Introduction to Machine Learning

Dan Roth
danroth@seas.upenn.edu|http://www.cis.upenn.edu/~danroth/|461C, 3401 Walnut

Slides were created by Dan Roth (for CIS519/419 at Penn or CS446 at UIUC),
Some slides were taken with approval from other authors who have made their ML slides available.

Penn Engineering

Course Overview

The class is recorded

Will be available on the web site

- Introduction: Basic problems and questions
- A detailed example: Linear classifiers; key algorithmic idea
- Two Basic Paradigms:
  - » Discriminative Learning & Generative/Probabilistic Learning
- Learning Protocols:
  - » Supervised; Unsupervised; Semi-supervised
- Algorithms
  - » Gradient Descent
  - » Decision Trees
  - » Linear Representations: (Perceptron; SVMs; Kernels)
  - » Neural Networks/Deep Learning
  - » Probabilistic Representations (naïve Bayes)
  - » Unsupervised /Semi supervised: EM
  - » Clustering; Dimensionality Reduction
- Modeling; Evaluation; Real world challenges
- Ethics

CIS 419/519: Applied Machine Learning

- Monday, Wednesday: 10:30pm-12:00pm On Zoom
- (My) Office hours: Mon 5-6 pm; Tue 12-1pm
- 13 TAs
- Assignments: 5 Problems set (Python Programming)
  - Weekly (light) on-line quizzes
- Weekly Discussion Sessions
- Mid Term Exam (take home)
- [Project] (We'll talk about it later)
- Final (take home)
- No real textbook:
  - Slides/Mitchell/Goldberg/Other Books/Lecture notes /Literature

Started this week
TAs Office Hours also started

HW0 !!!

Go to the web site

Be on Piazza

Registration for Class

CIS 519: What have you learned so far?

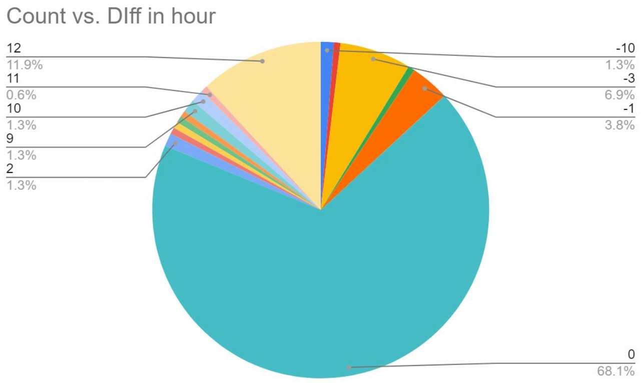
- What do you need to know:
  - Some exposure to:
    - Theory of Computation
    - Probability Theory
    - Linear Algebra
  - Programming (Python)
- Homework 0
  - If you could not comfortably deal with 2/3 of this within a few hours, please take the prerequisites first; come back next semester/year.

Participate, Ask Questions

- **Applied Machine Learning**
- Applied: mostly in HW
- **Machine learning:** mostly in class, quizzes, exams

Key Comments From Survey

- Math Background
  - We'll add recitations to help with that.
  - Still, HW0 is your yard stick
- Computation
  - We'll provide guidelines
- Time Zones
  - The vast majority of you are in "ok" times zones [ET-3,ET+12]. Please email if you are not.
- Additional time on exams
  - Penn has standard ways to deal with it; ask me if you don't know how to access it. [May not be needed, though]



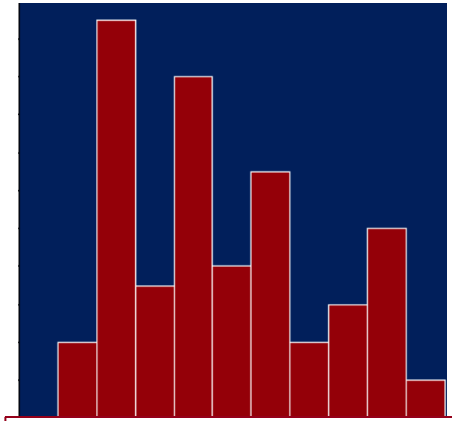
DIff in hour	Count
-10	2
-9	1
-3	11
-2	1
-1	6
0	109
2	2
4	1
5	1
7	1
8	1
9	2
10	2
11	1
12	19

CIS 519: Policies

- Cheating
  - No.
  - We take it very seriously.
- Homework:
  - Collaboration is encouraged
  - But, you have to write your own solution/code.
- Late Policy:
  - You have a credit of 4 days; That’s it.
- Grading:
  - Possible separate for grad/undergrads.
  - 40% - homework; 35%-final; 20%-midterm; 5% Quizzes
  - [Projects: 20%]
- Questions?

Class’ Web Page

Note also the Schedule Page and our Notes



A: 35-40% ; B: 40% C: 20%

Past Grade Distribution

CIS 519 on the web

- Check our class website:
  - Schedule, slides, videos, policies
    - https://www.seas.upenn.edu/~cis519/fall2020/
    - Sign up, participate in our Piazza forum:
    - Announcements and discussions
    - https://piazza.com/class/kec0q01gqceim
    - Check out our team
    - Office hours
    - [Optional] Discussion Sessions

Any additional administrative questions?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Today

- Introduce some key concepts in Machine Learning
  - At a high level
- Representation
- Supervised Learning Protocol
- New Concepts:
  - Instance Space
  - Label Space
  - Model
  - A Hypothesis Space

What is Learning?

- The Badges Game…
  - This is an example of the key learning protocol: supervised learning
- First question: Are you sure you got it?
  - Why?
- Issues:
  - Prediction or Modeling?
  - Representation
  - Problem setting
  - Background Knowledge
  - When did learning take place?
  - Algorithm

Do you that answer (the function used to generate this data)?

Yes, I am completely sure I know it.

Yes, but I am not sure it's right.

No, I don't.

Did not have time to think about it.

None of the above

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

If you know the answer, say concisely why are you sure you have the right answer? (don't give the answer, just say why you are sure)

What is Learning?

- The Badges Game…
  - This is an example of the key learning protocol: supervised learning
- First question: Are you sure you got it?
  - Why?
- Issues:
  - Prediction or Modeling?
  - Representation
  - Problem setting
  - Background Knowledge
  - When did learning take place?
  - Algorithm

Training data

+ Naoki Abe
- Myriam Abramson
+ David W. Aha
+ Kamal M. Ali
- Eric Allender
+ Dana Angluin
- Chidanand Apte
+ Minoru Asada
+ Lars Asker
+ Javed Aslam
+ Jose L. Balcazar
- Cristina Baroglio
+ Peter Bartlett
- Eric Baum
+ Welton Becket
- Shai Ben-David
+ George Berg
+ Neil Berkman
+ Malini Bhandaru
+ Bir Bhanu
+ Reinhard Blasig
- Avrim Blum
- Anselm Blumer
+ Justin Boyan
+ Carla E. Brodley
+ Nader Bshouty
- Wray Buntine
- Andrey Burago
+ Tom Bylander
+ Bill Byrne
- Claire Cardie
+ John Case
+ Jason Catlett
- Philip Chan
- Zhixiang Chen
- Chris Darken

The Badges game

+ Naoki Abe

- Eric Baum

- Conference attendees to the 1994 Machine Learning conference were given name badges labeled with + or –.
- What function was used to assign these labels?

Raw test data

Shivani Agarwal
Gerald F. DeJong
Chris Drummond
Yolanda Gil
Attilio Giordana
Jiarong Hong

J. R. Quinlan
Priscilla Rasmussen
Dan Roth
Yoram Singer
Lyle H. Ungar

Labeled test data

? Shivani Agarwal
+ Gerald F. DeJong
- Chris Drummond
+ Yolanda Gil
- Attilio Giordana
+ Jiarong Hong

- J. R. Quinlan
- Priscilla Rasmussen
+ Dan Roth
+ Yoram Singer
- Lyle H. Ungar

Which problem is easier? Yours, or the one presented to the conference attendees?

Ours

Theirs

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Which problem was easier, yours or the conference attendees (type: Ours, Theirs); why (brief reason)

If you were to write a computer program to solve this problem, what would you do?

If I ask you to write a program that solves this problem, how would you represent the input?

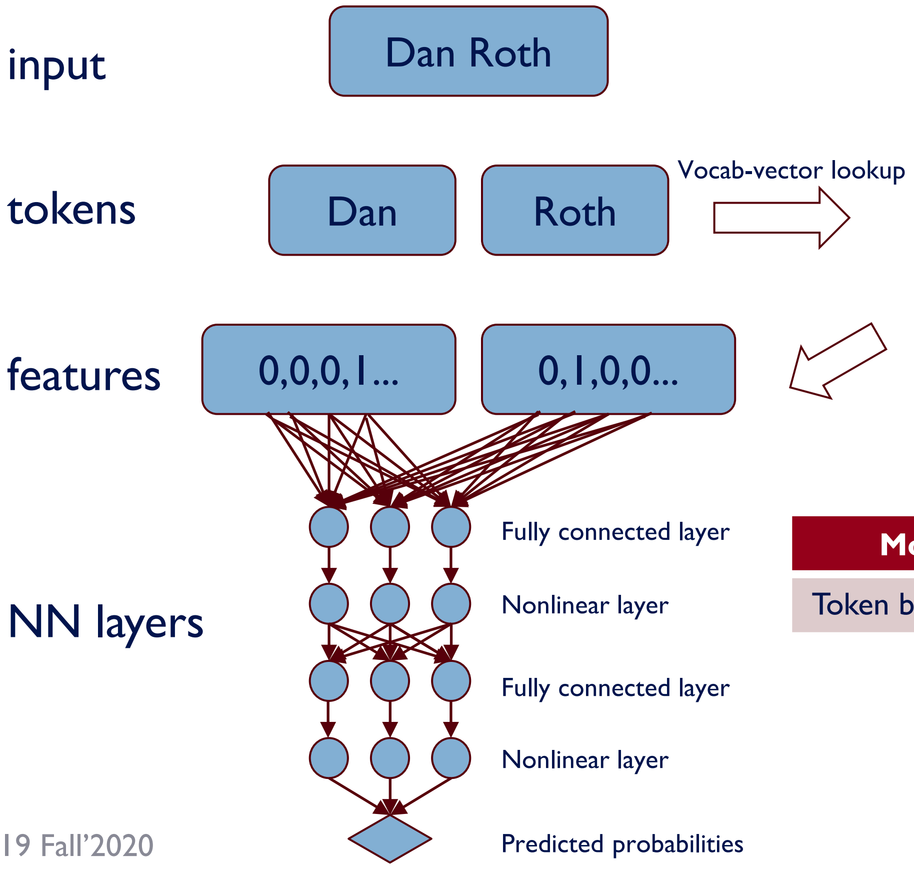
What is the function the generated the badges data?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Experiments with MLP

- MLP: Multi=Layer Perceptron (feed forward Neural Network)
- Architecture:
  - 2 layers; 32 nodes in each layer.
- Learning:
  - Loss function: Cross Entropy; optimizer: Adam
  - Learning rate: 5e-4
  - 500 epochs; batch size 16
  - Tuning: only for epochs and learning rate.
- Data:
  - 234 train / 60 test

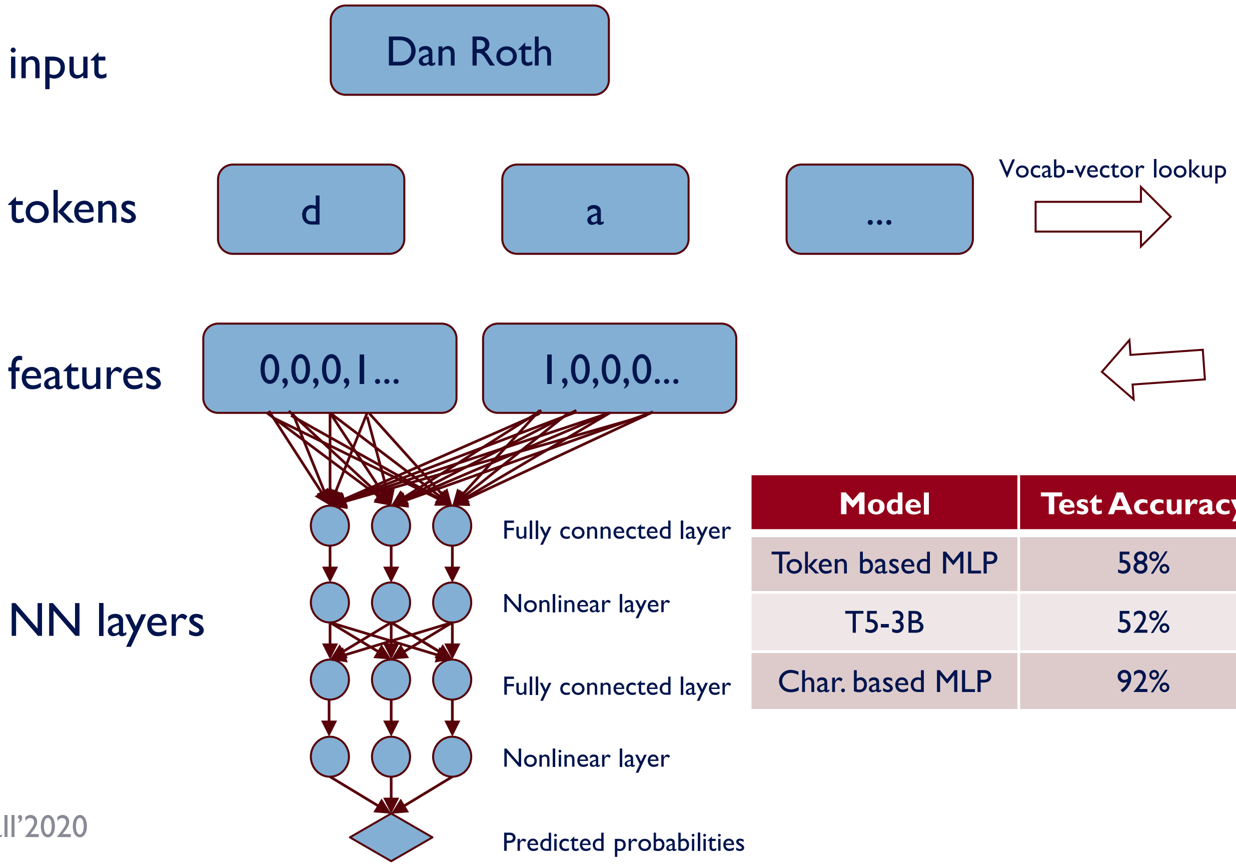
Token based MLP model



vocab	one-hot
Jane	1,0,0,0,…
…	…
Dan	0,0,0,1,…
…	
Roth	0,1,0,0,…

Model	Test Accuracy	Training Accuracy
Token based MLP	58%	100%

Character based MLP model

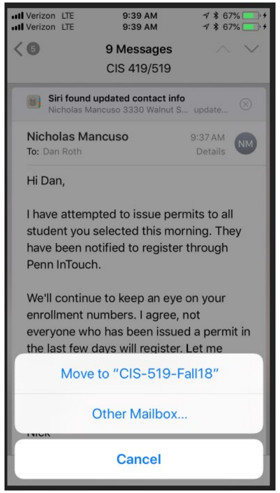


vocab	one-hot
a	1,0,0,0,...
...	...
d	0,0,0,1,...
...	
z	0,0,0,0,...

Model	Test Accuracy	Training Accuracy
Token based MLP	58%	100%
T5-3B	52%	100%
Char. based MLP	92%	100%

What is Learning

- The Badges Game…
  - This is an example of the key learning protocol: supervised learning
- First question: Are you sure you got it?
  - Why?
- Issues:
  - Which problem was easier?
    - Yours or the conference attendees?
  - Representation
    - Problem setting
  - Background Knowledge
    - When did learning take place?
  - Algorithm: can you write a program that takes this data as input and predicts the label for your name?


Verizon LTE 9:39 AM 67%
Verizon LTE 9:39 AM 67%

9 Messages
CIS 419/519

Siri found updated contact info
Nicholas Mancuso 3330 Walnut S… update…

Nicholas Mancuso 9:37 AM
To: Dan Roth Details

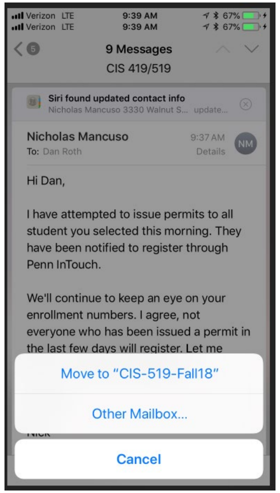
Hi Dan,

I have attempted to issue permits to all student you selected this morning. They have been notified to register through Penn InTouch.

We'll continue to keep an eye on your enrollment numbers. I agree, not everyone who has been issued a permit in the last few days will register. Let me

Other Examples

- I have a spelling checker, it came with my PC
- It plane lee marks four my revue
- Miss steaks aye can knot sea.
- Eye ran this poem threw it, your sure reel glad two no.
- Its vary polished in it's weigh
- My checker tolled me sew.
- A checker is a bless sing, it freeze yew lodes of thyme.
- It helps me right awl stiles two reed
- And aides me when aye rime.
- Each frays come posed up on my screen
- Eye trussed to bee a joule...



Verizon LTE 9:39 AM 67%
Verizon LTE 9:39 AM 67%

5 9 Messages
CIS 419/519

Siri found updated contact info
Nicholas Mancuso 3330 Walnut S... update...

