STA 314: Statistical Methods for Machine Learning I

Lecture 2 - Decision Trees

Chris J. Maddison

University of Toronto

arg min & arg max

- Given a function $f : \mathbb{R}^d \to \mathbb{R}$, we may want its minimum point, i.e., the point $x^\star \in \mathbb{R}^d$ such that for all $x \in \mathbb{R}^d$

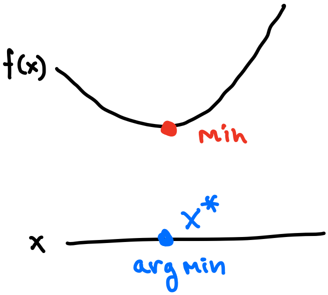
$$f(x^\star) \leq f(x)$$

- $\arg\min$ returns the minimum point,

$$x^\star = \arg\min_{x \in \mathbb{R}^d} f(x)$$

  $\arg\max$ returns the maximum point.

- $\arg\min_{x \in \mathbb{R}} (x - a)^2 = a$.
- If there is more than one minimum or maximum point, then the $\arg\min$ or $\arg\max$ are sets.



Last week

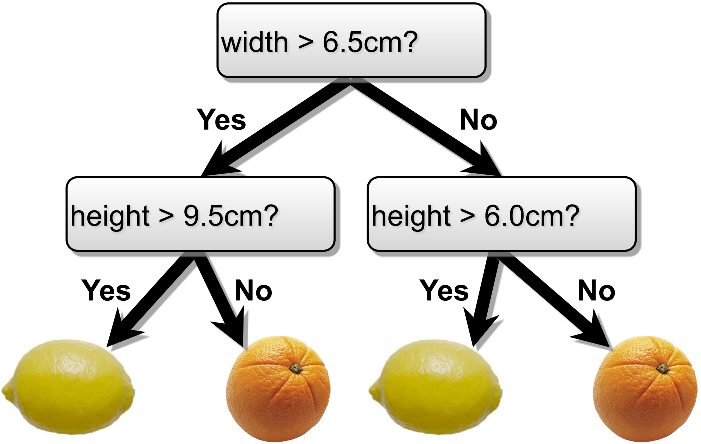
- Supervised learning
  - Given an input vector, learn to predict a label
- K-nearest neighbours
  - For a given test input, find the k nearest training inputs and output as your prediction the most common label among the neighbours.
  - The choice of distance metric has an impact on the performance (see HW1).
  - The properties of high dimensional data also has an impact on performance (see HW1).

Today

- Decision Trees
  - Simple but powerful learning algorithm
  - Used widely in Kaggle competitions
  - Lets us motivate concepts from information theory (entropy, mutual information, etc.)
- Loss functions and the question of generalization
  - We've been dancing around this question, let's formalize it a bit.

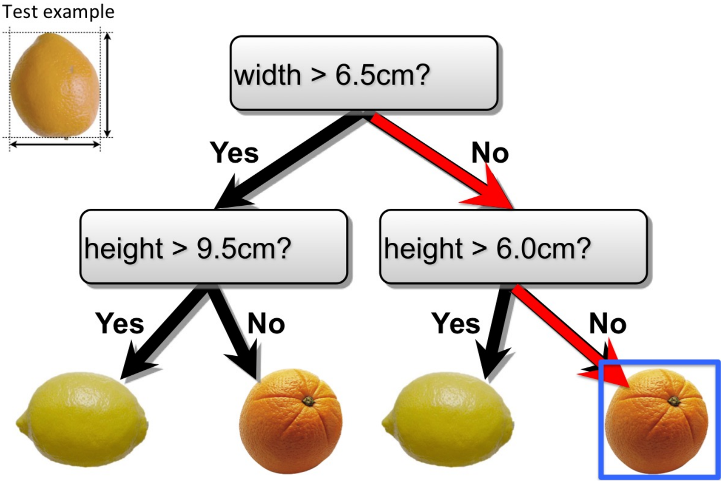
Decision Trees

- Make predictions by splitting on attributes according to a tree structure.



Decision Trees

- Make predictions by splitting on attributes according to a tree structure.



Decision Trees—Discrete attributes

First, what if attributes are discrete?

