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?

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

**Speech Recognition**

1. Learning to recognize spoken words

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

   NOW...Hidden Markov models... 

   (Mitchell, 1997)

   ![Image](https://example.com/sp00k.png)

---

**Robotics**

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... 

   NOW... 

   (Mitchell, 1997)

   ![Image](https://example.com/alvinn.png)

---

**Games / Reasoning**

3. Learning to beat the masters at board games

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

   NOW... 

   (Mitchell, 1997)

   ![Image](https://example.com/backgammon.png)

---

**Computer Vision**

4. Learning to recognize images

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

   NOW... 

   (LeCun et al., 1995)

   ![Image](https://example.com/convnet.png)

---

**Learning Theory**

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

   ![Image](https://example.com/learning.png)

   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?
Speech Recognition

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)

Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults
### Robotics

#### 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="waymo.com" alt="Image of autonomous vehicle" /></td>
</tr>
</tbody>
</table>

(Mitchell, 1997)

[Image: Machine Learning book cover]
### Robotics

#### 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.geek.com/wp-content/uploads/2016/03/uber.jpg" alt="Image of autonomous vehicle" /></td>
</tr>
</tbody>
</table>

(Mitchell, 1997)

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.argo.ai/" alt="Image of autonomous vehicle" /></td>
</tr>
</tbody>
</table>

(Mitchell, 1997)

https://www.argo.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 and Lee Sedol]
4. Learning to recognize images

“…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>( N &gt; \frac{1}{\delta^2} \left[ \log(</td>
<td>\mathcal{H}</td>
</tr>
<tr>
<td>( N = O(\frac{1}{\delta} \left[ \text{VC}(</td>
<td>\mathcal{H}</td>
</tr>
</tbody>
</table>

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
   "...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)

   ![Image of speech recognition technology]

   Source: https://www.istechhelp.com/speechrecognitiontips/

Robotics
2. Learning to drive an autonomous vehicle
   "...the ALVINN system (Pomerleau, 1989) has used its learned strategies to drive unmanned at 50 miles per hour for 90 miles on public highways among other cars..."
   (Mitchell, 1997)

   ![Image of autonomous vehicle]

   Source: wayneco.com

Games / Reasoning
3. Learning to beat the masters at board games
   "...the world's top computer program for backgammon, TD-GAMMON (Teatru, 1992, 1993), learned its strategy by playing over one million practice games against itself..."
   (Mitchell, 1997)

   ![Image of board game]

Computer Vision
4. Learning to recognize images
   "...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)

   ![Images of various objects]

   Source: https://blog.google.com/computer-vision/

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

   ![Sample Complexity Results]

   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 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.
ML Big Picture

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?
- boolean
  - Binary Classification
- categorical
  - Multiclass Classification
- ordinal
  - Ordinal Classification
- real
  - Regression
- ordering
  - Ranking
- multiple discrete
  - Structured Prediction
- multiple continuous
  - (e.g. dynamical systems)
- both discrete & cont.
  - (e.g. mixed graphical models)
  - cont.

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

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

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
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

- If: “I need directions to X”
  Then: directions(here, nearest(X))

- If: “X directions”
  Then: directions(here, nearest(X))

- If: “Is there an X nearby”
  Then: directions(here, nearest(X))
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 \( \{y^{(1)}, \ldots, y^{(n)}\} \)

- \( x^{(1)}: \) How do I get to Starbucks?
  - \( y^{(1)}: \) directions(here, nearest(Starbucks))

- \( x^{(2)}: \) Show me the closest Starbucks
  - \( y^{(2)}: \) map(nearest(Starbucks))

- \( x^{(3)}: \) Send a text to John that I’ll be late
  - \( y^{(3)}: \) txtmsg(John, I’ll be late)

- \( x^{(4)}: \) Set an alarm for seven in the morning
  - \( y^{(4)}: \) setalarm(7:00AM)
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?*

- boolean: Binary Classification
- categorical: Multiclass Classification
- ordinal: Ordinal Classification
- real: Regression
- ordering: Ranking
- multiple discrete: Structured Prediction
- multiple continuous: (e.g. dynamical systems)
- both discrete & cont.: (e.g. mixed graphical models)
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**

