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), Eric Eaton for CIS519/419 at Penn, or from other authors who have made their ML slides available.

Course Overview

- 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

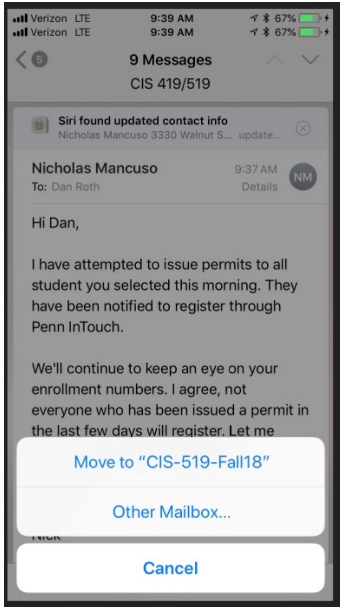
- Monday, Wednesday: 10:30pm-12:00pm 101 Levine
- Office hours: Mon/Tue 5-6 pm [my office]
- 10 TAs
- Assignments: 5 Problems set (Python Programming)
  - Weekly (light) on-line quizzes
- Weekly Discussion Sessions
- Mid Term Exam
- [Project] (look at the schedule)
- Final
- No real textbook:
  - Slides/Mitchell/Flach/Other Books/ Lecture notes /Literature

HW0 is mandatory

Go to the web site

Be on Piazza

Registration for Class



Nicholas Mancuso 9:37 AM

To: Dan Roth

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

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

Ask **during** class, not **after** class

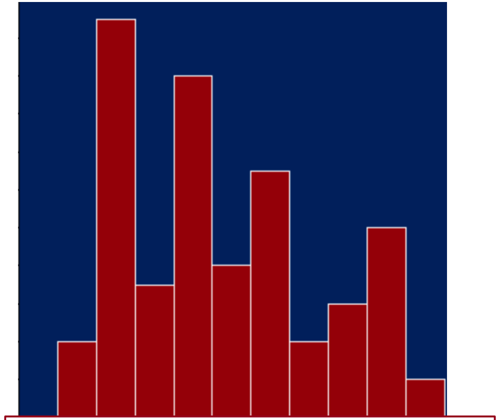
- **Applied Machine Learning**
- Applied: mostly in HW
- **Machine learning:** mostly in class, quizzes, exams

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%

CIS 519 on the web

- Check our class website:
  - Schedule, slides, videos, policies
    - http://www.seas.upenn.edu/~cis519/fall2019/
  - Sign up, participate in our Piazza forum:
    - Announcements and discussions
    - http://piazza.com/upenn/fall2019/cis419519
  - Check out our team
    - Office hours
    - [Optional] Discussion Sessions

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

CIS 519 Admin

Registration for Class

- Check our class website:
  - Schedule, slides, videos, policies
    - http://www.seas.upenn.edu/~cis519/fall2018/
  - Sign up, participate in our Piazza forum:
    - Announcements and discussions
    - http://piazza.com/upenn/fall2018/cis419519
  - Check out our team
    - Office hours

We start today

- Canvas:
  - Notes, homework and videos will be open.

HW0 is mandatory!

- [Optional] Discussion Sessions:
  - Starting this week: Wednesday 4pm, Thursday 5pm: Python Tutorial
  - Check the website for the location

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?

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

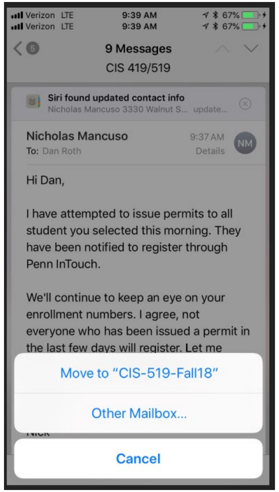
Labeled test data

? 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

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?
    - Prediction or Modeling?
  - 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?



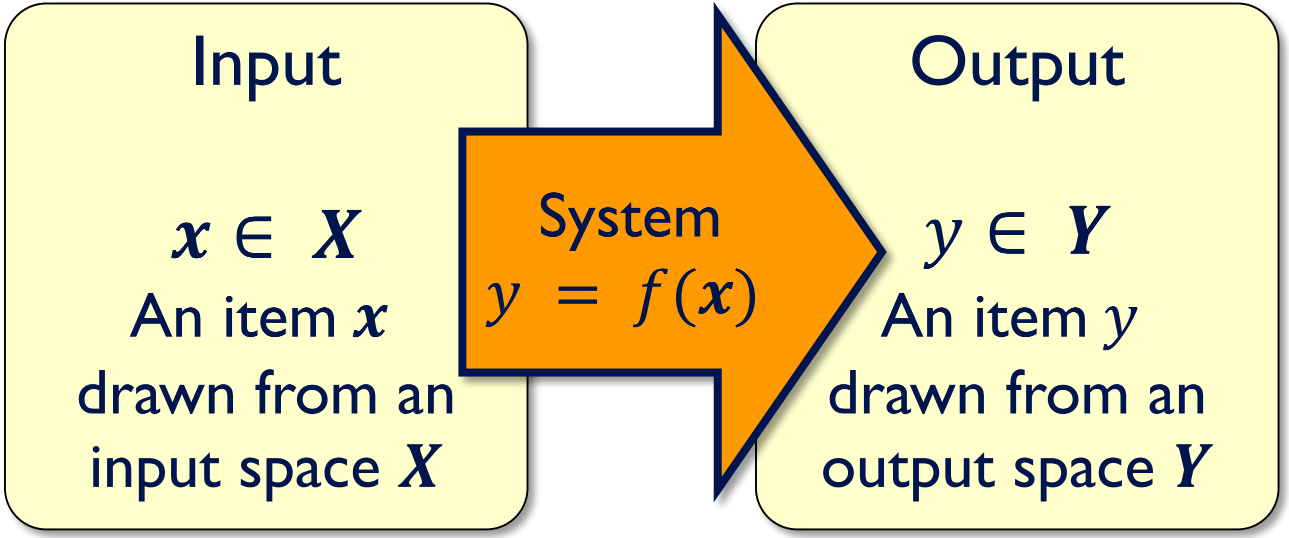
Nicholas Mancuso 9:37 AM
To: Dan Roth Details

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

Supervised Learning



Input

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

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

System

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

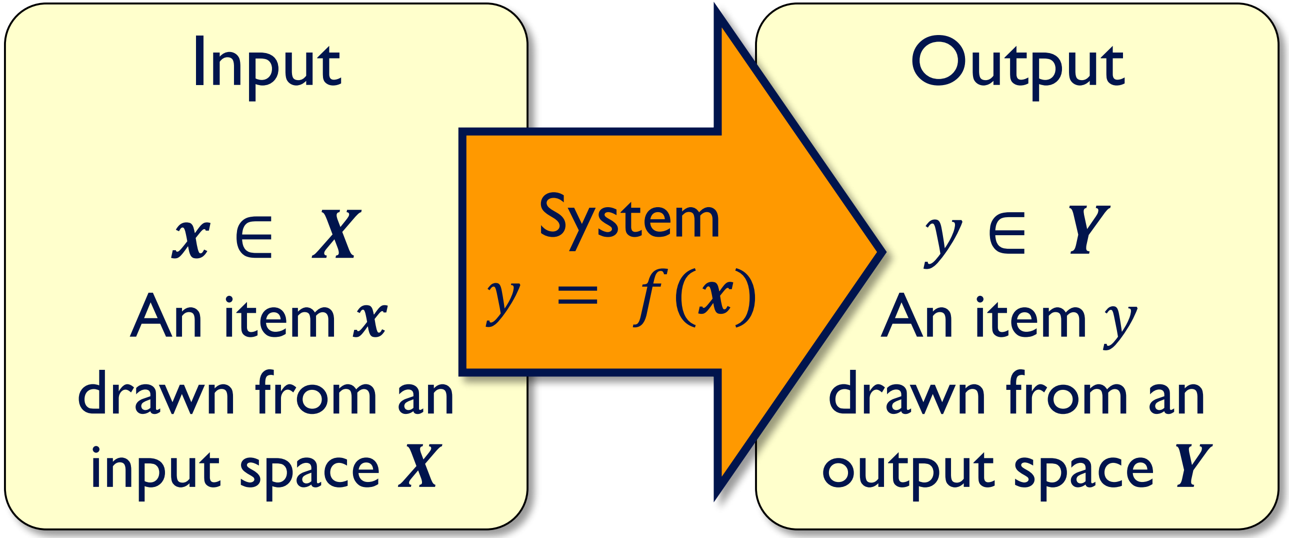
Output

$y \in \boldsymbol{Y}$

An item y drawn from an output space $\boldsymbol{Y}$

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

Supervised Learning



Input

$x \in X$

An item x drawn from an input space X

System

$y = f(x)$

Output

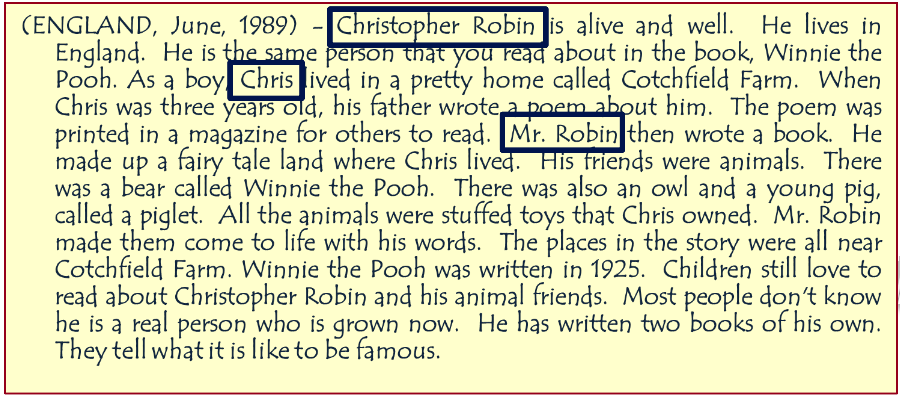
$y \in Y$

An item y drawn from an output space Y

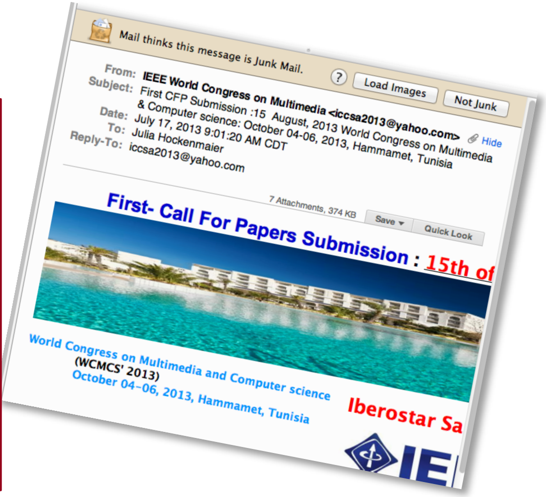
In (supervised) machine learning, we deal with systems whose $f(x)$ is learned from examples.