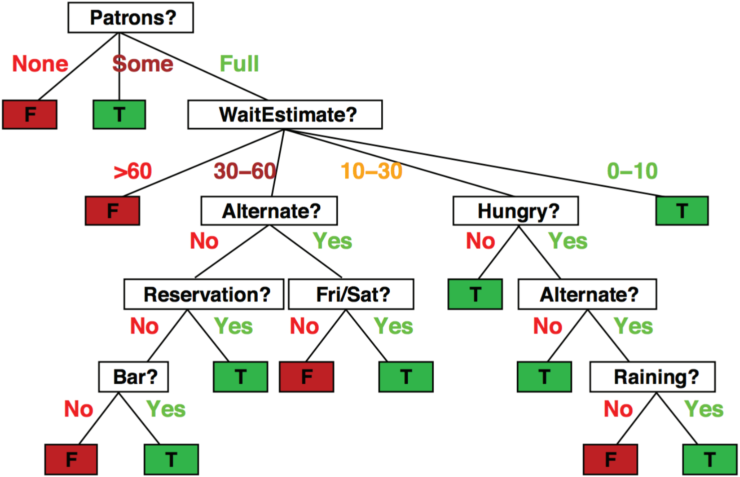
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

attributes:

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range ($, $$, $$$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

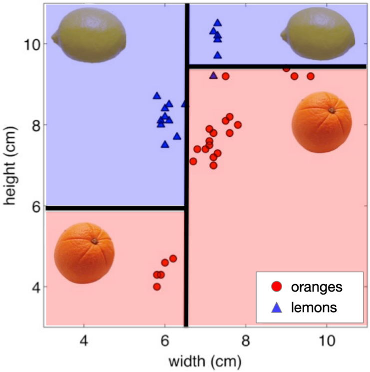
Decision Trees—Discrete attributes

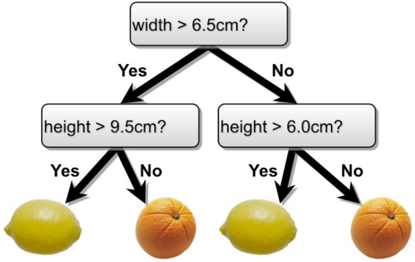
- Split *discrete attributes* into a partition of possible values.



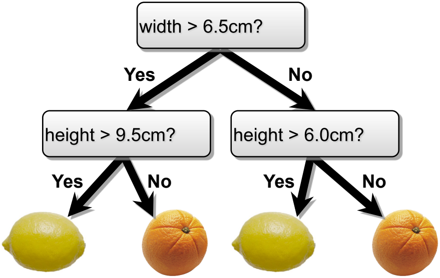
Decision Trees—Continuous attributes

- For *continuous attributes*, we partition the range by checking whether that attribute is greater than or less than some threshold.
- Decision boundary is made up of axis-aligned planes.





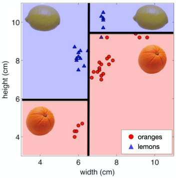
Decision Trees



- Internal nodes test a attribute, i.e., a dimension of the representation.
- Branching is determined by the attribute value.
- Children of a node partition the range of the attribute from the parent.
- Leaf nodes are outputs (predictions).

Decision Trees—Classification and Regression

- Each path from root to a leaf defines a region R_m of input space
- Let $\{(x^{(m_1)}, t^{(m_1)}), \ldots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m



- Classification tree (we will focus on this):
  - discrete output
  - leaf value y^m typically set to the most common value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$, i.e., majority vote
- Regression tree:
  - continuous output
  - leaf value y^m typically set to the mean value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$

Learning Decision Trees

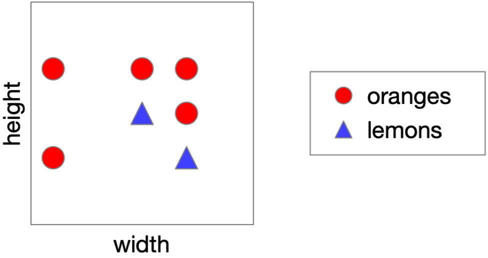
- For any training set we can construct a decision tree that has exactly one leaf for every training point, but it probably won't generalize.
  - Decision trees are universal function approximators.
- But, finding the smallest decision tree that correctly classifies a training set is computationally challenging.
  - If you are interested, check: Hyafil & Rivest'76.
- So, how do we construct a useful decision tree?

Learning Decision Trees

- Resort to a greedy heuristic:
  - Start with the whole training set and an empty decision tree, i.e., a tree with no internal nodes.
  - Pick a attribute and candidate split that would most reduce a measurement of loss.
  - Split on that attribute and recurse on subpartitions.
- Which loss should we use?
  - Let's see if misclassification rate is a good loss.

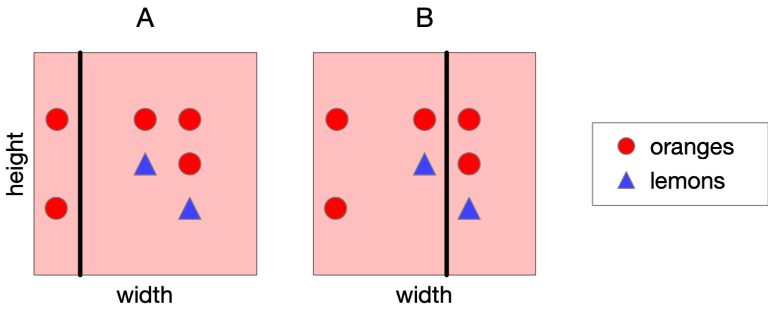
Choosing a Good Split

- Consider the following data. Let's split on width.



