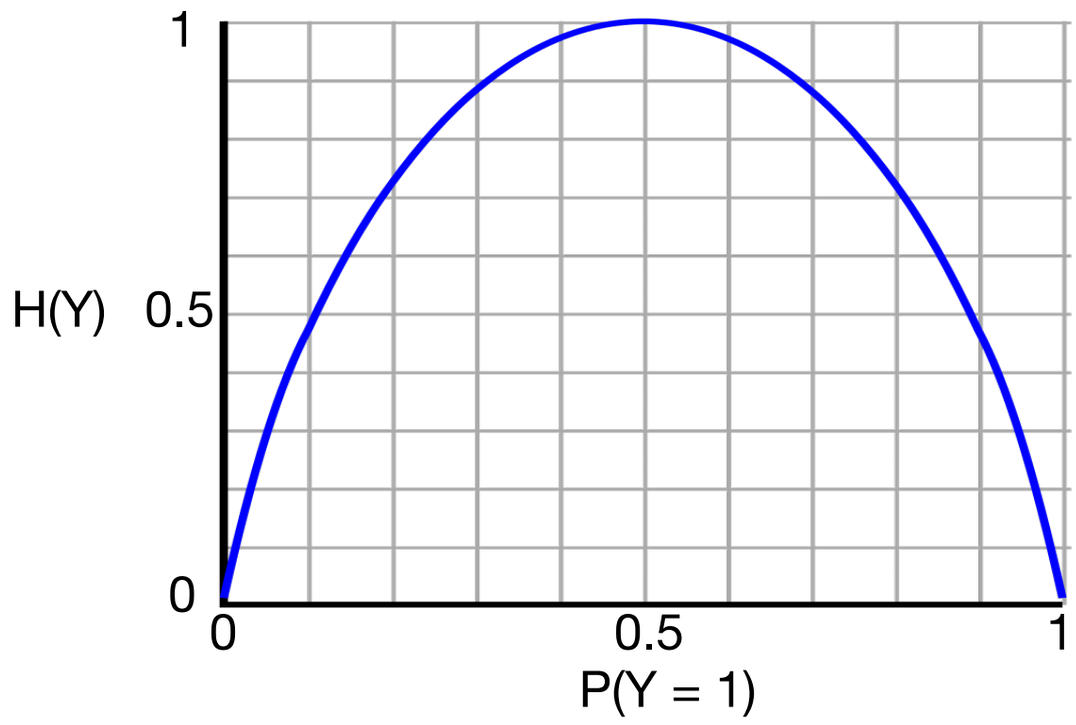
Properties of Entropy

If Y is "pure" (all of its probability mass is on a single outcome), then $H(Y) = 0$

Q: For what Y is $H(Y)$ maximized,
and what is the maximum value?

A:

Fact: Entropy is concave



Entropy for multiple outcomes

The definition of entropy generalizes to multiple outcomes:

Let Y be a random variable taking values $1, 2, \ldots, k$
with $P(Y = j) = p_j$ for $j = 1, 2, \ldots, k$

Then the ***entropy*** of Y is defined as $H(Y) = \sum_{j=1}^{k} -p_j \log p_j$

Entropy of a collection of examples

Now, let S be a collection of n examples in a node, consisting of a positive examples and b negative examples

How to extend definition of entropy to the collection S ?

Entropy of a collection of examples

Now, let S be a collection of n examples in a node, consisting of a positive examples and b negative examples

How to extend definition of entropy to the collection S ?

(1) Define Y_e as a random variable whose distribution is the *empirical distribution* of the class labels with respect to S :

$$P(Y_e = 1) = a/n \qquad P(Y_e = 0) = b/n$$

Entropy of a collection of examples

Now, let S be a collection of n examples in a node, consisting of a positive examples and b negative examples

How to extend definition of entropy to the collection S ?

(1) Define Y_e as a random variable whose distribution is the *empirical distribution* of the class labels with respect to S :