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

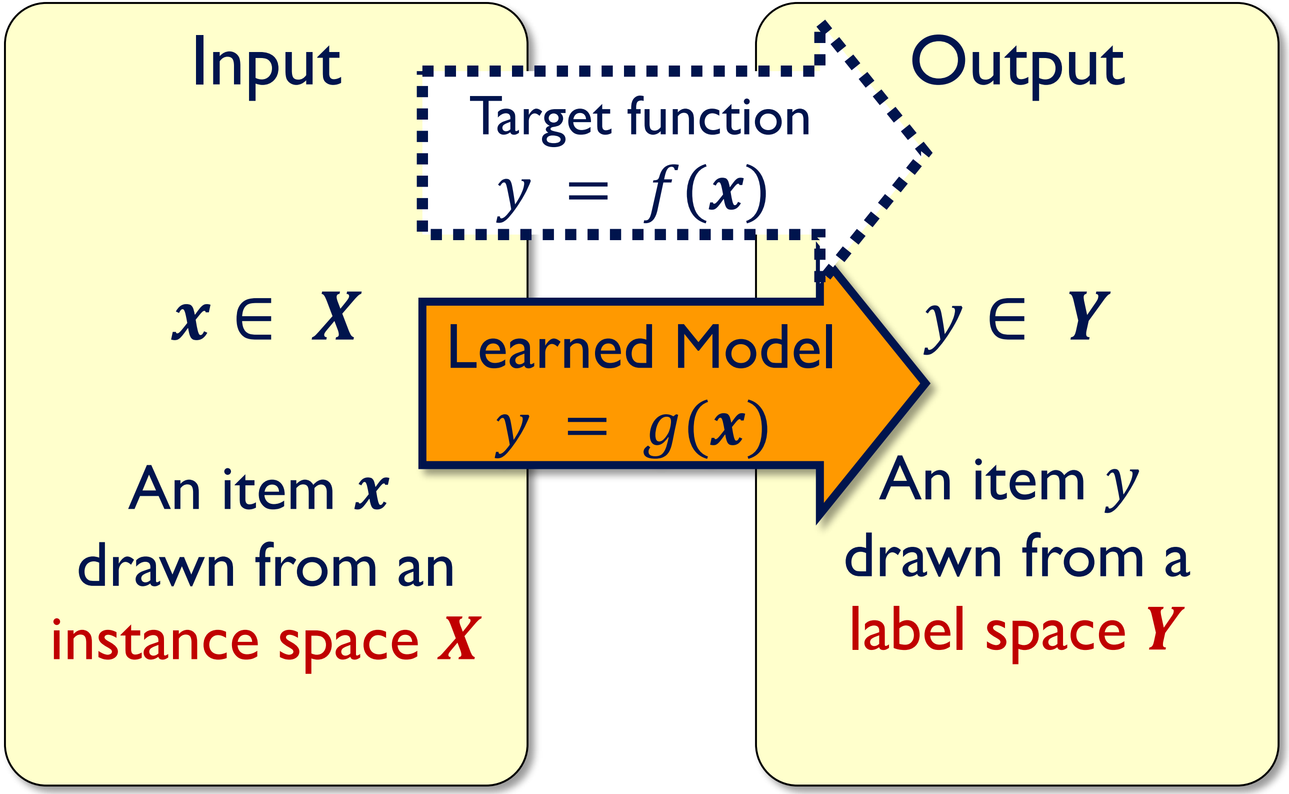
7 Attachments, 374 KB

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

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

Output

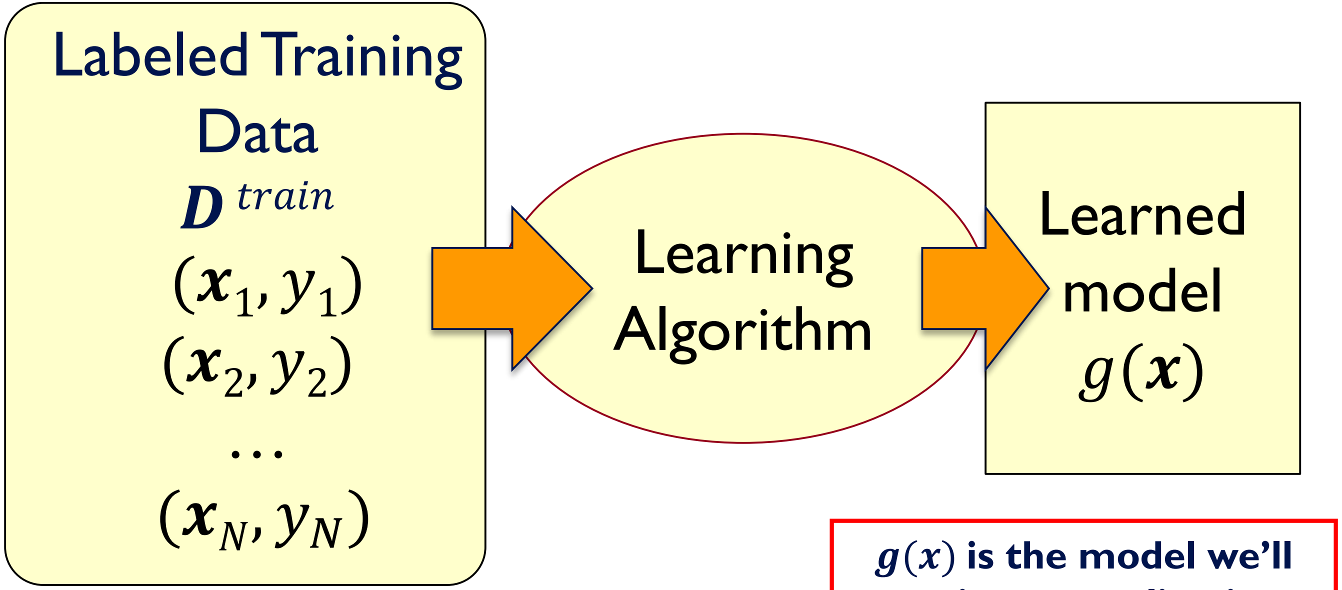
$y \in \boldsymbol{Y}$

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 g($\boldsymbol{x}$)

Can you suggest other learning protocols?


