Decision Trees

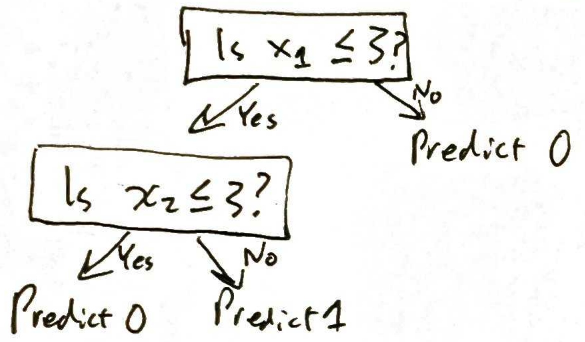
Prediction Rule:

```
f(x) = if ...
          then if ...
          ⋮
       else if ...
          ⋮
          then predict 0
          else predict 1
```

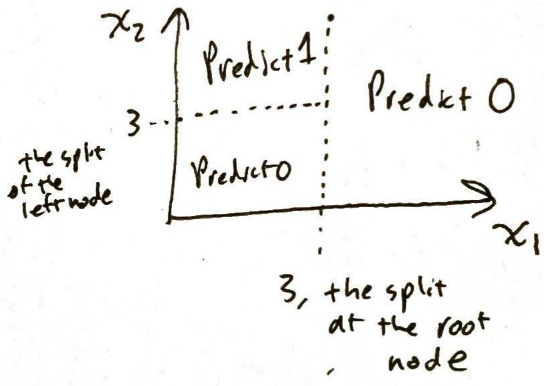
Example

```
If ($x_1 \le 3$)
   then if ($x_2 \le 3$)
      then predict 0
      else predict 1
else predict 0
```

We can represent rules of this form as a tree.
Let $x_1, \dots, x_d$ be the features of the vector x.



Geometrically, each node splits a region into two smaller regions



We need to choose what kind of decisions can appear in a node.
We will follow these conventions:

- Every decision is of the form "Is $x_f \le t$?" where x_f is one feature of our feature vectors, and t is a threshold value.
- The threshold t is chosen only among midpoint values of neighboring values for that feature at that node.

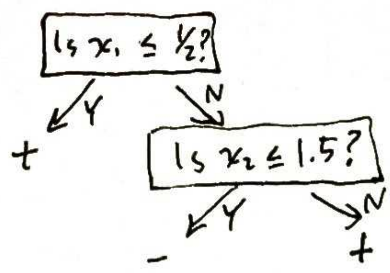
Note: with these conventions, when finding a decision rule for a node, there are only finitely many possibilities.

- Each leaf corresponds to label, which is the prediction made when x lies in the region that reaches that leaf in the tree.

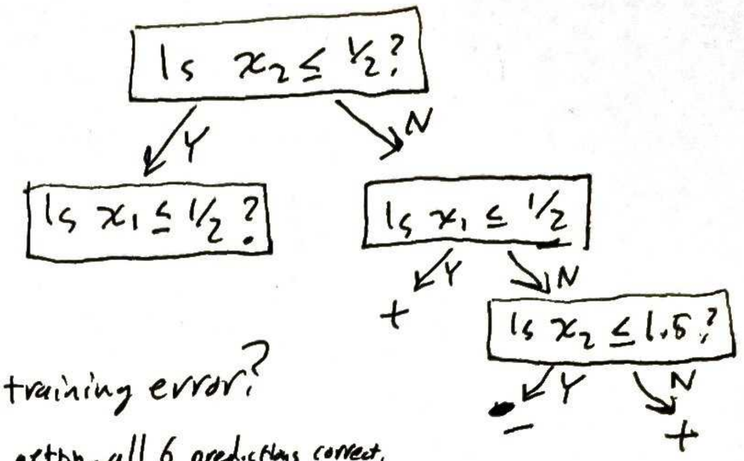
Note: multiple leaves might have the same predicted label.

Example: training data $((0,0),+)$ $((0,1),+)$ $((0,2),+)$
$((1,0),-)$ $((1,1),-)$ $((1,2),+)$

Possible Tree 1:



Possible Tree 2:



Which Tree is better? What is their training error?

$\rightarrow$ They both have 0 training error, getting all 6 predictions correct.

We prefer the <u>simpler</u> one (Tree 1).

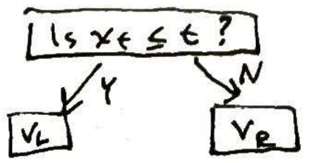
In general finding the smallest decision tree that fits a training data set is computationally difficult.

A commonly used algorithm that proceeds <u>greedily</u> is the <u>ID3 Decision Tree</u> algorithm

<u>ID3 Algorithm</u>

1. Initialize with all the data at a single root node.

2. While there is an impure node in the tree:
   - 2.1. Pick any impure node v
   - 2.2. Pick the best decision rule to split the data at v. ("best" to be defined later)
   - 2.3. Modify the tree by replacing v with this decision plus two child nodes



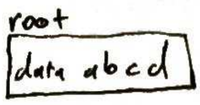
   - 2.4. If either new child v_L or v_R are pure, make them a leaf that predicts their single label.

