CSC 311: Introduction to Machine Learning

Lecture 2 - Decision Trees & Bias-Variance Decomposition

Rahul G. Krishnan & Amanjit Singh Kainth

University of Toronto, Fall 2024

Outline

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

Introduction

Today

- **Announcement:** HW1 (will be) released this week
- **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]

Decision Trees

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

Algorithms & data structures

Directed graph.

○ nodes

→ edges

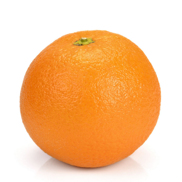
Represent predictive decisions as a graph.

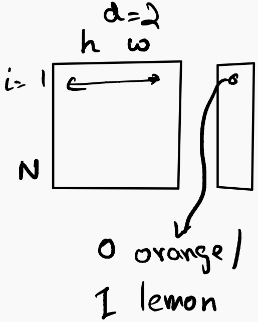
○ computation or decision

→ indicator for what to do based on the computation

Identical to making predictions with if-else statements.

Lemons or Oranges



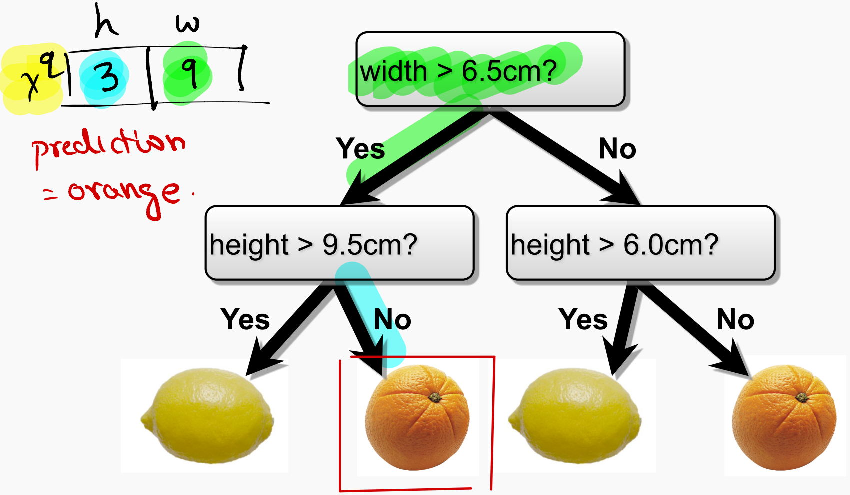


Scenario: You run a sorting facility for citrus fruits

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

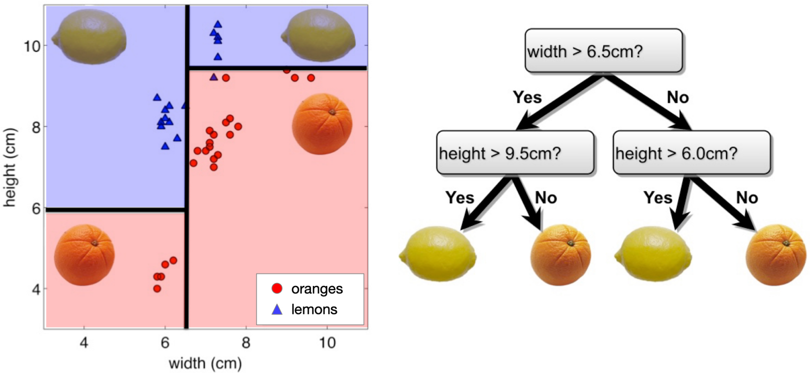
Decision Trees

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



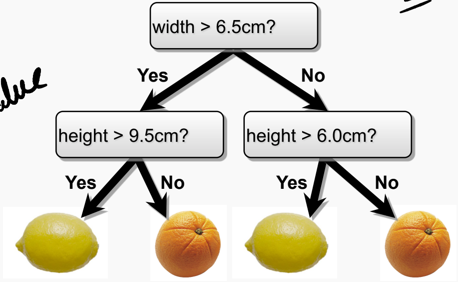
Decision Trees—Continuous Features

- 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

Parameters
- computation at each node
  → feature value
  → threshold.



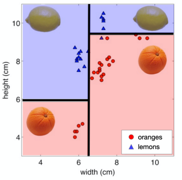
HP
- depth of tree
- nodes
- branching factor

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

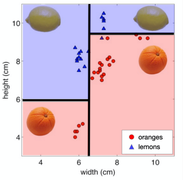
Question: What are the hyperparameters of this model?

