CSC 311: Introduction to Machine Learning

Lecture 2 - Decision Trees & Bias-Variance Decomposition

Rahul G. Krishnan   Alice Gao

University of Toronto, Fall 2022

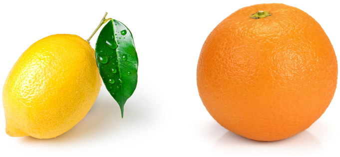
Outline

Today

- Announcement: HW1 released
- 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
  - Concept to motivate combining different classifiers.
- Ideas we will need in today's lecture
  - Trees [from algorithms]
  - Expectations, marginalization, chain rule [from probability]

1. Introduction
2. Decision Trees
3. Bias-Variance Decomposition

Lemons or Oranges



Scenario: You run a sorting facility for citrus fruits

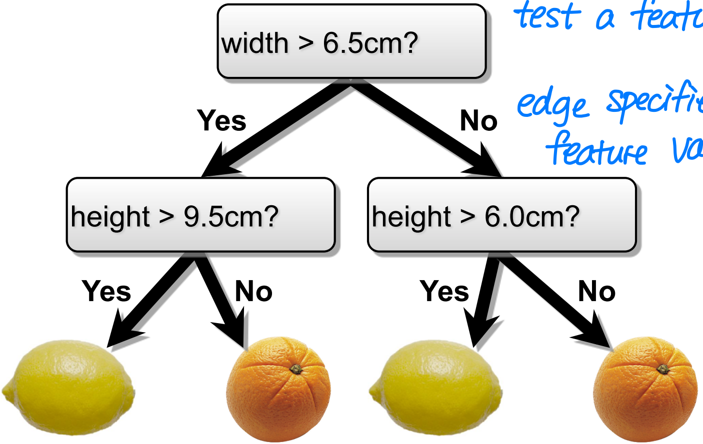
- Binary classification: lemons or oranges
- Features measured by sensor on conveyor belt: height and width

Decision Trees

a natural model we use everyday!

→ Alice : where do I get lunch today?

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



test a feature

edge specifies feature value

leaf specifies class label.

components of a decision tree

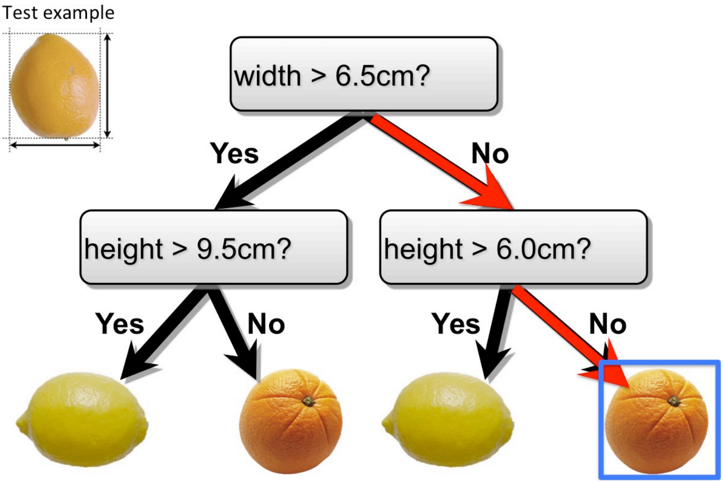
- test an input feature at each node.
- follow one edge corresponding to a value of the input feature.
- each leaf node corresponds to one class.

A natural model that we use everyday!

Decision Trees

once we built a DT, can use it to classify a new example.

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



How do we classify an example using a DT?

- start at root node.
- test input feature and follow corresponding edge.
- once we reach a leaf node, output class label at that node.

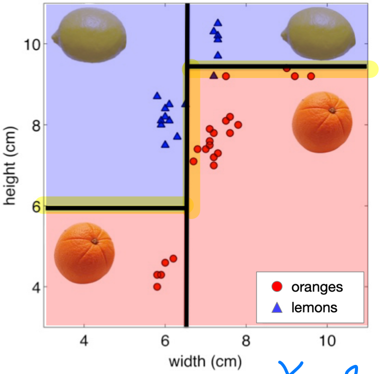
Decision Trees—Continuous Features

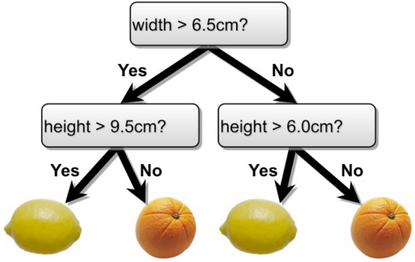
real-valued.

- 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.

contrast this w/ KNN.

