Lecture 2:

Introduction to Machine Learning

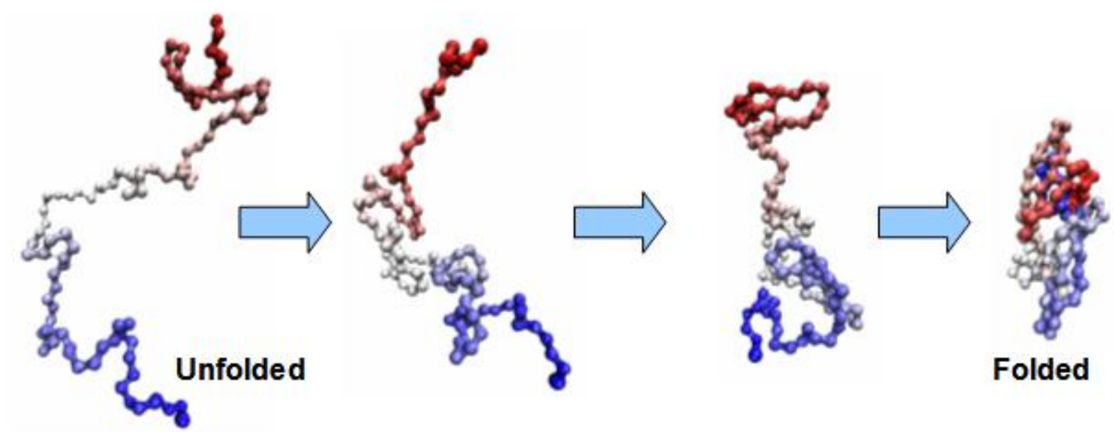
Winter 2018

Kai-Wei Chang
CS @ UCLA
kw+cm146@kwchang.net

The instructor gratefully acknowledges Eric Eaton (UPenn), who assembled the original slides, Jessica Wu (Harvey Mudd), David Kauchak (Pomona), Dan Roth (Upenn), Sriram Sankararaman (UCLA), whose slides are also heavily used, and the many others who made their course materials freely available online.

Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
  - E.g.: Identify DNA binding sites, predicting disease progression, predicting protein folding structure



Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
- Humans can perform a task, but can't describe how they do it
  - E.g.: Object recognition

Why we need machine learning?

- There is no (or limited numbers of) human expert for some problems
- Humans can perform a task, but can't describe how they do it
- The desired function is hard to be written down in a closed form
  - E.g.,: predict stock price

What is Learning

- The Badges Game……
  - This is an example of the key learning protocol: supervised learning

- Issues:
  - Prediction or Modeling?
  - Representation
  - Problem setting
  - Background Knowledge
  - When did learning take place?
  - Algorithm

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?

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

Raw test data

Gerald F. DeJong
Chris Drummond
Yolanda Gil
Attilio Giordana
Jiarong Hong
J. R. Quinlan
Priscilla Rasmussen
Dan Roth
Yoram Singer
Lyle H. Ungar

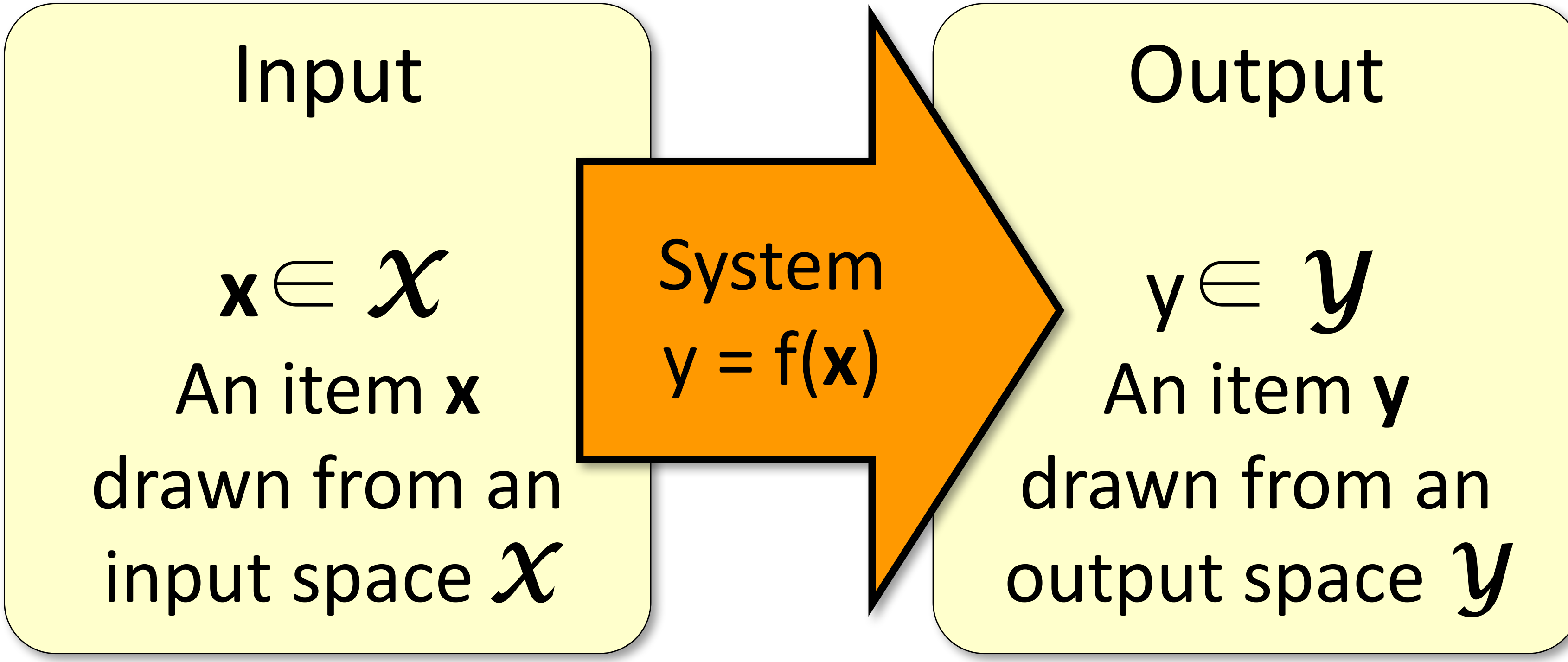
Labeled test data

+ 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

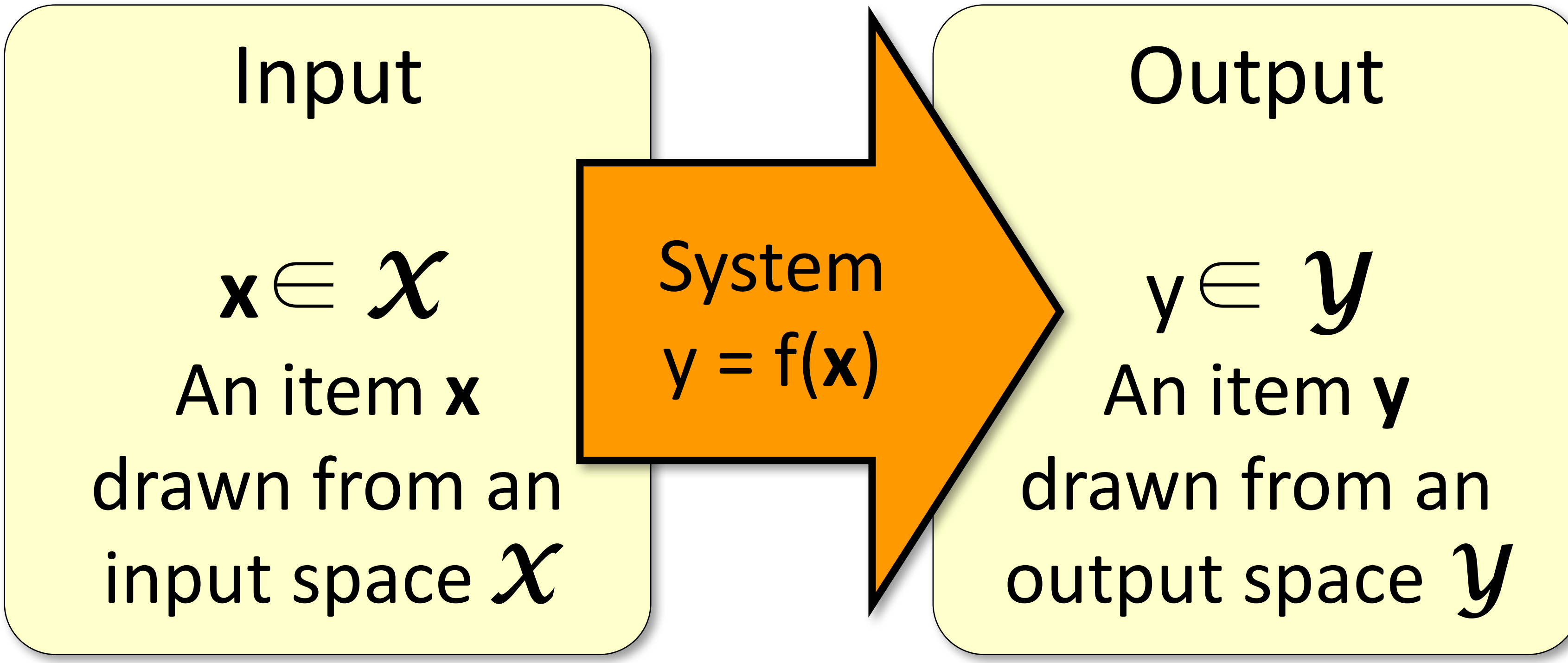
- The Badges Game……
  - This is an example of the key learning protocol: supervised learning
- Issues:
  - Prediction or Modeling?
  - Representation
  - Problem setting
  - Background Knowledge
  - When did learning take place?
  - Algorithm

Supervised Learning



- We consider systems that apply a function f() to input items **x** and return an output **y** = f(**x**).