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
- $m = 4$ on the right



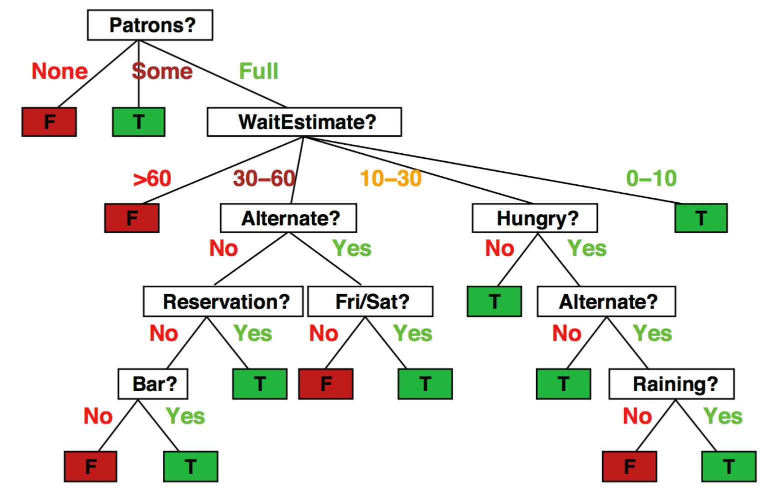
Decision Trees—Classification and Regression



- 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
- $m = 4$ on the right
- **Regression tree**:
  - 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?



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).

Implementing Decision Trees

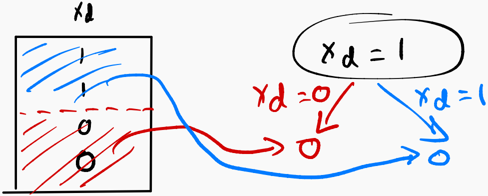
- Step 1: Understand the problem (is it prediction, learning a good representation). **Regression or classification**
- Step 2: Formulate the problem mathematically (create notation for your inputs and outcomes and model). **similar to KNN - vectorize inputs and labels**
- Step 3: Formulate an objective function that represents success for your model.
- Let $\mathcal{D} = \{(\mathbf{x}^1, t^1), \ldots, (\mathbf{x}^N, t^N)\}$ be the training set, $\mathcal{T}$ be the space of valid decision trees and $y(\mathbf{x})$ be the label predicted by running the decision tree on an input.
- **Objective:** $\mathcal{L} = \min_{\mathcal{T}} \sum_{i=1}^{N} \mathbb{I}[y^i \neq t^i]$ is to minimize the number of misclassifications.
- **Why is this difficult?**

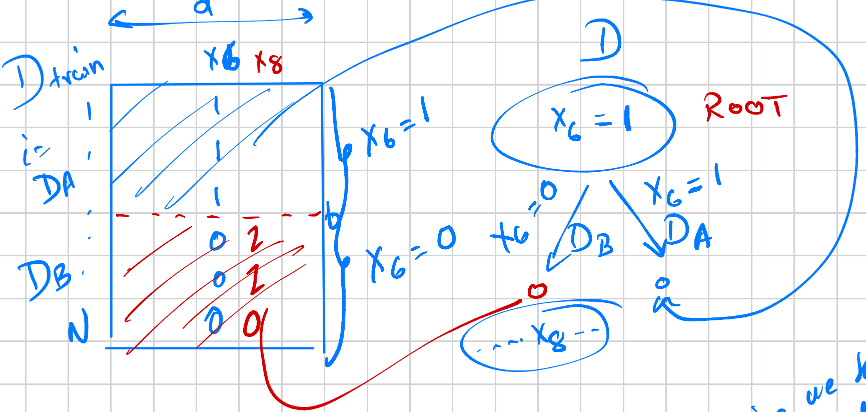
Hardness of learning Decision Trees

- 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 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**:
  - **Intuition:** Do the sensible thing locally and then repeat!
  - 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?