Nicholas Mancuso 9:37 AM
To: Dan Roth Details NM

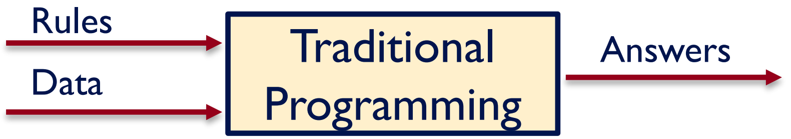
Hi Dan,

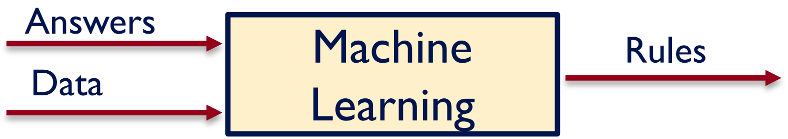
I have attempted to issue permits to all student you selected this morning. They have been notified to register through Penn InTouch.

We'll continue to keep an eye on your enrollment numbers. I agree, not everyone who has been issued a permit in the last few days will register. Let me

Machine Learning

Makes sense.
Use Machine
Learning





Sort these numbers in decreasing order

2, 4, 18, 1, 77, 0, 85

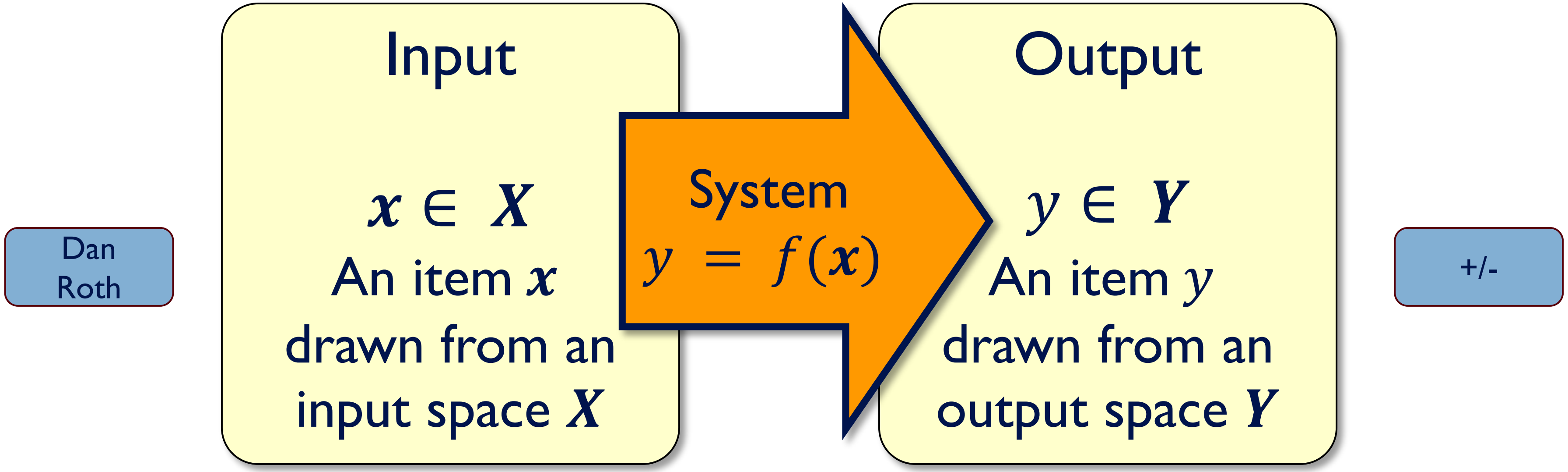
Does not make sense.
Do not use Machine
Learning

(what are the risks if you do?)

Questions? Comments?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

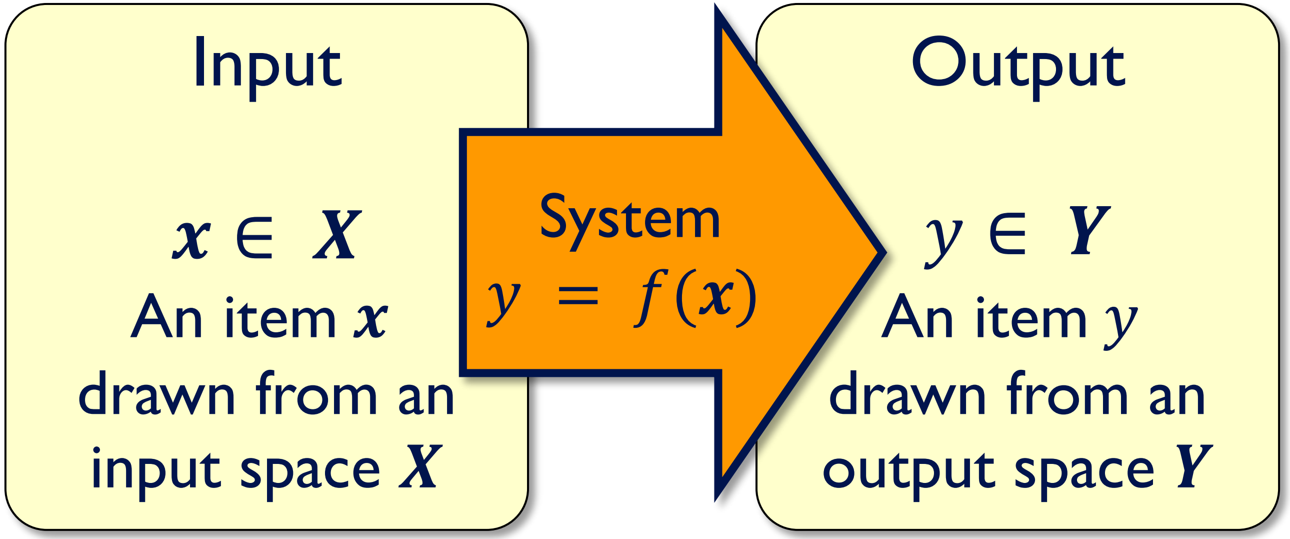
Supervised Learning



Dan Roth
Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an input space $\boldsymbol{X}$
System
$y = f(\boldsymbol{x})$
Output
$y \in \boldsymbol{Y}$
An item y drawn from an output space $\boldsymbol{Y}$
+/-

We consider systems that apply a function $f(\,)$ to input items **x** and return an output $y = f(\boldsymbol{x})$.

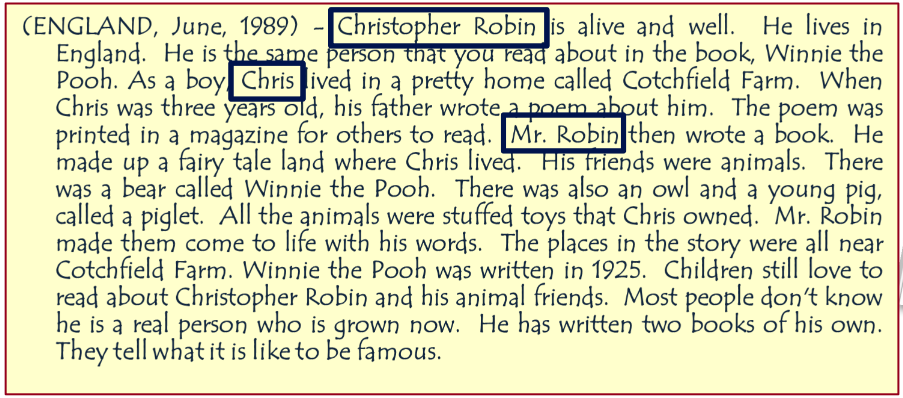
Supervised Learning


Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an input space $\boldsymbol{X}$
System
$y = f(\boldsymbol{x})$
Output
$y \in \boldsymbol{Y}$
An item y drawn from an output space $\boldsymbol{Y}$

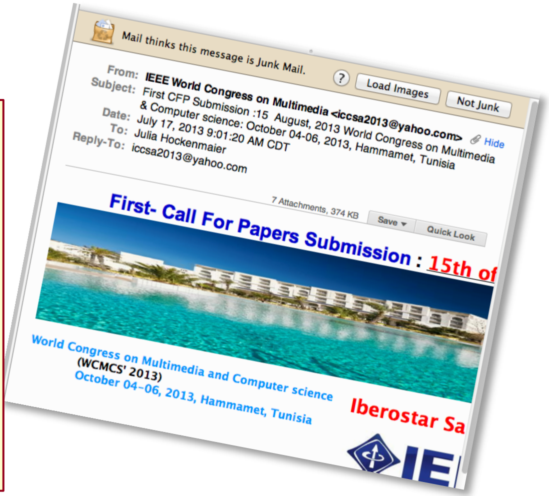
In (supervised) machine learning, we deal with systems whose $f(\boldsymbol{x})$ is learned from examples.

Why use learning?

- We typically use machine learning when the function $f(x)$ we want the system to apply is unknown to us, and we cannot "think" about it. The function could actually be simple.



(ENGLAND, June, 1989) - Christopher Robin is alive and well. He lives in England. He is the same person that you read about in the book, Winnie the Pooh. As a boy, Chris lived in a pretty home called Cotchfield Farm. When Chris was three years old, his father wrote a poem about him. The poem was printed in a magazine for others to read. Mr. Robin then wrote a book. He made up a fairy tale land where Chris lived. His friends were animals. There was a bear called Winnie the Pooh. There was also an owl and a young pig, called a piglet. All the animals were stuffed toys that Chris owned. Mr. Robin made them come to life with his words. The places in the story were all near Cotchfield Farm. Winnie the Pooh was written in 1925. Children still love to read about Christopher Robin and his animal friends. Most people don't know he is a real person who is grown now. He has written two books of his own. They tell what it is like to be famous.



Mail thinks this message is Junk Mail.

From: IEEE World Congress on Multimedia <iccsa2013@yahoo.com>
Subject: First CFP Submission :15 August, 2013 World Congress on Multimedia & Computer science: October 04-06, 2013, Hammamet, Tunisia
Date: July 17, 2013 9:01:20 AM CDT
To: Julia Hockenmaier
Reply-To: iccsa2013@yahoo.com

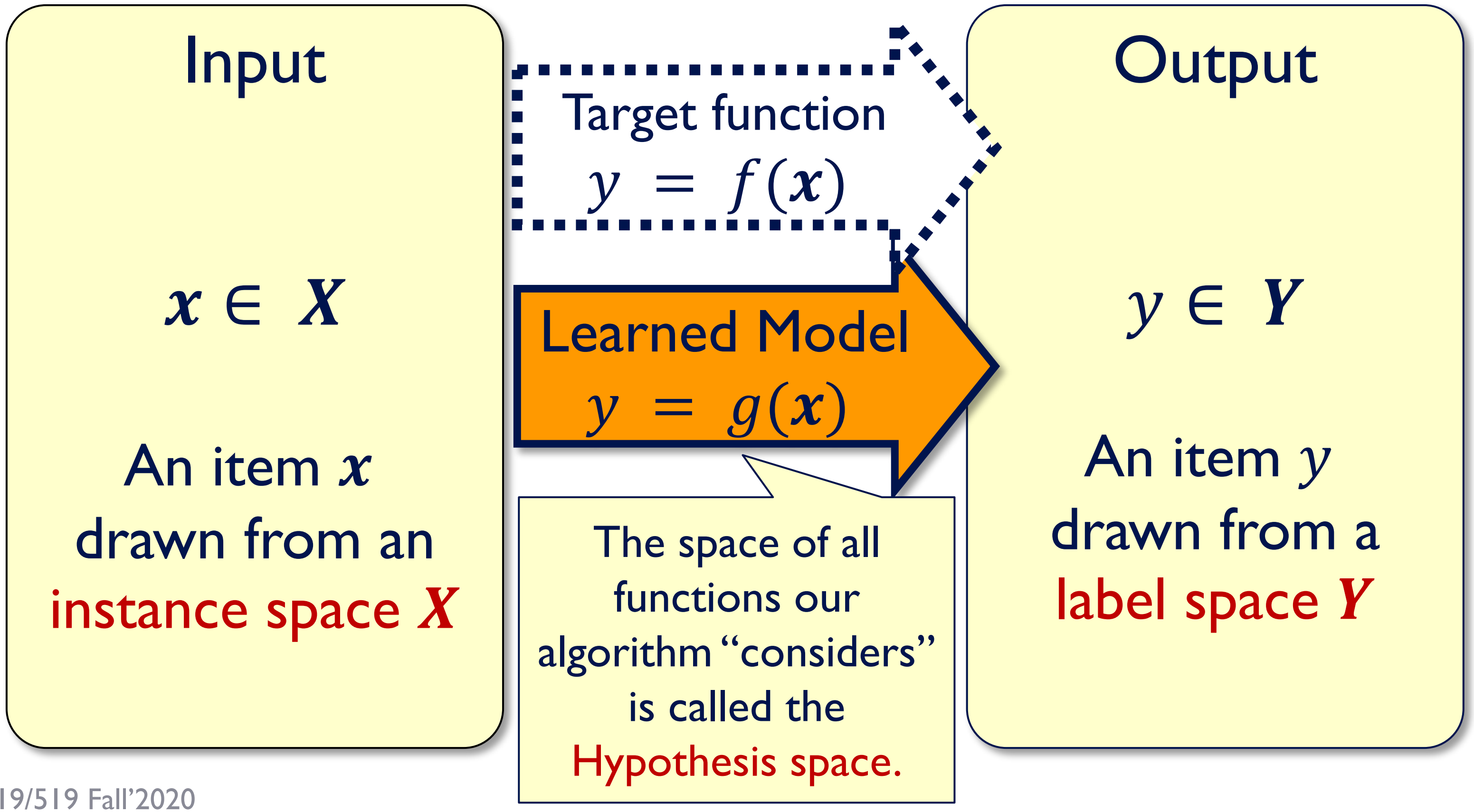
7 Attachments, 374 KB

First- Call For Papers Submission : 15th of

World Congress on Multimedia and Computer science
(WCMCS' 2013)
October 04-06, 2013, Hammamet, Tunisia

Iberostar Sa

Supervised Learning



Input

$\boldsymbol{x} \in \boldsymbol{X}$

An item $\boldsymbol{x}$ drawn from an instance space $\boldsymbol{X}$

Target function $y = f(\boldsymbol{x})$

Learned Model $y = g(\boldsymbol{x})$

The space of all functions our algorithm "considers" is called the Hypothesis space.

Output

$y \in \boldsymbol{Y}$

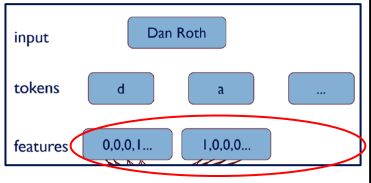
An item y drawn from a label space $\boldsymbol{Y}$

Supervised learning: Training

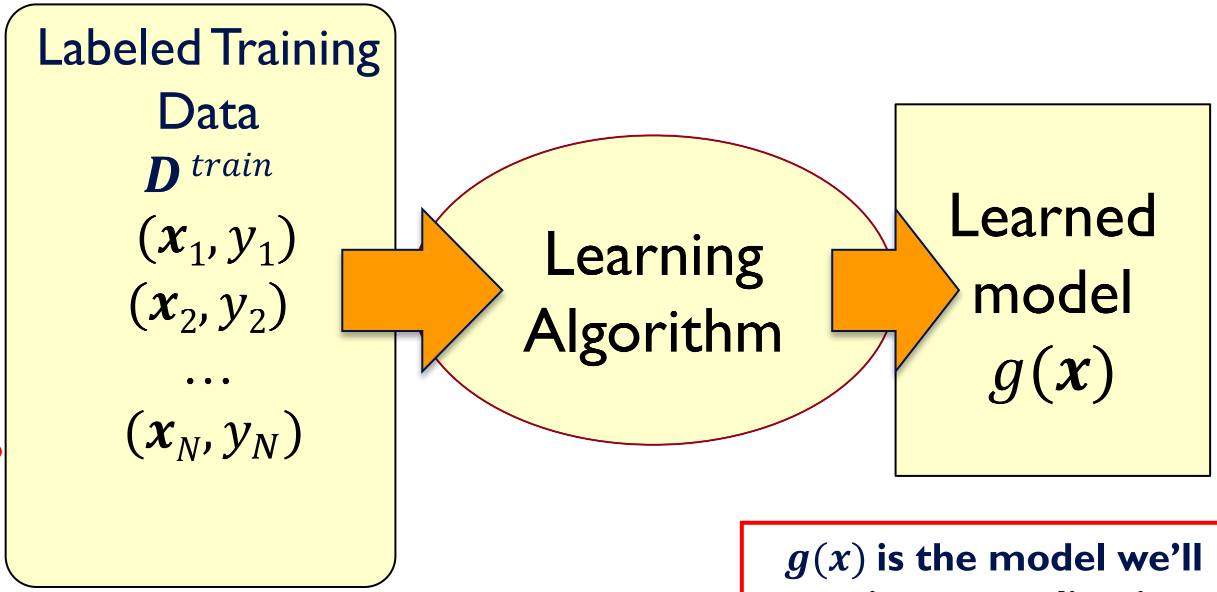
- Give the learner examples in $\boldsymbol{D}^{\,train}$
- The learner returns a model $\mathrm{g}(\boldsymbol{x})$

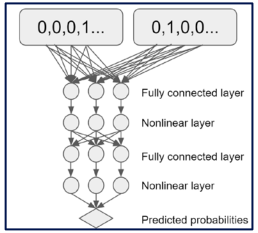
(Dan Roth, +)

An input example



An element in the instance Space


Labeled Training Data
$\boldsymbol{D}^{\,train}$
$(\boldsymbol{x}_1, y_1)$
$(\boldsymbol{x}_2, y_2)$
$\dots$
$(\boldsymbol{x}_N, y_N)$
Learning Algorithm
Learned model
$g(\boldsymbol{x})$



If
(the…character of the …token is..)
AND
(the …. is….)
then Negative.
Otherwise,
Positive.

