# INT305 Machine Learning

## Lecture 7

## Decision Trees & Bias-Variance Decomposition

Jimin Xiao

Department Intelligence Science

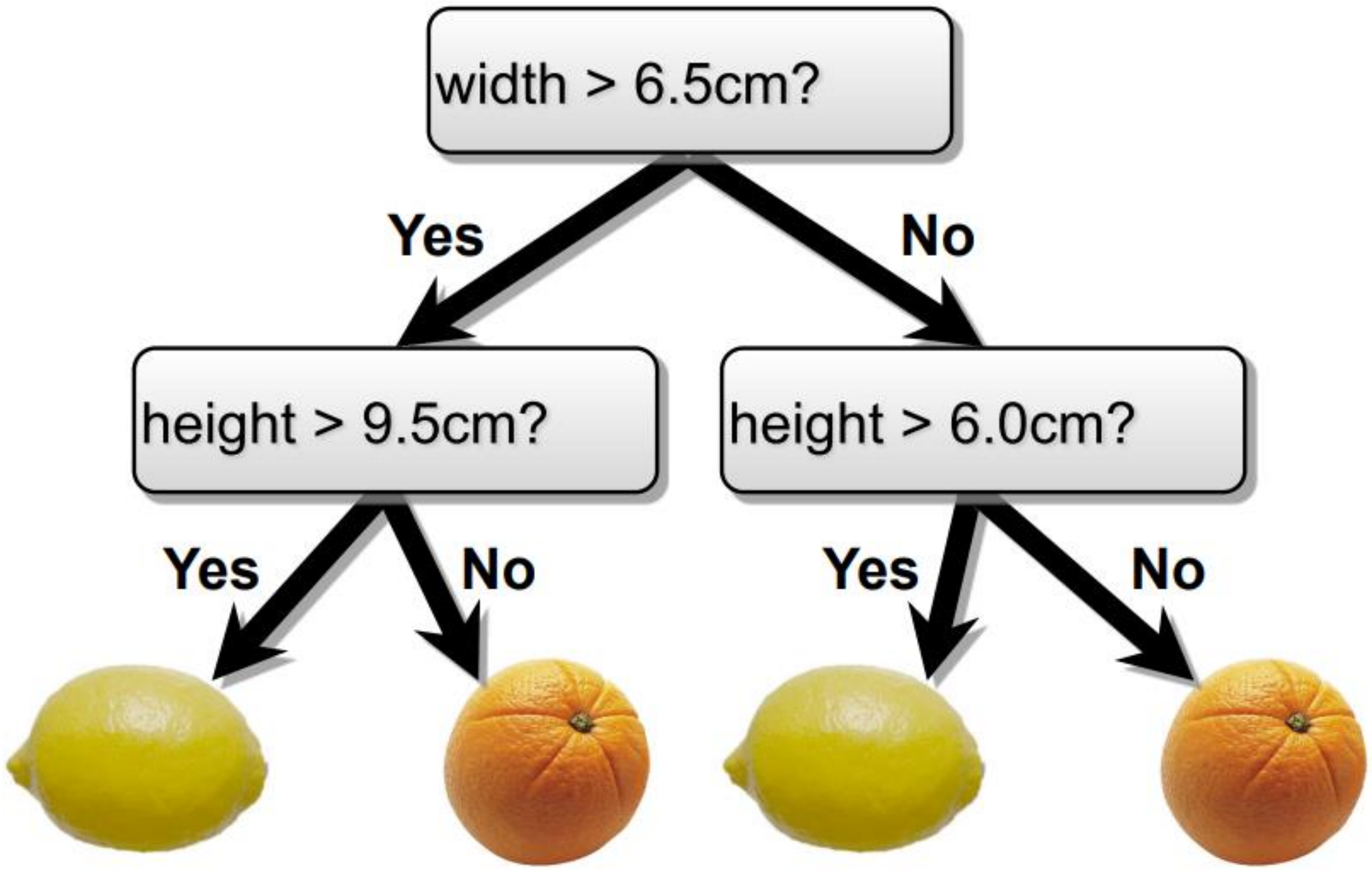
Jimin.xiao@xjtlu.edu.cn

# Today

- Decision Trees
  - Simple but powerful learning algorithm
  - Used widely in Kaggle competitions
  - Lets us motivate concepts from information theory (entropy, mutual information, etc.)
- Bias-variance decomposition
  - Lets us motivate methods for combining different classifiers.