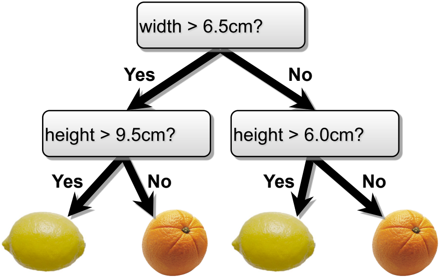
① pick a threshold
② perform a binary split.





(note to Alice: switching Yes & No matches the diagram better.)

Decision Trees



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

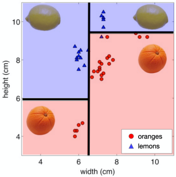
Question: What are the hyperparameters of this model?

hyper-parameters of decision tree:

- # of nodes in the tree.
- max depth of tree
- # of branches at split
- min # of examples at a node.
- max # of features to consider.

Decision Trees—Classification and Regression

- Each path from root to a leaf defines a region R_m of input space   4 regions.
- Let $\{(x^{(m_1)}, t^{(m_1)}), \ldots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m
- $m = 4$ on the right and k is the same across each region

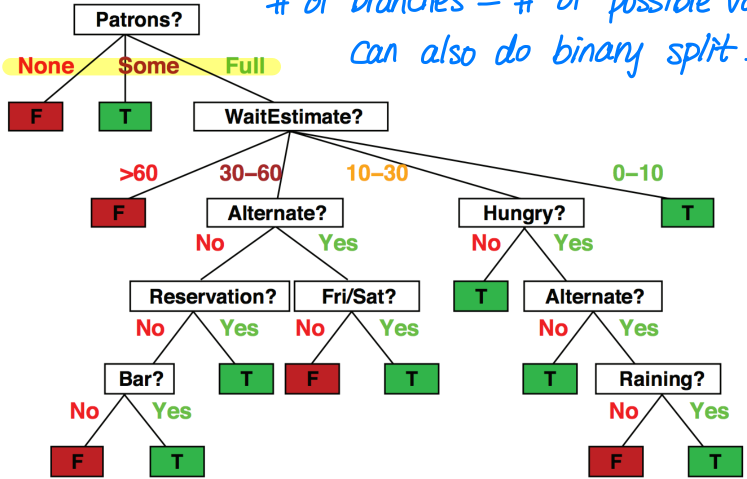


- Regression tree: e.g. predict house price.
  - continuous output
  - leaf value y^m typically set to the mean value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$
- Classification tree (we will focus on this):
  - discrete output
  - leaf value y^m typically set to the most common value in $\{t^{(m_1)}, \ldots,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

Decision Trees—Discrete Features

- Will I eat at this restaurant?

most natural split :
# of branches = # of possible values.
can also do binary split.



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

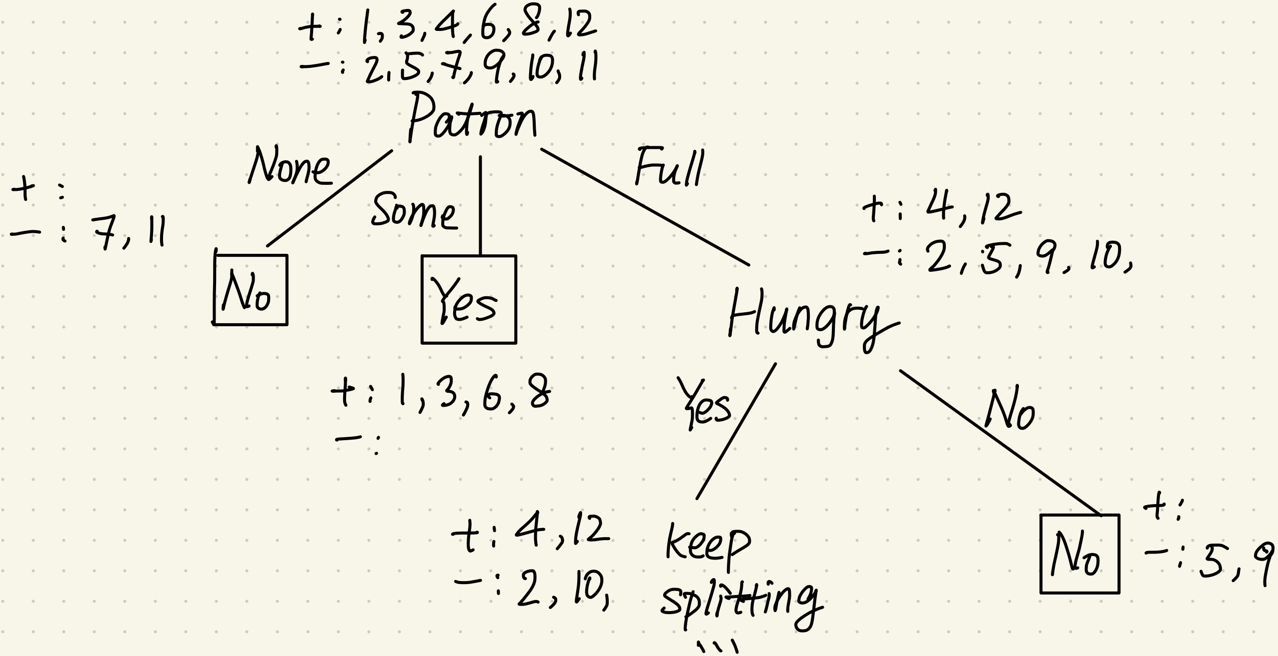
Features:

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).