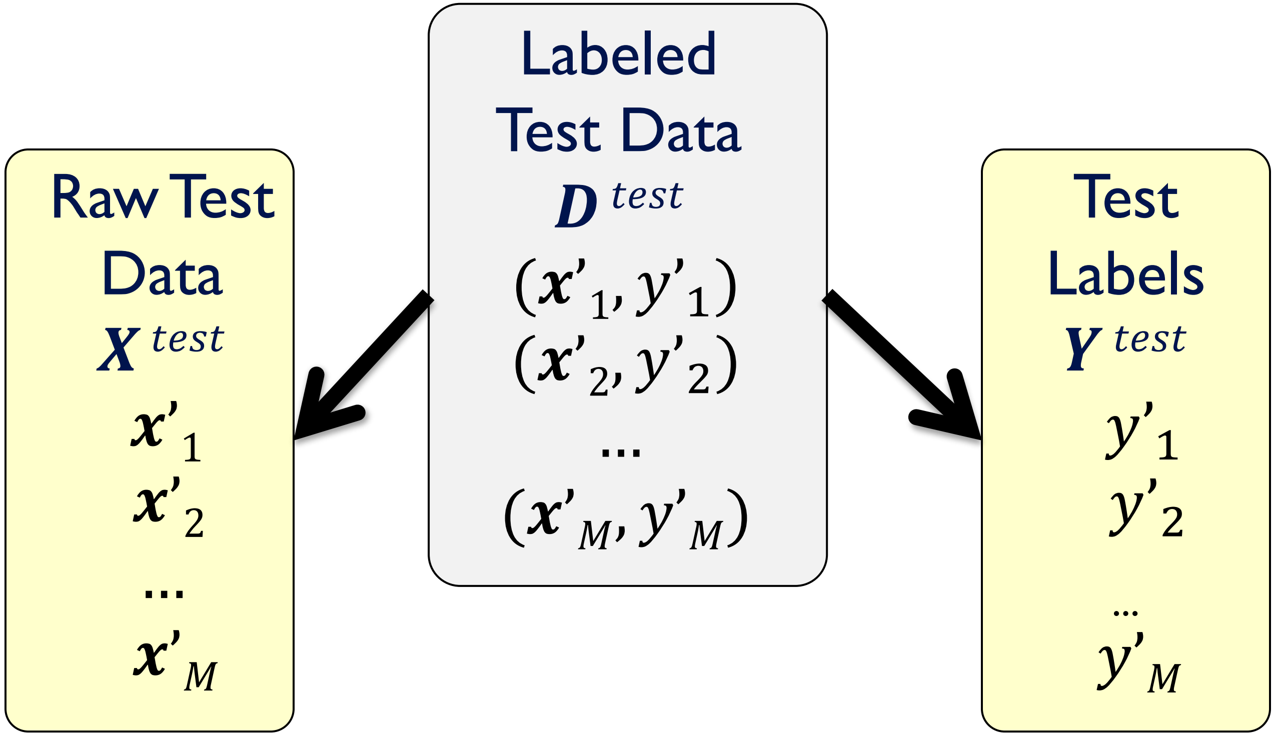
$g(x)$ is the model we'll use in our application

Supervised learning: Testing

- Reserve some labeled data for testing

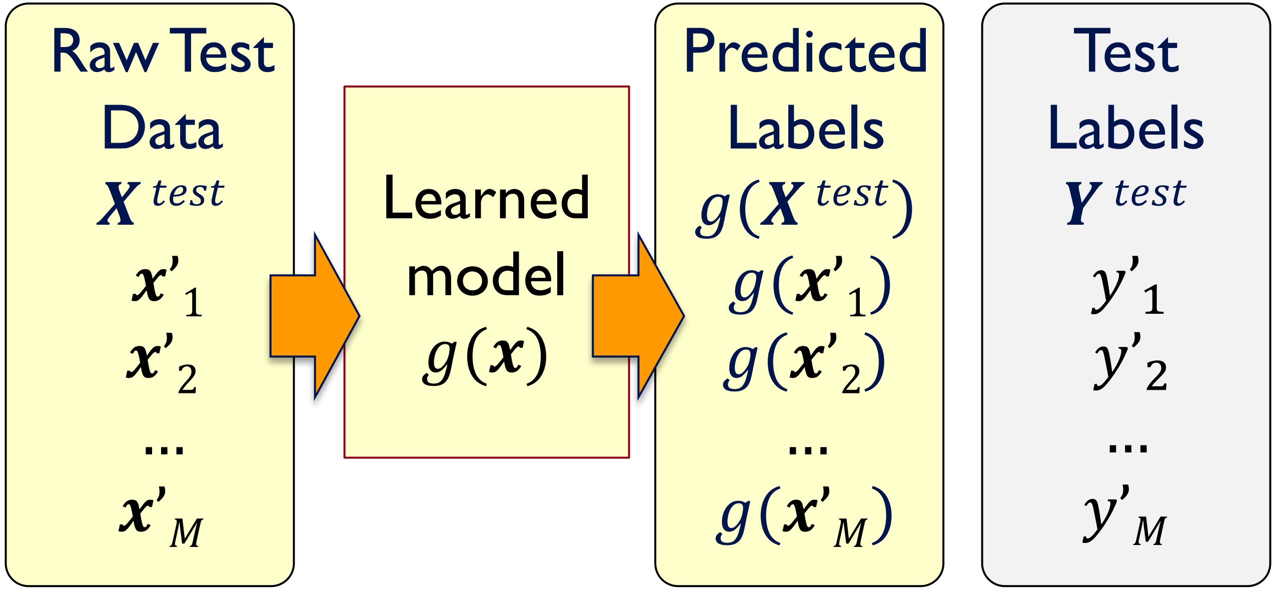
Labeled
Test Data
$\boldsymbol{D}^{test}$
$(\boldsymbol{x}'_1, y'_1)$
$(\boldsymbol{x}'_2, y'_2)$
$\ldots$
$(\boldsymbol{x}'_M, y'_M)$

Supervised learning: Testing



Supervised learning: Testing

- Apply the model to the raw test data
- Evaluate by comparing predicted labels against the test labels

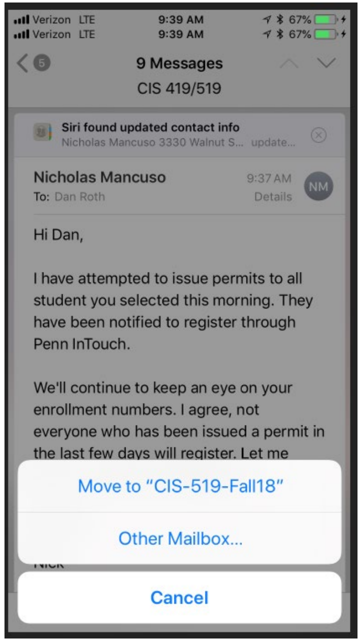


Key Issues in Machine Learning

- Modeling
  - How to formulate application problems as machine learning problems ? How to represent the data?
  - Learning Protocols (where is the data & labels coming from?)
- Representation
  - What <u>functions</u> should we learn (hypothesis spaces) ?
  - How to map raw <u>input</u> to an instance space?
    - Any rigorous way to find these? Any general approach?
- Algorithms
  - What are good algorithms?
  - How do we define success?
    - Generalization vs. over fitting
  - The computational problem

Using supervised learning

- What is our instance space?
  - Gloss: What kind of features are we using?
- What is our label space?
  - Gloss: What kind of learning task are we dealing with?
- What is our hypothesis space?
  - Gloss: What kind of functions (models) are we learning?
- What learning algorithm do we use?
  - Gloss: How do we learn the model from the labeled data?
- What is our loss function/evaluation metric?
  - Gloss: How do we measure success? What drives learning?



Verizon LTE 9:39 AM 67%
Verizon LTE 9:39 AM 67%
5
9 Messages
CIS 419/519
Siri found updated contact info
Nicholas Mancuso 3330 Walnut S... update...

Nicholas Mancuso 9:37 AM
To: Dan Roth Details

Hi Dan,

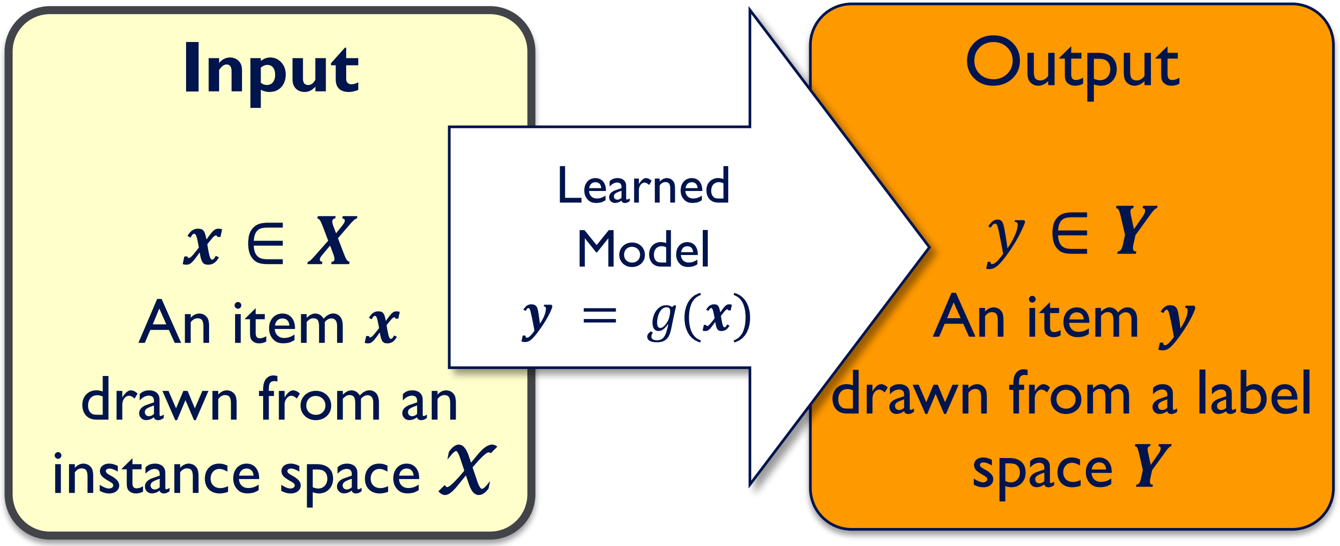
I have attempted to issue permits to all student you selected this morning. They have been notified to register through Penn InTouch.

We'll continue to keep an eye on your enrollment numbers. I agree, not everyone who has been issued a permit in the last few days will register. Let me

Questions so far?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

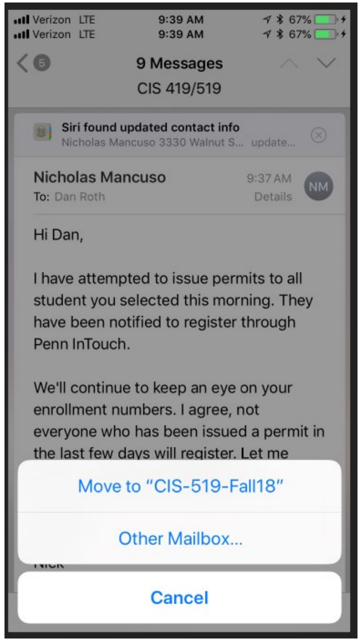
1. The instance space $\boldsymbol{X}$


Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an instance space $\mathcal{X}$
Learned Model
$\boldsymbol{y} = g(\boldsymbol{x})$
Output
$y \in \boldsymbol{Y}$
An item $\boldsymbol{y}$ drawn from a label space $\boldsymbol{Y}$

Designing an appropriate instance space $\boldsymbol{X}$ is crucial for how well we can predict y.

Instance Spaces

- What instance spaces would you consider?



Verizon LTE 9:39 AM 67%
Verizon LTE 9:39 AM 67%

9 Messages
CIS 419/519

Siri found updated contact info
Nicholas Mancuso 3330 Walnut S... update...

Nicholas Mancuso 9:37 AM
To: Dan Roth Details

Hi Dan,

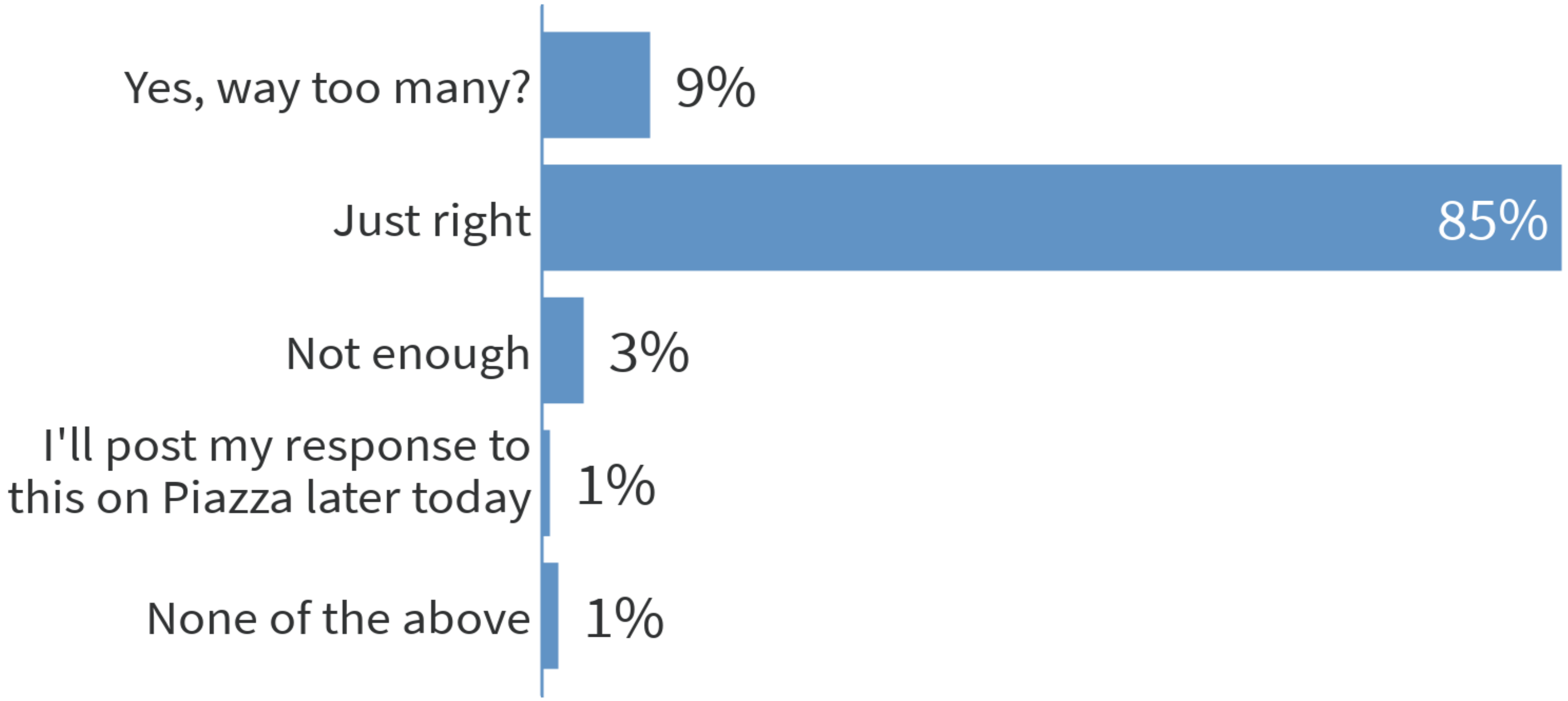
I have attempted to issue permits to all student you selected this morning. They have been notified to register through Penn InTouch.

We'll continue to keep an eye on your enrollment numbers. I agree, not everyone who has been issued a permit in the last few days will register. Let me

Nick

Suggest an instance space for email classification problem. Choose either (1) Boolean Features, (2) Numerical Features, (3) Others. Say which type you choose and give two examples.

Did I ask too many questions?



