10-301/601 Introduction to Machine Learning

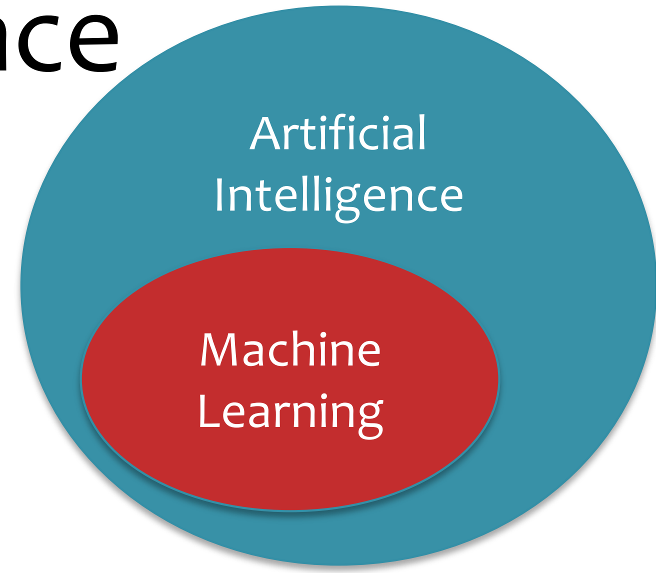
Machine Learning Department
School of Computer Science
Carnegie Mellon University

Course Overview

Matt Gormley
Lecture 1
Jan. 19, 2022

WHAT IS MACHINE LEARNING?

Artificial Intelligence

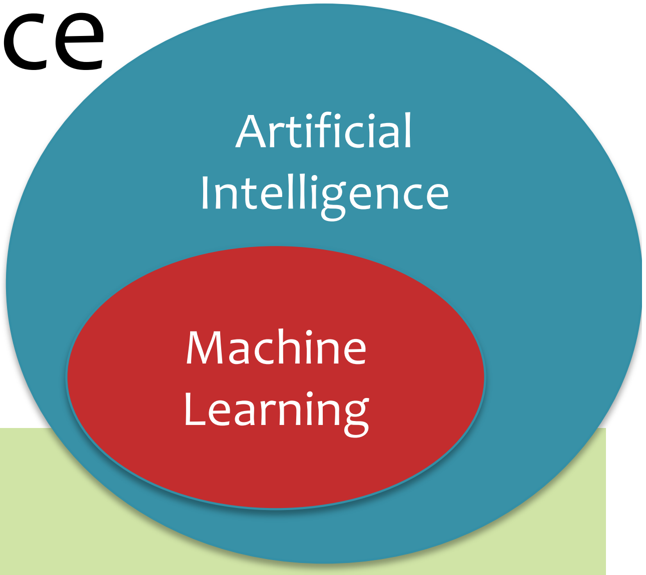


The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

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

Artificial Intelligence

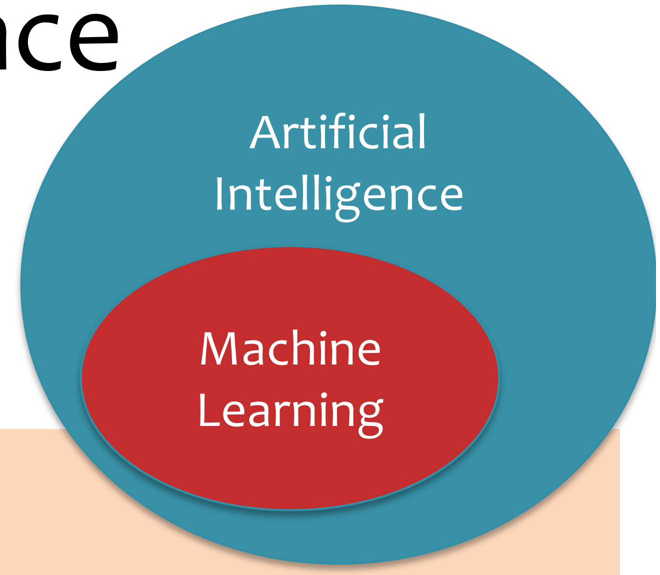


The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

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

Artificial Intelligence

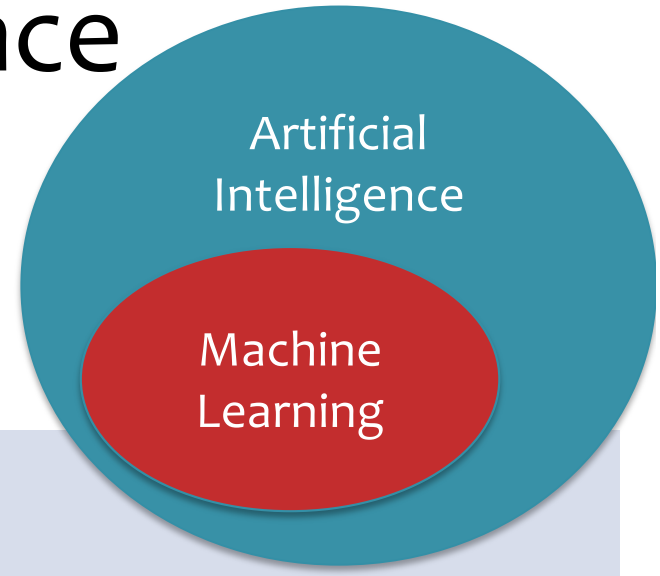


The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

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

Artificial Intelligence

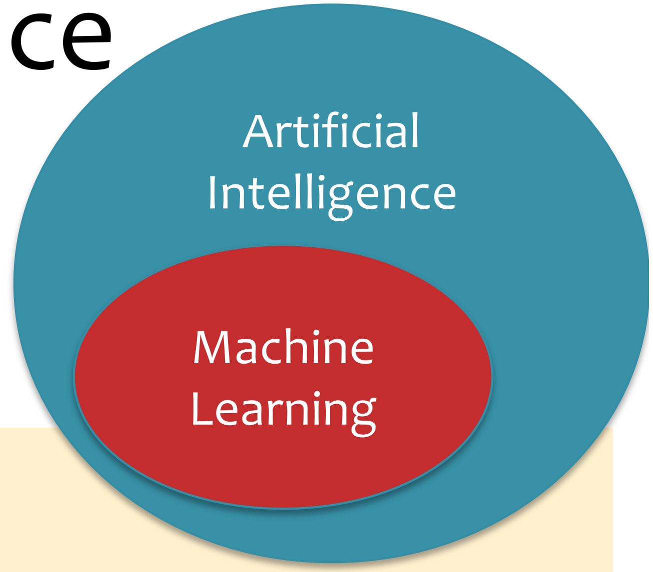


The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

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

Artificial Intelligence

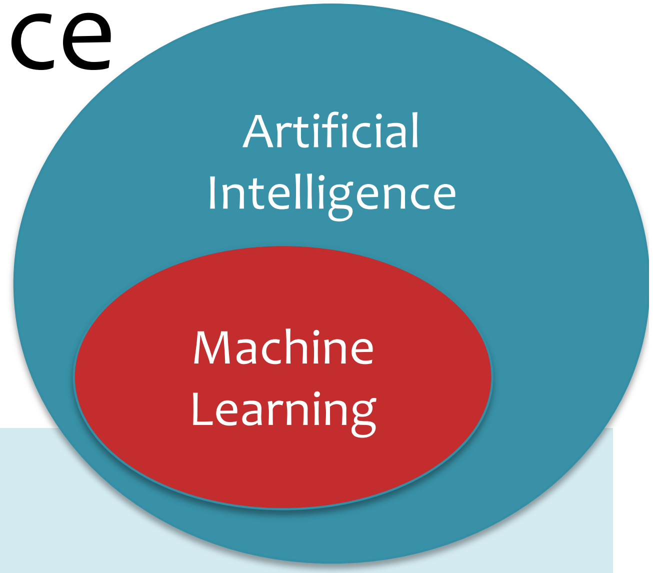


The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

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

Artificial Intelligence

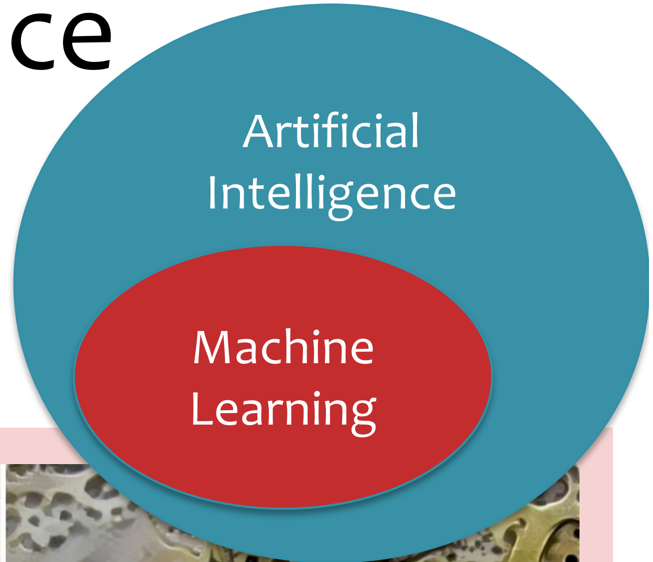


The basic goal of AI is to develop intelligent machines.