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
$g(\boldsymbol{x})$

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

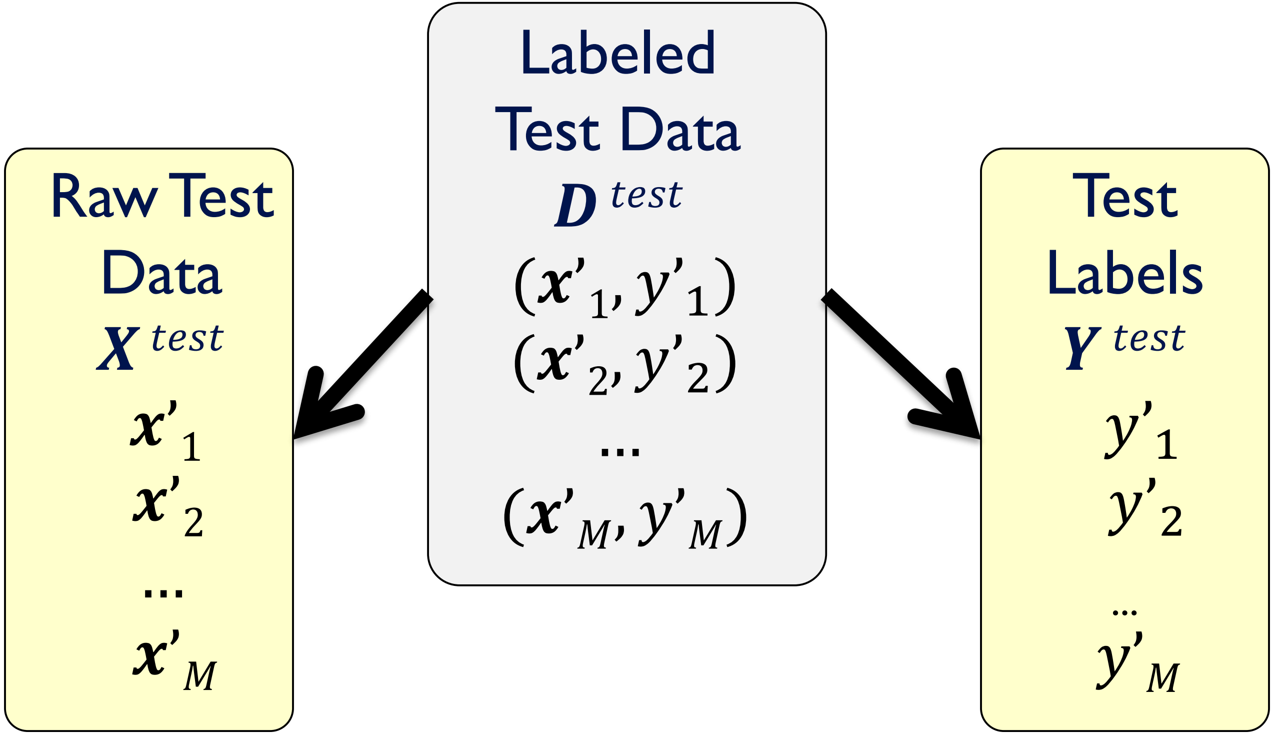
Supervised learning: Testing

- 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



Labeled Test Data

$\boldsymbol{D}^{test}$

$(\boldsymbol{x}'_1, y'_1)$

$(\boldsymbol{x}'_2, y'_2)$

…

$(\boldsymbol{x}'_M, y'_M)$

Raw Test Data

$\boldsymbol{X}^{test}$

$\boldsymbol{x}'_1$

$\boldsymbol{x}'_2$

…

$\boldsymbol{x}'_M$

Test Labels

$\boldsymbol{Y}^{test}$

y'_1

y'_2

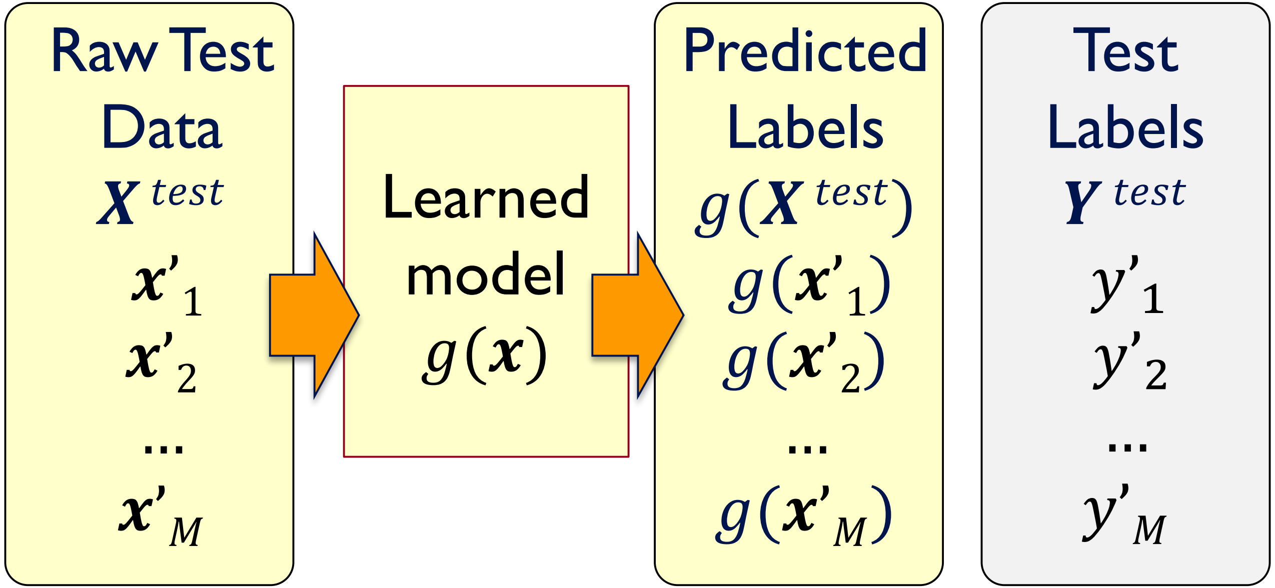
…

y'_M

Supervised learning: Testing

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

Can you use the test data otherwise?



Raw Test Data

$\boldsymbol{X}^{test}$

$\boldsymbol{x}'_1$

$\boldsymbol{x}'_2$

…

$\boldsymbol{x}'_M$

Learned model

$g(\boldsymbol{x})$

Predicted Labels

$g(\boldsymbol{X}^{test})$

$g(\boldsymbol{x}'_1)$

$g(\boldsymbol{x}'_2)$

…

$g(\boldsymbol{x}'_M)$

Test Labels

$\boldsymbol{Y}^{test}$

y'_1

y'_2

…

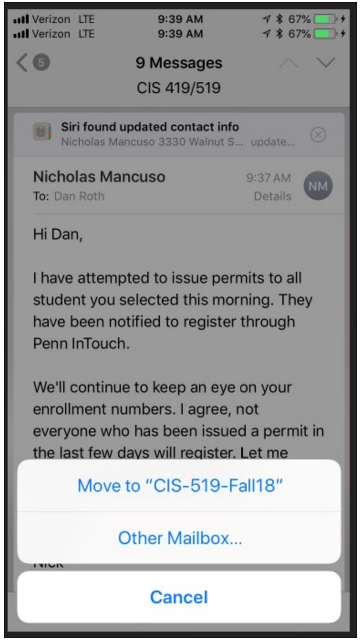
y'_M

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

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?



Nicholas Mancuso 9:37 AM
To: Dan Roth Details

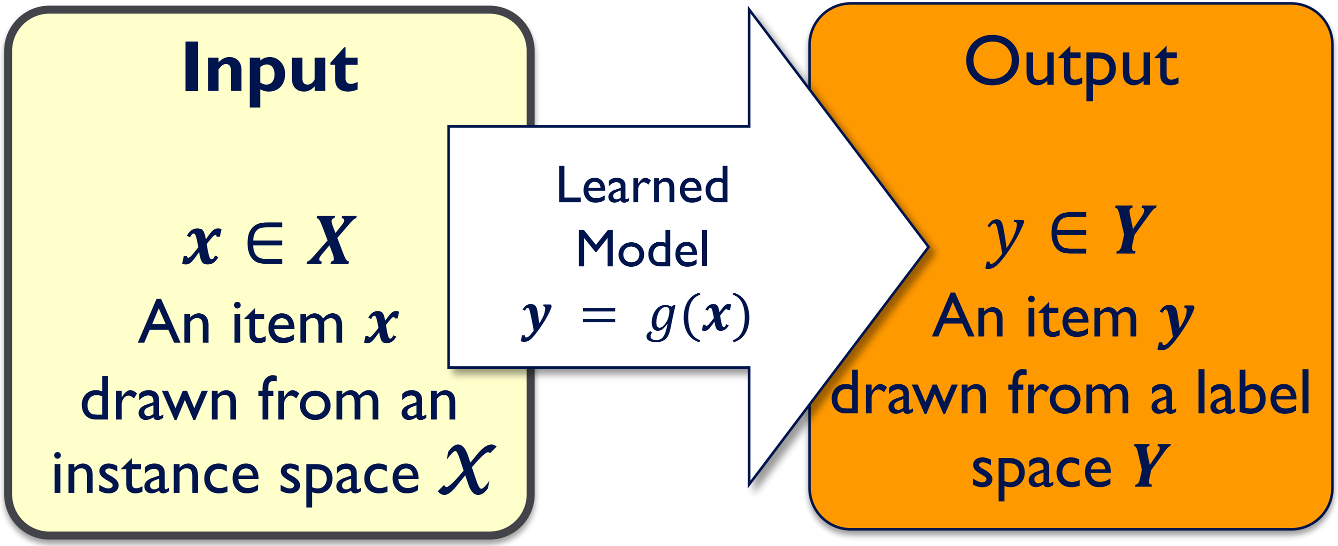
Hi Dan,

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

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.

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 contains the word 'class'?
    - » Does this email contains the word 'waiting'?
    - » Does this email contains the word 'class' and the word 'waiting'?
  - Numerical features:
    - » How often does 'learning' occur in this email?
    - » What long is 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

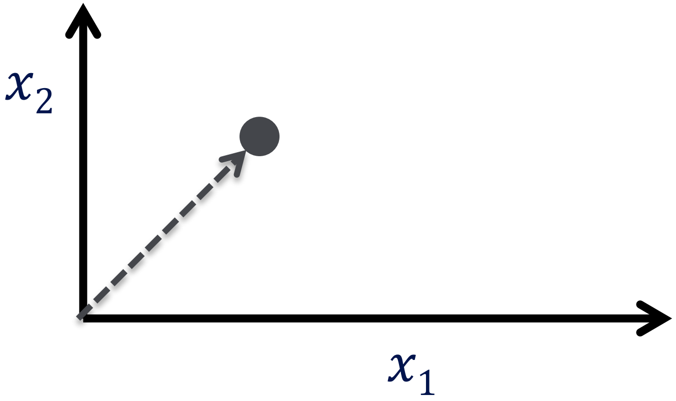
Does it add anything?

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

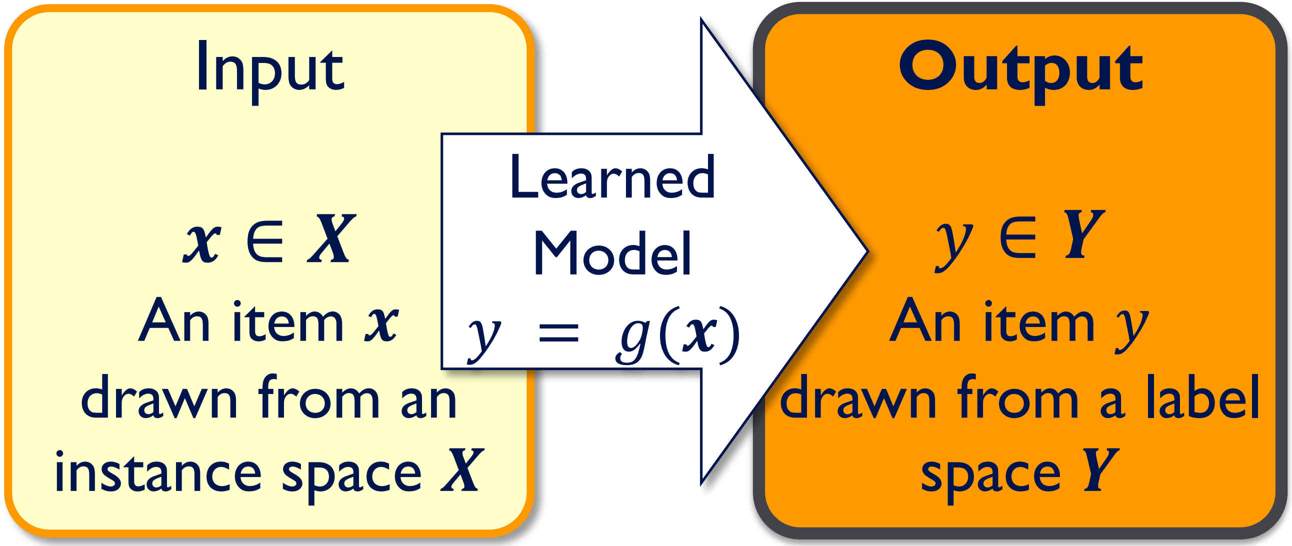
- $\boldsymbol{X}$ is an N-dimensional vector space (e.g. $\mathbb{R}^N$)
  - Each dimension = one feature.
- Each $\boldsymbol{x}$ is a feature vector (hence the boldface $\boldsymbol{x}$).
- Think of $\boldsymbol{x}$ = $[x_1 \dots x_N]$ as a point in $\boldsymbol{X}$:



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?
- We can’t teach you what specific features to use for your task.
  - But we will touch on some general principles

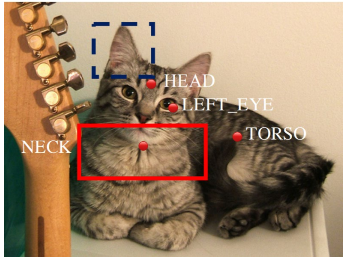
2. The label space $\boldsymbol{Y}$

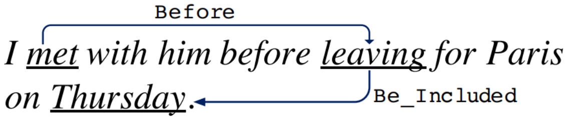

Input
$\boldsymbol{x} \in \boldsymbol{X}$
An item $\boldsymbol{x}$ drawn from an instance space $\boldsymbol{X}$
Learned Model
$y = g(\boldsymbol{x})$
Output
$y \in \boldsymbol{Y}$
An item y drawn from a label space $\boldsymbol{Y}$

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, etc.)
  - Structure learning

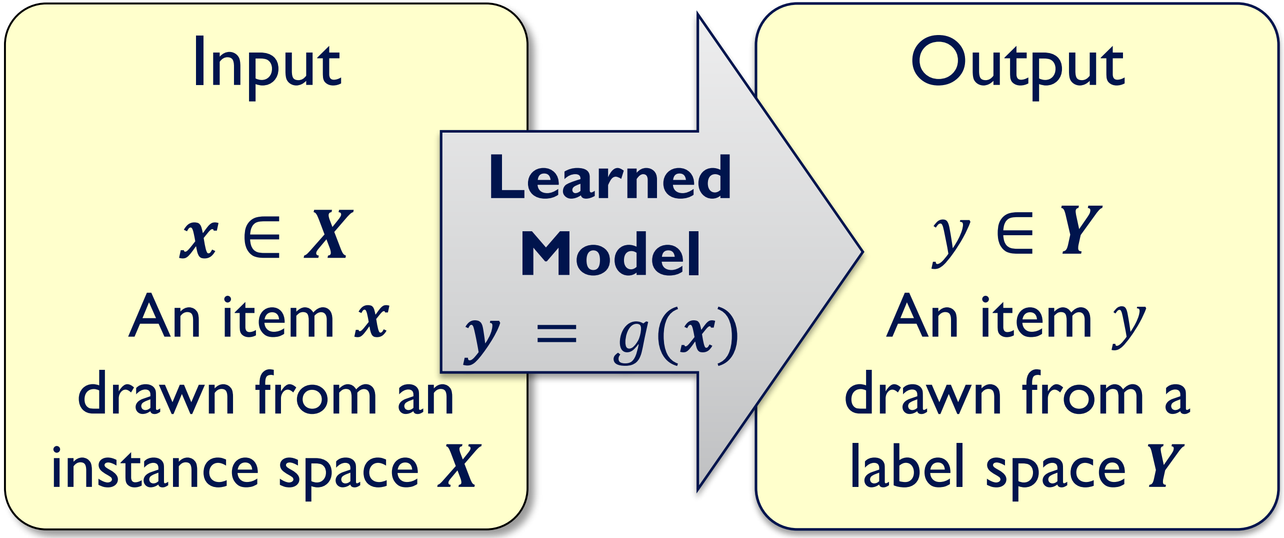



I met with him before leaving for Paris on Thursday.

Supervised learning tasks II

- Output labels $y \in Y$ are numerical:
  - Regression (linear/polynomial):
    - Labels are continuous-valued
    - Learn a linear/polynomial function $f(\boldsymbol{x})$
  - Ranking:
    - Labels are ordinal
    - Learn an ordering $f(x_1) > f(x_2)$ over input

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

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

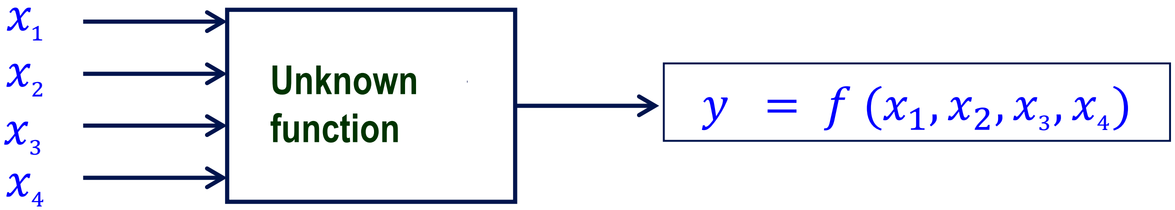
Output

$y \in \boldsymbol{Y}$

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

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

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?

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 $|\boldsymbol{Y}|^{|X|}$ 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 $\boldsymbol{H}$. $\boldsymbol{H} \subseteq |\boldsymbol{Y}|^{|X|}$

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 $y = x_i \wedge x_j \wedge x_k$

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

Rule	Counterexample	Rule	Counterexample
$y = c$		$x_2 \wedge x_3$	0011 1
x_1	1100 0	$x_2 \wedge x_4$	0011 1
x_2	0100 0	$x_3 \wedge x_4$	1001 1
x_3	0110 0	$x_1 \wedge x_2 \wedge x_3$	0011 1
x_4	0101 1	$x_1 \wedge x_2 \wedge x_4$	0011 1
$x_1 \wedge x_2$	1100 0	$x_1 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_3$	0011 1	$x_2 \wedge x_3 \wedge x_4$	0011 1
$x_1 \wedge x_4$	0011 1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0011 1

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”

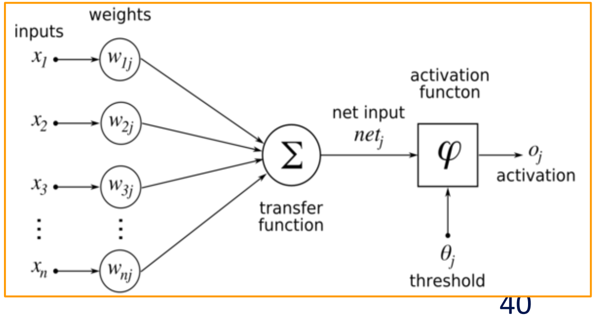
Notation: 2 variables from the set on the left.
Value: Index of the counterexample.

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

Found a consistent hypothesis!

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



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 an 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

Administration

- The class is still full.
  - All people in category <5 got in. Many in Category 5 too.
  - We will support petitions for more people as current students drop.
  - We are considering moving to a larger classroom to accommodate others. Stay tuned.
- You all need to complete HW0!
- 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.
- I do not have office hours this week.

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.

- An option: 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 whether points, some are weather points

Learning Protocol?

Supervised? Unsupervised?

Representation Step: What's Good?

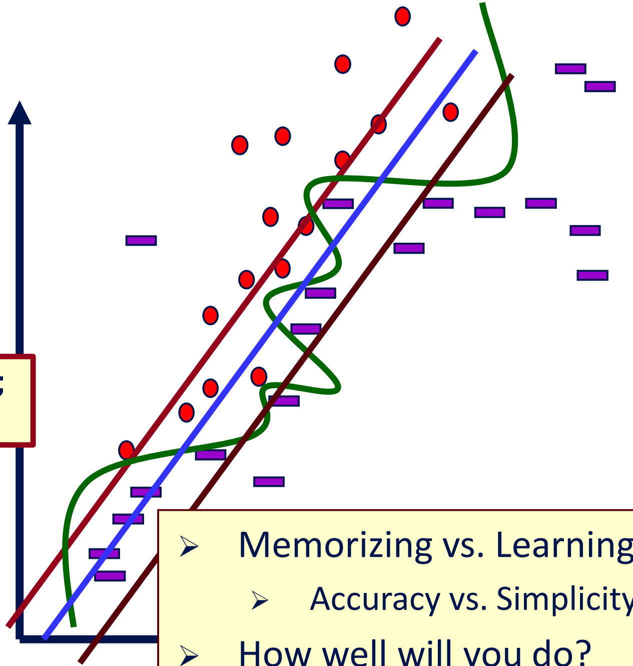
- Learning problem:
  - Find a function that best separates the data
- What function?
- What's best?
- (How to find it?)