$$P(Y_e = 1) = a/n \qquad P(Y_e = 0) = b/n$$

(2) Define entropy of S as entropy of Y_e

So, $H(S) = -(a/n)\log(a/n) - (b/n)\log(b/n)$

Consequence: If all examples in S have the same label, then entropy is **0**

Entropy after a split: Conditional entropy

Let X be random variable (like a feature) taking values $v_1, v_2, \ldots, v_k$

Conditional entropy of Y given X

$$H(Y \mid X) = \sum_{j=1}^{k} P(X = v_j) H(Y \mid X = v_j)$$

Entropy of conditional distribution $P(Y \mid X = v_j)$

Progress after a split: Information gain

Let X be random variable (like a feature) taking values $v_1, v_2, \ldots, v_k$

Conditional entropy of Y given X

$$H(Y \mid X) = \sum_{j=1}^{k} P(X = v_j) H(Y \mid X = v_j)$$

Entropy of conditional distribution $P(Y \mid X = v_j)$

Then the ***information gain*** of X relative to Y is

$$IG(Y, X) = H(Y) - H(Y \mid X)$$

Progress after a split: Information gain

Let X be random variable (like a feature) taking values $v_1, v_2, \ldots, v_k$

Conditional entropy of Y given X

$$H(Y \mid X) = \sum_{j=1}^{k} P(X = v_j) H(Y \mid X = v_j)$$

Entropy of conditional distribution $P(Y \mid X = v_j)$

Then the ***information gain*** of X relative to Y is

$$IG(Y, X) = H(Y) - H(Y \mid X)$$

Bonus fact

This is equivalent to the ***mutual information*** of X and Y

$$I(X; Y) = H(Y) - H(Y \mid X) = H(X) - H(X \mid Y)$$

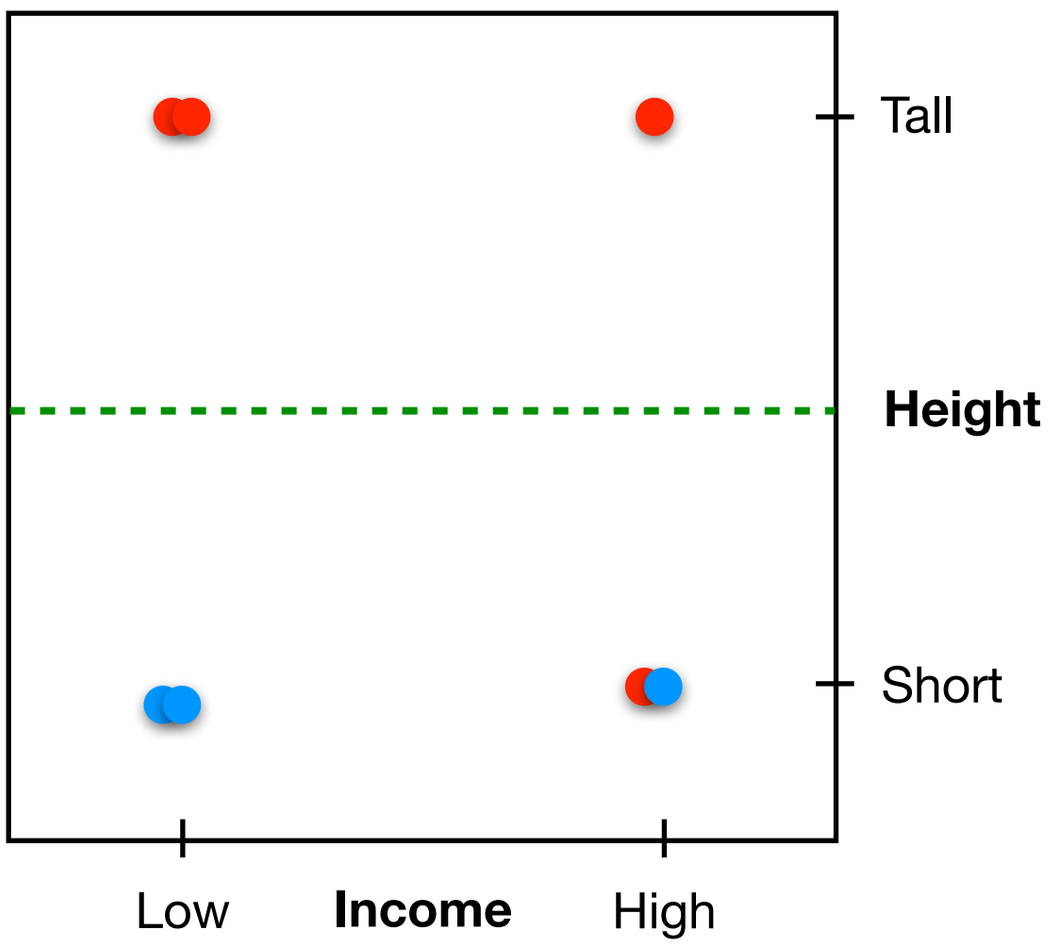
First equality is a definition. Second one is a nice exercise

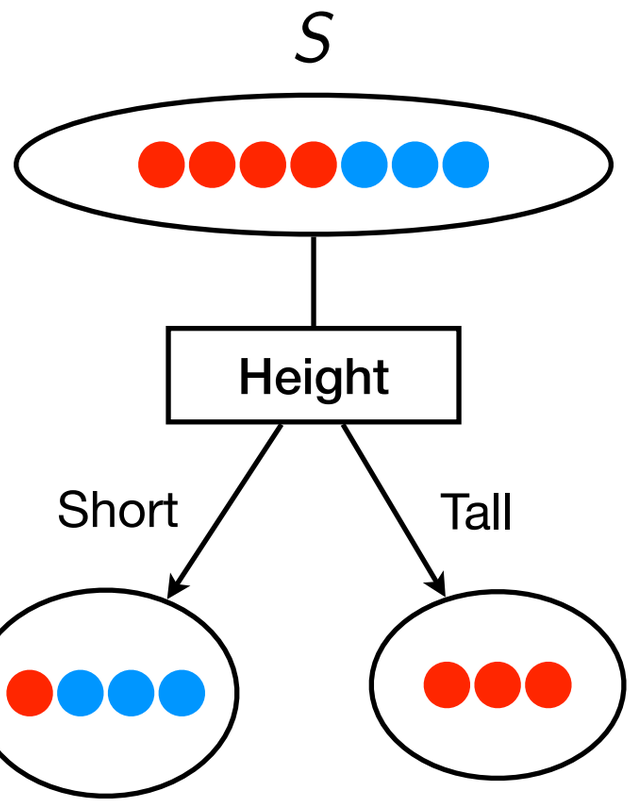
(you will need to recall Bayes rule)

Conditional entropy example

We want to select the feature that minimizes the conditional entropy

This is the same as selecting the feature A that *maximizes* information gain $H(S) - H(S \mid A)$





$H(S \mid \text{Height} = \text{Short}) = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}$

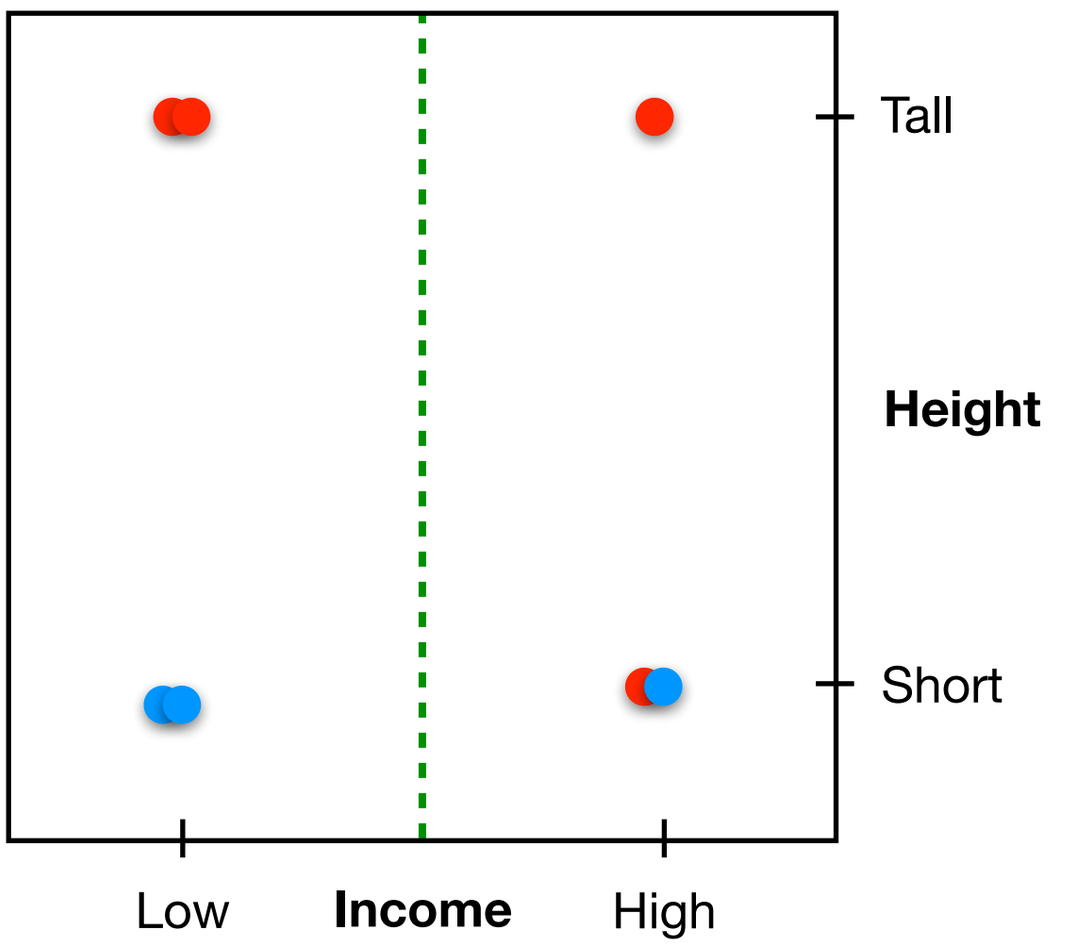
$H(S \mid \text{Height} = \text{Tall}) = -\frac{3}{3}\log\frac{3}{3} - \frac{0}{3}\log\frac{0}{3}$