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

① Patron

② Hungry.



similar to slide 35

Algorithm 1 Decision Tree Learner (examples, features)

3 base cases

```
if all examples are in the same class then
    return the class label.
else if no features left then
    return the majority decision.
else if no examples left then
    return the majority decision at the parent node.
else
    choose a feature f.
    for each value v of feature f do
        build edge with label v.
        build sub-tree using examples where the value of f is v.
```

recursive case

① no features left : multiple examples have the same feature values.
data is noisy. class may be influenced by an unobserved feature.

② no examples left : a combination of feature values in NOT present in the data set

Learning Decision Trees

very powerful, can classify any training set perfectly, but this overfits!

- Decision trees are universal function approximators.
  - 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.
  - Example - If all D features were binary, and we had $N = 2^D$ unique training examples, a **Full Binary Tree** would have one leaf per example.
- Finding the smallest optimal decision tree that correctly classifies a training set is NP complete. is too computationally expensive / not worth it.
  - If you are interested, check: Hyafil & Rivest'76.
- So, how do we construct a useful decision tree?

Learning Decision Trees

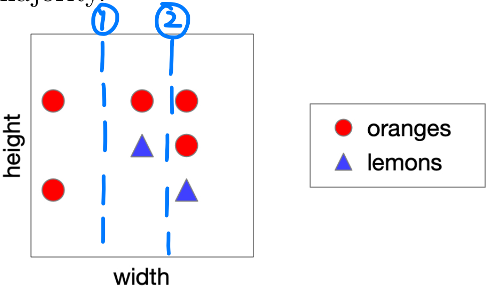
at each step, choose the most informative feature right now.
this is greedy because it's not forward looking.

- 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 a loss
  - Split on that feature and recurse on subpartitions.
- What is a loss?
  - When learning a model, we use a scalar number to assess whether we're on track
  - Scalar value: low is good, high is bad
- Which loss should we use?

optimal choice needs to think about the future.

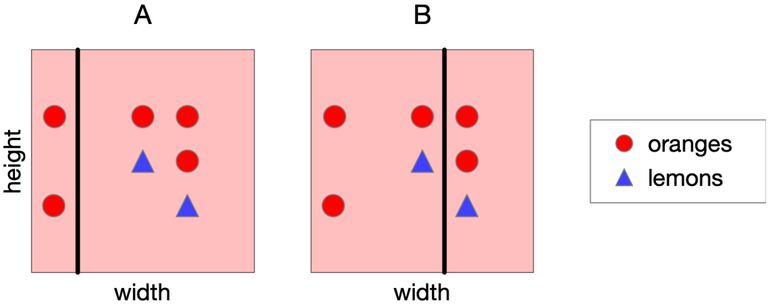
Choosing a Good Split

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



Choosing a Good Split

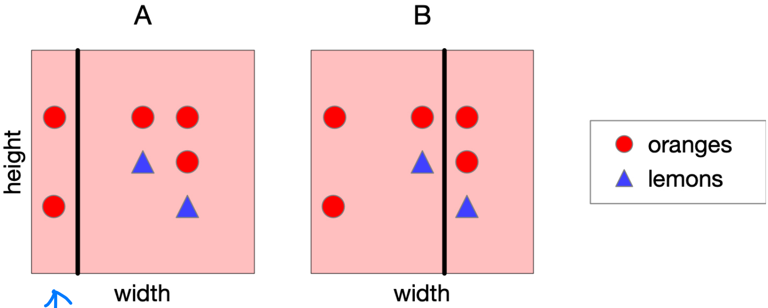
- Which is the best split? Vote!



Choosing a Good Split

In general, the faster we can assign a class label or reach a leaf node, the better.

- 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?



we can already pick red as the label.

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 half lemons, half oranges.
- **Idea:** Use counts at leaves to define probability distributions; use a probabilistic notion of uncertainty to decide splits.
- A brief detour through information theory...

Take examples, convert to counts, then to a probability distribution. Use a concept in information theory to measure the uncertainty in the distribution.