Choosing a Good Split

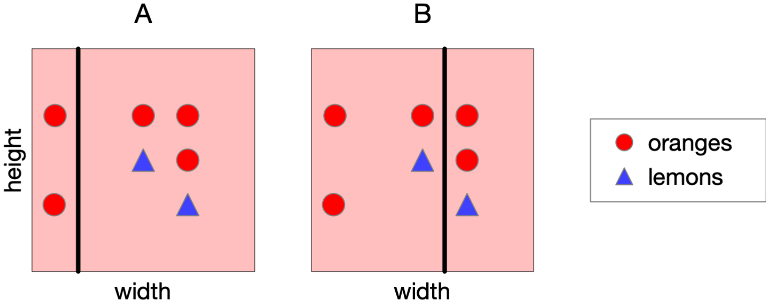
- Recall: classify by majority.



- A and B have the same misclassification rate, so which is the best split? Vote!

Choosing a Good Split

- A feels like a better split, because the left-hand region is very certain about whether the fruit is an orange.



- Can we quantify this?

Choosing a Good Split

- How can we quantify uncertainty in prediction for a given leaf node?
  - If all examples in leaf have same class: good, low uncertainty
  - If each class has same amount of examples in leaf: bad, high uncertainty
- **Idea:** Use counts at leaves to define probability distributions; use a probabilistic notion of uncertainty to decide splits.
- A brief detour through information theory...

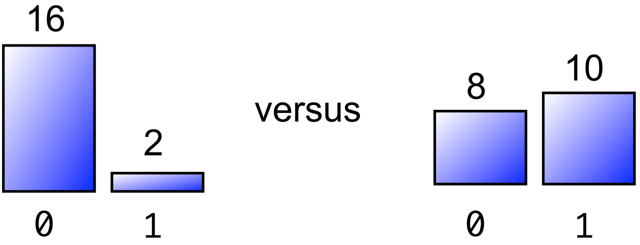
Quantifying Uncertainty

- The entropy of a discrete random variable is a number that quantifies the uncertainty inherent in its possible outcomes.
- The mathematical definition of entropy that we give in a few slides may seem arbitrary, but it can be motivated axiomatically.
  - If you're interested, check: *Information Theory* by Robert Ash.
- To explain entropy, consider flipping two different coins...

We Flip Two Different Coins

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

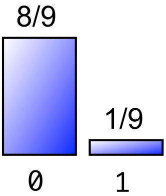
Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

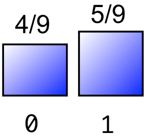


Quantifying Uncertainty

- The entropy of a loaded coin with probability p of heads is given by

$$-p\log_2(p) - (1-p)\log_2(1-p) \text{ for } 0 < p < 1, \text{ otherwise } 0$$

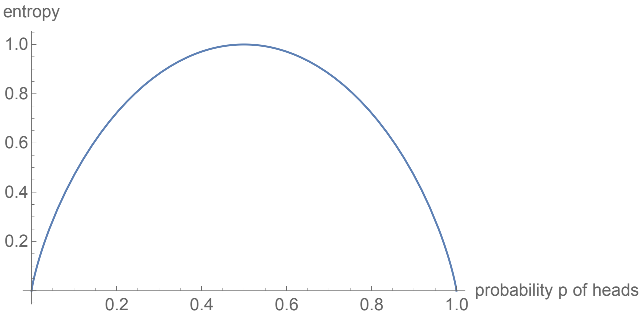




$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2} \qquad -\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- Notice: the coin whose outcomes are more certain has a lower entropy.
- In the extreme case $p = 0$ or $p = 1$, we were certain of the outcome before observing. So, we gained no certainty by observing it, i.e., entropy is 0.
- So, entropy can be seen as the amount of certainty that I gain when I see the outcome of a random variable.

Quantifying Uncertainty



- Claude Shannon showed: you cannot store the outcome of a random draw using fewer expected bits than the entropy without losing information.
- So units of entropy are bits; a fair coin flip has 1 bit of entropy.

Entropy

- More generally, the entropy of a discrete random variable Y is given by

$$H(Y) = -\sum_{y \in Y} p(y) \log_2 p(y)$$

  Interpret $p(y) \log_2 p(y) = 0$ if $p(y) = 0$.

- **"High Entropy"**:
  - Variable has a uniform like distribution over many outcomes
  - Flat histogram
  - Values sampled from it are less predictable
- **"Low Entropy"**
  - Distribution is concentrated on only a few outcomes
  - Histogram is concentrated in a few areas
  - Values sampled from it are more predictable
- To summarize: $H(Y)$ is the entropy of Y and it quantifies "how much uncertainty there is in Y".