<u>Impure node</u>: a node with data points of different labels

Each current leaf node corresponds to a <u>subset</u> of the training data that would reach that node.

We will discuss how we select the <u>best</u> decision rule (step 2.2) later.

Example: Training data $\underset{a}{\underset{\uparrow}{((0,0),1)}}$ $\underset{b}{\underset{\uparrow}{((0,1),0)}}$ $\underset{c}{\underset{\uparrow}{((1,0),0)}}$ $\underset{d}{\underset{\uparrow}{((2,0),0)}}$

Initially, just a root node 

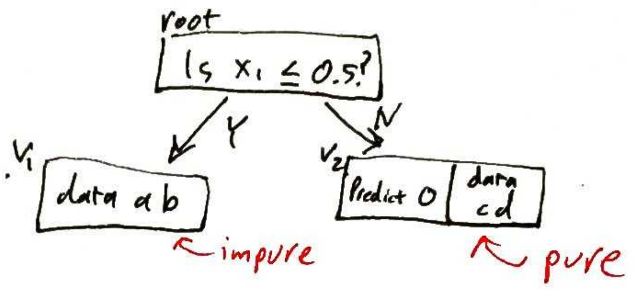
It is impure (label 0 and label 1 are present)

We select it to split. We have 3 candidate decision rules.

Is $x_1 \le 0.5$? Is $x_1 \le 1.5$? Is $x_2 \le 0.5$?

Suppose we pick $x_1 \le 0.5$? as our decision.

~~Upd~~ **Update the tree:**

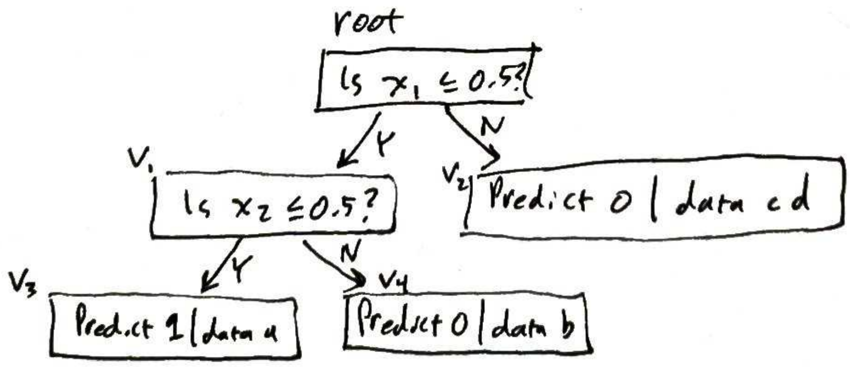


Now v_1, containing a,b, is impure.

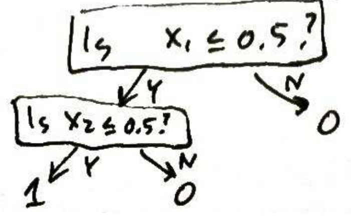
Suppose we select it to split. We have only one candidate decision rule.

We pick $x_2 \le 0.5$? as the decision rule for v_1

Update the tree:

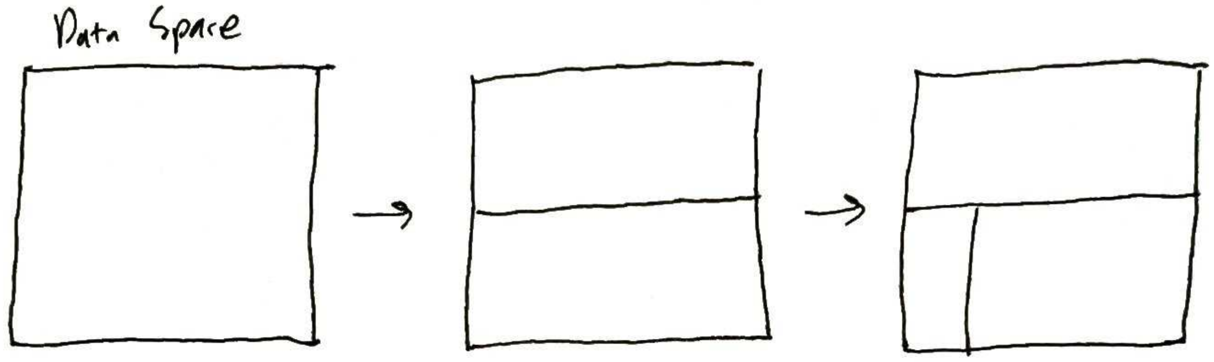


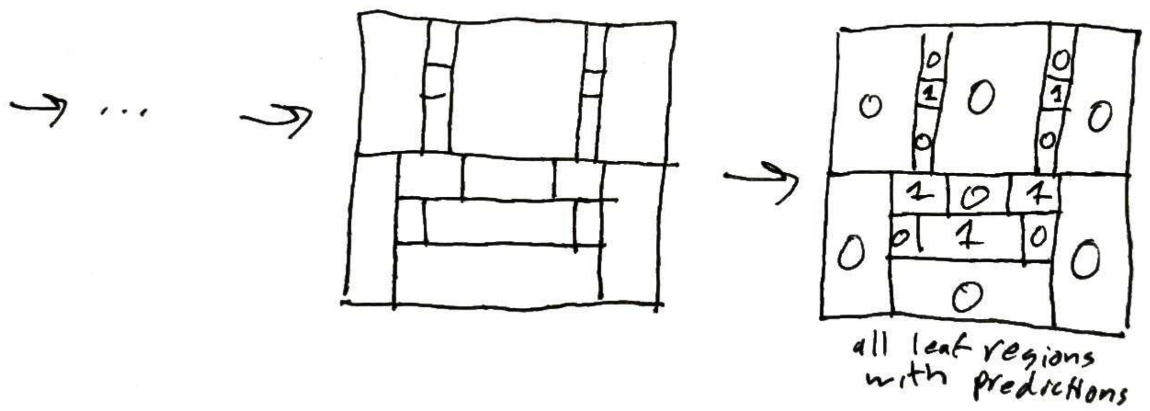
Now all our leaves are pure. We are done.

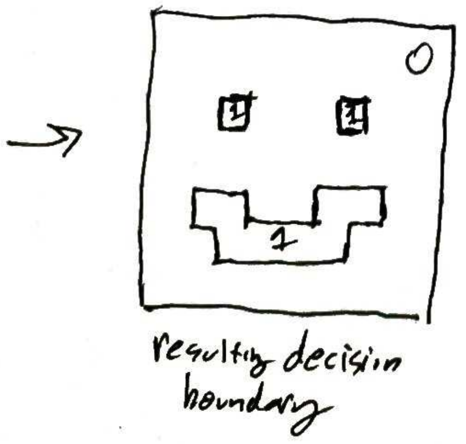
Final tree: 

Geometry of Decision Trees

With our convention of "is $x_t \le t$?" decision rules, each node splits a region into two regions, where the boundary is axis-aligned. With no limit on the number of nodes, this allows the decision tree to carve out arbitrarily complicated decision boundaries







Observe: If we go until all leaf nodes are pure, we will get zero training error.

Selecting a Decision Rule

In the ID3 algorithm we select the decision rule that reduces uncertainty the most.

We measure uncertainty through a quantity from information theory called entropy

Entropy: Let X be a random variable that takes values $v_1, \dots, v_K$ with probabilities $p_1, \dots, p_K$. Then, the entropy of X, denoted by $H(x)$, is defined as

$$H(x) = -\sum_{i=1}^{K} p_i \log p_i = -p_1 \log p_1 - p_2 \log p_2 - \cdots - p_K \log p_K$$

note: by convention, we will declare $0 \log 0 = 0$ even though $\log 0$ is undefined.

note: by convention, we will only use the natural log, ($\ln$ or $\log_e$), with the base e in entropy calculations

Example X: a 0/1 random variable (r.v.) with $Pr(x=1) = \frac{1}{2}$

$v_1 = 0 \quad v_2 = 1 \qquad p_1 = 1/2 \quad p_2 = 1/2$

$$H(x) = -p_0 \log p_0 - p_1 \log p_1 = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = -\log\frac{1}{2} = \log 2$$

$$H(x) = \log 2 \approx 0.69$$

Example X: a 0/1 r.v. with $Pr(x=1) = 0$

$v_1 = 0$
$v_2 = 1$ $\quad p_1 = 1 \quad p_2 = 0$

$$H(x) = -1 \log 1 - 0 \log 0 = -0 - 0 = 0$$

__Observe:__ for a 0/1 r.v., the closer $P_r(x=1)$ is to $1/2$, the higher the entropy.

Does this match what one might think, with the entropy being a measure of uncertainty?

__Example__ X is a r.v. that takes values $1, 2, \dots, k$ each with probability $1/k$

$$H(x) = -\frac{1}{k}\log\frac{1}{k} - \dots - \frac{1}{k}\log\frac{1}{k} = -k\frac{1}{k}\log\frac{1}{k} = -\log\frac{1}{k}$$
$$= \log k$$

__Properties of Entropy__

- $H(x)$ does not depend on the values taken by X $(v_1, \dots, v_k)$ but only on the probabilities $(p_1, \dots, p_k)$ of these distinct values.

- $H(x) \geq 0$ always. $H(x) = -\sum_i p_i \log p_i$

  $0 \leq p_i \leq 1$ Is p_i positive? Is $\log p_i$ positive?

- If X takes k values, $H(x) \leq \log k$. This maximum is achieved only when each value happens w.p. $1/k$

__Conditional Entropy__

This measures the uncertainty when we can condition on other information.

Let X, Z be two r.v.s. The conditional entropy of X given Z is defined as

$$H(X|Z) = \sum_v P_r(Z=v) \underbrace{H(X|Z=v)}_{\text{uses } P_r(X=v'|Z=v) \text{ instead of } P_r(X=v') \text{ in the entropy calculation.}}$$

Example X, Z joint distribution

$Z \backslash X$	0	1
0	$1/3$	$1/3$
1	$1/3$	0

Question: Are X and Z independent?
(No.)