Entropy - Quantifying uncertainty

- You may have encountered the term entropy quantifying the state of chaos in chemical and physical systems,
- In statistics, it is a property of a random variable,
- 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 or Elements of Information Theory by Cover and Thomas.
- To explain entropy, consider flipping two different coins...

will not dive into how people came up w/ entropy.
will simply take the formula and use it.

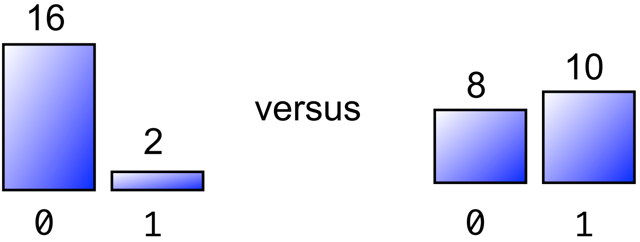
We Flip Two Different Coins

Two biased coins. How biased are they?

Each coin is a binary random variable with outcomes Heads (~~1~~ 0) or Tails (1)

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 ... ?

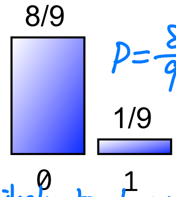


Quantifying Uncertainty

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

binary distribution $(p, 1-p)$

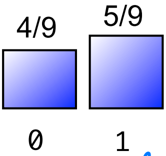
$$-p\log_2(p) - (1-p)\log_2(1-p)$$



$p = \frac{8}{9}$

very likely to observe 0 next.

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



$p = \frac{4}{9}$

closer to a fair coin
more uncertainty.

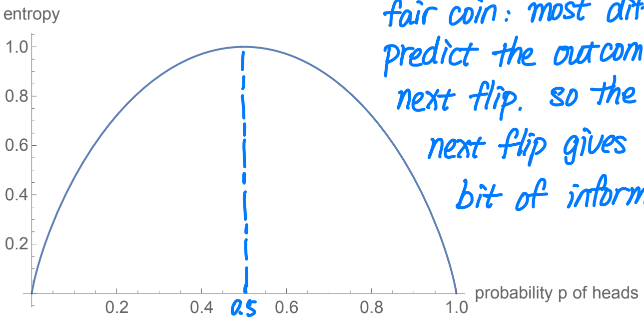
$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- 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.

$p = 0 \Longrightarrow -0\log 0 - 1\log 1 = 0$

Quantifying Uncertainty

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



fair coin: most difficult to predict the outcome of the next flip. so the result of next flip gives 1 full bit of information.

- 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

$\geqslant 2$ outcomes

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

- **"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

[Slide credit: Vibhav Gogate]

Entropy

is its width > 6.5cm?

lemon or orange.

- 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.

Knowing X gives us information about Y, and reduces our uncertainty about Y.

Entropy of a Joint Distribution

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

- when raining, almost cloudy for sure.

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

- when not raining, slightly more likely to be Not Cloudy.

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

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**?

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

→ if raining, very likely cloudy.

low uncertainty

- 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

P(raining) * P(Y| raining) + P(not raining) * P(Y| not raining)

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

$\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x)$

$= \sum_x \sum_y \overbrace{P(x) P(y|x)}^{P(x,y)} * \log_2 P(y|x)$

- The expected conditional entropy:

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

Comments on conditional entropy:

- knowing each value of x reduces my uncertainty about y.
- but I cannot observe x yet so I can only calculate the uncertainty reduction in expectation.
- expectation is over the prob of observing each value of x.

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?

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= \frac{1}{4} \underbrace{H(\text{cloudy}|\text{is raining})}^{0.24 \text{ bits}} + \frac{3}{4} H(\text{cloudy}|\text{not raining}) \\ &\approx 0.75 \text{ bits} \end{aligned}$$

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

mutual info: amount of info obtained about one random variable by observing the other random variable.

	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

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

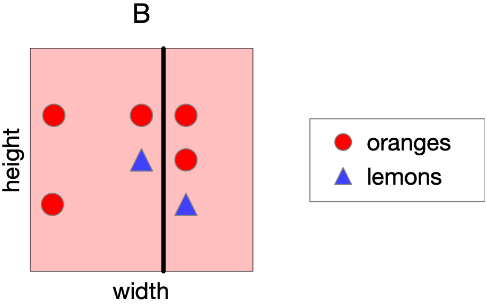
- If X is completely uninformative about Y: $IG(Y|X) = 0$
- If X is completely informative about Y: $IG(Y|X) = H(Y)$

Revisiting Our Original Example

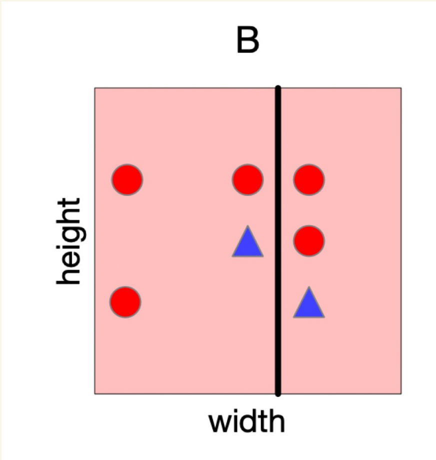
- Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree split!
- The information gain of a split: how much information (over the training set) about the class label Y is gained by knowing which side of a split you're on.