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

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

Linear = linear in the feature space
$\boldsymbol{x}$ = data representation; $\boldsymbol{w}$ = the classifier ($\boldsymbol{w}, \boldsymbol{x}$, column vectors of dimensionality n)

$$y = \mathrm{sgn}\{\boldsymbol{w}^T \boldsymbol{x}\}$$



- Memorizing vs. Learning
  - Accuracy vs. Simplicity
- How well will you do?
  - On what?
- Impact on Generalization

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

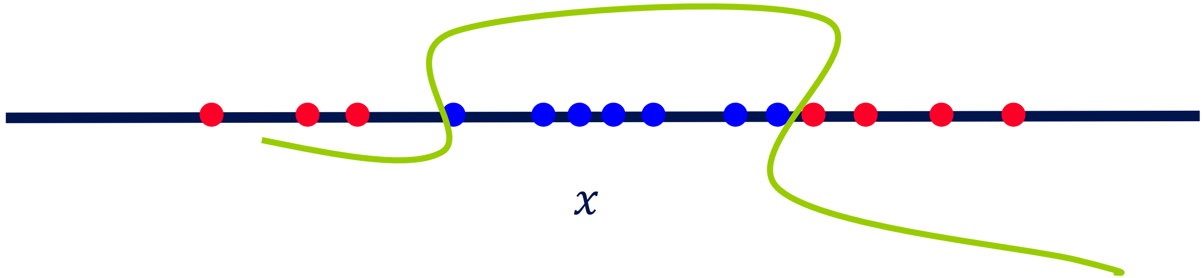
Expressivity

Probabilistic Classifiers as well

- $f(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^T \cdot \boldsymbol{x} \ - \theta\} = \mathrm{sgn}\{\sum_{i=1}^{n} w_i x_i \ - \theta\}$
- Many functions are Linear
  - Conjunctions:
    - $y = x_1 \wedge x_3 \wedge x_5$
    - $y = \mathrm{sgn}\{1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 \ - 3\}; \boldsymbol{w} = (1,0,1,0,1) \ \ \theta = 3$
  - 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
- Note: all the variables above are Boolean variables

Functions Can be Made Linear

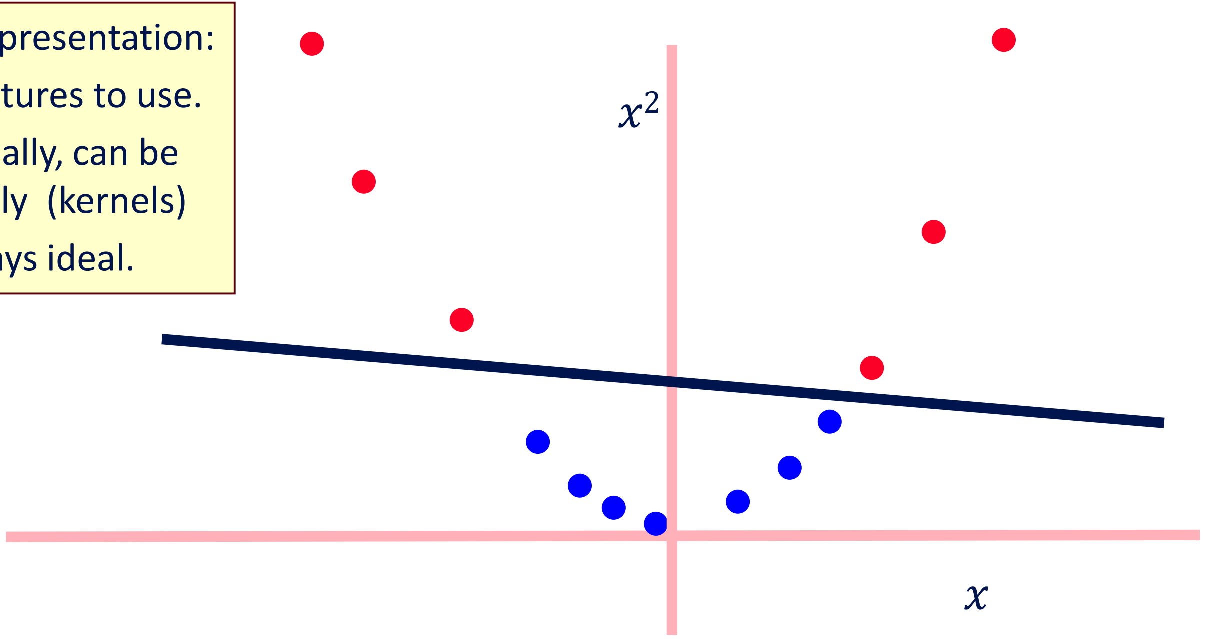
- Data points are not linearly separable in one dimension
- Not separable if you insist on using a specific class of functions (e.g., linear)



Blown Up Feature Space

- Data are separable in $< x, x^2 >$ space

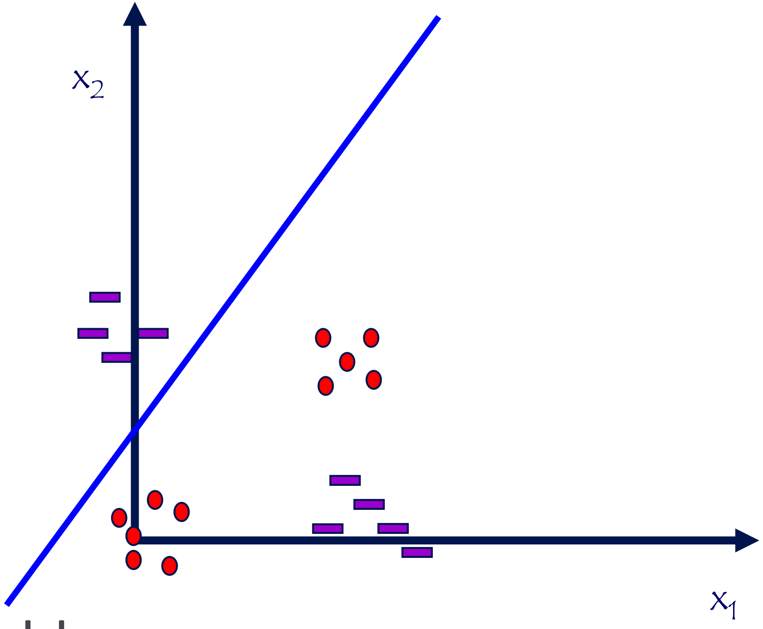
- Key issue: Representation:
  - what features to use.
- Computationally, can be done implicitly (kernels)
  - Not always ideal.



Exclusive-OR (XOR)

- $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$
- In general: a parity function.
- $x_i \in \{0,1\}$
- $\mathrm{f}(x_1, x_2, \ldots, x_n) = 1$

  iff $\sum x_i$ is even



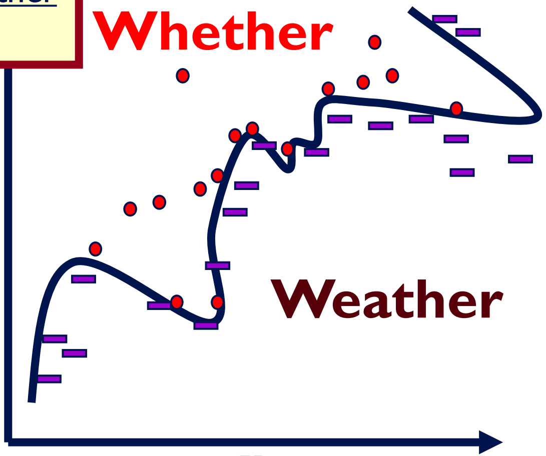
- This function is not linearly separable.

Functions Can be Made Linear

Discrete Case

$x_1\, x_2\, x_4 \lor x_2\, x_4\, x_5 \lor x_1\, x_3\, x_7$

A real Weather/Whether example

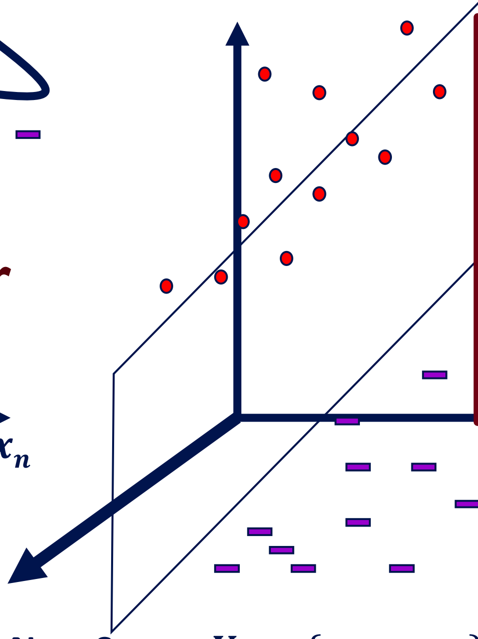


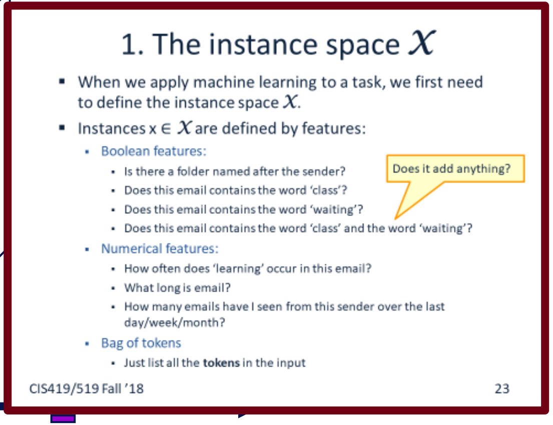
Space: $X = x_1, x_2, \ldots, x_n$

Input Transformation

$y_3 \lor y_4 \lor y_7$

New discriminator is functionally simpler





1. The instance space X

- When we apply machine learning to a task, we first need to define the instance space X.
- Instances x ∈ X are defined by features:
  - Boolean features:
    - Is there a folder named after the sender?
    - Does this email contains the word 'class'?
    - Does this email contains the word 'waiting'?
    - Does this email contains the word 'class' and the word 'waiting'?
  - Numerical features:
    - How often does 'learning' occur in this email?
    - What long is 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?