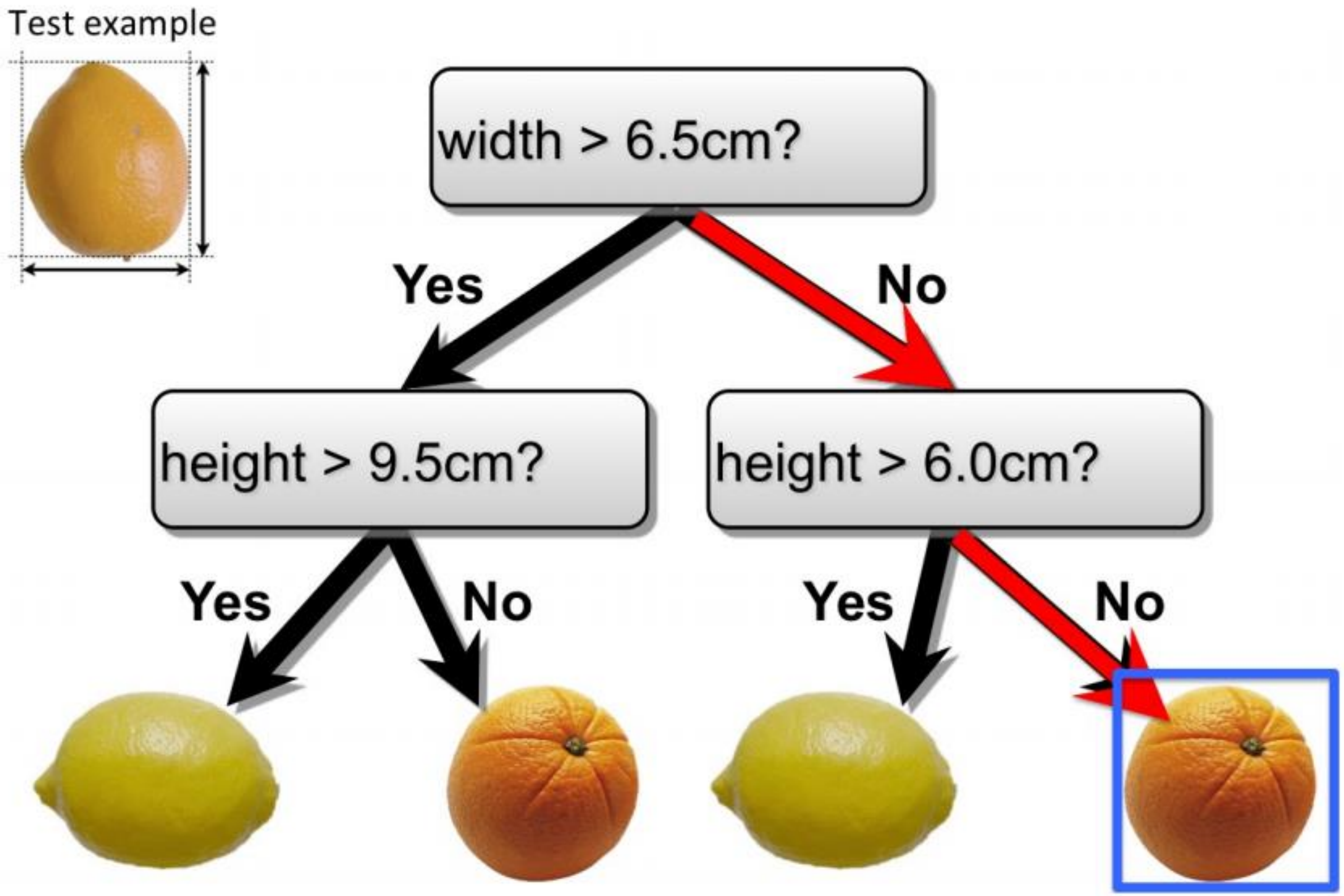
# Decision Trees

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



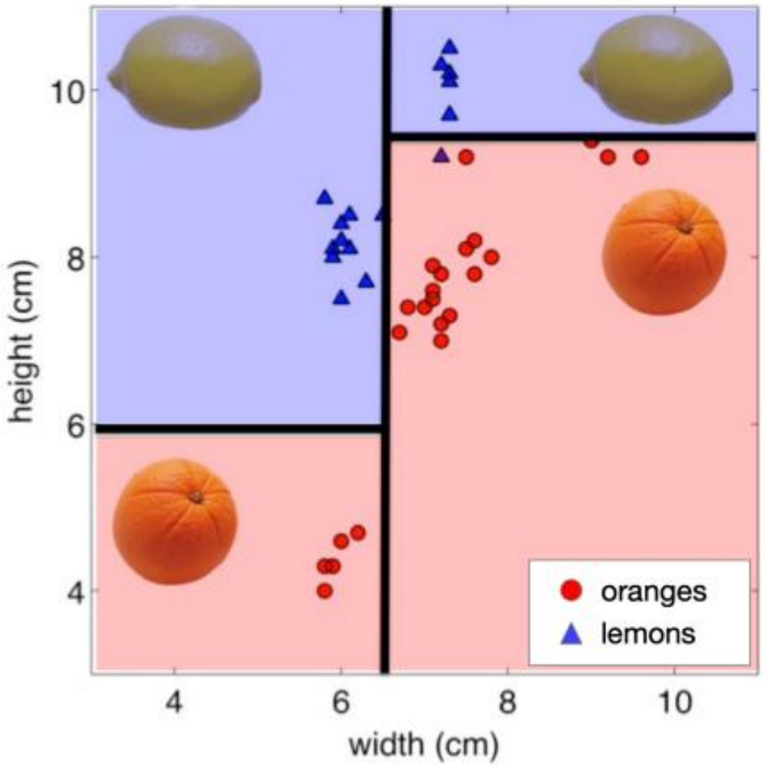
# Decision Trees

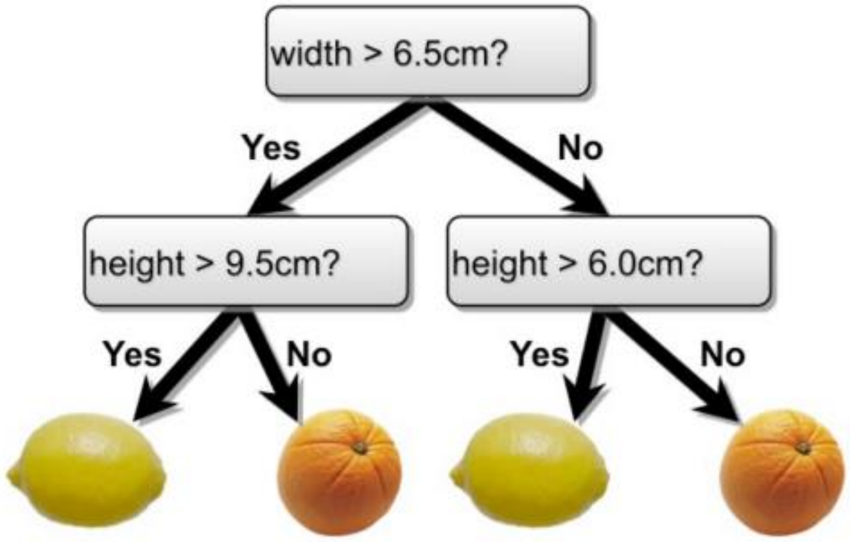
- Make predictions by splitting on features according to a tree structure.



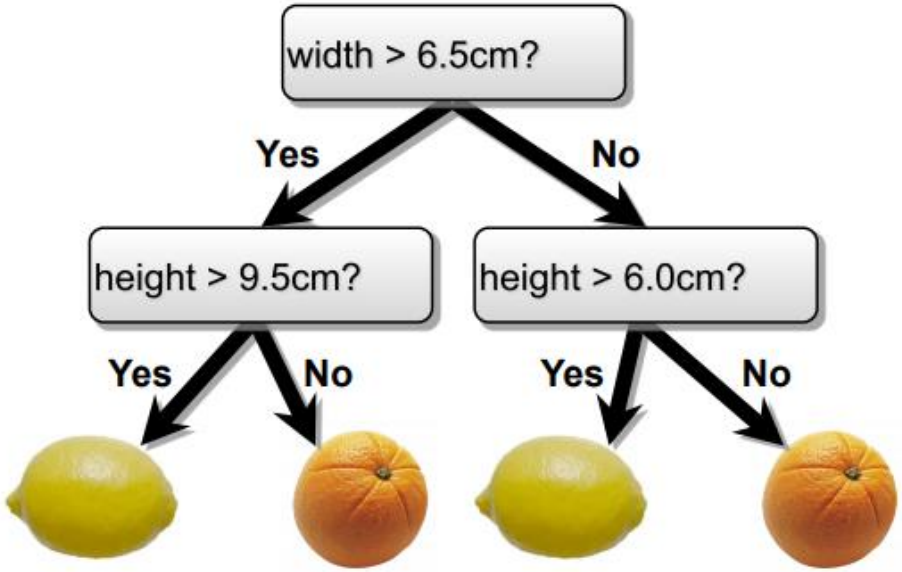
# Decision Trees— Continuous Features

- Split *continuous features* by checking whether that feature is greater than or less than some threshold.
- Decision boundary is made up of axis-aligned planes.



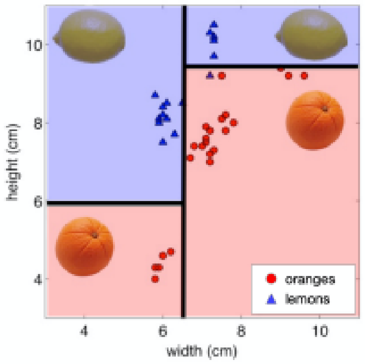



# Decision Trees




- Internal nodes test a feature
- Branching is determined by the feature value
- Leaf nodes are outputs (predictions)

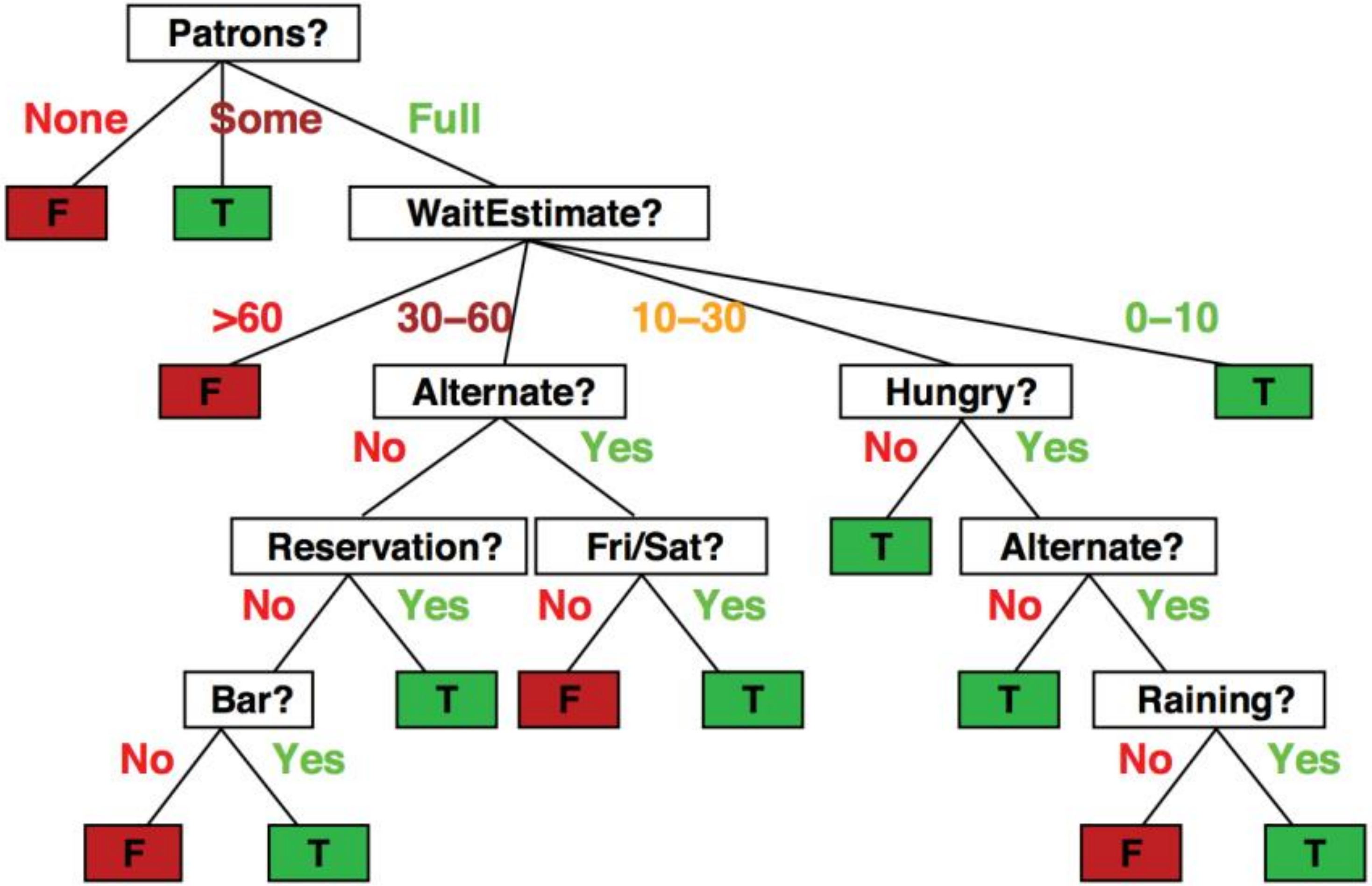
# Decision Trees—Classification and Regression




- 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 $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$
- Regression tree:
  - continuous output
  - leaf value $y^m$ typically set to the mean value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$

# Decision Trees—Discrete Features

- Will I eat at this restaurant?