How will you rate today's lecture?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

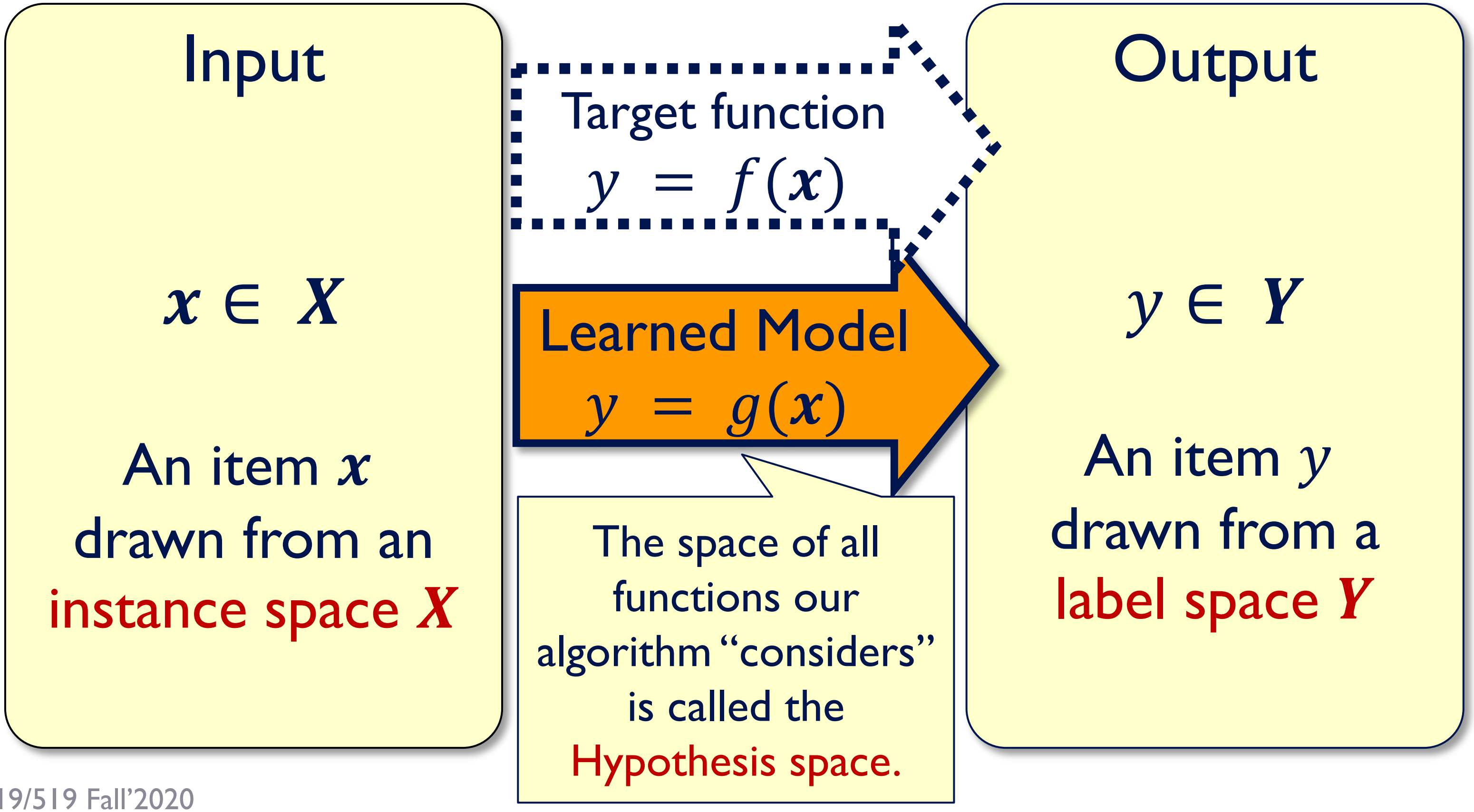
Administration (9/14/20)

Are we recording? YES!

Will be available on the web site

- HW0: please complete it. It’s mandatory (graded Pass/Fail)
- 1st quiz is out this week (Thursday night; due on Sunday night)
- HW 1 will be released next week.
- Questions?
  - Please ask/comment during class.
- Feedback/Suggestions?

Last Time: Supervised Learning



Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an instance space $\boldsymbol{X}$
Target function
$y = f(\boldsymbol{x})$
Learned Model
$y = g(\boldsymbol{x})$
The space of all functions our algorithm "considers" is called the Hypothesis space.
Output
$y \in \boldsymbol{Y}$
An item y drawn from a label space $\boldsymbol{Y}$

Key Issues in Machine Learning

- Modeling
  - How to formulate application problems as machine learning problems ? How to represent the data?
  - Learning Protocols (where is the data & labels coming from?)
- Representation
  - What functions should we learn (hypothesis spaces) ?
  - How to map raw input to an instance space?
    - Any rigorous way to find these? Any general approach?
- Algorithms
  - What are good algorithms?
  - How do we define success?
    - Generalization vs. over fitting
  - The computational problem

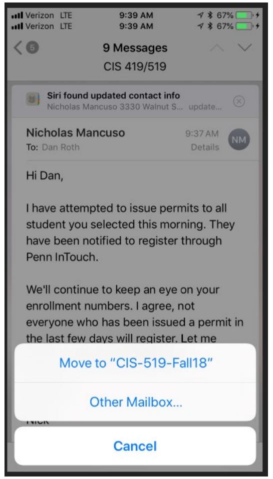
Today

- We’ll go in more details into the concepts introduced last time
  - Instance Space & Hypothesis Space
  - (since we’ll keep, for the most part, the labels to be Boolean {Yes/No})
- We’ll begin to formalize
  - Modeling and Representation
- We’ll see our first Learning Algorithm

1. The instance space $\boldsymbol{X}$

- When we apply machine learning to a task, we first need to define the instance space $\boldsymbol{X}$.
- Instances $\boldsymbol{x} \in \boldsymbol{X}$ are defined by features:
  - Boolean features:
    - » Is there a folder named after the sender?
    - » Does this email contain the word 'class'?
    - » Does this email contain the word 'waiting'?
    - » Does this email contain the word 'class' and the word 'waiting'?
  - Numerical features:
    - » How often does 'learning' occur in this email?
    - » How long is the email?
    - » How many emails have I seen from this sender over the last day/week/month?
  - Bag of tokens
    - » Just list all the **tokens** in the input

Does it add anything if you already have the previous two features?


Verizon LTE 9:39 AM 67%
Verizon LTE 9:39 AM 67%
9 Messages
CIS 419/519
Siri found updated contact info
Nicholas Mancuso 3330 Walnut S... update...
Nicholas Mancuso 9:37 AM
To: Dan Roth Details

Hi Dan,

I have attempted to issue permits to all student you selected this morning. They have been notified to register through Penn InTouch.

We'll continue to keep an eye on your enrollment numbers. I agree, not everyone who has been issued a permit in the last few days will register. Let me

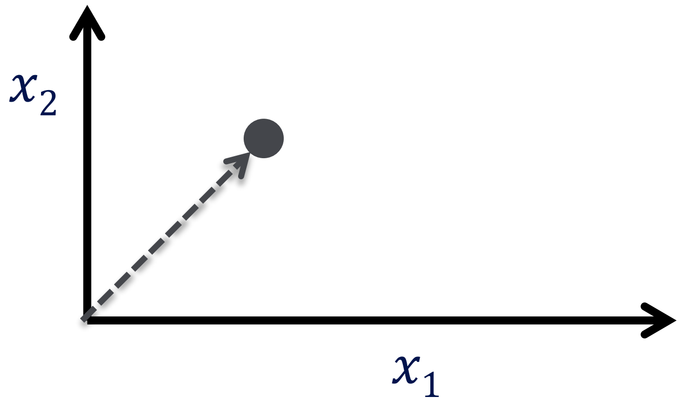
Move to "CIS-519-Fall18"
Other Mailbox...
Cancel

What’s $\boldsymbol{X}$ for the Badges game?

- Possible features:
    - Gender/age/country of the person?
    - Length of their first or last name?
    - Does the name contain letter ‘x’?
    - How many vowels does their name contain?
    - Is the n-th letter a vowel?
    - Height;
    - Shoe size

$\boldsymbol{X}$ as a vector space

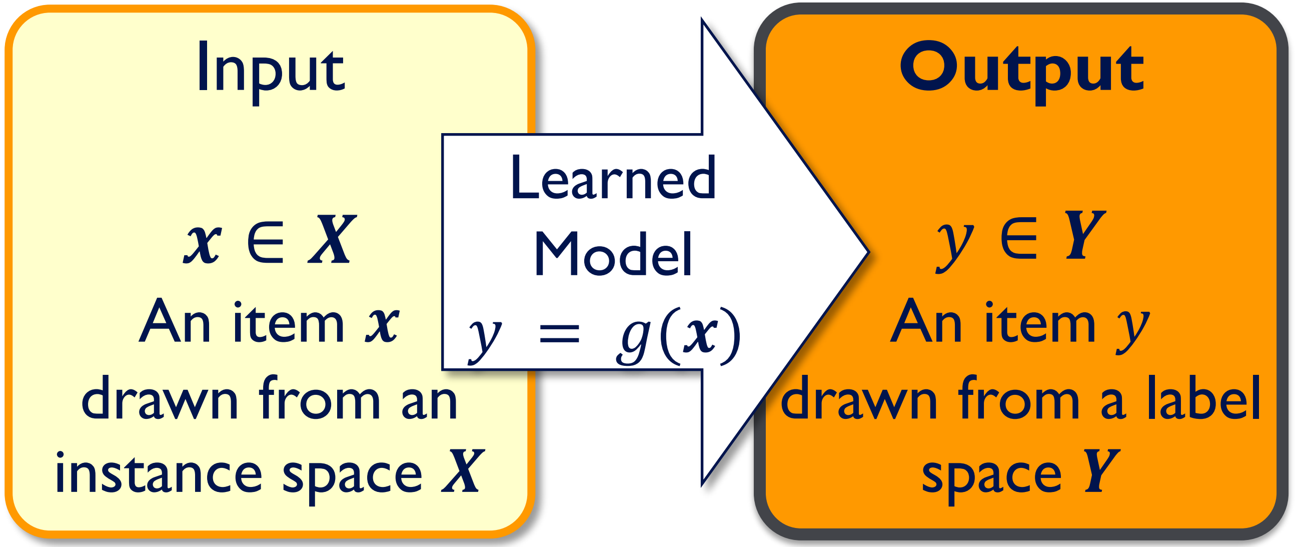
- $\boldsymbol{X}$ is an N-dimensional vector space (e.g. $\{0,1\}^N$, $\mathbb{R}^N$)
  - Each dimension = one feature.
- Each $\boldsymbol{x}$ is a feature vector (hence the boldface $\boldsymbol{x}$).
- Think of $\boldsymbol{x} = [x_1 \dots x_N]$ as a point in $\boldsymbol{X}$:



Good features are essential

- The choice of features is crucial for how well a task can be learned.
  - In many application areas (language, vision, etc.), a lot of work goes into designing suitable features.
  - This requires domain expertise.
- Think about the badges game
  - what if you were focusing on visual features?
  - Recall the two neural network examples: Token-based and Character-based
- We can’t teach you what specific features to use for your task.
  - But we will touch on some general principles

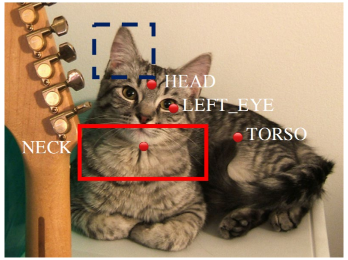
2. The label space $\boldsymbol{Y}$

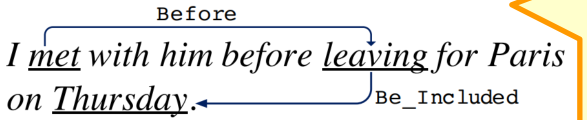

Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an instance space $\boldsymbol{X}$
Learned Model
$y = g(\boldsymbol{x})$
Output
$y \in \boldsymbol{Y}$
An item y drawn from a label space $\boldsymbol{Y}$

The label space $\boldsymbol{Y}$ determines what *kind* of supervised learning task we are dealing with

Supervised learning tasks I

- Output labels $y \in Y$ are categorical:
  - Binary classification: Two possible labels
  - Multiclass classification: k possible labels
  - Output labels $y \in Y$ are <u>structured objects</u> (sequences of labels, parse trees, graphs, etc.)
    - Structure learning: multiple labels that are related (thus constrained)

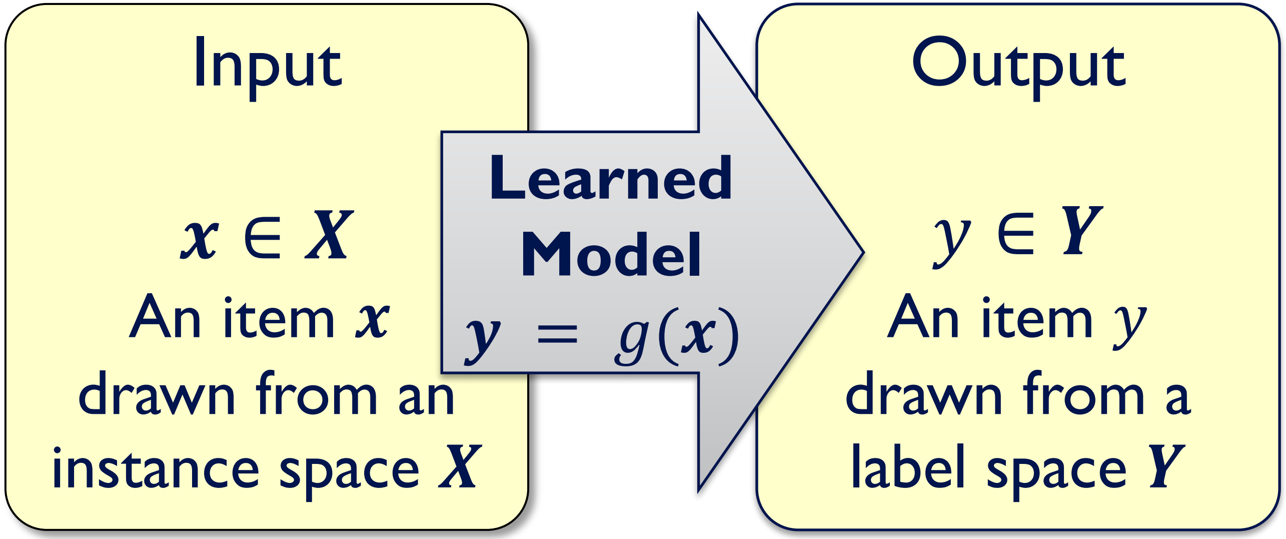




I met with him before leaving for Paris on Thursday.

Three events. When classifying the temporal relations between them we need to account for the relations between them.

3. The model $g(\boldsymbol{x})$



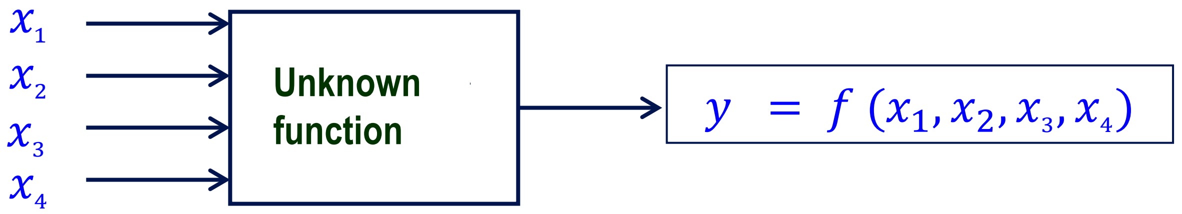
Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an instance space $\boldsymbol{X}$
Learned Model
$\boldsymbol{y} = g(\boldsymbol{x})$
Output
$y \in \boldsymbol{Y}$
An item y drawn from a label space $\boldsymbol{Y}$

We need to choose what *kind* of model we want to learn

Questions so far?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function? What is it?

Suggest a target function given this data (write it in words or mathematical symbols)

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Hypothesis Space

Complete Ignorance:

There are $2^{16} = 65536$ possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

After observing seven examples we still have 2^9 possibilities for f

Is Learning Possible?

- There are $|\boldsymbol{Y}|^{|\boldsymbol{X}|}$ possible functions $f(\boldsymbol{x})$ from the instance space $\boldsymbol{X}$ to the label space $\boldsymbol{Y}$.
- Learners typically consider only a *subset* of the functions from $\boldsymbol{X}$ to $\boldsymbol{Y}$, called the hypothesis space $\boldsymbol{H}$. $\boldsymbol{H} \subseteq |\boldsymbol{Y}|^{|\boldsymbol{X}|}$

Example	x_1	x_2	x_3	x_4	y
1	0	0	0	0	?
2	0	0	0	1	?
	0	0	1	0	0
	0	0	1	1	1
	0	1	0	0	0
	0	1	0	1	0
	0	1	1	0	0
	0	1	1	1	?
	1	0	0	0	?
	1	0	0	1	1
	1	0	1	0	?
	1	0	1	1	?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
16	1	1	1	1	?

Hypothesis Space (2)

Simple Rules: There are only 16 simple **conjunctive rules** of the form $y = x_i \wedge x_j \wedge x_k$

Rule	Counterexample
$y = c$	
x_1	1100 0
x_2	0100 0
x_3	0110 0
x_4	0101 1
$x_1 \wedge x_2$	1100 0
$x_1 \wedge x_3$	0011 1
$x_1 \wedge x_4$	0011 1

Rule	Counterexample
$x_2 \wedge x_3$	0011 1
$x_2 \wedge x_4$	0011 1
$x_3 \wedge x_4$	1001 1
$x_1 \wedge x_2 \wedge x_3$	0011 1
$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

No simple rule explains the data. The same is true for **simple clauses (disjunctions)**.

