Course Overview
WHAT IS MACHINE LEARNING?
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

• Perception
• Reasoning
• Control / Motion / Manipulation
• Planning
• Communication
• Creativity
• Learning
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

• Perception
• Reasoning
• Control / Motion / Manipulation
• Planning
• Communication
• Creativity
• Learning
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

• Perception

• Reasoning

• Control / Motion / Manipulation

• Planning

• Communication

• Creativity

• Learning
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

- Perception
- Reasoning
- Control / Motion / Manipulation
- Planning
- Communication
- Creativity
- Learning
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

• Perception
• Reasoning
• Control / Motion / Manipulation
• Planning
• Communication
• Creativity
• Learning
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:
- Perception
- Reasoning
- Control / Motion / Manipulation
- Planning
- Communication
- Creativity
- Learning
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

• Perception
• Reasoning
• Control / Motion / Manipulation
• Planning
• Communication
• Creativity
• Learning

“Deep Style” from https://deepdreamgenerator.com/#gallery
Artificial Intelligence

The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

• Perception
• Reasoning
• Control / Motion / Manipulation
• Planning
• Communication
• Creativity
• Learning
What is Machine Learning?

The goal of this course is to provide you with a toolbox:

- Machine Learning
- Statistics
- Probability
- Computer Science
- Optimization
What is ML?

- Machine Learning
- Optimization
- Statistics
- Probability
- Calculus
- Measure Theory
- Linear Algebra
- Computer Science
- Domain of Interest
What is ML?

**Speech Recognition**
1. Learning to recognize spoken words

   **THEN**
   "... the SPHINX system (e.g. Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal... neural network methods... hidden Markov models..."

   **NOW**
   "... the Google Duplex system (2018) uses state-of-the-art neural networks to recognize spoken words and sentences..."

   (Mitchell, 1997)

**Robotics**
2. Learning to drive an autonomous vehicle

   **THEN**
   "... the ALVINN system (Perrett, 1989) has used its learned strategies to drive unmanned in 70 mile per hour on 90 miles of public highways among other cars..."

   **NOW**
   "... the Google self-driving car (2017) automatically navigates city streets using advanced sensor fusion and machine learning algorithms..."

   (Mitchell, 1997)

**Games / Reasoning**
3. Learning to beat the masters at board games

   **THEN**
   "... the world's top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself..."

   **NOW**
   "... the AlphaGo system (2015) uses deep neural networks to play Go at a superhuman level...

   (Mitchell, 1997)

**Computer Vision**
4. Learning to recognize images

   **THEN**
   "... The recognizer is a convolution network that can be spatially replicated from the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level error...

   **NOW**
   "... the Google Inception network (2014) is a deep convolution neural network that recognizes images at a superhuman level...

   (LeCun et al., 1995)

**Learning Theory**
5. In what cases and how well can we learn?

   1. How many examples do we need to learn?
   2. How do we quantify our ability to generalize to unseen data?
   3. Which algorithms are better suited to specific learning settings?
1. Learning to recognize spoken words

“…the SPHINX system (e.g. Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal…neural network methods…hidden Markov models…”

(Mitchell, 1997)
2. Learning to drive an autonomous vehicle

**THEN**

“...the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars...”

(Mitchell, 1997)

**NOW**

[Image of an autonomous vehicle on a suburban street]

waymo.com
## 2. Learning to drive an autonomous vehicle

<table>
<thead>
<tr>
<th>THEN</th>
<th>NOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>“…the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars…”</td>
<td><img src="https://www.bloomberg.com/news/articles/2019-02-07/aurora-self-driving-startup-gets-funding-from-sequoia-amazon" alt="Self-driving car" /></td>
</tr>
</tbody>
</table>

(Mitchell, 1997)

2. Learning to drive an autonomous vehicle

“...the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars...”

(Mitchell, 1997)

Figure from https://locomation.ai/
3. Learning to beat the masters at board games

**THEN**

“…the world’s top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself…”

(Mitchell, 1997)

**NOW**

[Image of AlphaGo]
“...The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors....”

(LeCun et al., 1995)
Learning Theory

• 5. In what cases and how well can we learn?