This consists of many sub-goals:

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

Artificial Intelligence



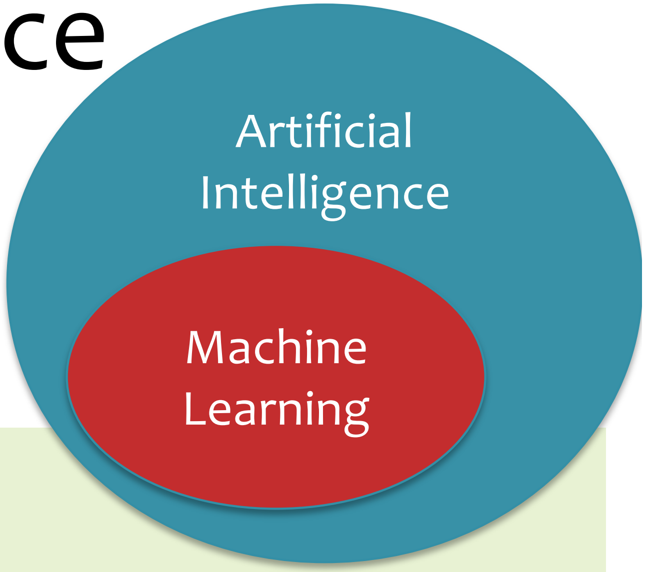
The basic goal of AI is to develop intelligent machines.

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

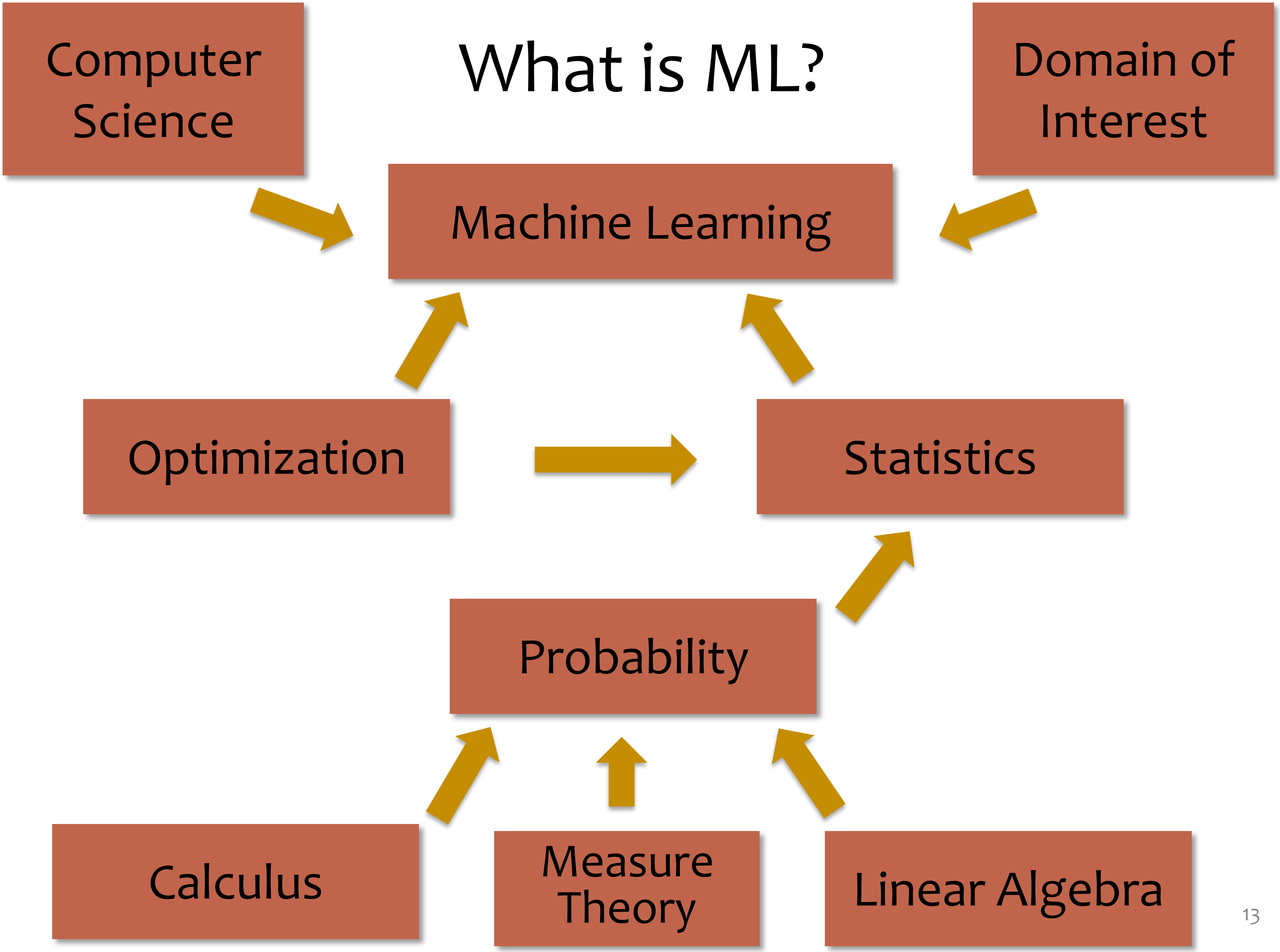
Statistics

Probability

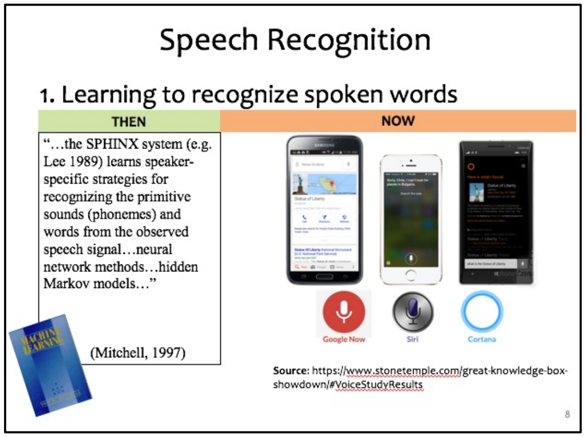
Computer Science

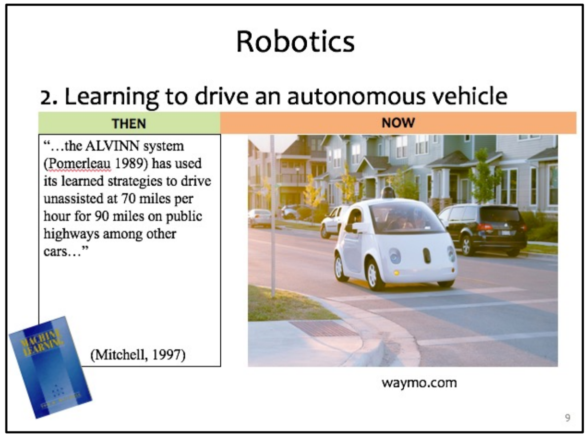
Optimization

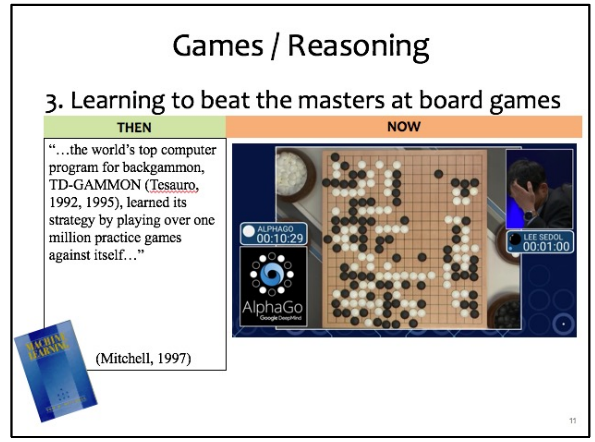
What is ML?

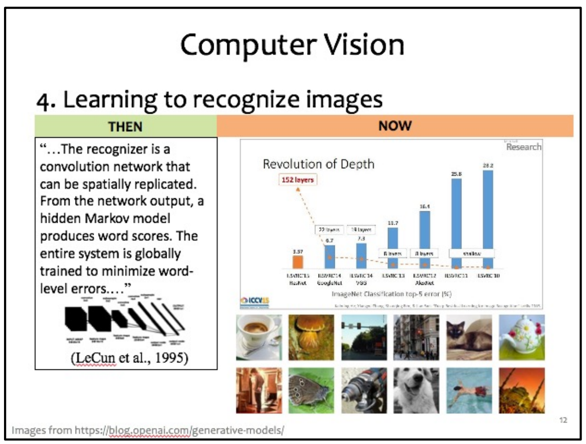

Computer Science
Domain of Interest
Machine Learning
Optimization
Statistics
Probability
Calculus
Measure Theory
Linear Algebra

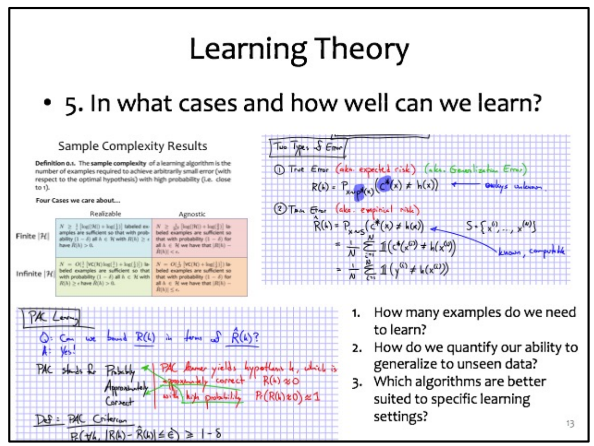
What is ML?