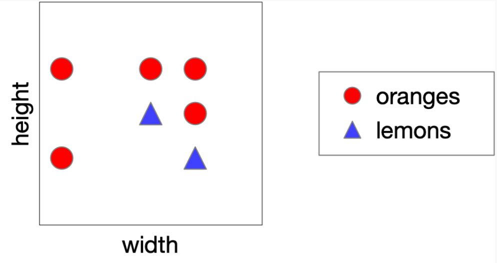




Assuming we know how to select the node in the decision tree

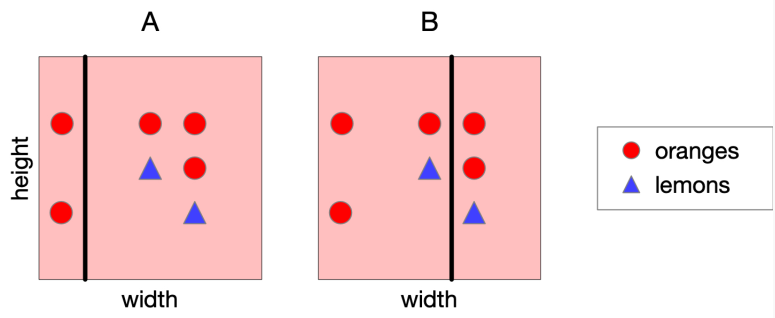
Choosing a Good Split

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



Choosing a Good Split

- Which is the best split? Vote!



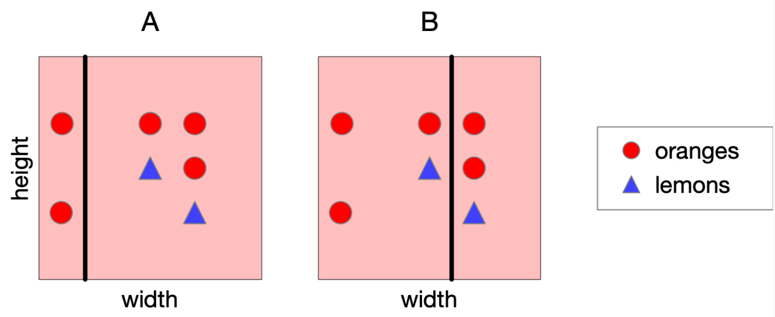
Probability in review

Three concepts you should page into memory for the next fifteen minutes:

- Expectation: $\mathbb{E}_x[f(x)] = \sum_{x \in X} p(x)f(x)$
- Chain rule of probabilities: $p(y|x)p(x) = p(x, y)$
- Marginalization of joint probabilities: $p(x) = \sum_y p(x, y)$

Choosing a Good Split

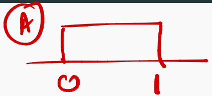
- A feels like a better split, because the left-hand region is very certain about whether the fruit is an orange.
- Can we quantify this?

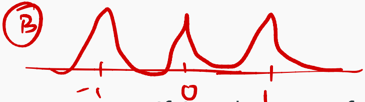


Choosing a Good Split

- 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...

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...

We Flip Two Different Coins

Each coin is a binary random variable with outcomes 1 or 0:

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 ... ?

We Flip Two Different Coins

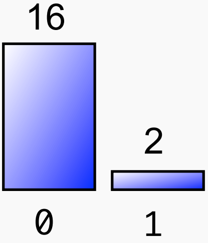
Each coin is a binary random variable with outcomes 1 or 0:

Sequence 1:

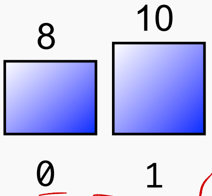
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:

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



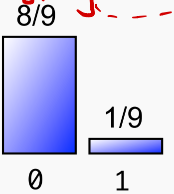
versus

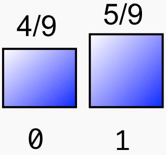


Quantifying Uncertainty

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

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



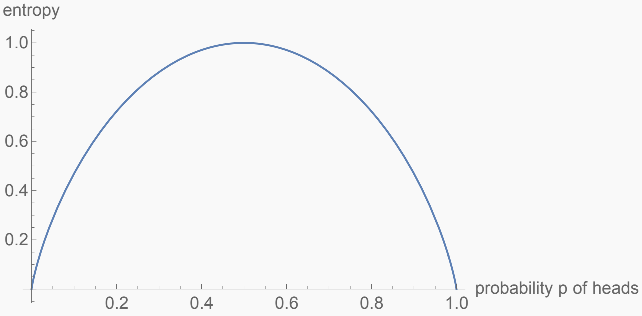


$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2} \qquad -\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- 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.