Supervised Learning



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

Supervised learning

Input

$\mathbf{x} \in \mathcal{X}$

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

Target function
y = f(**x**)

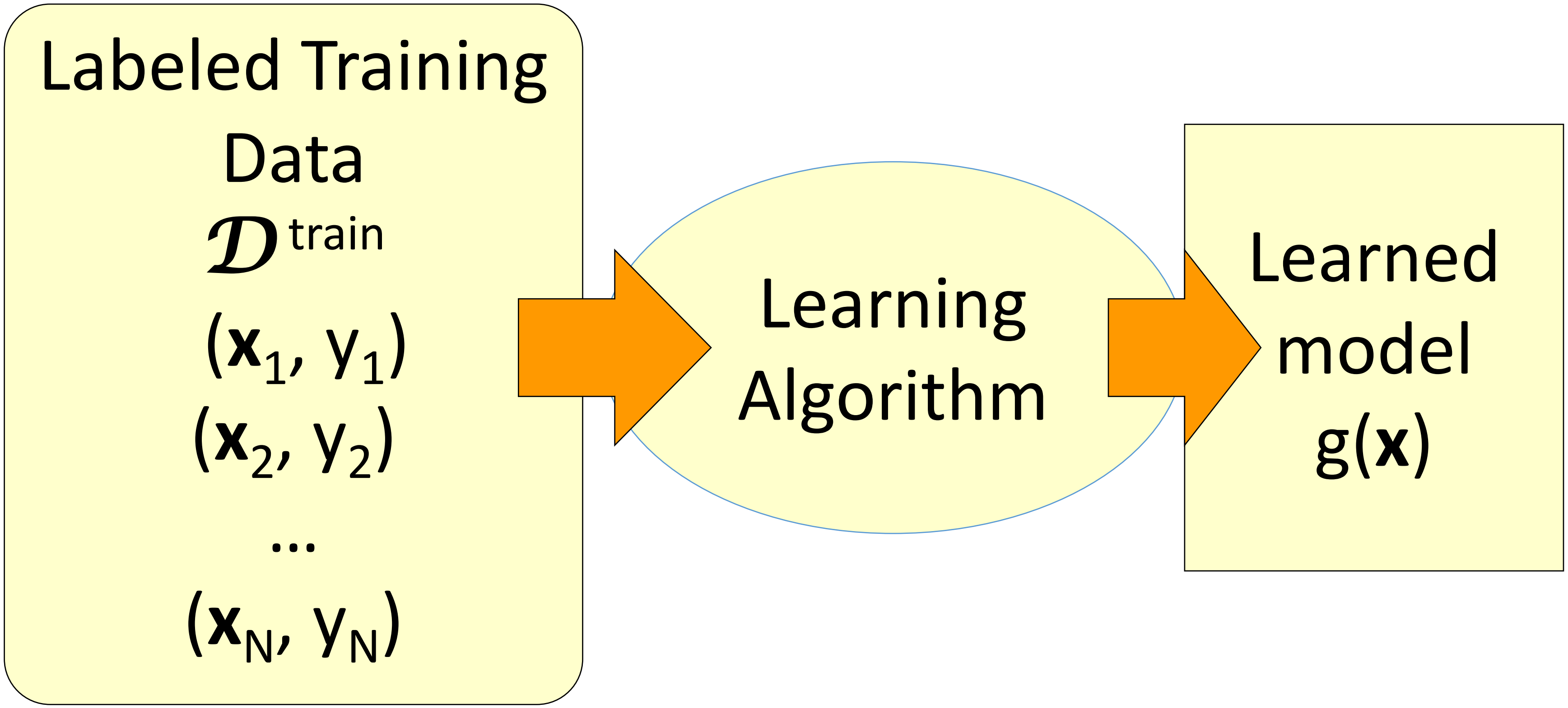
Learned Model
y = g(**x**)

Output

$y \in \mathcal{Y}$

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

Supervised learning: Training



- Give the learner examples in $\mathcal{D}^{\text{train}}$
- The learner returns a model $g(\mathbf{x})$

Supervised learning: Testing

Labeled

Test Data

$\mathcal{D}^{\text{test}}$

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

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

...

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

- Reserve some labeled data for testing

Supervised learning: Testing

Labeled Test Data

$\mathcal{D}^{\text{test}}$

$(\mathbf{x}'_1, y'_1)$
$(\mathbf{x}'_2, y'_2)$
…
$(\mathbf{x}'_M, y'_M)$

Raw Test Data

$\mathcal{X}^{\text{test}}$

$\mathbf{x}'_1$
$\mathbf{x}'_2$
….
$\mathbf{x}'_M$

Test Labels

$\mathcal{Y}^{\text{test}}$

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

Raw Test Data		Learned model		Predicted Labels	Test Labels
$\mathcal{X}^{\text{test}}$	→	$g(\mathbf{x})$	→	$g(\mathcal{X}^{\text{test}})$	$\mathcal{Y}^{\text{test}}$
$\mathbf{x}'_1$				$g(\mathbf{x}'_1)$	y'_1
$\mathbf{x}'_2$				$g(\mathbf{x}'_2)$	y'_2
….				….	…
$\mathbf{x}'_M$				$g(\mathbf{x}'_M)$	y'_M

Learning the mapping

Input

$\mathbf{x} \in \mathcal{X}$

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

Target function
$\mathbf{y} = f^*(\mathbf{x})$

Learned Model
$\mathbf{y} = f(x)$

Output

$y \in \mathcal{Y}$

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

Key Issues in Machine Learning

- Modeling
  - How to formulate application problems as machine learning problems
  - 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?

- Algorithms
  - What are good algorithms?
  - How do we define success?
  - 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?

1. Input: The instance space $\mathcal{X}$

Input

$\mathbf{x} \in \mathcal{X}$

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

x is represented in a feature space

- Typically $x \in \{0,1\}^n$ or R^N
- Usually represented as a vector
- We call it input vector

Example:

Boolean features:

Does this email contain the word 'money'?

Numerical features:

How often does 'money' occur in this email

What is the width/height of this bounding box?

What is the length of the first name?

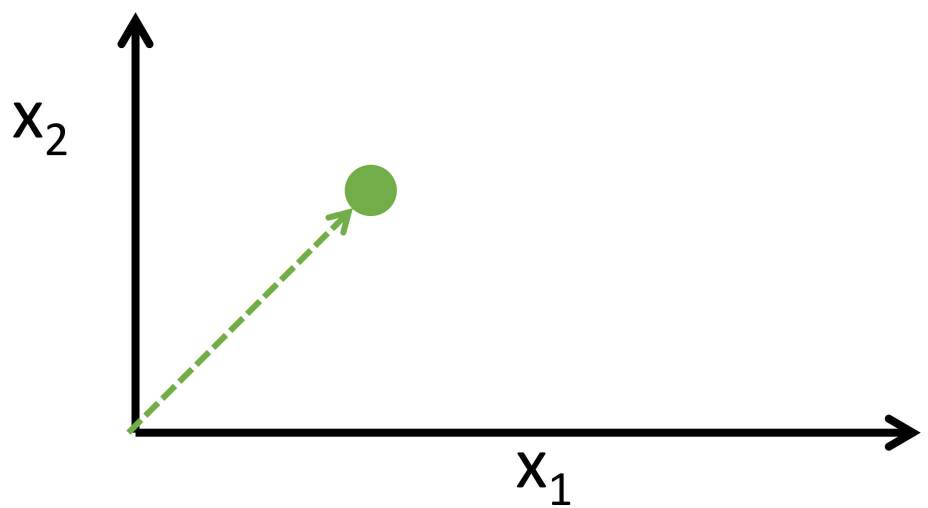
What's $\mathcal{X}$ for the Badges game?