What is $H(X|Z)$?

We need $\Pr(Z=0)$, $\Pr(Z=1)$, as well as $H(X|Z=0)$, $H(X|Z=1)$

$$\Pr(Z=0) = 2/3 \qquad \Pr(Z=1) = 1/3$$

$$\Pr(X=v \mid Z=0) = \frac{\Pr(X=v, Z=0)}{2/3}$$

$$\Pr(X=v \mid Z=1) = \frac{\Pr(X=v, Z=1)}{1/3}$$

$$\rightarrow \Pr(X=0|Z=0) = 1/2, \quad \Pr(X=1|Z=0) = 1/2$$
$$\Pr(X=0|Z=1) = 1, \quad \Pr(X=1|Z=1) = 0$$

$$H(X|Z=0) = -1/2 \log \tfrac{1}{2} - \tfrac{1}{2} \log \tfrac{1}{2} = \log 2$$

$$H(X|Z=1) = -1 \log 1 - 0 \log 0 = 0$$

$$H(X|Z) = \frac{2}{3} \log 2 + \frac{1}{3} \cdot 0 = \frac{2}{3} \log 2 \approx 0.462$$

Exercise: How does the value of $H(X|Z)$ compare to the value of $H(X)$?
Does knowing Z reduce our uncertainty about X?

Example

Joint Distribution:

$\Pr(z=0) = 3/4$

$\Pr(z=1) = 1/4$

$$\begin{array}{c|c|c|c} z \backslash x & 0 & 1 & 2 \\ \hline 0 & 0 & 1/2 & 1/4 \\ \hline 1 & 1/12 & 1/12 & 1/12 \end{array}$$

Conditional Distributions

$\Pr(X=0 \mid z=0) = 0$

$\Pr(x=1 \mid z=0) = 2/3$

$\Pr(x=2 \mid z=0) = 1/3$

$\Pr(x=0 \mid z=1) = 1/3$

$\Pr(x=1 \mid z=1) = 1/3$

$\Pr(x=2 \mid z=1) = 1/3$

$$H(x \mid z=0) = -0\log 0 - 2/3 \log 2/3 - \frac{1}{3}\log 1/3$$
$$= 0 - \frac{2}{3}\log 2 + \frac{2}{3}\log 3 - \frac{1}{3}\log 1/3$$
$$= 0 - 2/3 \log 2 + \log 3 = -2/3\log 2 + \log 3$$

$$H(x \mid z=1) = -\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} = \log 3$$

$$H(x \mid z) = \Pr(z=0)H(x \mid z=0) + \Pr(z=1)H(X \mid z=1)$$
$$= \frac{3}{4}\left(\log 3 - \frac{2}{3}\log 2\right) + \frac{1}{4}\log 3$$
$$= \log 3 - \frac{1}{2}\log 2$$

Properties of Conditional Entropy

- Suppose $X = Z$. What is $H(X|Z)$?

  $H(X|Z) = \sum_v \Pr(Z=v)\, H(X|Z=v)$

  Given $Z=v$, we know $X=v$ w.p. 1

  So $H(X|Z=v)=0$

  Thus $H(X|Z) = \sum_v \Pr(Z=v) \cdot 0 = 0$

- Suppose X and Z are independent. What is $H(X|Z)$?

  Given independence, $\Pr(X=v' | Z=v) = \Pr(X=v')$ for any values v, v'

  This means $X|Z=v$ has the same distribution as X

  so $H(X|Z=v) = H(X)$

  $\rightarrow H(X|Z) = \sum_v \Pr(Z=v) H(X|Z=v) = \sum_v \Pr(Z=v) H(X)$

  $= H(X) \sum_v \Pr(Z=v) = H(X) \cdot 1 = H(X)$

Information Gain

The Information Gain of a variable Z on another variable X is defined as the difference $H(X) - H(X|Z)$

i.e. how much of the entropy (uncertainty) of X is reduced by knowing Z.

It can be shown the information gain is always ≥ 0.

Back to our decision trees...

Selection Rule for Decisions: The "best" decision rule is the one with the largest information gain, i.e. the decision whose result will most reduce the uncertainty we have in predicting a label.

Let Y be a r.v. corresponding to the distribution of labels among the data at a node. (Ex. $\frac{2}{5}$ data points have label 0
$3/5$ datapoints have label 1)

Let Z be a r.v. corresponding to the distribution of answers to a decision rule among the data at a node.
ex. $Z=0$ corresponds to "no", $Z=1$ corresponds to "yes"

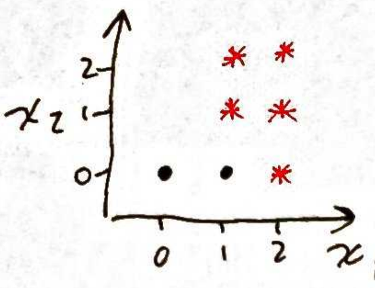
$Y|Z=0$: distribution of labels at this node only among those data points with $x_f > t$.