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

[Slide credit: Vibhav Gogate]

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

[Slide credit: Vibhav Gogate]

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

$$H(X,Y) = \mathbb{E}_{p(x,y)}\left[-\log p(x,y)\right]$$

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

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**?

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

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

$p(x,y)$ - joint probability

$p(y|x)$ - conditional prob.

$$p(y|x) = \frac{p(y,x)}{p(x)}$$ Bayes Rule.

$$= \frac{0}{\sum_{y} p(x,y)}$$

$$p(Y = \text{cloudy} \mid X = \text{rain}) = \frac{24/100}{24/100 + 1/100} = 24/25$$

Conditional Entropy

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

- The expected conditional entropy:

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

$= P(X=\text{rain}) \cdot H(Y|X=\text{rain}) + P(X=\neg\text{rain}) \cdot H(Y|X=\neg\text{rain})$

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

Splits in a decision tree → quantify uncertainty → $H(Y)$ → $H(Y,X)$ → $H(Y|X=\text{rain})$ → $H(Y|X)$

- 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

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

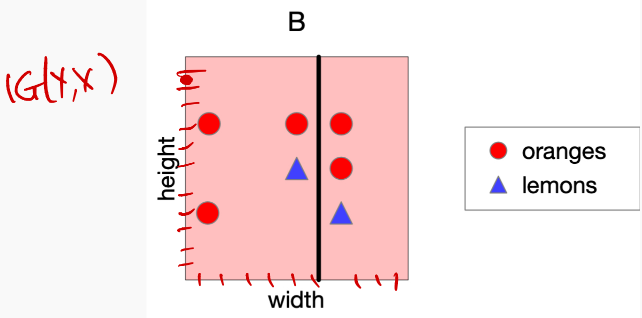
- If X is completely uninformative about Y: $IG(Y|X) = 0$
- If X is completely informative about Y: $IG(Y|X) = H(Y)$

Revisiting Our Original Example

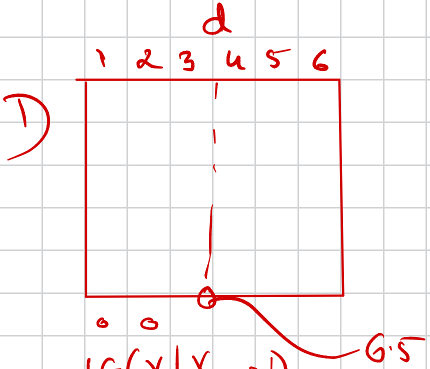
- Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree split!
- The information gain of a split: how much information (over the training set) about the class label Y is gained by knowing which side of a split you're on.

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



$IG(Y \mid X_{[\,]} = 1)$

$X_{[\,]} = 1$

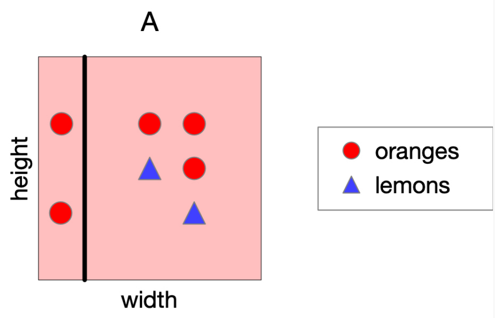
Y ___ X ______

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

binary
categorical.

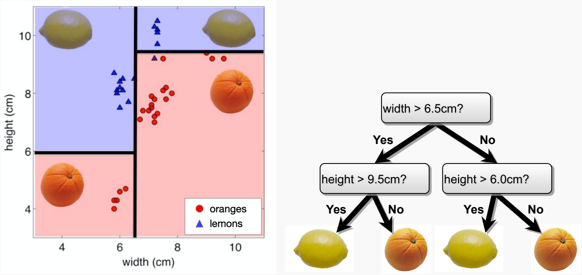
Information Gain of Split A

- 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

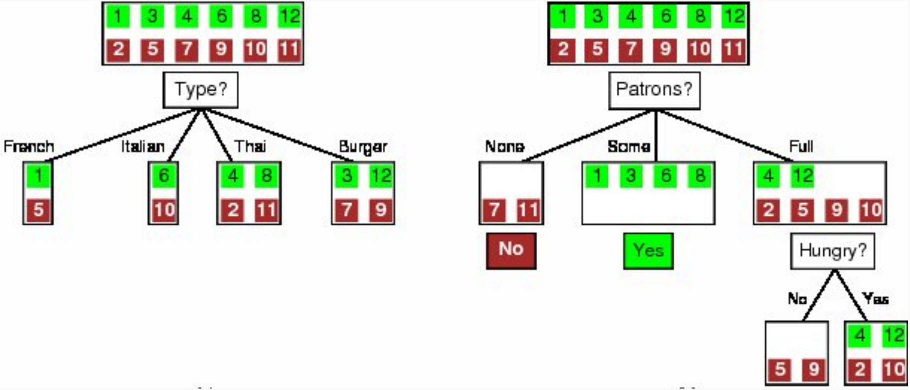
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3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