Information Gain of Split B

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



- Entropy of class outcome before split:
  $H(Y) = -\frac{2}{7}\log_2(\frac{2}{7}) - \frac{5}{7}\log_2(\frac{5}{7}) \approx 0.86$
- Conditional entropy of class outcome after split:
  $H(Y|left) \approx 0.81$, $H(Y|right) \approx 0.92$
- $IG(split) \approx 0.86 - (\frac{4}{7} \cdot 0.81 + \frac{3}{7} \cdot 0.92) \approx 0.006$



Y: blue: 2, red: 5 $\quad \left(\frac{2}{7}, \frac{5}{7}\right)$

$$H(Y) = -\frac{2}{7}\log_2\frac{2}{7} - \frac{5}{7}\log_2\frac{5}{7}.$$

left: (4) blue: 1, red: 3 $\quad \left(\frac{1}{4}, \frac{3}{4}\right)$

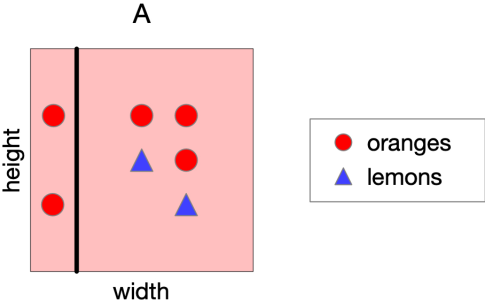
right: (3) blue: 1, red: 2 $\quad \left(\frac{1}{3}, \frac{2}{3}\right)$

$$H(Y|\text{split}) = \frac{4}{7}\left(-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right) + \frac{3}{7}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)$$

$$IG(\text{split}) = H(Y) - H(Y|\text{split})$$

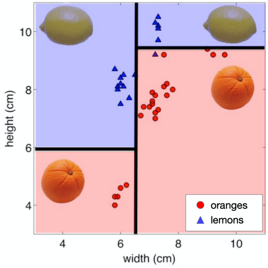
Information Gain of Split A

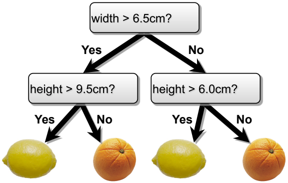
- What is the information gain of split A? Very informative!



- Entropy of class outcome before split:
  $H(Y) = -\frac{2}{7}\log_2(\frac{2}{7}) - \frac{5}{7}\log_2(\frac{5}{7}) \approx 0.86$
- Conditional entropy of class outcome after split:
  $H(Y|left) = 0$, $H(Y|right) \approx 0.97$
- $IG(split) \approx 0.86 - (\frac{2}{7} \cdot 0 + \frac{5}{7} \cdot 0.97) \approx 0.17!!$

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.
- Questions for discussion:
  - How do you choose the feature to split on?
  - How do you choose the threshold for each feature?

Back to Our Example

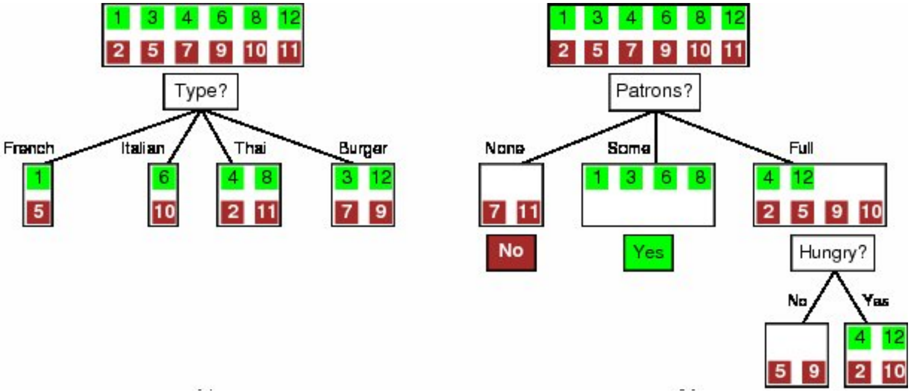
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3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

