Outline

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

```
width > 6.5cm?

height > 9.5cm?

height > 6.0cm?

Yes

Yes

Yes

Yes

leaf specifies class label.

No

No

No

No

edge specifies feature value

test a feature

Yes

Yes

Yes

Yes

Yes

Yes

Yes

Yes

No

No

No

No

```

Intro ML (UofT)  CSC311-Lec02

6 / 54
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.

![Decision Tree Diagram]

Test example

- width > 6.5cm?
  - Yes
    - height > 9.5cm?
      - Yes
        - [Lemon]
      - No
        - [Orange]
  - No
    - height > 6.0cm?
      - Yes
        - [Orange]
      - No
        - [Lemon]
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

- 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.

```
Patrons?
None   Some   Full
  F     T

WaitEstimate?
>60  30–60  10–30  0–10
  F    T

Alternate?
No   Yes
  F    T

Reservation?       Fri/Sat?
No   Yes   No   Yes
  F     T     F     T

Hungry?
No   Yes
  F    T

Alternate?
No   Yes
  F    T

Raining?
No   Yes
  F    T
```
**Decision Trees—Discrete Features**

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

<table>
<thead>
<tr>
<th>Example</th>
<th>Input Attributes</th>
<th>Goal</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Yes, No, No, Yes, Some, $$ $$ $$, No, Yes, French, 0–10</td>
<td>$y_1$ = Yes</td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>Yes, No, No, Yes, Full, $, No, No, Thai, 30–60</td>
<td>$y_2$ = No</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>No, Yes, No, No, Some, $, No, No, Burger, 0–10</td>
<td>$y_3$ = Yes</td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td>Yes, No, Yes, Yes, Full, $, Yes, No, Thai, 10–30</td>
<td>$y_4$ = Yes</td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td>Yes, No, Yes, No, Full, $$ $$ $$, No, Yes, French, &gt;60</td>
<td>$y_5$ = No</td>
<td></td>
</tr>
<tr>
<td>$x_6$</td>
<td>No, Yes, No, Yes, Some, $$, Yes, Yes, Italian, 0–10</td>
<td>$y_6$ = Yes</td>
<td></td>
</tr>
<tr>
<td>$x_7$</td>
<td>No, Yes, No, No, None, $, Yes, No, Burger, 0–10</td>
<td>$y_7$ = No</td>
<td></td>
</tr>
<tr>
<td>$x_8$</td>
<td>No, No, No, Yes, Some, $$, Yes, Yes, Thai, 0–10</td>
<td>$y_8$ = Yes</td>
<td></td>
</tr>
<tr>
<td>$x_9$</td>
<td>No, Yes, Yes, No, Full, $, Yes, No, Burger, &gt;60</td>
<td>$y_9$ = No</td>
<td></td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>Yes, Yes, Yes, Yes, Full, $$ $$ $$, No, Yes, Italian, 10–30</td>
<td>$y_{10}$ = No</td>
<td></td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>No, No, No, No, None, $, No, No, Thai, 0–10</td>
<td>$y_{11}$ = No</td>
<td></td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>Yes, Yes, Yes, Yes, Full, $, No, No, Burger, 30–60</td>
<td>$y_{12}$ = Yes</td>
<td></td>
</tr>
</tbody>
</table>

**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).
<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>0–10</td>
</tr>
<tr>
<td>x2</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
</tr>
<tr>
<td>x3</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
</tr>
<tr>
<td>x4</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
</tr>
<tr>
<td>x5</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>$$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
</tr>
<tr>
<td>x6</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Italian</td>
<td>0–10</td>
</tr>
<tr>
<td>x7</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
</tr>
<tr>
<td>x8</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>x9</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
</tr>
<tr>
<td>x10</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$$$</td>
<td>No</td>
<td>Yes</td>
<td>Italian</td>
<td>10–30</td>
</tr>
<tr>
<td>x11</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
</tr>
<tr>
<td>x12</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
</tr>
</tbody>
</table>