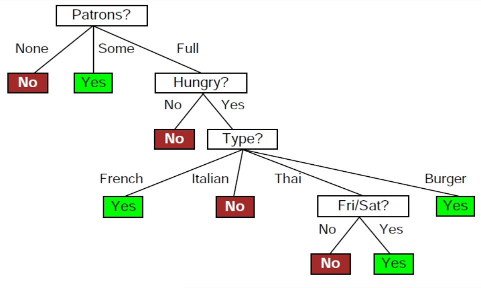


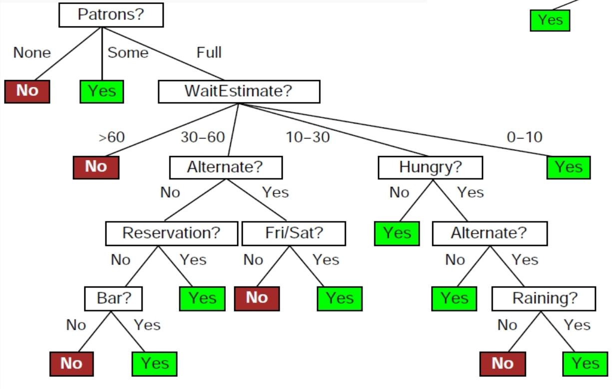
$$IG(Y) = H(Y) - H(Y|X)$$

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

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

Which Tree is Better? Vote!





What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data

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

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"

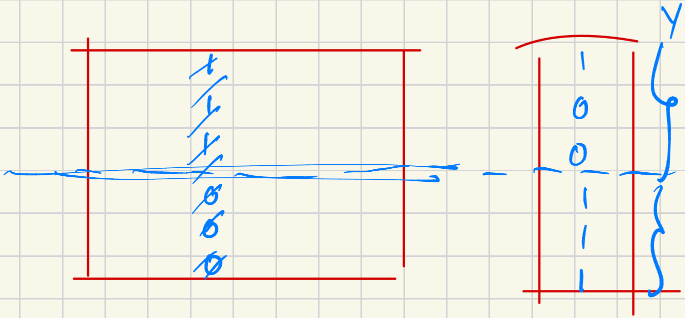
What Makes a Good Tree?

- 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

Steps to building decision trees

Below is a categorization of ML problems that you will see time, and time-again throughout this semester.

- Step 1: Understand the problem (is it prediction, learning a good representation).
- Step 2: Formulate the problem mathematically (create notation for your inputs and outcomes and model).
- Step 3: Formulate an objective function that represents success for your model.
- Step 4: Find a strategy to solve the optimization problem on pencil and paper. **Greedy algorithm to construct trees node by node**
- Step 5: Translate the algorithm into code. **Part of the homework excercise to translate this idea into code**
- Step 6: Analyze, iterate, improve design choices in your model and algorithm


Y

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

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

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 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.

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.

KNN versus Decision Trees

Advantages of decision trees over KNNs

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
- More interpretable

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
- More interpretable

Advantages of KNNs over decision trees

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
- More interpretable

Advantages of KNNs over decision trees

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

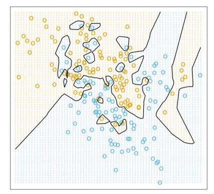
- We've seen many classification algorithms.
- We can combine multiple classifiers into an **ensemble**, which is a set of predictors whose individual decisions are combined in some way to classify new examples
    - E.g., (possibly weighted) majority vote
- For this to be nontrivial, the classifiers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

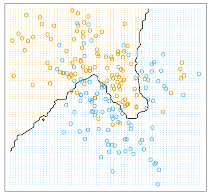
Bias-Variance Decomposition

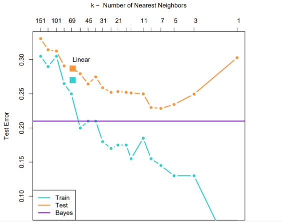
- Today, we deepen our understanding of generalization through a bias-variance decomposition.
  - This will help us understand ensembling methods.
- What is generalization?
  - Ability of a model to correctly classify/predict from unseen examples (from the same distribution that the training data was 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**.







