

Lecture 1:

Introduction to Machine Learning

Fall 2020

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.

Registration

- ❖ Please fill the following form for PTE:
<https://forms.gle/XGjnQww7x97RjLBo8>
- ❖ Limited by the resources
 - ❖ Unlikely we can give more PTEs
- ❖ Please drop the course if you're not planning to take it
- ❖ PTE will be given by the end of the first week
 - ❖ Check <http://my.ucla.edu>

Homework 0

- ❖ Sign up at Piazza

<http://piazza.com/ucla/fall2021/m146>

CS M146 Teaching Team

- ❖ Kai-Wei Chang
 - ❖ Mon, 3:00 PM – 4:00 PM
- ❖ TAs

Section: Location / Hours	
Teaching Assistant	(All times are in PT)
MENG, TAO	1A: LAKRETZ 110 / Friday / 2:00pm-3:50pm
WU, TE-LIN	1C: KAPLAN 169 / Friday / 4:00pm-5:50pm
YIN, DA	1B: KNSY PV 1200B / Friday / 2:00pm-3:50pm
YIN, FAN	1D: ONLINE / Friday / 4:00pm-5:50pm

CS M146 Teaching Team

- ❖ Kai-Wei Chang
 - ❖ Mon, 3:00 PM – 4:00 PM
- ❖ TAs

Section: Location / Hours	
Teaching Assistant	(All times are in PT)
MENG, TAO	1A: LAKRETZ 110 / Friday / 2:00pm-3:50pm
WU, TE-LIN	1C: KAPLAN 169 / Friday / 4:00pm-5:50pm
YIN, DA	1B: KNSY PV 1200B / Friday / 2:00pm-3:50pm
YIN, FAN	1D: ONLINE / Friday / 4:00pm-5:50pm

What is machine learning?

Machine Learning is the study of algorithms that

- improve their performance P
- at some task T
- with experience E .

A well-defined learning task is given by $\langle P, T, E \rangle$.

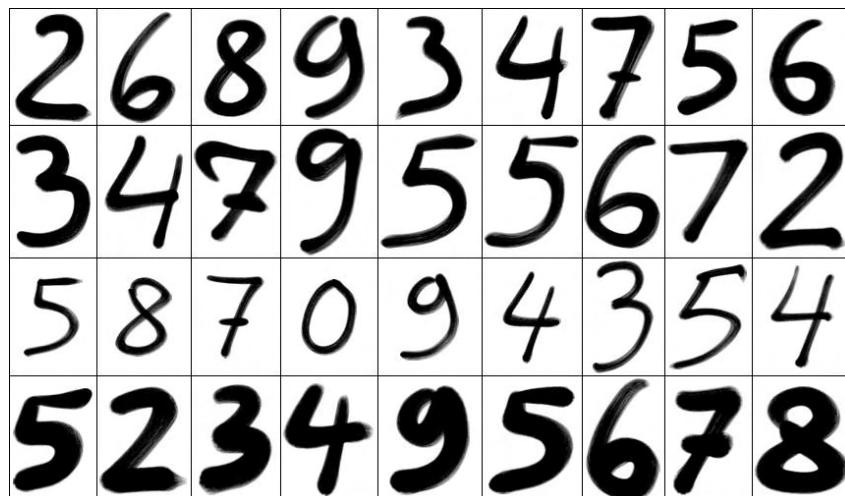
[Definition by Tom Mitchell (1998)]

Improve on task T with respect to performance P, based on experience E

T: Recognizing hand-written words

P: Percentage of words correctly classified

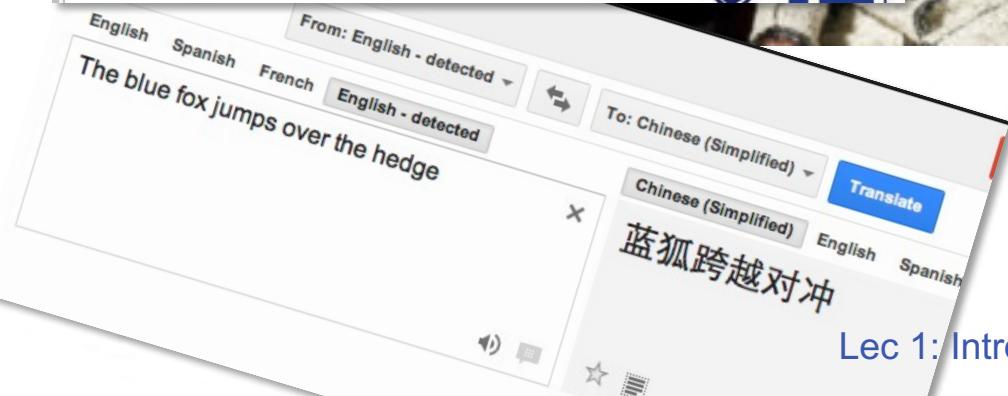
E: Database of human-labeled images of hand written words



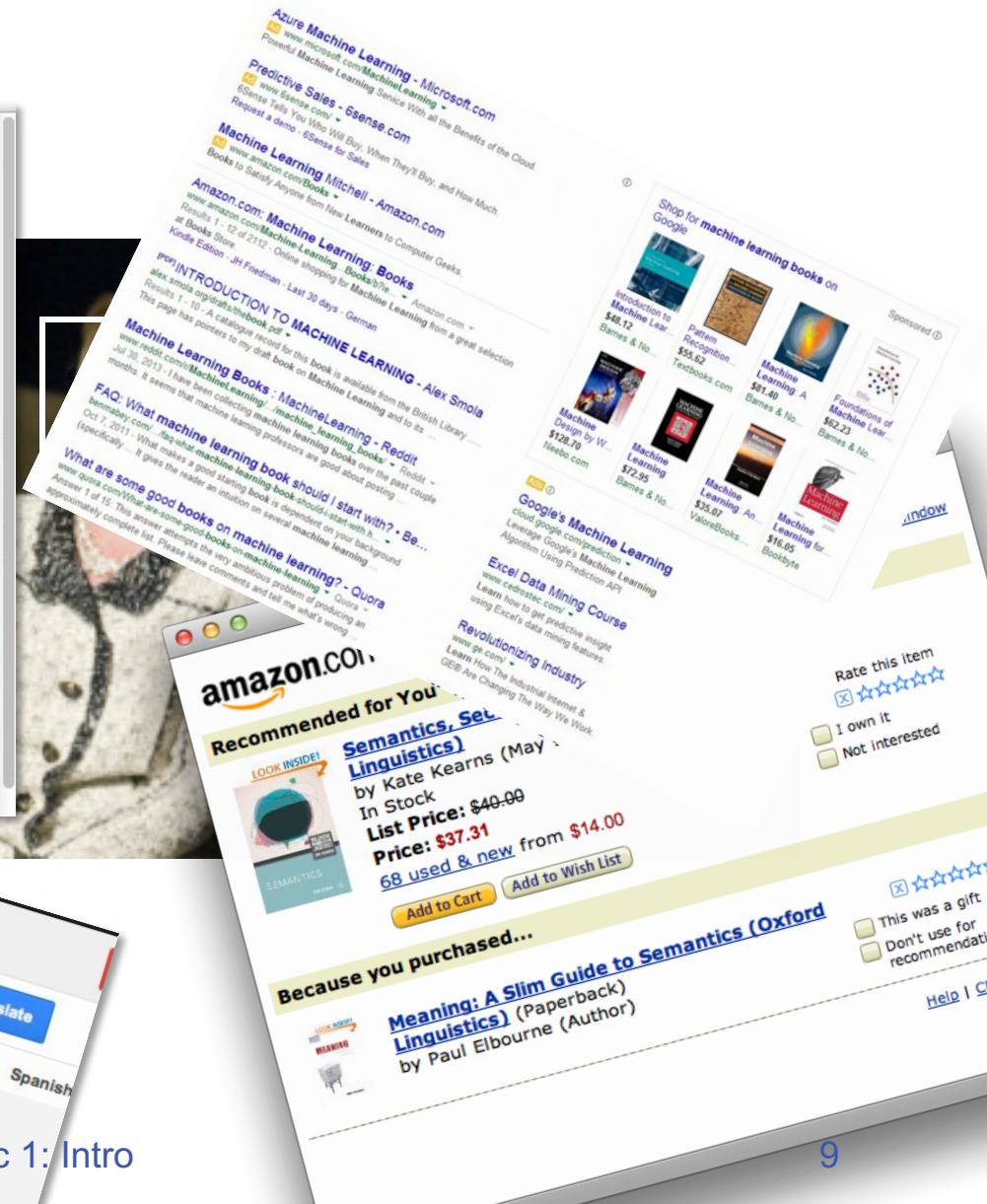
Improve on task T with respect to performance P, based on experience E



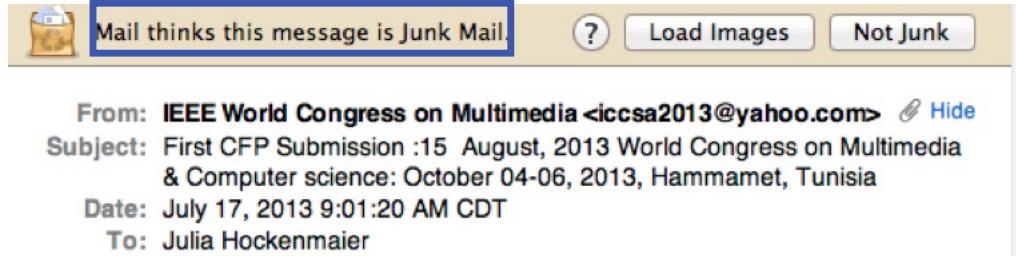
Machine learning is everywhere



Lec 1: Intro



Applications: Spam Detection



- ❖ This is a **binary classification task**:
Assign one of two labels (i.e. yes/no) to the input (here, an email message)
- ❖ Classification requires a **model (a classifier)** to determine which label to assign to items.
- ❖ In this class, we study **algorithms and techniques** to learn such models from data.

The Uses of Machine Learning

- ❖ **ML is at the core of teaching machine to**
 - ❖ Understand high level cognition (e.g., vision)



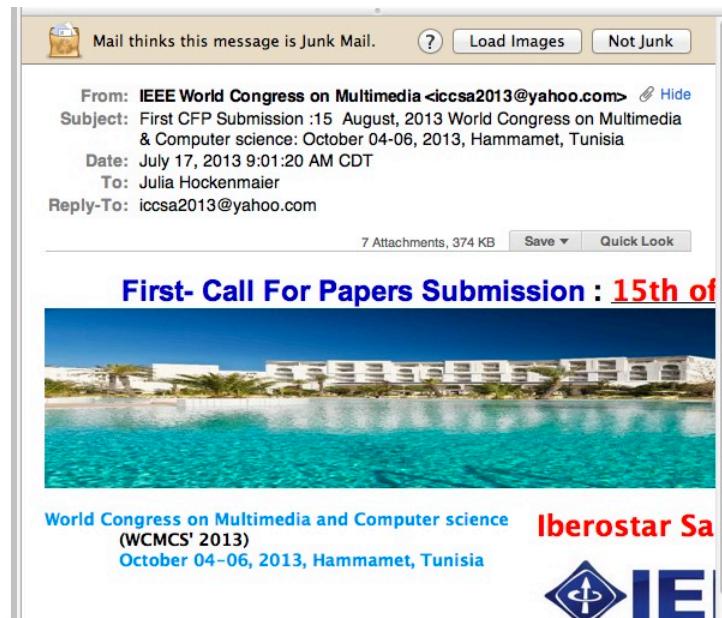
The Uses of Machine Learning

- ❖ **ML is at the core of teaching machine to**
 - ❖ Understand high level cognition
 - ❖ Perform knowledge intensive inferences



The Uses of Machine Learning

- ❖ **ML is at the core of teaching machine to**
 - ❖ Understand high level cognition
 - ❖ Perform knowledge intensive inferences
 - ❖ Deal with messy, real world data



Learning = Generalization

H. Simon (Turing Award 1975, Nobel Prize 1978)-

“Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the task or tasks drawn from the same population more efficiently and more effectively the next time.”

The ability to perform a task in a situation which has never been encountered before

Learning = Generalization



Mail thinks this message is junk
mail.

Not junk

- ❖ The learner has to be able to **classify** items it has never seen before.

Why Machine Learning?

- ❖ We learned how to write programs to automate
- ❖ Need computer systems with new capabilities to

Why Machine Learning?

- ❖ We learned how to write programs to automate
- ❖ Need computer systems with new capabilities to
 - ❖ develop systems that are too difficult or impossible to construct manually

Why Machine Learning?

- ❖ We learned how to write programs to automate
- ❖ Need computer systems with new capabilities to
 - ❖ develop systems that are too difficult or impossible to construct manually
 - ❖ develop systems that can automatically adapt and customize themselves

Why Machine Learning?

- ❖ We learned how to write programs to automate
- ❖ Need computer systems with new capabilities to
 - ❖ develop systems that are too difficult or impossible to construct manually
 - ❖ develop systems that can automatically adapt and customize themselves
 - ❖ discover knowledge and patterns in databases,
e.g. discovering purchasing patterns

Administrivia

CS M146 Team

❖ Kai-Wei Chang (Eng VI 374)

❖ Wed, 3:00 PM – 4:00 PM

❖ TAs

CHEN, ZEYUAN	1B: ONLINE / Friday / 10:00am-11:50am	zchen05@ucla.edu	280 055 4233	M 3:00PM - 4:00PM; F 9:00AM - 10:00AM;
DARABI, SAJAD	1A: ONLINE / Friday / 10:00am-11:50am	sajad10@ucla.edu	562 276 6807	W 8:00AM - 9:00AM; F 8:30AM - 9:30AM;
HUANG, KUAN-HAO	1D: ONLINE / Friday / 12:00pm-1:50pm	khuang@cs.ucla.edu	368 123 4567	W 11am-1pm
MENG, TAO	1C: ONLINE / Friday / 12:00pm-1:50pm	mengt18@ucla.edu		
PARVEZ, MD RIZWAN	1F: ONLINE / Friday / 2:00pm-3:50pm	rizwan@ucla.edu	553 138 7116	M 8:00AM-10:00AM
SRINIVASAN, AAKASH	1E: ONLINE / Friday / 2:00pm-3:50pm	s.aakash3431@gmail.com	957 701 9901	F 4:00PM-6:00PM

Prerequisites

- ❖ The pillars of machine learning
 - ❖ Probability and statistics
 - ❖ Linear algebra
 - ❖ Calculus/Optimization

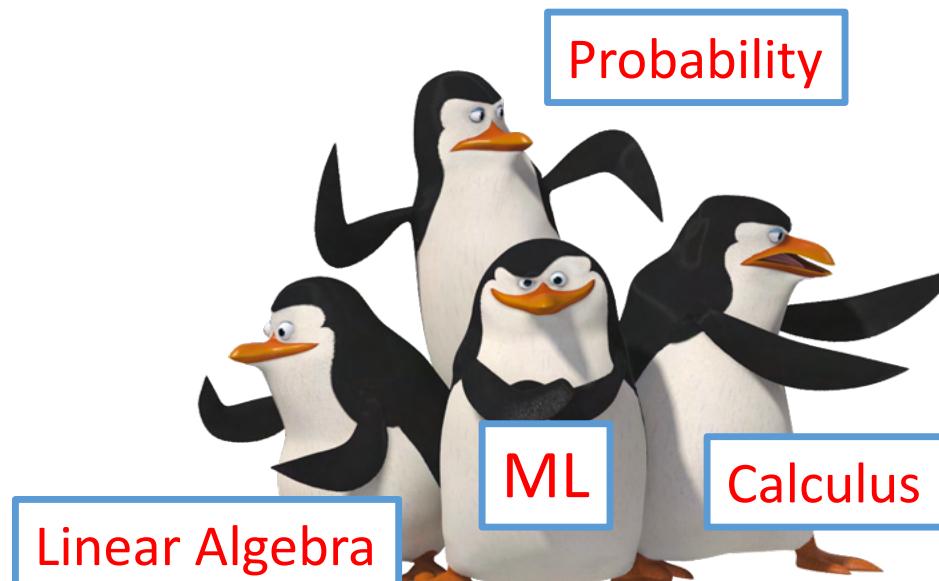


Image adapted from <http://pngimg.com/download/31173>
under CC by-NC 4.0

Prerequisites

- ❖ The pillars of machine learning
 - ❖ Probability and statistics
 - ❖ Linear algebra
 - ❖ Calculus/Optimization
 - ❖ Computer science background
 - ❖ Algorithms
 - ❖ Programming experience
- We will use **Python and scikit-learn**

Math background

- ❖ Mini quiz on math background
(in CCLE, Due end of 2nd week)

Online Lecture

- ❖ Turn your video on if possible
- ❖ Participate in the break-out discussion and online poll
- ❖ Ask questions in chat

Problem Set

- ❖ Problem Sets
 - ❖ Four problem sets (1 -- 4)
 - ❖ Due at 11:59pm on the due date
 - ❖ 24hr late credits for the entire quarter
 - ❖ Will be using GradeScope to manage submissions
(submission instructions will be provided in the discussion session)
- ❖ All solutions must be clearly written or typed.
 - ❖ Unreadable answers will not be graded. We encourage using LaTeX to type answers.
 - ❖ Solutions will be graded on both correctness and clarity
- ❖ For programming HW, upload your source code at CCLE

Exams

- ❖ Final is a **3hr take-home** within 24 hour period
- ❖ Exam will cover materials from **all the lectures** and the problem sets.
- ❖ No alternate or make-up exams
 - ❖ Except for disability/medical/emergency reasons documented and communicated to the instructor prior to the exam date.
 - ❖ Exam date and time **cannot** be changed to accommodate scheduling conflicts with other classes or job fair/interview.

Quiz

- ❖ We will have quizzes (almost) every week after week 2
- ❖ A handful multiple choice questions
- ❖ You have 2 tries for each quiz– the highest one will be counted
- ❖ One lowest quiz score will be dropped

Policies

- ❖ Attendance and class participation
 - ❖ Although not a formal component of the grade,
Attendance is important
 - ❖ We look forward to your active participation
- ❖ Video recordings
 - ❖ we aim to make it available within 24 hours (not guarantee).
 - ❖ You should not rely on these recordings as a substitute for lectures.

Policies

- ❖ Regrading request
 - ❖ Must be made **within one week** after the grade is released regardless of any reason
 - ❖ We reserve the right to regrade entire problem set for a given regrade request.