Speech Recognition
1. Learning to recognize spoken words
THEN
NOW
"…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)
Google Now
Siri
Cortana
Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults
8


Robotics
2. Learning to drive an autonomous vehicle
THEN
NOW
"…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)
waymo.com
9

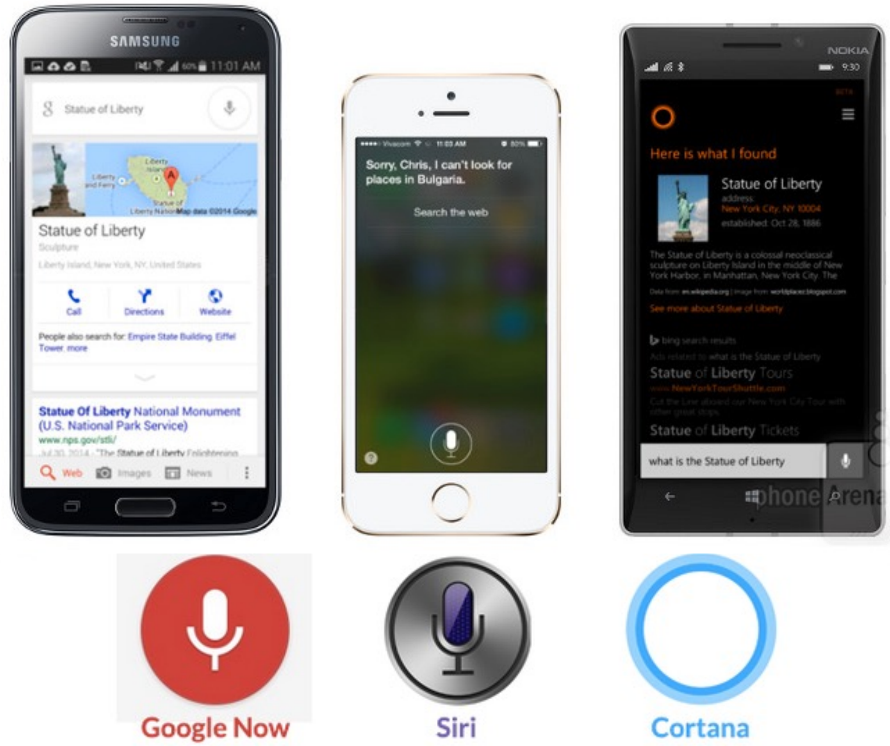

Games / Reasoning
3. Learning to beat the masters at board games
THEN
NOW
"…the world's top computer program for backgammon, TD-GAMMON (Tesauro, 1992, 1995), learned its strategy by playing over one million practice games against itself…"
(Mitchell, 1997)
ALPHAGO
00:10:29
AlphaGo
LEE SEDOL
00:01:00
11

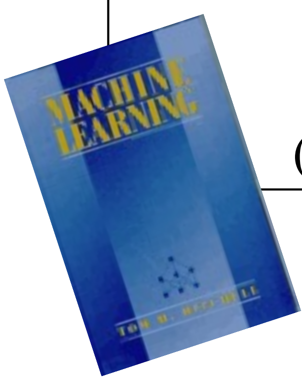

Computer Vision
4. Learning to recognize images
THEN
NOW
"…The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors.…"
(LeCun et al., 1995)
Revolution of Depth
152 layers
Research
ImageNet Classification top-5 error (%)
Images from https://blog.openai.com/generative-models/
12


Learning Theory
• 5. In what cases and how well can we learn?
Sample Complexity Results
Four Cases we care about…
Realizable
Agnostic
Two Types of Error
PAC Learning
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?
13

Speech Recognition

1. Learning to recognize spoken words

THEN	NOW
"…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)	

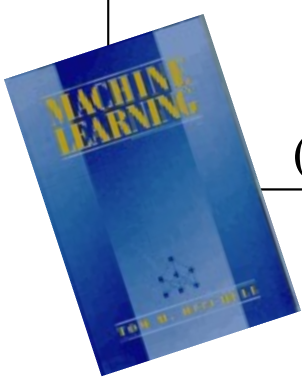


Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults

Robotics

2. Learning to drive an autonomous vehicle

THEN	NOW
"…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)	 waymo.com



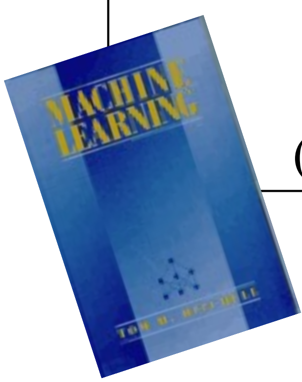
Robotics

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…"

(Mitchell, 1997)



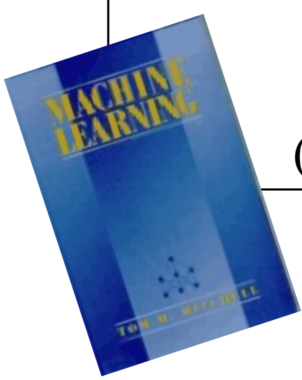
NOW

https://www.geek.com/wp-content/uploads/2016/03/uber.jpg

Robotics

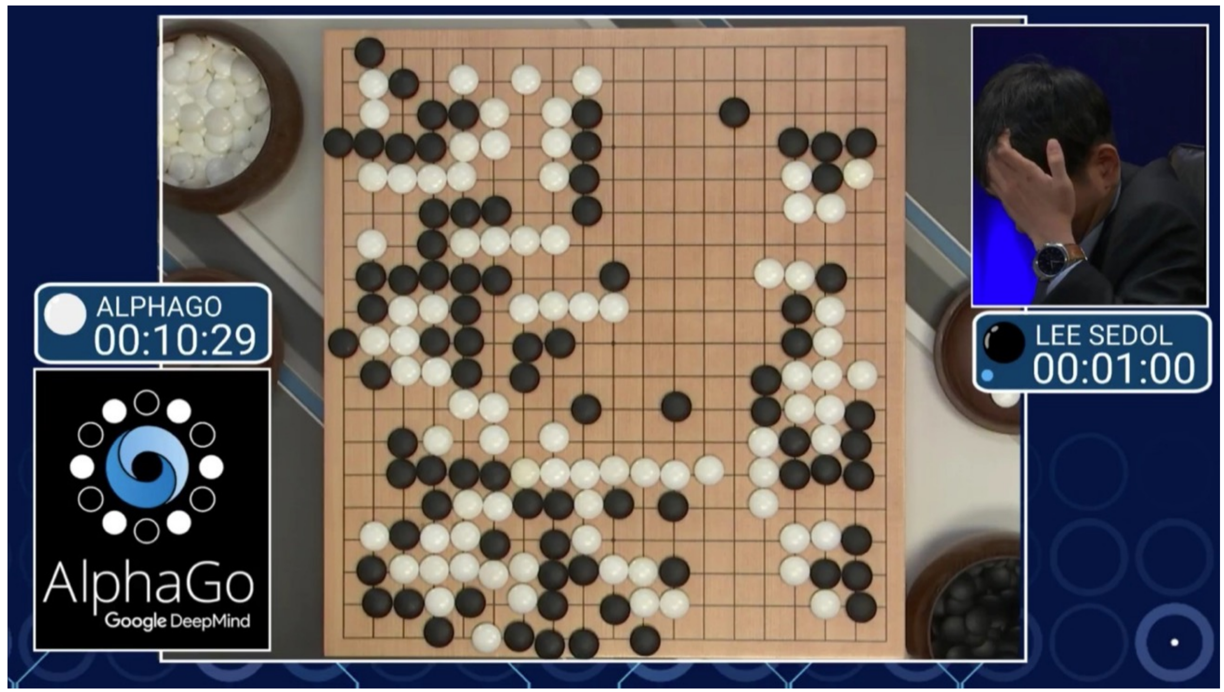
2. Learning to drive an autonomous vehicle

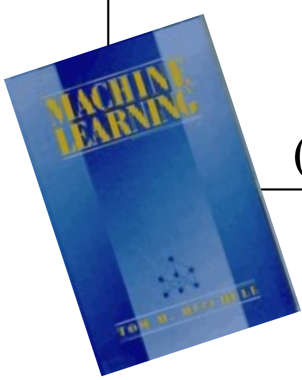
THEN	NOW
"…the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars…" (Mitchell, 1997)	 https://www.argo.ai/



Games / Reasoning

3. Learning to beat the masters at board games

THEN	NOW
"…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)	