CIS419/519 Fall '18 23

New Space: $Y = \{y_1, y_2, \ldots\} = \{x_i, x_i, x_j, x_i\, x_j\, x_{j, \ldots}\}$

Representation (1)

Feature Types:

(what does the algorithm know about the input):

1,2. relative position (+/-1) has this pos/w

3. Conjunctions of size two

4. word w occurs in (-2,+2) window around target

Note: 4 feature **types**; many features

```
1p=Det 0.972222 0 34
1w=the 0.961538 0 24
-1p=Punc 1p=Pro 0.96 0 23
-1p=V 1p=Adv 0.96 0 23
or 0.959184 3 93
1p=Adj 0.957447 1 44
not 0.956522 1 43
-1p=V 1p=Pro 0.956522 0 21
1p=Det 2p=Ns 0.954545 0 20
-1p=V 1p=Det 0.954545 0 20
1w=he 0.952381 0 19
-1p=Prep 1p=Pro 0.952381 0 19
-2w=to -1p=V 0.952381 0 19
1p=Pro 2p=Vpp 0.947368 0 17
question 0.947368 0 17
1p=Pro 0.946237 4 87
-1w=, 1p=Pro 0.944444 0 16
-1p=Punc 0.941176 2 47
should 0.941176 0 15
1w=they 0.941176 0 15
-1p=V 1w=the 0.941176 0 15
-1p=Ns 1p=Adj 0.9375 0 14
1p=Pro 2p=Was 0.9375 0 14
1p=Pro 2p=Vpt 0.9375 0 14
-1p=Punc 1p=Adj 0.9375 0 14
-2p=Ns -1p=Punc 0.933333 1 27
1p=Adj 2p=Prep 0.933333 0 13
1p=Ns 2p=Vpp 0.933333 0 13
-1p=Conj 1p=Pro 0.933333 0 13
you 0.933333 0 13
1p=Ns 2p=Was 0.933333 0 13
1w=it 0.933333 0 13
-1p=Punc 1p=Prep 0.933333 0 13
-1w=, 1p=Adj 0.928571 0 12
-1p[illegible] 1p=Ns 0.928571 1 25
-1p=V 1p=Adj 0.928571 0 12
1p=Ns 2w=was 0.928571 0 12
```

The feature resulting from instantiating the type in the given data

Some statistics (not part of the learning process; just for the understanding of the problem)

Representation (2)

Extracting features from the data:

(what does the algorithm know about the input):

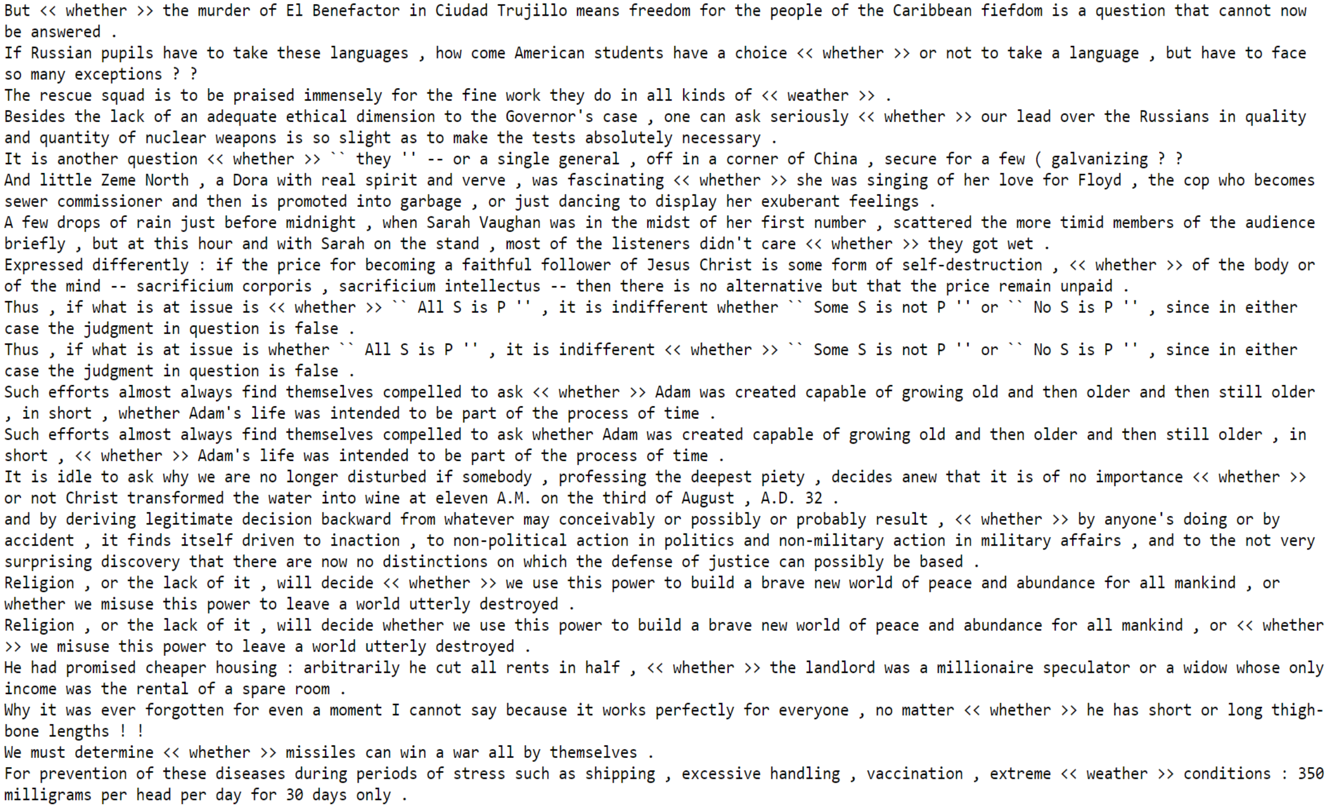
1,2. relative position (+/- 1); pos/w

3. Conjunctions of size two

4. Occurrence of a word in a window around the target

Note: 2 feature types; many features

For each feature type, the data gives rise to multiple features; you don't know which, before you see the data.



```
But << whether >> the murder of El Benefactor in Ciudad Trujillo means freedom for the people of the Caribbean fiefdom is a question that cannot now
be answered .
If Russian pupils have to take these languages , how come American students have a choice << whether >> or not to take a language , but have to face
so many exceptions ? ?
The rescue squad is to be praised immensely for the fine work they do in all kinds of << weather >> .
Besides the lack of an adequate ethical dimension to the Governor's case , one can ask seriously << whether >> our lead over the Russians in quality
and quantity of nuclear weapons is so slight as to make the tests absolutely necessary .
It is another question << whether >> `` they '' -- or a single general , off in a corner of China , secure for a few ( galvanizing ? ?
And little Zeme North , a Dora with real spirit and verve , was fascinating << whether >> she was singing of her love for Floyd , the cop who becomes
sewer commissioner and then is promoted into garbage , or just dancing to display her exuberant feelings .
A few drops of rain just before midnight , when Sarah Vaughan was in the midst of her first number , scattered the more timid members of the audience
briefly , but at this hour and with Sarah on the stand , most of the listeners didn't care << whether >> they got wet .
Expressed differently : if the price for becoming a faithful follower of Jesus Christ is some form of self-destruction , << whether >> of the body or
of the mind -- sacrificium corporis , sacrificium intellectus -- then there is no alternative but that the price remain unpaid .
Thus , if what is at issue is << whether >> `` All S is P '' , it is indifferent whether `` Some S is not P '' or `` No S is P '' , since in either
case the judgment in question is false .
Thus , if what is at issue is whether `` All S is P '' , it is indifferent << whether >> `` Some S is not P '' or `` No S is P '' , since in either
case the judgment in question is false .
Such efforts almost always find themselves compelled to ask << whether >> Adam was created capable of growing old and then older and then still older
, in short , whether Adam's life was intended to be part of the process of time .
Such efforts almost always find themselves compelled to ask whether Adam was created capable of growing old and then older and then still older , in
short , << whether >> Adam's life was intended to be part of the process of time .
It is idle to ask why we are no longer disturbed if somebody , professing the deepest piety , decides anew that it is of no importance << whether >>
or not Christ transformed the water into wine at eleven A.M. on the third of August , A.D. 32 .
and by deriving legitimate decision backward from whatever may conceivably or possibly or probably result , << whether >> by anyone's doing or by
accident , it finds itself driven to inaction , to non-political action in politics and non-military action in military affairs , and to the not very
surprising discovery that there are now no distinctions on which the defense of justice can possibly be based .
Religion , or the lack of it , will decide << whether >> we use this power to build a brave new world of peace and abundance for all mankind , or
whether we misuse this power to leave a world utterly destroyed .
Religion , or the lack of it , will decide whether we use this power to build a brave new world of peace and abundance for all mankind , or << whether
>> we misuse this power to leave a world utterly destroyed .
He had promised cheaper housing : arbitrarily he cut all rents in half , << whether >> the landlord was a millionaire speculator or a widow whose only
income was the rental of a spare room .
Why it was ever forgotten for even a moment I cannot say because it works perfectly for everyone , no matter << whether >> he has short or long thigh-
bone lengths ! !
We must determine << whether >> missiles can win a war all by themselves .
For prevention of these diseases during periods of stress such as shipping , excessive handling , vaccination , extreme << weather >> conditions : 350
milligrams per head per day for 30 days only .
```

Representation (3)