Final grade

- ❖ Default cut-off for letter grade is:

> 97	93	90	87	83	80	77	73	70	< 70
A +	A	A -	B +	B	B -	C +	C	C -	D

- ❖ We **will not** make adjustments for individuals
 - ❖ E.g., no round up (i.e., 89.99 = B+)
- ❖ In general, we **will not** curve the final grades, but may do some adjustment
 - ❖ The cut-off score will only get lower (i.e., you may get a better letter grade)
- ❖ This is a **heavy** course
- ❖ S/U options per department policy



Academic integrity policy

- ❖ **No cheating**
 - ❖ Homework and Exam
 - ❖ In particular, you are free to discuss homework problems. However, you **must** write up your own solutions (solution/program). You **must** also acknowledge all collaborators.
 - ❖ Please don't use any old solution you found.
 - ❖ **Don't post your HW/Exam solutions or upload course materials without consent**
 - ❖ All incidents will report to the student office

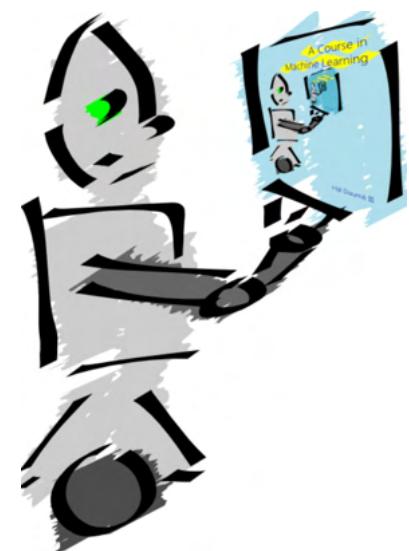
CM146 on Web



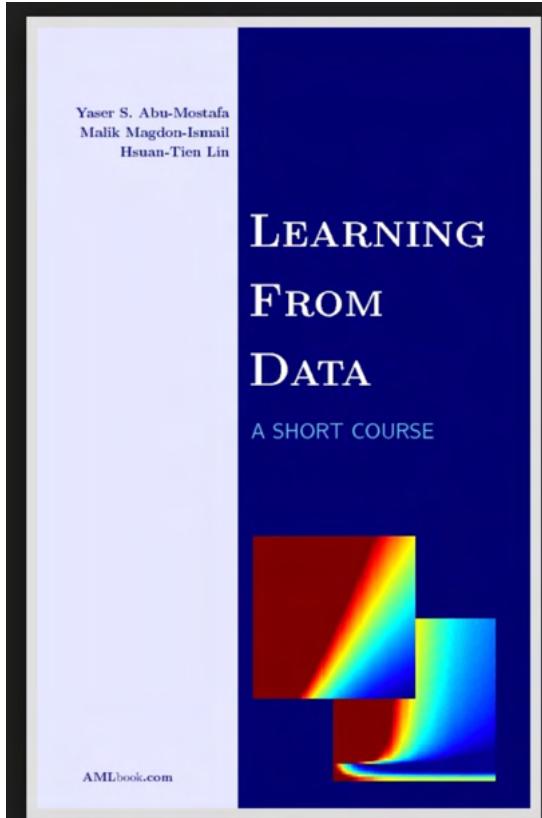
- ❖ Course website:
<https://ccle.ucla.edu/course/view/21F-COMSCIM146-1>
- ❖ Piazza:
<http://piazza.com/ucla/fall2021/m146>
 - ❖ Strongly encourage students to post here (publicly or privately) rather than email staff directly (you will get a faster response this way)
- ❖ Gradescope
 - ❖ Maintain homework/final
 - ❖ Access through CCLE

Textbook

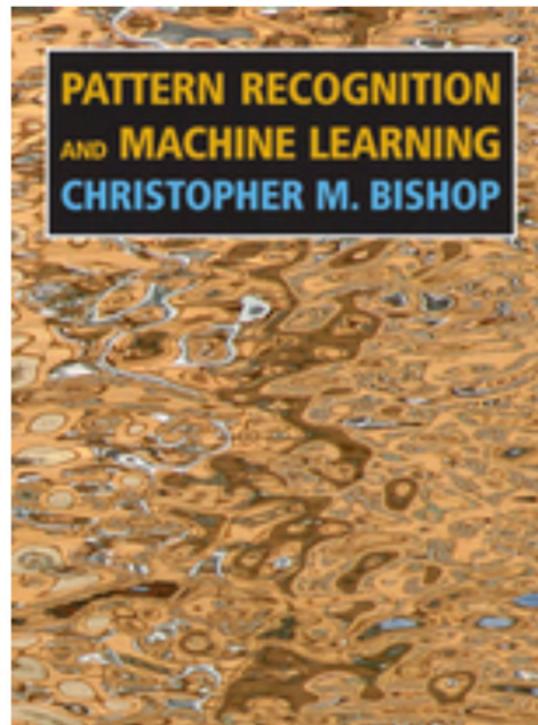
- ❖ No textbook
 - ❖ Primary reference:
A course in machine learning by Hal Daume III
(CIML). Freely available online <http://ciml.info/>
 - ❖ See syllabus for the reading list



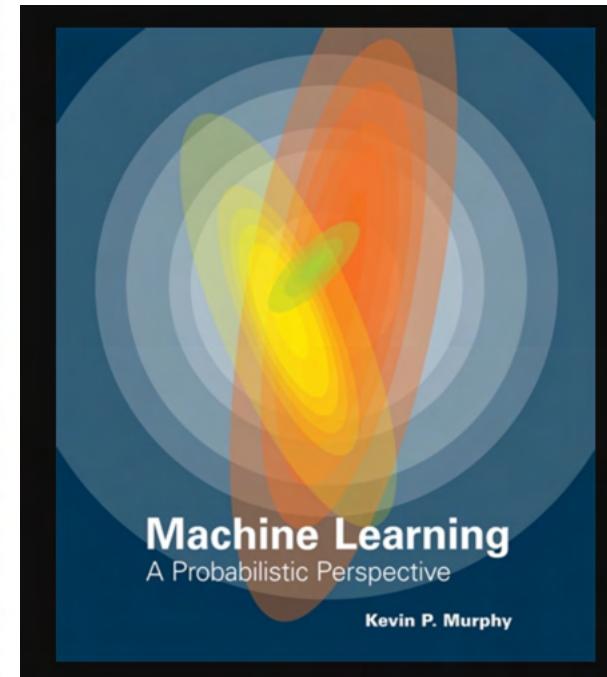
Other references



Basic



Comprehensive



Advanced

Why taking this course?

Building fundamental knowledge

A Regularized Framework for Sparse and Structured Neural Attention

Vlad Niculae*
Cornell University
Ithaca, NY
vlad@cs.cornell.edu

Mathieu Blondel
NTT Communication Science Laboratories
Kyoto, Japan
mathieu@mblondel.org

Abstract

Modern neural networks are often augmented with an attention mechanism, which tells the network where to focus within the input. We propose in this paper a new framework for sparse and structured attention, building upon a smoothed max operator. We show that the gradient of this operator defines a mapping from real values to probabilities, suitable as an attention mechanism. Our framework includes softmax and a slight generalization of the recently-proposed sparsemax as special cases. However, we also show how our framework can incorporate modern structured penalties, resulting in more interpretable attention mechanisms, that focus on entire segments or groups of an input. We derive efficient algorithms to compute the forward and backward passes of our attention mechanisms, enabling their use in a neural network trained with backpropagation. To showcase their potential as a drop-in replacement for existing ones, we evaluate our attention mechanisms on three large-scale tasks: textual entailment, machine translation, and sentence summarization. Our attention mechanisms improve interpretability without sacrificing performance; notably, on textual entailment and summarization, we outperform the standard attention mechanisms based on softmax and sparsemax.

1 Introduction

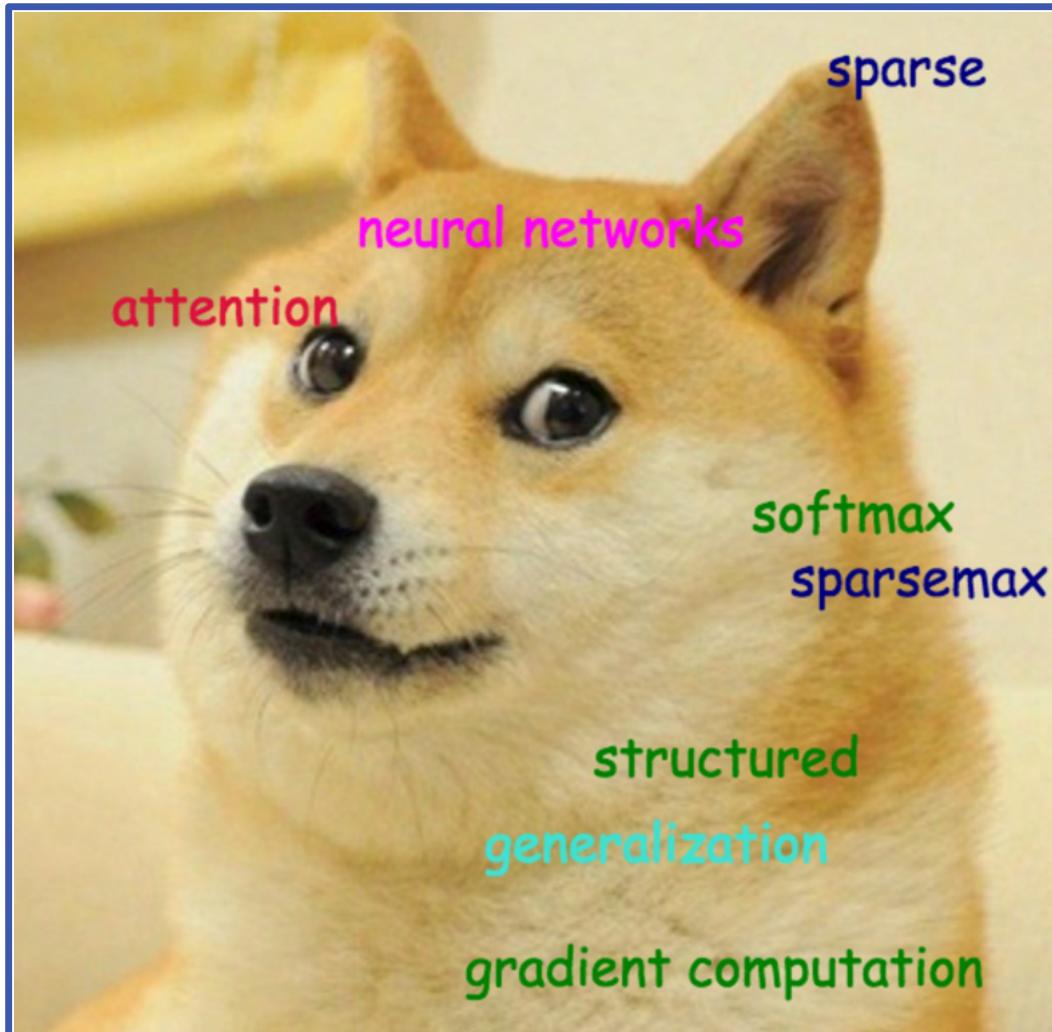
Modern neural network architectures are commonly augmented with an attention mechanism, which tells the network where to look within the input in order to make the next prediction. Attention-augmented architectures have been successfully applied to machine translation [2, 29], speech recognition [10], image caption generation [44], textual entailment [38, 31], and sentence summarization [39], to name but a few examples. At the heart of attention mechanisms is a mapping function that converts real values to probabilities, encoding the relative importance of elements in the input. For the case of sequence-to-sequence prediction, at each time step of generating the output sequence, attention probabilities are produced, conditioned on the current state of a decoder network. They are then used to aggregate an input representation (a variable-length list of vectors) into a single vector, which is relevant for the current time step. That vector is finally fed into the decoder network to produce the next element in the output sequence. This process is repeated until the end-of-sequence symbol is generated. Importantly, such architectures can be trained end-to-end using backpropagation.

Alongside empirical successes, neural attention—while not necessarily correlated with human attention—is increasingly crucial in bringing more **interpretability** to neural networks by helping explain how individual input elements contribute to the model’s decisions. However, the most commonly used attention mechanism, *softmax*, yields dense attention weights: all elements in the input always make at least a small contribution to the decision. To overcome this limitation, *sparsemax* was recently proposed [31], using the Euclidean projection onto the simplex as a sparse alternative to

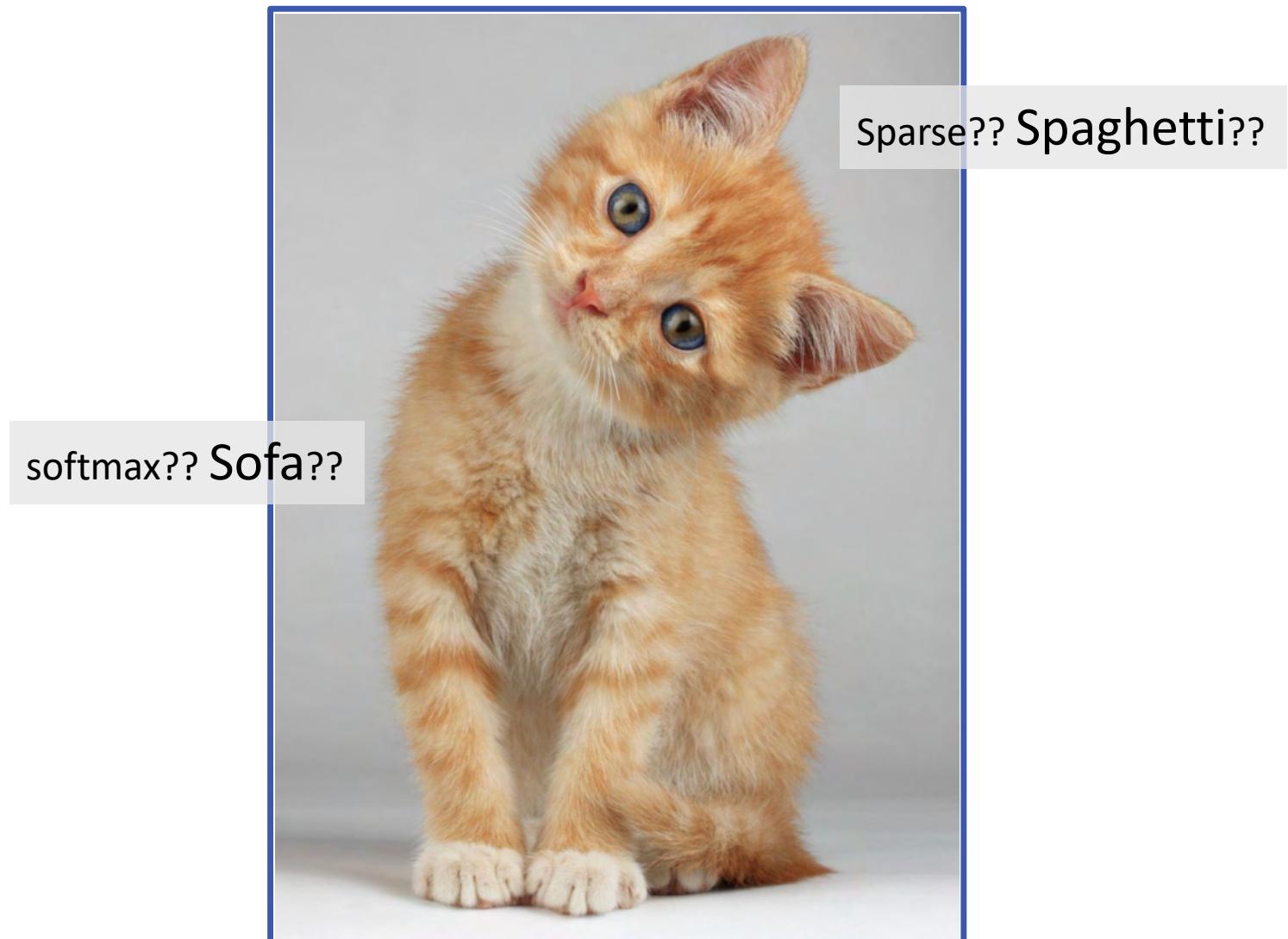
Modern **neural networks** **attention mechanism**, ... We propose in this paper a new framework for **sparse** and **structured** attention, building upon a **smoothed max operator**. We show that the **gradient** of this operator defines a mapping from real values to **probabilities**, suitable as an attention mechanism. Our framework includes **softmax** and a slight **generalization** of the recently-proposed **sparsemax** as **special cases**.

*Work performed during an internship at NTT Communication Science Laboratories, Kyoto, Japan.

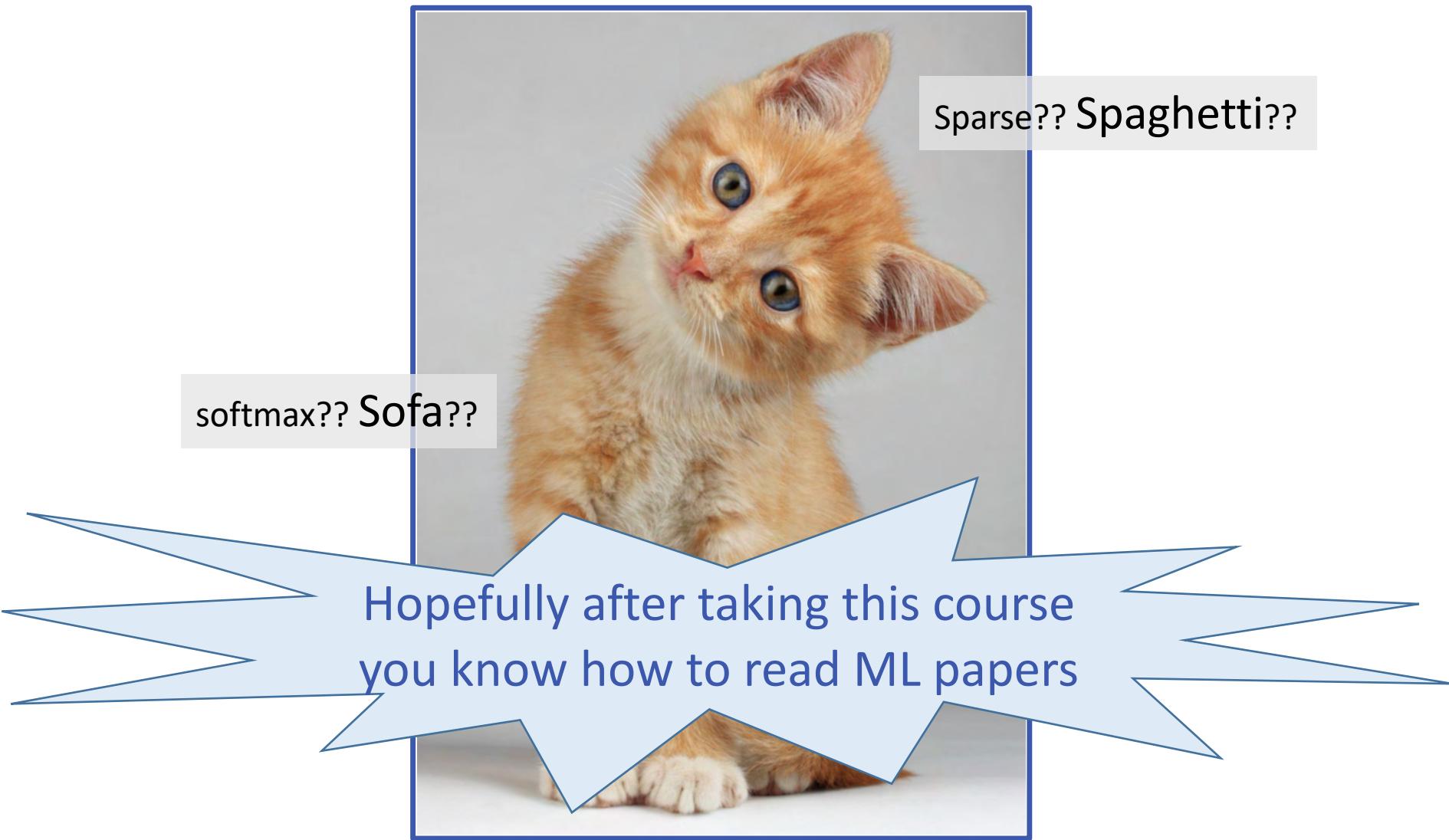
What it looks like to ML researchers



What it looks like to normal people



What it looks like to normal people



What ML beginners may do

How to train an
image classifier?

Deep Learning!

How to train a
model on a small
dataset

Deep Learning!

How to sort *five*
numbers

Deep Learning!

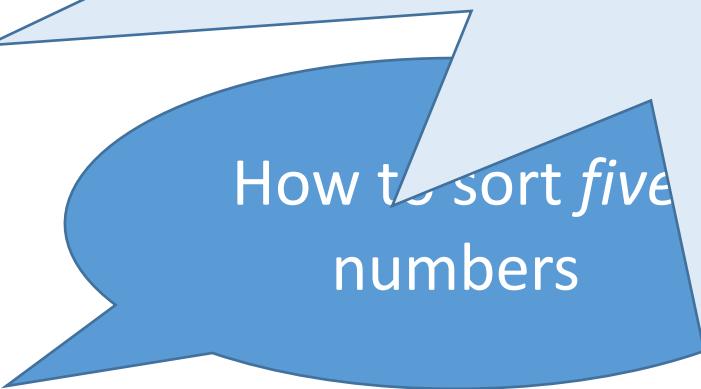


What does ML beginners may do

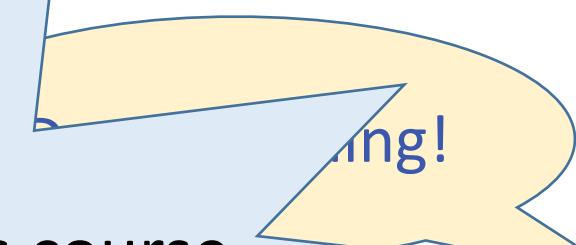


How to train an
image classifier?

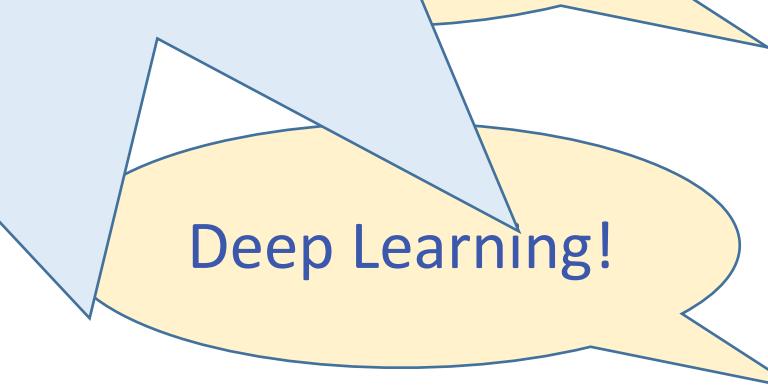
Hopefully after taking this course
you know how to choose a **right**
model for your application
(theoretically or empirically).



How to sort five
numbers



Deep Learning!



Deep Learning!

Goals of this course: Learn about

- ❖ Fundamental concepts and algorithms
 - ❖ Customize your own algorithm
- ❖ Common techniques/tools used
 - ❖ theoretical understanding
 - ❖ practical implementation
 - ❖ best practices
- ❖ How to "debug" ML system
- ❖ Black magic => systematic process

Why Study Learning Now?

- ❖ Exciting moments for ML:
 - ❖ Initial **algorithms** and **theory** in place.
 - ❖ Growing amounts of on-line data
 - ❖ Computational power available.

What will we learn?