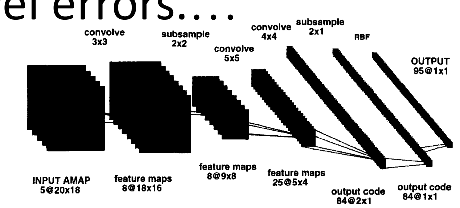


Computer Vision

4. Learning to recognize images

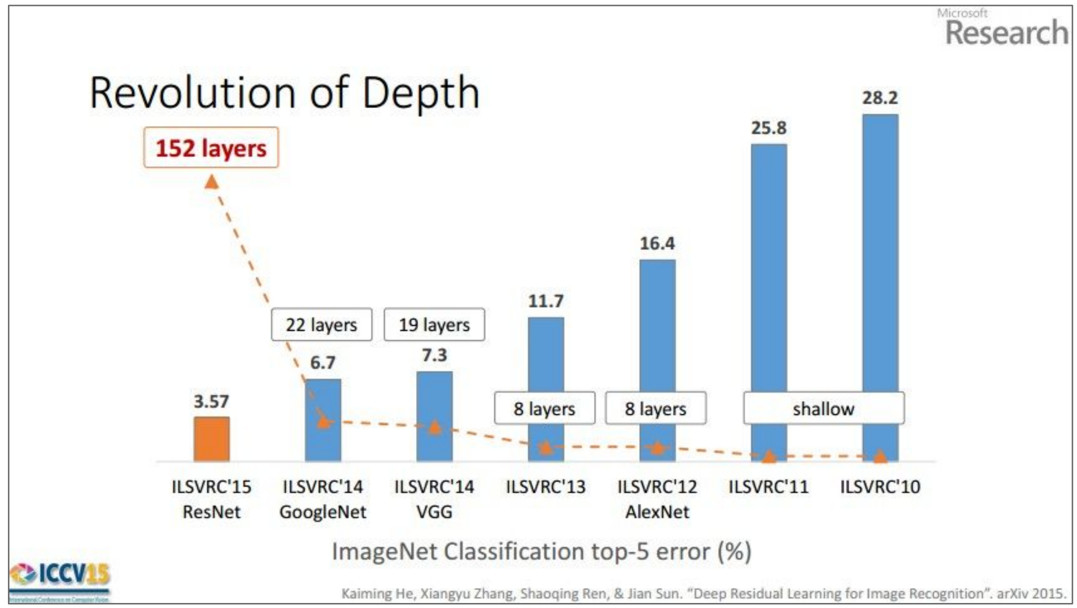
THEN

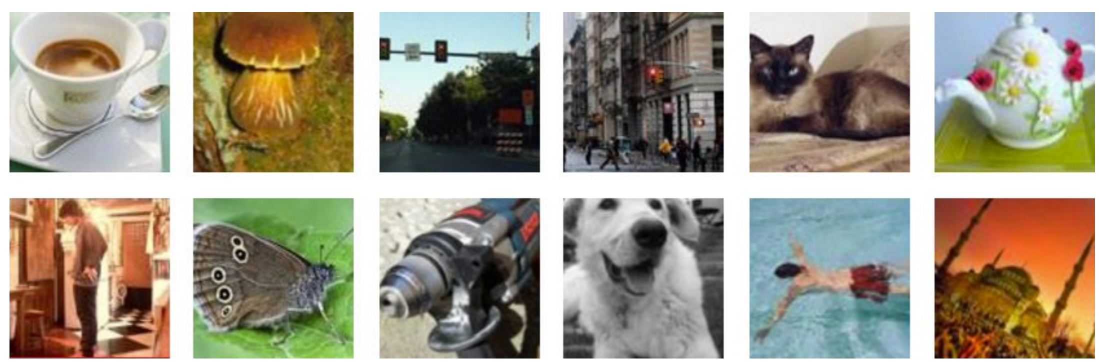
"…The recognizer is a convolution network that can be spatially replicated. From the network output, a hidden Markov model produces word scores. The entire system is globally trained to minimize word-level errors.…"



(LeCun et al., 1995)

NOW





Images from https://blog.openai.com/generative-models/

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...

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

Two Types of Error

① True Error (aka. expected risk) (aka. Generalization Error)

$$R(h) = P_{x \sim p^*(x)}\left(c^*(x) \neq h(x)\right) \leftarrow \text{always unknown.}$$

② Train Error (aka. empirical risk)

$$\hat{R}(h) = P_{x \sim S}\left(c^*(x) \neq h(x)\right) \qquad S = \{x^{(1)}, \ldots, x^{(N)}\}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left(c^*(x^{(i)}) \neq h(x^{(i)})\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left(y^{(i)} \neq h(x^{(i)})\right)$$

known, computable

PAC Learning

Q: Can we bound $R(h)$ in terms of $\hat{R}(h)$?

A: Yes!

PAC stands for Probably Approximately Correct

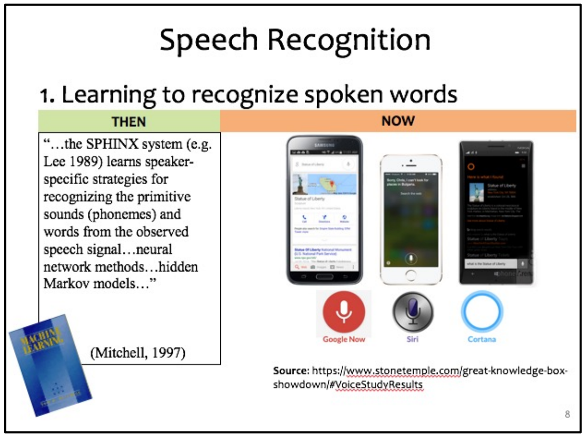
PAC learner yields hypothesis h, which is approximately correct $R(h) \approx 0$ with high probability $Pr(R(h) \approx 0) \approx 1$

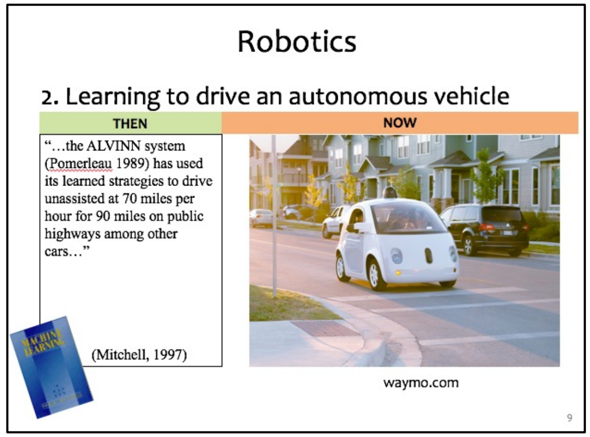
Def: PAC Criterion

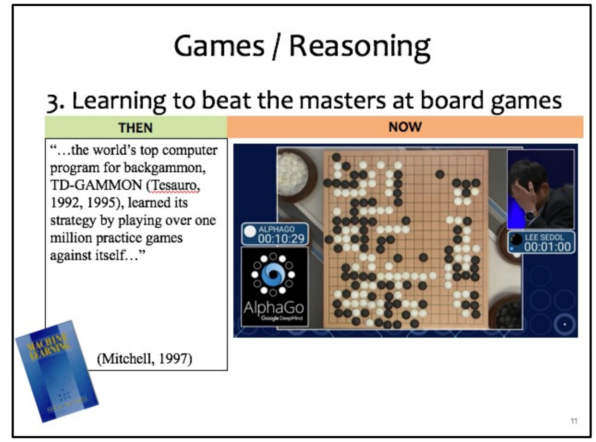
$$Pr(\forall h, |R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta$$

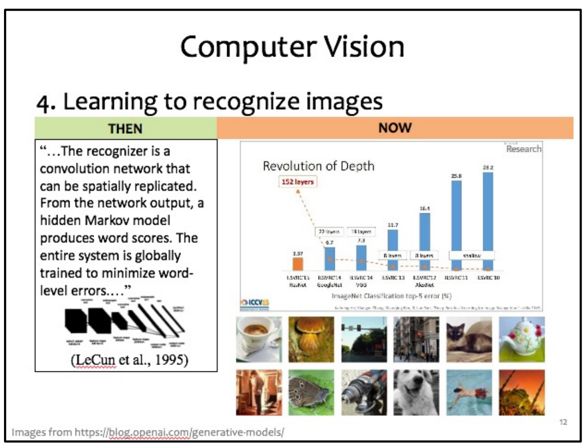
1. How many examples do we need to learn?
2. How do we quantify our ability to generalize to unseen data?
3. Which algorithms are better suited to specific learning settings?

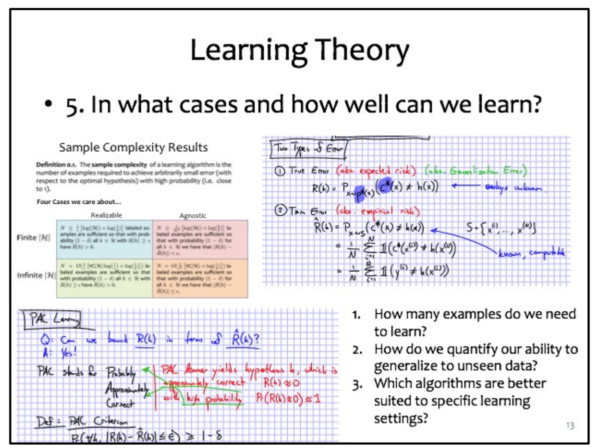
What is ML?