Sample Complexity Results

**Definition 0.1.** The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

**Four Cases we care about…**

<table>
<thead>
<tr>
<th>Realizable</th>
<th>Agnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>**Finite $</td>
<td>\mathcal{H}</td>
</tr>
</tbody>
</table>

- $N > \frac{1}{2} \left[ \log(|\mathcal{H}|) + \log\left(\frac{1}{\delta}\right) \right]$ labeled examples are sufficient so that with probability $(1 - \delta)$ all $h \in \mathcal{H}$ with $R(h) > \epsilon$ have $\hat{R}(h) > 0$.

- $N > \frac{1}{2\epsilon^2} \left[ \log(|\mathcal{H}|) + \log\left(\frac{1}{\delta}\right) \right]$ labeled examples are sufficient so that with probability $(1 - \delta)$ for all $h \in \mathcal{H}$ we have that $|R(h) - \hat{R}(h)| < \epsilon$.

| Infinite $|\mathcal{H}|$ |

- $N = O\left(\frac{1}{\epsilon^2} \left[ \text{VC}(|\mathcal{H}|) + \log\left(\frac{1}{\delta}\right) \right] \right)$ labeled examples are sufficient so that with probability $(1 - \delta)$ all $h \in \mathcal{H}$ with $R(h) \geq \epsilon$ have $\hat{R}(h) > 0$.

- $N = O\left(\frac{1}{\epsilon^2} \left[ \text{VC}(|\mathcal{H}|) + \log\left(\frac{1}{\delta}\right) \right] \right)$ labeled examples are sufficient so that with probability $(1 - \delta)$ for all $h \in \mathcal{H}$ we have that $|R(h) - \hat{R}(h)| \leq \epsilon$.

---

1. How many examples do we need to learn?
2. How do we quantify our ability to generalize to unseen data?
3. Which algorithms are better suited to specific learning settings?
What is ML?

Speech Recognition
1. Learning to recognize spoken words

THEN
...the SPHINA system (e.g., Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal...hidden Markov models...

NOW
(Mitchell, 1997)

Robotics
2. Learning to drive an autonomous vehicle

THEN
...the ALVINN system (Pomerleau, 1989) has used its learned strategies to drive autonomously at 40 miles per hour for 90 miles on public highways among other cars...

NOW
(Mitchell, 1997)

Games / Reasoning
3. Learning to beat the masters at board games

THEN
...the world’s top chess computer program for backgammon, TD-GAMMON (Tesauro, 1992; 1995), learned its strategy by playing over one million practice games against itself...

NOW
(Mitchell, 1997)

Computer Vision
4. Learning to recognize images

THEN
...The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level error...

NOW
(LoCascio et al., 1995)

Learning Theory
5. In what cases and how well can we learn?

- How many examples do we need to learn?
- How do we quantify our ability to generalize to unseen data?
- Which algorithms are better suited to specific learning settings?
What is Machine Learning?

The goal of this course is to provide you with a toolbox:

- Machine Learning
- Statistics
- Probability
- Computer Science
- Optimization

To solve all the problems above and more
Societal Impacts of ML

What ethical responsibilities do we have as machine learning experts?

**Question:** What are the possible societal impacts of machine learning for each case below?

**Answer:**

1) Search results for news are optimized for ad revenue.

2) An autonomous vehicle is permitted to drive unassisted on the road.

3) A doctor is prompted by an intelligent system with a plausible diagnosis for her patient.
Societal Impacts of ML

A 72-year-old congressman goes back to school, pursuing a degree in AI

By Meagan Flynn

December 28, 2022 at 6:00 a.m. EST

Normally Don Beyer doesn’t bring his multivariable calculus textbook to work, but his final exam was coming up that weekend.

“And I’m running out of time,” he said, plopping the textbook and a scribbled notebook filled with esoteric-looking calculations on a coffee table in his office, “because I have all these—”

His phone was ringing. “I’ll be there,” Beyer told a colleague wondering when he would be returning to the House floor for votes.

It seemed study time would have to wait.

That’s been the story of the year for Beyer (D-Va.), who has been moonlighting as a student at George Mason University in pursuit of a master’s degree in machine learning while balancing his duties as a congressman. Beyer — a science wonk, economist and former car salesman — has been taking one class per semester in a slow but steady march toward the degree, with hopes of one day applying his artificial-intelligence knowledge to his legislative work as the technology evolves further.