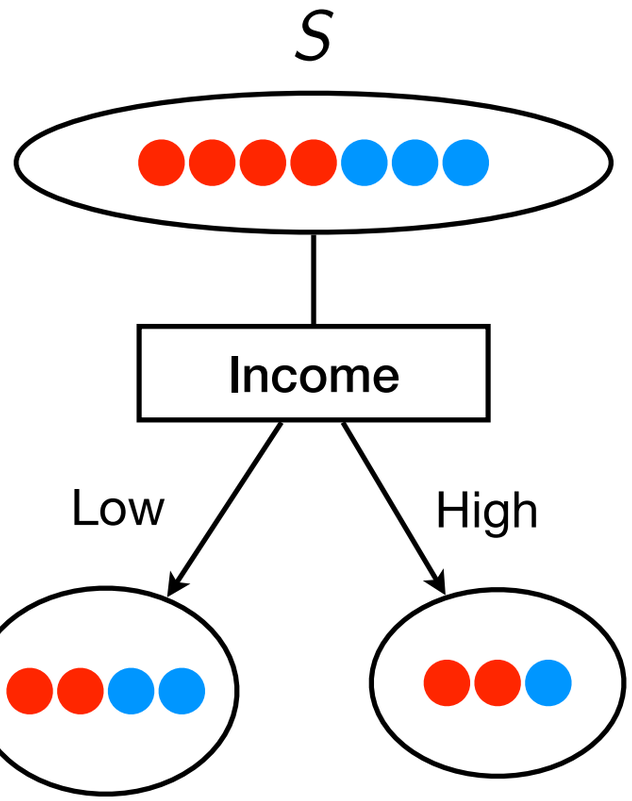
$$H(S \mid \text{Height}) = \frac{4}{7} \cdot H(S \mid \text{Height} = \text{Short}) + \frac{3}{7} \cdot H(S \mid \text{Height} = \text{Tall})$$

Conditional entropy example

We want to select the feature that minimizes the conditional entropy

This is the same as selecting the feature A that *maximizes* information gain $H(S) - H(S \mid A)$





$$H(S \mid \text{Income} = \text{Low}) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{2}\log\frac{2}{4}$$

$$H(S \mid \text{Income} = \text{High}) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}$$

$$H(S \mid \text{Income}) = \frac{4}{7} \cdot H(S \mid \text{Income} = \text{Low}) + \frac{3}{7} \cdot H(S \mid \text{Income} = \text{High})$$

Summary so far

We can devise a greedy strategy for selecting the split feature by using a measure of impurity

One measure of impurity is entropy; we select the attribute which minimizes conditional entropy (maximizes information gain)

Are there other impurity measures that are commonly used?

Yes!

Gini index

The Gini index is another measure of impurity

For Y having k values and $P(Y = j) = p_j$, the *Gini index* is:

$$\text{Gini}(Y) = 1 - \sum_{j=1}^{k} p_j^2$$

Where does this formula come from?

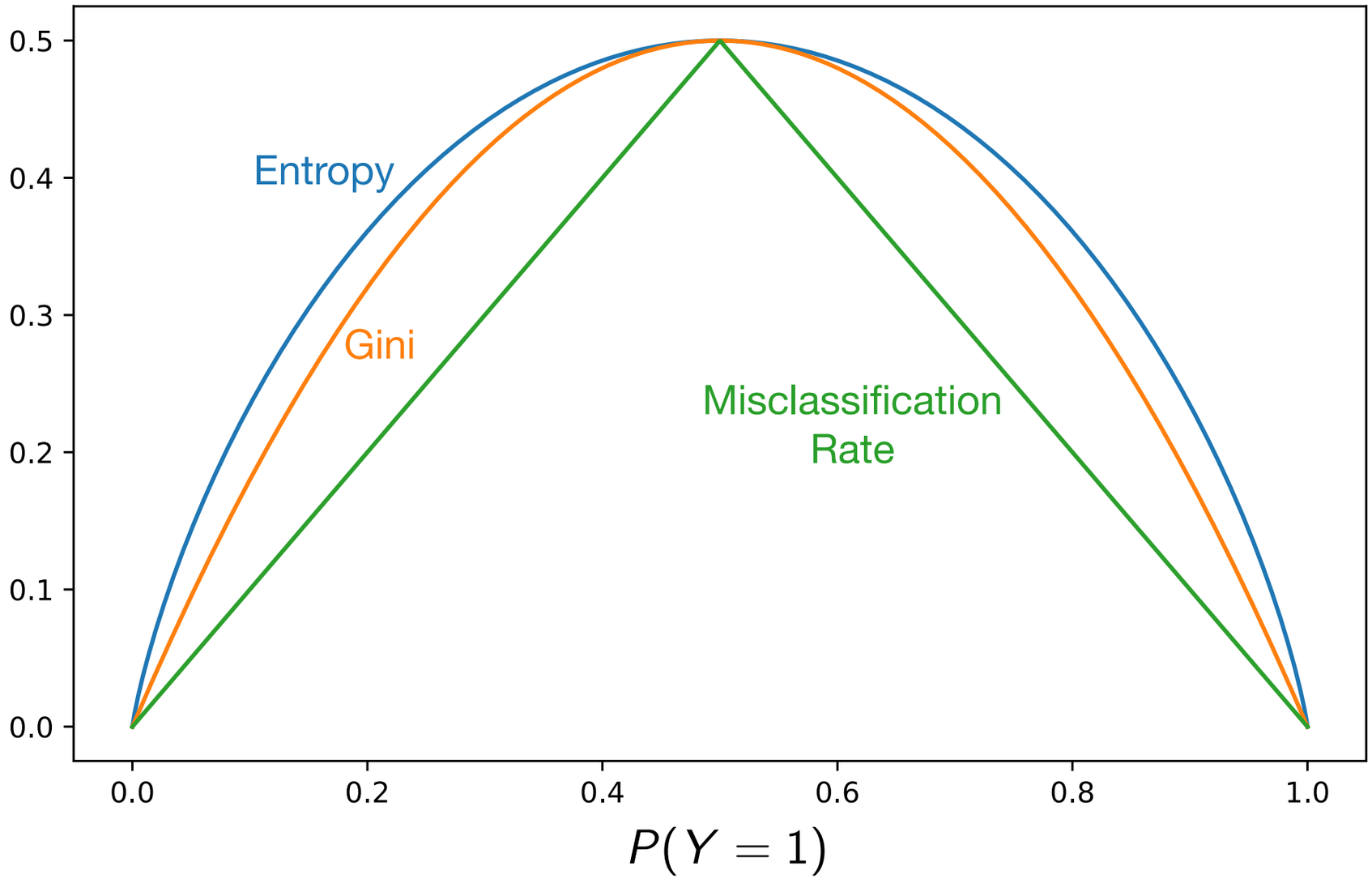
$P(\hat{Y} = j) = p_j$

Consider expected error of *probability matching* predictor assuming its belief about the true label Y is correct

EXERCISE: Show that $\mathbb{E}\left[\mathbf{1}[\hat{Y} \neq Y]\right] = \text{Gini}(Y)$

So, show that $\mathbb{E}\left[\mathbf{1}[\hat{Y} \neq Y]\right] = 1 - \sum_{j=1}^{K} p_j^2$

Gini index vs Entropy vs Misclassification rate for a 2-class problem



Gini is similar to entropy but a bit less aggressive in encouraging purity

Misclassification rate is the probability of an error when predicting according to the majority

Regression

Regression - predicting a continuous target (a real number)

Questions:

What should we predict in a leaf node?