# Decision Trees—Discrete Features

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

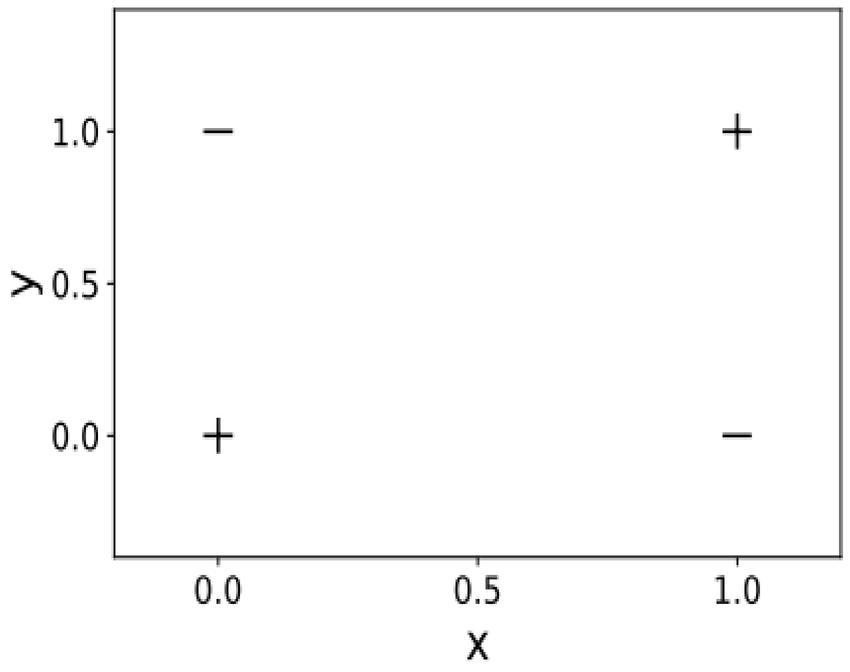
| Example | Input Attributes | | | | | | | | | | Goal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *WillWait* |
| $\mathbf{x}_1$ | Yes | No | No | Yes | Some | $$$ | No | Yes | French | 0–10 | $y_1 =$ Yes |
| $\mathbf{x}_2$ | Yes | No | No | Yes | Full | $ | No | No | Thai | 30–60 | $y_2 =$ No |
| $\mathbf{x}_3$ | No | Yes | No | No | Some | $ | No | No | Burger | 0–10 | $y_3 =$ Yes |
| $\mathbf{x}_4$ | Yes | No | Yes | Yes | Full | $ | Yes | No | Thai | 10–30 | $y_4 =$ Yes |
| $\mathbf{x}_5$ | Yes | No | Yes | No | Full | $$$ | 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 |

Features:

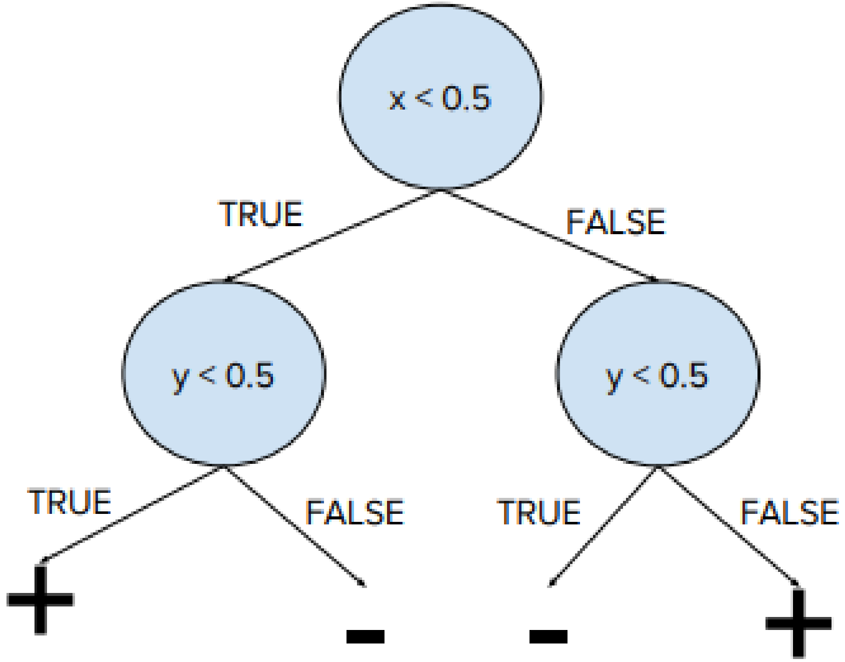
| | |
|---|---|
| 1. | Alternate: whether there is a suitable alternative restaurant nearby. |
| 2. | Bar: whether the restaurant has a comfortable bar area to wait in. |
| 3. | Fri/Sat: true on Fridays and Saturdays. |
| 4. | Hungry: whether we are hungry. |
| 5. | Patrons: how many people are in the 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). |

# Attempts

The drawing below shows a dataset. Each example in the dataset has two inputs features $x$ and $y$, and maybe classified as a positive example (labelled +) or a negative example (labelled −). Draw a decision tree which correctly classifies each example in the dataset.



# Attempts



# Learning Decision Trees

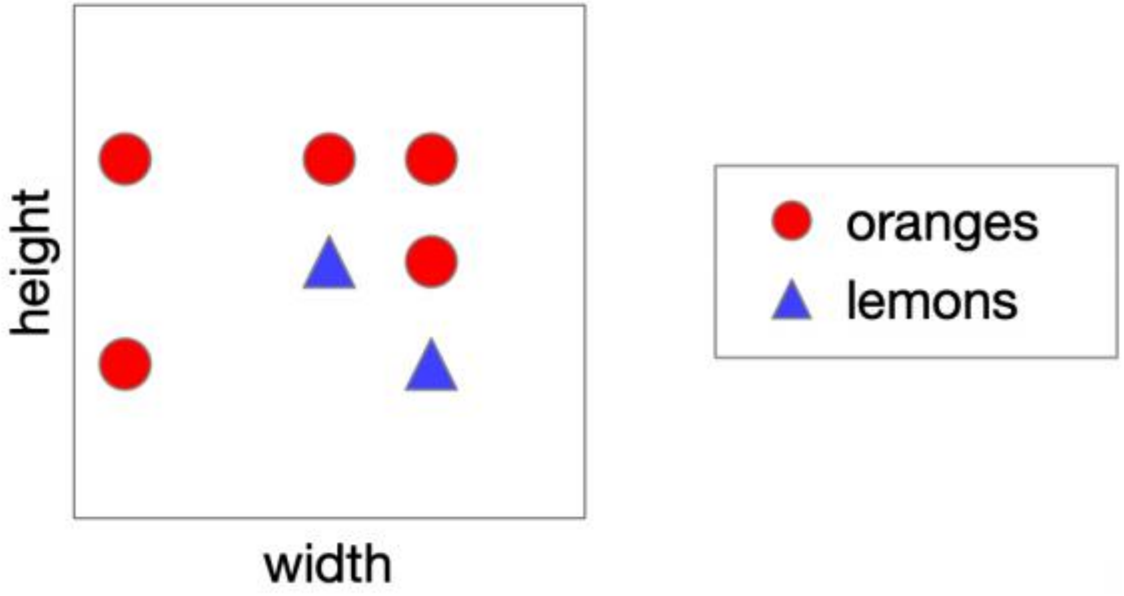
- For any training set we can construct a decision tree that has exactly the one leaf for every training point, but it probably won't generalize.
  - Decision trees are universal function approximators.
- But, finding the smallest decision tree that correctly classifies a training set is NP complete.
  - 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.
  - Pick a feature and candidate split that would most reduce the loss.
  - Split on that feature and recurse on subpartitions.
- Which loss should we use?
  - Let's see if misclassification rate is a good loss.