Hypothesis Space (3)

Don't worry, this function is actually a neural network…

m-of-n rules: There are 32 possible rules of the form "$y = 1$ if and only if at least m of the following n variables are 1"

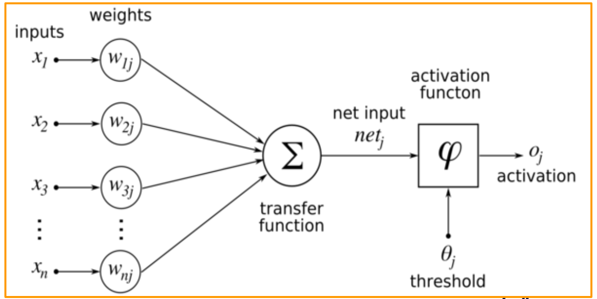
Notation: 2 variables from the set on the left.
Value: Index of the counterexample.

variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	-	-	-
$\{x_2\}$	2	-	-	-
$\{x_3\}$	1	-	-	-
$\{x_4\}$	7	-	-	-
$\{x_1, x_2\}$	2	3	-	-
$\{x_1, x_3\}$	1	3	-	-
$\{x_1, x_4\}$	6	3	-	-
$\{x_2, x_3\}$	2	3	-	-

variables	1-of	2-of	3-of	4-of
$\{x_2, x_4\}$	2	3	-	-
$\{x_3, x_4\}$	4	4	-	-
$\{x_1, x_2, x_3\}$	1	3	3	-
$\{x_1, x_2, x_4\}$	2	3	3	-
$\{x_1, x_3, x_4\}$	1	* * *	3	-
$\{x_2, x_3, x_4\}$	1	5	3	-
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

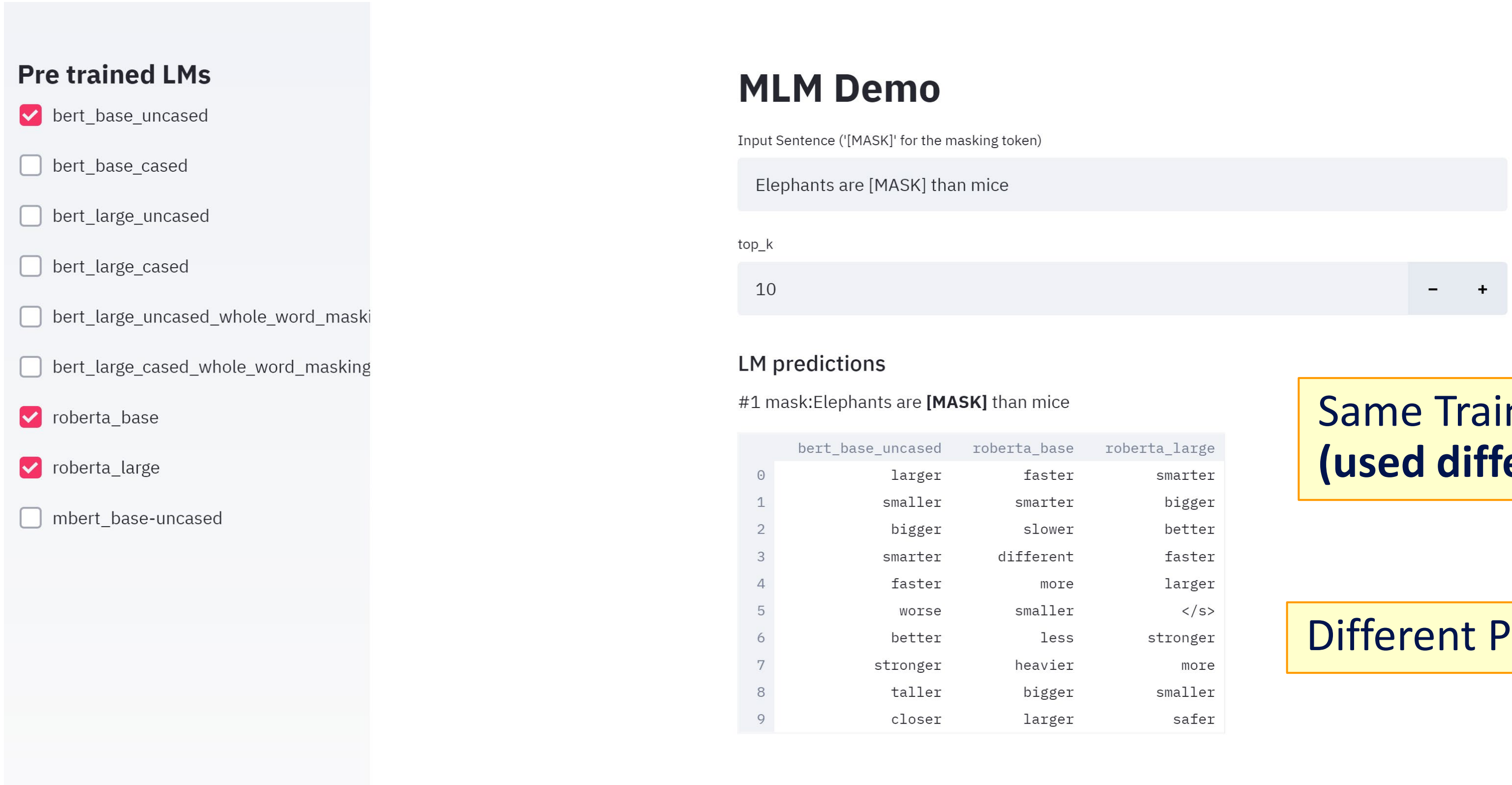
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Found a consistent hypothesis!



Is this Typical (1)?

- Language Models: https://nlp.biu.ac.il/~lazary/lms/



Pre trained LMs

- [✓] bert_base_uncased
- [ ] bert_base_cased
- [ ] bert_large_uncased
- [ ] bert_large_cased
- [ ] bert_large_uncased_whole_word_maski
- [ ] bert_large_cased_whole_word_masking
- [✓] roberta_base
- [✓] roberta_large
- [ ] mbert_base-uncased

MLM Demo

Input Sentence ('[MASK]' for the masking token)

Elephants are [MASK] than mice

top_k

10

LM predictions

#1 mask:Elephants are **[MASK]** than mice

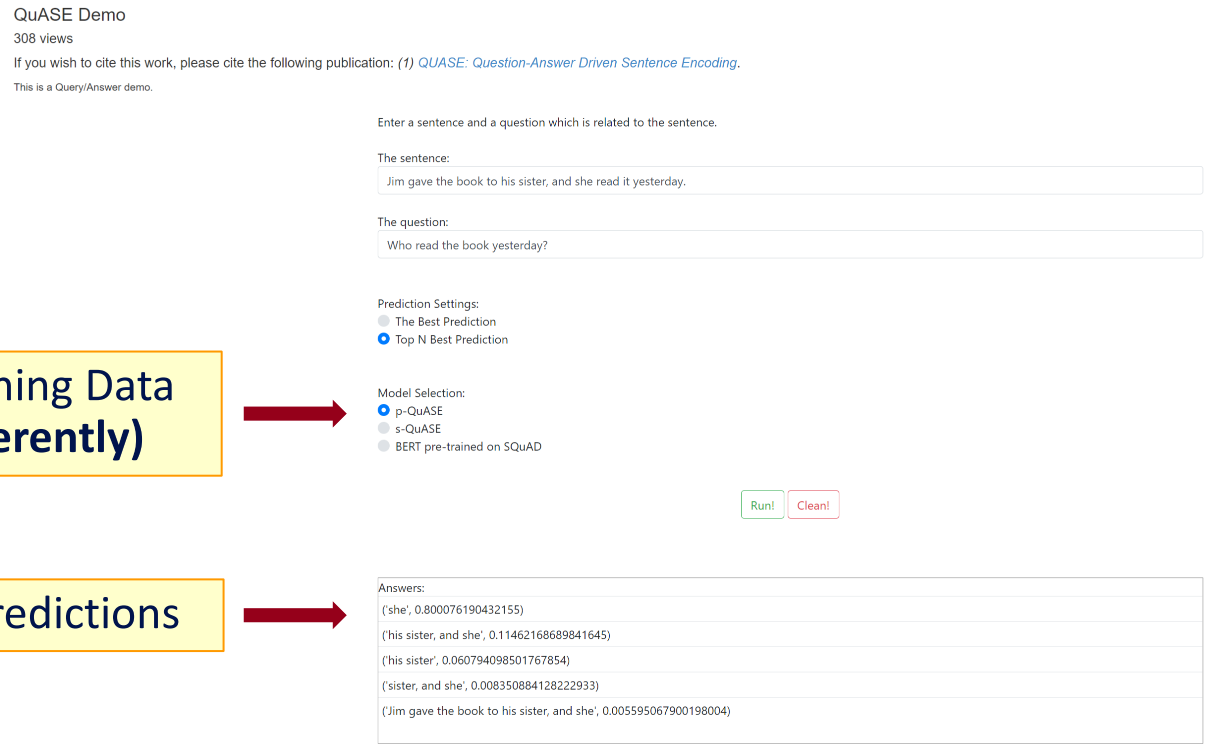
	bert_base_uncased	roberta_base	roberta_large
0	larger	faster	smarter
1	smaller	smarter	bigger
2	bigger	slower	better
3	smarter	different	faster
4	faster	more	larger
5	worse	smaller	</s>
6	better	less	stronger
7	stronger	heavier	more
8	taller	bigger	smaller
9	closer	larger	safer

Same Training Data **(used differently)**

Different Predictions

Is this Typical (2)?

- Question Answering Models: https://cogcomp.seas.upenn.edu/page/demo_view/QuASE



QuASE Demo

308 views

If you wish to cite this work, please cite the following publication: *(1) QUASE: Question-Answer Driven Sentence Encoding*.

This is a Query/Answer demo.

Enter a sentence and a question which is related to the sentence.

The sentence:

Jim gave the book to his sister, and she read it yesterday.

The question:

Who read the book yesterday?

Prediction Settings:
- The Best Prediction
- [✓ Top N Best Prediction]

Model Selection:
- [✓ p-QuASE]
- s-QuASE
- BERT pre-trained on SQuAD

```
Answers:
('she', 0.800076190432155)
('his sister, and she', 0.11462168689841645)
('his sister', 0.060794098501767854)
('sister, and she', 0.008350884128222933)
('Jim gave the book to his sister, and she', 0.005595067900198004)
```

Same Training Data **(used differently)**

Different Predictions

Views of Learning

- Learning is the removal of our remaining uncertainty:
  - Suppose we knew that the unknown function was an m-of-n Boolean function, then we could use the training data to infer which function it is.
- Learning requires guessing a good hypothesis class:
  - We can start with a very small class and enlarge it until it contains a hypothesis that fits the data.
  - The hypothesis set selection could also happen due to algorithmic bias
- We could be wrong !
  - Our prior knowledge might be wrong:
    - $y = x_4 \wedge one\ of\ \{x_1, x_3\}$ is also consistent
  - Our guess of the hypothesis space could be wrong

- If this is the unknown function, then we will make errors when we are given new examples, and are asked to predict the value of the function

General strategies for Machine Learning

- Develop flexible hypothesis spaces:
  - Decision trees, neural networks, nested collections.
  - Constraining the hypothesis space is done algorithmically
- Develop representation languages for restricted classes of functions:
  - Serve to limit the expressivity of the target models
  - E.g., Functional representation (n-of-m); Grammars; linear functions; stochastic models;
  - Get flexibility by augmenting the feature space
- In either case:
  - Develop algorithms for finding a hypothesis in our hypothesis space, that fits the data
  - And hope that they will generalize well

Key Issues in Machine Learning

- Modeling
  - How to formulate application problems as machine learning problems ? How to represent the data?
  - Learning Protocols (where is the data & labels coming from?)
- Representation
  - What functions should we learn (hypothesis spaces) ?
  - How to map raw input to an instance space?
  - Any rigorous way to find these? Any general approach?
- Algorithms
  - What are good algorithms?
  - How do we define success?
  - Generalization Vs. over fitting
  - The computational problem

An Example: Context Sensitive Spelling

- I don't know {whether, weather} to laugh or cry

This is the Modeling Step

<u>**How can we make this a learning problem?**</u>

What is the hypothesis space?

Input/Instance space?

- We will look for a function
  f: Sentences→ {whether, weather}
- We need to define the **domain** of this function better.

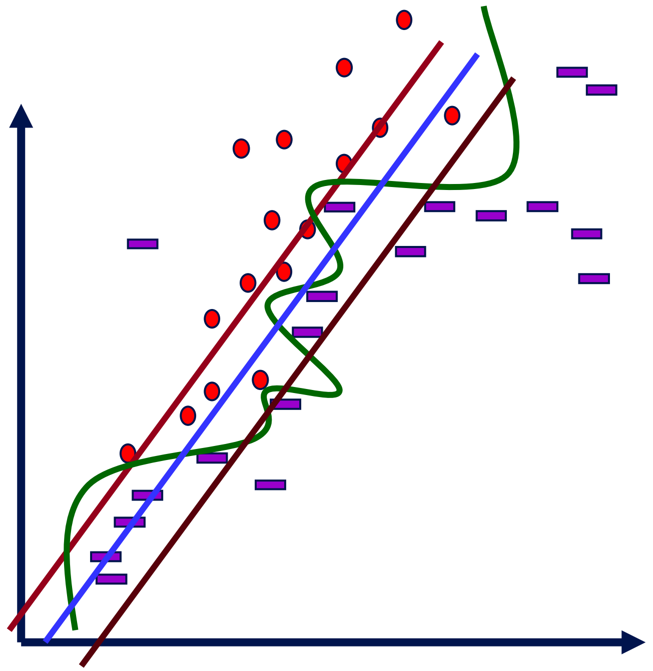
- <u>An option</u>: For each word w in English define a Boolean feature x_w :
  $[x_w = 1]$ iff w is in the sentence
- This maps a sentence to a point in $\{0,1\}^{50,000}$
- In this space: some points are <u>whether</u> points, some are <u>weather</u> points

Learning Protocol?

Supervised? Unsupervised?

Representation Step: What's Good?

- Learning problem:
  - Find a function that best separates the data
- What function?
- What's best?
- (How to find it?)



Which of the functions do you think is best here?

Must be the green one; it separates the data perfectly

Definitely not Green, it will not do well on new data

Red, not so good now, but will generalize well

Blue; simpler than green, and seems to be better that red and brown

No idea

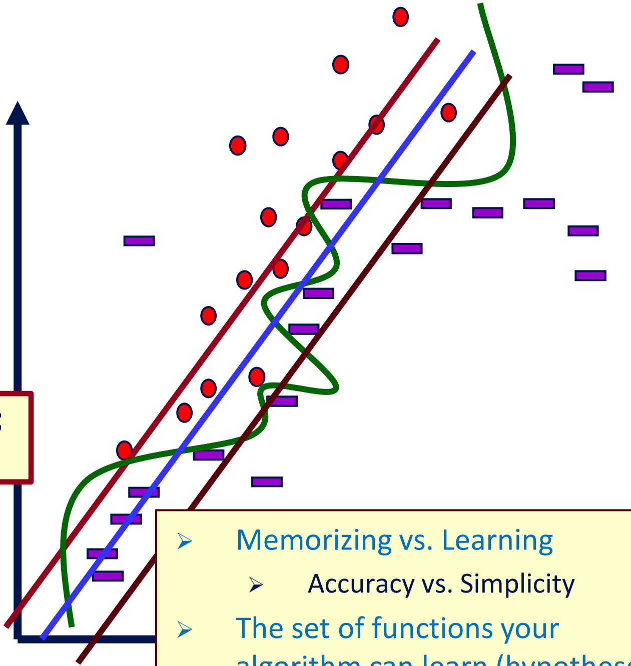
Representation Step: What's Good?

- Learning problem:
  - Find a function that best separates the data
- What function?
- What's best?
- (How to find it?)

$$\boldsymbol{w}^T \cdot \boldsymbol{x} = \sum_{i=1}^{n} w_i x_i$$

Linear = linear in the feature space
$\boldsymbol{x}$ = data representation;
$\boldsymbol{w}$ = the classifier: a weight for each feature
($\boldsymbol{w}, \boldsymbol{x}$, column vectors of dimensionality n)
The prediction: $y = \text{sgn}\{\boldsymbol{w}^T \boldsymbol{x}\}$

$\text{sgn}(z) = 0 \; if \; z < 0$;
1 otherwise



- A possibility: Define the learning problem to be:
  - A (linear) function that best separates the data

> Memorizing vs. Learning
>> Accuracy vs. Simplicity

> The set of functions your algorithm can learn (hypotheses space) determines how the learned model will do.