Figure from https://www.washingtonpost.com/dc-md-va/2022/12/28/beyer-student-artificial-intelligence-degree/
## Learning Paradigms:
What data is available and when? What form of prediction?
- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- active learning
- imitation learning
- domain adaptation
- online learning
- density estimation
- recommender systems
- feature learning
- manifold learning
- dimensionality reduction
- ensemble learning
- distant supervision
- hyperparameter optimization

## Problem Formulation:
What is the structure of our output prediction?
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Binary Classification</td>
</tr>
<tr>
<td>categorical</td>
<td>Multiclass Classification</td>
</tr>
<tr>
<td>ordinal</td>
<td>Ordinal Classification</td>
</tr>
<tr>
<td>real</td>
<td>Regression</td>
</tr>
<tr>
<td>ordering</td>
<td>Ranking</td>
</tr>
<tr>
<td>multiple discrete</td>
<td>Structured Prediction</td>
</tr>
<tr>
<td>multiple continuous</td>
<td>(e.g. dynamical systems)</td>
</tr>
<tr>
<td>both discrete &amp; continuous</td>
<td>(e.g. mixed graphical models)</td>
</tr>
</tbody>
</table>

## Facets of Building ML Systems:
How to build systems that are robust, efficient, adaptive, effective?
1. Data prep
2. Model selection
3. Training (optimization / search)
4. Hyperparameter tuning on validation data
5. (Blind) Assessment on test data

## Theoretical Foundations:
What principles guide learning?
- probabilistic
- information theoretic
- evolutionary search
- ML as optimization

## Big Ideas in ML:
Which are the ideas driving development of the field?
- inductive bias
- generalization / overfitting
- bias-variance decomposition
- generative vs. discriminative
- deep nets, graphical models
- PAC learning
- distant rewards

## Application Areas:
Key challenges?
- NLP, Speech, Computer Vision, Robotics, Medicine, Search
Topics

• Foundations
  – Probability
  – MLE, MAP
  – Optimization

• Classifiers
  – KNN
  – Naïve Bayes
  – Logistic Regression
  – Perceptron
  – SVM

• Regression
  – Linear Regression

• Important Concepts
  – Kernels
  – Regularization and Overfitting
  – Experimental Design

• Unsupervised Learning
  – K-means / Lloyd’s method
  – PCA
  – EM / GMMs

• Neural Networks
  – Feedforward Neural Nets
  – Basic architectures
  – Backpropagation
  – CNNs, LSTMs

• Graphical Models
  – Bayesian Networks
  – HMMs
  – Learning and Inference

• Learning Theory
  – Statistical Estimation (covered right before midterm)
  – PAC Learning

• Other Learning Paradigms
  – Matrix Factorization
  – Reinforcement Learning
  – Information Theory
DEFINING LEARNING PROBLEMS
Well-Posed Learning Problems

Three components $<T,P,E>$:
1. Task, $T$
2. Performance measure, $P$
3. Experience, $E$

Definition of learning:
A computer program learns if its performance at task $T$, as measured by $P$, improves with experience $E$.

Definition from (Mitchell, 1997)
Example Learning Problems

Learning to beat the masters at **chess**

1. Task, $T$:

2. Performance measure, $P$:

3. Experience, $E$: 


Example Learning Problems

Learning to respond to voice commands (Siri)

1. Task, $T$:

2. Performance measure, $P$:

3. Experience, $E$: 
Solution #1: Expert Systems

Over 20 years ago, we had rule-based systems:

1. Put a bunch of linguists in a room
2. Have them think about the structure of their native language and write down the rules they devise

<table>
<thead>
<tr>
<th>Question</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Give me directions to Starbucks</td>
<td>If: “give me directions to X” Then: directions(here, nearest(X))</td>
</tr>
<tr>
<td>How do I get to Starbucks?</td>
<td>If: “how do i get to X” Then: directions(here, nearest(X))</td>
</tr>
<tr>
<td>Where is the nearest Starbucks?</td>
<td>If: “where is the nearest X” Then: directions(here, nearest(X))</td>
</tr>
</tbody>
</table>
Solution #1: Expert Systems

- Over 20 years ago, we had rule-based systems:
  1. Put a bunch of linguists in a room
  2. Have them think about the structure of their native language and write down the rules they devise