Information

- Suppose we observe partial information X about a random variable Y
  - For example, $X = \mathrm{sign}(Y)$.
- We want to work to quantify the expected amount of information that will be conveyed about Y by observing X.
  - Or equivalently, the expected reduction in our uncertainty about Y after observing X.
- We will work towards a quantitative definition of information gain.
- To do this we define various notions of entropy.

Entropy of a Joint Distribution

- Example:
  $X = \{\text{Raining, Not raining}\}$,
  $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

$$\begin{aligned} H(X,Y) &= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_2 p(x,y) \\ &= -\frac{24}{100}\log_2\frac{24}{100} - \frac{1}{100}\log_2\frac{1}{100} - \frac{25}{100}\log_2\frac{25}{100} - \frac{50}{100}\log_2\frac{50}{100} \\ &\approx 1.56\text{bits} \end{aligned}$$

Specific Conditional Entropy

- Example:
  $X = \{\text{Raining, Not raining}\}$,
  $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- What is the entropy of cloudiness Y, given that it is raining?

$$
\begin{aligned}
H(Y|X=x) &= -\sum_{y \in Y} p(y|x) \log_2 p(y|x) \\
&= -\frac{24}{25}\log_2\frac{24}{25} - \frac{1}{25}\log_2\frac{1}{25} \approx 0.24\text{bits}
\end{aligned}
$$

- We used: $p(y|x) = \frac{p(x,y)}{p(x)}$, and $p(x) = \sum_y p(x,y)$ (sum in a row)

Conditional Entropy

- Example:
  $X = \{\text{Raining, Not raining}\}$,
  $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- The expected conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x) \end{aligned}$$

Conditional Entropy

- Example:
  $X = \{\text{Raining, Not raining}\}$,
  $Y = \{\text{Cloudy, Not cloudy}\}$

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- Entropy of cloudiness given the knowledge of whether or not it is raining?

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X=x) \\ &= \frac{1}{4} H(Y|\text{is raining}) + \frac{3}{4} H(Y|\text{not raining}) \\ &\approx 0.75 \text{ bits} \end{aligned}$$

Conditional Entropy

- Some useful properties:
  - Non-negative: $H(X) \geq 0$
  - Chain rule: $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
  - Independence: If X and Y independent, then X does not affect our uncertainty about Y: $H(Y|X) = H(Y)$
  - Knowing Y makes our knowledge of Y certain: $H(Y|Y) = 0$
  - Knowing X can only decrease uncertainty about Y: $H(Y|X) \leq H(Y)$

Information Gain

	Cloudy	Not Cloudy
Raining	24/100	1/100
Not Raining	25/100	50/100

- How much *more* certain am I about whether it's cloudy if I'm told whether it is raining?
  - My uncertainty in Y minus my expected uncertainty that would remain in Y after seeing X.
- This is the information gain $IG(Y, X)$ in Y due to X, or the mutual information of Y and X

$$IG(Y, X) = H(Y) - H(Y|X) \tag{1}$$

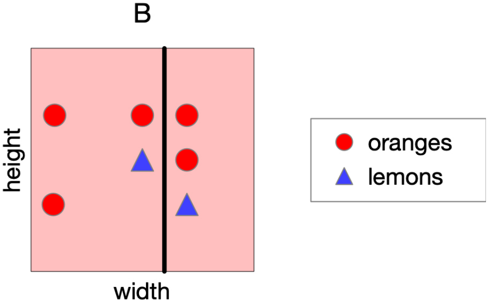
- If X is completely uninformative about Y: $IG(Y, X) = 0$
- If X is completely informative about Y: $IG(Y, X) = H(Y)$

Revisiting Our Original Example

- Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree split!
- The information gain of a split: how much information (over the training set) about the class label, $Y = \{red, blue\}$, is gained by knowing that you are considering data on one side of the split, $X = \{left, right\}$.

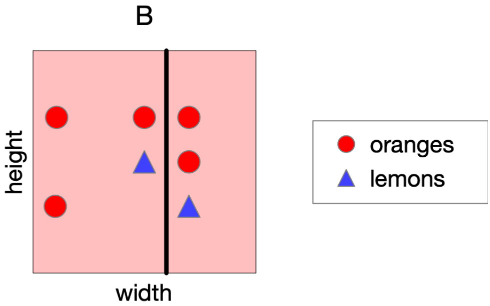
Revisiting Our Original Example

Let's compute $IG(Y, X)$ for example.



Revisiting Our Original Example

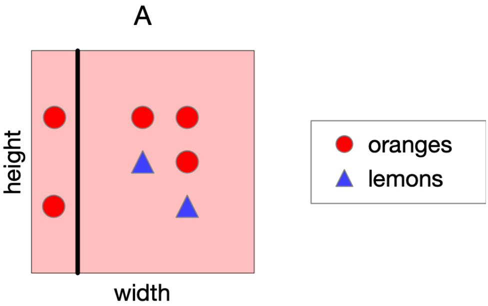
- What is the information gain of split B? Not terribly informative...