Aside: Quick review of sampling

- **Sampling** is the process of drawing random variables from a distribution that describes its behavior.
- $x \sim \mathcal{N}(0, 1)$ (univariate sampling from a standard normal distribution). Empirical samples: $\{x^1, x^2, \dots, x^N\}$, $x^i \in \mathbb{R}$
- $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$ (multivariate sampling from a normal distribution with covariance Σ). Empirical samples: $\{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N\}$, $\mathbf{x}^i \in \mathbb{R}^d$
- $y \sim \mathcal{N}(5x + 12, 1)$ (univariate sampling from a conditional distribution whose mean is conditional on input). Empirical (conditional) samples: $\{y^1, y^2, \dots, y^N\}$ given $\{x^1, x^2, \dots, x^N\}$, $x^i, y^i \in \mathbb{R}$

Aside: Quick review

- Previously, we knew what the distribution was and how they were parameterized.
- The samples are independent and identically distributed.
- For many phenomena, we may not know how data is distributed.
- Make assumptions on how data are distributed, we'll use ideas from statistics to better understand our model's generalization error.

Roadmap

(A) Create steps that we will assume our data are generated according to. (May not be perfect but our goal is to have this set of steps be defensible)

(B) Understand how well a classifier trained on setup in (A) would do.

Basic Setup for Classification

imagine that nature controls this

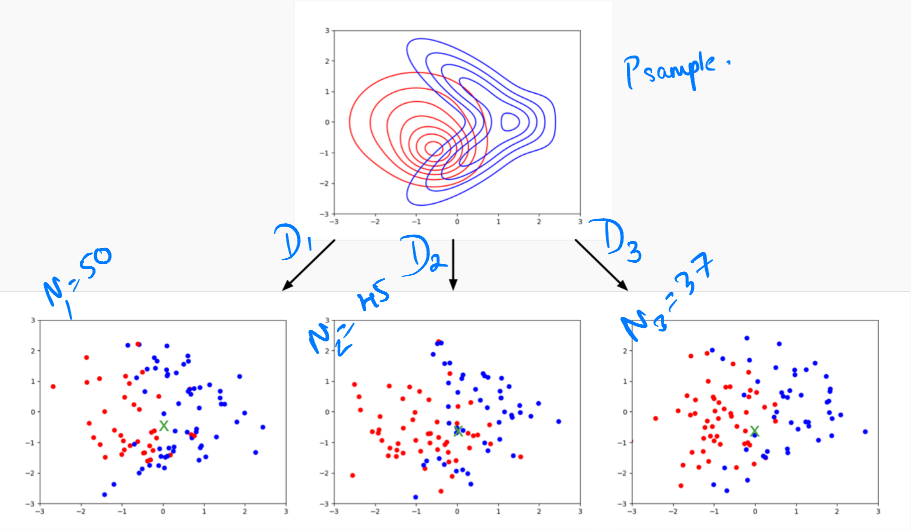
describes joint distribution of inputs, labels

- p_{sample} is a **data generating distribution**.
  For lemons and oranges, $p_{\text{sample}}(x, t)$ characterizes the true heights, widths, and labels.
- Think of this as the (true, but unknown) distribution of heights and widths of oranges and lemons in **nature**.
- Similarly we have the (true, but unknown) distribution of the target (orange or lemon) conditional on the heights and widths of the fruit **nature**: $p_{\text{target}}(t|x)$.
- We assume that the 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

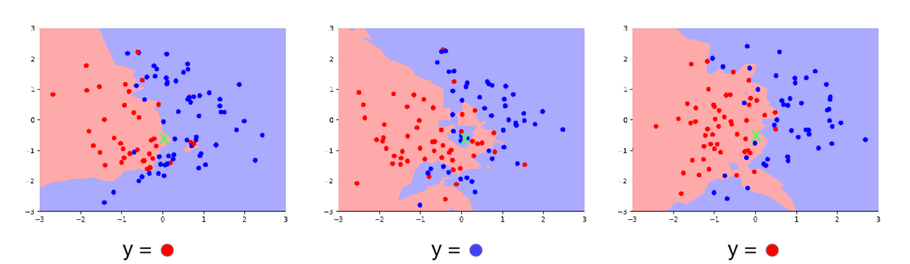
- How do we use the idea of a data generating distribution to understand generalization?
- Generalization is about model performance on a new point – lets pick one!
- Pick a fixed query point $\mathbf{x}$ (denoted with a green x).
  We want to get a prediction y at $\mathbf{x}$.