What impurity measure to use?

When should we stop splitting?

What should we predict in a leaf node?

Suppose we have decided to stop splitting a node further, so it becomes a leaf

In classification, our prediction would be the majority vote

For regression?

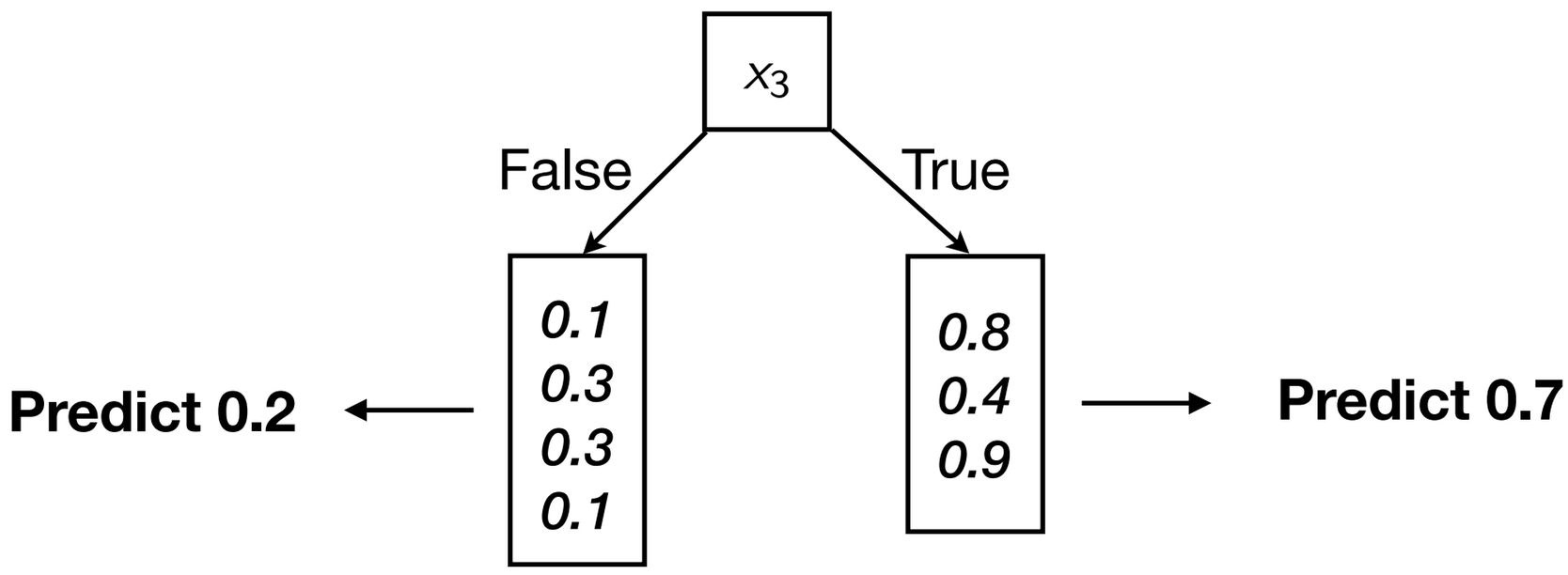
What should we predict in a leaf node?

Suppose we have decided to stop splitting a node further, so it becomes a leaf

In classification, our prediction would be the majority vote

For regression?

Predict the sample mean (i.e. the average)



What impurity measure to use?

Entropy is no longer useful. Why?

A good alternative is to use the (sample) variance

Recall that for targets $y_1, y_2, \ldots, y_n$, the sample variance is

$$S_n^2 = \frac{1}{n-1} \sum_{j=1}^{n} (y_j - \bar{y}_n)^2$$

sample mean

Recall: $\bar{y}_n = \frac{1}{n} \sum_{i=1}^{n} y_i$

When should we stop splitting?

With variance as the impurity measure, it is unlikely that the impurity measure will ever be zero.

So, when do we stop splitting?

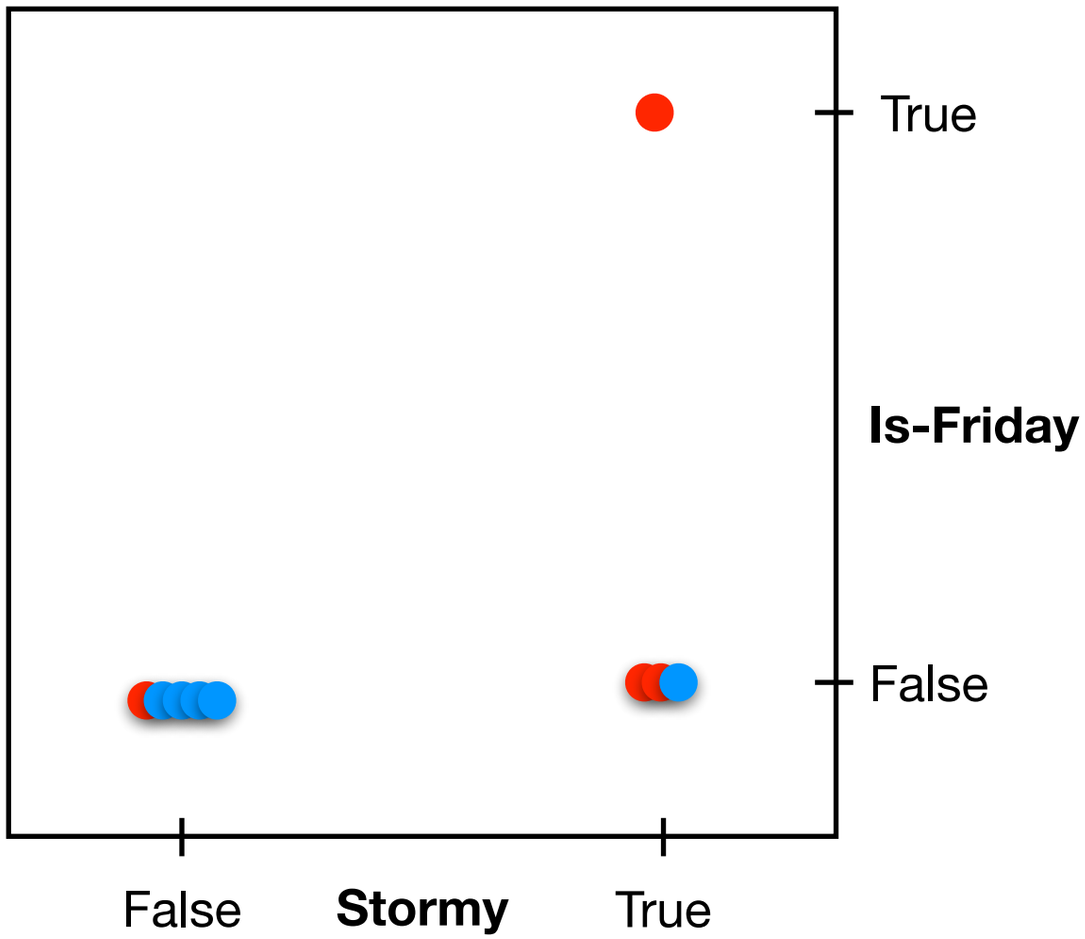
This is actually a general question, also relevant for classification.