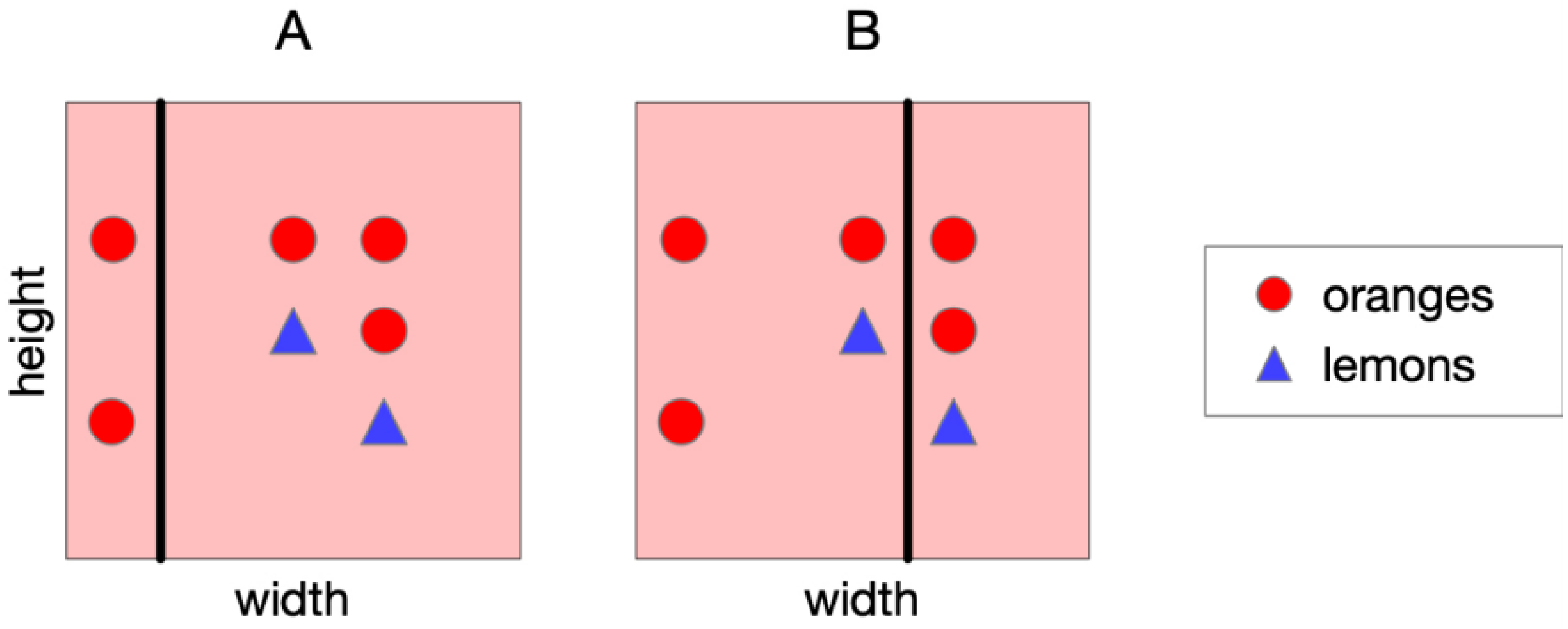
# Choosing a Good Split

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



# Choosing a Good Split

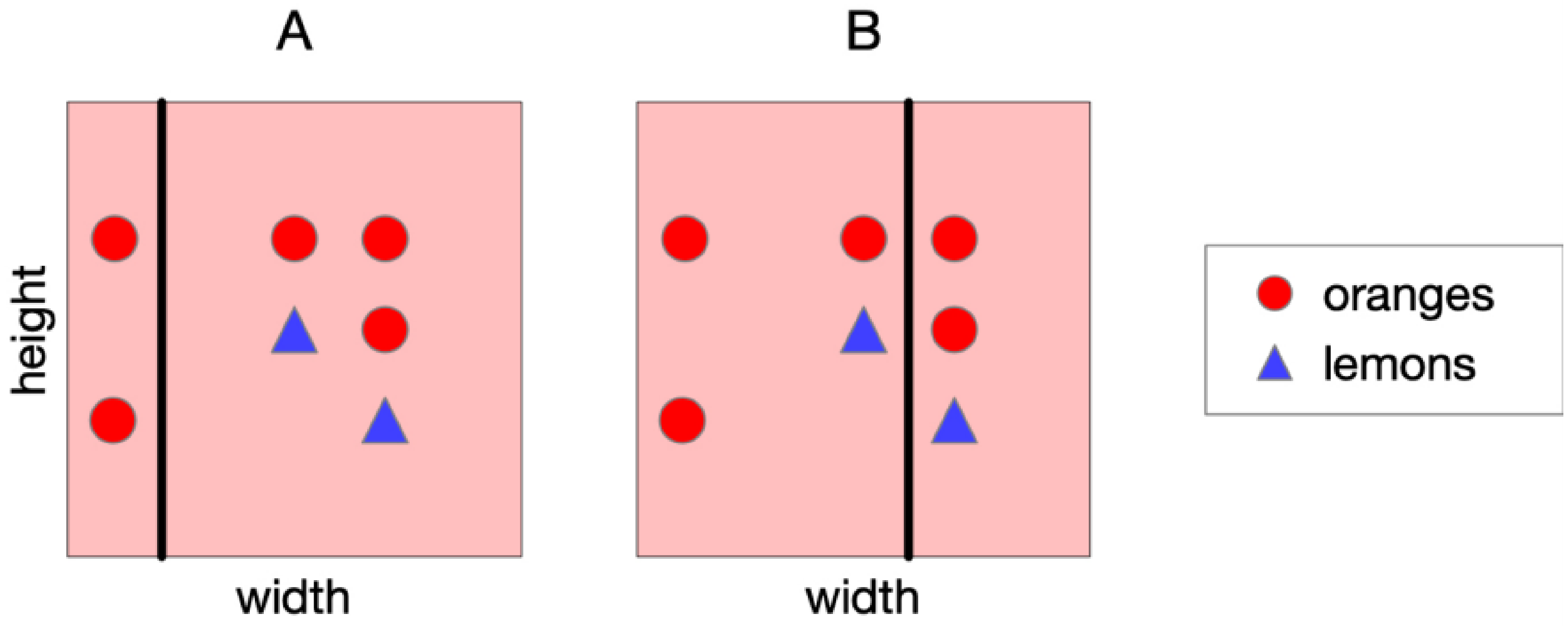
- Recall: classify by majority.




- 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...

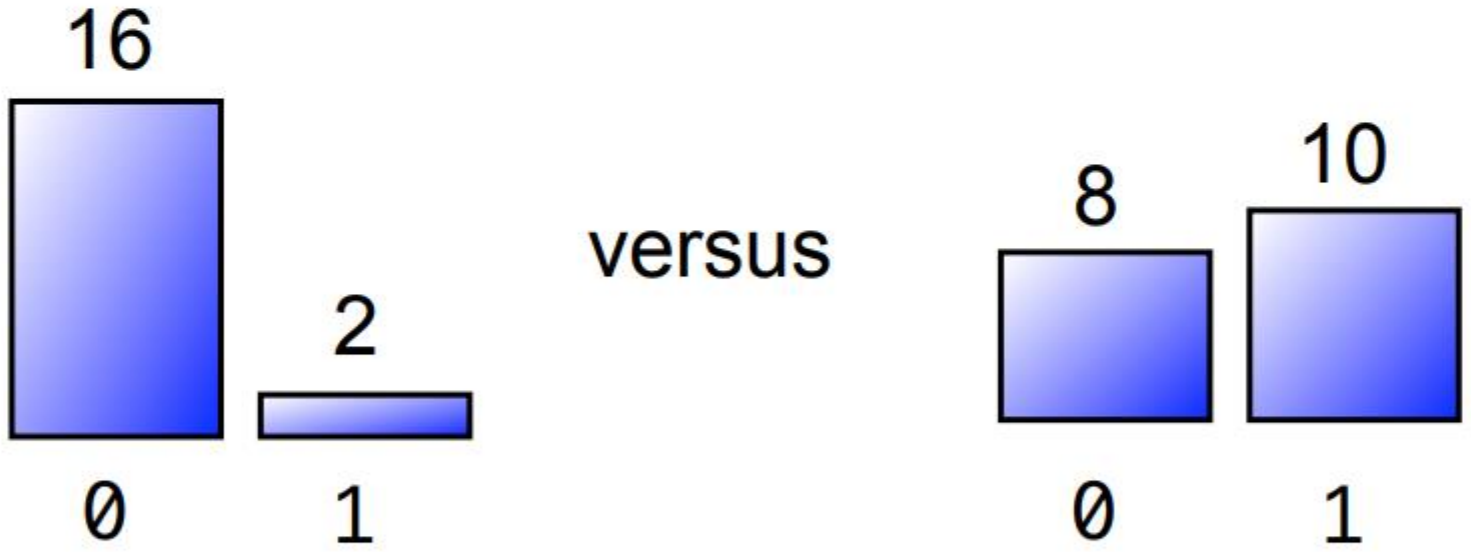
# We Flip Two Different Coins

Sequence 1:

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:

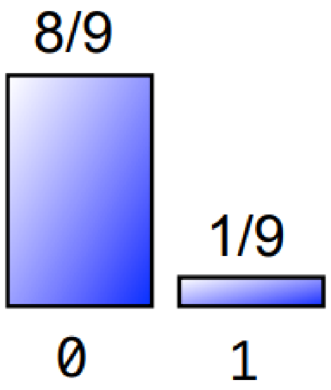
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