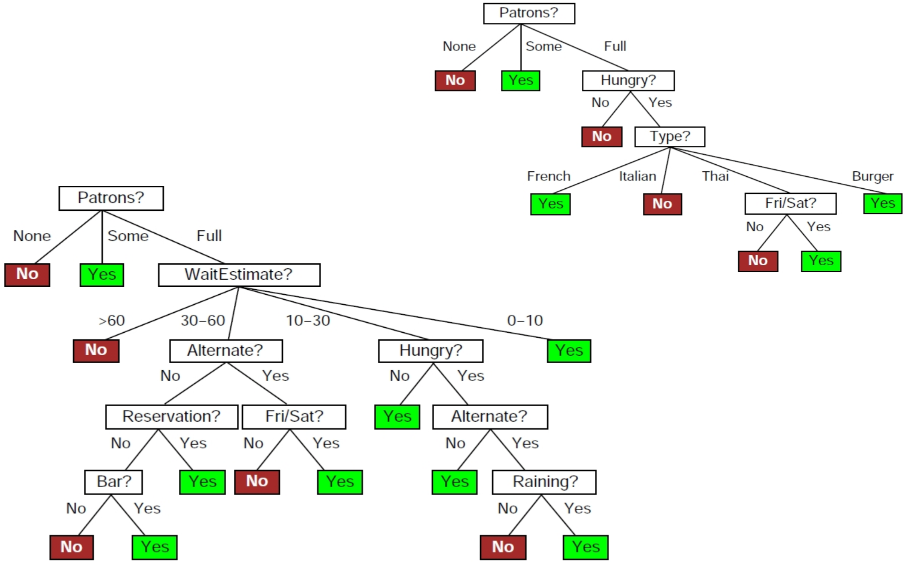


$$IG(Y) = H(Y) - H(Y|X)$$

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

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

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

- prune leaves w/ too little data
- optimal choice at each step

$\not\Rightarrow$ a globally optimal tree.

- 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.

KNN versus Decision Trees

Advantages of decision trees over KNNs

- Simple to deal with discrete features, missing values, and poorly scaled data
- Fast at test time KNN needs to iterate thru entire data set.
- More interpretable easily explain the decision making process.

Advantages of KNNs over decision trees

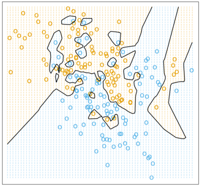
- Few hyperparameters (K)
- Can incorporate interesting distance measures (e.g. shape contexts)

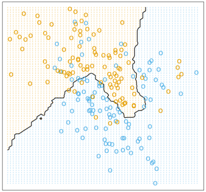
- We've seen many classification algorithms.
- We can combine multiple classifiers into an ensemble, which is a set of predictors whose individual decisions are combined in some way to classify new examples
  - E.g., (possibly weighted) majority vote
- For this to be nontrivial, the classifiers must 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.

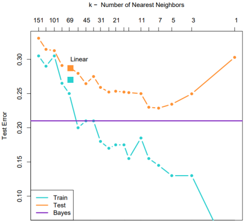
- Today, we deepen our understanding of generalization through a bias-variance decomposition.
  - This will help us understand ensembling methods.
- What is generalization?
  - Ability of a model to correctly classify/predict from unseen examples (from the same distribution that the training data was drawn from).
  - **Why does this matter?** Gives us confidence that the model has correctly captured the right patterns in the training data and will work when deployed.

Bias-Variance Decomposition

- Overly simple models underfit the data, and overly complex models overfit.
- We can quantify underfitting and overfitting in terms of the bias/variance decomposition.



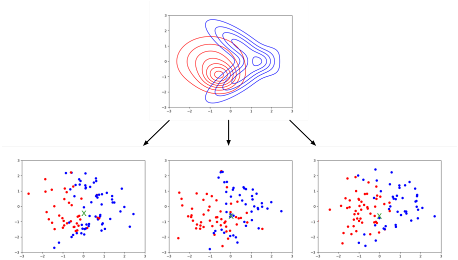




Basic Setup for Classification

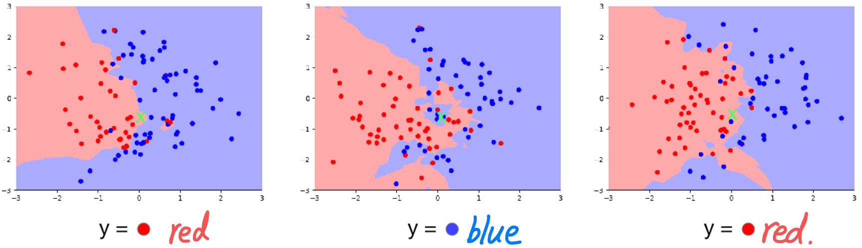
never observe this

- p_{sample} is a data generating distribution.
  For lemons and oranges, p_{sample} characterizes heights and widths.
- Pick a fixed query point $\mathbf{x}$ (denoted with a green x).
  We want to get a prediction y at $\mathbf{x}$.
- A training set $\mathcal{D}$ consists of pairs $(\mathbf{x}_i, t_i)$ sampled
  independent and identically distributed (i.i.d.) from p_{sample}.
- We can sample lots of training sets independently from p_{sample}.