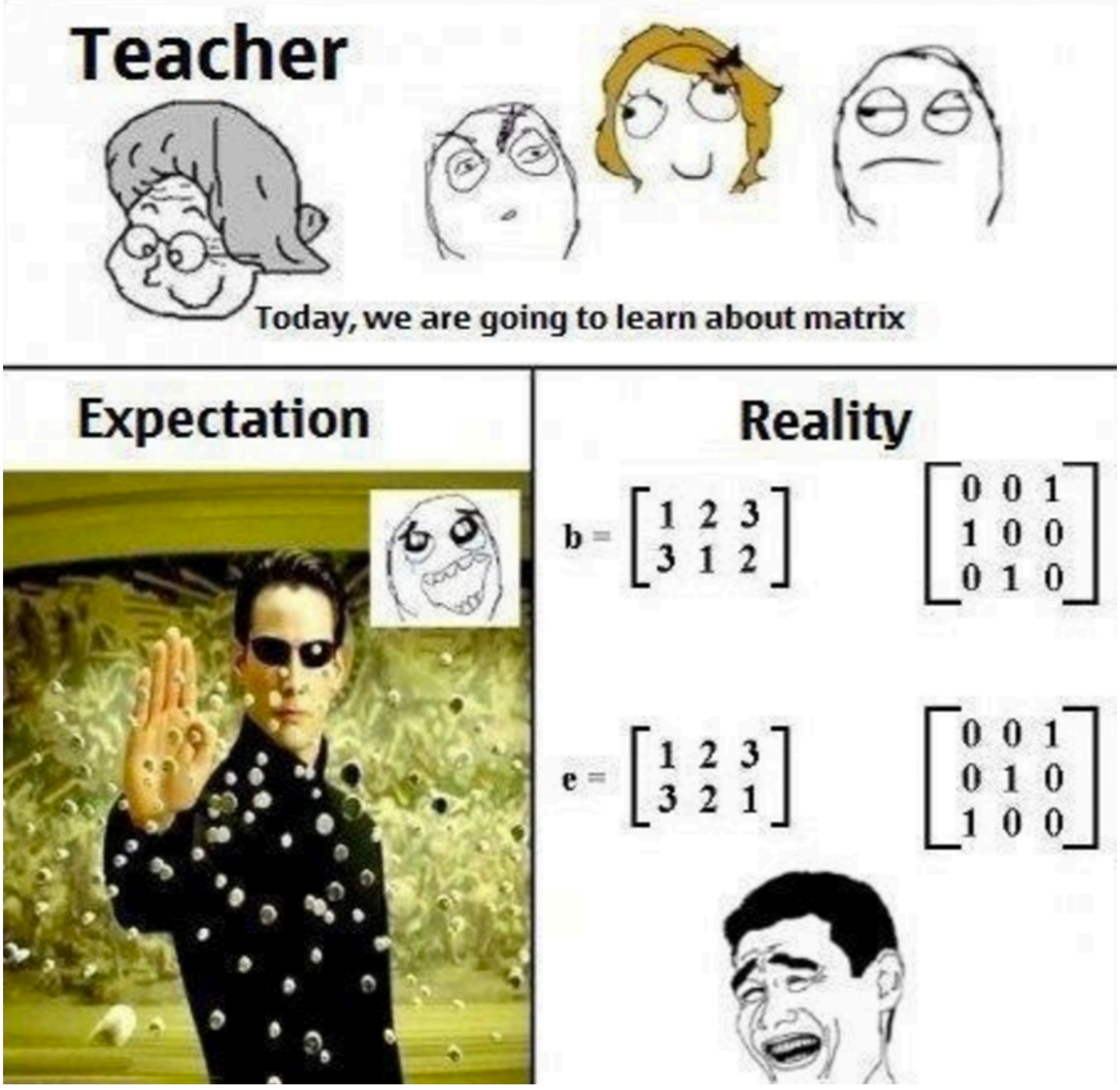
- Possible features:
  - Gender/age/country of the person?
  - Length of their first or last name?
  - Does the name contain letter 'x'?
  - How many vowels does their name contain?
  - Is the n-th letter a vowel?

+ Naoki Abe	+ Peter Bartlett	+ Carla E. Brodley
- Myriam Abramson	- Eric Baum	+ Nader Bshouty
+ David W. Aha	+ Welton Becket	- Wray Buntine
+ Kamal M. Ali	- Shai Ben-David	- Andrey Burago
- Eric Allender	+ George Berg	+ Tom Bylander
+ Dana Angluin	+ Neil Berkman	+ Bill Byrne

$\mathcal{X}$ as a vector space

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





Teacher
Today, we are going to learn about matrix
Expectation
Reality
b = [1 2 3; 3 1 2]
[0 0 1; 1 0 0; 0 1 0]
e = [1 2 3; 3 2 1]
[0 0 1; 0 1 0; 1 0 0]

Example: the badge game

+ Naoki Abe	+ Peter Bartlett	+ Carla E. Brodley
- Myriam Abramson	- Eric Baum	+ Nader Bshouty
+ David W. Aha	+ Welton Becket	- Wray Buntine
+ Kamal M. Ali	- Shai Ben-David	- Andrey Burago
- Eric Allender	+ George Berg	+ Tom Bylander
+ Dana Angluin	+ Neil Berkman	+ Bill Byrne

[first-char is vowel, first-char is A, first-char is N, second-char is vowel …]

+ Naoki Abe

[0 , 0 , 1 , 1 …]

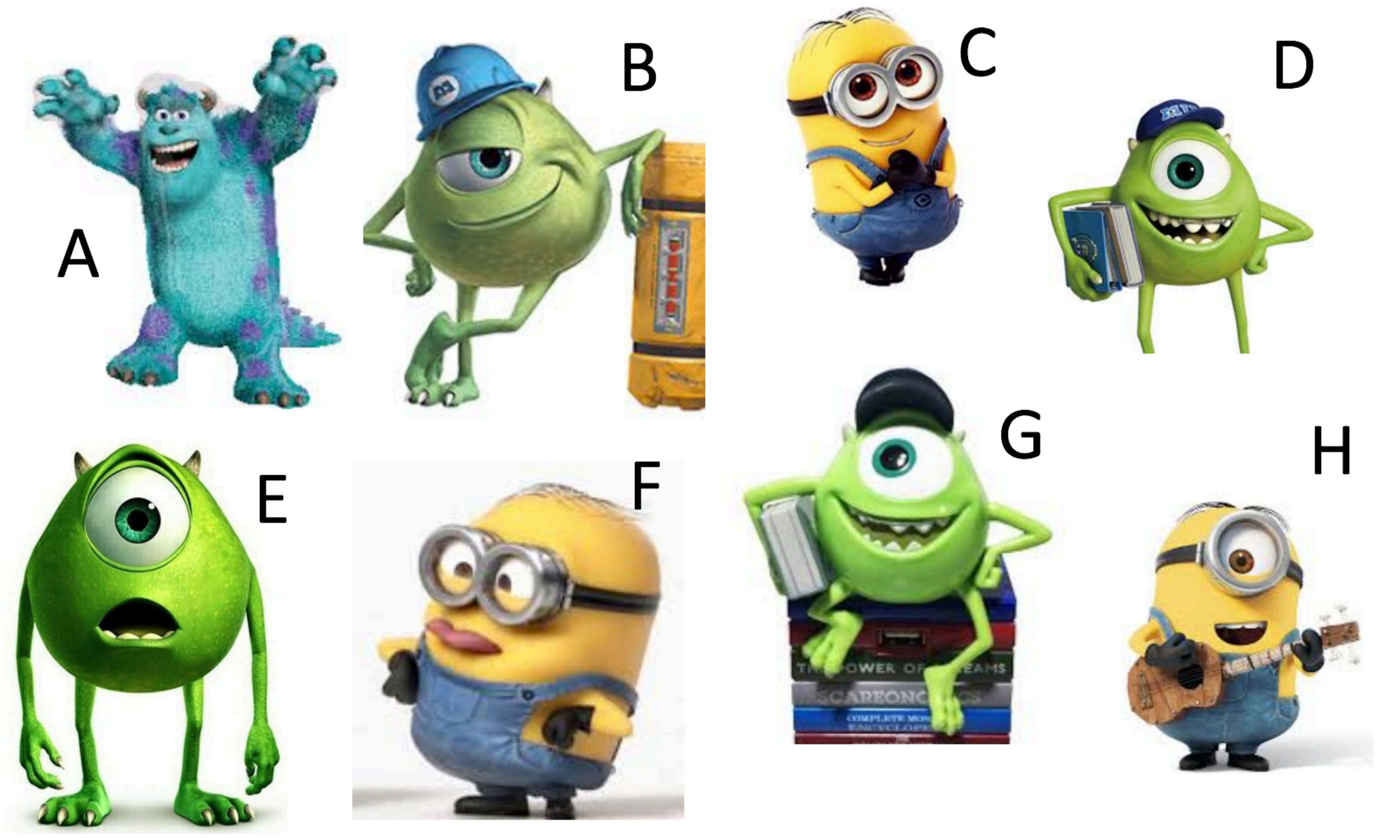
- Avrim Blum

[1 , 1 , 0 , 0 …]

From feature templates to vectors

- When designing features, we often think in terms of templates, not individual features:
- **Encoding a name by encoding it characters:**
- **What is the *i*-th letter?**
- **Abe** → [**1** 0 0 0 0… 0 1 0 0 0 0… 0 0 0 0 **1** …]
  - 26*2 + 1 positions in each group;
  - # of groups == upper bounds on length of names

Your turn – How many features you can find?



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.