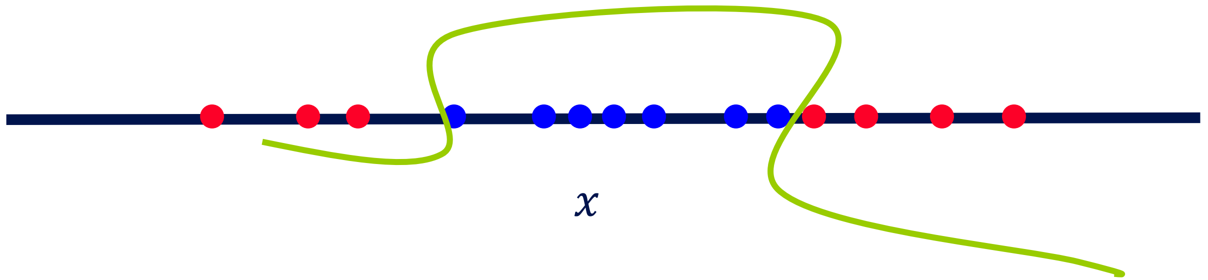
> Will do on what?
>> Impact on Generalization

A Small Detour

- Let’s talk a bit about expressivity of functions

Functions Can be Made Linear

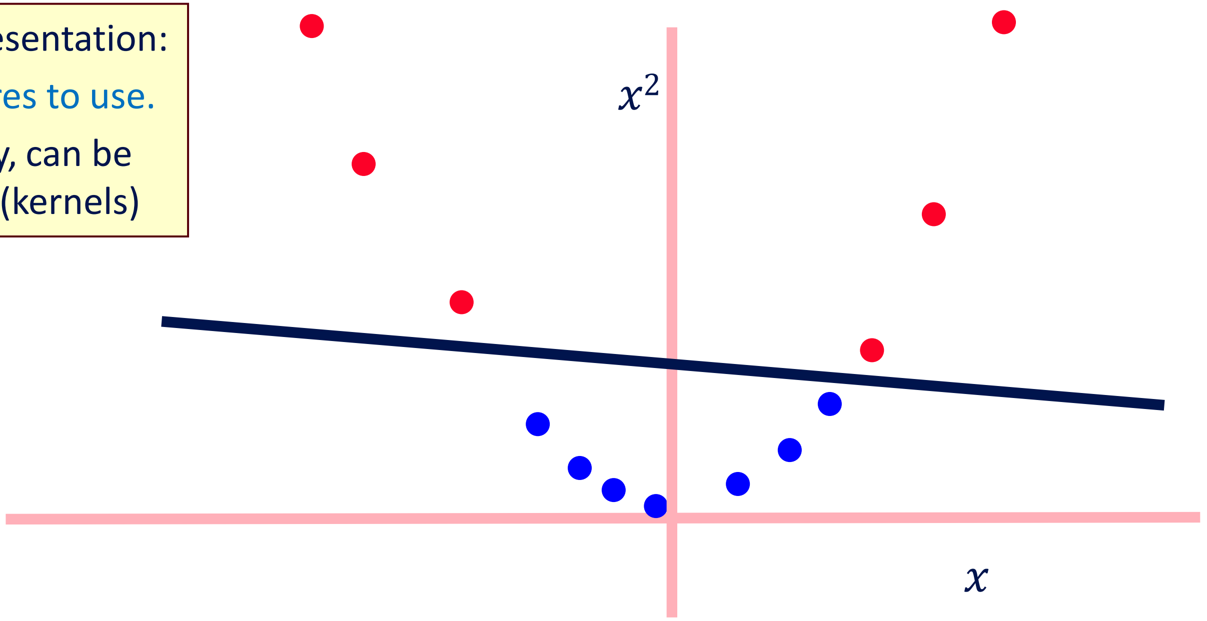
- Data points are not linearly separable in one dimension
- Not separable if you insist on using a specific class of functions (e.g., linear)



Blown Up Feature Space

- But, we can change the way we represent the data
- Data are separable in $< x, x^2 >$ space

> - Key issue: Representation:
>   - what features to use.
> - Computationally, can be done implicitly (kernels)

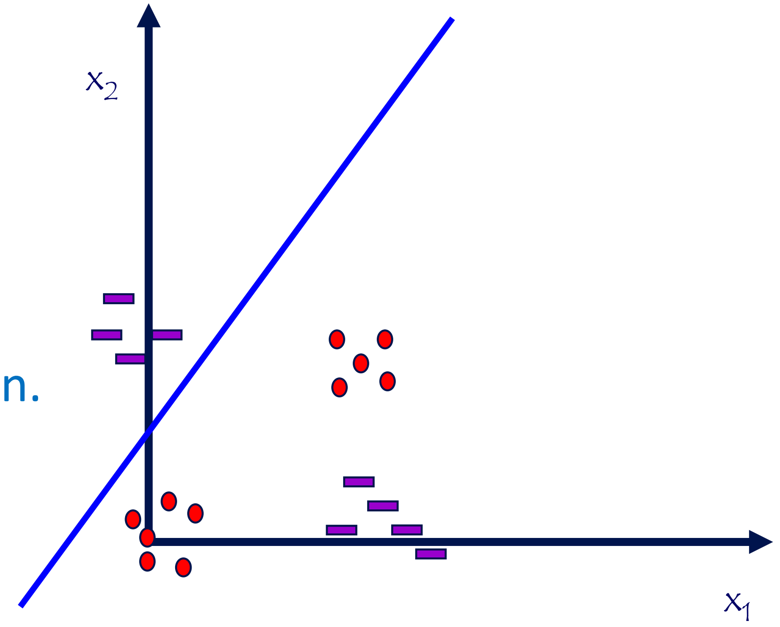


Similar Idea in a Discrete Space: Exclusive-OR (XOR)

- The function $(x_1 \wedge x_2)$ is linear (why?)
- But:
  - $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$
- Is not linear

- Note: this is a special case of a parity function.
- $x_i \in \{0,1\}$
- $f(x_1, x_2, \ldots, x_n) = 1$
  iff $\sum x_i$ is even

- This function is not linearly separable.



Linear Functions

- $f(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^T \cdot \boldsymbol{x} \ - \theta\} = \mathrm{sgn}\{\sum_{i=1}^{n} w_i x_i \ - \theta\}$
- Many functions are Linear

All variables here are Boolean variables

  – Conjunctions (over $\boldsymbol{x}^T = \ (x_1, x_2, \ldots x_5)$):
    - $y = x_1 \wedge x_3 \wedge x_5$
    - Dogs = have four legs AND Bark AND have a tails AND….
      – A bit simplistic, but you get the point…

  – Being a linear function means that
    - There is a set of weights (vector $\boldsymbol{w}$) and a threshold θ
    - So that $y = x_1 \wedge x_3 \wedge x_5$ is 1 if and only if $\mathrm{sgn}(\boldsymbol{w}^T \cdot \boldsymbol{x} \ - \theta\}$ = 1

What do I mean when I say that the function x1 ^ x3 ^ x5 (read: X1 AND X3 AND x5) is a linear function? Give a set of weights (w1,...w5) and a threshold T, that show that.

Linear Functions

All variables here are Boolean variables

- $f(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^T \cdot \boldsymbol{x} \ - \theta\} = \mathrm{sgn}\{\sum_{i=1}^{n} w_i x_i \ - \theta\}$
- Many functions are Linear
  - Conjunctions:
    - $y = x_1 \wedge x_3 \wedge x_5$
    - $y = \mathrm{sgn}\{1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 \ - 3\}; \boldsymbol{w} = (1{,}0{,}1{,}0{,}1) \ \ \theta = 3$
  - Disjunctions
    - Do it yourself
  - At least m of n:
    - $y = at\ least\ 2\ of\ \{x_1, x_3, x_5\}$
    - $y = \mathrm{sgn}\{1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 \ - 2\}; \boldsymbol{w} = (1{,}0{,}1{,}0{,}1) \ \ \theta = 2$
- Many functions are not
  - Xor: $y = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$
  - Non trivial DNF: $y = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$
- But can be made linear (it is linear if we invent new variables)
  - $z_1 = x_1 \wedge x_2$ and $z_2 = (\neg x_1 \wedge \neg x_2)$
- Since now: $y = (z_1 \vee z_2)$

Representation

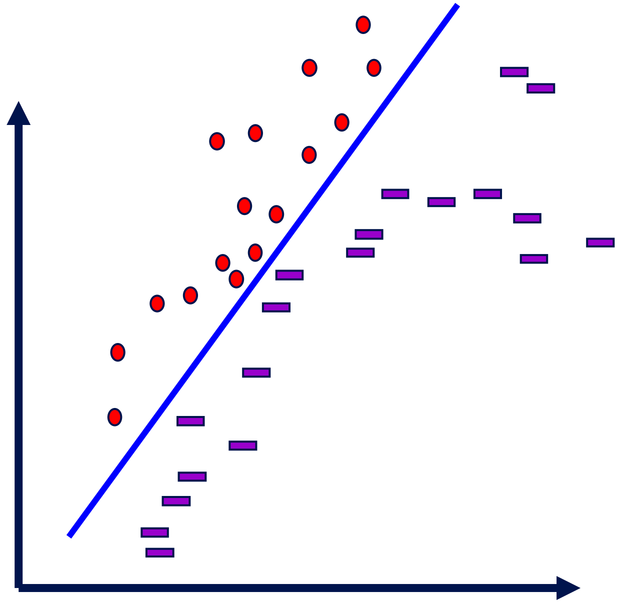
- The next few hidden slides provide a detailed explanation of feature generation in the discrete case.
- View
- We may discuss later

Key Issues in Machine Learning

- Modeling
  - How to formulate application problems as machine learning problems ? How to represent the data?
  - Learning Protocols (where is the data & labels coming from?)
- Representation
  - What functions should we learn (hypothesis spaces) ?
  - How to map raw input to an instance space?
  - Any rigorous way to find these? Any general approach?
- Algorithms
  - What are good algorithms?
  - How do we define success?
  - Generalization Vs. over fitting
  - The computational problem

Third Step: How to Learn?

- A possibility: Local search
  - Start with a linear threshold function.
  - ➡ See how well you are doing.
  - Correct
  - Repeat until you converge.
- There are other ways that do not search directly in the hypotheses space
  - Directly compute the hypothesis



How can we learn the blue linear separator given the red (positive) and purple (negative) examples? Note that the blue separator is represented as a vector of weights

Questions so far?

"is there any textbook you would recommend? "

"How to know which labels /features to use if there are 100s ? Does it change based on the question? "

"How did we know we needed neural networks for the badges example? "

"no "

"Nope "

"What is the difference between the function we learn and the learning algorithm we use? "

"none "

"no "

"no "

"Nope "

"nope "

"No "

"no "

"nope "

Comments/feedback on Today's Lecture?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Administration (9/16/20)

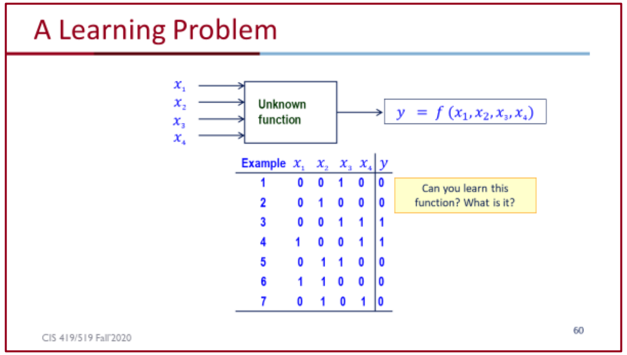
Are we recording? YES!

Available on the web site

- 1st quiz is out this week (Thursday night; due on Sunday night)
- HW 1 will be released next week.
- Questions?
  - Please ask/comment during class.
- Feedback/Suggestions?

Last Time

- The importance of a hypothesis space
  - The need to make some assumptions of the function we are trying to learn
- Expressivity of functions and feature spaces
- And Linear functions



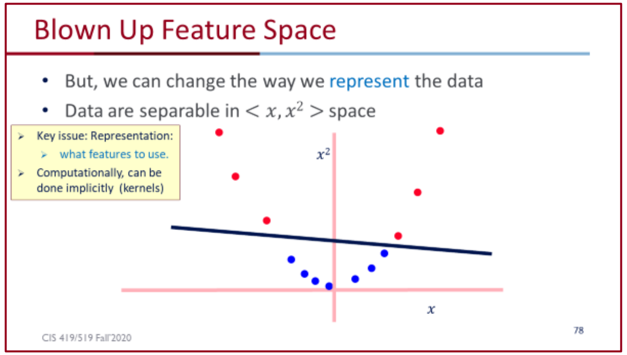
A Learning Problem

$$\begin{tikzcd} x_1 \arrow[r] & \boxed{\text{Unknown function}} \arrow[r] & \boxed{y = f(x_1,x_2,x_3,x_4)} \\ x_2 \arrow[ru] & & \\ x_3 \arrow[ruu] & & \\ x_4 \arrow[ruuu] & & \end{tikzcd}$$

Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function? What is it?

CIS 419/519 Fall'2020 60

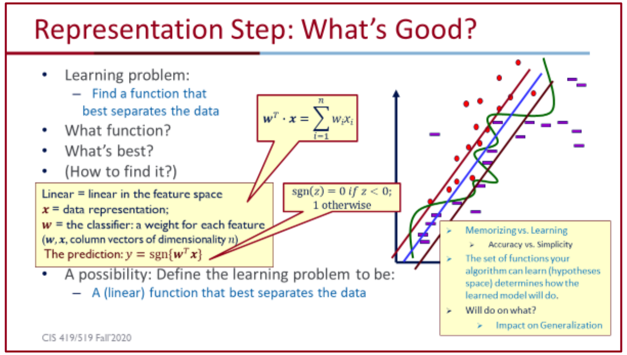


Blown Up Feature Space

- But, we can change the way we represent the data
- Data are separable in $< x, x^2 >$ space

> Key issue: Representation:
> > what features to use.
> Computationally, can be done implicitly (kernels)

CIS 419/519 Fall'2020 78



Representation Step: What's Good?

- Learning problem:
  - Find a function that best separates the data
- What function?
- What's best?
- (How to find it?)

$w^T \cdot x = \sum_{i=1}^{n} w_i x_i$

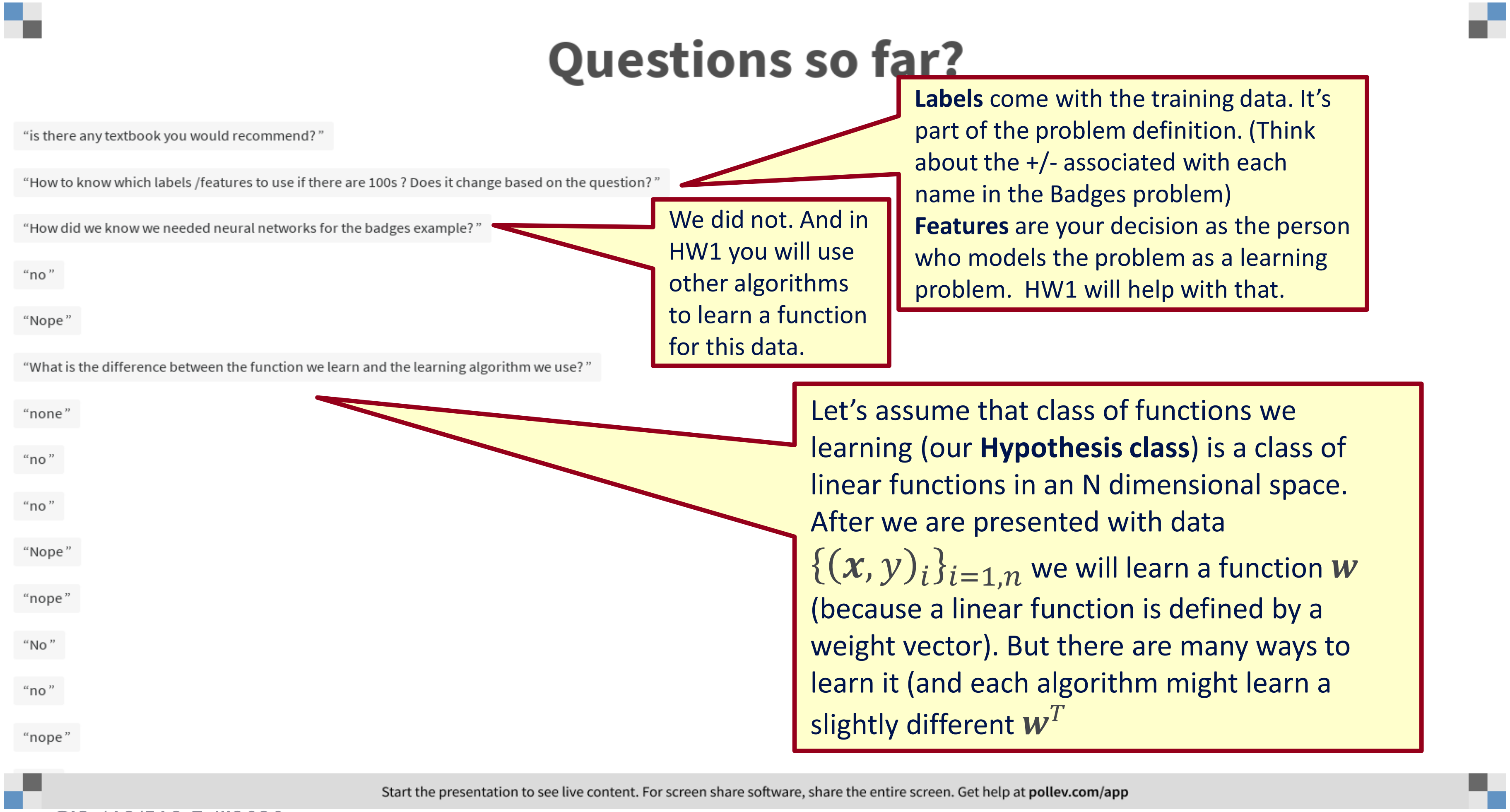
Linear = linear in the feature space
x = data representation;
w = the classifier: a weight for each feature
(w, x, column vectors of dimensionality n)
The prediction: $y = \text{sgn}\{w^T x\}$

$\text{sgn}(z) = 0 \; if \; z < 0$;
1 otherwise

- A possibility: Define the learning problem to be:
  - A (linear) function that best separates the data

> Memorizing vs. Learning
> > Accuracy vs. Simplicity
> The set of functions your algorithm can learn (hypotheses space) determines how the learned model will do.
> Will do on what?
> > Impact on Generalization

CIS 419/519 Fall'2020



Questions so far?

"is there any textbook you would recommend?"

"How to know which labels /features to use if there are 100s ? Does it change based on the question?"

"How did we know we needed neural networks for the badges example?"

"no"

"Nope"

"What is the difference between the function we learn and the learning algorithm we use?"

"none"

"no"

"no"

"Nope"

"nope"

"No"

"no"

"nope"

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Labels come with the training data. It's part of the problem definition. (Think about the +/- associated with each name in the Badges problem)
Features are your decision as the person who models the problem as a learning problem. HW1 will help with that.

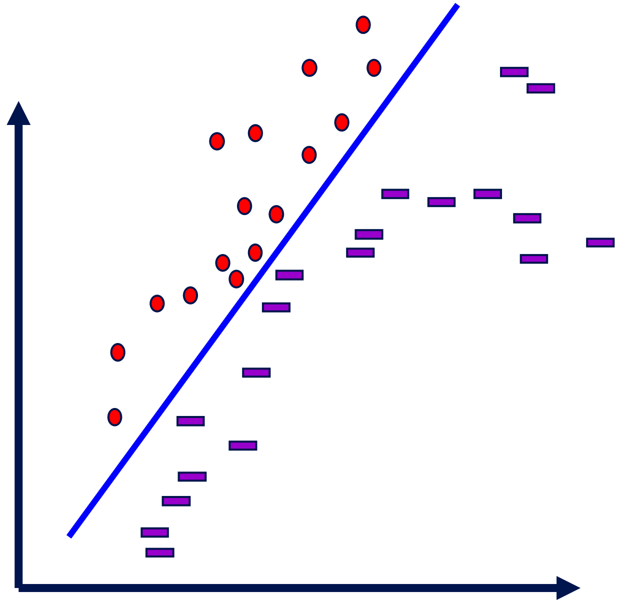
We did not. And in HW1 you will use other algorithms to learn a function for this data.