Here the first index (0/1) is the label)

```
1,2,3,11,48,51,82,87,92,105,176,198,199,279,327,413,439,482,498,747,963,964,965,975,1029,1072,1166,1176:
1,6,9,42,43,44,45,47,73,155,157,216,226,238,240,322,441,497,498,757,829,971,978,987,995,1000,1047,1060,1073,1077,1078,1086,1116,1118,1119,1132,1138,11
39,1141,1172:
0,57,94,109,112,119,153,305,387,482,759,855,963,964,965,1002,1050,1068,1142,1207,1226:
1,7,52,60,65,141,197,240,469,497,772,963,964,965,989,1045,1067,1072:
1,6,16,57,69,116,137,153,156,201,213,280,302,304,322,325,432,468,497,830,986,987,1058,1143:
1,17,26,32,36,46,119,148,157,239,292,299,322,329,433,474,497,919,963,964,978,1001,1045,1061,1071,1116,1131,1209:
1,8,15,17,20,25,57,103,116,224,370,414,458,497,719,956,963,964,987,1068:
1,6,21,27,29,41,58,67,69,155,238,271,304,460,497,963,964,1027,1111:
1,54,69,78,91,100,110,115,153,156,195,250,283,302,305,325,497,656,753,756,833,957,977,986,1065,1066:
1,6,9,37,69,153,156,302,305,325,443,497,597,729,986,1013,1040,1065,1111,1116:
1,7,12,32,36,39,89,116,152,154,157,222,240,329,827,938,943,963,1045,1188:
1,21,27,41,67,68,98,121,133,162,232,240,329,489,497,886,963,964,965,1045:
1,6,9,42,43,44,45,47,69,155,156,157,216,238,441,497,757,833,954,963,964,966,968,971,987,995,1040,1078,1086,1101,1118,1143,1172:
1,6,7,21,24,27,29,35,38,41,56,58,61,67,87,125,156,157,223,232,238,455,489,497,774,1203:
1,6,8,17,62,99,116,147,156,220,237,240,275,276,284,298,322,326,486,497,499,559,561,961,963,964,970,1039:
1,6,13,17,28,62,90,93,101,103,119,147,158,161,169,194,237,240,275,276,305,322,323,326,482,487,491,497,961,963,970,987,1029,1045:
1,2,3,6,11,21,27,41,48,61,67,87,98,121,125,133,162,204,217,229,256,305,322,329,439,497,517,630,964,965:
1,6,8,14,17,75,89,103,116,119,154,156,157,182,229,296,497,499,536,679,684,700,784,888,927,966,971,976,987,1040,1058,1086,1106,1143,1190:
1,99,116,117,205,212,237,305,322,455,499,601,943,989,999,1068,1071,1091,1127:
0,119,217,440,487,491,497,743,791,847,938,987,1016,1046,1061,1071,1116,1129,1145,1168,1196:
0,305,322,329,482,812,963,964,965,1002,1045,1050,1068,1142,1196,1207:
0,37,49,127,153,172,201,324,354,490,497,498,531,689,908,963,964,965,966,972,987,994,1000,1007,1028,1045,1047,1049,1052,1056,1057,1065,1066,1072,1081,1
086,1094,1116,1117,1120,1122,1135,1138,1140,1143,1204:
0,87,198,321,446,497,908,954,963,964,966,974,987,997,1001,1015,1045,1052,1072,1116,1120,1140:
1,7,9,57,73,80,157,197,226,237,240,295,356,437,442,484,493,497,893,917,1067,1072,1101,1120:
1,9,57,356,428,453,485,493,497,893,963,964,965,1032,1045,1072,1120:
1,6,7,21,24,29,38,61,69,80,87,157,208,211,238,295,304,364,419,437,442,606,628,739,776,803,963,964,965,1220:
1,6,87,97,113,238,240,322,327,328,750,751,809,881,988:
1,7,16,52,54,69,110,115,221,264,289,290,497,963,964,965:
1,5,6,9,40,84,87,97,107,113,116,122,154,229,238,240,253,320,321,327,328,357,385,415,439,497,508,966,969,984,987,1099,1172:
1,2,6,54,69,72,110,115,119,156,157,200,236,238,294,322,383,430,473,497,691,755,964,989,1025,1046,1055:
1,6,83,90,101,119,157,158,161,169,184,194,202,323,482,486,487,491,497,963,964,967,971,973,987,989,998,1001,1029,1045,1070,1109:
1,7,13,16,17,34,51,68,69,87,104,118,222,332,360,386,482,486,497,657,658,731,830,957,963,964,965,993,1004,1121,1142:
1,4,6,14,17,18,21,38,41,56,69,133,144,157,160,162,179,229,233,240,295,408,442,497,578,579,702,737,738,926,957,965,1001,1068:
1,6,21,41,61,87,125,219,305,419,488,497,963,965,1046,1110,1209:
```

Each example corresponds to one target occurrence; all the features for this target are collected into a vector, and the label is added.

Here:

- Sparse Representation of the feature vector. **Why?**
- Variable size: **Why?**

Administration

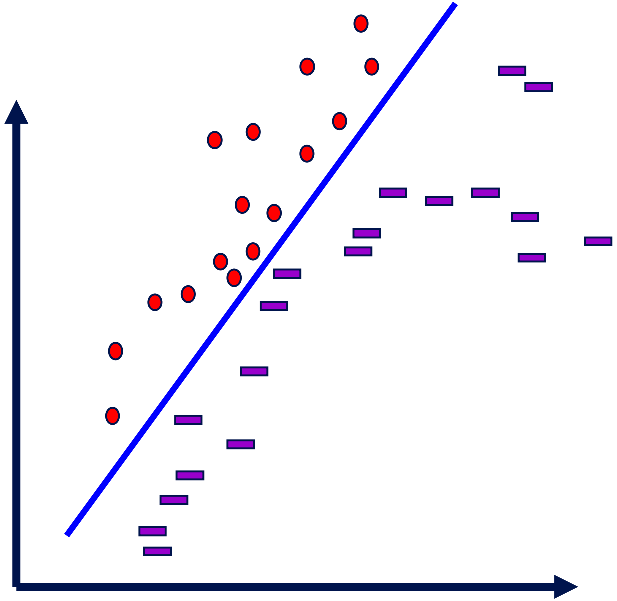
- "Easy" Registration Period is over.
  - But there will be chances to petition and get in as people drop.
  - If you want to switch 419/519, talk with Nicholas Mancuso nmancuso@seas.upenn.edu
- You all need to complete HW0!
- 1st quiz was due last night.
- HW 1 will be released early next week.
- Questions?
  - Please ask/comment during class.

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



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 $\{(\boldsymbol{x}, y)_i\}_{i=1,n}$
- Most relevant - Classification: $y \in \{0,1\}$ (or $y \in \{1,2, \dots, k\}$)
  - (But, within the same framework can also talk about Regression, y ϵ $\boldsymbol{R}$)
- What do we want $f(\boldsymbol{x})$ to satisfy?
  - We want to minimize the Risk: $L\left(f(\,)\right) = E_{X,Y}(\,[f(\boldsymbol{x}) \neq y])$
  - Where: $E_{X,Y}$ denotes the expectation with respect to the true distribution.

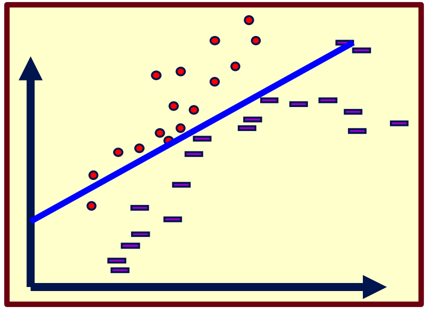
Simple loss function: # of mistakes
[…] is a indicator function

A General Framework for Learning (II)

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

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

- We cannot minimize this loss
- 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

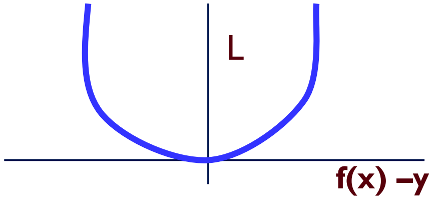
- A Loss Function $L(f(\boldsymbol{x}), y)$ measures the penalty incurred by a classifier f on example $(\boldsymbol{x}, y)$.
- There are many different loss functions one could define:
  - Misclassification Error:

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

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

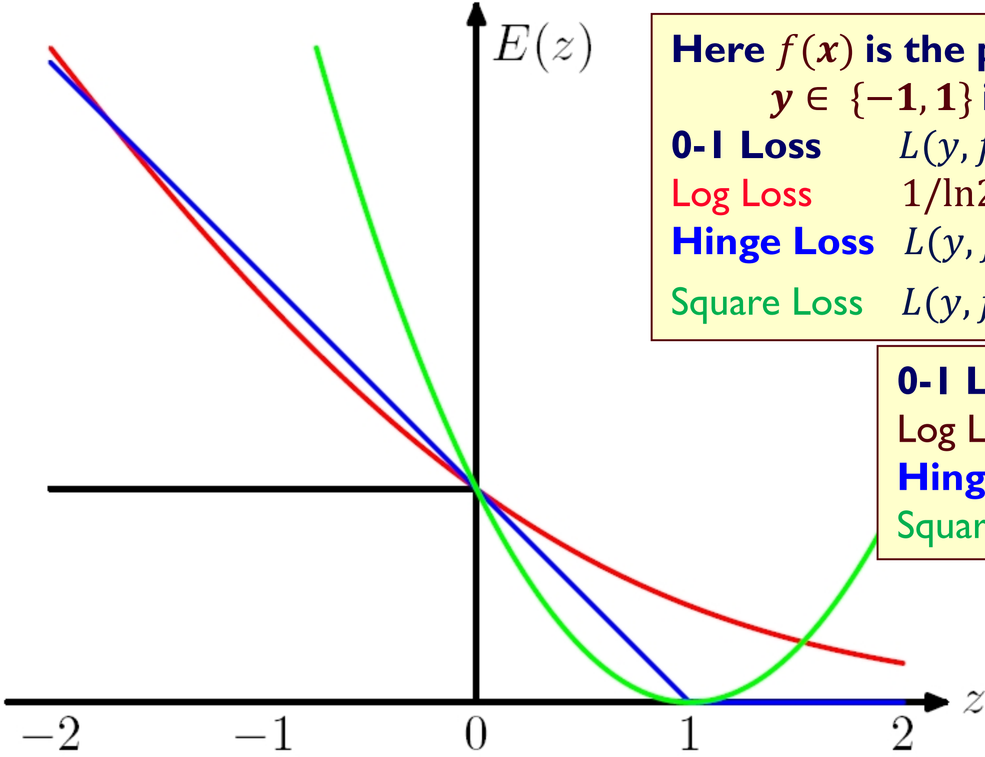
    $L(f(\boldsymbol{x}), y) = 0$ if $f(\boldsymbol{x}) = y$; $c(\boldsymbol{x})$otherwise.