- ❖ Supervised learning
 - ❖ Decision tree, Perceptron, Linear models, support vector machines, kernel methods, probabilistic models
- ❖ Unsupervised learning
 - ❖ Clustering,
 - ❖ EM algorithms
- ❖ Learning theory
- ❖ Deep learning (representation learning)
- ❖ Practical Issues
 - ❖ Experimental evaluation; Implementing ML models

Machine Learning is Interdisciplinary

- ❖ Makes Use of:
 - ❖ Probability and Statistics; Linear Algebra; Calculus; Theory of Computation;
- ❖ Related to:
 - ❖ Philosophy, Psychology ,Neurobiology, Linguistics, Vision, Robotics,....
- ❖ Has applications in:
 - ❖ AI (Natural Language; Vision; Planning; HCI)
 - ❖ Engineering (Agriculture; Civil; ...)
 - ❖ Computer Science (Compilers; Architecture; Systems; data bases...)

Other Related Courses

- ❖ CS145 - Introduction to Data Mining
- ❖ CM148 - Introduction to Data Science
- ❖ CS161 - Fundamentals of Artificial Intelligence
- ❖ Computer Vision/Bioinformatic/NLP
- ❖ Related course at ECE, Stats, Math dep.
- ❖ Graduate-level courses

State of the art applications of ML

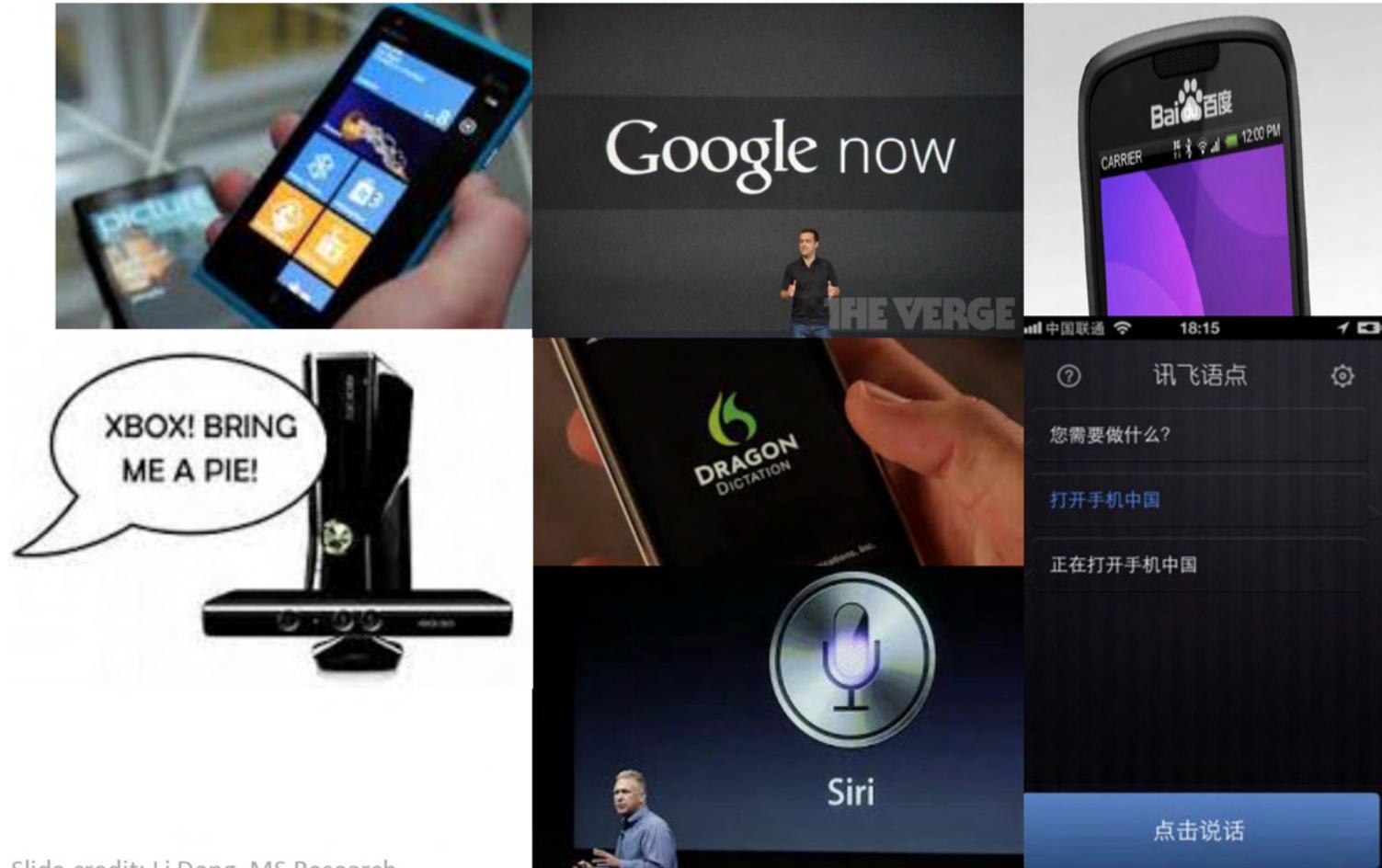
State of the art applications of ML

Computer Vision



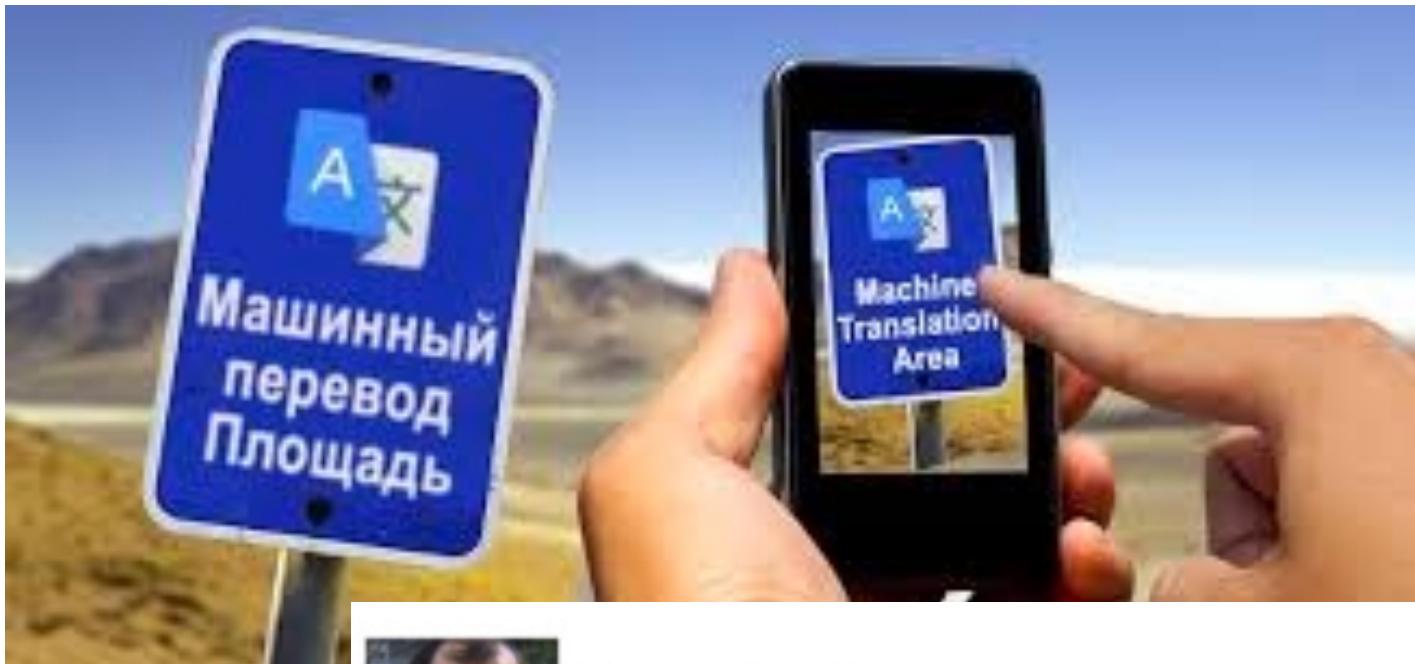
State of the art applications of ML

Speech Recognition



Slide credit: Li Deng, MS Research

Machine translation



Necip Fazil Ayan

1 hr •



Onların, İzmir'in neden hayır dediğini anlamalarını beklemiyoruz.

We don't expect them to understand why Izmir said no.



• Rate this translation

State of the art applications of ML

❖ Reinforcement Learning



Discussion

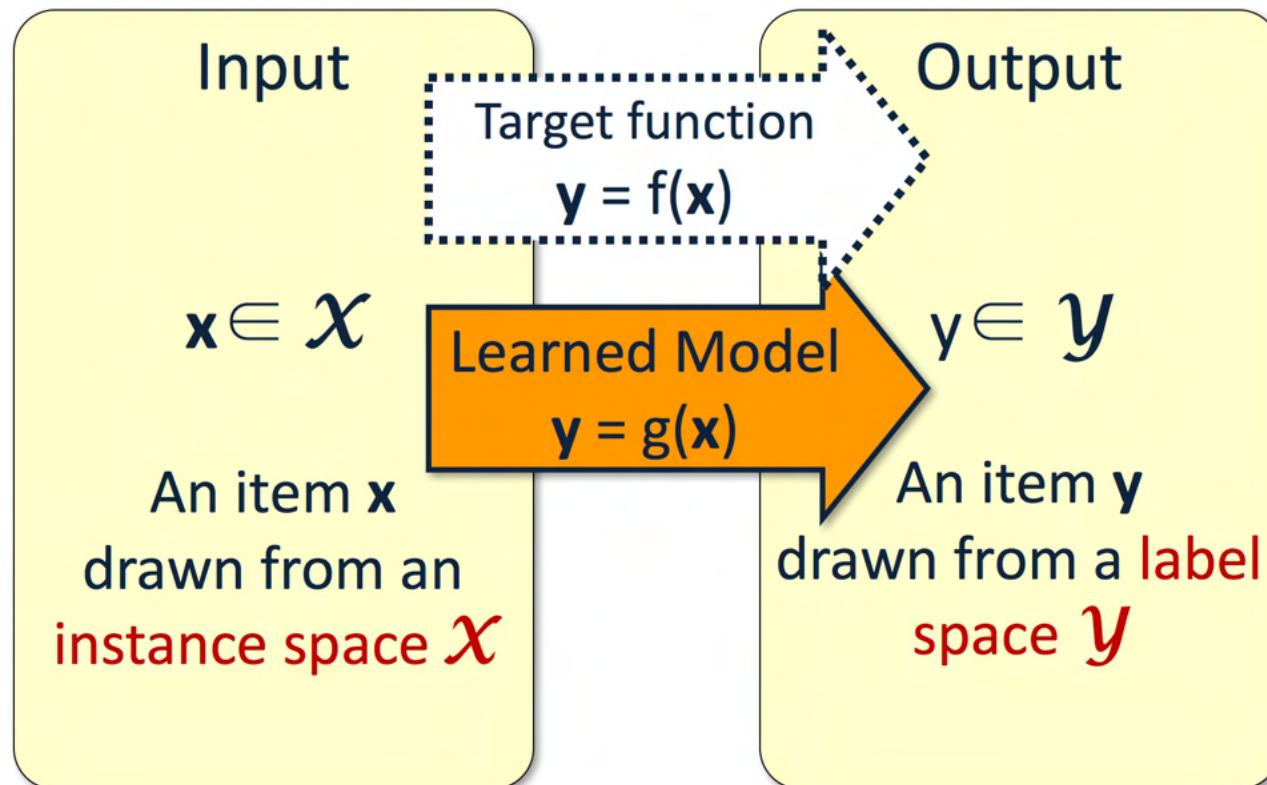
- ❖ [2 min] Introduce yourself to your group
 - ❖ Your name, your major and one interesting fact
- ❖ [5 min] Brainstorm:
What is your dream application of ML?
 - ❖ Define (Task, Performance, Experience)
 - ❖ Can be something not exist
(e.g., a robot writing homework for you)
- ❖ [3 min] Pick the best answer and the presenter
and post the presenter's name at chat box

Type of learning protocols

Types of learning (protocols)

❖ Supervised learning

- ❖ Given: **labeled** training instances (or examples)
- ❖ Goal: learn mapping that predicts label for test instance



Training phase:



lion



Not lion

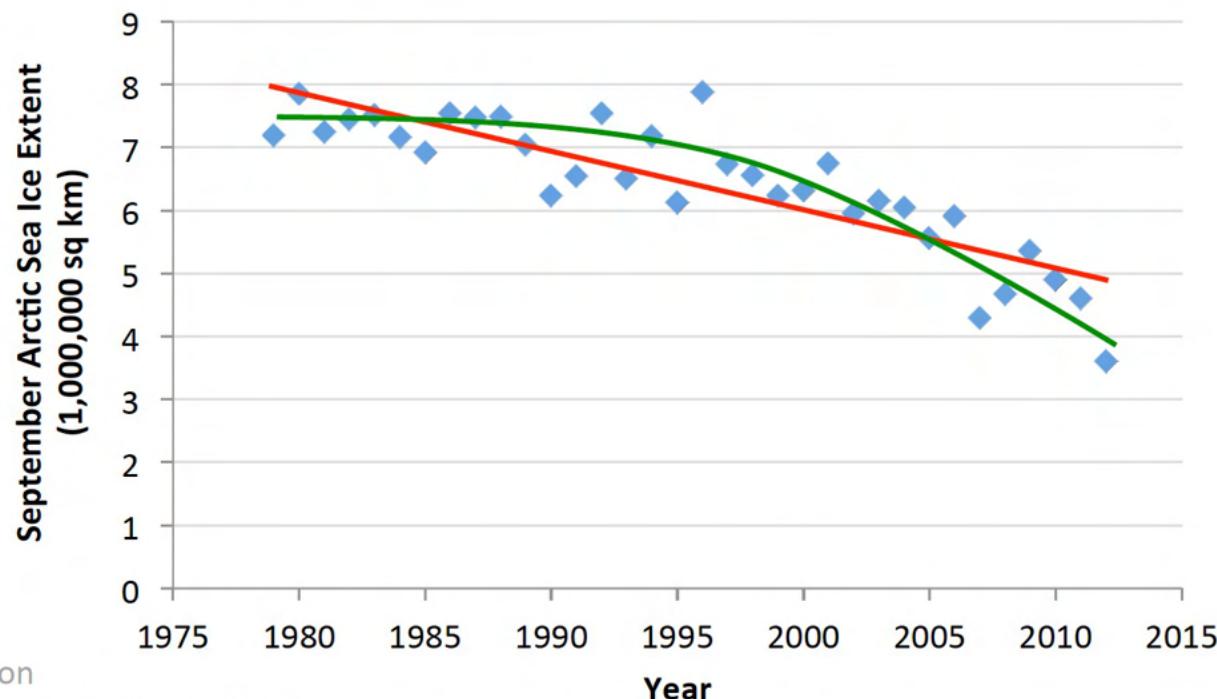
Test phase:



??

Supervised Learning (Regression)

- Given $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
 - y is real-valued == regression



Slide credit: Eric Eaton

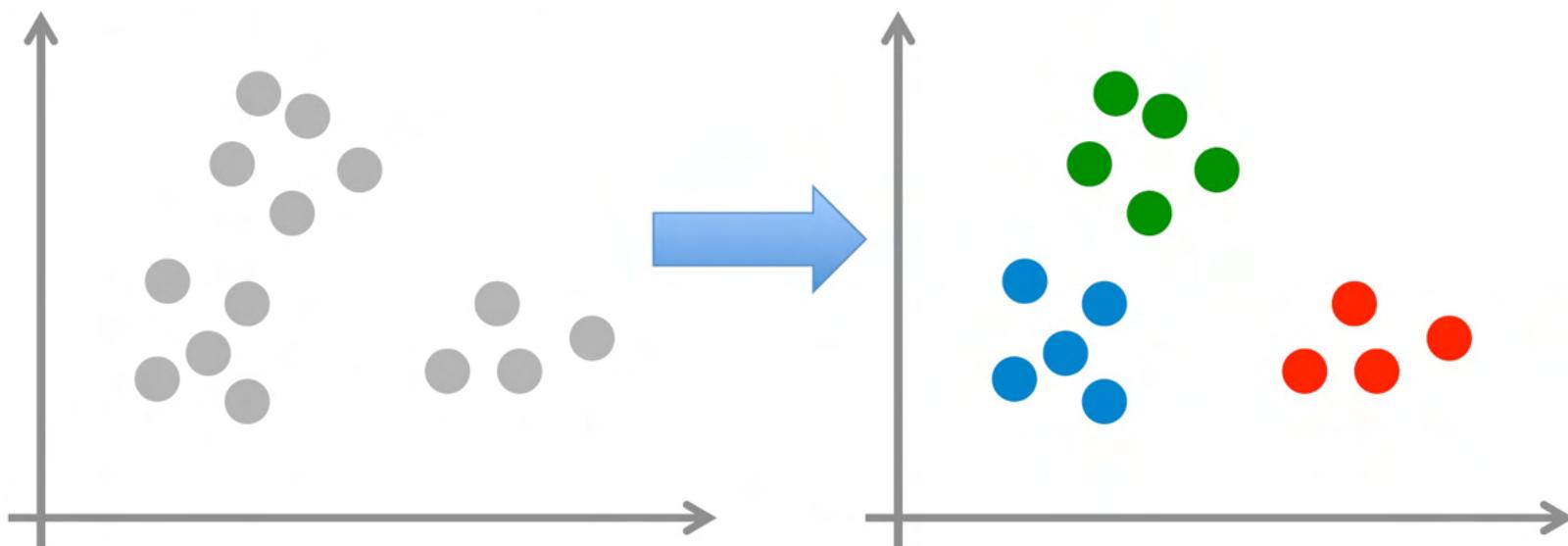
Data from G. Witt. Journal of Statistics

Education, Volume 21, Number 1 (2013)

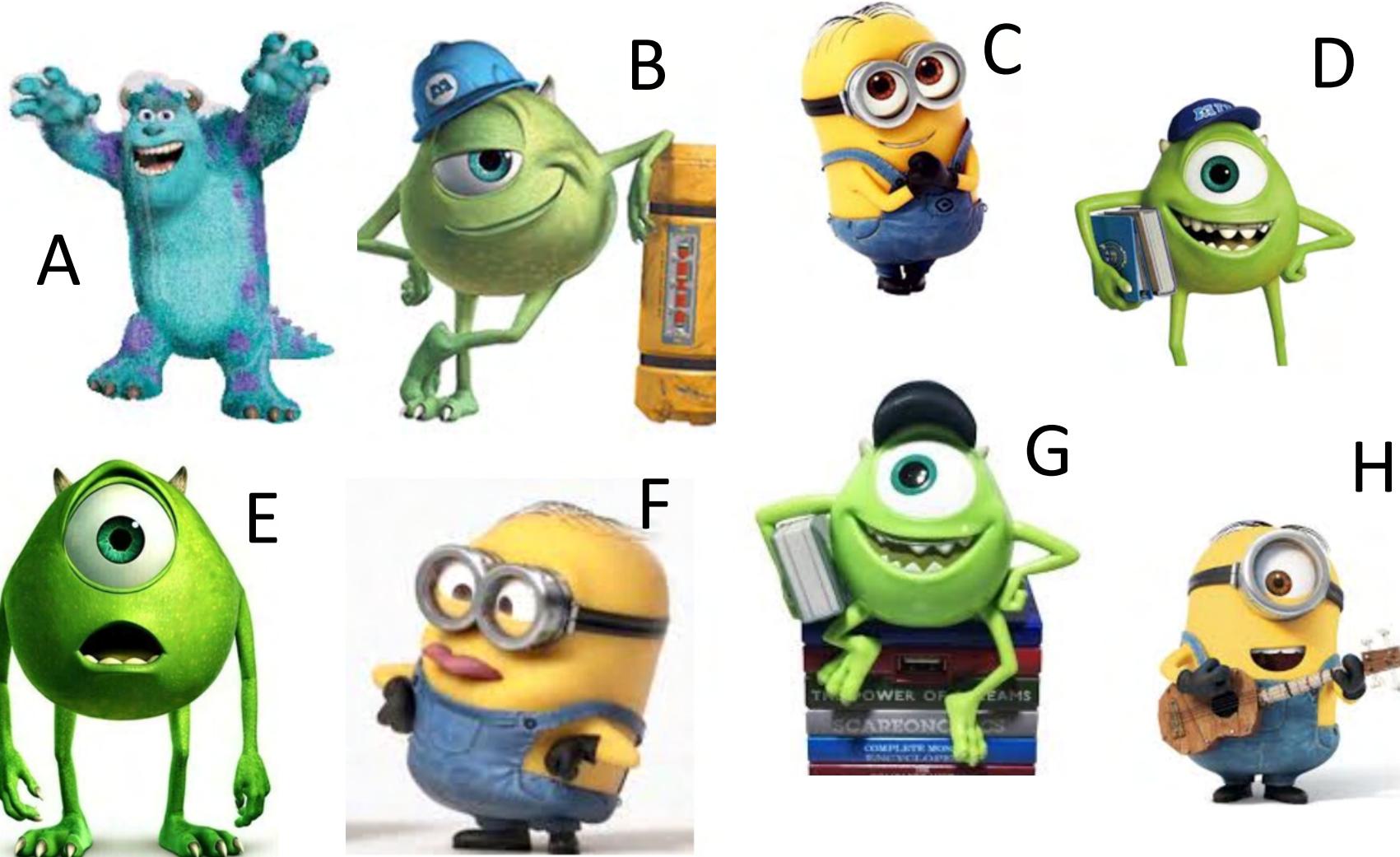


Unsupervised learning

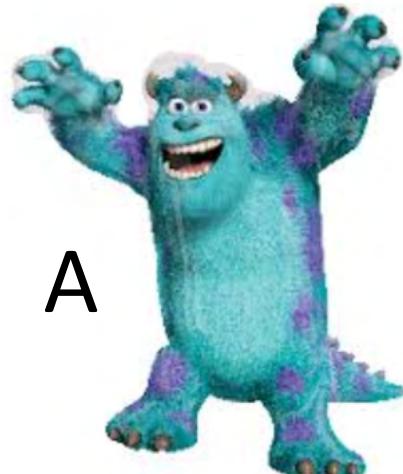
- ❖ Given: **unlabeled** inputs
- ❖ Goal: learn some intrinsic structure in inputs



How many “kinds of monsters” are there?