**Goal**

WillWait

<table>
<thead>
<tr>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
<th>y5</th>
<th>y6</th>
<th>y7</th>
<th>y8</th>
<th>y9</th>
<th>y10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Patron**

- Yes: 1, 3, 6, 8, 12
- No: 2, 5, 7, 9, 10, 11

**Hungry**

- Yes: 4, 12
- No: 2, 5, 9
Algorithm 1 Decision Tree Learner (examples, features)

1: if all examples are in the same class then
2: return the class label.
3: else if no features left then
4: return the majority decision.
5: else if no examples left then
6: return the majority decision at the parent node.
7: else
8: choose a feature $f$.
9: for each value $v$ of feature $f$ do
10: build edge with label $v$.
11: build sub-tree using examples where the value of $f$ is $v$.

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 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!

- oranges
- lemons
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

- 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.
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

Two biased coins. How biased are they?

Each coin is a binary random variable with outcomes Heads (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 ... ?

16

versus

8

10

0

1

2

Intro ML (UofT)
Quantifying Uncertainty

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

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

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.

$P = 0 \implies -0 \log 0 - 1 \log 1 = 0$
Quantifying Uncertainty

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

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**

[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 $X$.

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

Is its width $> 6.5\text{cm}$? Lemon or orange.
Entropy of a Joint Distribution

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

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

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

$= - \frac{24}{100} \log_2 \frac{24}{100} - \frac{1}{100} \log_2 \frac{1}{100} - \frac{25}{100} \log_2 \frac{25}{100} - \frac{50}{100} \log_2 \frac{50}{100}$

$\approx 1.56 \text{bits}$
Conditional Entropy

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

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

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

\[
H(Y|X = x) = -\sum_{y \in Y} p(y|x) \log_2 p(y|x)
\]

\[
= -\frac{24}{25} \log_2 \frac{24}{25} - \frac{1}{25} \log_2 \frac{1}{25}
\]

\[
\approx 0.24 \text{bits} \quad \text{low uncertainty}
\]

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

$$H(Y|X) = \mathbb{E}_x [H(Y|x)]$$

$$= \sum_{x \in X} p(x) H(Y|X = x)$$

$$= \sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log_2 p(y|x)$$
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: \( X = \{ \text{Raining, Not raining} \}, \ Y = \{ \text{Cloudy, Not cloudy} \} \)

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

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

\[
H(Y|X) = \sum_{x \in X} p(x)H(Y|X = x)
\]

\[
= \frac{1}{4}H(\text{cloudy}|\text{is raining}) + \frac{3}{4}H(\text{cloudy}|\text{not raining})
\]

\[
\approx 0.75 \text{ bits}
\]

\[
\approx 0.24 \text{ bits}
\]
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

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

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 $X$

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

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

Entropy of class outcome before split:
\[ H(Y) = -\frac{2}{7} \log_2\left(\frac{2}{7}\right) - \frac{5}{7} \log_2\left(\frac{5}{7}\right) \approx 0.86 \]

Conditional entropy of class outcome after split:
\[ H(Y|left) \approx 0.81, \quad H(Y|right) \approx 0.92 \]

Information Gain (split):
\[ IG(split) \approx 0.86 - \left(\frac{4}{7} \cdot 0.81 + \frac{3}{7} \cdot 0.92\right) \approx 0.006 \]
\[ H(Y) = -\frac{2}{7} \log_2 \frac{2}{7} - \frac{5}{7} \log_2 \frac{5}{7} \]

Left:
- Blue: 1 \((\frac{1}{4}, \frac{3}{4})\)
- Red: 3

Right:
- Blue: 1 \((\frac{1}{3}, \frac{2}{3})\)
- Red: 2

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

\[ IG(\text{split}) = H(Y) - H(Y | \text{split}) \]
What is the information gain of split A? Very informative!