<table>
<thead>
<tr>
<th>I need directions to Starbucks</th>
</tr>
</thead>
<tbody>
<tr>
<td>If: “I need directions to X”</td>
</tr>
<tr>
<td>Then: directions(here, nearest(X))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Starbucks directions</th>
</tr>
</thead>
<tbody>
<tr>
<td>If: “X directions”</td>
</tr>
<tr>
<td>Then: directions(here, nearest(X))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Is there a Starbucks nearby?</th>
</tr>
</thead>
<tbody>
<tr>
<td>If: “Is there an X nearby”</td>
</tr>
<tr>
<td>Then: directions(here, nearest(X))</td>
</tr>
</tbody>
</table>
Capturing the Knowledge of Experts

Solution #2: Annotate Data and Learn

• Experts:
  – Very good at answering questions about specific cases
  – Not very good at telling HOW they do it

• 1990s: So why not just have them tell you what they do on SPECIFIC CASES and then let MACHINE LEARNING tell you how to come to the same decisions that they did
Solution #2: Annotate Data and Learn

1. Collect raw sentences \( \{x^{(1)}, \ldots, x^{(n)}\} \)
2. Experts annotate their meaning \( \{y^{(1)}, \ldots, y^{(n)}\} \)

<table>
<thead>
<tr>
<th>x^{(1)}: How do I get to Starbucks?</th>
<th>x^{(3)}: Send a text to John that I’ll be late</th>
</tr>
</thead>
<tbody>
<tr>
<td>y^{(1)}: directions(here, nearest(Starbucks))</td>
<td>y^{(3)}: txtmsg(John, I’ll be late)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x^{(2)}: Show me the closest Starbucks</th>
<th>x^{(4)}: Set an alarm for seven in the morning</th>
</tr>
</thead>
<tbody>
<tr>
<td>y^{(2)}: map(nearest(Starbucks))</td>
<td>y^{(4)}: setalarm(7:00AM)</td>
</tr>
</tbody>
</table>
Example Learning Problems

Learning to **respond to voice commands (Siri)**

1. **Task,** $T$:  
   predicting action from speech  
2. **Performance measure,** $P$:  
   percent of correct actions taken in user pilot study  
3. **Experience,** $E$:  
   examples of (speech, action) pairs
Problem Formulation

• Often, the same task can be formulated in more than one way:
• Ex: Loan applications
  – creditworthiness/score (regression)
  – probability of default (density estimation)
  – loan decision (classification)

Problem Formulation:
What is the structure of our output prediction?

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Model Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Binary Classification</td>
</tr>
<tr>
<td>categorical</td>
<td>Multiclass Classification</td>
</tr>
<tr>
<td>ordinal</td>
<td>Ordinal Classification</td>
</tr>
<tr>
<td>real</td>
<td>Regression</td>
</tr>
<tr>
<td>ordering</td>
<td>Ranking</td>
</tr>
<tr>
<td>multiple discrete</td>
<td>Structured Prediction</td>
</tr>
<tr>
<td>multiple continuous</td>
<td>(e.g. dynamical systems)</td>
</tr>
<tr>
<td>both discrete &amp; cont.</td>
<td>(e.g. mixed graphical models)</td>
</tr>
</tbody>
</table>
Well-posed Learning Problems

In-Class Exercise
1. Select a **task**, T
2. Identify **performance measure**, P
3. Identify **experience**, E
4. Report ideas back to rest of class

Example Tasks
- Identify objects in an image
- Translate from one human language to another
- Recognize speech
- Assess risk (e.g. in loan application)
- Make decisions (e.g. in loan application)
- Assess potential (e.g. in admission decisions)
- Categorize a complex situation (e.g. medical diagnosis)
- Predict outcome (e.g. medical prognosis, stock prices, inflation, temperature)
- Predict events (default on loans, quitting school, war)
- Plan ahead under perfect knowledge (chess)
- Plan ahead under partial knowledge (poker, bridge)

Examples from Roni Rosenfeld
SUPERVISED LEARNING

(without any math!)
Building a Trash Classifier

- Suppose the ask CMU to build a robot for collecting trash along Pittsburgh’s rivers
- You are tasked with building a classifier that detects whether an object is a piece of trash (+) or not a piece of trash (-)
- The robot can detect an object’s color, sound, and weight
- You manually annotate the following dataset based on objects you find

<table>
<thead>
<tr>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
</tr>
<tr>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>
WARNING!