- Root entropy of class outcome: $H(Y) = -\frac{2}{7}\log_2(\frac{2}{7}) - \frac{5}{7}\log_2(\frac{5}{7}) \approx 0.86$
- Leaf conditional entropy of class outcome: $H(Y|X = \mathit{left}) \approx 0.81$, $H(Y|X = \mathit{right}) \approx 0.92$
- $IG(Y, X) \approx 0.86 - (\frac{4}{7} \cdot 0.81 + \frac{3}{7} \cdot 0.92) \approx 0.006$

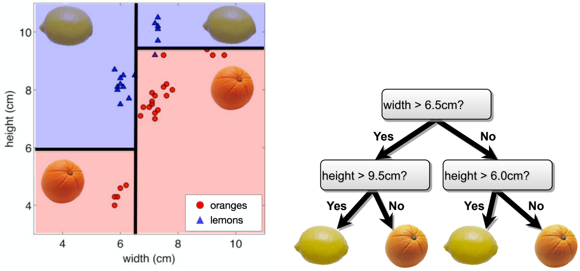
Revisiting Our Original Example

- What is the information gain of split A? Very informative!



- Root entropy of class outcome: $H(Y) = -\frac{2}{7}\log_2(\frac{2}{7}) - \frac{5}{7}\log_2(\frac{5}{7}) \approx 0.86$
- Leaf conditional entropy of class outcome: $H(Y|X = \textit{left}) = 0$, $H(Y|X = \textit{right}) \approx 0.97$
- $IG(Y, X) \approx 0.86 - (\frac{2}{7} \cdot 0 + \frac{5}{7} \cdot 0.97) \approx 0.17$!!

Constructing Decision Trees



- At each level, one must choose:
  1. Which attribute to split.
  2. Possibly where to split it.
- Choose them based on how much information we would gain from the decision! (choose attribute that gives the highest gain)

Decision Tree Construction Algorithm

- Simple, greedy, recursive approach, builds up tree node-by-node
  1. pick a attribute to split at a non-terminal node
  2. split examples into groups based on attribute value
  3. for each group:
     - if no examples – return majority from parent
     - else if all examples in same class – return class
     - else loop to step 1
- Terminates when all leaves contain only examples in the same class or are empty.

Back to Our Example

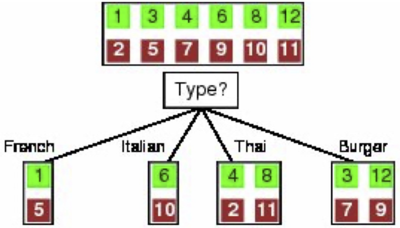
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

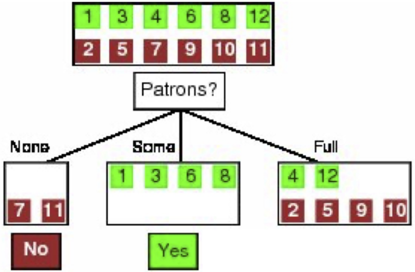
attributes:

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range ($, $$, $$$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

[from: Russell & Norvig]

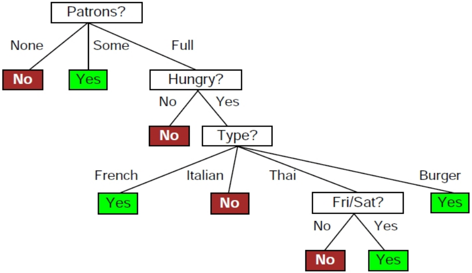
attribute Selection

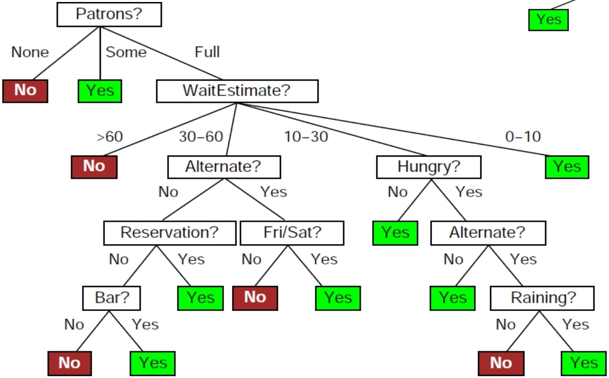




$$
\begin{aligned}
IG(Type, Y) &= 1 - \left[\frac{2}{12}H(Y|\text{Fr.}) + \frac{2}{12}H(Y|\text{It.}) + \frac{4}{12}H(Y|\text{Thai}) + \frac{4}{12}H(Y|\text{Bur.})\right] \\
&= 0 \\
IG(Patron, Y) &= 1 - \left[\frac{2}{12}H(Y|\text{None}) + \frac{4}{12}H(Y|\text{Some}) + \frac{6}{12}H(Y|\text{Full})\right] \approx 0.541
\end{aligned}
$$

Which Tree is Better? Vote!