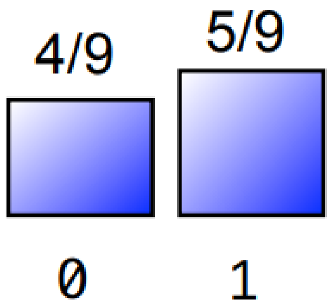


# Quantifying Uncertainty

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

$$-p\log_2(p) - (1-p)\log_2(1-p)$$



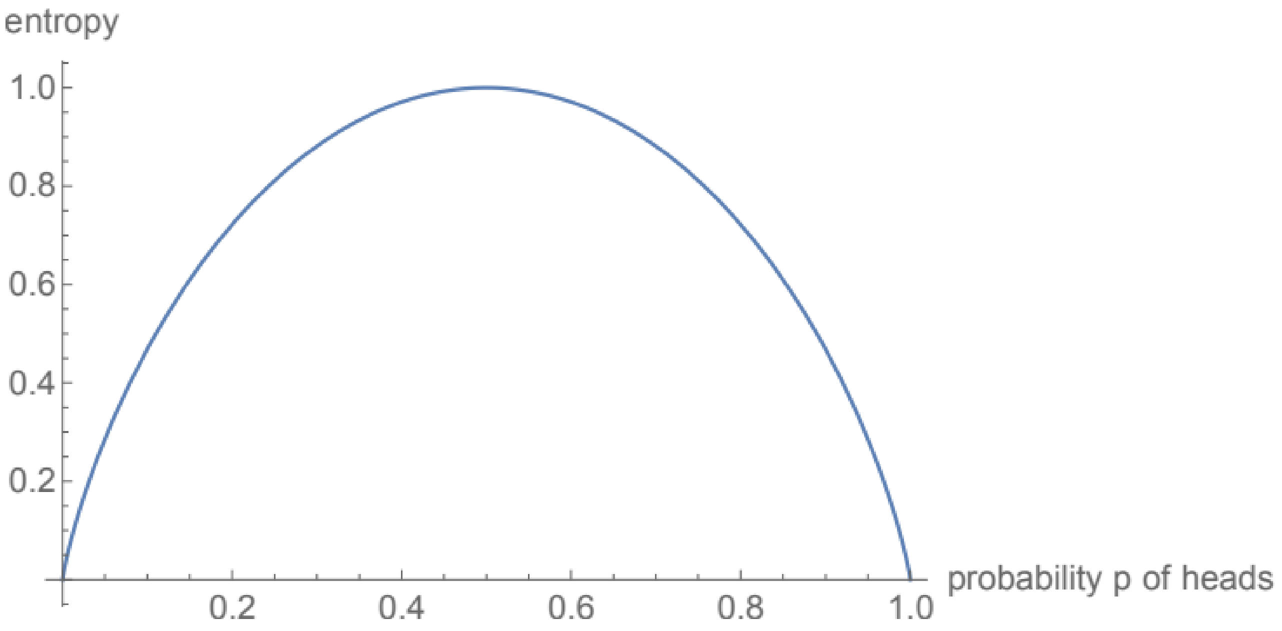

$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- 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.

# Quantifying Uncertainty

- Can also think of entropy as the expected information content of a random draw from a probability distribution.




- 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

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

- **"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

# Entropy

- Suppose we observe partial information $X$ about a random variable $Y$
  - For example, $X = \text{sign}(Y)$.
- We want to work towards a definition of 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 $X$.

# 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

| | 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 |

- What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

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

# Conditional Entropy

- Some useful properties:
  - $H$ is always non-negative
  - Chain rule: $H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
  - If $X$ and $Y$ independent, then $X$ does not affect our uncertainty about $Y$: $H(Y|X) = H(Y)$
  - But knowing $Y$ makes our knowledge of $Y$ certain: $H(Y|Y) = 0$
  - By knowing $X$, we 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 called the information gain $IG(Y|X)$ in $Y$ due to $X$, or the mutual information of $Y$ and $X$

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

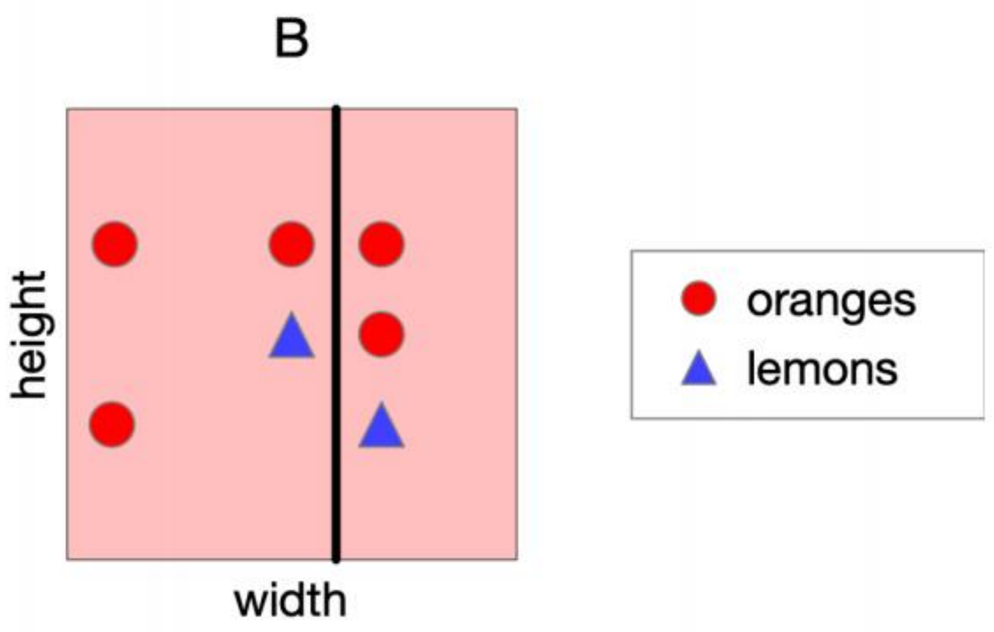
- If $X$ is completely uninformative about $Y$: $IG(Y|X) = 0$
- If $X$ is completely informative about $Y$: $IG(Y|X) = H(Y)$

# 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 $Y$ is gained by knowing which side of a split you're on.

# Revisiting Our Original Example

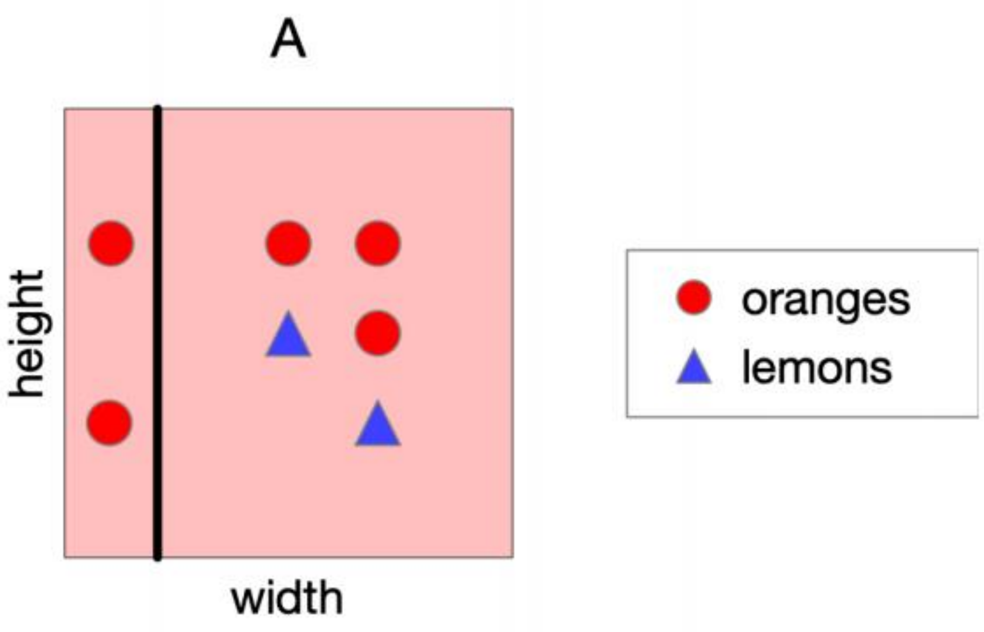
- What is the information gain of split B? Not terribly informative...