$Y|Z=1$: distribution of labels at this node only among those data points with $x_f \leq t$.

We pick the decision rule such that its r.v. Z maximizes the information gain on Y.

maximize $\quad H(Y) - H(Y|Z)$

Example Consider these seven training points



What is $H(Y)$? 5 points have label ✱
2 points have label ●

$$H(Y) = -\frac{5}{7}\log\frac{5}{7} - \frac{2}{7}\log\frac{2}{7} \simeq 0.598$$

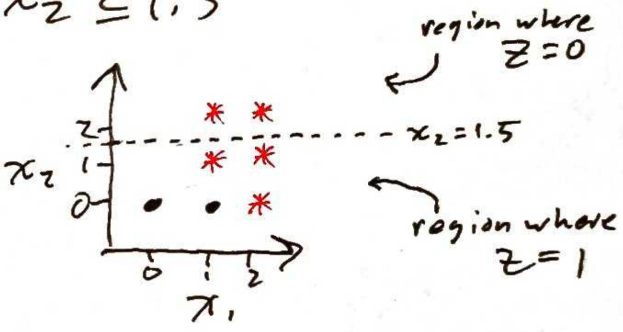
How many candidate decision rules do we have?

4: $x_1 \leq 0.5?$ $x_2 \leq 0.5?$
$x_1 \leq 1.5?$ $x_2 \leq 1.5?$

Let's calculate the information gain for "$x_2 \leq 1.5?$"

$Z=0 \iff$ "no." $\iff x_2 > 1.5$

$Z=1 \iff$ "yes." $\iff x_2 \leq 1.5$



So $Y|Z=0$ has 2 ✱ labels
and 0 ● labels

$\Pr(Y=✱ \mid Z=0) = 1$

$\Pr(Y=● \mid Z=0) = 0$

$H(Y|Z=0) = 0$

and $Y|Z=1$ has 3 ✱ labels
and 2 ● labels

$\Pr(Y=✱ \mid Z=1) = 3/5$

$\Pr(Y=● \mid Z=1) = 2/5$

$$H(Y|Z=1) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} \simeq 0.673$$

Also $\Pr(Z=0) = \frac{2}{7}$, $\Pr(Z=1) = \frac{5}{7}$

So $H(Y|Z) = \Pr(Z=0)H(Y|Z=0) + \Pr(Z=1)H(Y|Z=1) \simeq 0.48$

Information Gain: $H(Y) - H(Y|Z) \approx 0.12$

Let's quickly consider the other decision rules.

$x_1 \leq 0.5?$: $Z=1$: 0*, 1•
$Z=0$: 5*, 1•
$H(Y|Z) \approx 0.38$
infogain ≈ 0.22

$x_1 \leq 1.5?$: $Z=1$: 2*, 2•
$Z=0$: 3*, 0•
$H(Y|Z) \approx 0.39$
info gain ≈ 0.21

$x_2 \leq 0.5?$: $Z=1$: 1*, 2•
$Z=0$: 4*, 0•
$H(Y|Z) \approx 0.27$
info gain ≈ 0.33

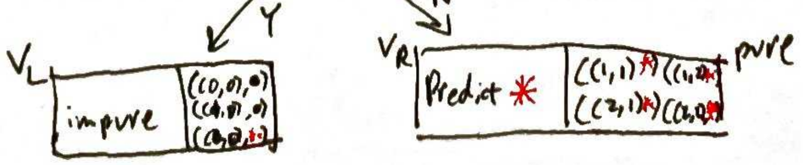
$x_2 \leq 1.5?$: $Z=1$: 3*, 2•
(we computed this one earlier) $Z=0$: 2*, 0•
$H(Y|Z) \approx 0.48$
info gain ≈ 0.12

The distributions of counts of labels on the two sides of the decision are the important part in calculating information gain.

Exercise: try computing these values.

It might help to plot the boundaries of these decision rules

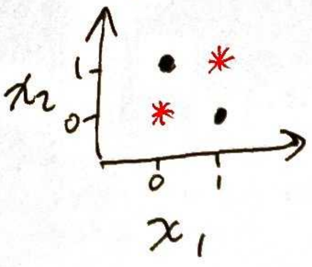
So ID3 will select "Is $x_2 \leq 0.5$?"
to get



... and the process will continue, since we still have an impure node, ...
Exercise: The node v_L has only two candidate decision rules. What are they? Which will be chosen?