How many “kinds of monsters” are there?

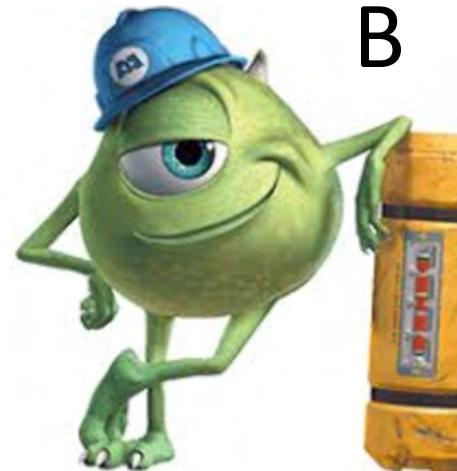


How many “kinds of monsters” are there?

H



B



C



F



E



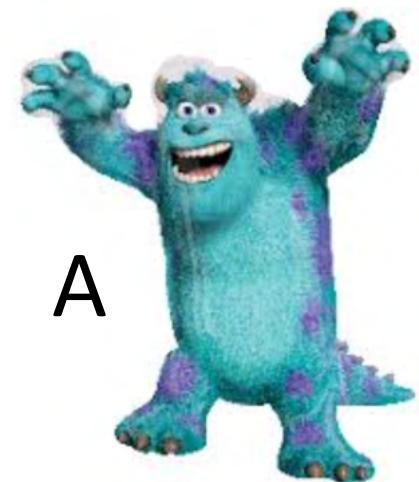
G



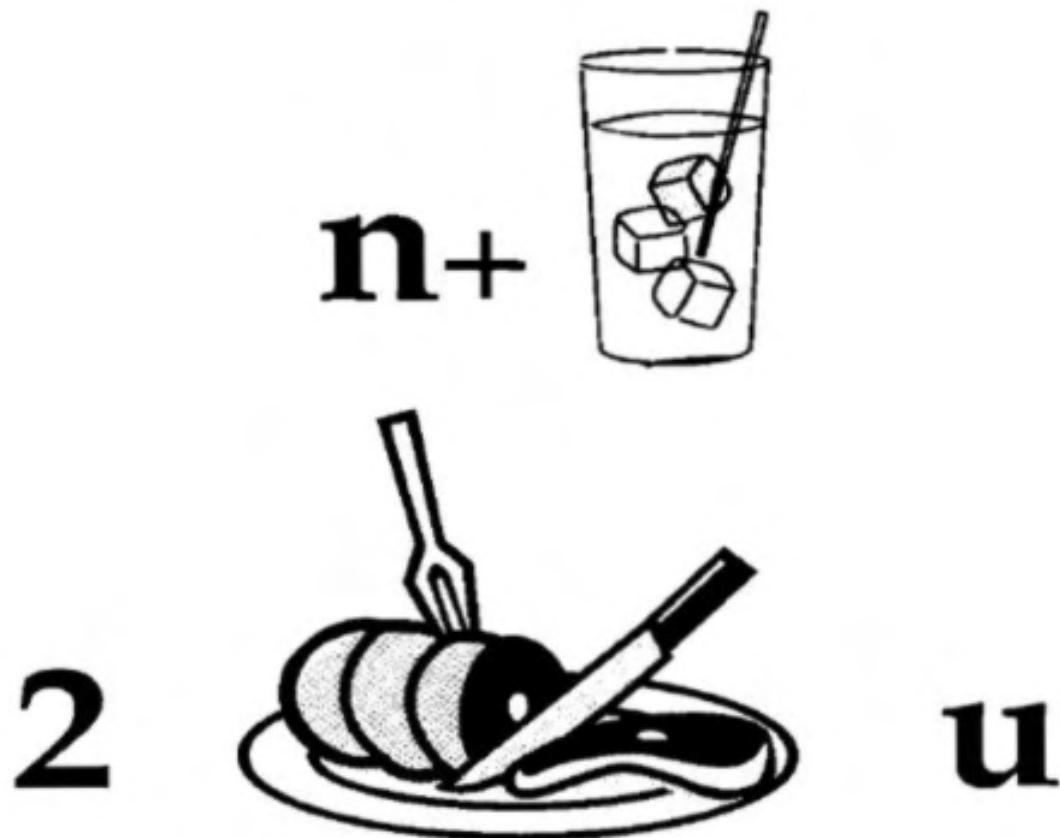
D



A

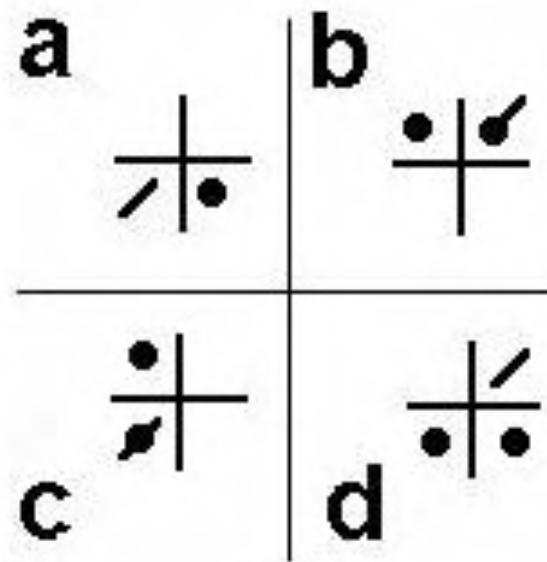
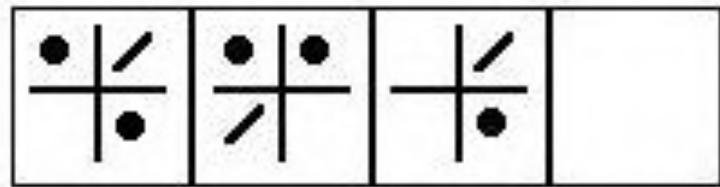


Decipher



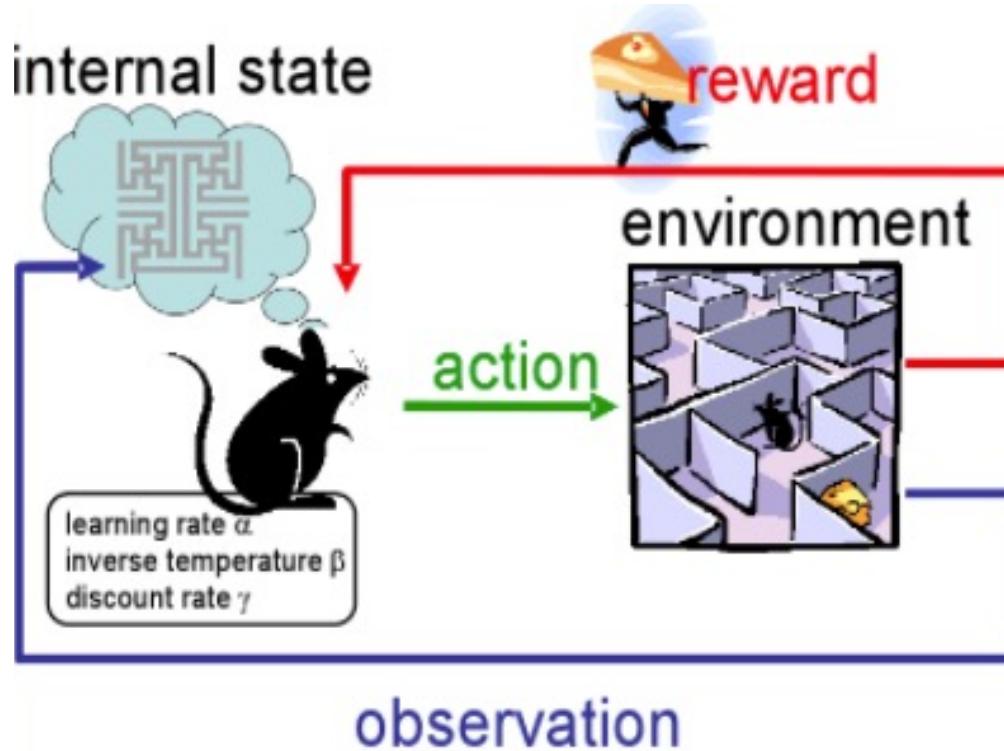
Credit: Dan Roth

IQ test



Reinforcement Learning

- ❖ Given sequence of states and actions with rewards
- ❖ Learn policy that maximizes agent's reward



(image taken from [Cyber Rodent Project](#))

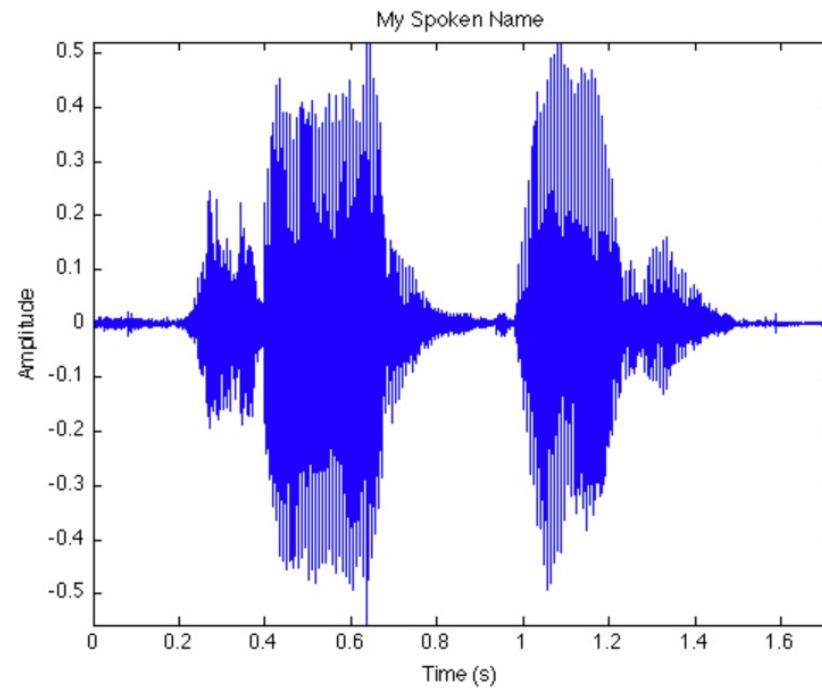


64

Challenges in ML

Challenges: Representation

- ❖ Representation:
How to represent input/output?



Challenges: What is the right model?

- ❖ Usually depends on size of data, type of problem, prior knowledge, annotation quality ...
- ❖ Also depends on the goal: model size / test-time budget / accuracy v.s. speed



Why the model
doesn't work?

Challenges: Structured Inference

- ❖ Many predictions are compositional
 - ❖ Require an inference process

Challenges: Structured Inference



Carefully
Slide

A screenshot of a web-based translation tool. The input field contains the Chinese text "小心地滑" (Xiǎoxīn dì huá). The output field shows the English translation "Carefully slide". A black arrow points from the Chinese input field down to the English output field. The interface includes language selection dropdowns for English, Spanish, French, Chinese - detected, and others, along with a "Translate" button.

Challenges: Structured Inference



小心:
Carefully
Careful
Take
Care
Caution

地滑:
Slide
Landslip
Wet Floor
Smooth

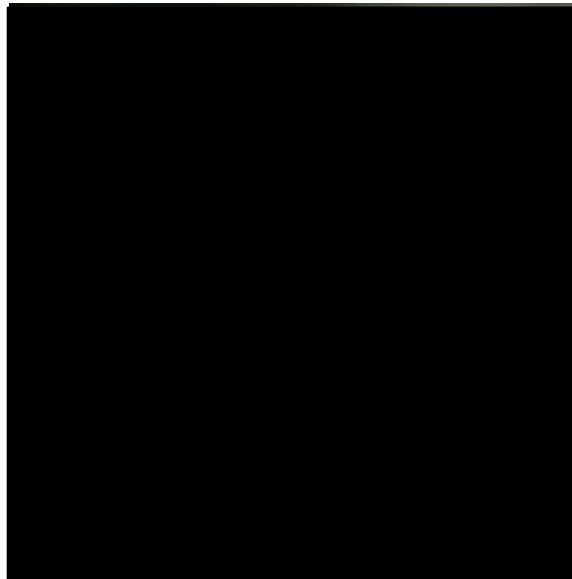
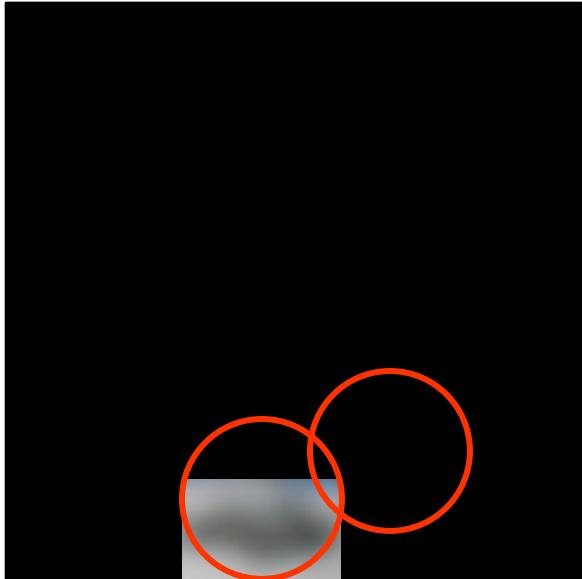
Translate

English Spanish French Chinese - detected English Spanish Arabic Translate

小心地滑 Carefully slide

Xiǎoxīn dì huá

Challenges: Structured Inference



Challenges: Robustness

Car or shoe?

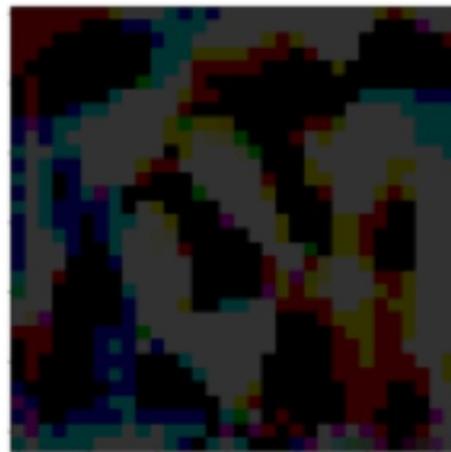


Challenges: Adversarial Attack



93%, 20 Km/h Sign

+ ϵx



$sign(\nabla * J(\theta, x, y))$

=



90%, 80 Km/h Sign



Lec 1: Intro

<https://arxiv.org/abs/1712.09327v1>

NLP Models are Vulnerable to Attack

Generating Natural Language Adversarial Examples [EMNLP 18]

Original Text Prediction: **Entailment** (Confidence = 86%)

Premise: *A runner wearing purple strives for the finish line.*

Hypothesis: *A **runner** wants to head for the finish line.*

Adversarial Text Prediction: **Contradiction** (Confidence = 43%)

Premise: *A runner wearing purple strives for the finish line.*

Hypothesis: *A **racer** wants to head for the finish line.*

Synonyms are selected by a genetic algorithm from a retrofitted word embedding.

Fairness in ML

Select photo  

X The photo you want to upload does not meet our criteria because:

- Subject eyes are closed

Please refer to the technical requirements.
You have 9 attempts left.

Check the photo [requirements](#).

[Read more about common photo problems and](#)

Subject eyes are closed

start again and re-enter the CAPTCHA security check.

Reference number: 20161206-81

Filename: Untitled.jpg

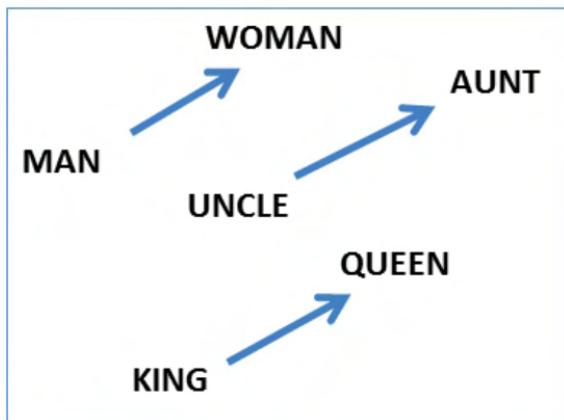
If you wish to [contact us](#) about the photo, you must provide us with the reference number given above.

Please print this information for your records.



Fairness in ML-- Word embedding bias

❖ $v_{man} - v_{woman} + v_{uncle} \sim v_{aunt}$



he: __	she: __
uncle	aunt
lion	
surgeon	
architect	
beer	
professor	



We use Google w2v embedding trained from the news

TODO

- ❖ Sign up at Piazza

<http://piazza.com/ucla/fall2021/m146>

Lecture 2:

Overview Fall 2021

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.

Announcement

- ❖ There is a discussion session on Friday
 - ❖ See session/time/loc on myUCLA
- ❖ Math Review Quiz is on CCLE

Brain Storm

- ❖ [2 min] Introduce yourself to your group
 - ❖ Your name, your major and one interesting fact
- ❖ [5 min] Brainstorm:
What is your dream application of ML?
 - ❖ Define (Task, Performance, Experience)
 - ❖ Can be something not exist
(e.g., a robot writing homework for you)
- ❖ [3 min] Pick the best answer and the presenter
and post the presenter's name at chat box

This Lecture

- ❖ Learning Protocols
 - ❖ Supervised Learning
 - ❖ Unsupervised Learning
- ❖ Framing Learning Problems

Type of learning protocols

Supervised Learning

Training phase:



lion



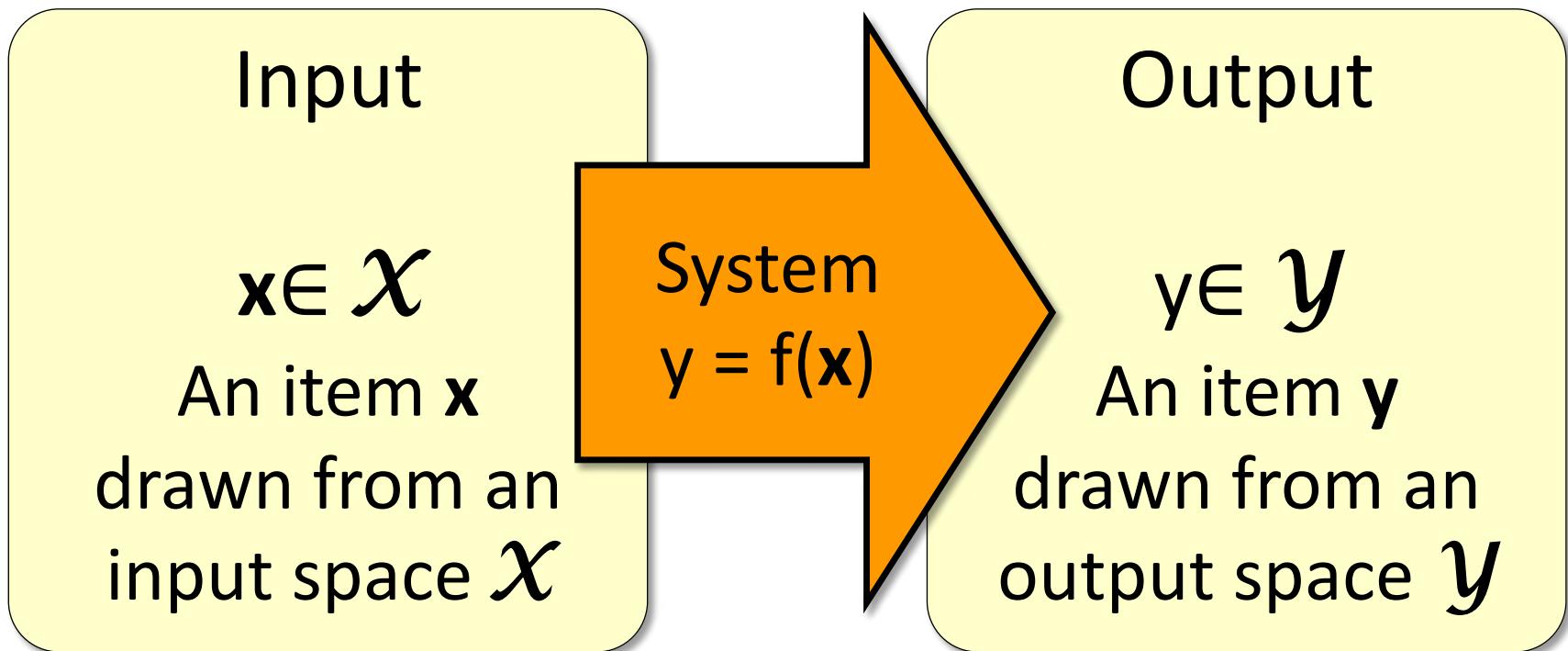
Not lion

Test phase:



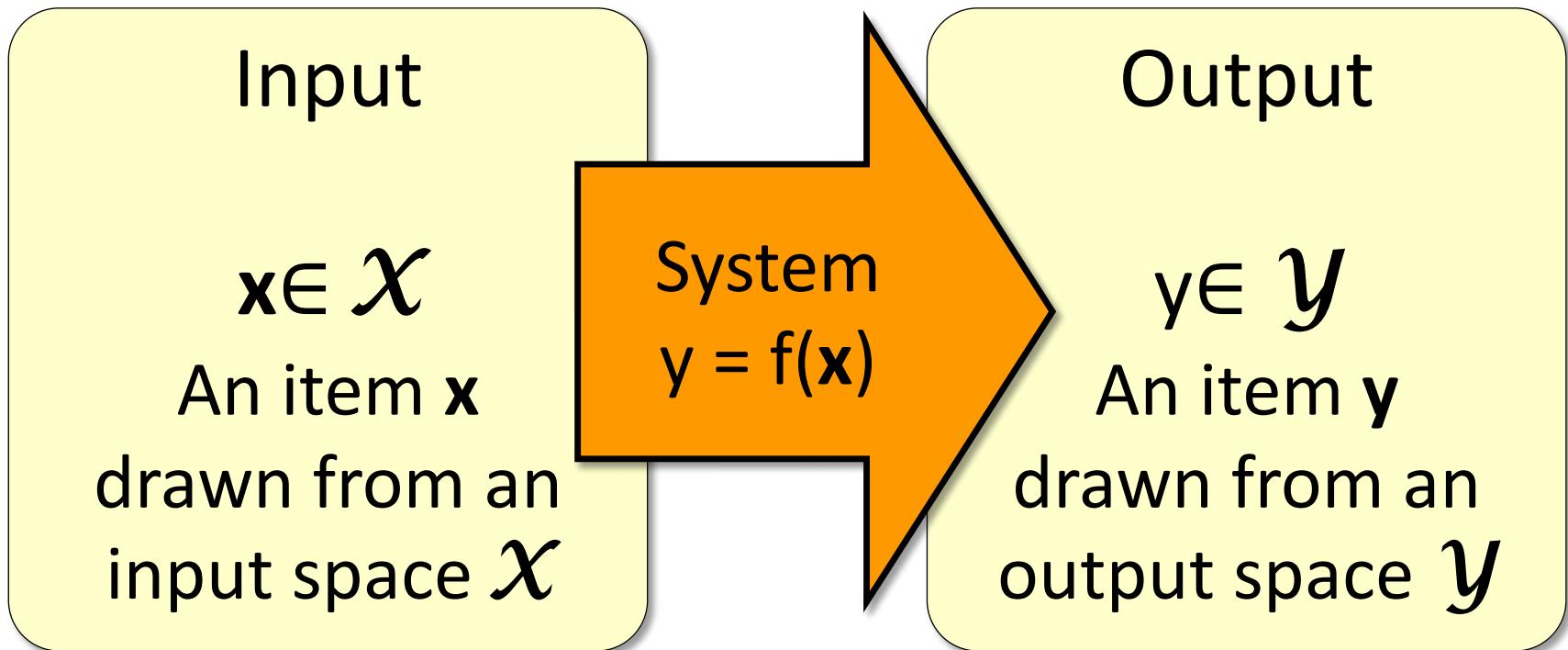
??

Supervised Learning



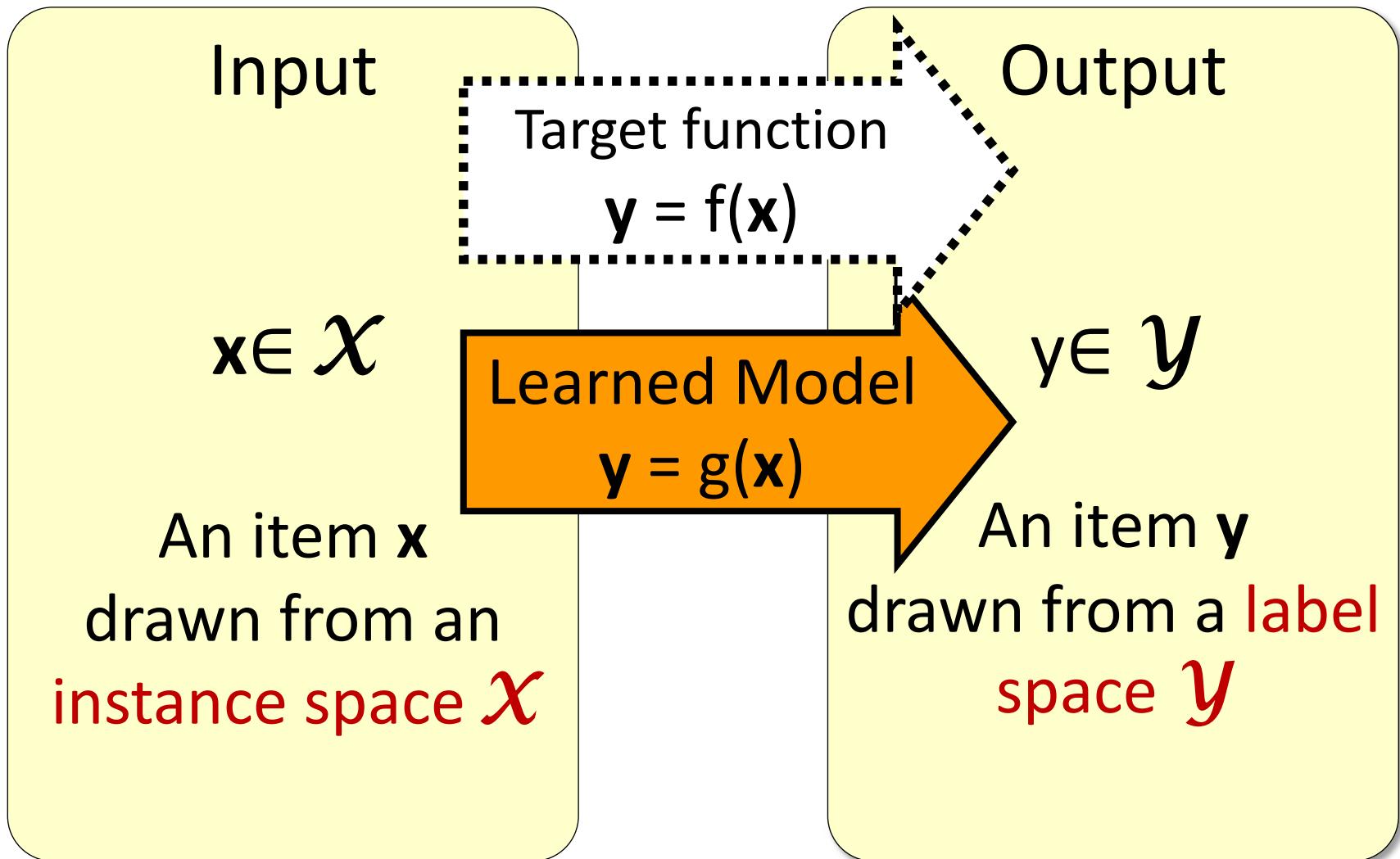
- ❖ We consider systems that apply a function $f()$ to input items x and return an output $y = f(x)$.

Supervised Learning

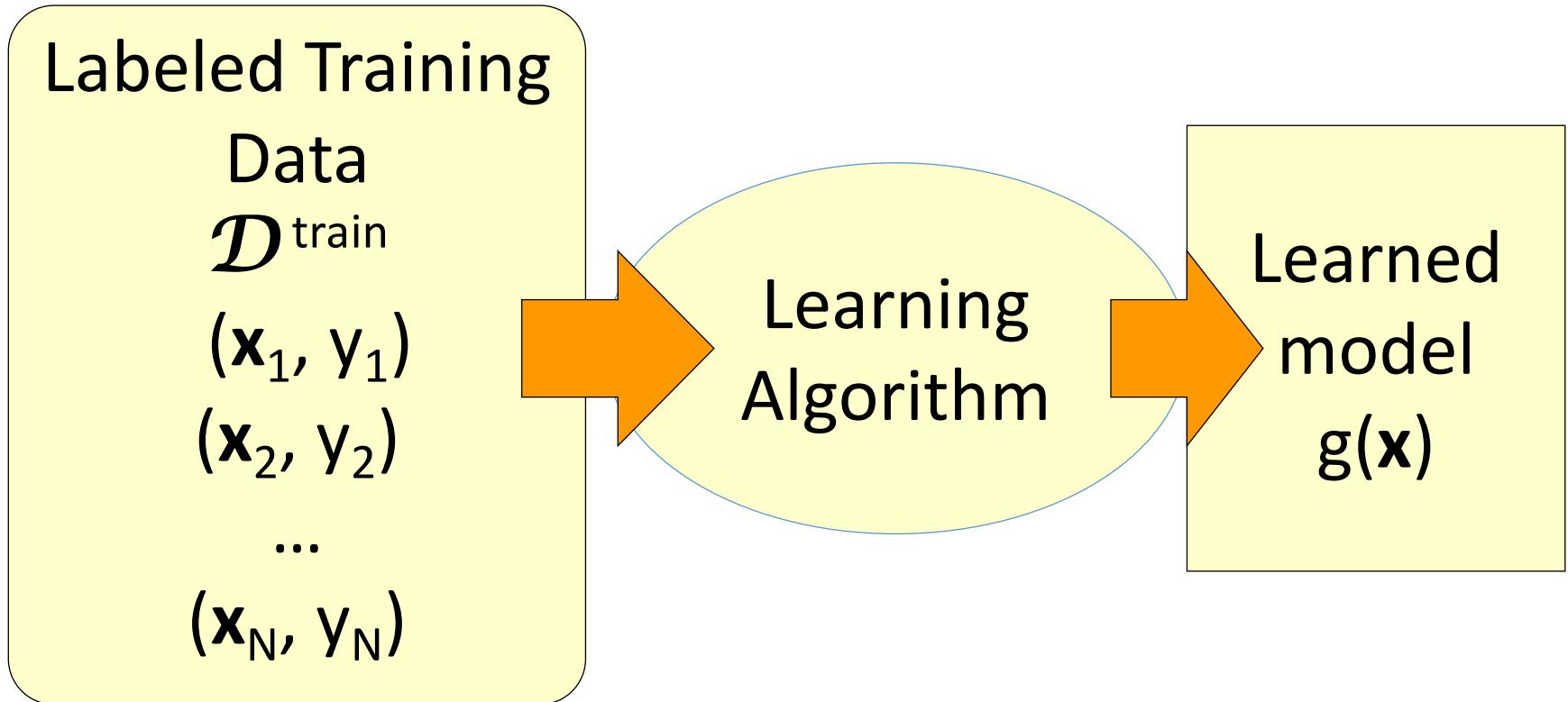


- ❖ In (supervised) machine learning, we deal with systems whose $f(\mathbf{x})$ is learned from examples.

Supervised Learning



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

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

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

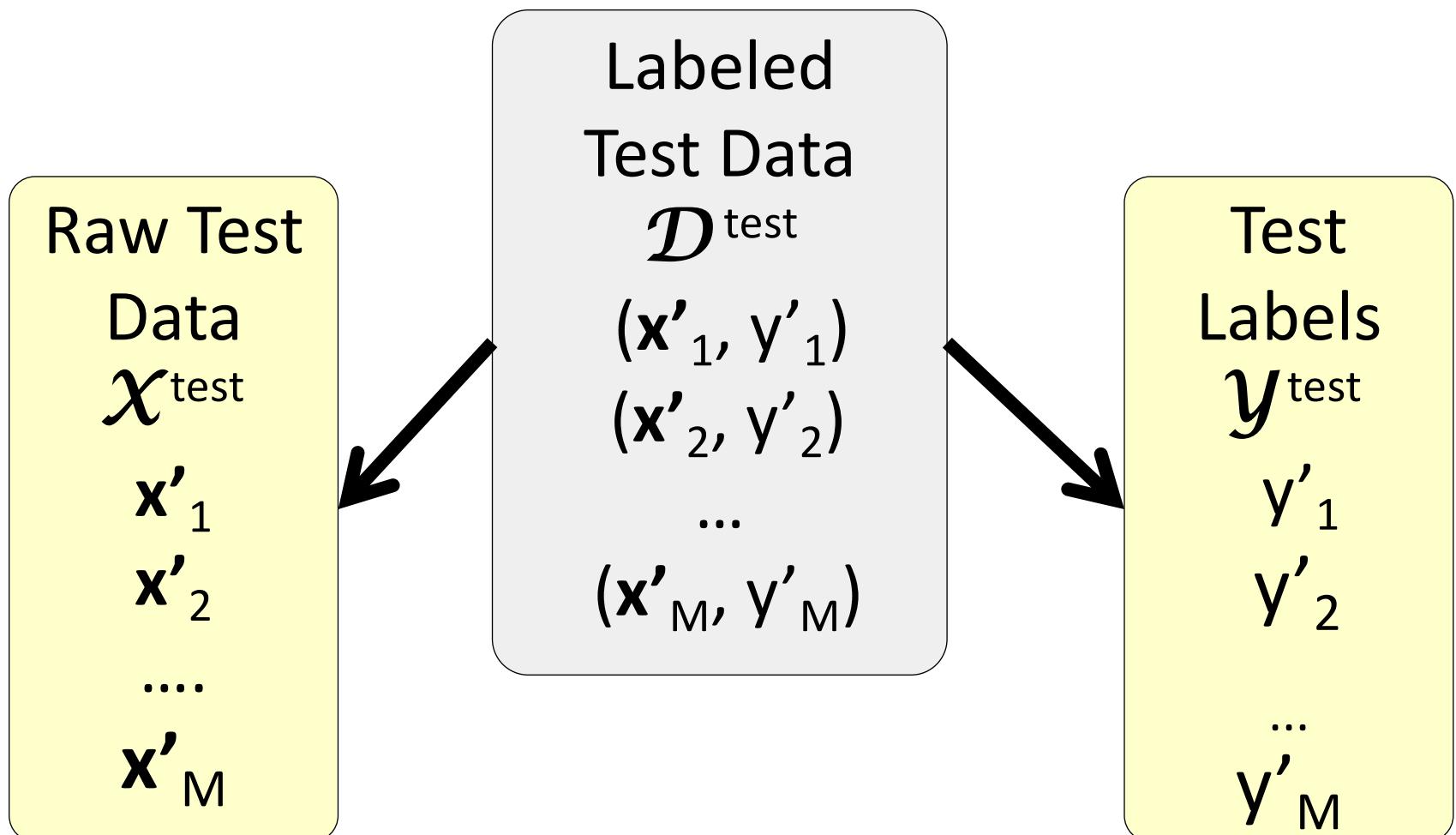
$(\mathbf{x}'_2, \mathbf{y}'_2)$

...

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

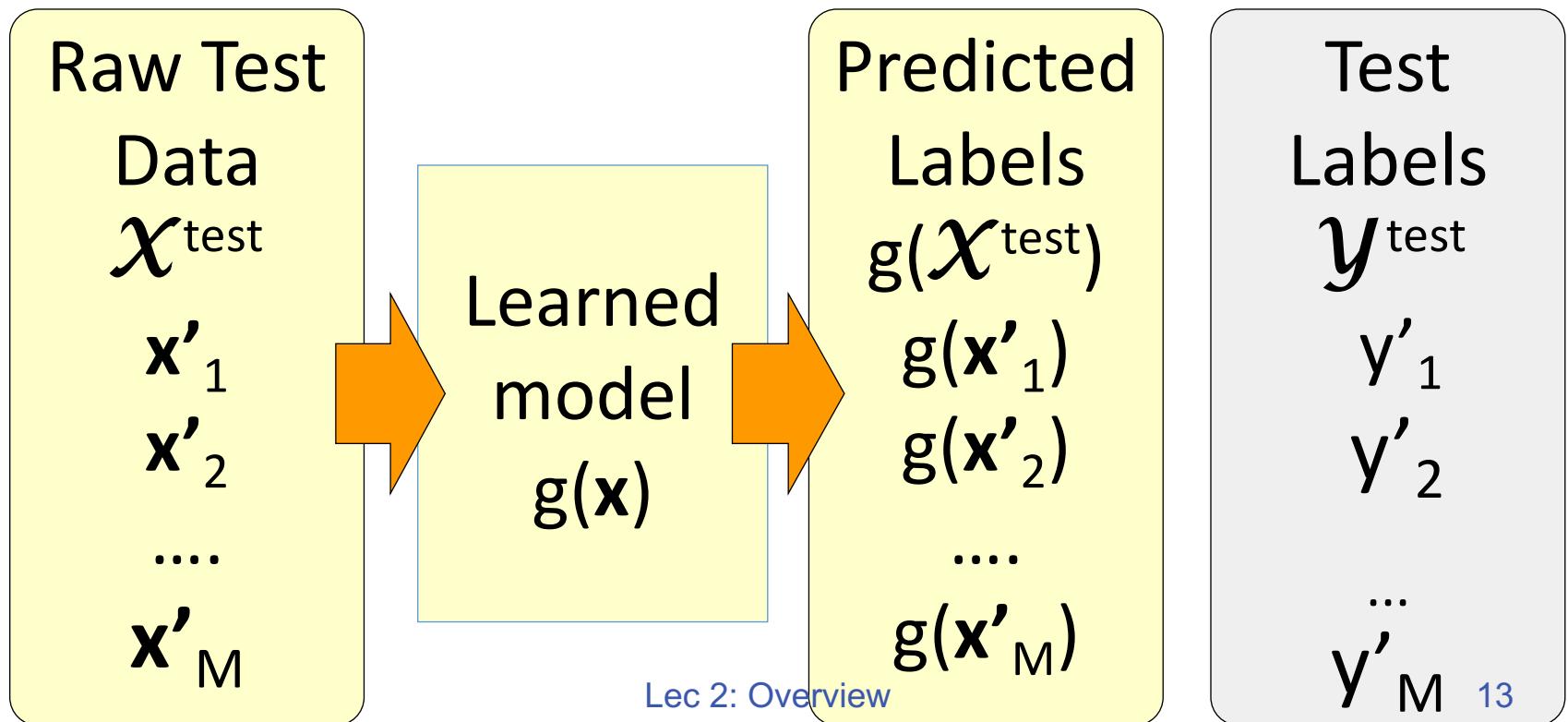
- ❖ Reserve some labeled data for testing