Like many fields, Machine Learning is riddled with copious amounts of technical jargon!

For many terms we’ll define in this class, you’ll find four or five different terms in the literature that refer to the same thing.
Supervised Binary Classification

- *Def:* an **example** contains a **label** (aka. class) and **features** (aka. point or attributes)

- *Def:* a **labeled dataset** consists of rows, where each row is an example

- *Def:* an **unlabeled dataset** only has **features**

### One example:

<table>
<thead>
<tr>
<th>label</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>trash?</td>
<td>color</td>
</tr>
<tr>
<td>-</td>
<td>brown</td>
</tr>
</tbody>
</table>

### Labeled Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>label</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>trash?</td>
<td>color</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
</tr>
</tbody>
</table>

### Unlabeled Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>color</td>
</tr>
<tr>
<td>1</td>
<td>brown</td>
</tr>
<tr>
<td>2</td>
<td>clear</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
</tr>
</tbody>
</table>
Supervised Binary Classification

- **Def:** an example contains a label (aka. class) and features (aka. point or attributes)
- **Def:** a labeled dataset consists of rows, where each row is an example
- **Def:** an unlabeled dataset only has features

### Labeled Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

### Unlabeled Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

### One example:

<table>
<thead>
<tr>
<th>label</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>trash?</td>
<td>color</td>
</tr>
<tr>
<td>-</td>
<td>brown</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>high</td>
</tr>
</tbody>
</table>
Supervised Binary Classification

- **Def:** An example contains a label (aka. class) and features (aka. point or attributes)
- **Def:** A labeled dataset consists of rows, where each row is an example
- **Def:** An unlabeled dataset only has features

### Training Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>label</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
<td></td>
</tr>
</tbody>
</table>

### Test Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>label</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
<td></td>
</tr>
</tbody>
</table>
Supervised Binary Classification

- **Def:** predictions are the output of a trained classifier
- **Def:** error rate is the proportion of examples on which we predicted the wrong label

**Test Predictions:**

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

**(Unlabeled) Test Dataset:**

<table>
<thead>
<tr>
<th>index</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>color</td>
</tr>
<tr>
<td>1</td>
<td>brown</td>
</tr>
<tr>
<td>2</td>
<td>clear</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
</tr>
</tbody>
</table>

- **Def:** a **classifier** is a function that takes in features and predicts a label
- **Def:** a **training dataset** is a labeled dataset used to learn a classifier
- **Def:** a **test dataset** is a labeled dataset used to evaluate a classifier
Supervised Binary Classification

- **Def:** predictions are the output of a trained classifier.
- **Def:** error rate is the proportion of examples on which we predicted the wrong label.

### Test Predictions:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

**error rate = 1/3**

### (Labeled) Test Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

- **Def:** a **classifier** is a function that takes in features and predicts a label.
- **Def:** a **training dataset** is a labeled dataset used to **learn** a classifier.
- **Def:** a **test dataset** is a labeled dataset used to **evaluate** a classifier.
Supervised Binary Classification

- **Step 1: training**
  - Given: labeled **training dataset**
  - Goal: learn a **classifier** from the training dataset

- **Step 2: prediction**
  - Given: unlabeled **test dataset**
    - Given: learned classifier
    - Goal: **predict** a label for each instance

- **Step 3: evaluation**
  - Given: **predictions** from Step II
    - Given: labeled **test dataset**
    - Goal: compute the **test error rate** (i.e. error rate on the test dataset)

Error rate = 1/3
Supervised Binary Classification

• Step 1: training
  – Given: labeled training dataset
  – Goal: learn a classifier from the training dataset

• Step 2: prediction
  – Given: unlabeled test dataset
    : learned classifier
  – Goal: predict a label for each instance

• Step 3: evaluation
  – Given: predictions from Step II
    : labeled test dataset
  – Goal: compute the test error rate (i.e. error rate on the test dataset)

Error rate = 1/3
Supervised Binary Classification

• Step 1: training
  – Given: labeled **training dataset**
  – Goal: learn a **classifier** from the training dataset