What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
  - Human interpretability
- "Occam's Razor": find the simplest hypothesis that fits the observations
  - Useful principle, but hard to formalize (how to define simplicity?)
  - See Domingos, 1999, "The role of Occam's razor in knowledge discovery"
- We desire small trees with informative nodes near the root

Decision Tree Miscellany

- Problems:
  - You have exponentially less data at lower levels
  - Too big of a tree can overfit the data
  - Greedy algorithms don't necessarily yield the global optimum
- Handling continuous attributes
  - Split based on a threshold, chosen to maximize information gain
- Decision trees can also be used for regression on real-valued outputs. Choose splits to minimize squared error, rather than maximize information gain.

Comparison to k-NN

Advantages of decision trees over k-NN

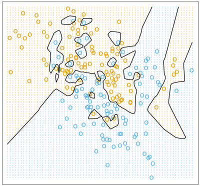
- Good when there are lots of attributes, but only a few are important
- Good with discrete attributes
- Easily deals with missing values (just treat as another value)
- Robust to scale of inputs
- Fast at test time
- More interpretable

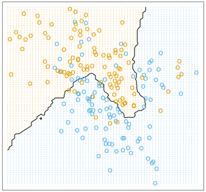
Advantages of k-NN over decision trees

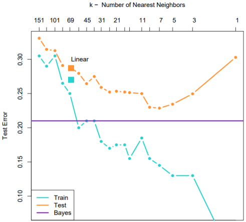
- Few hyperparameters
- Able to handle attributes/features that interact in complex ways (e.g. pixels)
- Can incorporate interesting distance measures (e.g. shape contexts)
- Typically make better predictions in practice

- Today, we deepen our understanding of generalization.
  - This will help us understand how to combine classifiers to get better performance (ensembling methods).

- Recall that we said that overly simple learning algorithms underfit the data, and overly complex ones overfit.







- Today we will be a bit more precise about what this means and what the goal of supervised learning is in general.

Loss Functions

- Given an input-label pair (x, t), a loss function $L(y, t)$ defines how bad it is if the algorithm predicts y.
- Example: 0-1 loss for classification

$$L_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$

  - Average 0-1 loss gives the error rate.
- Example: squared error loss for regression

$$L_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2$$

  - The average squared error loss is called mean squared error (MSE).
- Let's focus on 0-1 loss with inputs $\mathbf{x} \in \mathbb{R}^d$ and labels $t \in \{0, 1\}$.

Loss Functions

- Both k-NN and decision trees make predictions for all queries $\mathbf{x}$.
- We can think of the predictions of our learning algorithm forming a mapping $y : \mathbb{R}^d \to \{0, 1\}$ that we call a predictor.
- For a random data point drawn $(\mathbf{x}, t) \sim p_{\text{data}}$ from some data generating distribution, we can measure the expected error for the predictor y:

$$\mathcal{R}[y] := \sum_{t \in \{0,1\}} \int L_{0-1}(y(\mathbf{x}), t) p_{\text{data}}(\mathbf{x}, t) \; d\mathbf{x}$$

- For a finite data set $\mathcal{D} = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$, we can measure the average error:

$$\hat{\mathcal{R}}[y, \mathcal{D}] := \frac{1}{N} \sum_{i=1}^{N} L_{0-1}(y(\mathbf{x}^{(i)}), t^{(i)})$$

Goal of Supervised Learning

- The goal of supervised learning is to find a predictor y that achieves the lowest expected loss.

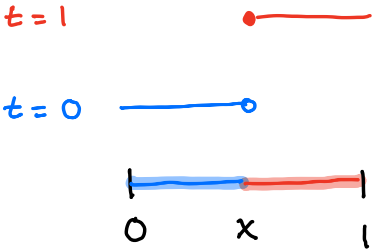
$$y^* = \arg \min_{y:\mathbb{R}^d \to \{0,1\}} \mathcal{R}[y]$$

  - If we're performing regression, we will optimize over $y : \mathbb{R}^d \to \mathbb{R}$.
  - If we're performing classification, we will optimize over $y : \mathbb{R}^d \to \{1, \ldots, C\}$.

Example

$$x \sim \text{uniform}[0, 1]$$

$$t(x) = \begin{cases} 0 & \text{if } x < 0.5 \\ 1 & \text{if } x \geq 0.5 \end{cases}$$



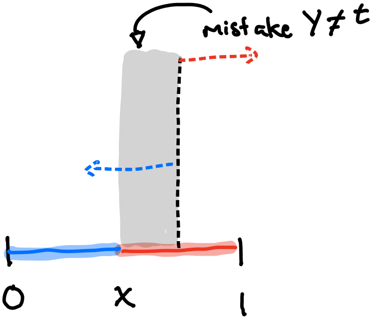
Example

$$x \sim \text{uniform}[0,1]$$

$$t(x) = \begin{cases} 0 & \text{if } x < 0.5 \\ 1 & \text{if } x \geq 0.5 \end{cases}$$