Speech Recognition
1. Learning to recognize spoken words
THEN
NOW
"…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)
Google Now
Siri
Cortana
Source: https://www.stonetemple.com/great-knowledge-box-showdown/#VoiceStudyResults
8


Robotics
2. Learning to drive an autonomous vehicle
THEN
NOW
"…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)
waymo.com
9


Games / Reasoning
3. Learning to beat the masters at board games
THEN
NOW
"…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)
ALPHAGO 00:10:29
AlphaGo
LEE SEDOL 00:01:00
11


Computer Vision
4. Learning to recognize images
THEN
NOW
"…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)
Revolution of Depth
152 layers
Research
Images from https://blog.openai.com/generative-models/
12


Learning Theory
• 5. In what cases and how well can we learn?
Sample Complexity Results
Realizable
Agnostic
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?
13

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

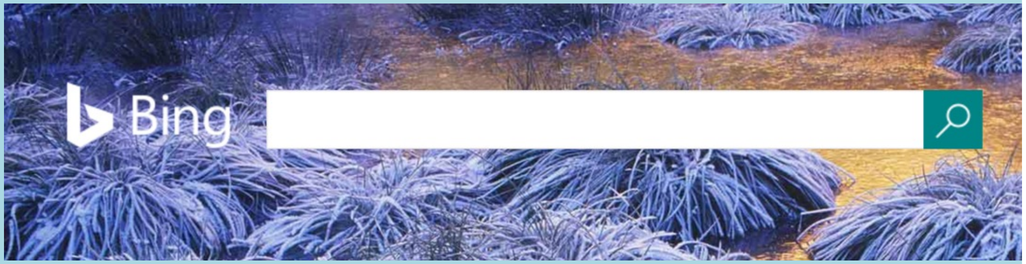
Societal Impacts of ML

What ethical responsibilities do we have as machine learning experts?

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

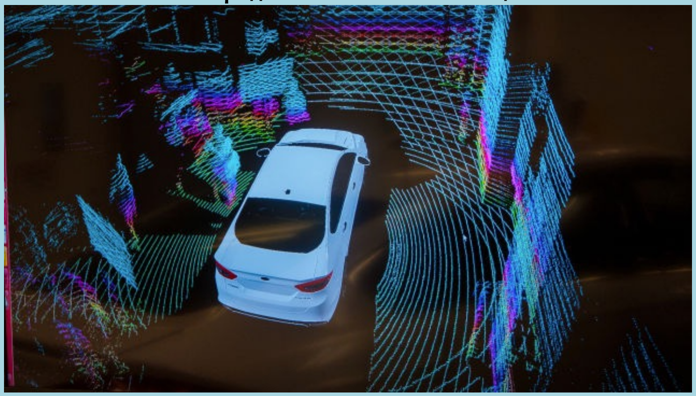
Answer:

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



http://bing.com/

http://arstechnica.com/



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.

https://flic.kr/p/HNJUzV

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)

Application Areas

Key challenges?

NLP, Speech, Computer Vision, Robotics, Medicine, Search

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*

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*:

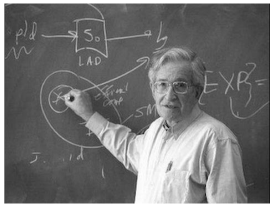
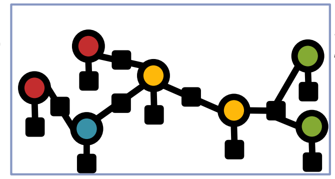
3. Experience, *E*:

Example Learning Problems

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

1. Task, *T*:
2. Performance measure, *P*:
3. Experience, *E*:

Capturing the Knowledge of Experts

1980  1990  2000  2010

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

Give me directions to Starbucks

```
If: "give me directions to X"
Then: directions(here, nearest(X))
```

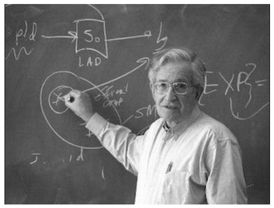
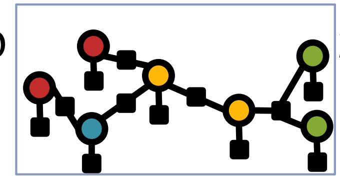
How do I get to Starbucks?

```
If: "how do i get to X"
Then: directions(here, nearest(X))
```

Where is the nearest Starbucks?

```
If: "where is the nearest X"
Then: directions(here, nearest(X))
```

Capturing the Knowledge of Experts

1980  1990  2000  2010

Solution #1: Expert Systems

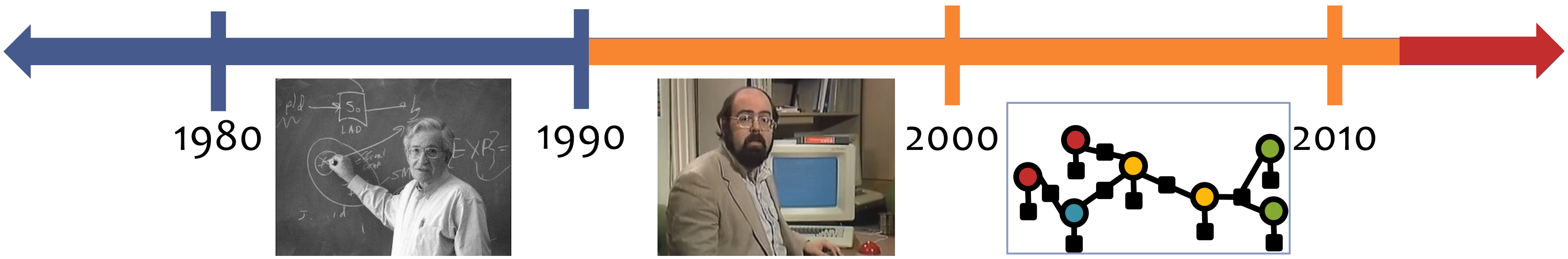
- Over 20 years ago, we had rule-based systems:
  1. Put a bunch of linguists in a room
  2. Have them think about the structure of their native language and write down the rules they devise

I need directions to Starbucks
`If: "I need directions to X"` `Then: directions(here, nearest(X))`

Starbucks directions
`If: "X directions"` `Then: directions(here, nearest(X))`

Is there a Starbucks nearby?
`If: "Is there an X nearby"` `Then: directions(here, nearest(X))`

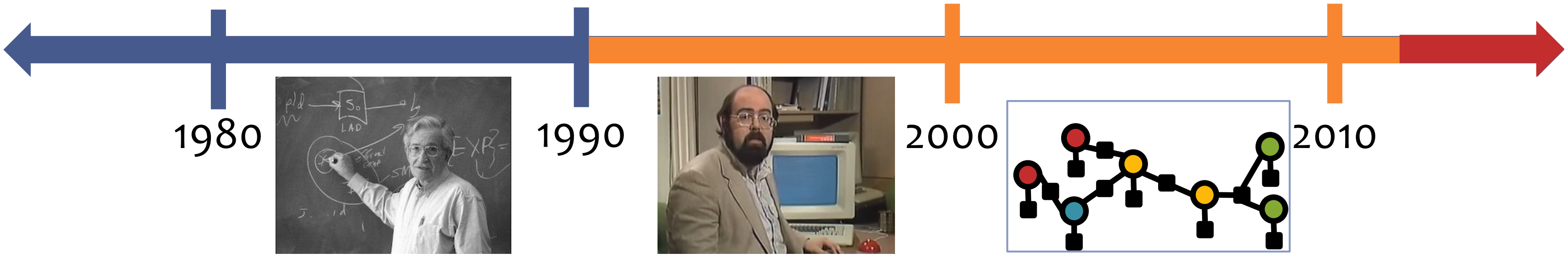
Capturing the Knowledge of Experts



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

Capturing the Knowledge of Experts



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

(without any math!)

SUPERVISED LEARNING

Building a Trash Classifier

- Suppose the friends OF THE RIVERFRONT 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

trash?	color	sound	weight
+	green	crinkly	high
-	brown	crinkly	low
-	grey	none	high
+	clear	none	low
-	green	none	low

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:

	label	features		
	trash?	color	sound	weight
	-	brown	none	high

Labeled Dataset:

	label	features		
index	trash?	color	sound	weight
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

Unlabeled Dataset:

	features		
index	color	sound	weight
1	brown	none	high
2	clear	crinkly	low
3	brown	none	low

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:

	label	features		
	trash?	**color**	**sound**	**weight**
	-	brown	none	high

Labeled Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

Unlabeled Dataset:

	features		
index	**color**	**sound**	**weight**
1	brown	none	high
2	clear	crinkly	low
3	brown	none	low

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

Classifier

features → label

- *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

Training Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Test Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

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

- *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 t used to **evaluate** a er

Classifier
features → label

Test Predictions:

predictions

index	trash?
1	+
2	+
3	-

(Unlabeled) Test Dataset:

features

index	color	sound	weight
1	brown	none	high
2	clear	crinkly	low
3	brown	none	low

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

- *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

error rate = 1/3

Test Predictions:

index	trash? (predictions)
1	+
2	+
3	-

(Labeled) Test Dataset:

index	trash? (label)	color (features)	sound (features)	weight (features)
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

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:

	label	features		
index	trash?	color	sound	weight
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Classifier
features → label

(Unlabeled) Test Dataset:

		features		
index		color	sound	weight
1		brown	none	high
2		clear	crinkly	low
3		brown	none	low

(Labeled) Test Dataset:

	label	features		
index	trash?	color	sound	weight
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

Test Predictions:

	predictions
index	trash?
1	+
2	+
3	-

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:

	label	features		
index	trash?	color	sound	weight
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Classifier
features → label

(Unlabeled) Test Dataset:

		features		
index		color	sound	weight
1		brown	none	high
2		clear	crinkly	low
3		brown	none	low

(Labeled) Test Dataset:

	label	features		
index	trash?	color	sound	weight
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