Example:

Training Data



Initially, 2 • and 2 *

$H(Y) = \log 2 \approx 0.69$

Two candidate splits

$x_1 \leq 0.5?$: $H(Y|Z) = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = \log 2 \approx 0.69$

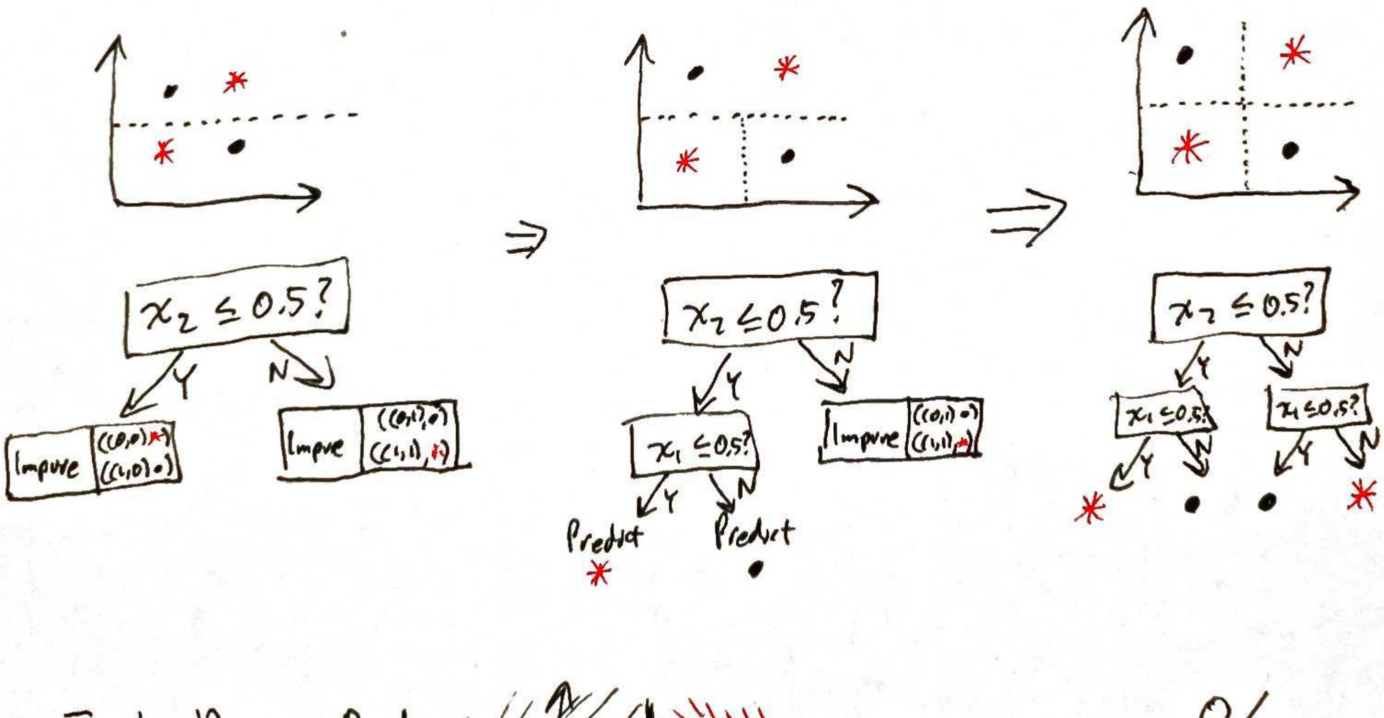
info gain $= 0$

$x_2 \leq 0.5?$: $H(Y|Z) = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = \log 2 \approx 0.69$

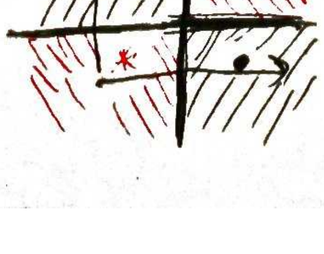
info gain $= 0$

The best info gain is 0. Do we stop? <u>No</u>. We still have impure nodes.

There is a tie? Randomly select one of the tied rules.



Final Decision Boundary:



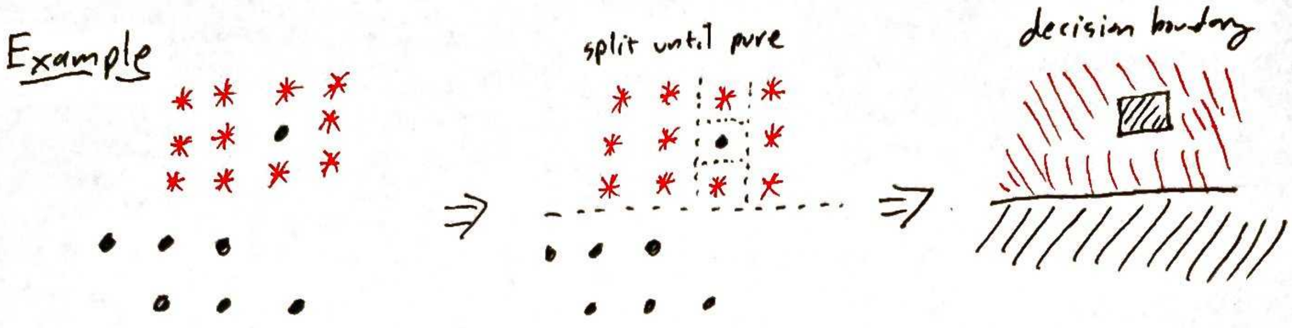
Training Error: 0/4

ID3 Stops when All Nodes are Pure

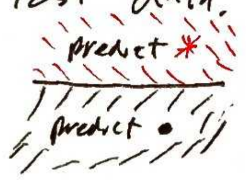
This is always possible if we assume every training point has a distinct feature vector.