- CS146 can't teach you what specific features to use for your task.
  - But we will touch on some general principles

2. Output space

y is represented in output space (label space)
Different kinds of output:

- Binary classification:
  $y \in \{-1,1\}$
- Multiclass classification:
  $y \in \{1,2,3, \dots K\}$
- Regression:
  $y \in R$
- Structured output
  $y \in \{1,2,3, \dots K\}^N$

Output

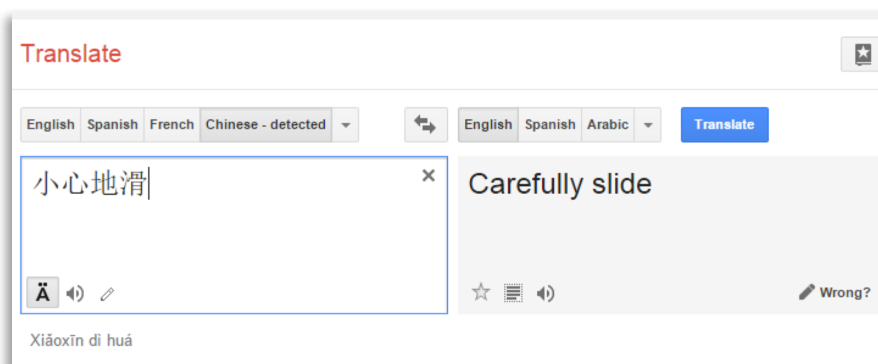
$y \in \mathcal{Y}$

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

Output space can be compositional

小心:
Carefully
Careful
Take
Care
Caution

地滑:
Slide
Landslip
Wet Floor
Smooth



Supervised Learning : Examples

- Disease diagnosis
  - x: Properties of patient (symptoms, lab tests)
  - y : Disease (or maybe: recommended therapy)
- Part-of-Speech tagging
  - x: An English sentence (e.g., The <u>can</u> will rust)
  - y : The part of speech of a word in the sentence
- Face recognition
  - x: Bitmap picture of person’s face
  - y : Name the person (or maybe: a property of)
- Automatic Steering
  - x: Bitmap picture of road surface in front of car
  - y : Degrees to turn the steering wheel

3. The model g(**x**)

Input

$\mathbf{x} \in \mathcal{X}$

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

Learned Model

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

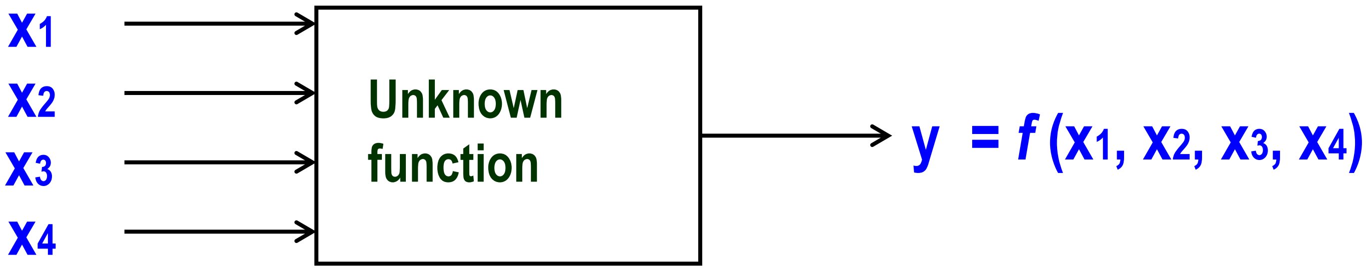
Output

$y \in \mathcal{Y}$

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

- 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?

A function g is consistent to a dataset $\mathrm{D} = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall\, i$

How may function is consistent to D on the left

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?

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

Hypothesis Space

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

We c

corre

poss

After

have 2^9 possibilities for f

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

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

Is Learning Possible?

Hypothesis Space (2)

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**

Simple Rules: There are only 16 simple **conjunctive rules**

of the form $y=x_i \wedge x_j \wedge x_k$

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**.

Hypothesis Space (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

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
{X1}				
{X2}				
{X3}				
{X4}				
{X1,X2}				
{X1, X3}				
{X1, X4}				
{X2,X3}				

variables	1-of	2-of	3-of	4-of
{X2, X4}				
{X3, X4}				
{X1,X2, X3}				
{X1,X2, X4}				
{X1,X3,X4}				
{X2, X3,X4}				
{X1, X2, X3,X4}				

Hypothesis Space (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

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.

Hypothesis Space (3)

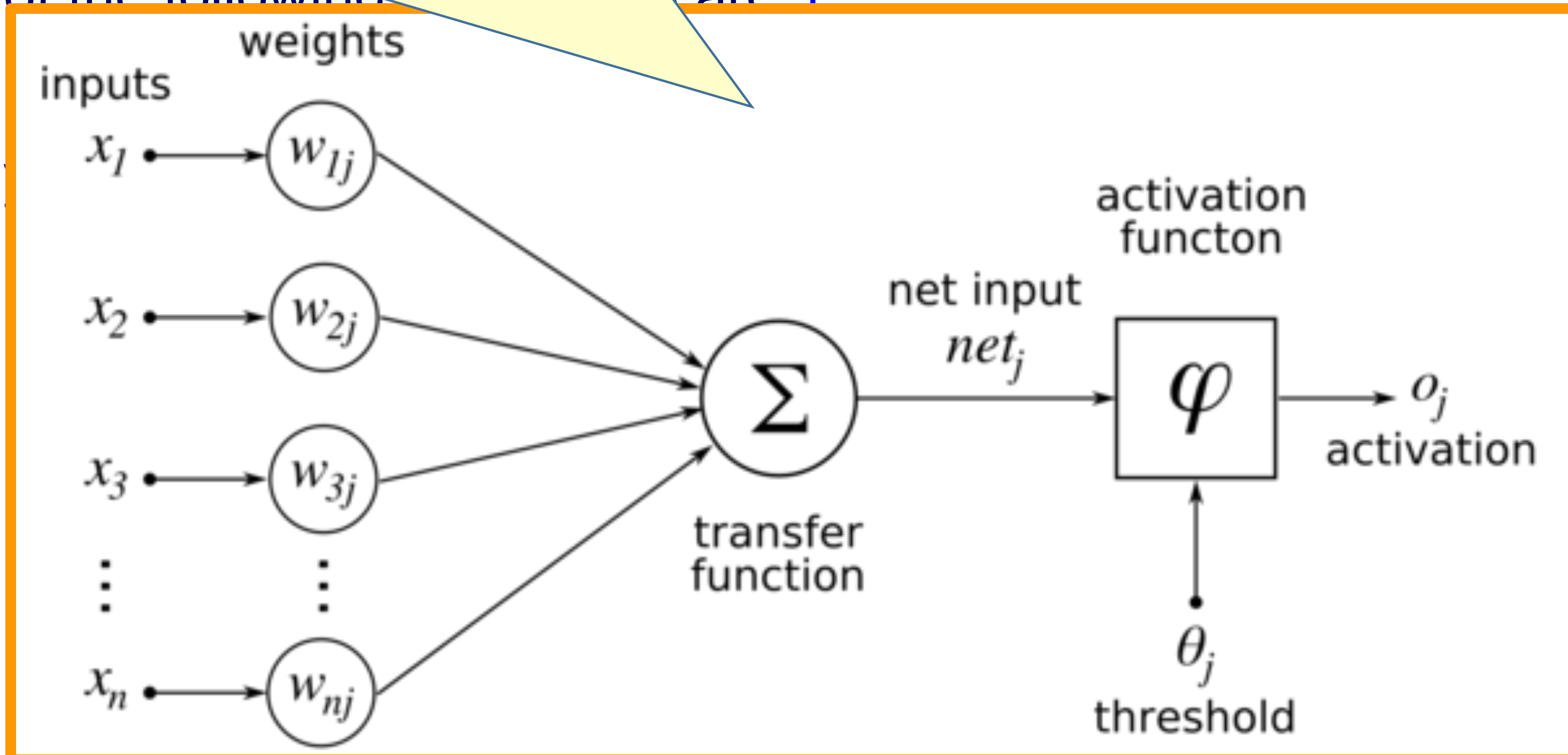
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
				1	1
				0	0
				0	0
				1	0

m-o

of th

of the followin… s are 1"

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



Found a consistent hypothesis.

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, small hypothesis class:
  - We can start with a very small class and enlarge it until it contains an hypothesis that fits the data.
- We could be wrong !
  - Our guess of the hypothesis space could be wrong
    - $y = x_4 \land$ one-of (x_1, x_3) is also consistent

General strategies for Machine Learning

- Develop flexible hypothesis spaces:
  - Decision trees, neural networks, nested collections.
- 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

General strategies for Machine Learning

- Develop flexible hypothesis spaces:
  - Decision trees, neural networks, nested collections.
- 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

An Example

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

How can we make this a learning problem?