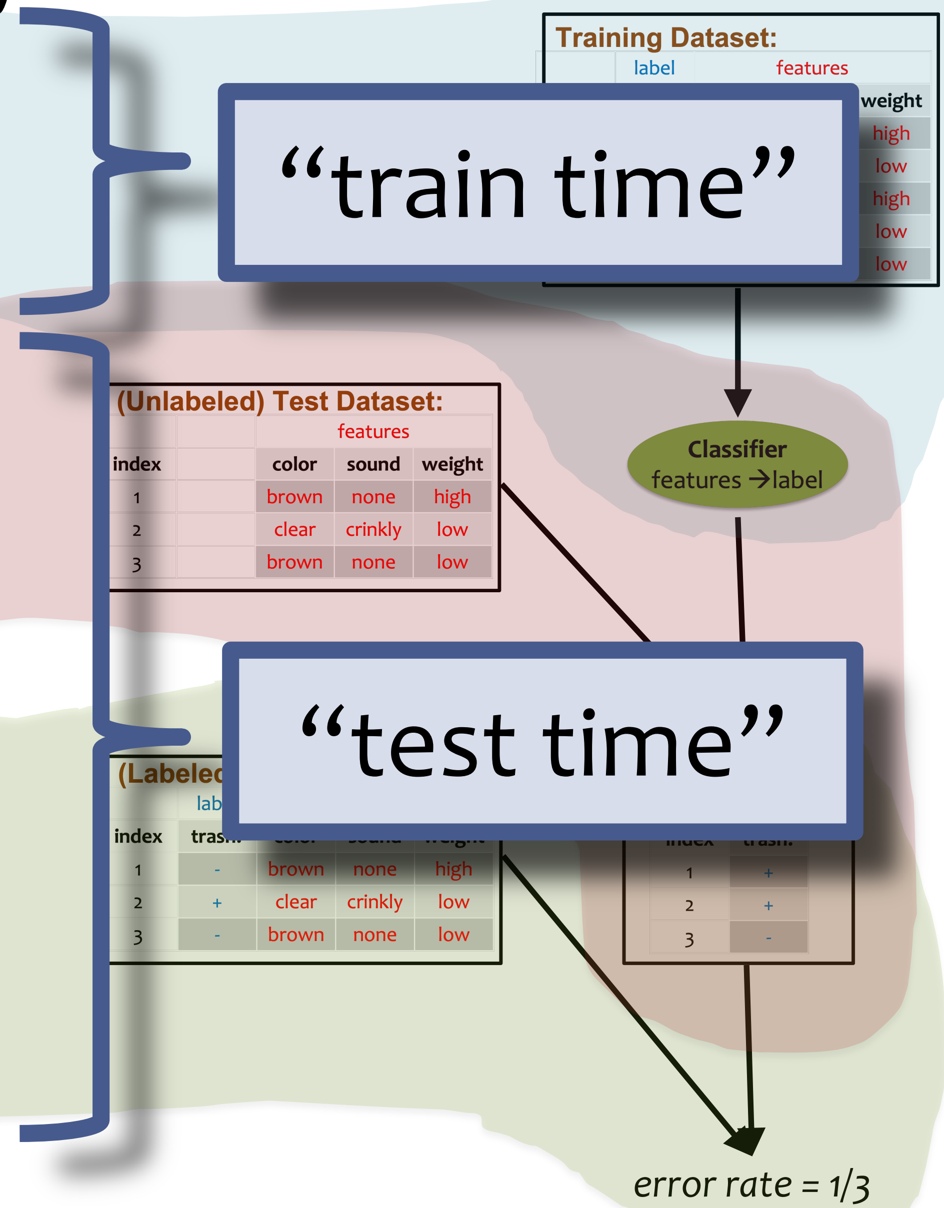
Test Predictions:

	predictions
index	trash?
1	+
2	+
3	-

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)



Supervised Binary Classification

- Step 1: training
  - *Given*: labeled **training dataset**
  - *Goal*: learn a **classifier** from the training dataset
- Step 2: prediction
  - *Given*: unlabeled **test dat**
    : learned classifier
  - *Goal*: **predict** a label for e
    instance
- Step 3: evaluation
  - *Given*: **predictions** from
    : labeled **test datas**
  - *Goal*: compute the **test e**
    rate (i.e. error rate on th
    dataset)

Training Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

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!**

Classifier
features → random!

Training Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Test Predictions:

	predictions
index	**trash?**
1	-
2	-
3	+

error rate = 2/3

Test Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

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!

Training Dataset:

index	trash? (label)	color (features)	sound (features)	weight (features)
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Test Predictions:

index	trash? (predictions)
1	+
2	+
3	-

error rate = 1/3

Test Dataset:

index	trash? (label)	color (features)	sound (features)	weight (features)
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

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!

Training Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Test Predictions:

	predictions
index	**trash?**
1	+
2	-
3	+

error rate = 3/3

Test Dataset:

	label	features		
index	**trash?**	**color**	**sound**	**weight**
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

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:

index	trash? (label)	color (features)	sound (features)	weight (features)
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Test Predictions:

index	trash? (predictions)
1	-
2	-
3	-

error rate = 1/3

Test Dataset:

index	trash? (label)	color (features)	sound (features)	weight (features)
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

Majority Vote Classifier

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

Classifier
features → always predict "-"

…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!

Train Predictions:

index	trash? (predictions)
1	-
2	-
3	-
4	-
5	-

error rate = 2/5

Training Dataset:

index	trash? (label)	color (features)	sound (features)	weight (features)
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

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)

Training Dataset:

	label	features		
index	trash?	color	sound	weight
1	+	green	crinkly	high
2	-	brown	crinkly	low
3	-	grey	none	high
4	+	clear	none	low
5	-	green	none	low

Classifier
features → always predict "-"

(Unlabeled) Test Dataset:

		features		
index		color	sound	weight
1		brown	none	high
2		clear	crinkly	low
3		brown	none	low

(Labeled) Test Dataset:

	label	features		
index	trash?	color	sound	weight
1	-	brown	none	high
2	+	clear	crinkly	low
3	-	brown	none	low

Test Predictions:

	predictions
index	trash?
1	-
2	-
3	-

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

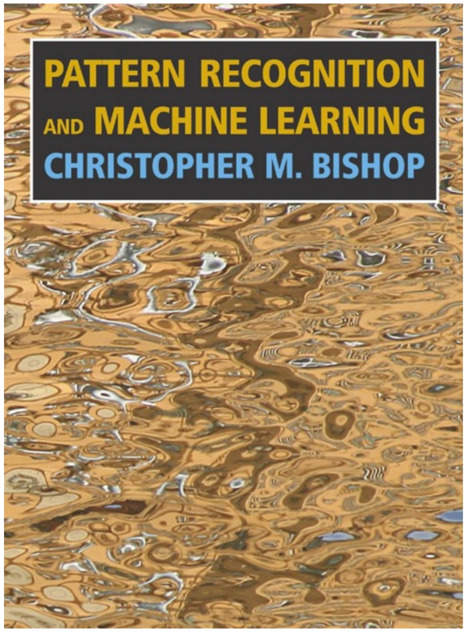
- **Grading**: 50% homework, 15% exam 1, 15% exam 2, 15% exam 3, 5% participation
- **Exam 1**: evening exam, Thu, Feb. 17
- **Exam 2**: evening exam, Thu, Mar. 31
- **Exam 3**: final exam week, date TBD by registrar
- **Homework:** ~3 written and ~6 written + programming (Python)
  - 8 grace days for 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

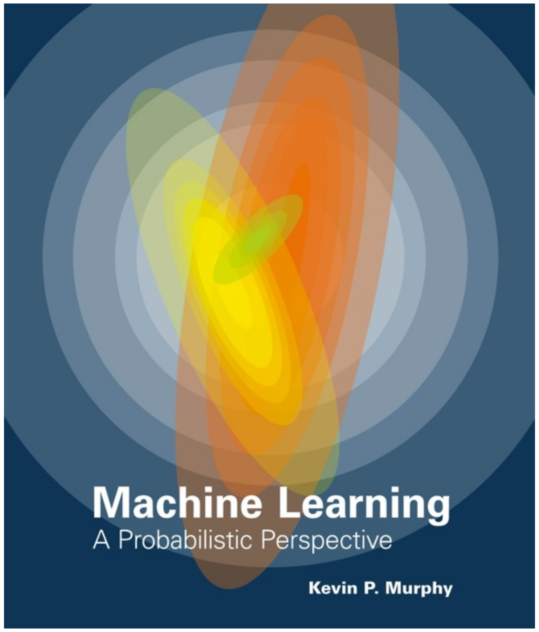
Lectures

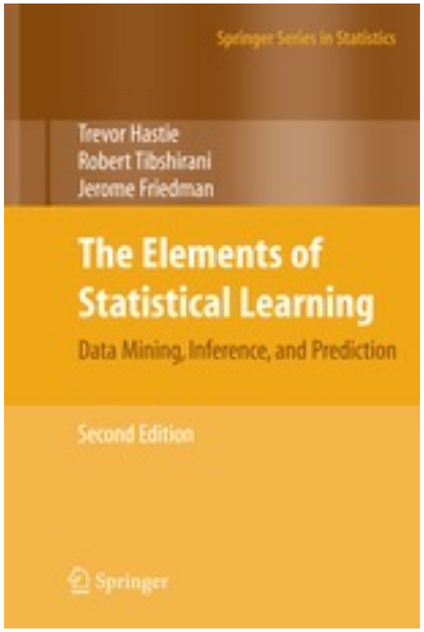
- You should ask lots of questions
  - Interrupting (by raising a hand) to ask your question is strongly encouraged
  - Asking questions later (or in real time) on Piazza is also great
- When I ask a question…
  - I want you to answer
  - Even if you don’t answer, think it through as though I’m about 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!