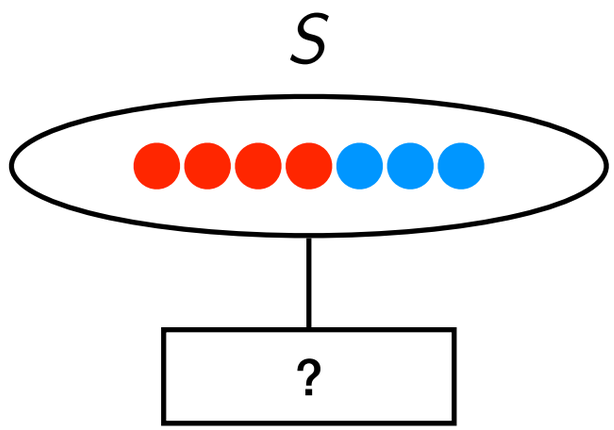
A common solution: Use a "tuning parameter" *B*

> Stop splitting once the number of examples in the node drops below some level *B*

Overfitting

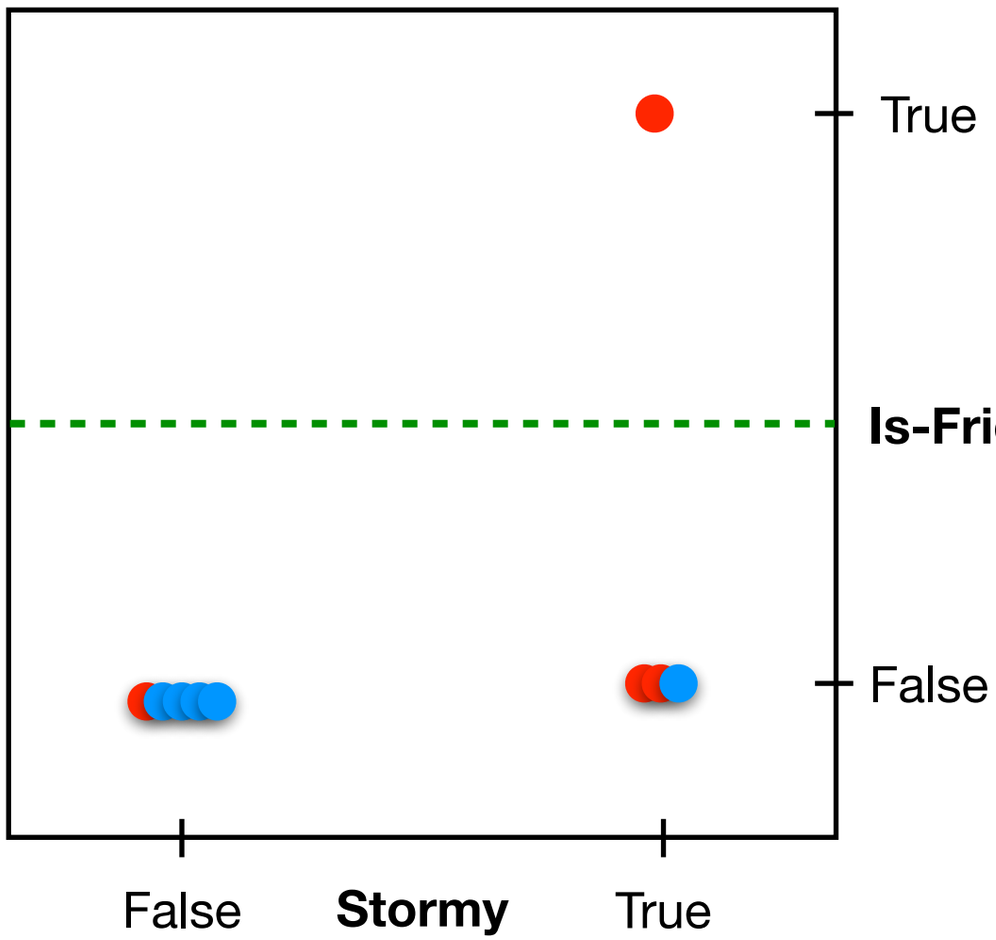
Ferry-on-time dataset

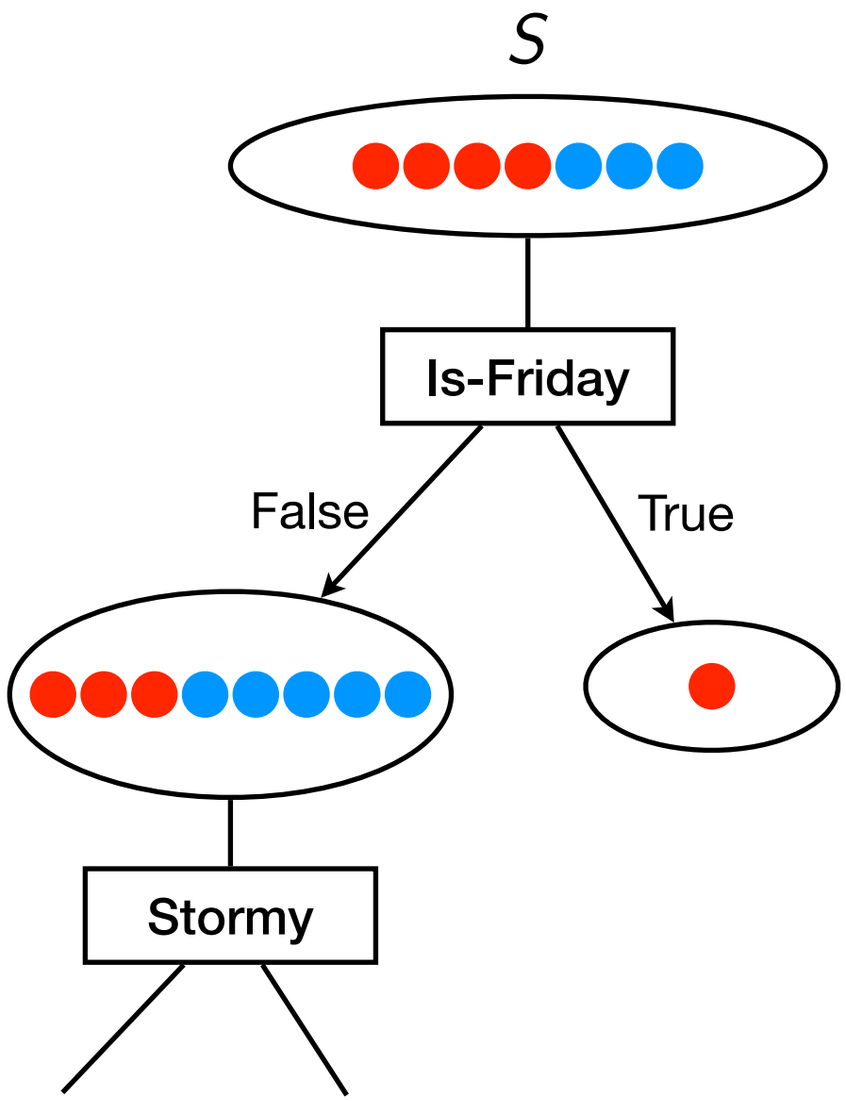