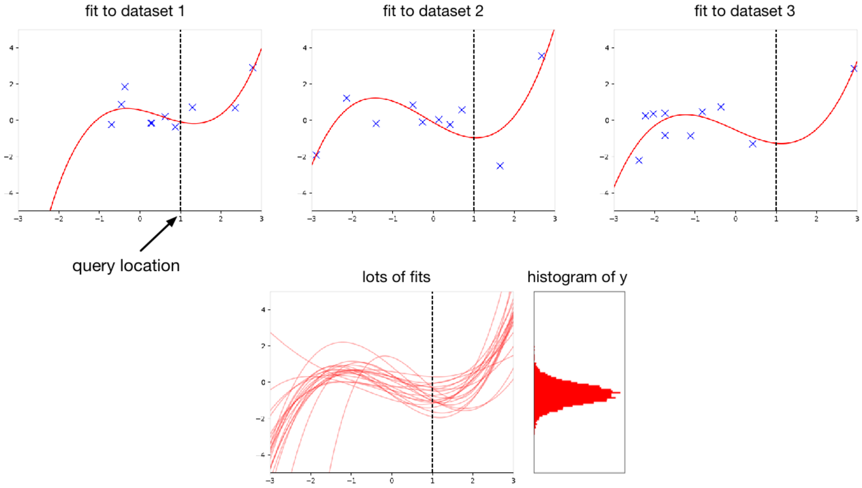


Basic Setup for Classification

- 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.
- Since y is a random variable, we can compute its expectation, variance, etc.



Basic Setup for Regression



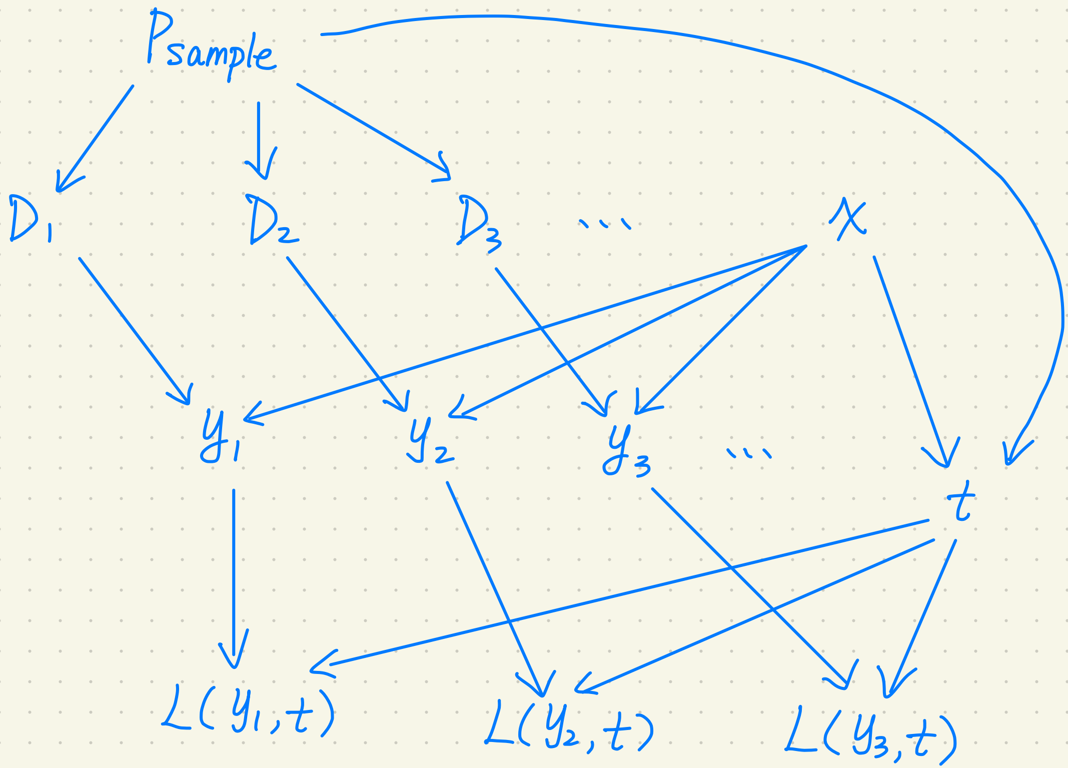
Basic Setup

- Fix a query point $\mathbf{x}$.
- Repeat:
  - Sample a random training dataset $\mathcal{D}$ i.i.d. from the data generating distribution p_{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 $L(y, t)$.