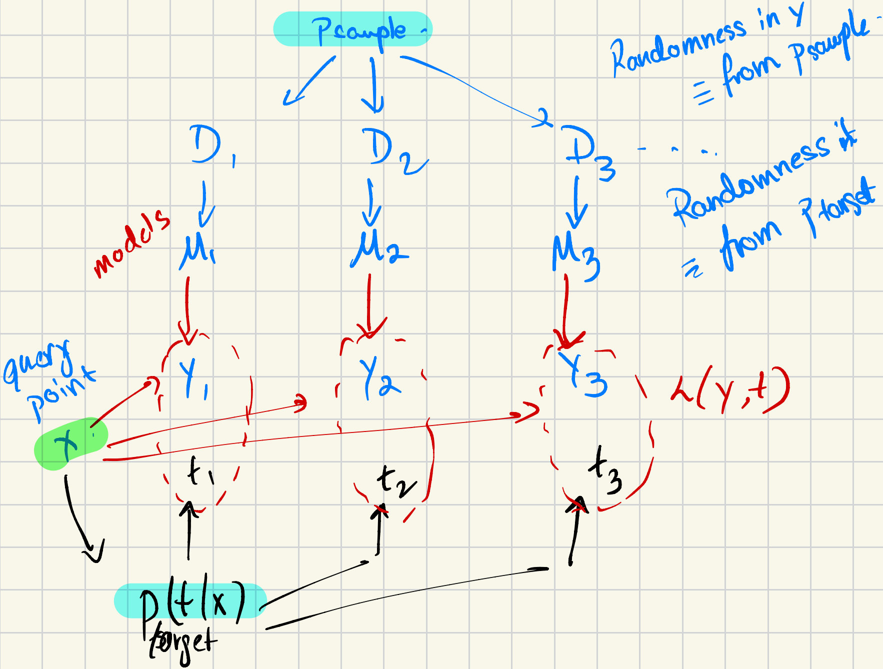
Basic Setup for Classification



Basic Setup for Classification

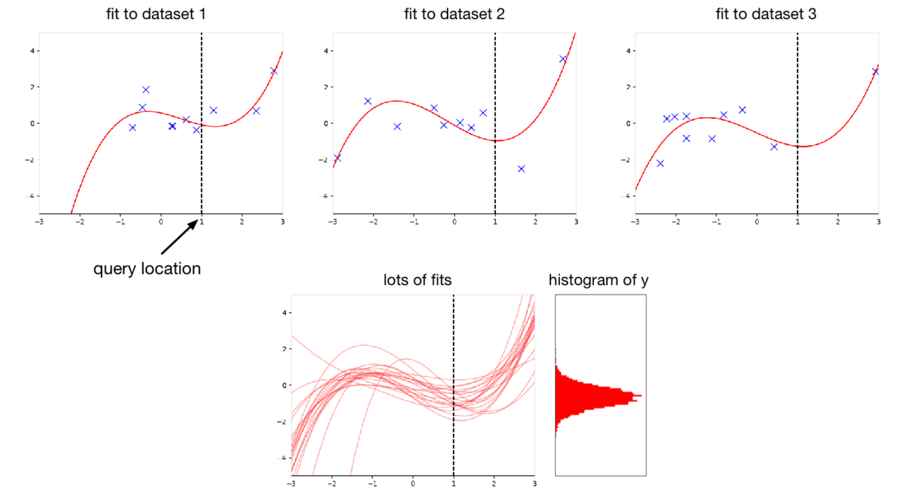
- Run our (deterministic) learning algorithm on each training set, and compute its prediction y at the query point $\mathbf{x}$.
- We can view y as a random variable, where the randomness comes from the choice of training set.
- The classification accuracy is determined by the distribution of y.
- Since y is a random variable, we can compute its expectation, variance, etc.