What is the expected error?

$$y(x) = \begin{cases} 0 & \text{if } x < 0.75 \\ 1 & \text{if } x \geq 0.75 \end{cases} \tag{2}$$



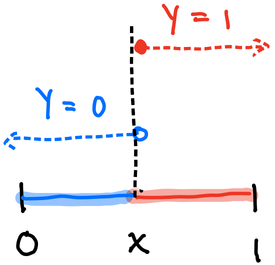
Example

$$x \sim \text{uniform}[0, 1]$$

$$t(x) = \begin{cases} 0 & \text{if } x < 0.5 \\ 1 & \text{if } x \geq 0.5 \end{cases}$$

$$y^\star(x) = t(x)$$

Opt. predictor is $y^\star = t$.



Supervised Learning in practice

- y is taken from a more restricted set of functions $\mathcal{H} \subset \{y : \mathbb{R}^d \to \{0, 1\}\}$ called a hypothesis space.
  - $\mathcal{H}$ may correspond to the set of all decisions boundaries that can be represented by a k-NN algorithm.
  - $\mathcal{H}$ may correspond to the set of all decisions boundaries that can be represented by a decision tree.
- We have a training set $\mathcal{D}_{\text{train}} = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$, which we assume to be independent and identically distributed (i.i.d.) draws from p_{data}.

Supervised Learning in practice

- Pick y by minimizing the loss on the training set

$$\min_{y \in \mathcal{H}} \hat{\mathcal{R}}[y, \mathcal{D}_{\text{train}}] \to \hat{y}^\star$$

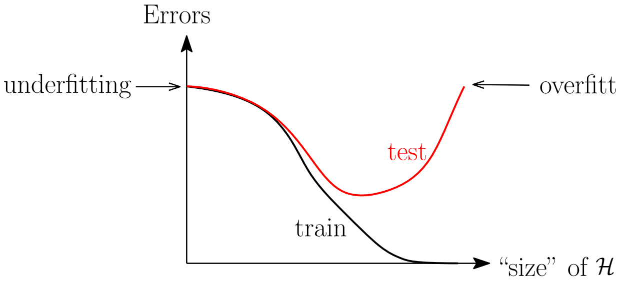
- But we really care about performance of $\hat{y}^\star$ in terms of expected loss.
- So, we measure its average error on an unseen test set $\mathcal{D}_{\text{test}} = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{M}$ i.i.d. p_{data} to approximate how well it does on the true data generating distribution,

$$\hat{\mathcal{R}}[\hat{y}^\star, \mathcal{D}_{\text{test}}] \approx \mathcal{R}[\hat{y}^\star]$$

- When we say that we want $\hat{y}^\star$ to generalize from the training set to the test set, we mean that we want $\mathcal{R}[\hat{y}^\star]$ to be as small as it can be.

Underfitting & Overfitting

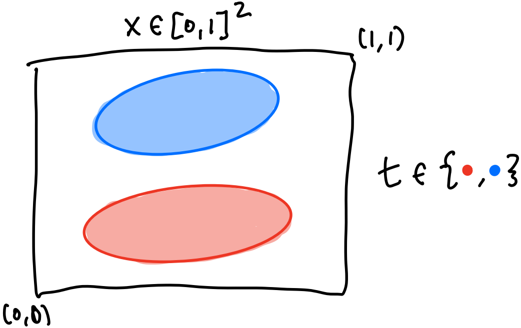
- This is the essence of supervised learning.
  - many open questions, depending on the choice of $\mathcal{H}$.
  - can study this problem as $N \to \infty$ or as $\mathcal{H}$ changes.
- Let's study this as $\mathcal{H}$ changes and return to underfitting and overfitting.



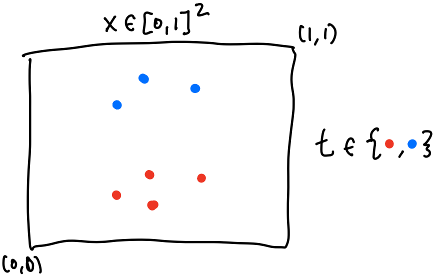
Source: Francis Bach. Learning Theory from First Principles.