Overfitting

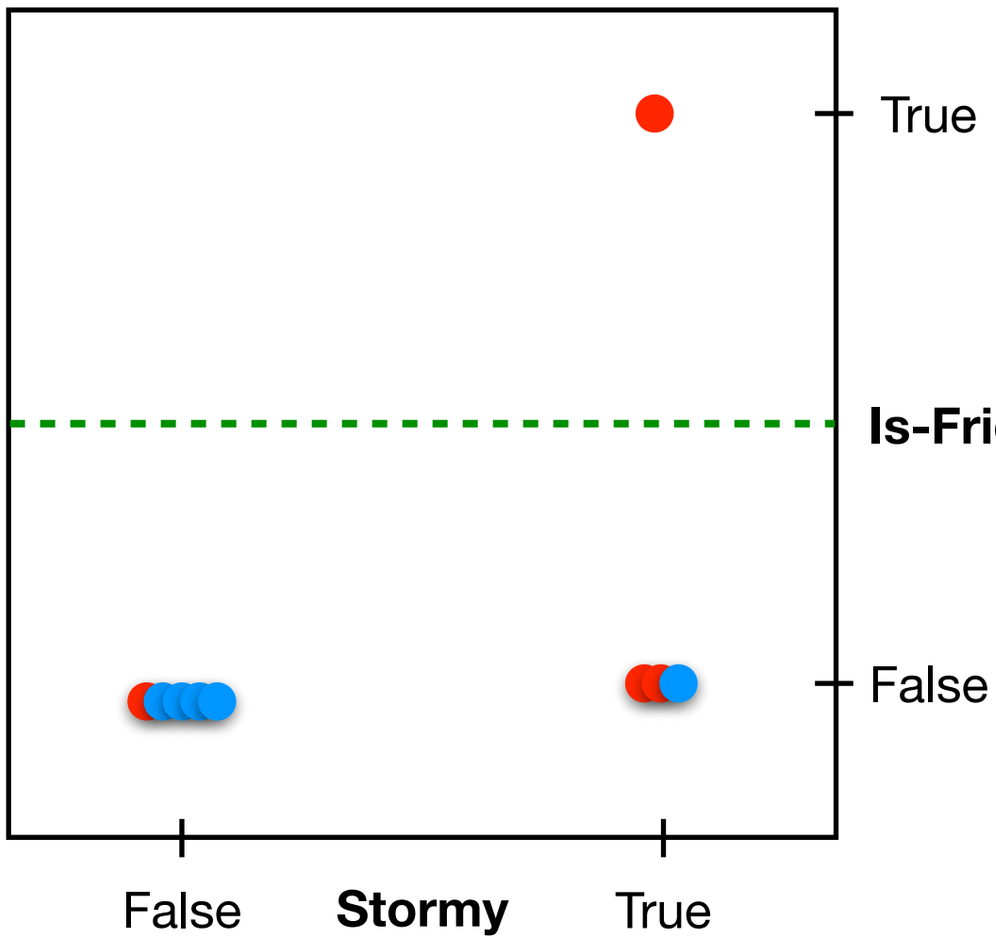
Ferry-on-time dataset

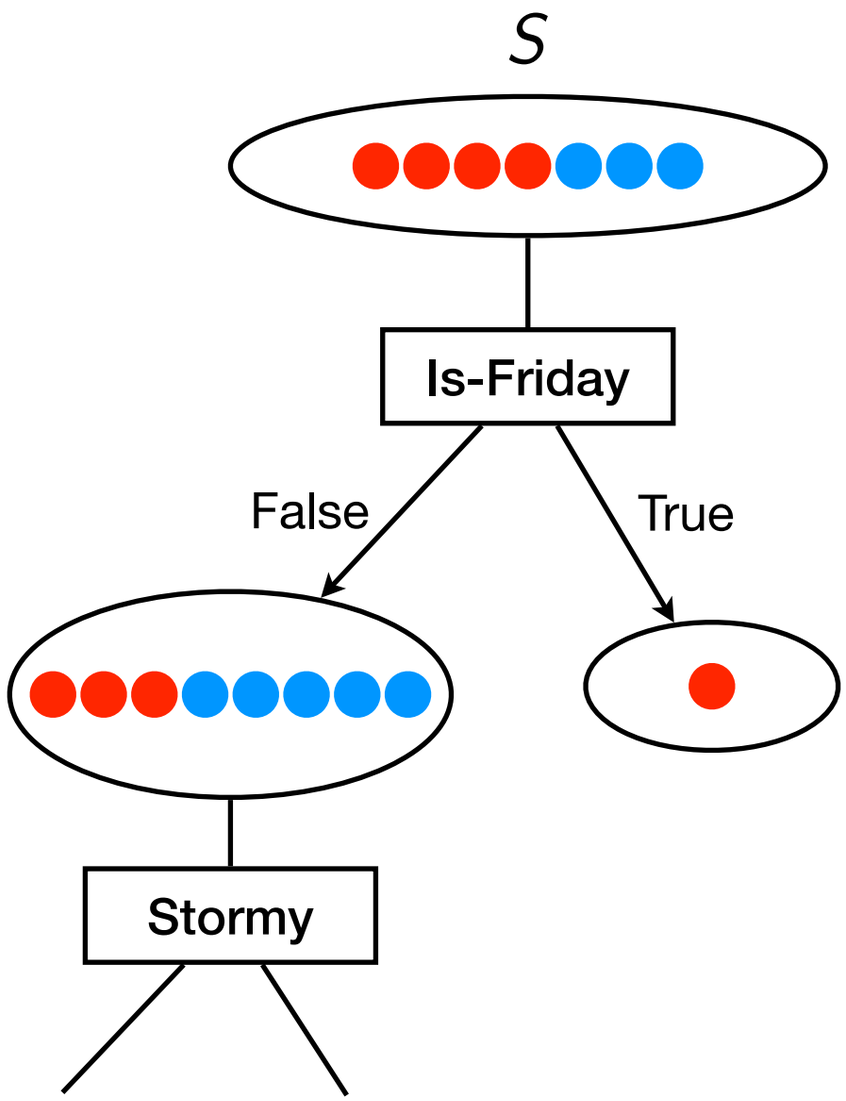




Overfitting

Ferry-on-time dataset





At test time:

It's a Friday. Will the ferry be on time?

Prediction: No. Because it's a Friday (?!)

What went wrong here?