Psample
D1
D2
D3
models
M1
M2
M3
Randomness in y
= from Psample
Randomness in t
= from Ptarget
query point
x
Y1
Y2
Y3
L(y, t)
t1
t2
t3
p(t|x)
target

Basic Setup for Regression



Basic Setup

- For a fixed query point $\mathbf{x}$, repeat:
  - Sample a random training set $\mathcal{D}$ i.i.d. from 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:

- The random variable corresponding to the prediction y is independent of the t – Why?

Basic Setup

- For a fixed query point $\mathbf{x}$, repeat:
  - Sample a random training set $\mathcal{D}$ i.i.d. from 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:

- The random variable corresponding to the prediction y is independent of the t – Why?
- The above algorithm gives a distribution over the loss at $\mathbf{x}$, with expectation $\mathcal{L}_{\text{query}} = \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{p(t \,|\, \mathbf{x})}[L(y, t) \,|\, \mathbf{x}]]$.

Basic Setup

- For a fixed query point $\mathbf{x}$, repeat:
  - Sample a random training set $\mathcal{D}$ i.i.d. from 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:

- The random variable corresponding to the prediction y is independent of the t – Why?
- The above algorithm gives a distribution over the loss at $\mathbf{x}$, with expectation $\mathcal{L}_{\text{query}} = \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{p(t\,|\,\mathbf{x})}[L(y,t)\,|\,\mathbf{x}]]$.
- We've made progress! We've precisely written down a mathematical expression corresponding to the generalization error that we incur!

Basic Setup

- For a fixed query point $\mathbf{x}$, repeat:
  - Sample a random training set $\mathcal{D}$ i.i.d. from 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:

- The random variable corresponding to the prediction y is independent of the t – Why?
- The above algorithm gives a distribution over the loss at $\mathbf{x}$, with expectation $\mathcal{L}_{\text{query}} = \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{p(t\,|\,\mathbf{x})}[L(y, t)\,|\,\mathbf{x}]]$.
- We've made progress! We've precisely written down a mathematical expression corresponding to the generalization error that we incur!
- If our model has generalized, then it means the expected loss is low. When does this happen?

Choosing a prediction y

- For convenience we'll work in regression and assumed the following function to quantify the error in our prediction (square loss), $L(y, t) = \frac{1}{2}(y - t)^2$.
- Imagine that we knew the conditional distribution $p_{\text{target}}(t \mid \mathbf{x})$.
  What value of y should we predict?
    - Treat t as a random variable and choose y.

Choosing a prediction y

- For convenience we'll work in regression and assumed the following function to quantify the error in our prediction (square loss), $L(y,t) = \frac{1}{2}(y-t)^2$.
- Imagine that we knew the conditional distribution $p_{\text{target}}(t \mid \mathbf{x})$. What value of y should we predict?
  - Treat t as a random variable and choose y.
- **Claim:** $y_\star = \mathbb{E}_{p_{\text{target}}(t \mid \mathbf{x})}[t \mid \mathbf{x}]$ is the best possible prediction.
- **Proof:**

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

Identity

$\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

Bayes Optimality

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

- The first term is nonnegative, and can be made 0 by setting $y = y_\star$.
- 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 $p_{\text{target}}(t \mid \mathbf{x})$ is an example of **decision theory**.

Decomposition Continued

- 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):

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

Bayes Optimality

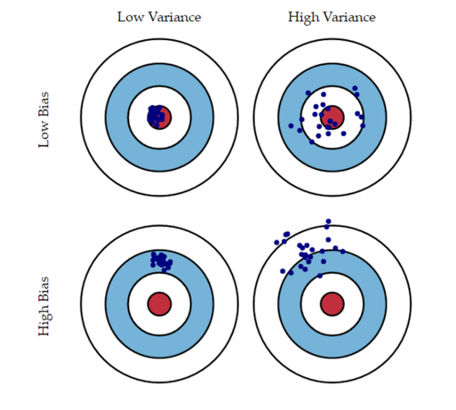
$$\mathbb{E}_{\mathcal{D}}[\mathbb{E}_{p(t)}[(y-t)^2]] = \underbrace{(y_\star - \mathbb{E}_{\mathcal{D}}[y])^2}_{\text{bias}} + \underbrace{\text{Var}(y)}_{\text{variance}} + \underbrace{\text{Var}(t)}_{\text{Bayes error}}$$

We 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

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.