2D Example

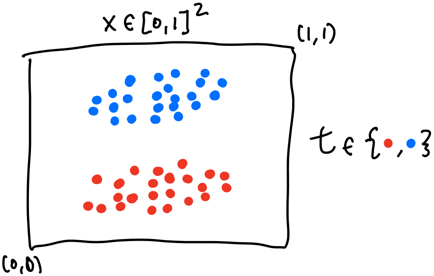
- $\mathbf{x}$ is uniform on the ellipses.
- $t \in \{\color{red}{\bullet}, \color{blue}{\bullet}\}$ depends on which ellipse $\mathbf{x}$ falls in



2D Example



Train Set

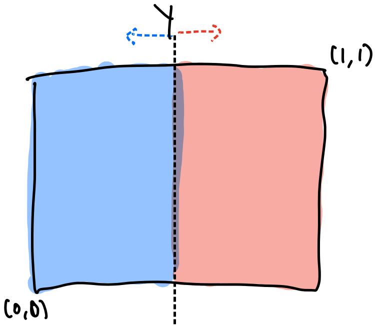


Test Set

2D Example

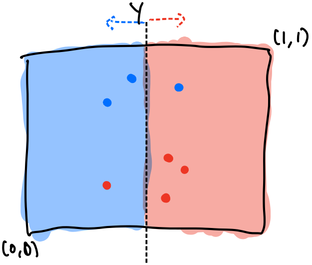
Let's consider a simple hypothesis class.

$\mathcal{H}$ = $\{y$ with vertical decision boundaries$\}$.

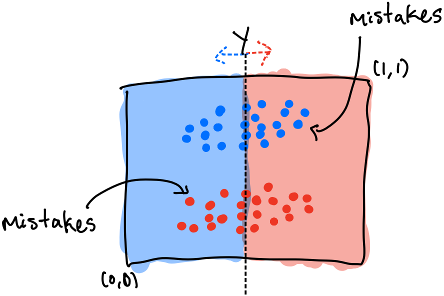


2D Example

Best predictor in terms of 0-1 loss on training set does poorly on both the training set and test set.



Train Set



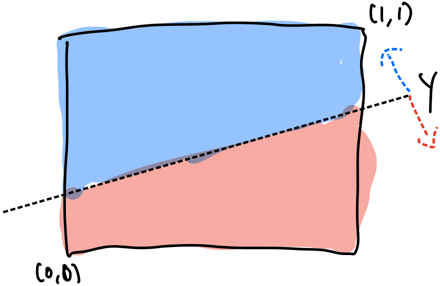
Test Set

This is underfitting.

2D Example

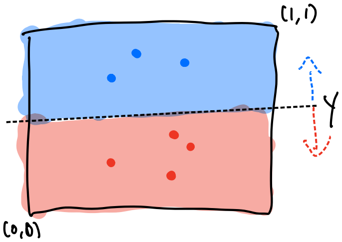
Let's consider a more complex hypothesis class.

$\mathcal{H} = \{y$ with linear decision boundaries$\}$.

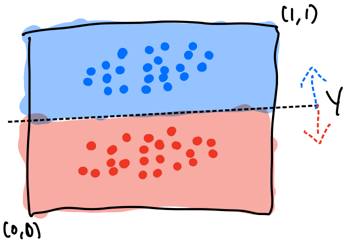


2D Example

Best predictor on training set does well on both the training set and test set.



Train Set



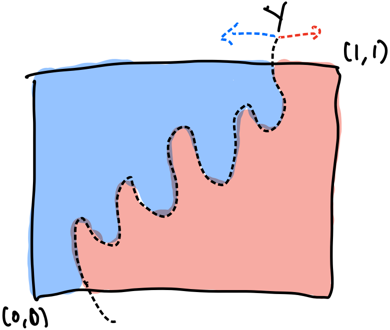
Test Set

This is well fit.

2D Example

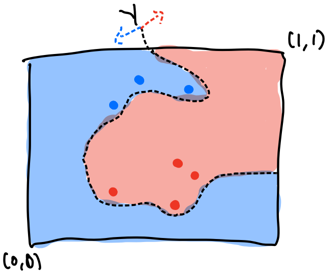
Let's consider a very complex hypothesis class.

$\mathcal{H}$ = $\{y$ with curved decision boundaries$\}$.

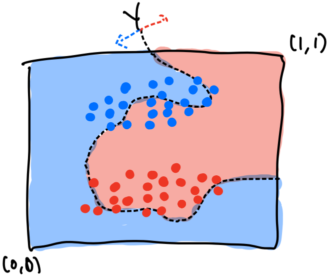


2D Example

Best predictor on training set does poorly on test set, but well on training set.



Train Set

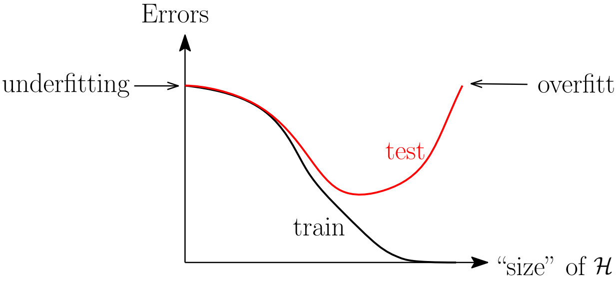


Test Set

This is overfitting.

Summary

- We have now talked about two hypothesis classes: k-NN and decision trees.
- We can understand supervised learning through the complexity of the hypothesis class.



Source: Francis Bach. Learning Theory from First Principles.