- 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

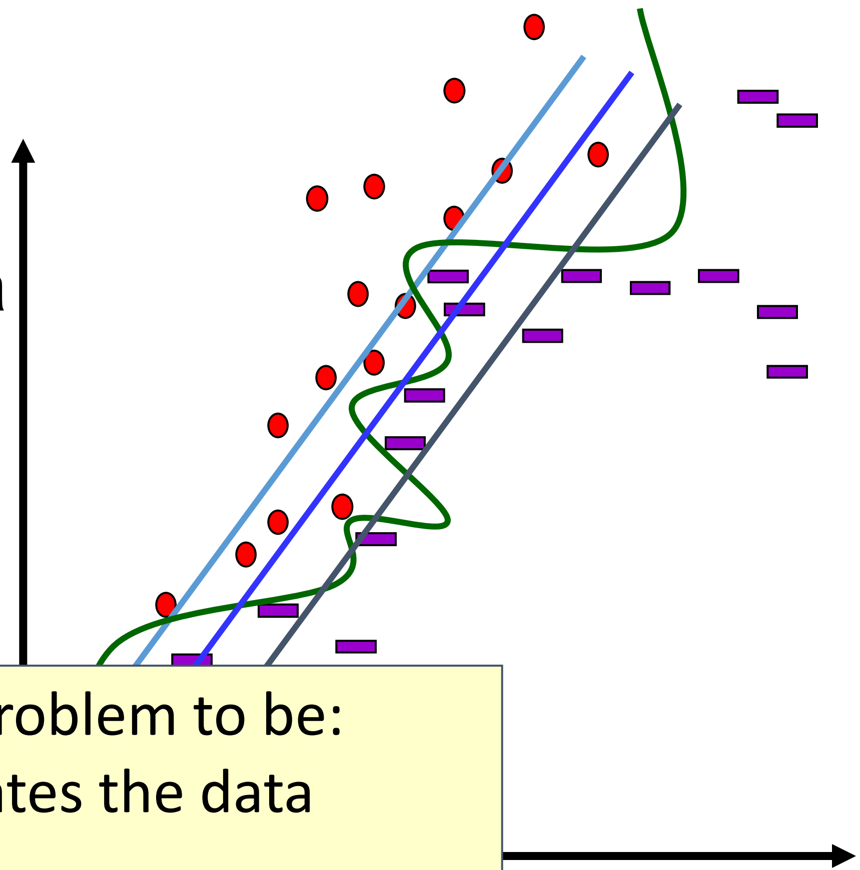
An Example

- This is the modeling step
- What is the hypothesis space?
  - Boolean feature x_w :

    $[x_w = 1]$ iff w is in the sentence
- 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?)



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

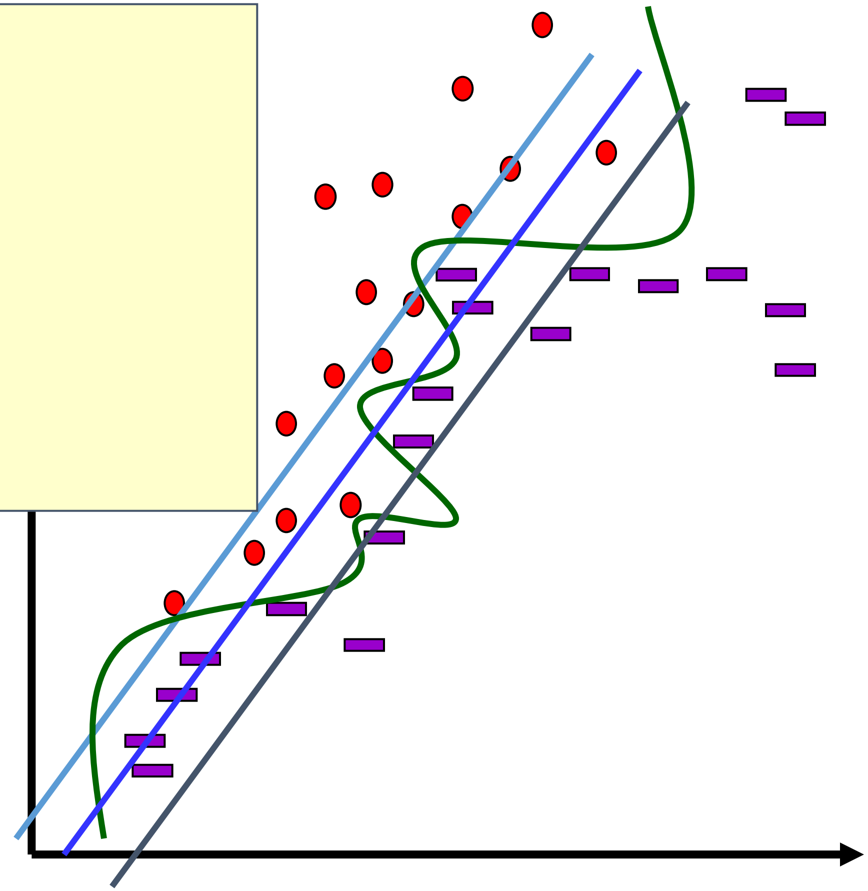
Linear = linear in the feature space

x= data representation; w = the classifier

$$y = \text{sgn}\,\{w^T x\}$$

Representation Step: What's Good?

- 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 – Linear Classifier

Also written as <x, w> or $x^T w$

$$f(x) = \mathrm{sgn}\{x \cdot w - \theta\} = \mathrm{sgn}\{\sum_i w_i x_i - \theta\}$$

- Many functions are Linear
  - Conjunctions:
    - $y = x_1 \land x_3 \land x_5$
    - $y = \mathrm{sgn}\{1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 - 3\}$;
      $w = (1, 0, 1, 0, 1)$ $\theta=3$
  - At least m of n:
    - $y =$ at least 2 of $\{x_1, x_3, x_5\}$
    - $y = \mathrm{sgn}\{1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 - 2\}$ }
      $w = (1, 0, 1, 0, 1)$ $\theta=2$

Expressivity

Also written as <x, w> or $x^T w$

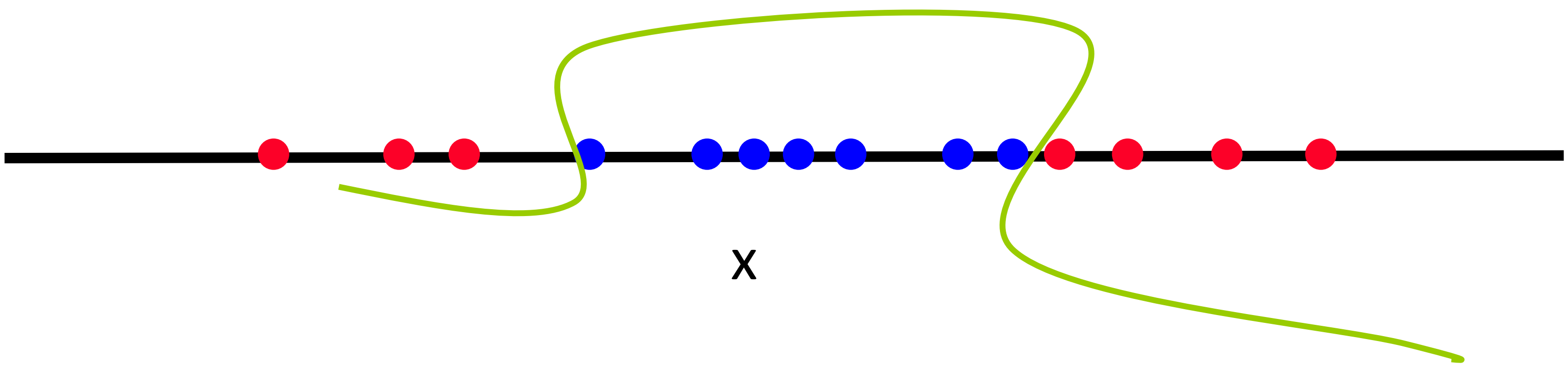
$$f(x) = \text{sgn}\,\{x \cdot w - \theta\} = \text{sgn}\{\textstyle\sum_i w_i x_i - \theta\}$$

- Many functions are not
  - Xor: $y = x_1 \wedge \neg x_2 \vee \neg x_1 \wedge x_2$
  - Non trivial DNF: $y = x_1 \wedge x_2 \vee x_3 \wedge x_4$
- But can be made linear