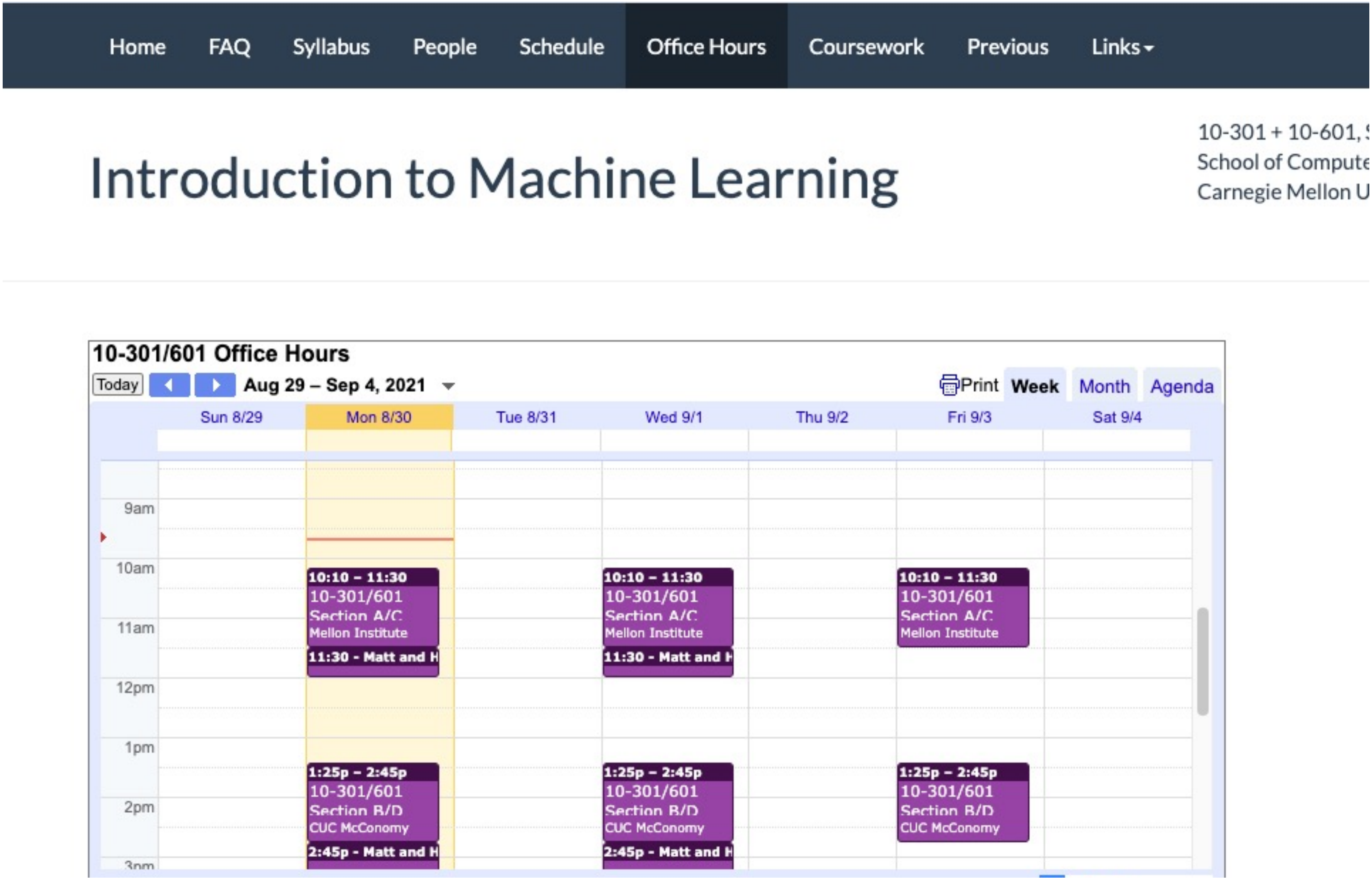


Where can I find…?

Home | FAQ | Syllabus | People | Schedule | Office Hours | Coursework | Previous | Links

Date	Lecture	Readings	Announcements
Classification & Regression			
Mon, 1-Feb	Lecture 1 : Course Overview [Slides]	• *10601 Notation Crib Sheet*. Matt Gormley (2018). • *Command Line and File I/O Tutorial*. 10601 Course Staff (2020). • *10601 Learning Objectives*. Matt Gormley (2018). • *Visual Information Theory*. Christopher Olah (2015). blog.	
Wed, 3-Feb	Lecture 2 : Decision Trees, Overfitting [Slides]	• *Decision Trees*. Hal Daumé III (2017). CIML, Chapter 1.	HW1 out
Fri, 5-Feb	Recitation: HW1 [Handout] [Solutions]		
Mon, 8-Feb	Lecture 3 : Generalizing from exampes - the Big Picture [Slides] [Poll]	• *Limits of Learning*. Hal Daumé III (2017). CIML, Chapter 2.	
Wed, 10-Feb	Lecture 4 : k-Nearest Neighbors [Slides] [Whiteboard] [Poll]	• *Geometry and Nearest Neighbors*. Hal Daumé III (2017). CIML, Chapter 3.	HW1 due HW2 out
Fri, 12-Feb	Recitation: HW2 [Handout] [Solutions]		
Mon, 15-Feb	Lecture 5 : Model Selection [Slides] [Whiteboard] [Poll]		
Wed, 17-Feb	Lecture 6 : Perceptron [Slides] [Whiteboard] [Poll]	• *The Perceptron*. Hal Daumé III (2017). CIML, Chapter 4.	HW1 solution session (Thursday)

Where can I find…?



Where can I find…?

Home FAQ Syllabus People Schedule Office Hours Coursework Previous Links

Introduction to Machine Learning

10-301 + 10-601, S
School of Compute
Carnegie Mellon U

Assignments

There will be 8 homework assignments during the semester in addition to the exams. The assignments will consist of both theoretical and pr
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**…

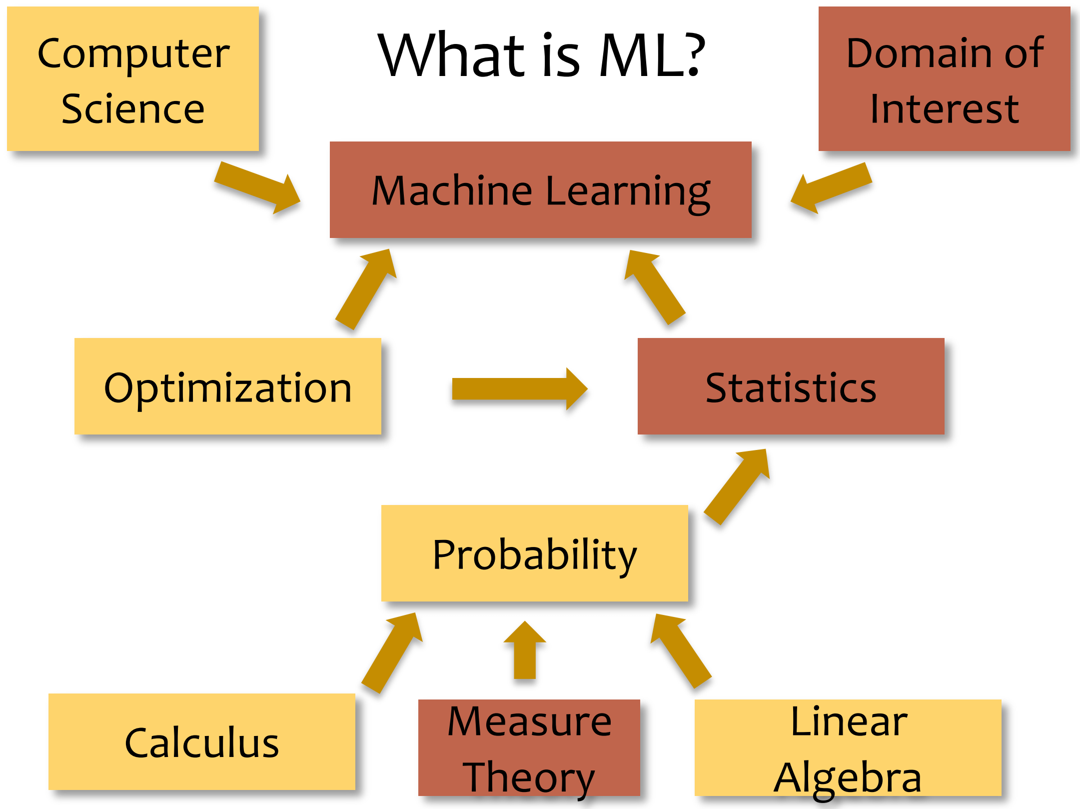
motivated by (carefully chosen) **real-world problems** that arise in **machine learning**…