Supervised Learning: Testing

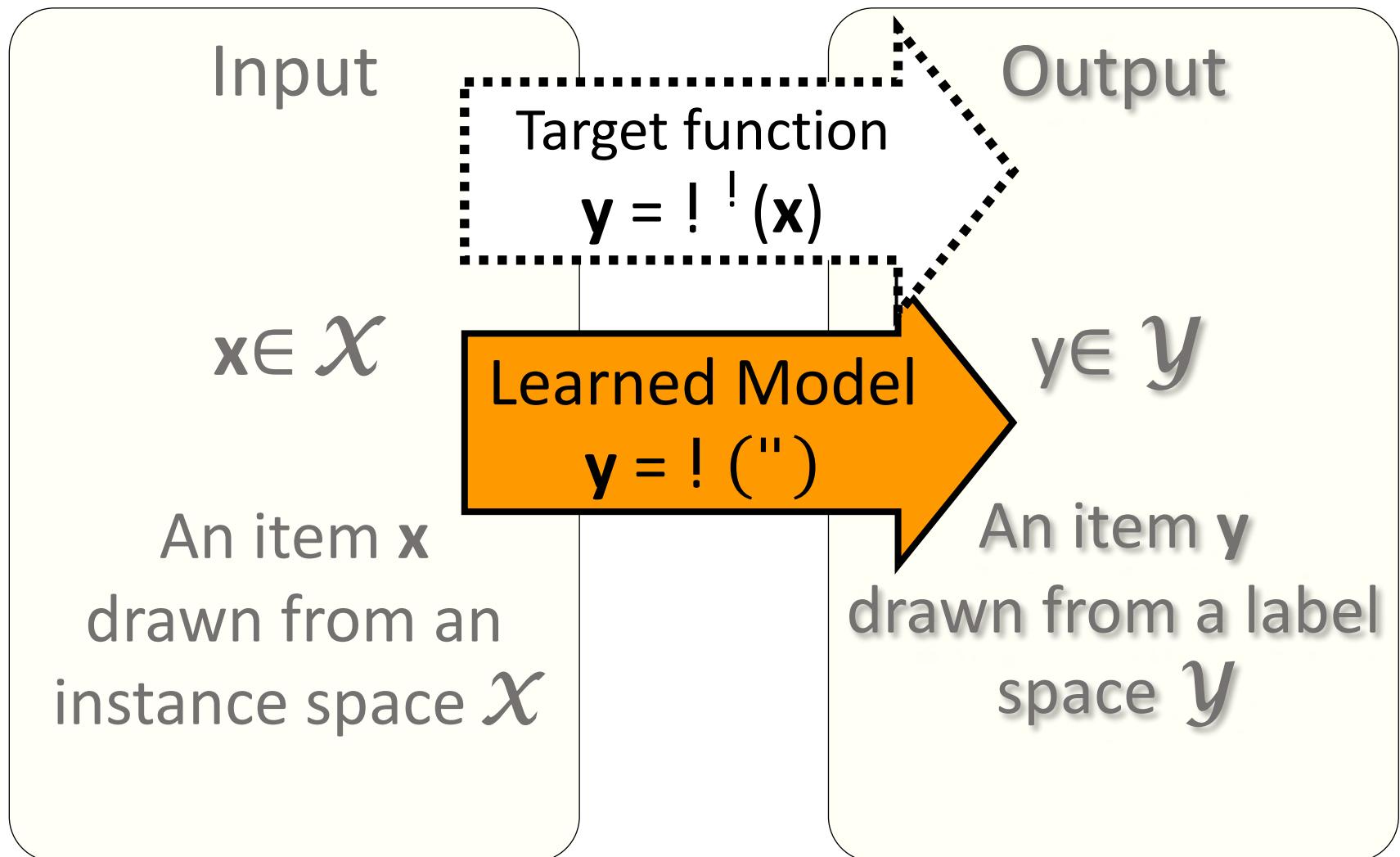


Supervised Learning: Testing

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

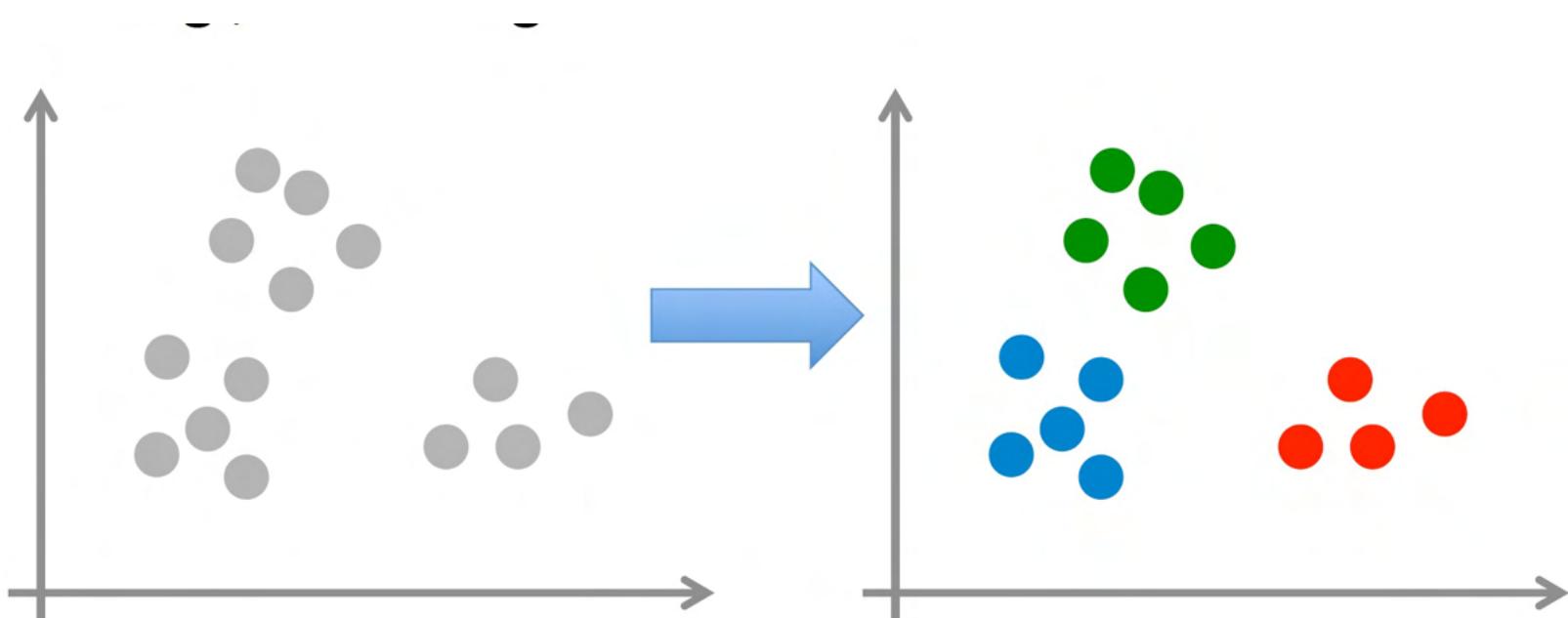


Learning the Mapping

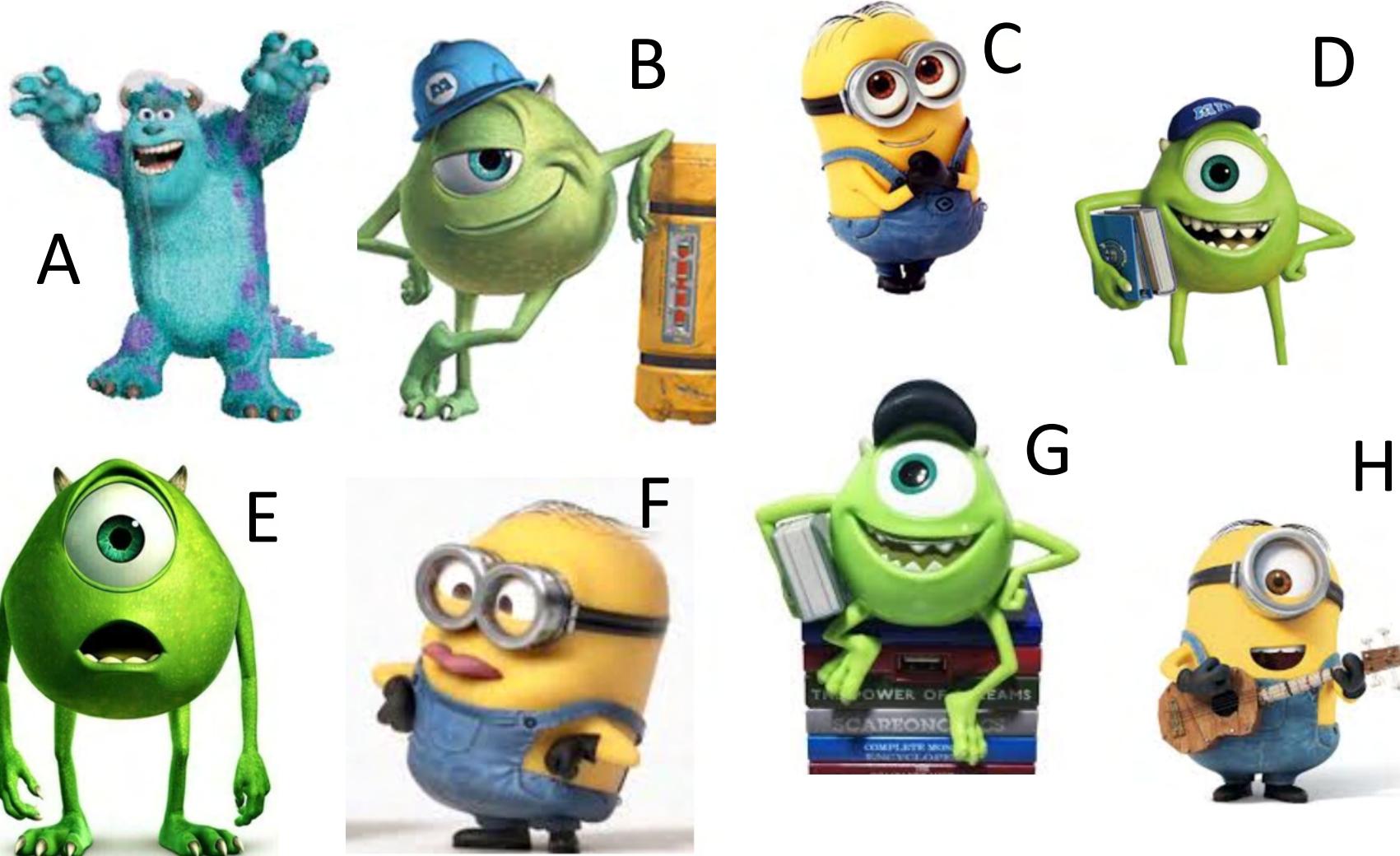


Unsupervised learning

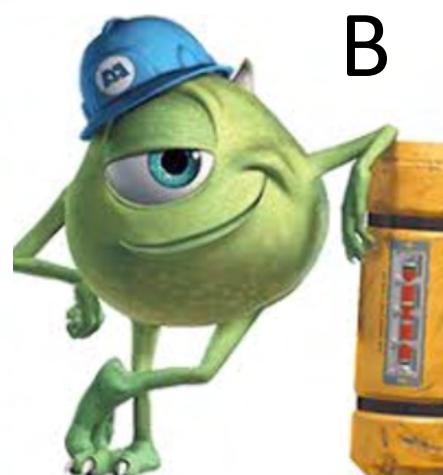
- ❖ Given: **unlabeled** inputs
- ❖ Goal: learn some intrinsic structure in inputs



How many “kinds of monsters” are there?



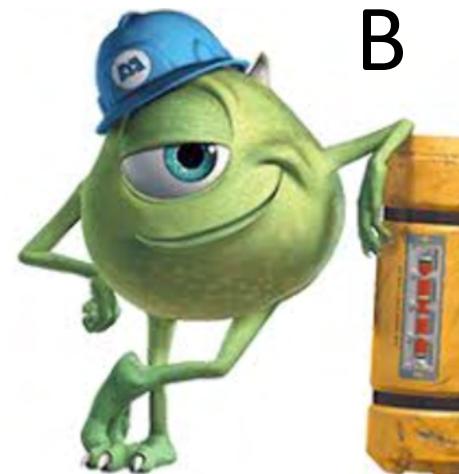
How many “kinds of monsters” are there?



How many “kinds of monsters” are there?



H



B



C



F



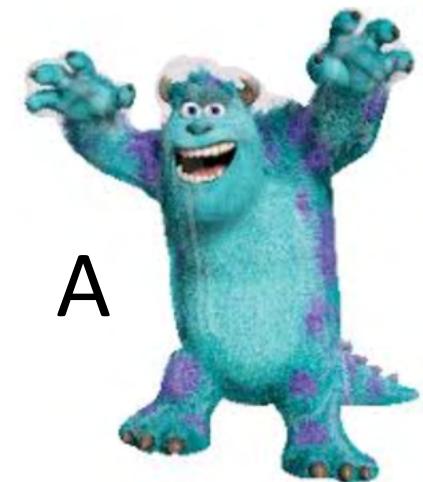
E



G

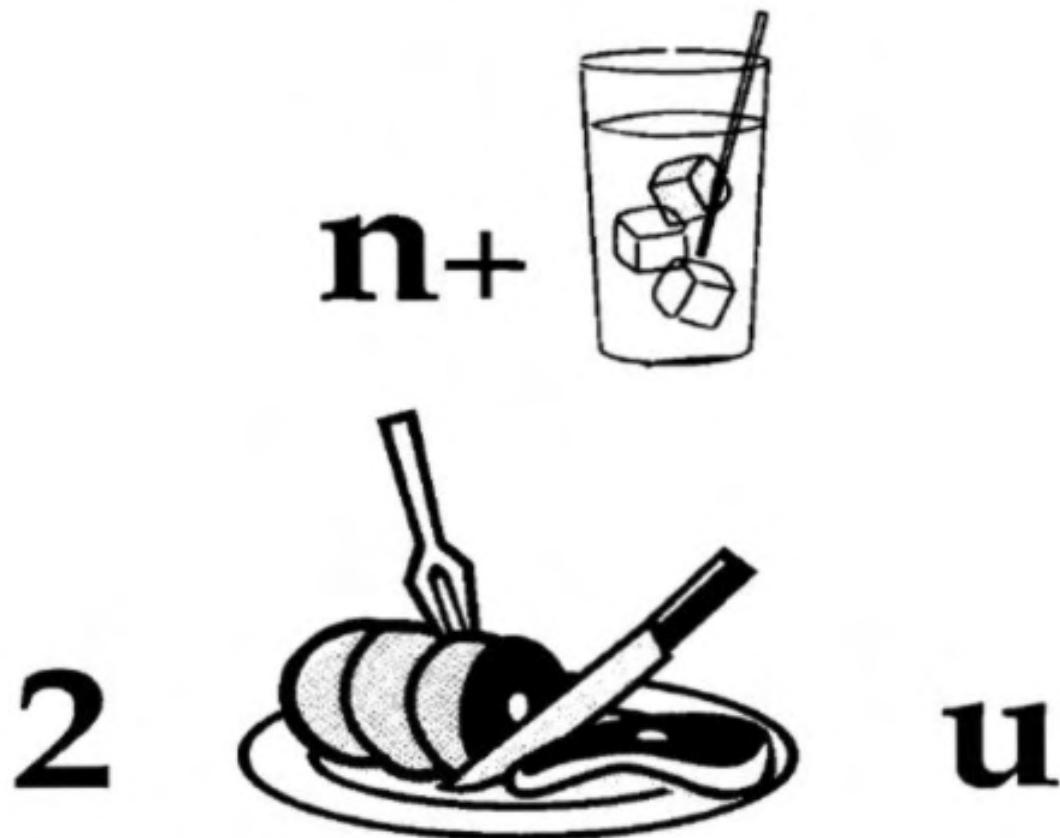


D



A

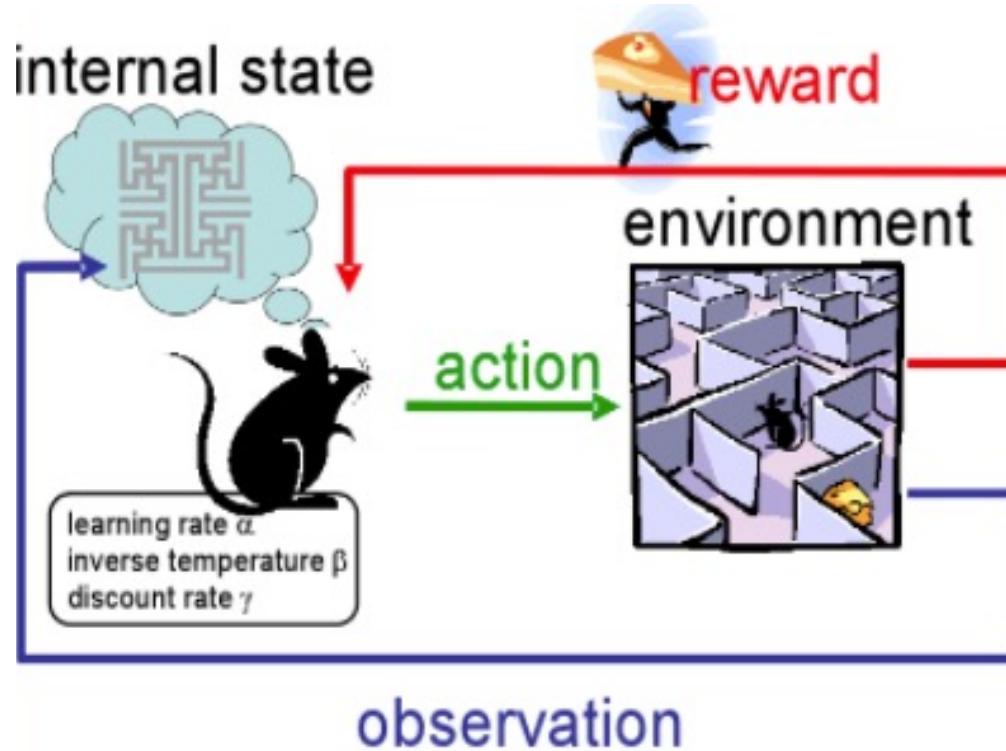
Decipher



Credit: Dan Roth

Reinforcement Learning

- ❖ Given sequence of states and actions with rewards
- ❖ Learn policy that maximizes agent's reward



(image taken from [Cyber Rodent Project](#))

Framing a Learning Problem

What is Learning

- ❖ The Badges Game.....
 - ❖ This is an example of the key learning protocol:
supervised learning
- ❖ Issues:
 - ❖ Representation
 - ❖ Problem setting
 - ❖ 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 | + 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 |
| - Chidanand Apte | + Malini Bhandaru | - Claire Cardie |
| + Minoru Asada | + Bir Bhanu | + John Case |
| + Lars Asker | + Reinhard Blasig | + Jason Catlett |
| + Javed Aslam | - Avrim Blum | - Philip Chan |
| + Jose L. Balcazar | - Anselm Blumer | - Zhixiang Chen |
| - Cristina Baroglio | + Justin Boyan | - 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

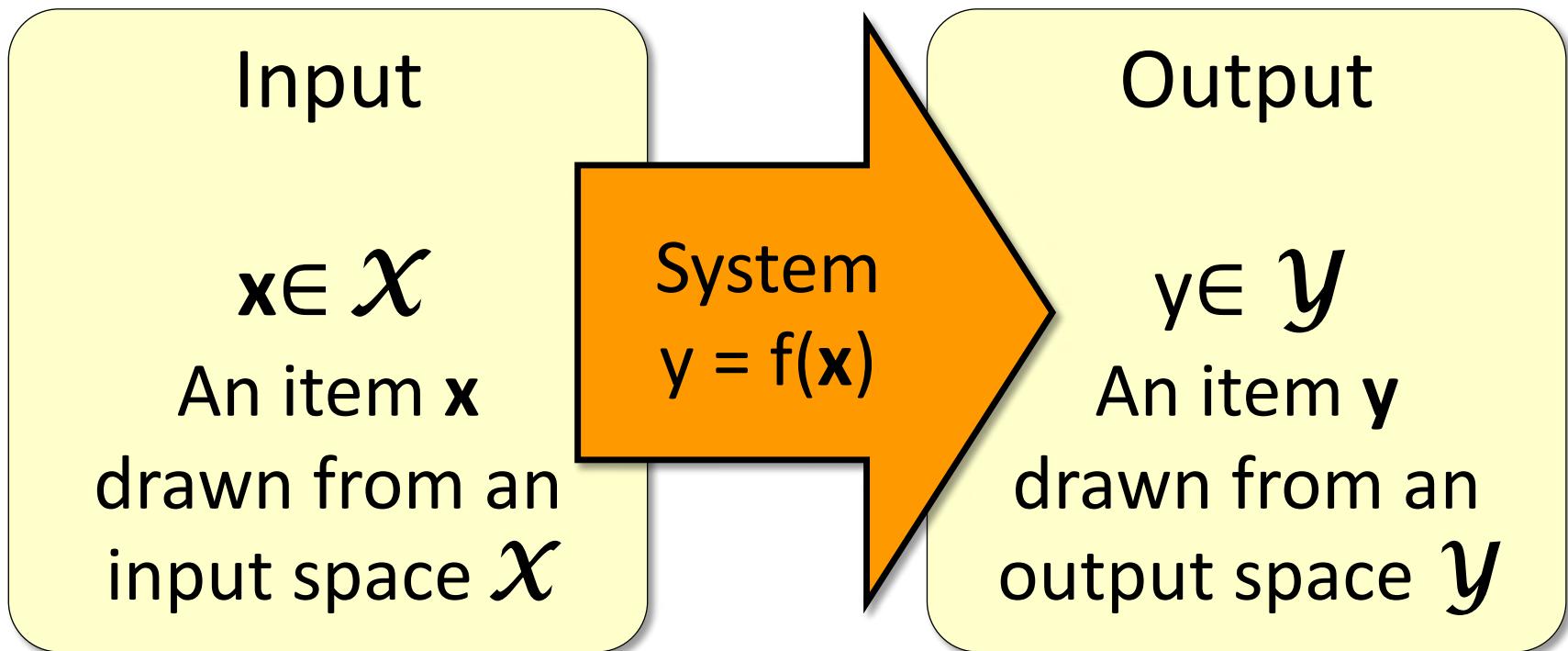
Labeled test data

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

Exercise: What is the rule?

- | | | |
|---------------------|-------------------|--------------------|
| + 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 |
| - Chidanand Apte | + Malini Bhandaru | - Claire Cardie |
| + Minoru Asada | + Bir Bhanu | + John Case |
| + Lars Asker | + Reinhard Blasig | + Jason Catlett |
| + Javed Aslam | - Avrim Blum | - Philip Chan |
| + Jose L. Balcazar | - Anselm Blumer | - Zhixiang Chen |
| - Cristina Baroglio | + Justin Boyan | - Chris Darken |

Supervised Learning



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

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

$x \in \mathcal{X}$

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 χ for the Badges game?