Entropy of class outcome before split:

\[ H(Y) = -\frac{2}{7} \log_2\left(\frac{2}{7}\right) - \frac{5}{7} \log_2\left(\frac{5}{7}\right) \approx 0.86 \]

Conditional entropy of class outcome after split:

\[ H(Y|left) = 0, \quad H(Y|right) \approx 0.97 \]

\[ IG(split) \approx 0.86 - \left(\frac{2}{7} \cdot 0 + \frac{5}{7} \cdot 0.97\right) \approx 0.17!! \]
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

### 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).

### Input Attributes

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>0–10</td>
</tr>
<tr>
<td>x₂</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>30–60</td>
</tr>
<tr>
<td>x₃</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Some</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0–10</td>
</tr>
<tr>
<td>x₄</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Thai</td>
<td>10–30</td>
</tr>
<tr>
<td>x₅</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
</tr>
<tr>
<td>x₆</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Italian</td>
<td>0–10</td>
</tr>
<tr>
<td>x₇</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>0–10</td>
</tr>
<tr>
<td>x₈</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>x₉</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>&gt;60</td>
</tr>
<tr>
<td>x₁₀</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$$$</td>
<td>No</td>
<td>Yes</td>
<td>Italian</td>
<td>10–30</td>
</tr>
<tr>
<td>x₁₁</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>x₁₂</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>30–60</td>
</tr>
</tbody>
</table>

### Goal

- **WillWait**
  - \( y_1 = \text{Yes} \)
  - \( y_2 = \text{No} \)
  - \( y_3 = \text{Yes} \)
  - \( y_4 = \text{Yes} \)
  - \( y_5 = \text{Yes} \)
  - \( y_6 = \text{Yes} \)
  - \( y_7 = \text{No} \)
  - \( y_8 = \text{Yes} \)
  - \( y_9 = \text{No} \)
  - \( y_{10} = \text{No} \)
  - \( y_{11} = \text{No} \)
  - \( y_{12} = \text{Yes} \)

[from: Russell & Norvig]
Feature Selection

\[ IG(Y) = H(Y) - H(Y|X) \]

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

\[ IG(Patrons) = 1 - \left[ \frac{2}{12} H(0, 1) + \frac{4}{12} H(1, 0) + \frac{6}{12} H\left(\frac{2}{6}, \frac{4}{6}\right) \right] \approx 0.541 \]
Which Tree is Better? Vote!

- **Patrons?**
  - None: No
  - Some: Yes
  - Full: Yes

- **Wait Estimate?**
  - >60: No
  - 30-60: No
  - 10-30: No
  - 0-10: Yes

- **Hungry?**
  - No: Yes
  - Yes: No

- **Type?**
  - French: Yes
  - Italian: No
  - Thai: Yes
  - Burger: Yes

- **Fri/Sat?**
  - No: Yes
  - Yes: No
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 ➞ 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
- More interpretable

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.
1 Introduction

2 Decision Trees

3 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.
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

- \( p_{\text{sample}} \) is a data generating distribution.
  For lemons and oranges, \( p_{\text{sample}} \) 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 \( p_{\text{sample}} \).
- We can sample lots of training sets independently from \( p_{\text{sample}} \).
Basic Setup for Classification

- Run our learning algorithm on each training set, and compute its prediction $y$ at the query point $x$.
- 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

fit to dataset 1

fit to dataset 2

fit to dataset 3

query location

lots of fits

histogram of y
Basic Setup

- Fix a query point \( x \).
- Repeat:
  - Sample a random training dataset \( D \) i.i.d. from the data generating distribution \( p_{\text{sample}} \).
  - Run the learning algorithm on \( D \) to get a prediction \( y \) at \( x \).
  - Sample the (true) target from the conditional distribution \( p(t|x) \).
  - Compute the loss \( L(y, t) \).