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

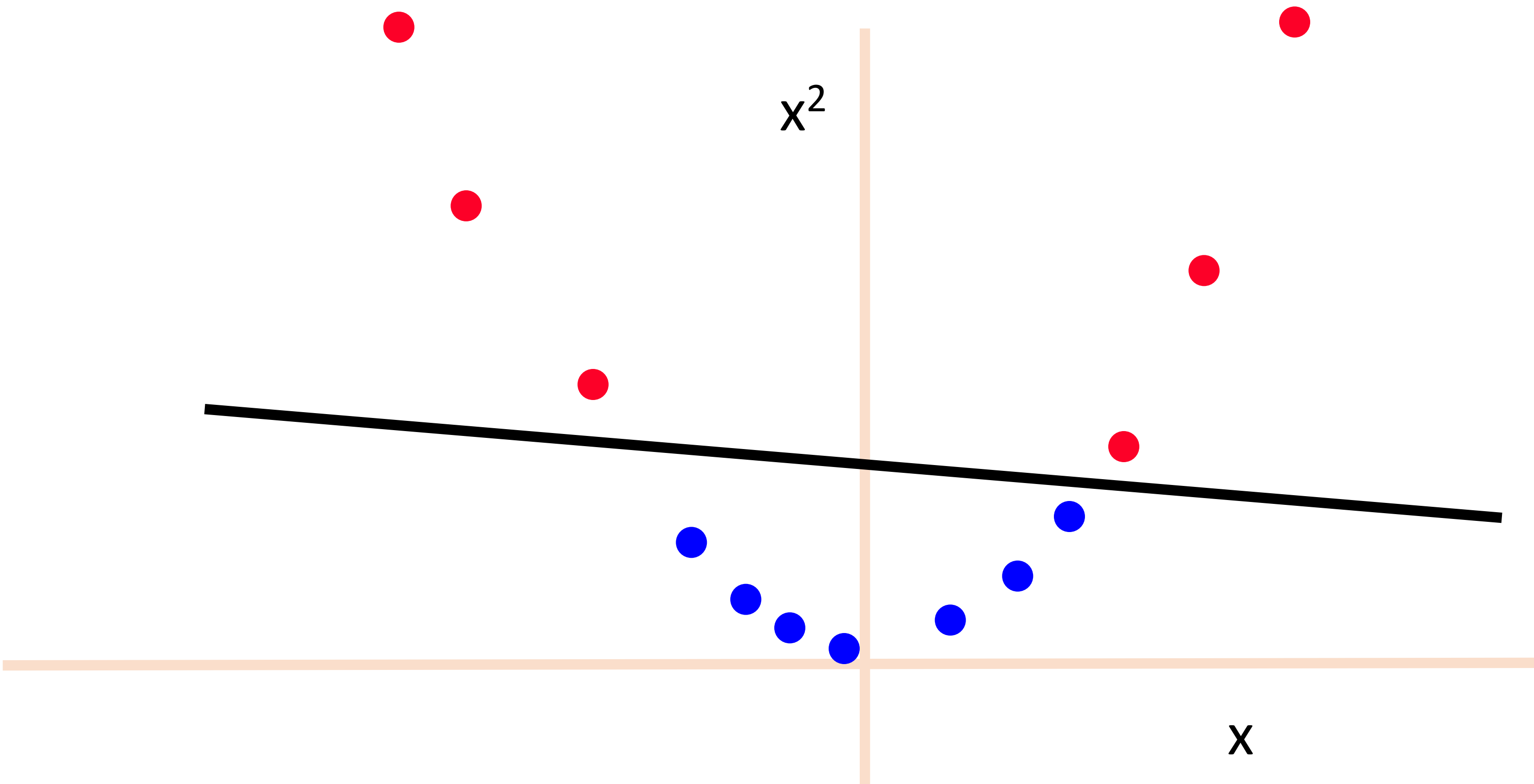
Functions Can be Made Linear

- Data are not linearly separable in one dimension
- Not separable if you insist on using a specific class of functions



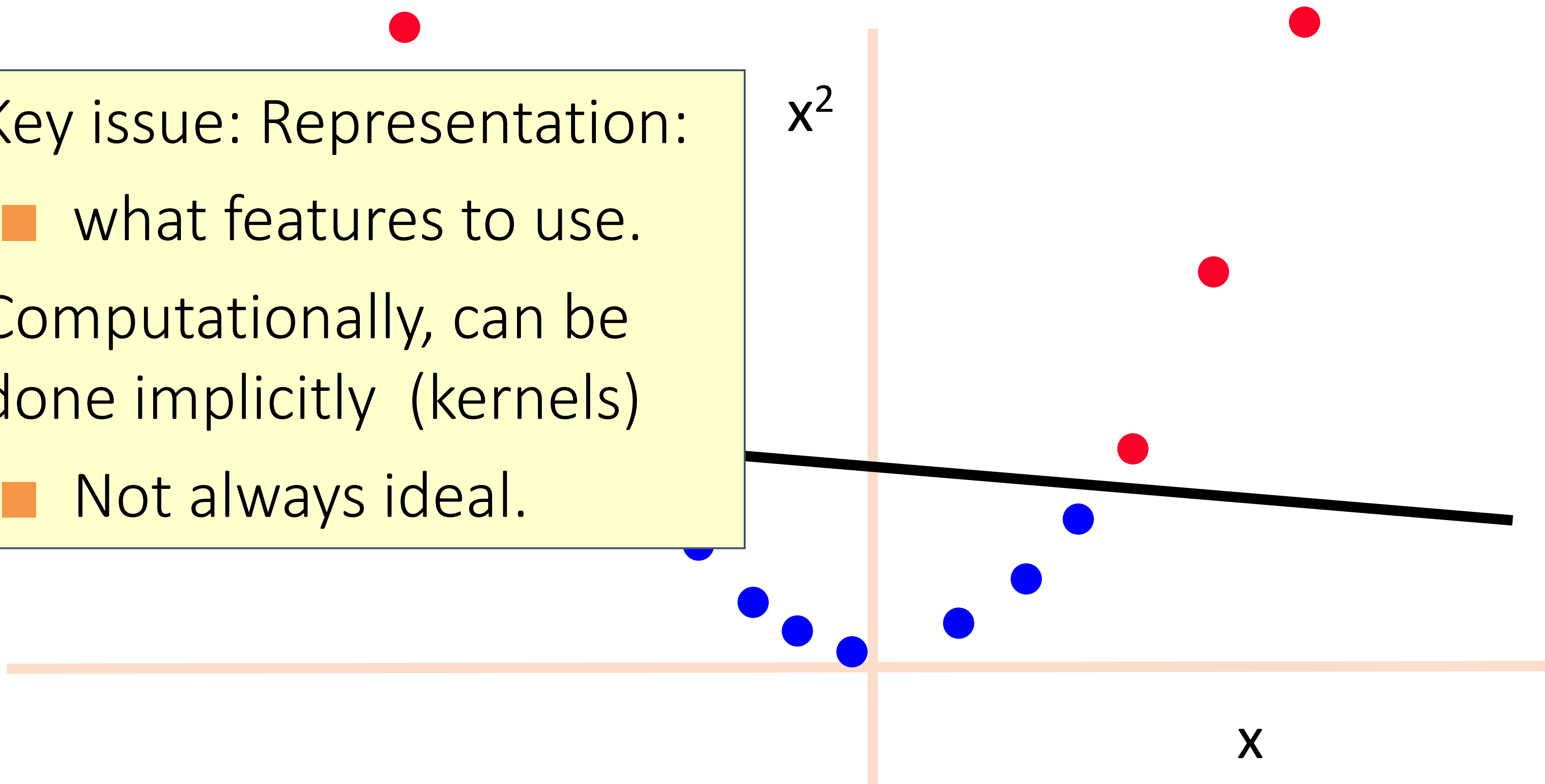
Blown Up Feature Space

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



Blown Up Feature Space

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

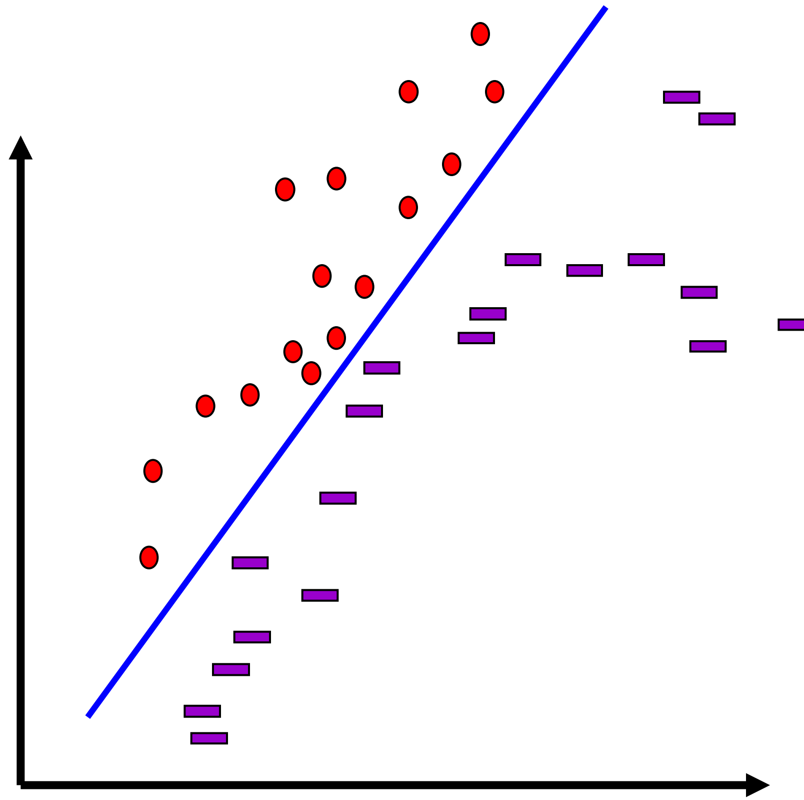


- Key issue: Representation:
  - what features to use.
- Computationally, can be done implicitly (kernels)
  - Not always ideal.

How to learn?

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

General Framework for Learning

Problem Setting

- Set of possible instances X
- Set of possible labels Y
- Unknown target function *f* : X → Y
- Set of function hypotheses *H* = {*h* | *h* : X → Y}

Input: Training instances drawn from data generating distribution *p*

$$\{(x_i, y_i)\}_{i=1}^{n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

Output: Hypothesis *h* in *H* that best approximates *f*

Learning Problem

Output: Hypothesis *h* in *H* that best approximates *f*

h should do well (as measured by the loss) on future instances

Formally, *h* should have low expected (test) loss/Risk

$$\mathbb{E}_{(x,y)\sim p}\left[L(y,h(x))\right] = \sum_{x,y} p(x,y)L(y,h(x))$$

Problem?