Let's assume that class of functions we learning (our **Hypothesis class**) is a class of linear functions in an N dimensional space. After we are presented with data $\{(\boldsymbol{x}, y)_i\}_{i=1,n}$ we will learn a function $\boldsymbol{w}$ (because a linear function is defined by a weight vector). But there are many ways to learn it (and each algorithm might learn a slightly different $\boldsymbol{w}^T$

Some Questions You Asked

- Any Text Book?
  - Not Really
  - Some options:
    - Machine Learning by Tom Mitchell
    - A Course in Machine Learning by Hal Daumé III
    - Pattern Recognition and Machine Learning by Bishop

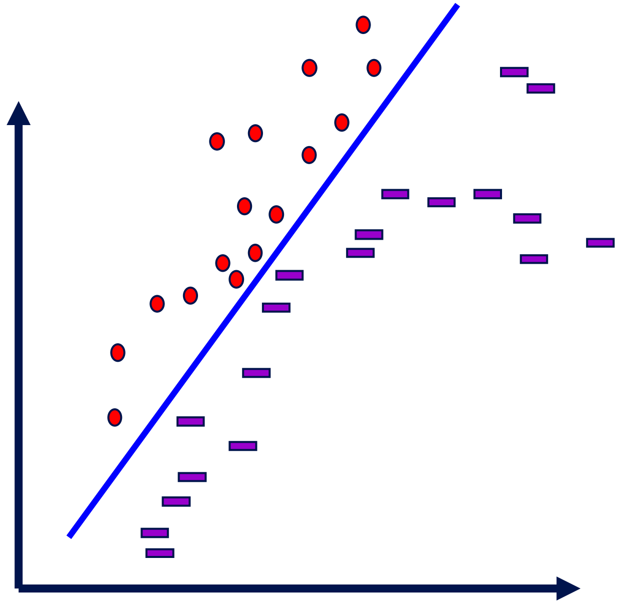
Third Step: How to Learn?



How can we learn the blue linear separator given the red (positive) and purple (negative) examples? Note that the blue separator is represented as a vector of weights

Third Step: How to Learn?

- A possibility: Local search
  - Start with a linear threshold function.
  - See how well you are doing.
  - Correct
  - Repeat until you converge.
- There are other ways that do not search directly in the hypotheses space
  - Directly compute the hypothesis



A General Framework for Learning

- Goal: predict an unobserved output value $y \in \boldsymbol{Y}$ based on an observed input vector $\boldsymbol{x} \in \boldsymbol{X}$
- Estimate a functional relationship $y \sim f(\boldsymbol{x})$ from a set $\{(\boldsymbol{x}, y)_i\}_{i=1,n}$
  - This is the blue separator we are after
- We will deal now with Classification: $y \in \{0,1\}$
  - (But, within the same framework can be used for $y \in \{1,2, \ldots, k\}$ or Regression, y ϵ $\boldsymbol{R}$)

Simple loss function: # of mistakes
[…] denotes an indicator function

- What do we want $f(\boldsymbol{x})$ to satisfy?
  - We want to minimize the **Risk:** $L\left(l(f(x), y)\right) = E_{X,Y}([f(\boldsymbol{x}) \neq y])$
  - Where: $E_{X,Y}$ denotes the expectation with respect to the true distribution over XxY.
  - In general, a loss function is a mapping l : YxY → R that measures how much the prediction of the current model is far from the desired prediction (the gold label)

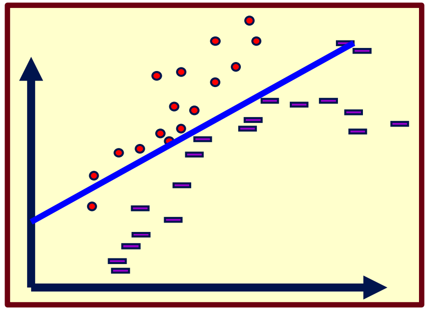
A General Framework for Learning (II)

- We want to minimize the Risk: $L\left(f(\,)\right) = E_{X,Y}(\,[f(X) \neq Y])$
- **Where:** $E_{X,Y}$ **denotes the expectation with respect to the true distribution**.

Side note: If the distribution over X ×Y is known, predict: $\boldsymbol{y = argmax_y\, P(y|x)}$
The best possible (optimal Bayes' error).

- We cannot minimize this loss (we don't know the distribution)
- Instead, we try to minimize the empirical classification error.
- For a set of training examples $\{(x_i, y_i)\}_{i=1,m}$
- Try to minimize: $L'(f()) = 1/m\, \Sigma_i\, [f(x_i) \neq y_i]$ (m=# of examples)
  - (Issue I: why/when is this good enough? Not now)
- This minimization problem is typically NP hard.
- To alleviate this computational problem, minimize a new function – a convex upper bound of the (real) **classification error function:**

$$I(f(\boldsymbol{x}), y) = [f(\boldsymbol{x}) \neq y] = \{1\ when\ f(\boldsymbol{x}) \neq y;\ 0\ otherwise\}$$



Algorithmic View of Learning: an Optimization Problem

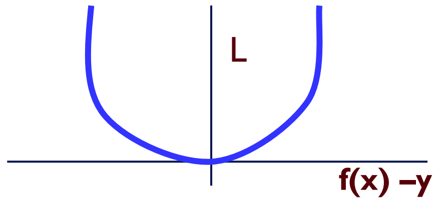
- A Loss Function $L(f(\boldsymbol{x}), y)$ measure how far is the prediction f(x) from the desired y;
  - the penalty incurred by a classifier f on example $(\boldsymbol{x}, y)$.
- There are many different loss functions one could define:
  - Misclassification Error: (0-1 loss)

  $L(f(\boldsymbol{x}), y) = 0$ if $f(\boldsymbol{x}) = y$; 1 otherwise
  - Squared Loss:

  $L(f(\boldsymbol{x}), y) = (f(\boldsymbol{x}) - y)^2$
  - Input dependent loss:

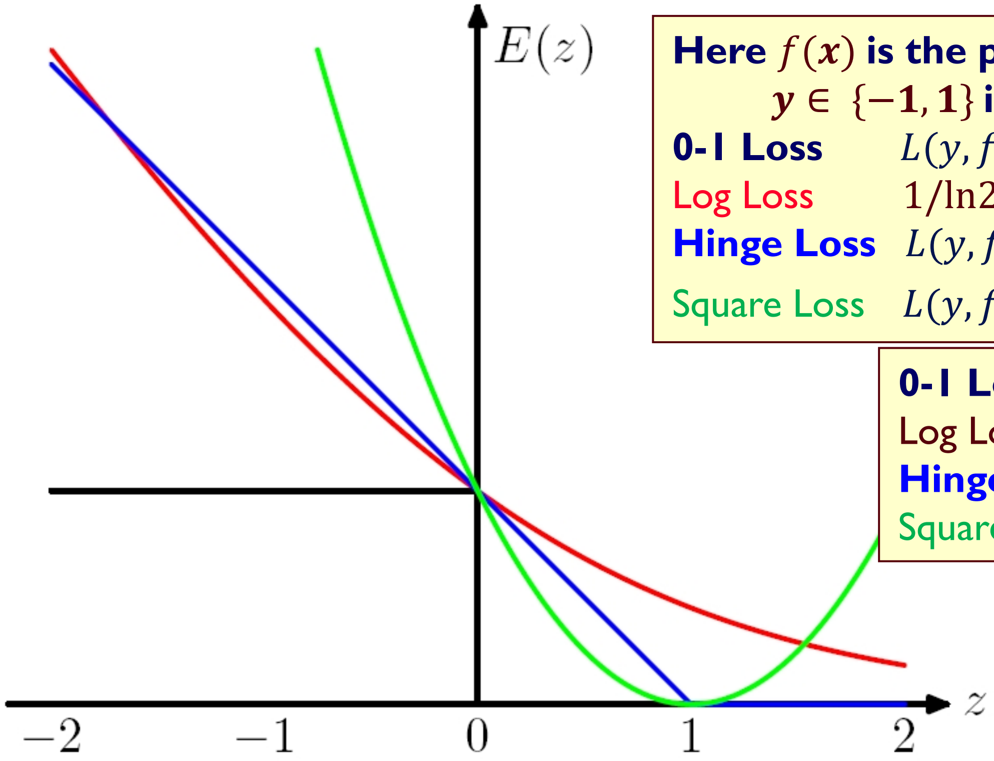
  $L(f(\boldsymbol{x}), y) = 0$ if $f(\boldsymbol{x}) = y$; $c(\boldsymbol{x})$otherwise.

A continuous convex loss function allows a simpler optimization algorithm.



Loss

$\mathrm{sgn}(z) = -1 \; if \; z < 0$
$\quad +1$ otherwise



Here $f(x)$ is the prediction $\in R$
$y \in \{-1, 1\}$ is the correct value

0-1 Loss	$L(y, f(x)) = ½\,(1 - \mathrm{sgn}(y f(\boldsymbol{x})))$
Log Loss	$1/\ln 2 \,\log(1 + \exp\{-y f(x)\})$
Hinge Loss	$L(y, f(x)) = \max(0, 1 - y\, f(\boldsymbol{x}))$
Square Loss	$L(y, f(x)) = (y - f(\boldsymbol{x}))^2$

0-1 Loss :	z axis = $y f(\boldsymbol{x})$
Log Loss:	z axis = $y f(\boldsymbol{x})$
Hinge Loss:	z axis = $y f(\boldsymbol{x})$
Square Loss:	z axis = $(y - f(\boldsymbol{x}) + 1)$

Example

Putting it all together:

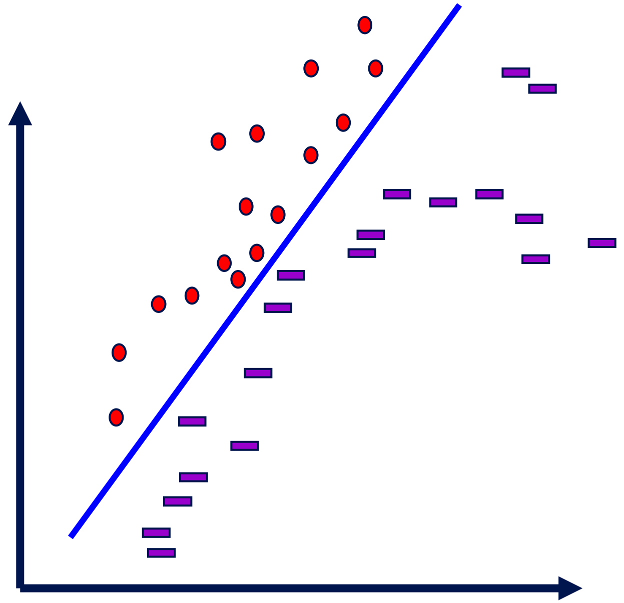
A Learning Algorithm

Third Step: How to Learn?

- A possibility: Local search
  - Start with a linear threshold function.
  - ➡ See how well you are doing.
  - Correct
  - Repeat until you converge.
- There are other ways that

do not search directly in the hypotheses space

  - Directly compute the hypothesis

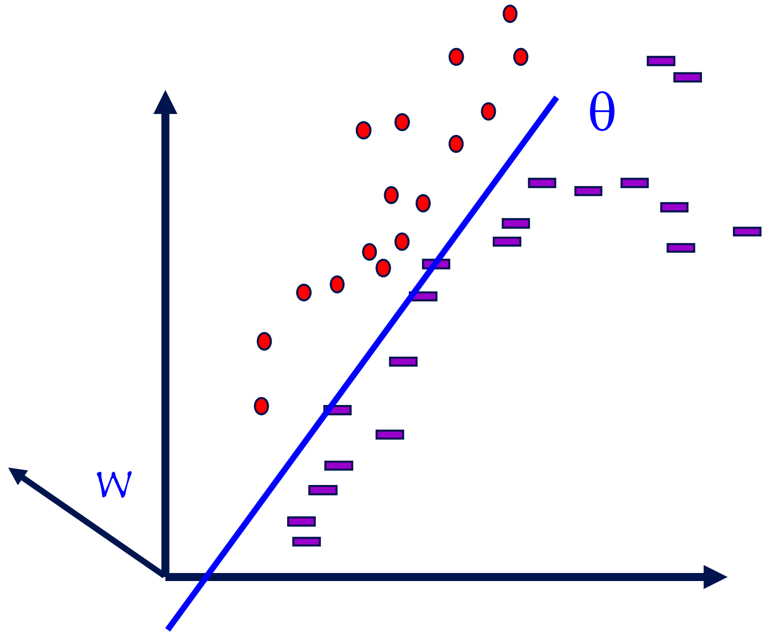


Learning Linear Separators

(LTU=Linear Threshold Unit; a single layer neural network)

$$f(\boldsymbol{x}) = \operatorname{sgn}\{\boldsymbol{w}^T \cdot \boldsymbol{x} - \theta\} = \operatorname{sgn}\{\textstyle\sum_{i=1}^{n} w_i x_i - \theta\}$$

- $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ is the feature based encoding of the data point
- $\boldsymbol{w}^T = (w_1, w_2, \ldots, w_n) \in \boldsymbol{R}^n$ is the target function.
- θ determines the shift with respect to the origin



Canonical Representation

$$f(\boldsymbol{x}) = sgn\{\boldsymbol{w}^T \cdot \boldsymbol{x} - \theta\} = sgn\{\sum_{i=1}^{n} w_i x_i - \theta\}$$

- **Note:** $sgn\ \{\boldsymbol{w}^T \cdot \boldsymbol{x} - \theta\} = sgn\ \{\boldsymbol{w}'^T \cdot \boldsymbol{x}'\}$
- Where:
  - $\boldsymbol{x}' = (\boldsymbol{x}, -1)$ and $\boldsymbol{w}' = (\boldsymbol{w}, \theta)$
- Moved from an n dimensional representation to an $(n + 1)$ dimensional representation, but now can look for hyperplanes that go through the origin.
- Basically, that means that we learn both w and θ

General Learning Principle

- Our goal is to find a $\boldsymbol{w}$ that *minimizes* the expected risk

  $\boldsymbol{E(w)} = \boldsymbol{E_{X,Y}\, Q(x, y, w)}$
- We cannot do it.
- **Instead,** we approximate $E(\boldsymbol{w})$ using a finite training set of independent samples (x_i, y_i)

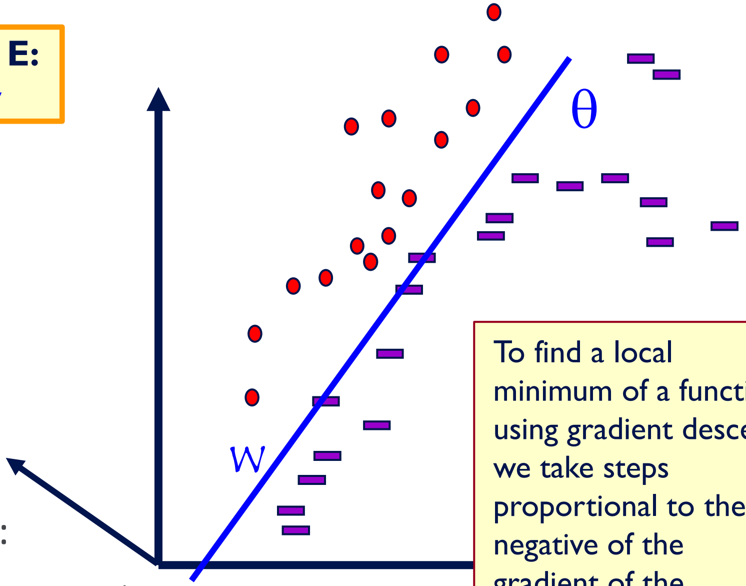
  $\boldsymbol{E(w)} \sim = \sim 1/m \sum_{\boldsymbol{1,m}} Q(x_i, y_i, \boldsymbol{w})$
- To find the *minimum*, we use a **batch gradient descent** algorithm
- That is, we successively compute estimates $\boldsymbol{w}^t$ of the optimal parameter vector $\boldsymbol{w}$:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \nabla E(\boldsymbol{w}) = \boldsymbol{w}^t - \frac{1}{m}\sum_{1,m} \nabla Q(x_i, y_i, \boldsymbol{w})$$

The loss Q: a function of x, w and y

The Risk (Err) E: a function of w

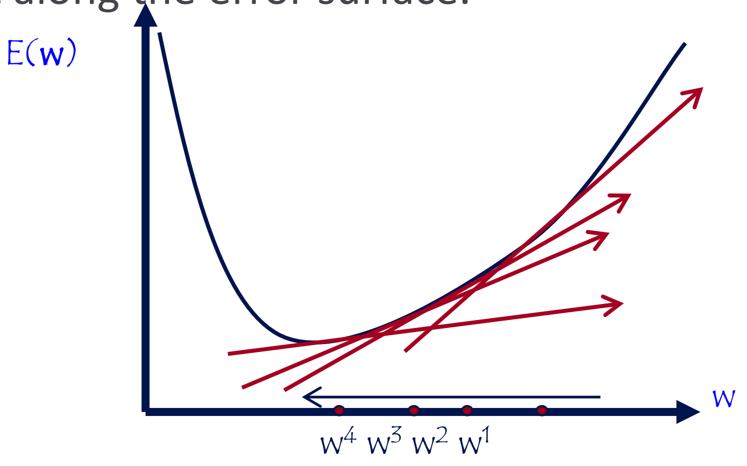
t here is "time" or "iteration" #



To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient of the function at the current point.