• Step 2: prediction
  – Given: unlabeled **test dataset**
    → learned classifier
  – Goal: **predict** a label for each instance

• Step 3: evaluation
  – Given: **predictions** from Phase II
    → labeled **test dataset**
  – Goal: compute the **test error rate** (i.e., error rate on the test dataset)

\[
\text{error rate} = \frac{1}{3}
\]
Supervised Binary Classification

- **Step 1: training**
  - *Given*: labeled training dataset
  - *Goal*: learn a classifier from the training dataset

- **Step 2: prediction**
  - *Given*: unlabeled test dataset
  - *Given*: learned classifier
  - *Goal*: predict a label for each instance

- **Step 3: evaluation**
  - *Given*: predictions from Phase II
  - *Given*: labeled test dataset
  - *Goal*: compute the test error rate (i.e. error rate on the test dataset)

**Key question in Machine Learning:**
How do we learn the classifier from data?
Random Classifier

The random classifier takes in the features and always predicts a random label.

... this is a terrible idea. It completely ignores the training data!

<table>
<thead>
<tr>
<th>Training Dataset:</th>
<th>label</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>trash?</td>
<td>color</td>
</tr>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Predictions:</th>
<th>predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>trash?</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>+</td>
</tr>
</tbody>
</table>

Test Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

error rate = 2/3
Random Classifier

The random classifier takes in the features and always predicts a random label.

…this is a terrible idea. It completely ignores the training data!

Classifier features → random!

Test Predictions:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

error rate = 1/3

Test Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

Training Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>
The **random classifier** takes in the features and always predicts a random label.

...this is a terrible idea. It completely **ignores the training data**!

---

**Training Dataset:**

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

**Test Predictions:**

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>+</td>
</tr>
</tbody>
</table>

**Test Dataset:**

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

error rate = 3/3
Majority Vote Classifier

The **majority vote classifier** takes in the features and always predicts the **most common label** in the training dataset.

... this is still a pretty bad idea. It completely ignores the features!

**Classifier features → always predict “-”**

---

### Training Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

### Test Predictions:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

**error rate = 1/3**

---

### Test Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>
Majority Vote Classifier

The **majority vote classifier** takes in the features and always predicts the **most common label** in the training dataset.

...this is still a pretty bad idea. It completely **ignores the features**!

---

**Training Dataset:**

<table>
<thead>
<tr>
<th>index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

**Error rate = 2/5**
Majority Vote Classifier

• Step 1: training
  – **Given:** labeled **training dataset**
  – **Goal:** learn a **classifier** from the training dataset

• Step 2: prediction
  – **Given:** unlabeled **test dataset**
    : learned classifier
  – **Goal:** predict a label for each instance

• Step 3: evaluation
  – **Given:** predictions from Step II
    : labeled **test dataset**
  – **Goal:** compute the **test error rate** (i.e. error rate on the test dataset)

---

**Table:**

**Training Dataset:**

<table>
<thead>
<tr>
<th>Index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>green</td>
<td>crinkly</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>grey</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>clear</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>green</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

**Table:**

**Unlabeled Test Dataset:**

<table>
<thead>
<tr>
<th>Index</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

**Table:**

**Labeled Test Dataset:**

<table>
<thead>
<tr>
<th>Index</th>
<th>trash?</th>
<th>color</th>
<th>sound</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

**Test Predictions:**

<table>
<thead>
<tr>
<th>Index</th>
<th>trash?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

**Classifier:**

features → always predict “-”

**Error Rate:**

error rate = 1/3
SYLLABUS HIGHLIGHTS
Syllabus Highlights

The syllabus is located on the course webpage:

http://www.cs.cmu.edu/~mgormley/courses/10601

or

http://mlcourse.org

The course policies are required reading.
Syllabus Highlights

• **Grading**: 50% homework, 15% exam 1, 15% exam 2, 15% exam 3, 5% participation
• **Exam 1**: evening, Thu, Feb. 16
• **Exam 2**: evening, Thu, Mar. 30
• **Exam 3**: final exam week, date TBD by registrar
• **Homework**: 3 written and 6 written + programming (Python)
  – 6 grace days for homework assignments
  – Late submissions: 75% day 1, 50% day 2, 25% day 3
  – No submissions accepted after 3 days w/o extension; HW3, HW6, HW9 only 2 days
  – Extension requests: for emergencies, see syllabus
• **Recitations**: Fridays, same time/place as lecture (optional, interactive sessions)
• **Readings**: required, online PDFs, recommended for after lecture
• **Technologies**: Piazza (discussion), Gradescope (homework), Google Forms (polls)
• **Academic Integrity**:
  – Collaboration encouraged, but must be documented
  – Solutions must always be written independently
  – No re-use of found code / past assignments
  – Severe penalties (e.g. -100%)
• **Office Hours**: posted on Google Calendar on “Office Hours” page
Lectures

• You should ask lots of questions
  – Interrupting (by raising a hand) to ask your question is strongly encouraged
  – Asking questions later (or in real time) on Piazza is also great

• When I ask a question…
  – I want you to answer
  – Even if you don’t answer, think it through as though I’m about to call on you

• Interaction improves learning (both in-class and at my office hours)
Textbooks

You are not required to read a textbook, but it will help immensely!
## Where can I find...?

<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture</th>
<th>Readings</th>
<th>Announcements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed, 3-Feb</td>
<td>Lecture 2: Decision Trees, Overfitting [Slides]</td>
<td><a href="#">Decision Trees</a>, Hal Daumé III (2017). CIML, Chapter 1.</td>
<td>HW1 out</td>
</tr>
<tr>
<td>Fri, 5-Feb</td>
<td>Recitation: HW1 [Handout] [Solutions]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon, 8-Feb</td>
<td>Lecture 3: Generalizing from examples - the Big Picture [Slides] [Poll]</td>
<td><a href="#">Limits of Learning</a>, Hal Daumé III (2017). CIML, Chapter 2.</td>
<td></td>
</tr>
<tr>
<td>Wed, 10-Feb</td>
<td>Lecture 4: k-Nearest Neighbors [Slides] [Whiteboard] [Poll]</td>
<td><a href="#">Geometry and Nearest Neighbors</a>, Hal Daumé III (2017). CIML, Chapter 3.</td>
<td>HW1 due HW2 out</td>
</tr>
<tr>
<td>Fri, 12-Feb</td>
<td>Recitation: HW2 [Handout] [Solutions]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon, 15-Feb</td>
<td>Lecture 5: Model Selection [Slides] [Whiteboard] [Poll]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed, 17-Feb</td>
<td>Lecture 6: Perceptron [Slides] [Whiteboard] [Poll]</td>
<td><a href="#">The Perceptron</a>, Hal Daumé III (2017). CIML, Chapter 4.</td>
<td>HW1 solution session (Thursday)</td>
</tr>
</tbody>
</table>
Where can I find...?

Introduction to Machine Learning
Assignments

There will be 8 homework assignments during the semester in addition to the exams. The assignments will consist of both theoretical and practical problems. Assignments will be released via a Piazza announcement explaining where to find the handout, starter code, LaTeX template, etc.

- Homework 1: Background Material (written / programming)
  Handout
- Homework 2: Decision Trees (written / programming)
  Handout
- Homework 3: KNN, Perceptron, and Linear Regression (written)
  Handout
- Mock Exam 1:
  Handout and Solution
- Homework 4: Logistic Regression (written / programming)
  Handout
- Homework 5: Neural Networks (written / programming)
  Handout
- Homework 6: Neural Networks and Reinforcement Learning (written / programming)
  Handout
- Homework 7: Graphical Models (written / programming)
In-Class Polls

Q: How do these In-Class Polls work?

A: Don’t worry about it for today. We won’t use them until the second week of class, i.e. the third lecture.

Details are on the syllabus.
PREREQUISITES
Prerequisites

What they are:

• Significant programming experience (15-122)
  – Written programs of 100s of lines of code
  – Comfortable learning a new language
• Probability and statistics (36-217, 36-225, etc.)
• Mathematical maturity: discrete mathematics (21-127, 15-151), linear algebra, and calculus
Prerequisites

What if you need additional review?
• Consider first taking 10-606/607: Mathematical/Computational Foundations for Machine Learning
• More details here: https://www.cs.cmu.edu/~pvirtue/10606/

How to describe 606/607 to a friend

606/607 is...