Overfitting

Overfitting generally means an algorithm learns a hypothesis that captures too much information specific to a particular training set, and the information captured is not as relevant to predicting on new examples drawn from the same distribution

More formally: suppose that we learn a hypothesis h for which:

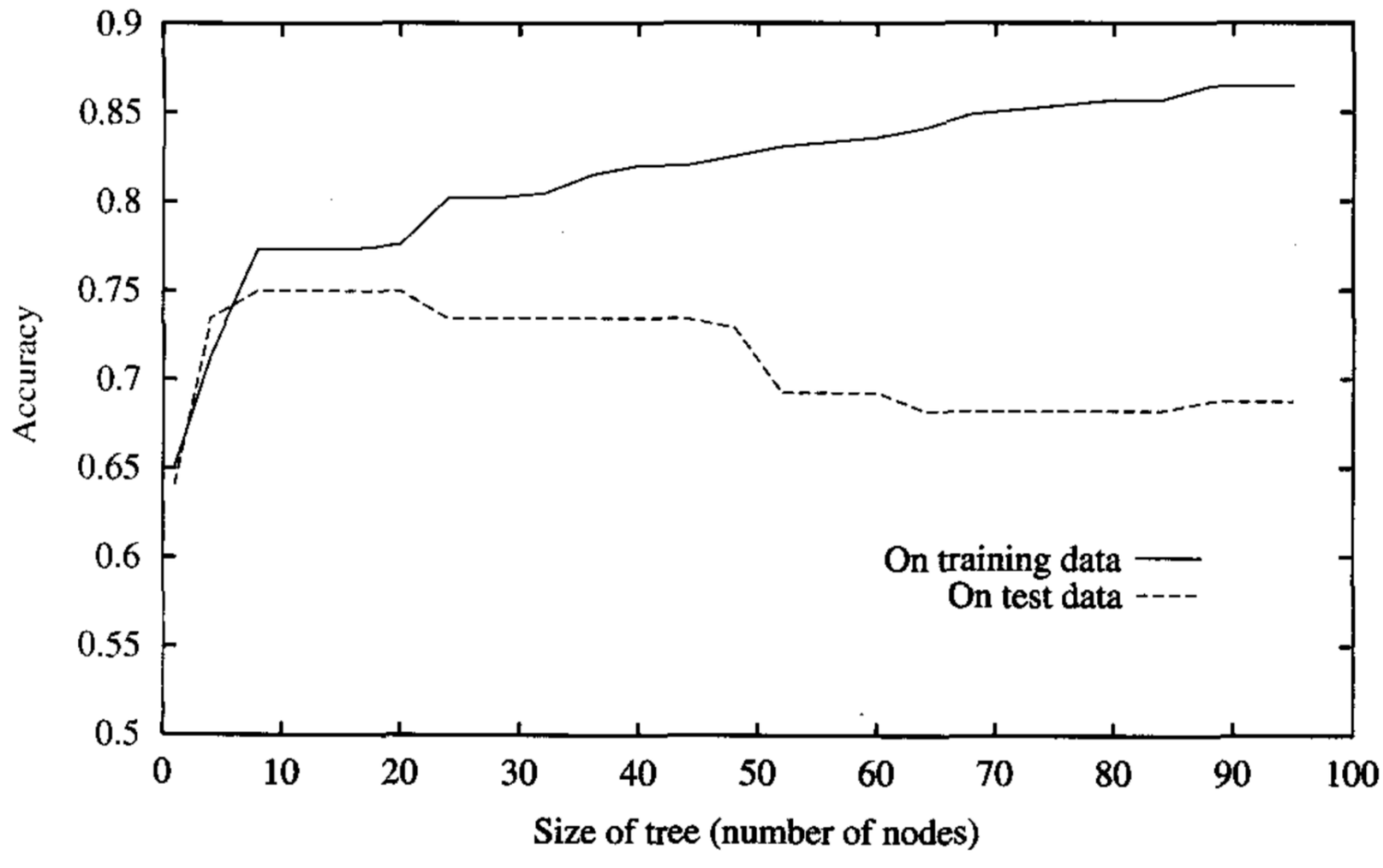
There is another hypothesis h' with higher training error but lower expected error (with respect to a random example being drawn from the underlying distribution of examples)

Then we say h overfits

Overfitting typically happens when we learn a rule that is more complicated than necessary

Common Causes of Overfitting

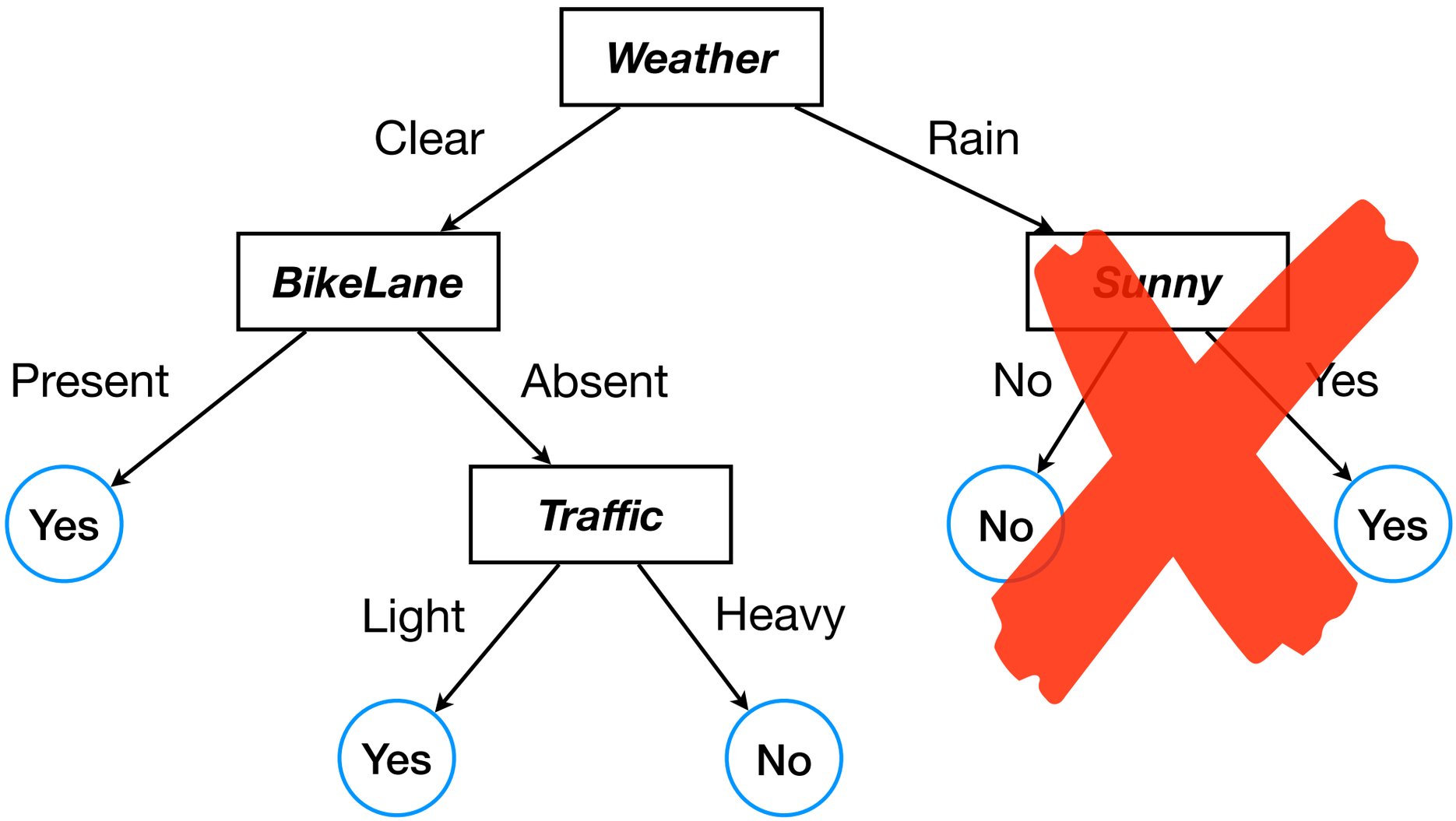
- Noisy examples
- Using a model that is too complicated given the size of the training set



Pruning to combat overfitting

Two types of pruning:

- Pre-pruning (early stopping)
- Post-pruning - First, construct the decision tree completely, resulting in a complex tree. Then, start the actual pruning process.