Gradient Descent

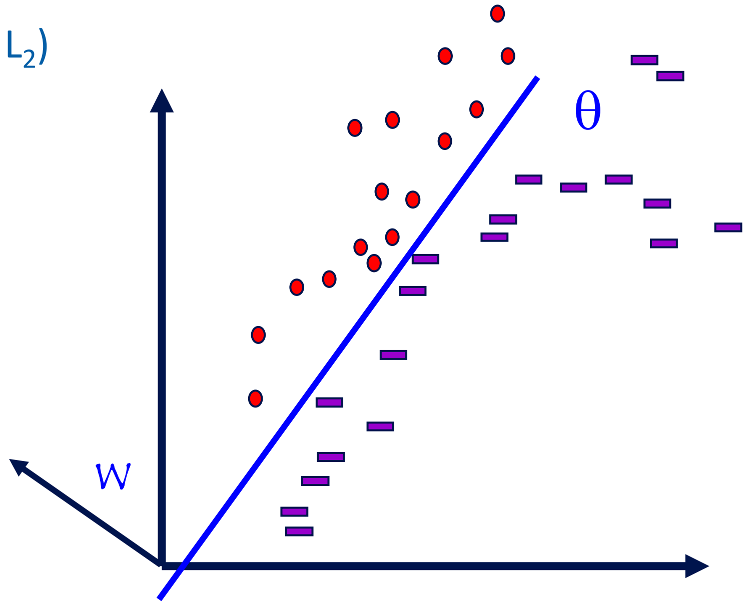
- We use gradient descent to determine the weight vector that minimizes $E(\boldsymbol{w})\ (= \ Err\ (\boldsymbol{w}))$;
- Fixing the set D of examples, E=Err is a function of $\boldsymbol{w}$
- At each step, the weight vector is modified in the direction that produces the steepest descent along the error surface.



To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient of the function at the current point.

LMS: An Optimization Algorithm

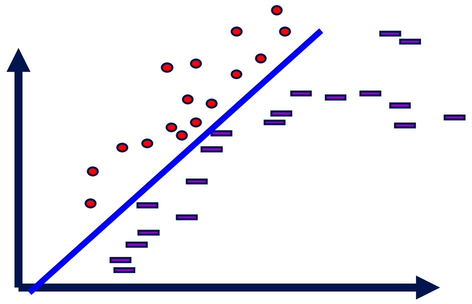
- Our Hypothesis Space is the collection of **Linear Threshold Units**
- Loss function:
  - Squared loss: LMS (Least Mean Square, L_2)
  - $Q(\boldsymbol{x}, y, \boldsymbol{w}) = ½ (\boldsymbol{w}^T \boldsymbol{x} - y)^2$



LMS: An Optimization Algorithm

- (i (subscript) – vector component; j (superscript) - time; d – example #)

Assumption: $\mathbf{x} \in R^n$; $\mathbf{u} \in R^n$ is the target weight vector; the target (label) is $t_d = \mathbf{u}^T \cdot \mathbf{x}$ Noise has been added; so, possibly, no weight vector is consistent with the data.



- Let $\boldsymbol{w^j}$ be the current weight vector we have
- Our prediction on the d-th example $\boldsymbol{x}$ is:

$$O_d = \mathbf{w}^{(j)T} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i^{(j)} \quad x_i$$

- Let t_d be the target value for this example
- The error the current hypothesis makes on the data set is:

$$E(\boldsymbol{w}) = \mathrm{Err}(\mathbf{w}^j) = \frac{1}{2}\sum_{d \in D}(t_d - O_d)^2$$

Gradient Descent

- To find the best direction in the weight space $\boldsymbol{w}$ we compute the gradient of E with respect to each of the components of

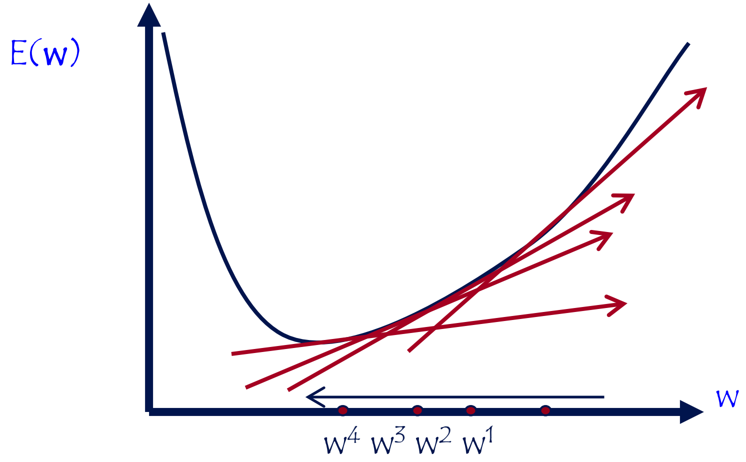
$$\nabla E(\boldsymbol{w}) = [\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots \frac{\partial E}{\partial w_n}]$$

- This vector specifies the direction that produces the steepest increase in E;
- We want to modify $\boldsymbol{w}$ in the direction of $-\nabla E(\boldsymbol{w})$

- Where (with a fixed step size [learning rate] R):

$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} + \Delta \boldsymbol{w}$$

$$\Delta \mathbf{w} = -\mathrm{R}\, \nabla E(\boldsymbol{w})$$



Gradient Descent: LMS

- We have: $E(\boldsymbol{w}) = \mathrm{Err}(\mathbf{w}^j) = \frac{1}{2}\sum_{\mathrm{d}\in D}(t_d - O_d)^2$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 =$$

- Therefore:

$$=\frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2 =$$

$$=\frac{1}{2}\sum_{d\in D} 2(t_d - o_d) \quad \frac{\partial}{\partial w_i}(t_d - \boldsymbol{w}_d \cdot \boldsymbol{x}_d) =$$

$$=\sum_{d\in D}(t_d - o_d)(-x_{id})$$

Recall:

$$\boxed{O_d = \mathbf{w}^{(j)T} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i^{(j)} \quad x_i}$$

Alg1: Gradient Descent: LMS

- Weight update rule:

$$\Delta w_i = R \sum_{d \in D} (t_d - O_d) x_{id}$$

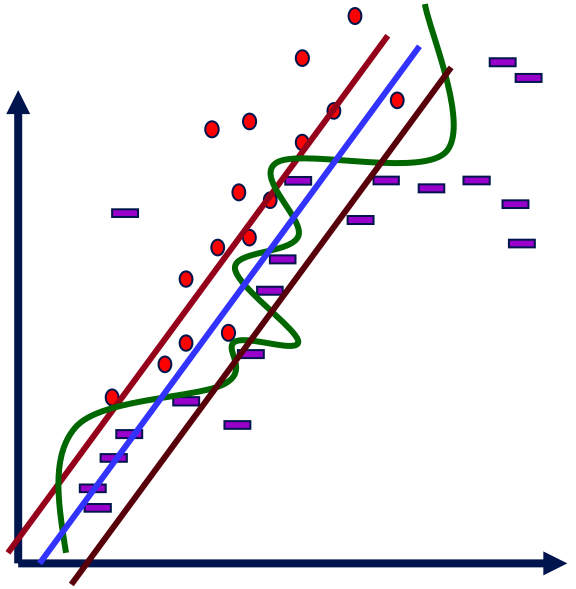
- Gradient descent algorithm for training linear units:
  - Start with an initial random weight vector
  - For every example d with target value t_d do:
    - Evaluate the linear unit $O_d = \mathbf{w}^{(j)T} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i^{(j)} \ x_i$
  - Update w by adding Δw_i to each component
  - Continue until E below some threshold

This algorithm always converges to a local minimum of $E(\boldsymbol{w})$, for small enough steps. Here (LMS for linear regression), the surface contains only a single global minimum, so the algorithm converges to a weight vector with minimum error, regardless of whether the examples are linearly separable.

The surface may have local minimum if the loss function is different.

What Have We Done So Far?

- We wanted to separate Red point from Purple points
  - With the goal of minimizing the number of mistakes
- It's hard to do, so we decided to use a surrogate loss function (LMS) and use it to guide our search for a good linear separator
- The algorithm is guaranteed to minimize the LMS loss
  - But, is it guaranteed to minimize the number of mistakes (0-1 loss) ?



Given a training set, we run Alg1. What will happen to the (a) LMS loss and (b) to the number of mistakes as a function of the number of iterations?

Both will go down to 0

Both will go down to some level above 0

The LMS will go down and converge; we don't know about the number of mistakes

The number of mistake will go down; we don't know about the LMS

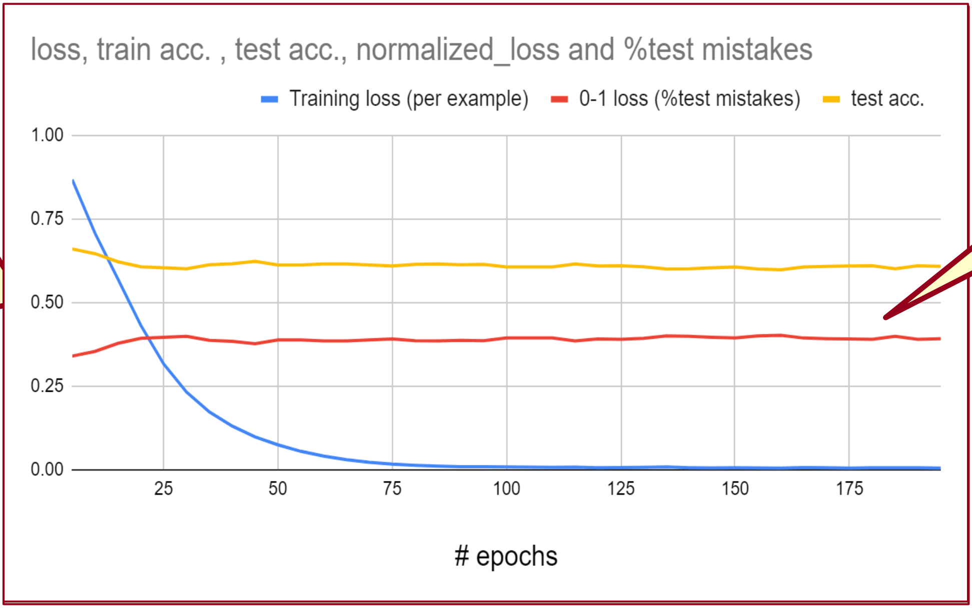
None of the above

LMS Loss and 0-1 Loss

- The data used here is a re-labeling of the Badges data
  - (re-labaling is done since the original function was too easy)

LMS Loss (per example)
We knew that the loss will go down and converge.

The fact that it converges to 0 is just since we used an expressive model (not a single layer neural network, a three layer one)



The 0-1 loss did not go to 0, so we are still making mistakes on this dataset.

Alg1: Gradient Descent: LMS

- Weight update rule:

$$\Delta w_i = R \sum_{d \in D} (t_d - O_d) x_{id}$$

- Gradient descent algorithm for training linear units:
  - Start with an initial random weight vector
  - For every example d with target value t_d do:
    - Evaluate the linear unit $O_d = \mathbf{w}^{(j)T} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i^{(j)} \ x_i$
  - Update w by adding Δw_i to each component
  - Continue until E below some threshold

This algorithm always converges to a local minimum of $E(\boldsymbol{w})$, for small enough steps. Here (LMS for linear regression), the surface contains only a single global minimum, so the algorithm converges to a weight vector with minimum error, regardless of whether the examples are linearly separable.

The surface may have local minimum if the loss function is different.

Alg 2: Incremental (Stochastic) Gradient Descent: (LMS)

- Weight update rule:

$$\Delta w_i = R(t_d - O_d)x_{id}$$

Dropped the averaging operation.
Instead of averaging the gradient of the loss over the complete training set, choose at random a sample (x,y) (or a subset of examples) and update $\mathbf{w}^t$

- Gradient descent algorithm for training linear units:
  - Start with an initial random weight vector
  - For every example d with target value t_d do:
    - » Evaluate the linear unit $O_d = \boldsymbol{w}^{(j)T} \cdot \boldsymbol{x} = \sum_{i=1}^{n} w_i^{(j)} \quad x_i$
    - » update $\boldsymbol{w}$ by incrementally by adding Δw_i to each component (update without summing over all data)
  - Continue until E below some threshold
- In general - does not converge to global minimum
- But, on-line algorithms are sometimes advantageous…
  - Typically, not used as a complete on-line algorithm, but rather **run in small batches.**
- Decreasing R with time guarantees convergence

Learning Rates and Convergence

- In the general (non-separable) case the learning rate R must decrease to zero to guarantee convergence.
- The learning rate is called the step size. There are more sophisticated algorithms that choose the step size automatically and converge faster.
  - We will see versions of this algorithm as we go
- Choosing a better starting point also has impact.

- The gradient descent and its stochastic version are very simple algorithms
- But, almost all the algorithms we will learn in the class
  - In particular, all the neural networks learning algorithms
- Are gradient descent algorithms (with some bells and whistles) for different loss functions and different hypotheses spaces.

Computational Issues

- Assume the data is linearly separable.
- Sample complexity:
  - Suppose we want to ensure that our LTU has an error rate (on new examples) of less than ϵ with high probability (at least $(1 - \delta)$)
  - How large does m (the number of examples) must be in order to achieve this? It can be shown that for n dimensional problems:

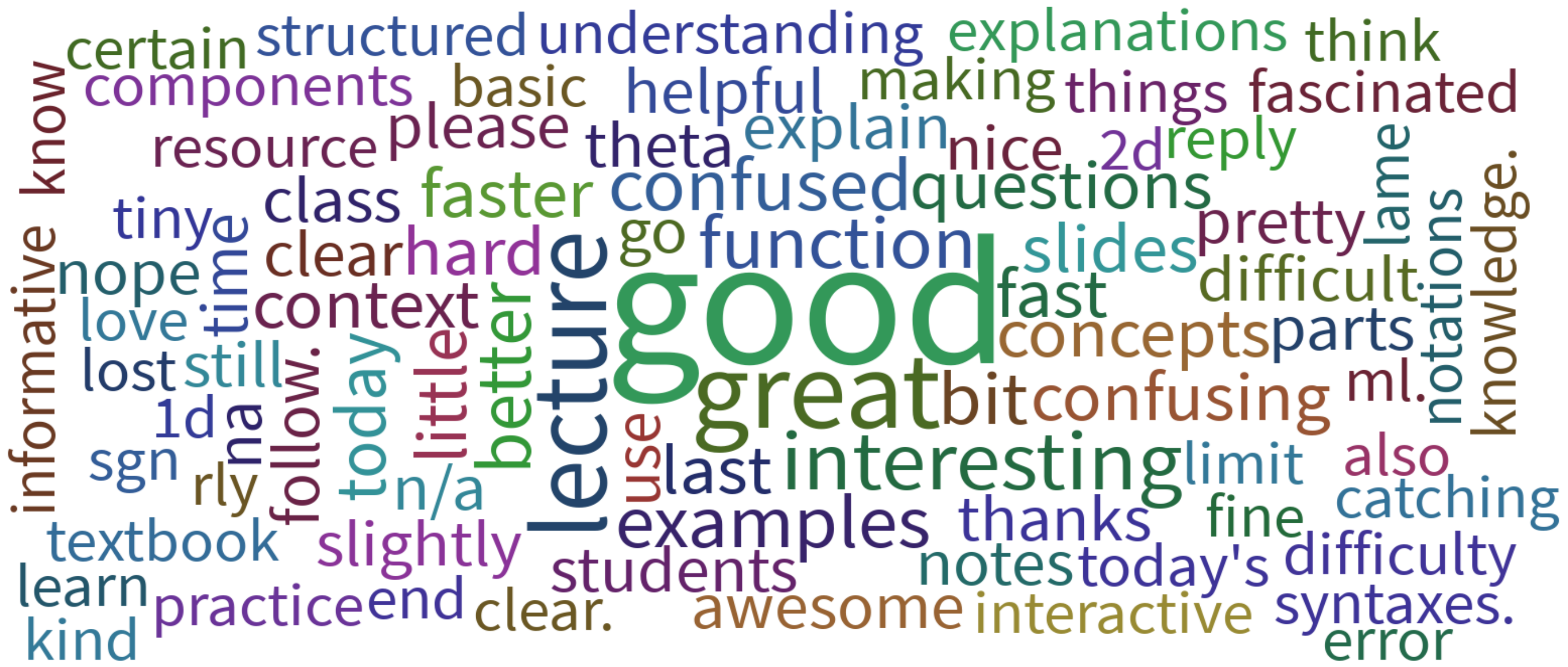
$$m = O(\frac{1}{\epsilon} [\ln\left(\frac{1}{\delta}\right) + (n+1)\ln\left(\frac{1}{\epsilon}\right)])$$

- Computational : What can be said?
  - It can be shown that there exists a polynomial time algorithm for finding consistent LTU (by reduction from linear programming).
  - [Contrast with the NP hardness for 0-1 loss optimization]
  - (On-line algorithms have inverse quadratic dependence on the margin)

Other Methods for LTUs

- Fisher Linear Discriminant:
  - A direct computation method
- Probabilistic methods (naïve Bayes):
  - Produces a stochastic classifier that can be viewed as a linear threshold unit.
- Winnow/Perceptron
  - A multiplicative/additive update algorithm with some sparsity properties in the function space (a large number of irrelevant attributes) or features space (sparse examples)
- Logistic Regression, SVM…many other algorithms

Comments/feedback on Today's Lecture?



Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

How will you rate today's lecture?

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**