We don't know what *p* is

But we are given samples drawn from *p*

Slide credit: Eric Eaton
Based on slide by Tom Mitchell

Learning Problem

We instead approximate the risk by the training error/ empirical risk

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i))$$

When is this reasonable ?

Both the training data **and** the test set are generated based on this distribution

$D = \{x_i, y_i\}_{i=1}^{n}$ is called training data

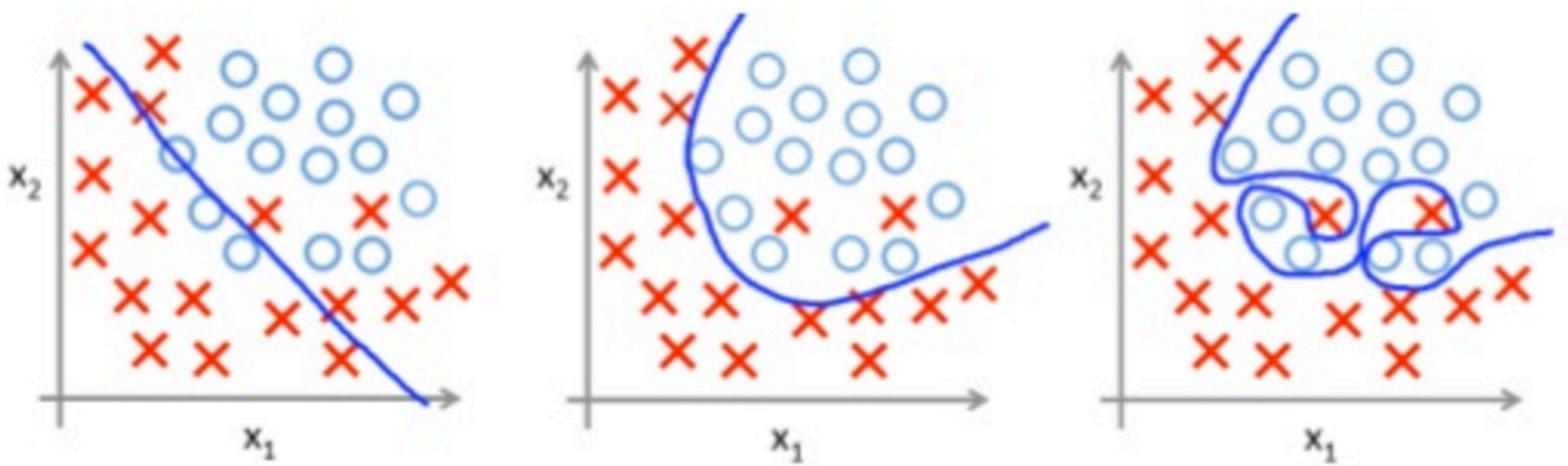
Challenges

1. Can make the training error zero by memorizing if the hypothesis space is expressive.

2. Minimizing. $\frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i))$ is in general a NP-hard problem

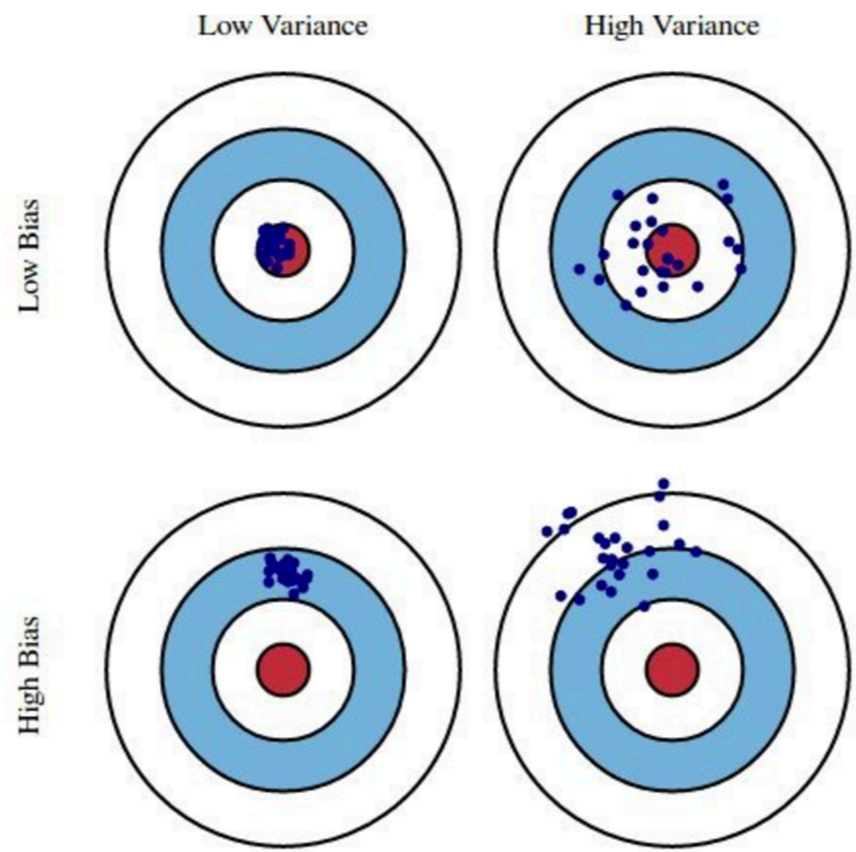
Under-fitting and over-fitting

- Which classifier (blue line) is the best one?



Bias V.S. Variance

- Remember, training data are subsamples drawn from the true distribution
- Exam strategy:
  - Study every chapter well
    - A+: Low var & bias
  - Study only a few chapters
    - A+? B? C? Low bias; High var
  - Study every chapter roughly
    - B+: Low var; high bias
  - Go to sleep
    - B ~D: High var, high bias



Prevent overfitting

- Using a less-expressive model
  - E.g., linear model
- Adding regularization
  - Promote simper models
- Data perturbation
  - Make the model more robust
  - Can be done algorithmically (e.g., dropout)
- Stop the optimization process earlier
  - Sounds bad in theory; but works in practice.

Challenges

1. Can make the training error zero by memorizing if the hypothesis space is expressive.

2. Minimizing. $\frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i))$ is in general a NP-hard problem
   - **To alleviate this computational problem, minimize a new function – a convex upper bound of the classification error function**

Algorithmic View of Learning: an Optimization Problem

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

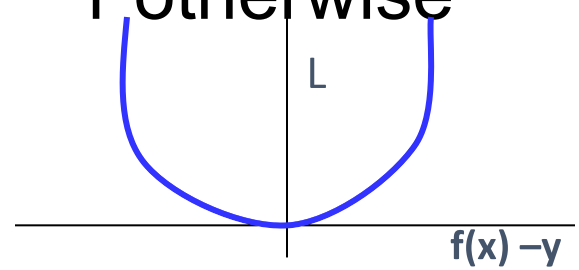
    $$L(h(x),y) = 0 \text{ if } h(x) = y; \qquad 1 \text{ otherwise}$$

  - Squared Loss:

    $$L(h(x),y) = (h(x) - y)^2$$

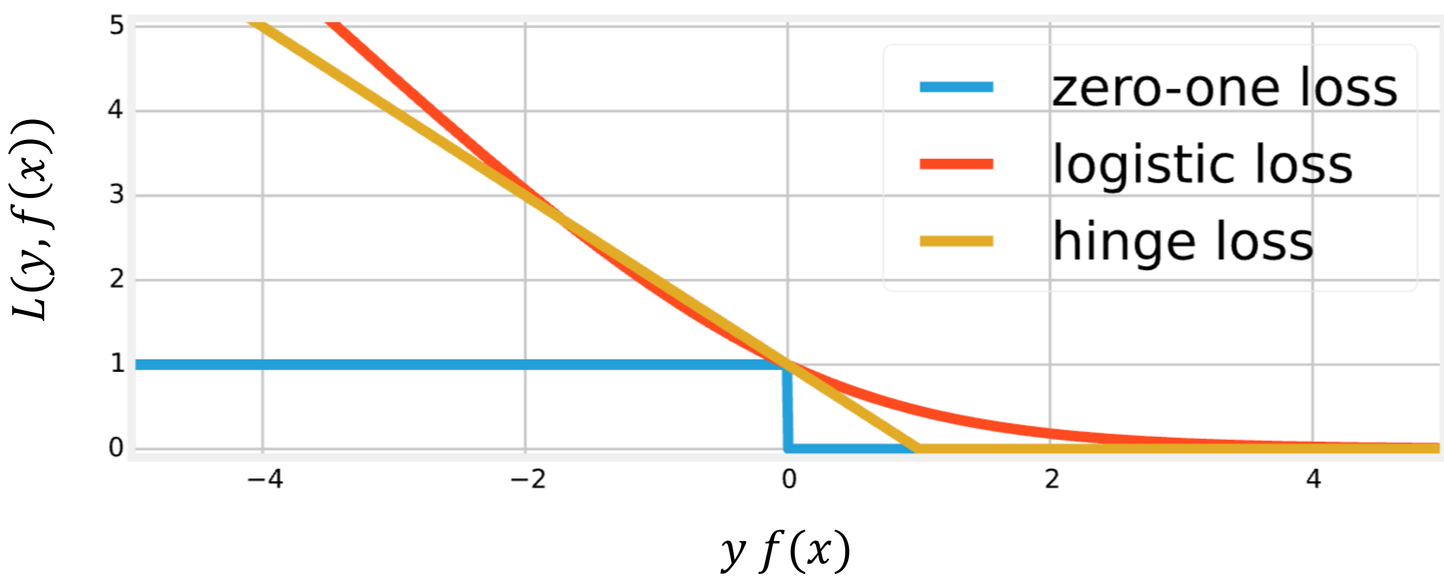
  - Input dependent loss:

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



How about the loss function?