A continuous convex loss function allows a simpler optimization algorithm.



Loss

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



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

0-1 Loss $L(y, f(x)) = ½\,(1 - \mathrm{sgn}(yf(\boldsymbol{x})))$

Log Loss $1/\ln 2 \log(1 + \exp\{-yf(x)\})$

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 : x axis = $yf(\boldsymbol{x})$

Log Loss: x axis = $yf(\boldsymbol{x})$

Hinge Loss: x axis = $yf(\boldsymbol{x})$

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

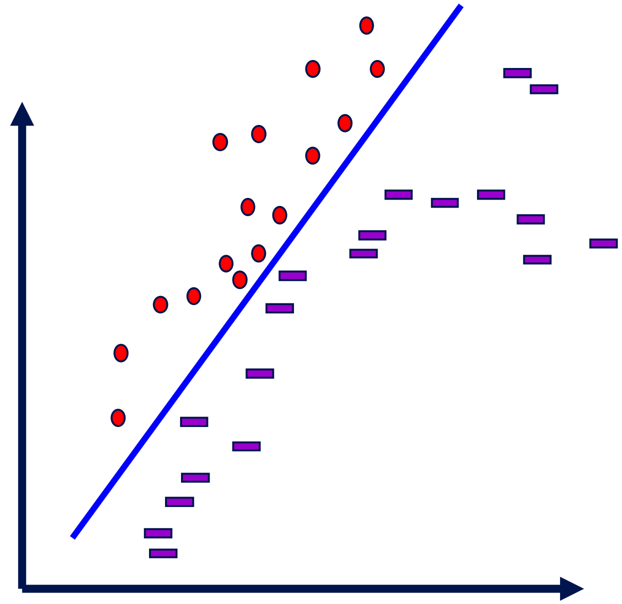
Example

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

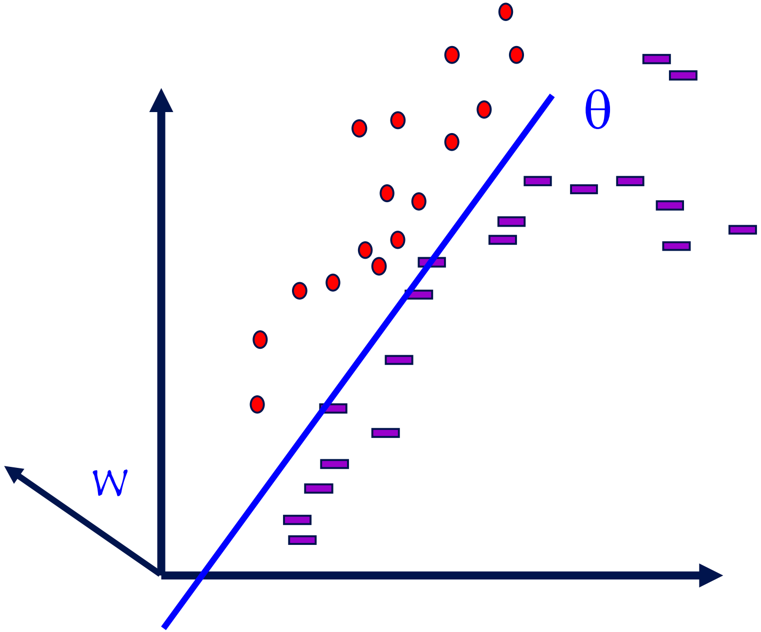


Learning Linear Separators

(LTU=Linear Threshold Unit)

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

- $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ 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

$$\boldsymbol{E}(\boldsymbol{w}) = \boldsymbol{E}_{X,Y}\, \boldsymbol{Q}(\boldsymbol{x}, y, \boldsymbol{w})$$

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

The Risk (Err) E: a function of w

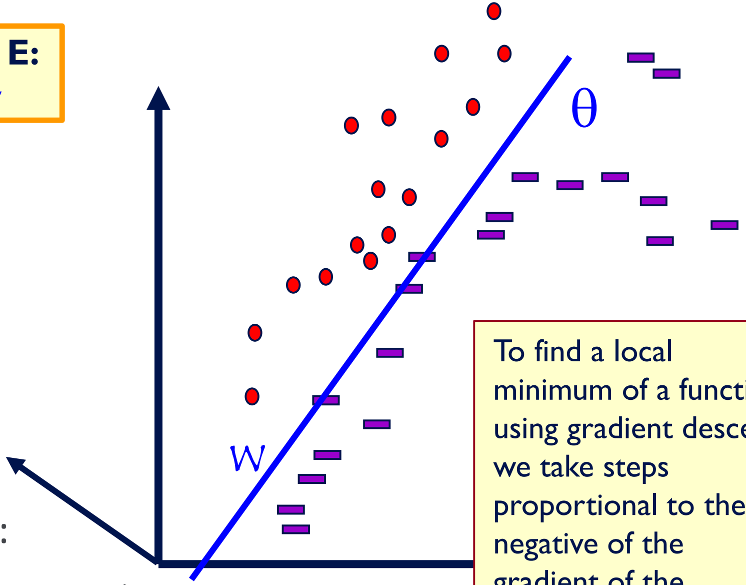
- We cannot do it.
- **Instead,** we approximate $E(\boldsymbol{w})$ using a finite training set of independent samples (x_i, y_i)

$$\boldsymbol{E}(\boldsymbol{w}) \sim = \sim 1/m \sum_{\boldsymbol{1},\, \boldsymbol{m}} Q(x_i\,, y_i, \boldsymbol{w})$$

- 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 $\boldsymbol{w}$:

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

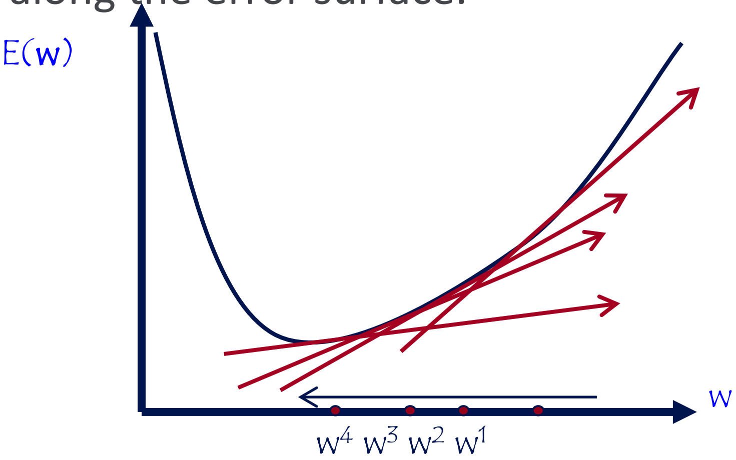
t here is "time" or "iteration" #



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

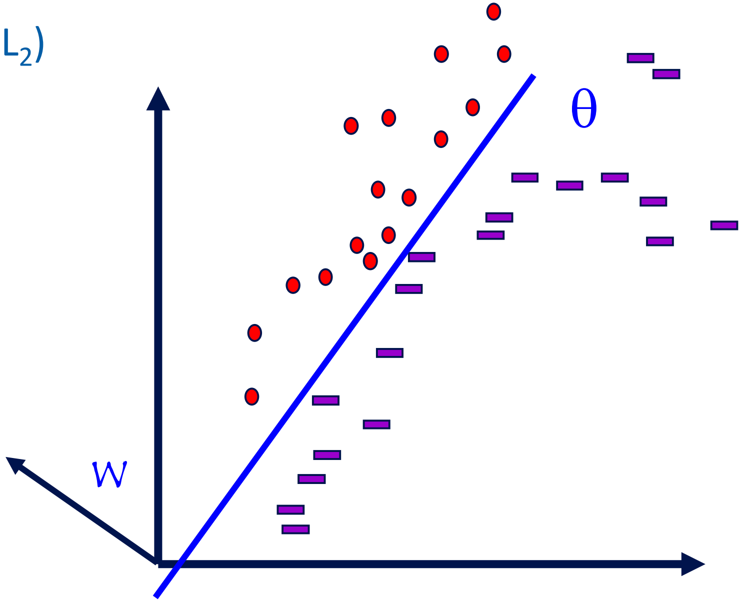
- We use gradient descent to determine the weight vector that minimizes $E(\boldsymbol{w})\ (= \ Err\ (\boldsymbol{w}))$;
- Fixing the set D of examples, E=Err is a function of $\boldsymbol{w}$
- At each step, the weight vector is modified in the direction that produces the steepest descent along the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

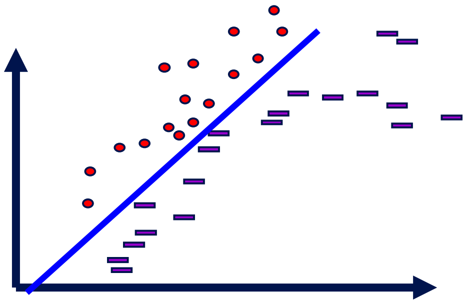
- Our Hypothesis Space is the collection of **Linear Threshold Units**
- Loss function:
  - Squared loss: LMS (Least Mean Square, L_2)
  - $Q(\boldsymbol{x}, y, \boldsymbol{w}) = ½ (\boldsymbol{w}^T \boldsymbol{x} - y)^2$



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 $t_{\boldsymbol{d}}$ be the target value for this example
- The error the current hypothesis makes on the data set is:

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

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

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

- 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 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:

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

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

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

Alg1: Gradient Descent: LMS

- Weight update rule:

$$\Delta \mathrm{w}_{\mathrm{i}} = R \sum_{\mathrm{d} \in D} (t_d - O_d) x_{id}$$

- 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 $\Delta \mathrm{w}_\mathrm{i}$ 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:

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

- 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

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 $\mathbf{w}^t$

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.
- 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 – including those for non-linear hypothesis spaces – can be traced back to gradient descent algorithms 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 complexity: 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