- 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: $H(Y|left) \approx 0.81$, $H(Y|right) \approx 0.92$
- $IG(split) \approx 0.86 - (\frac{4}{7}\cdot 0.81 + \frac{3}{7}\cdot 0.92) \approx 0.006$

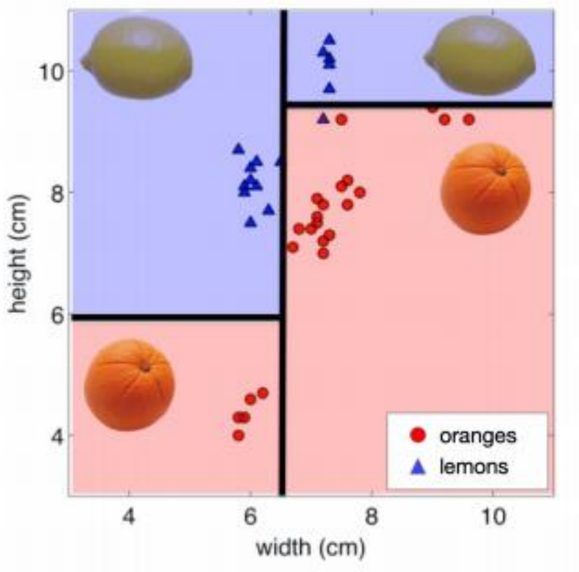
# Revisiting Our Original Example

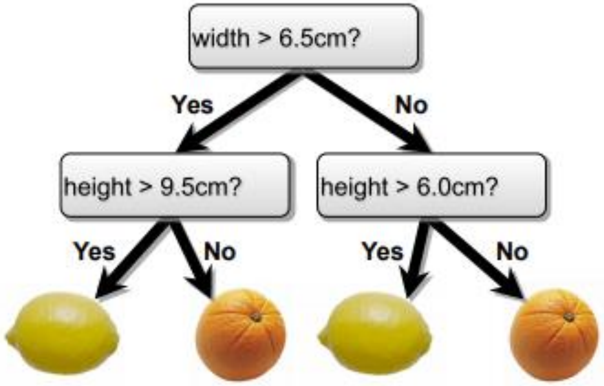
- What is the information gain of split A? Very informative!




- 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: $H(Y|left) = 0$, $H(Y|right) \approx 0.97$
- $IG(split) \approx 0.86 - (\frac{2}{7} \cdot 0 + \frac{5}{7} \cdot 0.97) \approx 0.17!!$

# Constructing Decision Trees




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

# Decision Tree Construction Algorithm

- Simple, greedy, recursive approach, builds up tree node-by-node
  1. pick a feature to split at a non-terminal node
  2. split examples into groups based on feature 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

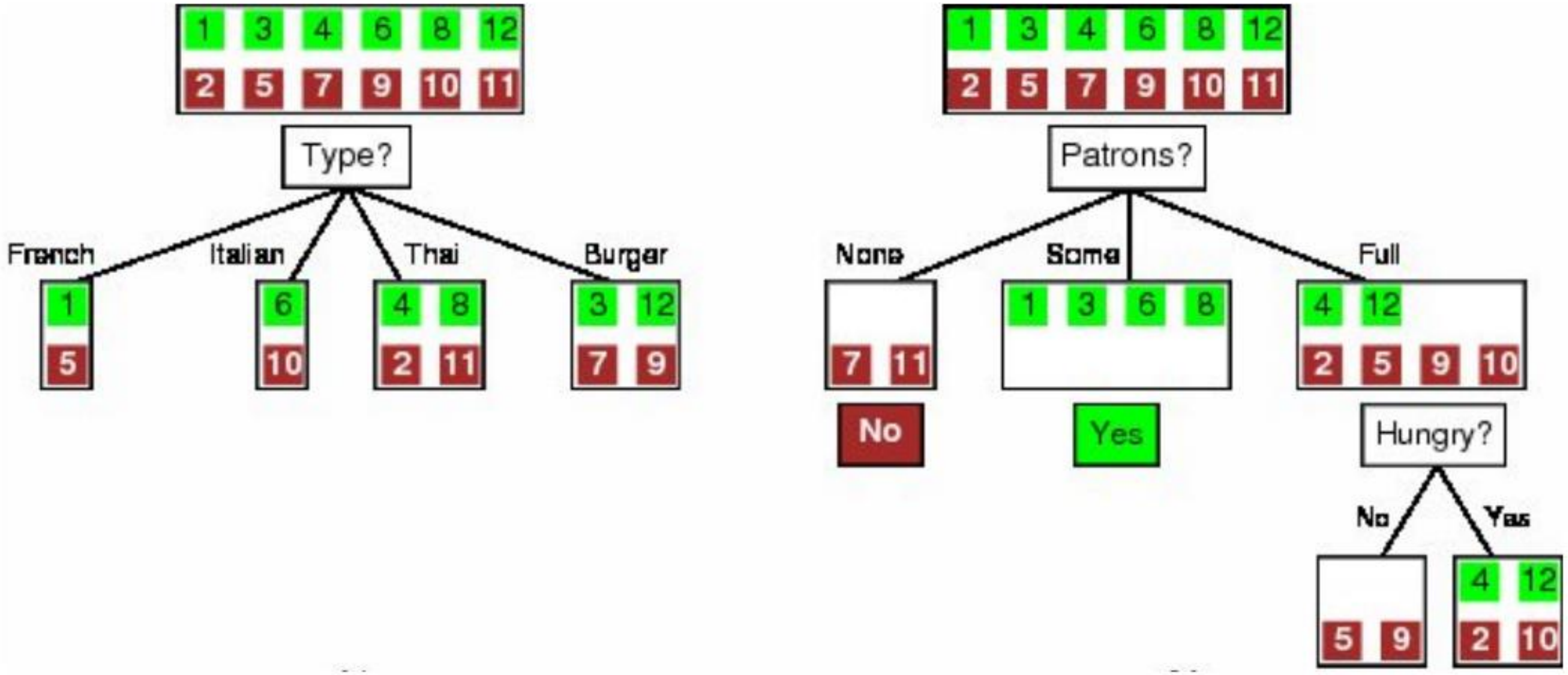
| Example | Input Attributes | | | | | | | | | | Goal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *WillWait* |
| $\mathbf{x}_1$ | Yes | No | No | Yes | Some | $$$ | No | Yes | French | 0–10 | $y_1 =$ Yes |
| $\mathbf{x}_2$ | Yes | No | No | Yes | Full | $ | No | No | Thai | 30–60 | $y_2 =$ No |
| $\mathbf{x}_3$ | No | Yes | No | No | Some | $ | No | No | Burger | 0–10 | $y_3 =$ Yes |
| $\mathbf{x}_4$ | Yes | No | Yes | Yes | Full | $ | Yes | No | Thai | 10–30 | $y_4 =$ Yes |
| $\mathbf{x}_5$ | Yes | No | Yes | No | Full | $$$ | 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 |

Features:

| | |
|---|---|
| 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]

# Feature Selection




$$IG(Y) = H(Y) - H(Y|X)$$

$$IG(type) = 1 - \left[\frac{2}{12}H(Y|\text{Fr.}) + \frac{2}{12}H(Y|\text{It.}) + \frac{4}{12}H(Y|\text{Thai}) + \frac{4}{12}H(Y|\text{Bur.})\right] = 0$$

$$IG(Patrons) = 1 - \left[\frac{2}{12}H(0,1) + \frac{4}{12}H(1,0) + \frac{6}{12}H(\frac{2}{6}, \frac{4}{6})\right] \approx 0.541$$

# 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

# Comparison to some other classifiers

Advantages of decision trees over KNNs and neural nets

- Simple to deal with discrete features, missing values, and poorly scaled data
- Fast at test time
- More interpretable

Advantages of KNNs over decision trees

- Few hyperparameters
- Can incorporate interesting distance measures (e.g. shape contexts)

Advantages of neural nets over decision trees

- Able to handle attributes/features that interact in very complex ways (e.g. pixels)

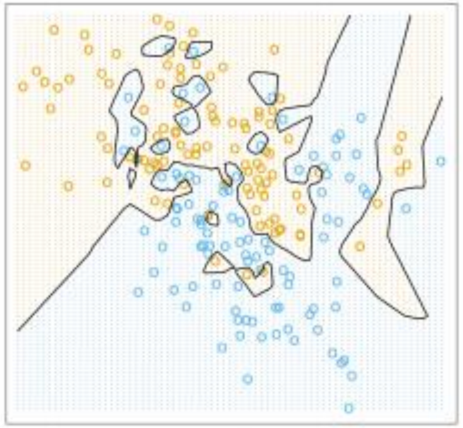
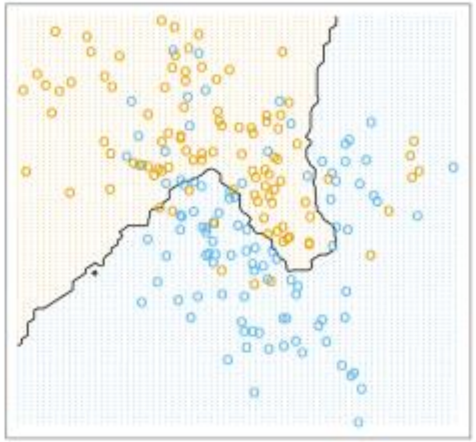
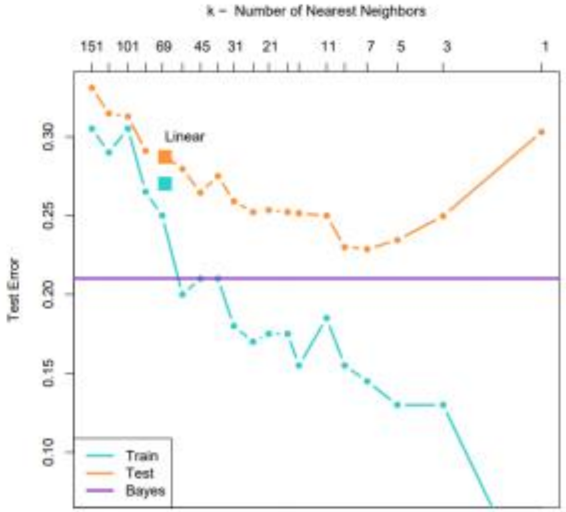
# Comparison to some other classifiers

- We've seen many classification algorithms.
- We can combine multiple classifiers into an ensemble, which is a set of predictors whose individual decisions are combined in some way to classify new examples
  - E.g., (possibly weighted) majority vote
- For this to be nontrivial, the classifiers must differ somehow, e.g.
  - Different algorithm
  - Different choice of hyperparameters
  - Trained on different data
  - Trained with different weighting of the training examples
- Next lecture, we will study some specific ensembling techniques.