When all are pure, we get zero training error.
But this might overfit the data.

Example



That isolated • point might be an outlier, or a misrecorded label
Having that isolated region of predicting • might lead to many mistakes on unseen test data. A simpler tree might be better!



Recall the example about 1-NN vs. 3-NN vs. 5-NN.

What is Overfitting?

Overfitting is when a model or prediction rule too closely matches the peculiarities of a specific set of data, at the cost of performing well on additional unseen data.

We can observe overfitting by looking at the training error versus the true error of a classifier, as we vary the complexity of the classifier.

Assume $(x, y) \sim D$, and we have a training set $(x_1, y_1) \dots (x_n, y_n)$ sampled from D

Training error, for a prediction rule $h(x)$

$$\hat{err}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(h(x_i) \neq y_i)$$

$\mathbb{1}(\cdot)$ is an indicator function

$$= \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

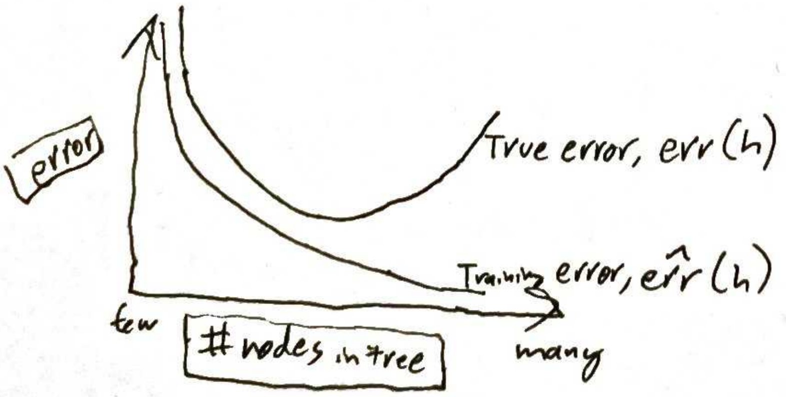
True error, for a prediction rule $h(x)$

$$err(h) = \Pr_{(x,y) \sim D} (h(x) \neq y)$$

(With an unseen test set, we can get a good estimate of $err(h)$ using the test set error rate.)

For a fixed learning algorithm, generally more data helps ensure $\hat{err}(h)$ approaches $err(h)$. More data means outliers are less misleading.

If we plot $\hat{err}(h)$ vs $err(h)$ for a fixed training set, and vary the complexity of the model returned by the ~~[illegible]~~ learning algorithm (e.g. number of nodes allowed in a decision tree) we usually see this behavior:



As the complexity increases, the algorithm can learn more from the data, and the error decreases.

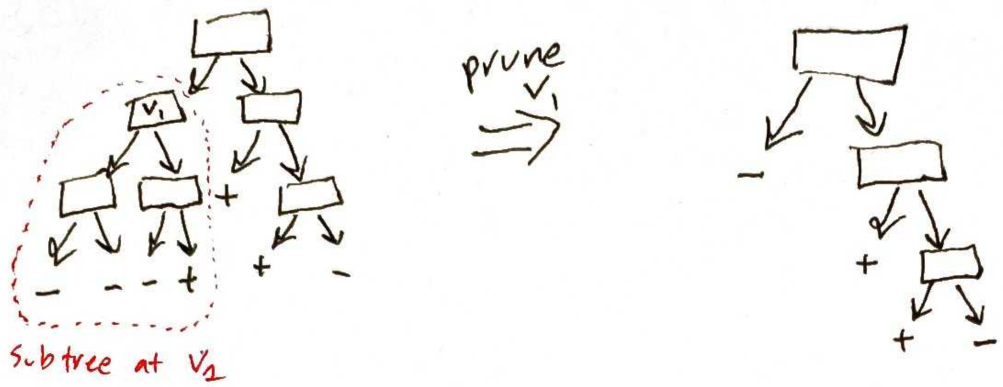
At some point, the true error stops improving and may get worse. Past this point, we say the algorithm is over fitting.

The distribution D has some structure, and we want to learn it for making predictions. The training data shows some of this structure, but it might also have some spurious structure of its own by random chance, which we should not be modelling in our predictions.

Reducing the Complexity of a Decision Tree: Pruning

After splitting until all nodes are pure, our tree may be very complex, many nodes.
Some of these nodes might be carving out small precise regions that overfit.

So we can prune, or cut out entire subtrees at a node



We replace a decision node with a leaf that predicts the majority label of the data at that new leaf.

But how can we tell which nodes to prune?

With not-yet-seen validation data to estimate the true error.