Comments:

- Notice: $\underset{\mathcal{D}}{y}$ is independent of $\underset{P_{sample}}{t}$. (Why?)

- produced y using the training set D.
- cannot recover P_{sample} from D. (if we could, we don't need D anymore.)



P_{sample}
D_1
D_2
D_3
...
X
y_1
y_2
y_3
...
t
$L(y_1, t)$
$L(y_2, t)$
$L(y_3, t)$

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_{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 $L(y, t)$.

Comments:

- Notice: y is independent of t. (Why?)
- 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 $\mathbf{x} \sim p_{\text{sample}}$.

Choosing a prediction y

- Consider squared error loss, $L(y, t) = \frac{1}{2}(y - t)^2$.
- Suppose that we knew the conditional distribution $p(t \mid \mathbf{x})$.
  What value of y should we predict?
  - Treat t as a random variable and choose y.

Choosing a prediction y

- Consider squared error loss, $L(y,t) = \frac{1}{2}(y-t)^2$.
- Suppose that we knew the conditional distribution $p(t \mid \mathbf{x})$. What value of y should we predict?
  - Treat t as a random variable and choose y.
- **Claim:** $y_* = \mathbb{E}[t \mid \mathbf{x}]$ is the best possible prediction.
- **Proof:**

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

How do we choose y to minimize $E[(y-t)^2|x]$?

$$E[(y-t)^2|x] = E[(y^2-2yt+t^2)|x]$$

(E is linear.) $\quad = E[y^2|x] - E[2yt|x] + E[t^2|x]$

(we choose y.) $\quad = y^2 - 2yE[t|x] + E[t^2|x]$

$(Var[x] = E[x^2] - (E[x])^2)$ $\quad = y^2 - 2yE[t|x] + (E[t|x])^2 + Var[t|x]$

(let $y^* = E[t|x]$) $\quad = (y - E[t|x])^2 + Var[t|x]$

$$= (y-y^*)^2 + \underbrace{Var[t|x]}_{\text{Bayes error}}$$

y cannot influence $Var[t|x]$ since y and t are independent.

best choice of y is $y = y^* = E[t|x]$

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 is the Bayes error, or
  the noise or inherent unpredictability of the target t.
  - An algorithm that achieves it is Bayes optimal.
  - This term doesn't depend on y.
  - Best we can ever hope to do with any learning algorithm.
- This process of choosing a single value y_* based on $p(t \mid \mathbf{x})$ is an example of decision theory.

Decomposition Continued

$y^* = E[t|x]$

- Now let's treat y as a random variable (where the randomness comes from the choice of dataset).
- We can decompose the expected loss further (suppressing the conditioning on $\mathbf{x}$ for clarity):

$$
\begin{aligned}
\mathbb{E}[(y-t)^2] &= \mathbb{E}[(y-y_\star)^2] + \mathrm{Var}(t) \\
\text{expanding the square.} \quad &= \mathbb{E}[y_\star^2 - 2y_\star y + y^2] + \mathrm{Var}(t) \\
\text{linearity of } E. \quad &= y_\star^2 - 2y_\star \mathbb{E}[y] + \mathbb{E}[y^2] + \mathrm{Var}(t) \\
Var[y] = E[y^2] - (E[y])^2 \quad &= y_\star^2 - 2y_\star \mathbb{E}[y] + \mathbb{E}[y]^2 + \mathrm{Var}(y) + \mathrm{Var}(t) \\
\text{regrouping terms.} \quad &= \underbrace{(y_\star - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(y)}_{\text{variance}} + \underbrace{\mathrm{Var}(t)}_{\text{Bayes error}}
\end{aligned}
$$

Bayes Optimality

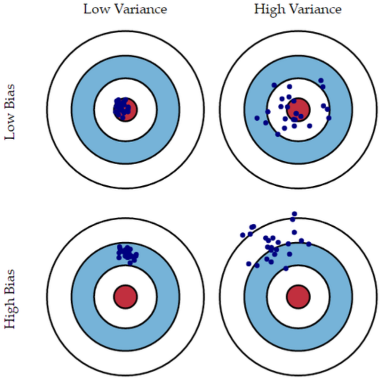
$$\mathbb{E}[(y-t)^2] = \underbrace{(y_\star - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(y)}_{\text{variance}} + \underbrace{\mathrm{Var}(t)}_{\text{Bayes error}}$$

We split the expected loss into three terms:

- bias: how wrong the expected prediction is (corresponds to underfitting) large bias $\rightarrow$ underfitting
- variance: the amount of variability in the predictions (corresponds to overfitting) large variance $\rightarrow$ overfitting.
- Bayes error: the inherent unpredictability of the targets

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.