Post-Pruning

How to prune at a node?

- Make the node a leaf, and assign the label according to *majority vote (classification) or sample mean (regression)*

How to decide whether to prune a node?

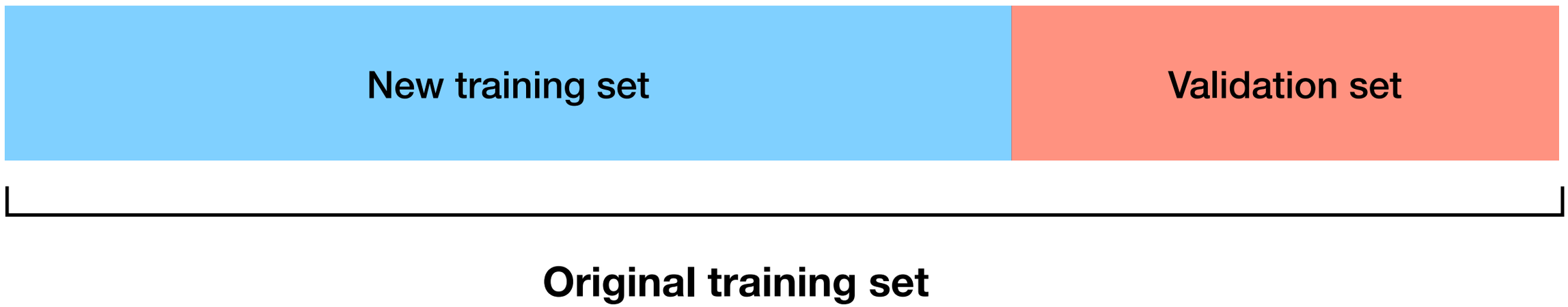
- Use a *validation set*

Validation Set

Randomly split the original training set into two pieces, a new training set and a validation set.

Use the new training set when constructing decision tree

When pruning, we use the error on the validation set as a proxy for the true error



Post-Pruning: Reduced-Error Pruning

How to prune at a node?

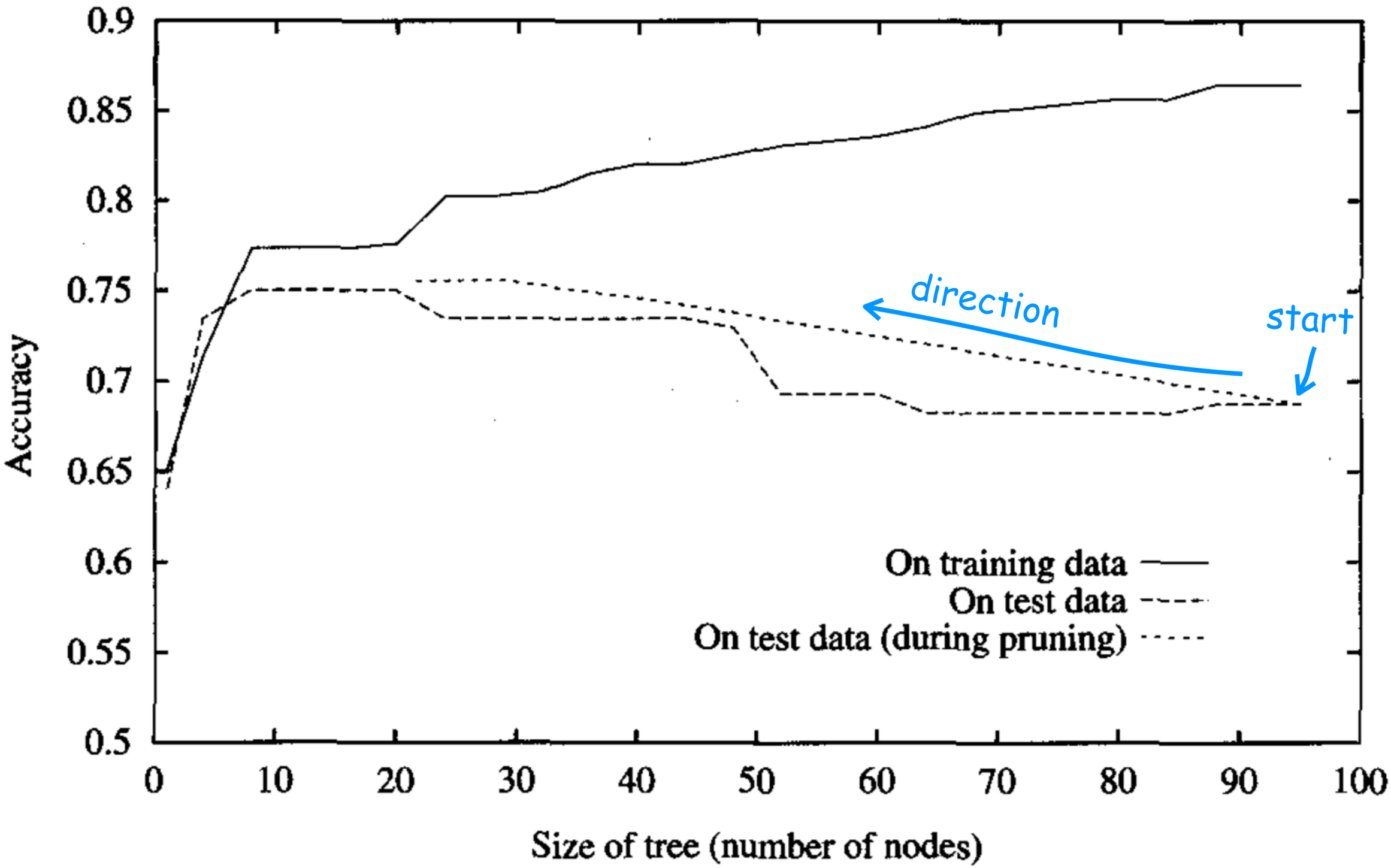
Make the node a leaf and assign the label according to majority vote

How to decide whether to prune a node and which node to prune first?

Consider nodes in post-order (bottom-up traversal). That is, before a node is considered for pruning, each of its children must first be considered for pruning.

For each node, compute the change in validation error due to pruning. If pruning leads to a decrease in validation error, then prune!

The results of reduced-error pruning



From (Mitchell, 1997)