❖ Possible features:

- 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

- Myriam Abramson

+ David W. Aha

+ Kamal M. Ali

- Eric Allender

+ Dana Angluin

+ Peter Bartlett

- Eric Baum

+ Welton Becket

- Shai Ben-David

+ George Berg

+ Neil Berkman

+ Carla E. Brodley

+ Nader Bshouty

- Wray Buntine

- Andrey Burago

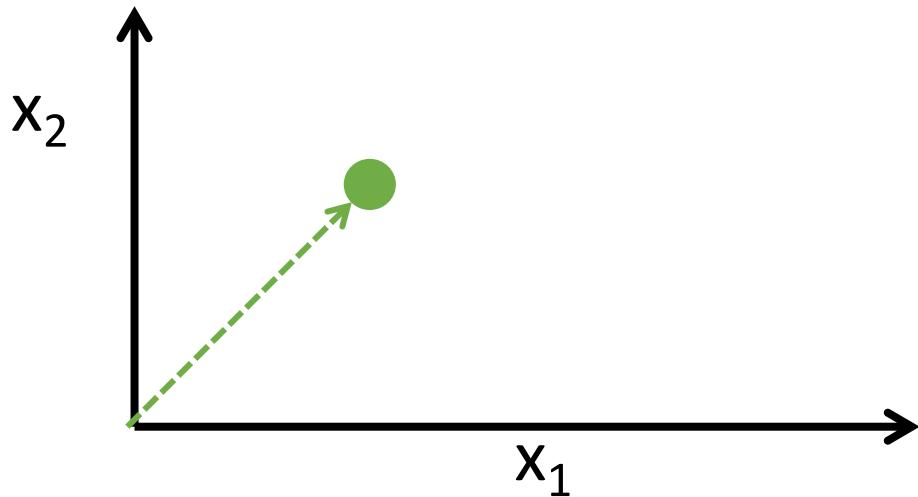
+ Tom Bylander

+ Bill Byrne



\mathcal{X} as a vector space

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



Example: the badge game

+ Naoki Abe

- Myriam Abramson

+ David W. Aha

+ Kamal M. Ali

- Eric Allender

+ Dana Angluin

+ Peter Bartlett

- Eric Baum

+ Welton Becket

- Shai Ben-David

+ George Berg

+ Neil Berkman

+ Carla E. Brodley

+ Nader Bshouty

- Wray Buntine

- Andrey Burago

+ Tom Bylander

+ Bill Byrne

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

+ Naoki Abe

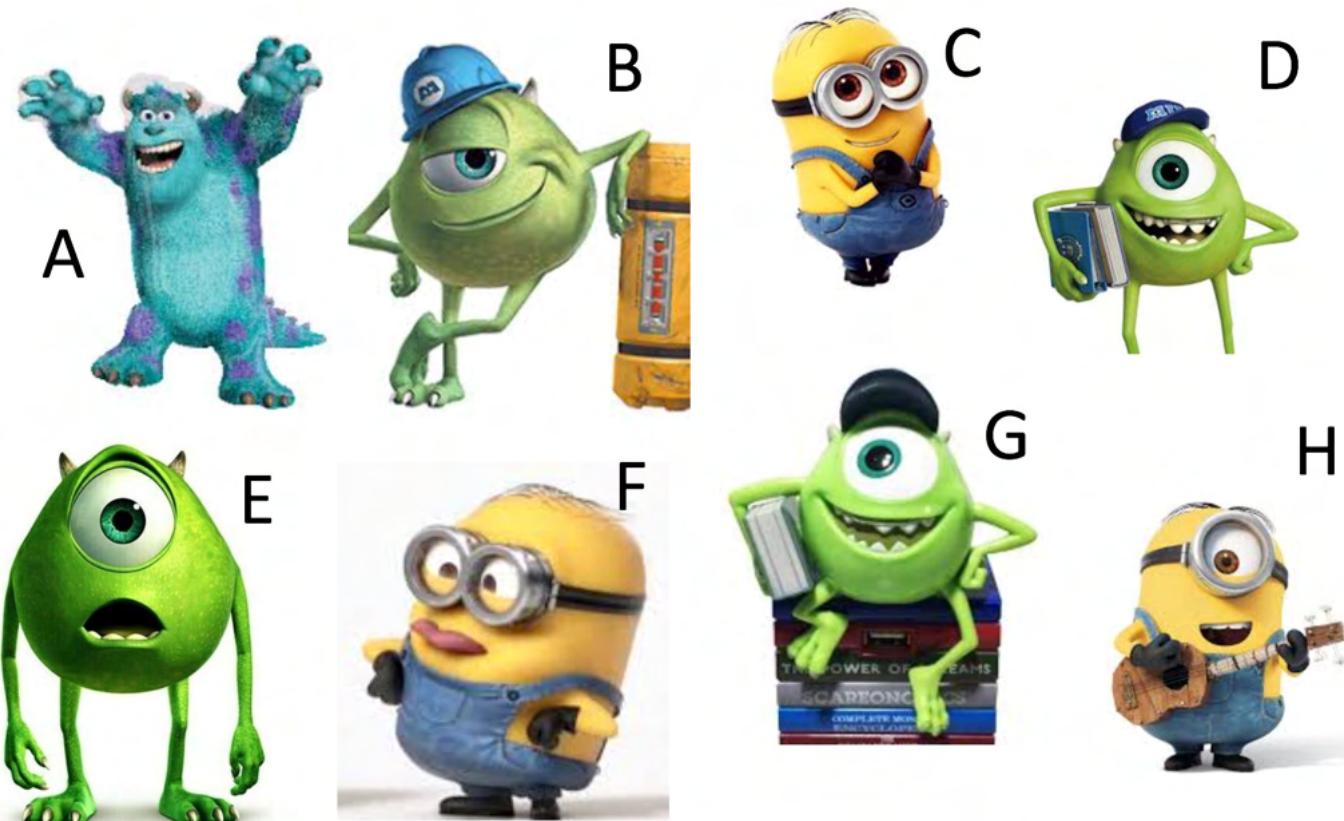
[0 , 0 , 1 , 1 ...]

- Avrim Blum

[1 , 1 , 0 , 0 ...]



Your turn – What 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}

Supervised Learning : Examples

Animal recognition

- ❖ x : Bitmap picture of the animal
- ❖ y :



Lion? Yes/[No](#)

Lion/Cat/[Dog](#)

Lion/[Mammal](#)/[Dog](#)/Fish

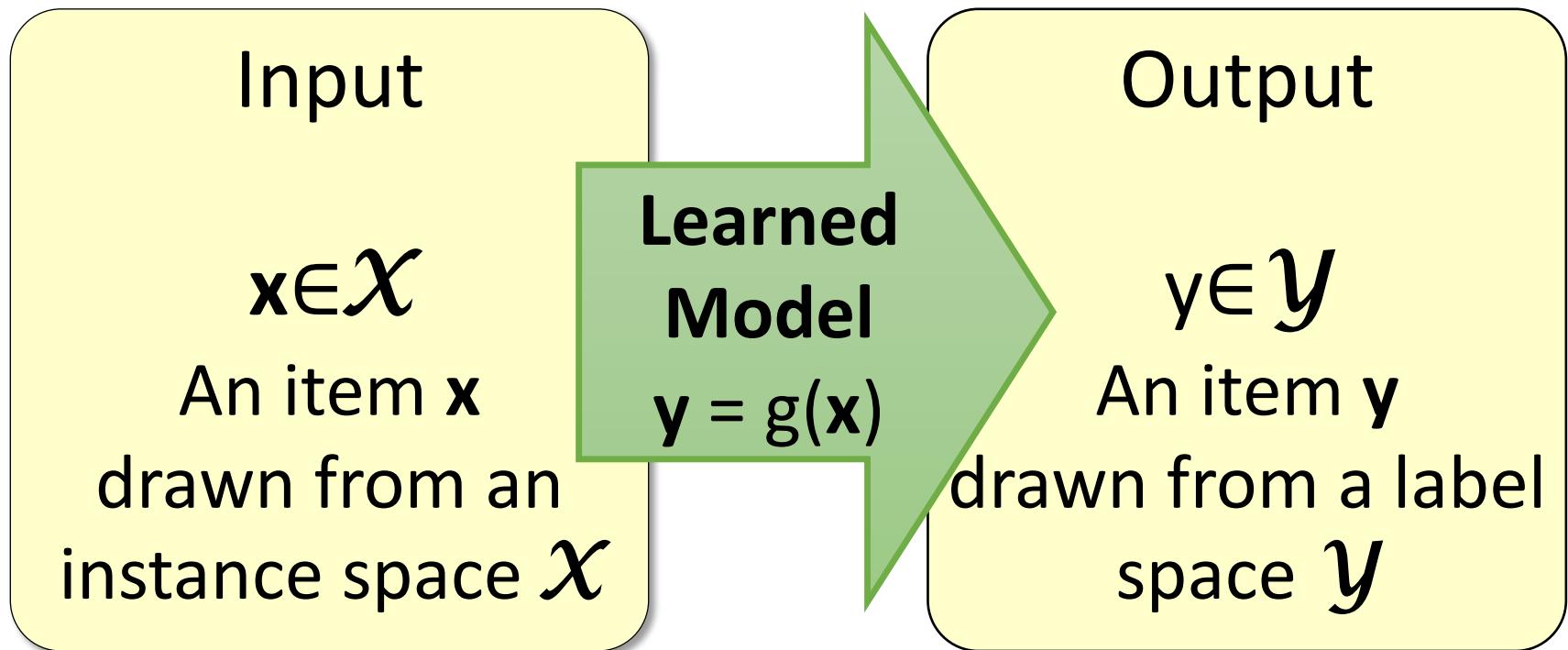
Binary output

Multiclass output

Multilabel output

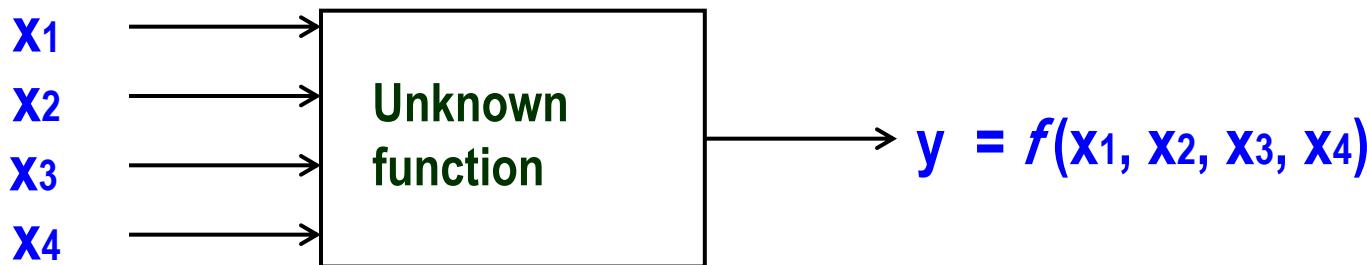
Output of the applications may different from the output of ML models.

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



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

A Learning Problem



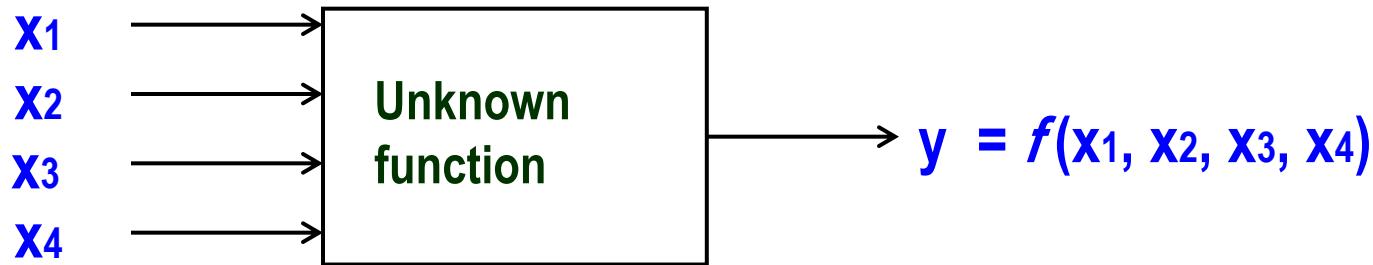
Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

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

Discussion: A Learning Problem



Example	x_1	x_2	x_3	x_4	y	
1	0	0	1	0	0	Can you learn this function?
2	0	1	0	0	0	What is it?
3	0	0	1	1	1	A function g is consistent to a dataset
4	1	0	0	1	1	$D = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall i$
5	0	1	1	0	0	
6	1	1	0	0	0	How many possible functions over four features?
7	0	1	0	1	0	How many function is consistent to D on the left

Lecture 3:

Hypothesis Space & KNN

Fall 2021

Kai-Wei Chang

CS @ UCLA

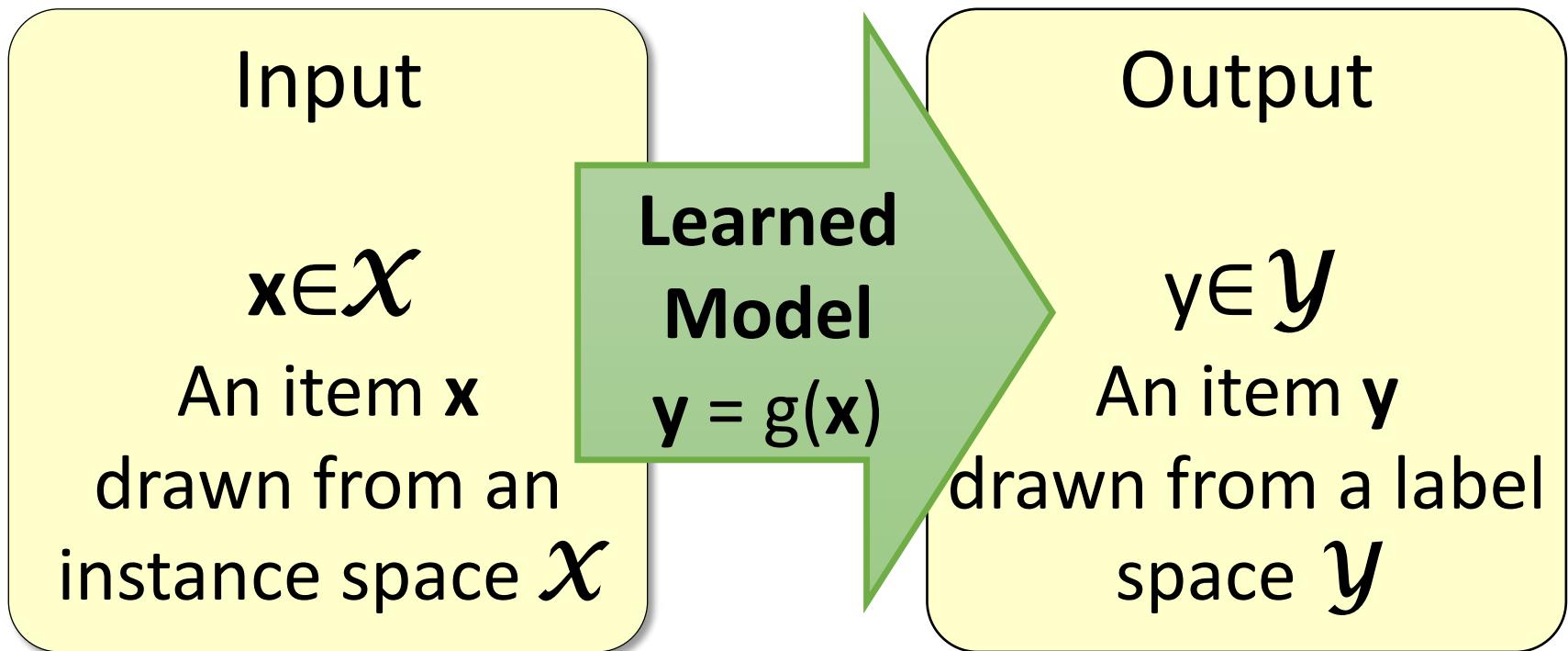
kw+cm146@kwchang.net

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.

Announcement

- ❖ Homework 1 will be released on Thu and due two weeks after

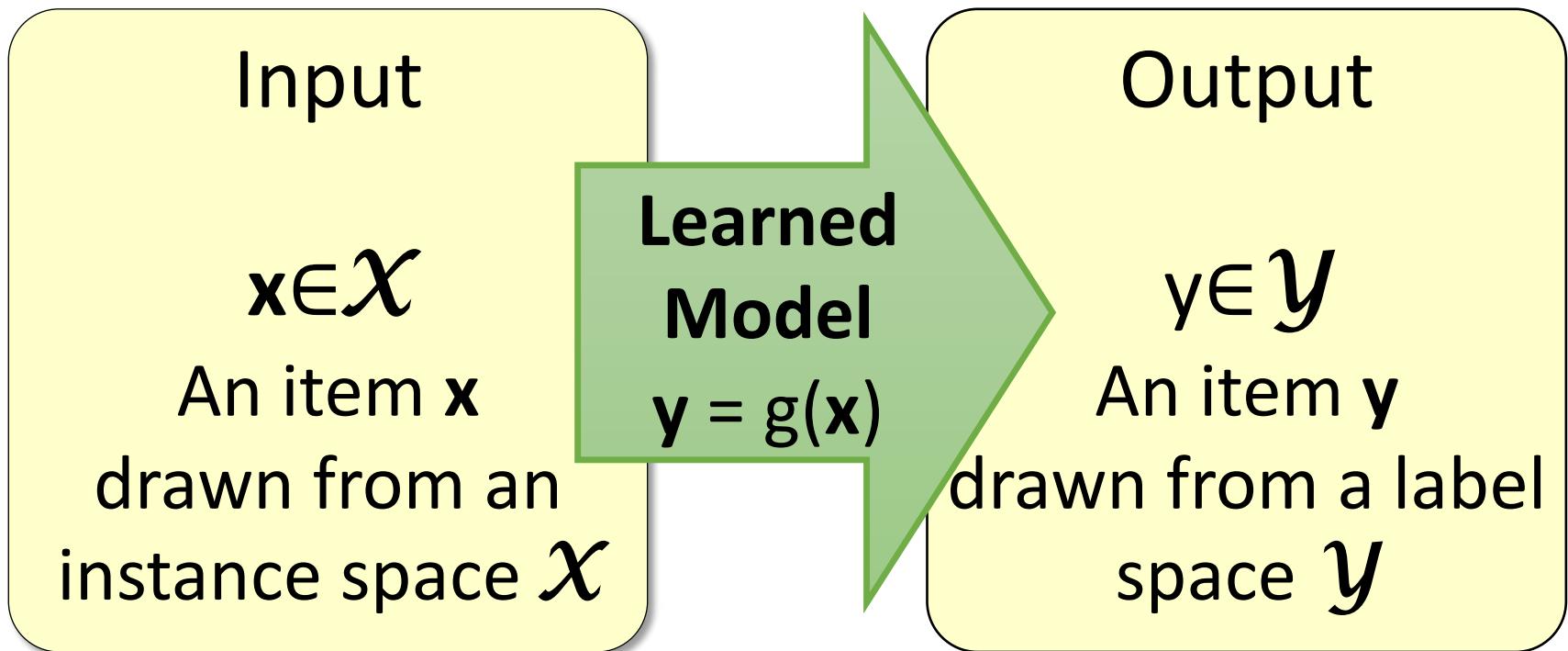
Supervised Learning



Using supervised learning

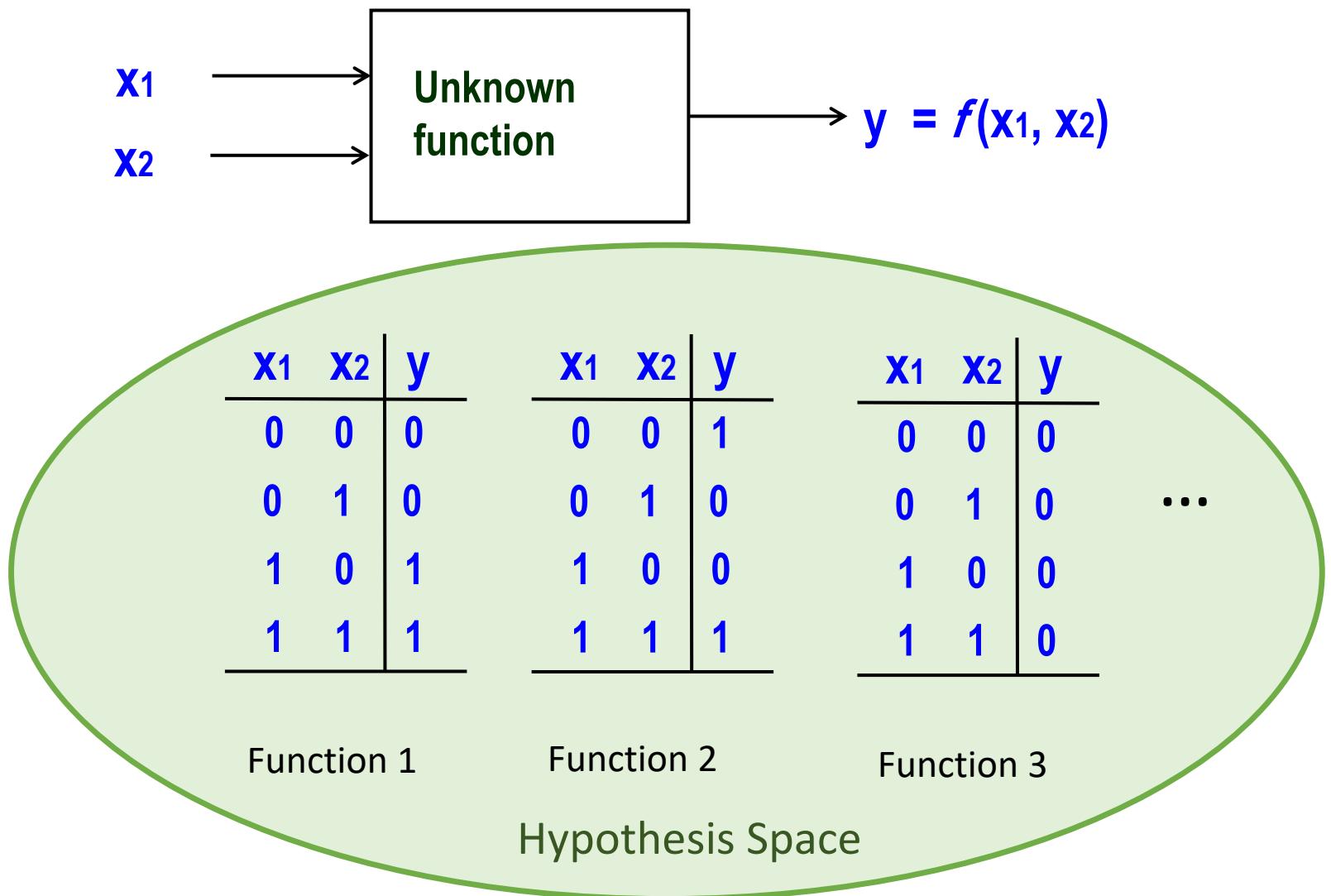
- ❖ What is our instance space?
 - ❖ Gloss: What kind of features are we using?
- ❖ What is our label space?
 - ❖ Gloss: What kind of learning task are we dealing with?
- ❖ What is our hypothesis space?
 - ❖ Gloss: What kind of functions (models) are we learning?
- ❖ What learning algorithm do we use?
 - ❖ 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?