- Usually, we cannot minimize 0-1 loss
  - It is a combinatorial optimization problem: NP-hard
- Idea: minimizing its upper-bound



How about the loss function?

rd

Example

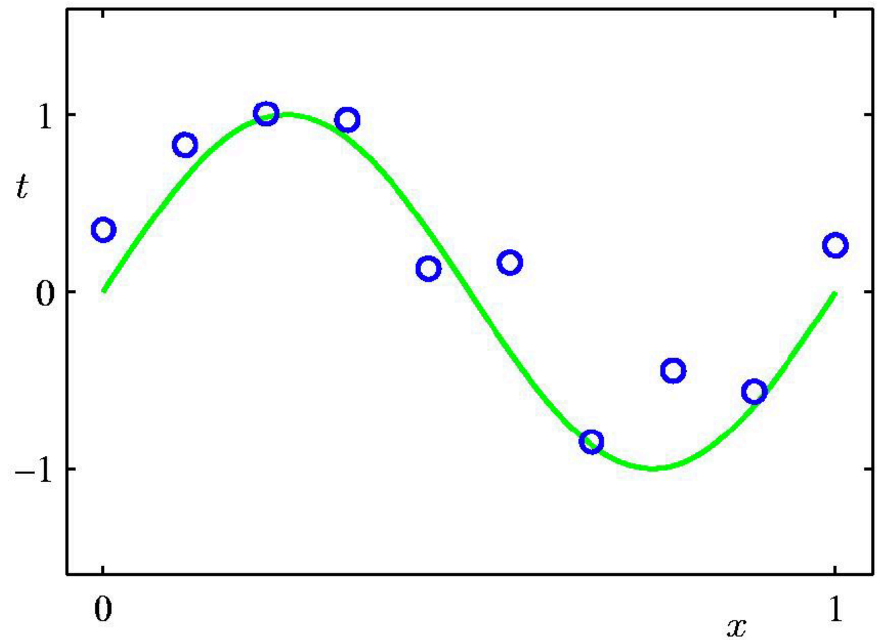
Putting it all together:

A Learning Algorithm

Example: Regression Problem

- Consider simple regression dataset
  - f : X ➔ Y
  - $x \in$ R
  - $y \in$ R
- Question 1: How should we pick the hypothesis space H ?
- Question 2: How do we find the best h in this space?

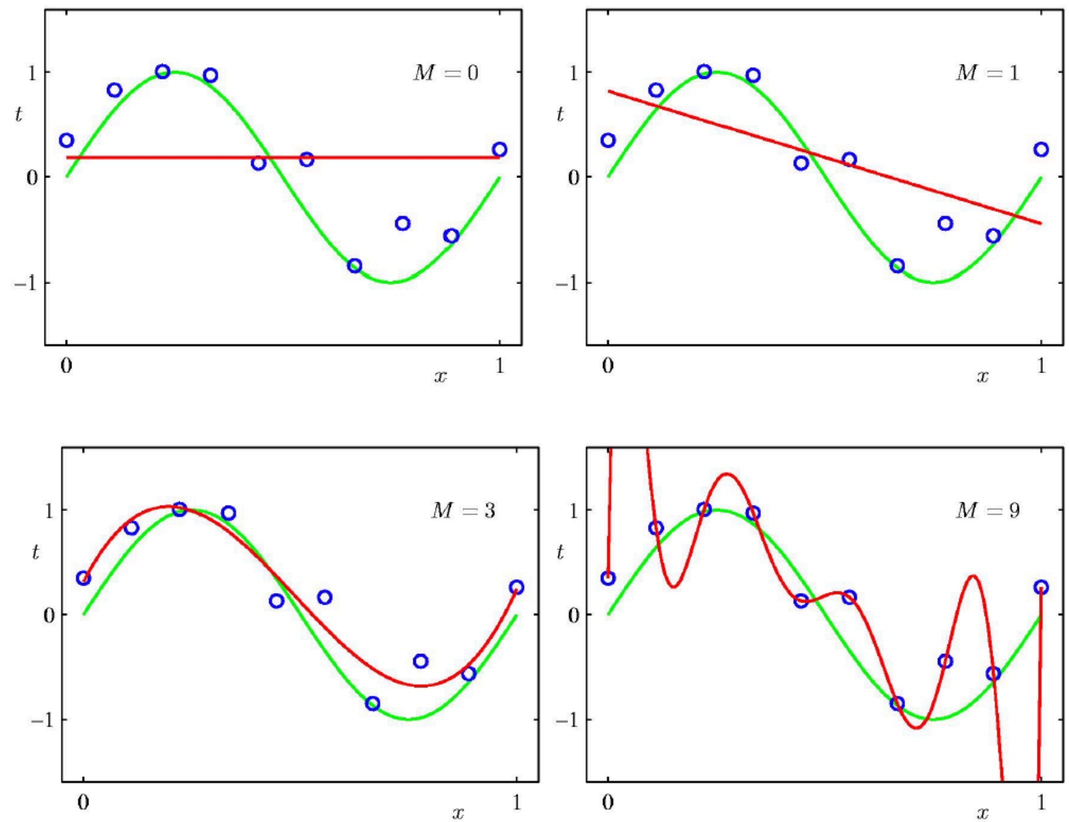
Dataset: 10 points generated from sin function with noise



Based on slide by David Sontag
Images from Bishop [PRML]

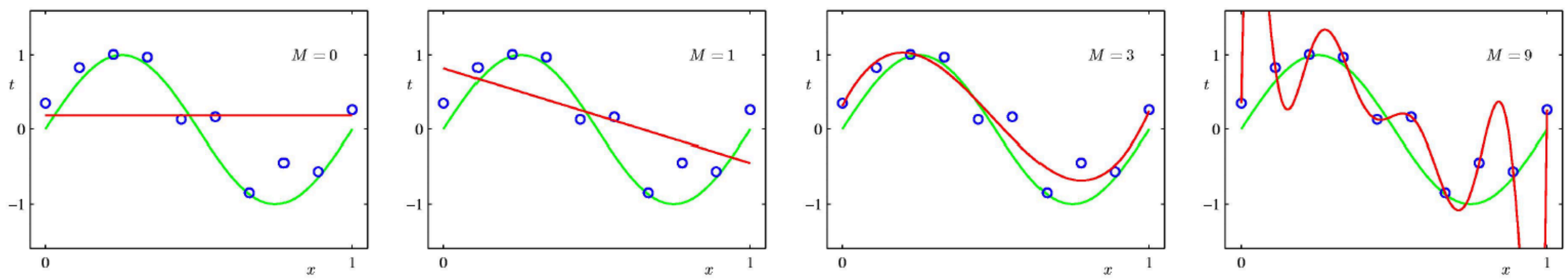
Hypothesis Space: Degree-M Polynomials

- Infinitely many hypotheses
- Which one is **best**?



Based on slide by David Sontag
Images from Bishop [PRML]

Hypothesis Space: Degree-M Polynomials

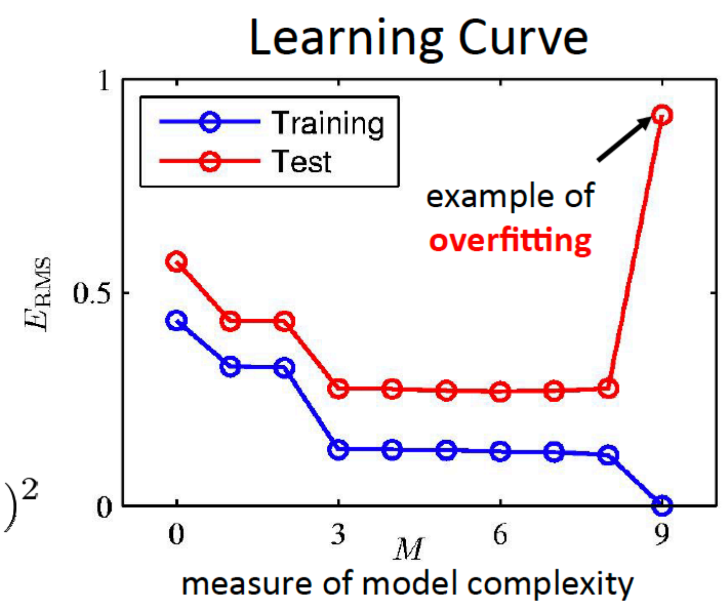


- For regression, common choice is squared loss

$$L(y_i, h(x_i)) = (y_i - h(x_i))^2$$

- *Empirical loss* of function *h* applied to training data is then

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i)) = \frac{1}{n}\sum_{i=1}^{n} (y_i - h(x_i))^2$$



Based on slide by David Sontag
Images from Bishop [PRML]