### Labeled Dataset:

<table>
<thead>
<tr>
<th>index</th>
<th>label</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>brown, none, high</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>clear, crinkly, low</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>brown, none, low</td>
</tr>
</tbody>
</table>

### Unlabeled Dataset:

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

### Labeled Dataset:

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

### One example:

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

| - | brown none high |
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>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 Dataset:**

<table>
<thead>
<tr>
<th>index</th>
<th>label</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 training dataset is a labeled dataset used to learn a classifier
- **Def:** a classifier is a function that takes in features and predicts a label
- **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>

### (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>
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

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

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>

(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)

<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>
<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>brown</td>
<td>sound</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>clear</td>
<td>crinkly</td>
<td>low</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
<td>none</td>
<td>low</td>
<td></td>
</tr>
</tbody>
</table>

<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 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 ➔ label**

**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)

### 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>

### 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>

### 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>

### Error Rate

- **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)

- "Train time"
- "Test time"
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?
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!**

**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>
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**!

**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 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>

**Error rate = 3/3**
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 \(\rightarrow \) always predict “-”

<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>

Training Dataset:

Test Predictions: 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 = 1/3
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!

... The majority vote classifier even ignores the features if it’s making predictions on the training dataset!

**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>

**Train 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>
<tr>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</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**
  - 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)

---

**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>crinky</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>brown</td>
<td>crinky</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>

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

**Training 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>crinky</td>
<td>low</td>
</tr>
<tr>
<td>3</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>

**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 exam, Thu, Feb. 17
- **Exam 2**: evening exam, Thu, Mar. 31
- **Exam 3**: final exam week, date TBD by registrar
- **Homework**: ~3 written and ~6 written + programming (Python)
  - 8 grace days for homework assignments
  - Late submissions: 80% day 1, 60% day 2, 40% day 3, 20% day 4
  - No submissions accepted after 4 days w/o extension; HW3, HW6, HW9 only 2 days
  - Extension requests: 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. failure)
- **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...?

**Classification & Regression**

<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture</th>
<th>Readings</th>
<th>Announcements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• <strong>Command Line and File I/O Tutorial</strong>, 10601 Course Staff (2020).</td>
<td></td>
</tr>
<tr>
<td>Fri, 5-Feb</td>
<td>Recitation: HW1 [Handout] [Solutions]</td>
<td></td>
<td></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>
</tbody>
</table>
Where can I find...?

Introduction to Machine Learning

<table>
<thead>
<tr>
<th>10-301/601 Office Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun 8/29</td>
</tr>
<tr>
<td>Mon 8/30</td>
</tr>
<tr>
<td>Tue 8/31</td>
</tr>
<tr>
<td>Wed 9/1</td>
</tr>
<tr>
<td>Thu 9/2</td>
</tr>
<tr>
<td>Fri 9/3</td>
</tr>
<tr>
<td>Sat 9/4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>9am</td>
<td></td>
</tr>
<tr>
<td>10am</td>
<td>10:10 - 11:30 10-301/601 Section A/C</td>
</tr>
<tr>
<td></td>
<td>Mellon Institute</td>
</tr>
<tr>
<td>11am</td>
<td>11:30 - Matt and H</td>
</tr>
<tr>
<td>12pm</td>
<td></td>
</tr>
<tr>
<td>1pm</td>
<td>1:25p - 2:45p 10-301/601 Section R/D</td>
</tr>
<tr>
<td></td>
<td>CUC McConomy</td>
</tr>
<tr>
<td>2pm</td>
<td>2:45p - Matt and H</td>
</tr>
<tr>
<td>3pm</td>
<td></td>
</tr>
</tbody>
</table>
Assignments

There will be 8 homework assignments during the semester in addition to the exams. The assignments will consist of both theoretical and practical work and 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)
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.
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/
Oh, the Places You'll Use Probability!

By Dr. Seuss
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 \( D \).

  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:**

  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!

(see instructions of written portion of HW1)
Reminders

• Homework 1: Background
  – Out: Wed, Jan 19 (1st lecture)
  – Due: Wed, Jan 26 at 11:59pm
  – Two parts:
    1. written part to Gradescope
    2. programming part to Gradescope
  – unique policy for this assignment:
    1. two submissions for written (see writeup for details)
    2. unlimited submissions for programming (i.e. keep submitting until you get 100%)
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