- Today, we deepen our understanding of generalization through a bias-variance decomposition.
  - This will help us understand ensembling methods.

# Bias-Variance Decomposition

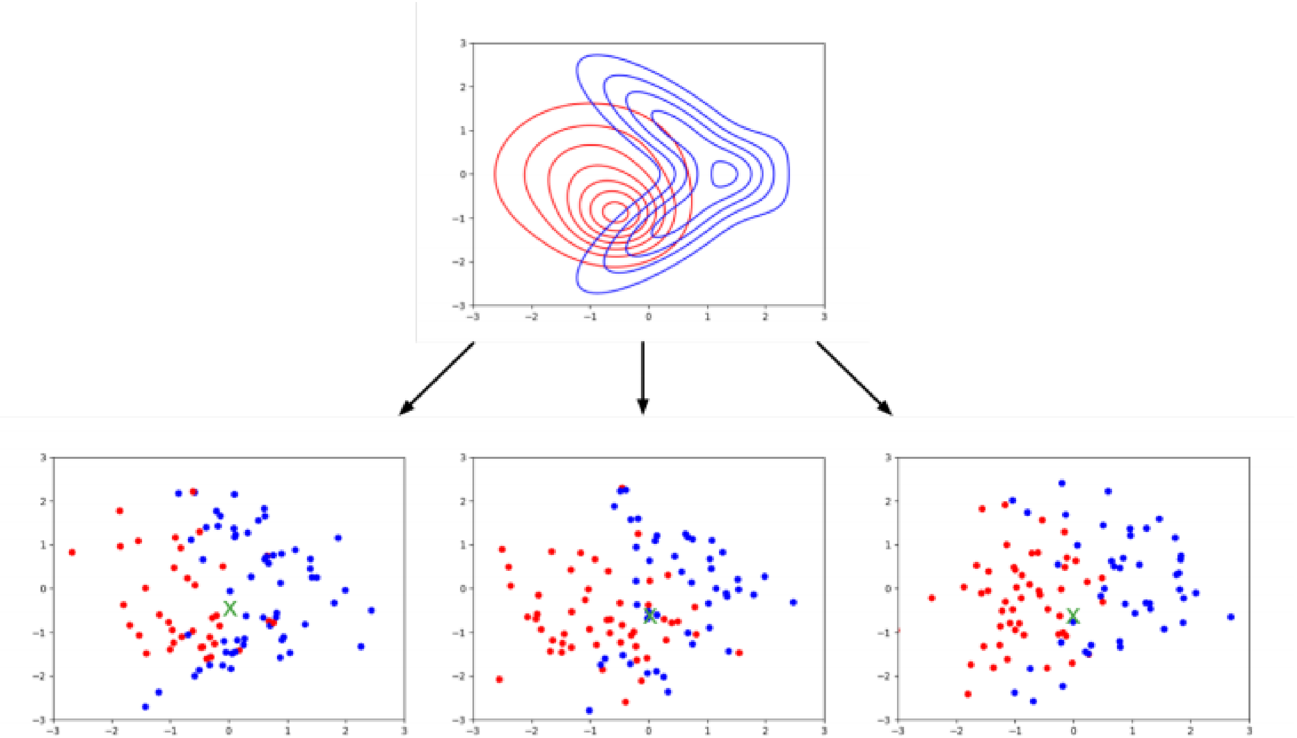
- Recall that overly simple models underfit the data, and overly complex models overfit.


- We can quantify this effect in terms of the bias/variance decomposition.
  - Bias and variance of what?

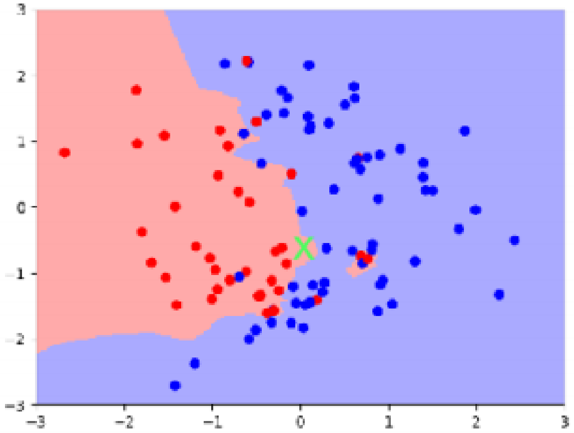
# Bias-Variance Decomposition: Basic Setup

- Suppose the training set $\mathcal{D}$ consists of pairs $(\mathbf{x}_i, t_i)$ sampled independent and identically distributed (i.i.d.) from a single data generating distribution $p_{\text{sample}}$.
- Pick a fixed query point $\mathbf{x}$ (denoted with a green x).
- Consider an experiment where we sample lots of training sets independently from $p_{\text{sample}}$.

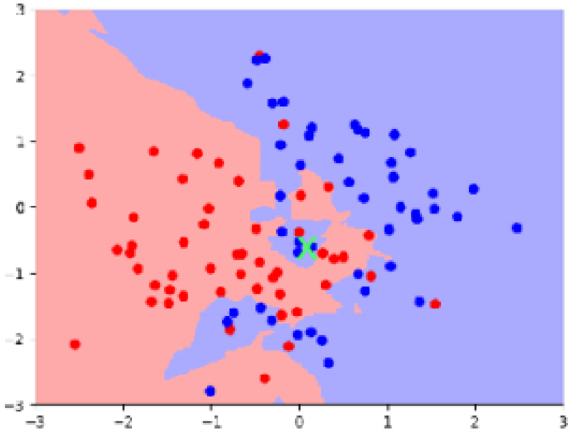


# Bias-Variance Decomposition: Basic Setup

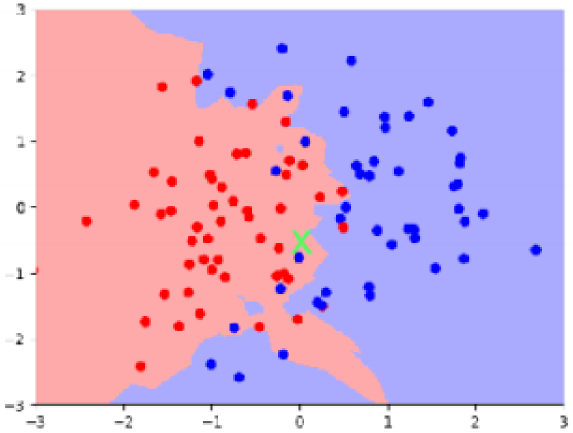
- Let's run our learning algorithm on each training set, and compute its prediction $y$ at the query point $\mathbf{x}$.
- We can view $y$ as a random variable, where the randomness comes from the choice of training set.
- The classification accuracy is determined by the distribution of $y$.