- a formal presentation of mathematics and computer science...
- motivated by (carefully chosen) real-world problems that arise in machine learning...
- where the broader picture of how those problems arise is treated somewhat informally.
Oh, the Places You’ll Use Probability!

Supervised Classification

• Naïve Bayes

\[ p(y|x_1, x_2, \ldots, x_n) = \frac{1}{Z} p(y) \prod_{i=1}^{n} p(x_i|y) \]

• Logistic regression

\[ P(Y = y|X = x; \theta) = p(y|x; \theta) = \frac{\exp(\theta_y \cdot f(x))}{\sum_{y'} \exp(\theta_{y'} \cdot f(x))} \]

Note: This is just motivation – we’ll cover these topics later!
Oh, the Places You’ll Use Probability!

ML Theory
(Example: Sample Complexity)

- **Goal:** \( h \) has small error over \( D \).

  \[
  \text{True error: } err_D(h) = \Pr_{x \sim D} (h(x) \neq c^*(x))
  \]

  How often \( h(x) \neq c^*(x) \) over future instances drawn at random from \( D \)

- **But, can only measure:**

  \[
  \text{Training error: } err_S(h) = \frac{1}{m} \sum_i I(h(x_i) \neq c^*(x_i))
  \]

  How often \( h(x) \neq c^*(x) \) over training instances

**Sample complexity:** **bound** \( err_D(h) \) in terms of \( err_S(h) \)

*Note: This is just motivation – we’ll cover these topics later!*
Oh, the Places You’ll Use Probability!

Deep Learning
(Example: Deep Bi-directional RNN)

Note: This is just motivation – we’ll cover these topics later!
Oh, the Places You’ll Use Probability!

Graphical Models

- **Hidden Markov Model (HMM)**

- **Conditional Random Field (CRF)**

Note: This is just motivation – we’ll cover these topics later!
Prerequisites

What if I’m not sure whether I meet them?
• Don’t worry: we’re not sure either
• However, we’ve designed a way to assess your background knowledge so that you know what to study!
Syllabus Highlights

Background Test

• **When**: Fri, Jan 20, in-class
• **Where**: this lecture hall
• **What**: prerequisite material (probability, statistics, linear algebra, calculus, geometry, computer science, programming)
• **Why**:
  – an assessment tool to show you what prereq topics to brush up on
  – to save you some time on HW1 if you already know it all
• **How**:
  – $\alpha = \%$ of points on Background Test
  – $\beta = \%$ of points on Background Exercises
  – Grade: $\gamma = \alpha + (1-\alpha)\beta$

Your grade on HW1 will give you very little information about which topics to study. Hopefully, the Background Test does.
Syllabus Highlights

Background Test
- **When**: Fri, Jan 20, in-class
- **Where**: this lecture hall
- **What**: prerequisite material (probability, statistics, linear algebra, calculus, geometry, computer science, programming)
- **Why**:
  - an assessment tool to show you what prereq topics to brush up on
  - to save you some time on HW1 if you already know it all
- **How**:
  - $\alpha = \%$ of points on Background Test
  - $\beta = \%$ of points on Background Exercises
  - Grade: $\gamma = \alpha + (1-\alpha)\beta$

Correlation between Homework Average and Midterm Exam:
- Pearson: 0.32 (weak - moderate)
- Spearman: 0.25 (weak)

Correlation between Background Test and Midterm Exam:
- Pearson: 0.46 (moderate)
- Spearman: 0.43 (moderate)
Reminders

- **Background Test**
  - Fri, Jan 20, in-class

- **Homework 1: Background**
  - Out: Fri, Jan 20
  - Due: Wed, Jan 25 at 11:59pm
  - Two parts:
    1. written part to Gradescope
    2. programming part to Gradescope
Learning Objectives

You should be able to...

1. Formulate a well-posed learning problem for a real-world task by identifying the task, performance measure, and training experience
2. Describe common learning paradigms in terms of the type of data available, when it’s available, the form of prediction, and the structure of the output prediction
3. Implement Decision Tree training and prediction (w/simple scoring function)
4. Explain the difference between memorization and generalization [CIML]
5. Identify examples of the ethical responsibilities of an ML expert
Q&A

(my office hours are right now, just outside the lecture hall)