ID3 with Pruning

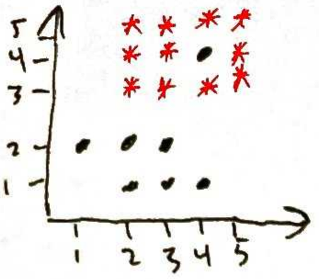
1. Split our labelled data into two sets: training set S and validation set V
2. Build full tree T with ID3 on the training set S.
3. Prune the tree:

```
For each node $v$ in $T$
    Make $T'$, which is $T$ modified to prune at $v$ (replace subtree at $v$ with
                                                      a majority-predicting leaf)
    If error($T'$) on $V \leq$ error($T$) on $V$, then update $T$ to keep this prune
      (update $T = T'$)
```

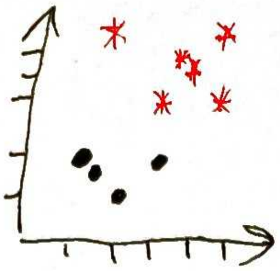
If error$(T') \leq$ error(T), then the validation data suggests pruning this node is a good idea.

Example

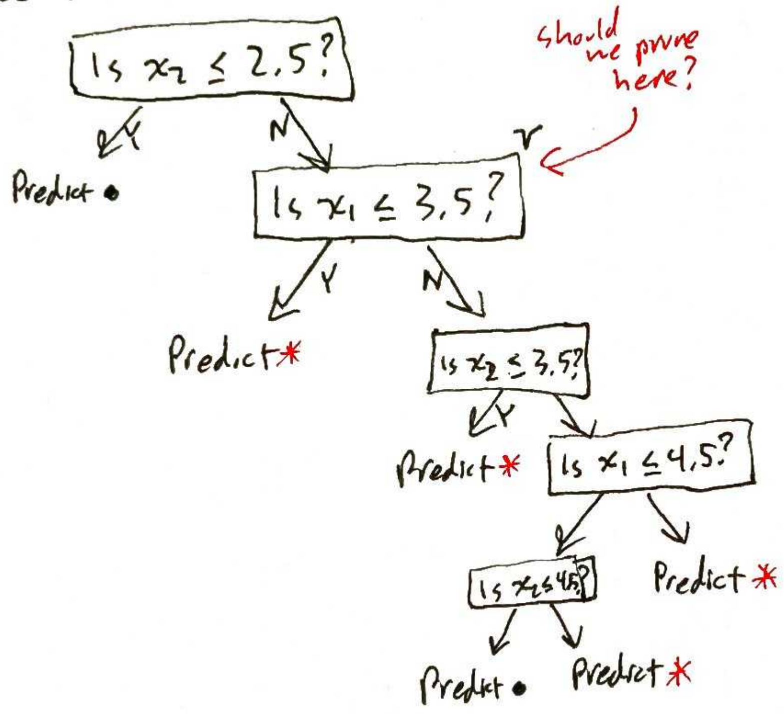
Training Data



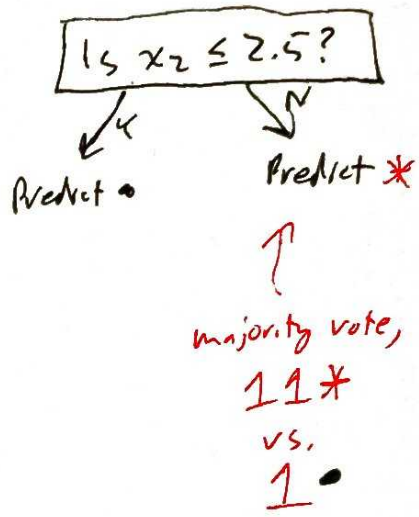
Validation Data



tree T:

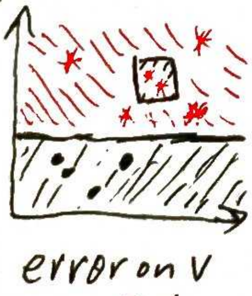


Tree T':



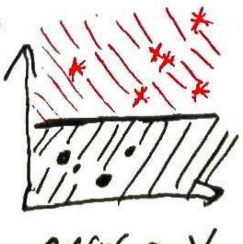
Here our validation data suggests the isolated • point is misleading, and the error rates on V suggest we should prune at node v

T on V



error on V $= 2/10$

T' on V



error on V $= 0/10$

By improving our validation error, we have reason to believe that this pruning will also improve the true error, since the validation error is a measure of how well this tree performs on some unseen data drawn from D.

Let's take a moment to compare our ID3 learning algorithm with K-NN

ID3 with pruning	K-NN
• longer training time (evaluate all candidate decisions at every node)	• short training procedure (memorize training data)
• shorter prediction run time (just answer some simple questions at each node as we descend the tree)	• long prediction run time (compute all distances...) scales poorly with # training points and # dimensions.
• unpruned tree gets zero training error	• 1-NN gets zero training error
• we can prune to avoid overfitting. Use validation data to select when and where to prune	• we can increase k to avoid overfitting. Use validation data to select K
• storing the model is just storing a tree of decisions	• storing the model involves storing all of the training dat
• somewhat intepretable, we can look at the tree to see what was found to be most informative. We can also look at regions of its decision boundary	• not very interpretable, but we could try to look at regions of its decision boundary.
• a little trickier to implement	• simple to implement

When might ID3 be better than K-NN? When might K-NN be better?

If we did not care about run time or storage costs, how might we use data to tell us which is a better choice for a certain task?