Comments:

- Notice: \( y \) is independent of \( t \). (Why?)
  - Produced \( y \) using the training set \( D \).
  - Cannot recover \( p_{\text{sample}} \) from \( D \). (If we could, we don't need \( D \) anymore.)
$P_{\text{sample}}$ → $D_1$ → $y_1$ → $L(y_1, t)$

$P_{\text{sample}}$ → $D_2$ → $y_2$ → $L(y_2, t)$

$P_{\text{sample}}$ → $D_3$ → $y_3$ → $L(y_3, t)$

$\ldots$
Basic Setup

- Fix a query point $x$.
- Repeat:
  - Sample a random training dataset $D$ i.i.d. from the data generating distribution $p_{\text{sample}}$.
  - Run the learning algorithm on $D$ to get a prediction $y$ at $x$.
  - Sample the (true) target from the conditional distribution $p(t|x)$.
  - Compute the loss $L(y,t)$.

Comments:

- Notice: $y$ is independent of $t$. (Why?)
- This gives a distribution over the loss at $x$, with expectation $\mathbb{E}[L(y,t) | x]$.
- For each query point $x$, the expected loss is different. We are interested in minimizing the expectation of this with respect to $x \sim p_{\text{sample}}$. 
Choosing a prediction $y$

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

Claim: $y^\ast = \mathbb{E}[t \mid x]$ is the best possible prediction.

Proof:

$$
\mathbb{E}[(y - t)^2 \mid x] = \mathbb{E}[y^2 - 2yt + t^2 \mid x] = y^2 \mathbb{E}[t \mid x] + \mathbb{E}[t^2 \mid x] - 2y \mathbb{E}[t \mid x] + \mathbb{E}[t^2 \mid x] = y^2 \mathbb{E}[t \mid x] + \mathbb{E}[t^2 \mid x] + \text{Var}[t \mid x] = (y - y^\ast)^2 + \text{Var}[t \mid x].
$$
Choosing a prediction $y$

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

\[
\mathbb{E}[(y - t)^2 \mid x] = \mathbb{E}[y^2 - 2yt + t^2 \mid x]
\]
\[
= y^2 - 2y\mathbb{E}[t \mid x] + \mathbb{E}[t^2 \mid x]
\]
\[
= y^2 - 2y\mathbb{E}[t \mid x] + \mathbb{E}[t \mid x]^2 + \text{Var}[t \mid x]
\]
\[
= y^2 - 2yy_* + y_*)^2 + \text{Var}[t \mid x]
\]
\[
= (y - y_*)^2 + \text{Var}[t \mid x]
\]
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.)

(we choose $y$.)

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

(Var $[x] = E[x^2] - (E[x])^2$)

$$= y^2 - 2yE[t|x] + E[t^2|x]$$

(let $y^* = E[t|x]$)

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

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

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

\[ \mathbb{E}[(y - t)^2 | x] = (y - y_*)^2 + \text{Var}[t | x] \]

- 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 \( p(t | x) \) 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 $x$ for clarity):

$$y^* = \mathbb{E}[t|x]$$

$$\mathbb{E}[(y - t)^2] = \mathbb{E}[(y - y^*)^2] + \text{Var}(t)$$

expanding the square.

linearity of $\mathbb{E}$.

$$\text{Var}[y] = \mathbb{E}[y^2] - (\mathbb{E}[y])^2$$

regrouping terms.

$$= \mathbb{E}[y^2] - 2y^* \mathbb{E}[y] + \mathbb{E}[y]^2 + \text{Var}(t)$$

$$= y^2 - 2y^* \mathbb{E}[y] + \mathbb{E}[y]^2 + \text{Var}(y) + \text{Var}(t)$$

$$= (y^* - \mathbb{E}[y])^2 + \text{Var}(y) + \text{Var}(t)$$

bias variance \hspace{1cm} \text{Bayes error}
Bayes Optimality

\[ \mathbb{E}[(y - t)^2] = (y - \mathbb{E}[y])^2 + \text{Var}(y) + \text{Var}(t) \]

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

\[ \text{large bias} \rightarrow \text{underfitting} \]

\[ \text{large variance} \rightarrow \text{overfitting} \]
Bias and Variance

- Throwing darts = predictions for each draw of a dataset

- Be careful, what doesn’t this capture?
  - We average over points \( x \) from the data distribution.