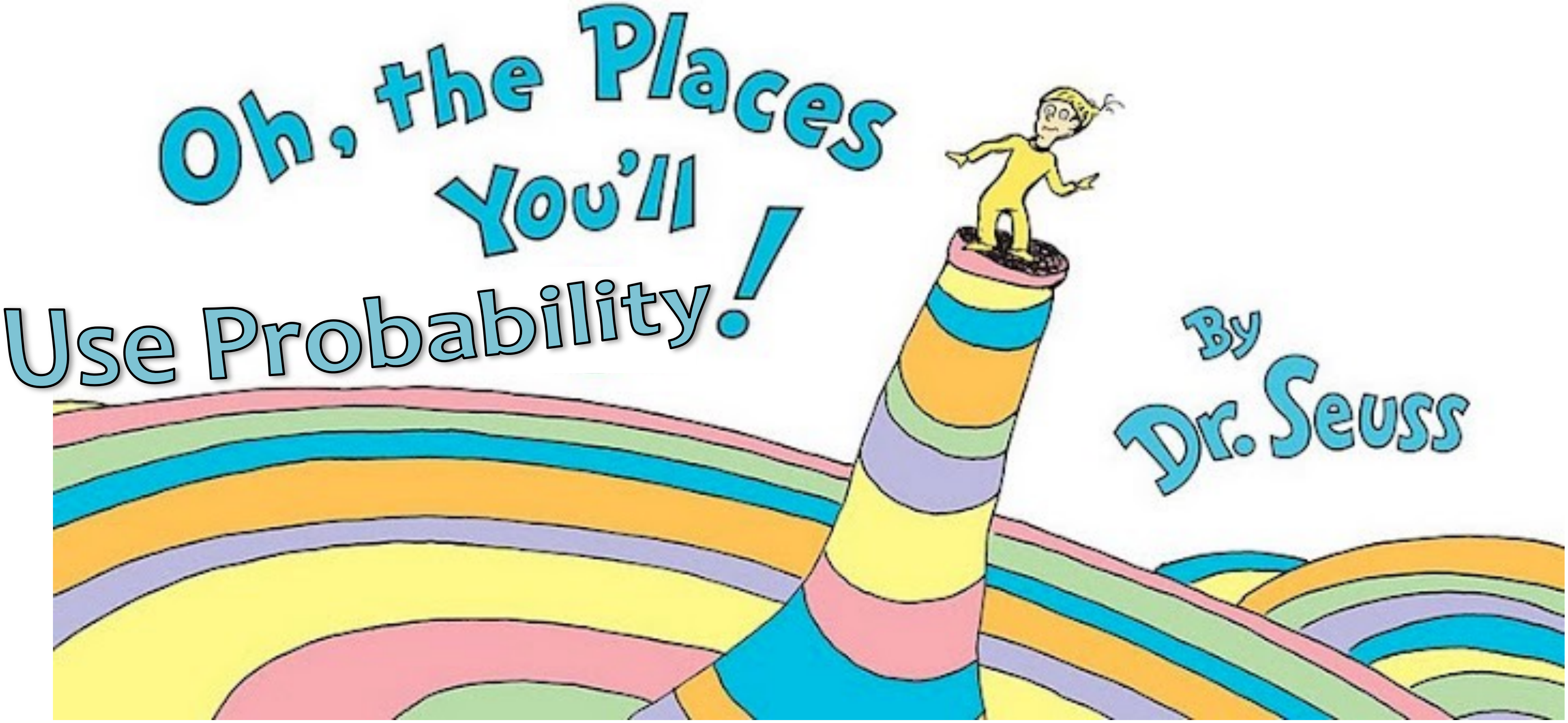
where the **broader picture** of how those problems arise is treated **somewhat informally.**

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

$$\begin{aligned} P(Y = y|X = x; \boldsymbol{\theta}) &= p(y|x; \boldsymbol{\theta}) \\ &= \frac{\exp(\boldsymbol{\theta}_y \cdot \mathbf{f}(x))}{\sum_{y'} \exp(\boldsymbol{\theta}_{y'} \cdot \mathbf{f}(x))} \end{aligned}$$

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: $err_S(h) = \frac{1}{m}\sum_i I\big(h(x_i) \neq c^*(x_i)\big)$

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

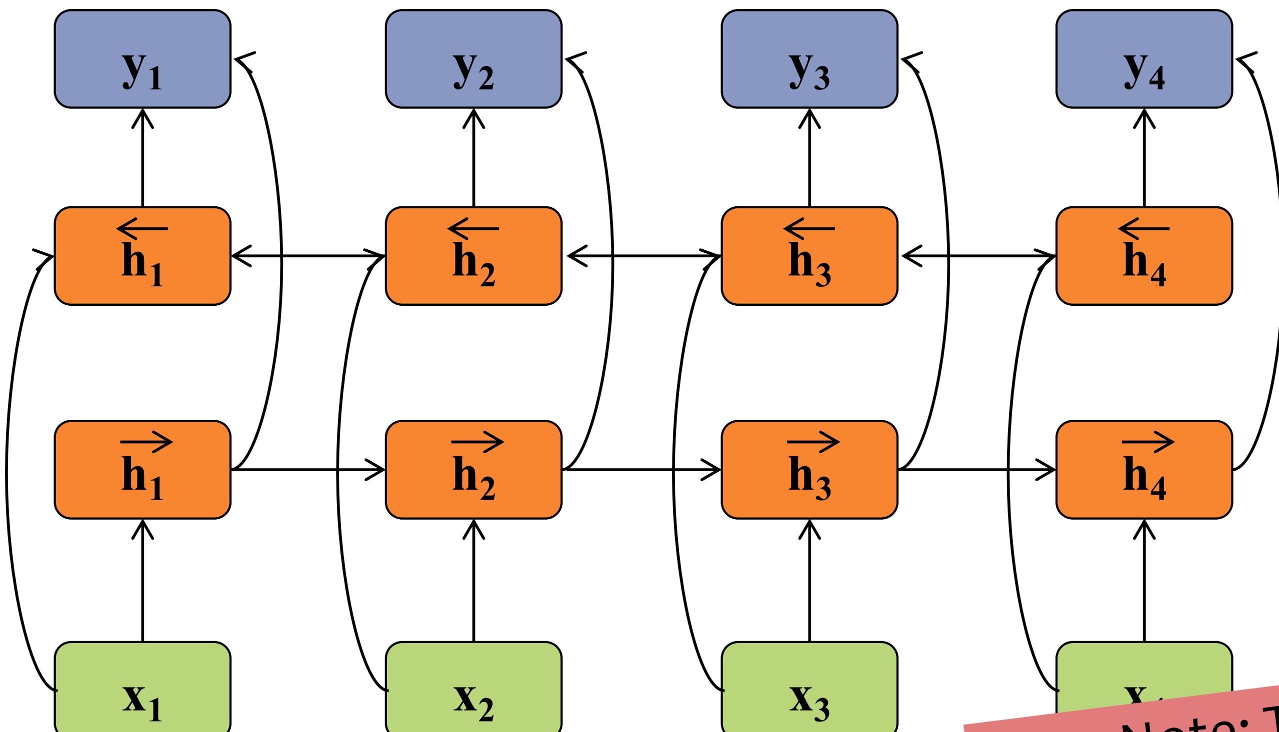
Sample complexity: bound $err_D(h)$ in terms of $err_S(h)$

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)

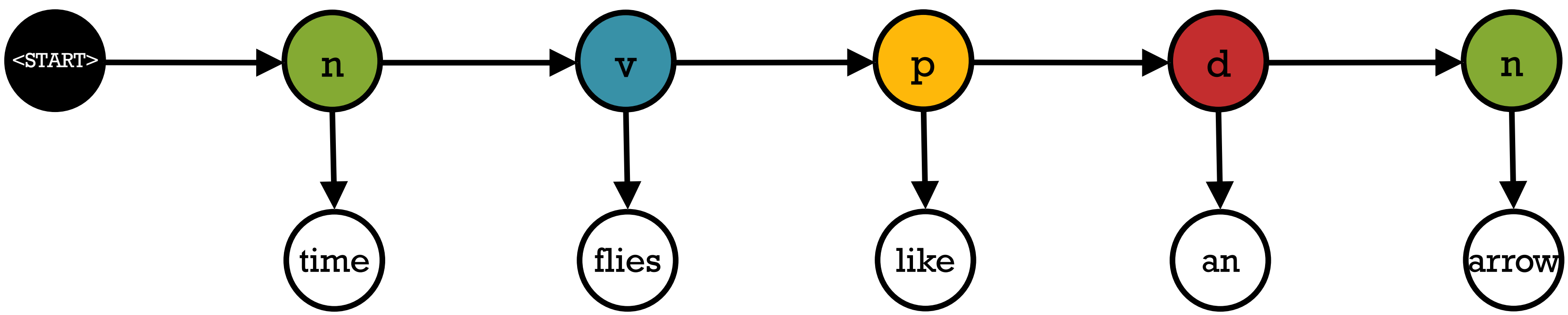


Note: This is just motivation – we'll cover these topics later!

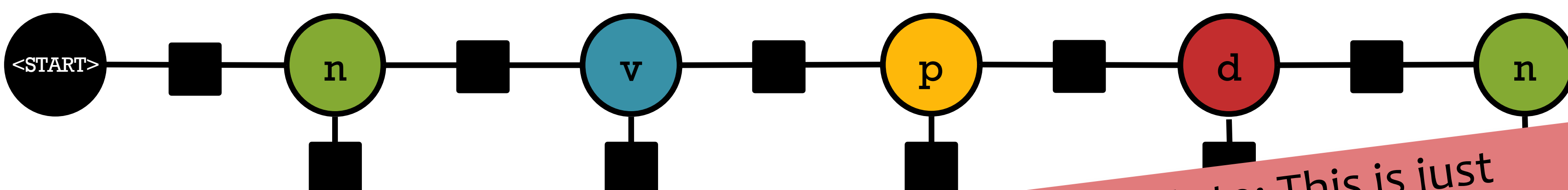
Oh, the Places You'll Use Probability!

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