y = ●



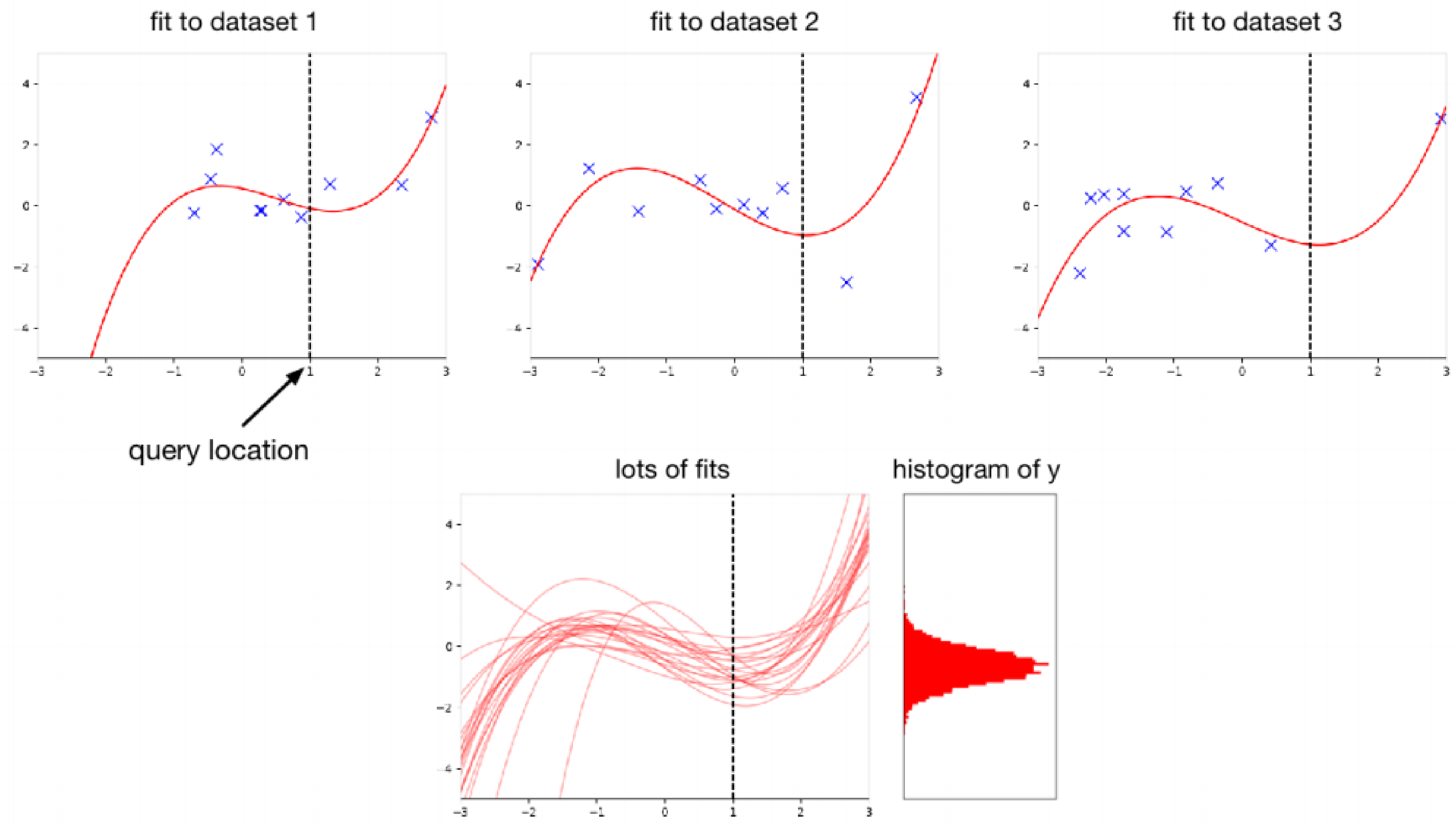
y = ●



y = ●

# Bias-Variance Decomposition: Basic Setup

Here is the analogous setup for regression:




Since $y$ is a random variable, we can talk about its expectation, variance, etc.

# Bias-Variance Decomposition: Basic Setup

- Recap of basic setup:
  - Fix a query point $\mathbf{x}$.
  - Repeat:
    - Sample a random training dataset $\mathcal{D}$ i.i.d. from the data generating distribution $p_{\text{sample}}$.
    - Run the learning algorithm on $\mathcal{D}$ to get a prediction $y$ at $\mathbf{x}$.
    - Sample the (true) target from the conditional distribution $p(t|\mathbf{x})$.
    - Compute the loss $L(y, t)$.
- Notice: $y$ is independent of $t$.
- This gives a distribution over the loss at $\mathbf{x}$, with expectation $\mathbb{E}[L(y, t) \,|\, \mathbf{x}]$.
- For each query point $\mathbf{x}$, the expected loss is different. We are interested in minimizing the expectation of this with respect to $\mathbf{x} \sim p_{\text{sample}}$.

# Bayes Optimality

- For now, focus on squared error loss, $L(y, t) = \frac{1}{2}(y - t)^2$.
- A first step: suppose we knew the conditional distribution $p(t \mid \mathbf{x})$. What value $y$ should we predict?
    - Here, we are treating $t$ as a random variable and choosing $y$.
- **Claim:** $y_* = \mathbb{E}[t \mid \mathbf{x}]$ is the best possible prediction.
- **Proof:**

$$
\begin{aligned}
\mathbb{E}[(y-t)^2 \mid \mathbf{x}] &= \mathbb{E}[y^2 - 2yt + t^2 \mid \mathbf{x}] \\
&= y^2 - 2y\mathbb{E}[t \mid \mathbf{x}] + \mathbb{E}[t^2 \mid \mathbf{x}] \\
&= y^2 - 2y\mathbb{E}[t \mid \mathbf{x}] + \mathbb{E}[t \mid \mathbf{x}]^2 + \mathrm{Var}[t \mid \mathbf{x}] \\
&= y^2 - 2yy_* + y_*^2 + \mathrm{Var}[t \mid \mathbf{x}] \\
&= (y - y_*)^2 + \mathrm{Var}[t \mid \mathbf{x}]
\end{aligned}
$$

# Bayes Optimality

$$\mathbb{E}[(y-t)^2 \mid \mathbf{x}] = (y-y_*)^2 + \mathrm{Var}[t \mid \mathbf{x}]$$

- The first term is nonnegative, and can be made 0 by setting $y = y_*$.
- The second term corresponds to the inherent unpredictability, or noise, of the targets, and is called the Bayes error.
  - This is the best we can ever hope to do with any learning algorithm. An algorithm that achieves it is Bayes optimal.
  - Notice that this term doesn't depend on $y$.
- This process of choosing a single value $y_*$ based on $p(t \mid \mathbf{x})$ is an example of decision theory.

# Bayes Optimality

- Now return to treating $y$ as a random variable (where the randomness comes from the choice of dataset).
- We can decompose out the expected loss (suppressing the conditioning on $\mathbf{x}$ for clarity):

$$\begin{aligned}
\mathbb{E}[(y-t)^2] &= \mathbb{E}[(y-y_\star)^2] + \mathrm{Var}(t) \\
&= \mathbb{E}[y_\star^2 - 2y_\star y + y^2] + \mathrm{Var}(t) \\
&= y_\star^2 - 2y_\star \mathbb{E}[y] + \mathbb{E}[y^2] + \mathrm{Var}(t) \\
&= y_\star^2 - 2y_\star \mathbb{E}[y] + \mathbb{E}[y]^2 + \mathrm{Var}(y) + \mathrm{Var}(t) \\
&= \underbrace{(y_\star - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(y)}_{\text{variance}} + \underbrace{\mathrm{Var}(t)}_{\text{Bayes error}}
\end{aligned}$$

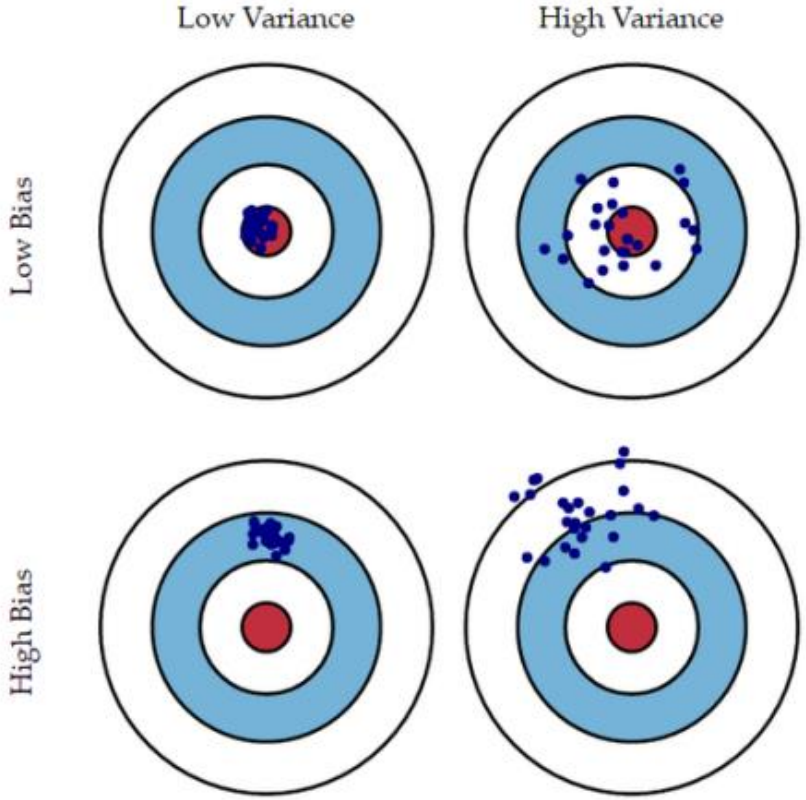
# Bayes Optimality

$$\mathbb{E}[(y-t)^2] = \underbrace{(y_\star - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(y)}_{\text{variance}} + \underbrace{\mathrm{Var}(t)}_{\text{Bayes error}}$$

- We just split the expected loss into three terms:
  - bias: how wrong the expected prediction is (corresponds to underfitting)
  - variance: the amount of variability in the predictions (corresponds to overfitting)
  - Bayes error: the inherent unpredictability of the targets
- Even though this analysis only applies to squared error, we often loosely use "bias" and "variance" as synonyms for "underfitting" and "overfitting".

# Bias and Variance

- Throwing darts = predictions for each draw of a dataset




- Be careful, what doesn't this capture?
  - We average over points $\mathbf{x}$ from the data distribution.