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

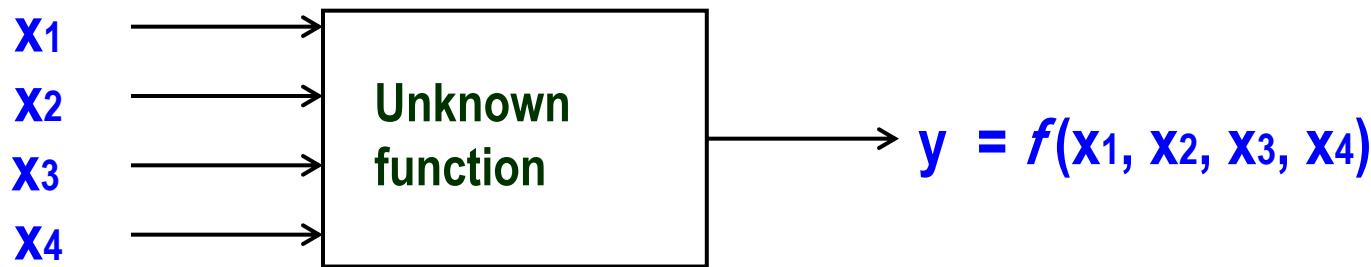


- ❖ We need to choose what *kind* of model we want to learn

Boolean Function



A Learning Problem



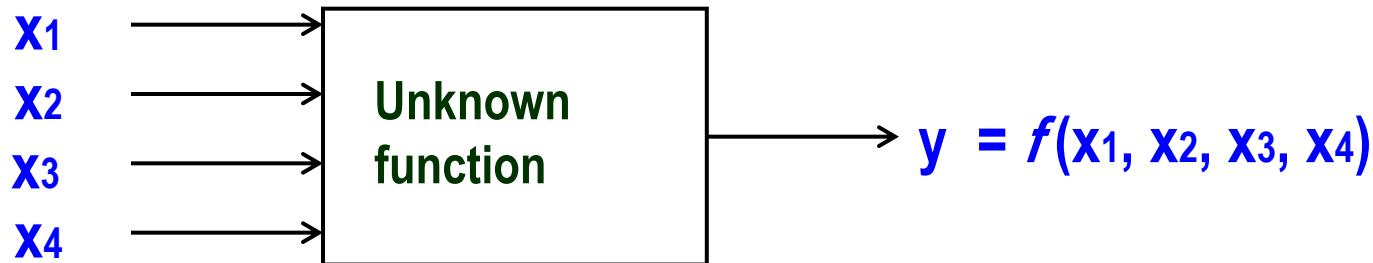
Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Can you learn this function?

What is it?

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

Discussion: A Learning Problem



Example	x_1	x_2	x_3	x_4	y	
1	0	0	1	0	0	Can you learn this function?
2	0	1	0	0	0	What is it?
3	0	0	1	1	1	A function g is consistent to a dataset
4	1	0	0	1	1	$D = \{(x_i, y_i)\}$ if $g(x_i) = y_i, \forall i$
5	0	1	1	0	0	
6	1	1	0	0	0	How many possible functions over four features?
7	0	1	0	1	0	How many function is consistent to D on the left

Hypothesis Space

How many possible functions over four features?

Complete Ignorance:

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

Example	x1	x2	x3	x4	y
0	0	0	0	0	?
0	0	0	0	1	?
0	0	0	1	0	?
0	0	0	1	1	?
0	1	0	0	0	?
0	1	0	0	1	?
0	1	1	0	0	?
0	1	1	1	1	?
1	0	0	0	0	?
1	0	0	0	1	?
1	0	1	0	0	?
1	0	1	1	0	?
1	1	0	0	0	?
1	1	0	0	1	?
1	1	1	0	0	?
1	1	1	1	0	?
1	1	1	1	1	?

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?

which one is the most likely one?

Example	X1	X2	X3	X4	y
0	0	0	0	0	?
0	0	0	0	1	?
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	?
1	0	0	0	0	?
1	0	0	0	1	1
1	0	1	0	0	?
1	0	1	1	1	?
1	1	0	0	0	0
1	1	0	1	1	?
1	1	1	0	1	?
1	1	1	1	0	?
1	1	1	1	1	?

Hypothesis Space

Complete Ignorance:

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

We can correctly
possibly

After

have 2^T possibilities for T

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

Is Learning Possible?

Example	X1	X2	X3	X4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
	1	0	0	0	1
	0	1	0	0	0
	0	0	0	0	0
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space (2)

Simple Rules: **conjunctive rules**

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

e.g., $y = x_2 \wedge x_3$

$y = x_1 \wedge x_2 \wedge X_4$

How large is the hypothesis space?

Hypothesis Space (2)

1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	0	0	1	1	1	1
4	1	0	0	0	1	1
5	1	0	1	1	0	0
6	1	1	1	0	0	0
7	0	1	0	1	0	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
$y=1$		$x_2 \wedge x_3$	
x_1		$x_2 \wedge x_4$	
x_2		$x_3 \wedge x_4$	
x_3		$x_1 \wedge x_2 \wedge x_3$	
x_4		$x_1 \wedge x_2 \wedge x_4$	
$x_1 \wedge x_2$		$x_1 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_3$		$x_2 \wedge x_3 \wedge x_4$	
$x_1 \wedge x_4$		$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	

Hypothesis Space (2)

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

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"

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

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

<u>variables</u>	<u>1-of</u>	<u>2-of</u>	<u>3-of</u>	<u>4-of</u>	<u>variables</u>	<u>1-of</u>	<u>2-of</u>	<u>3-of</u>	<u>4-of</u>
{X ₁ }					{X ₂ , X ₄ }				
{X ₂ }					{X ₃ , X ₄ }				
{X ₃ }					{X ₁ , X ₂ , X ₃ }				
{X ₄ }					{X ₁ , X ₂ , X ₄ }				
{X ₁ , X ₂ }					{X ₁ , X ₃ , X ₄ }				
{X ₁ , X ₃ }					{X ₂ , X ₃ , X ₄ }				
{X ₁ , X ₄ }					{X ₁ , X ₂ , X ₃ , X ₄ }				
{X ₂ , X ₃ }									

Hypothesis Space (3)

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"

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

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

variables	1-of	2-of	3-of	4-of	variables	1-of	2-of	3-of	4-of
{X1}	3	-	-	-	{X2, X4}	2	3	-	-
{X2}	2	-	-	-	{X3, X4}	4	4	-	-
{X3}	1	-	-	-	{X1, X2, X3}	1	3	3	-
{X4}	7	-	-	-	{X1, X2, X4}	2	3	3	-
{X1, X2}	2	3	-	-	{X1, X3, X4}	1	***	3	-
{X1, X3}	1	3	-	-	{X2, X3, X4}	1	5	3	-
{X1, X4}	6	3	-	-	{X1, X2, X3, X4}	1	5	3	3
{X2, X3}	2	3	-	-					

Views of Learning

- ❖ Learning is the removal of our remaining uncertainty:
- ❖ Learning requires guessing a good hypothesis class
 - ❖ Start with a 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
 - ❖ $y=x4 \wedge$ 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:
 - ❖ E.g., Functional representation (n-of-m); Grammars; linear functions; stochastic models

General strategies for Machine Learning

- ❖ Develop flexible hypothesis spaces:
 - ❖ Decision trees, neural networks, nested collections.
- ❖ Develop representation languages for restricted classes of functions:

In either case:

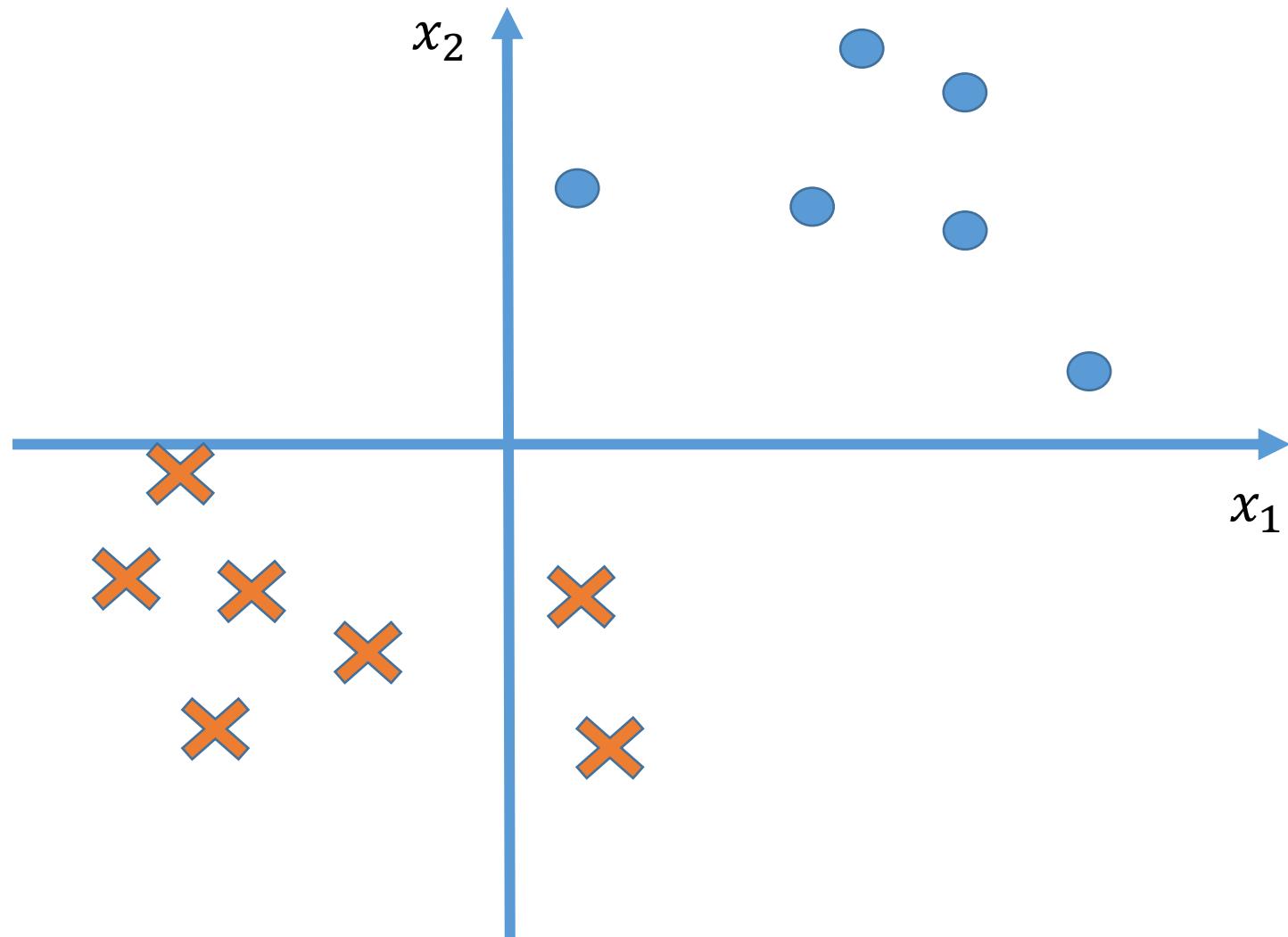
- ❖ Develop algorithms for finding a hypothesis in our hypothesis space, that fits the data
- ❖ And hope that they will generalize well

Hypothesis Space -- Real-Value Features

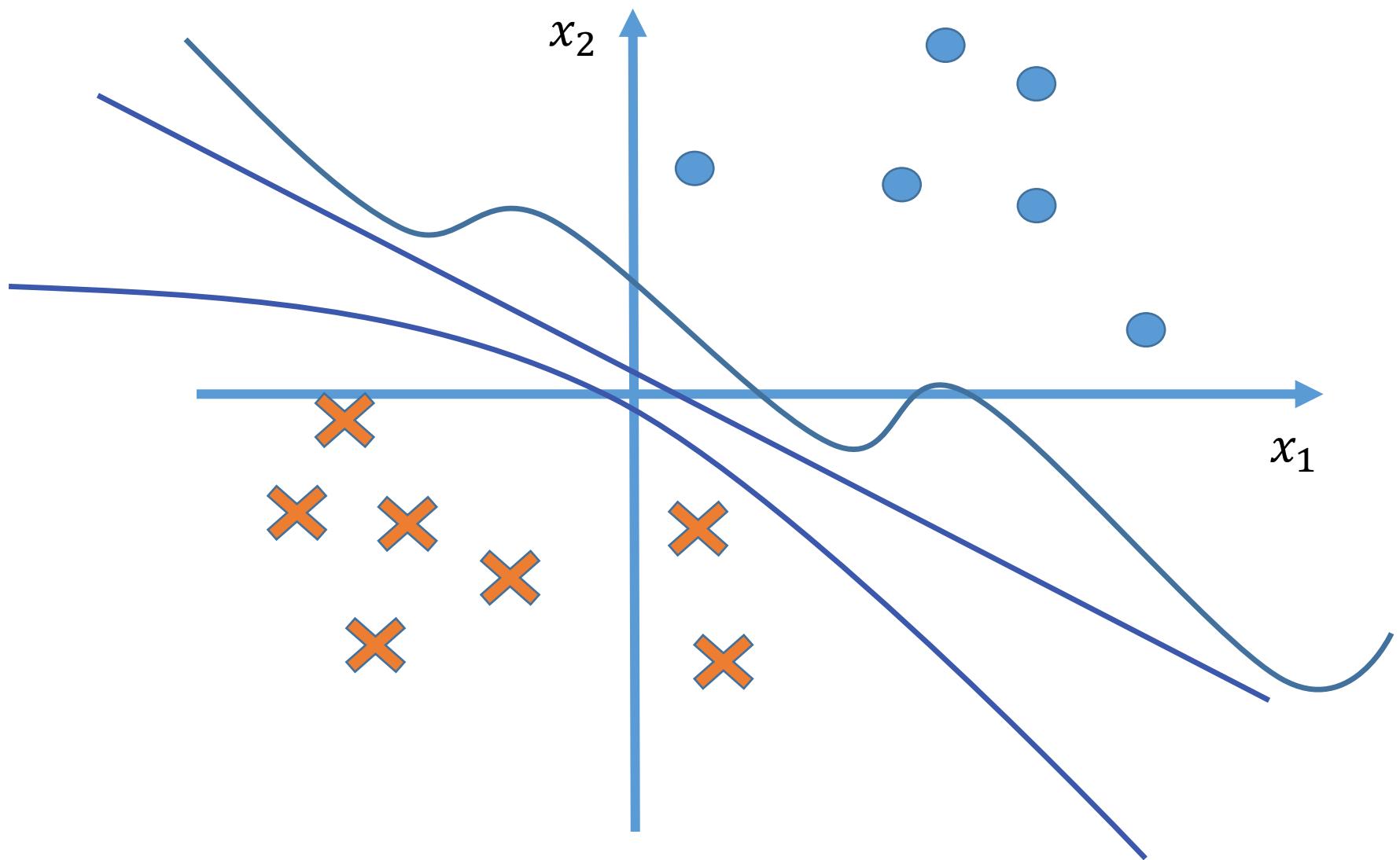
WHAAAAA?!?!



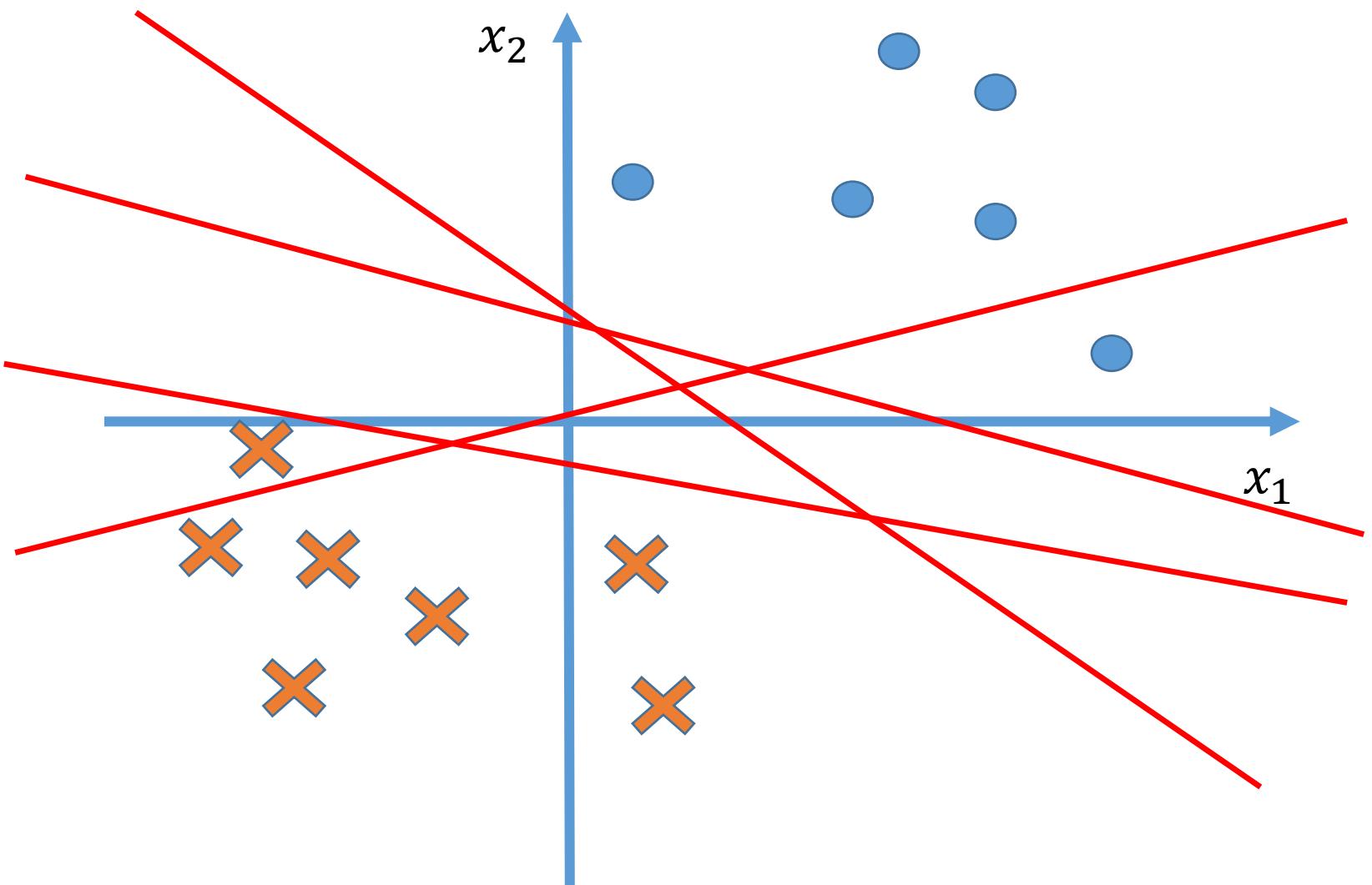
Example problem



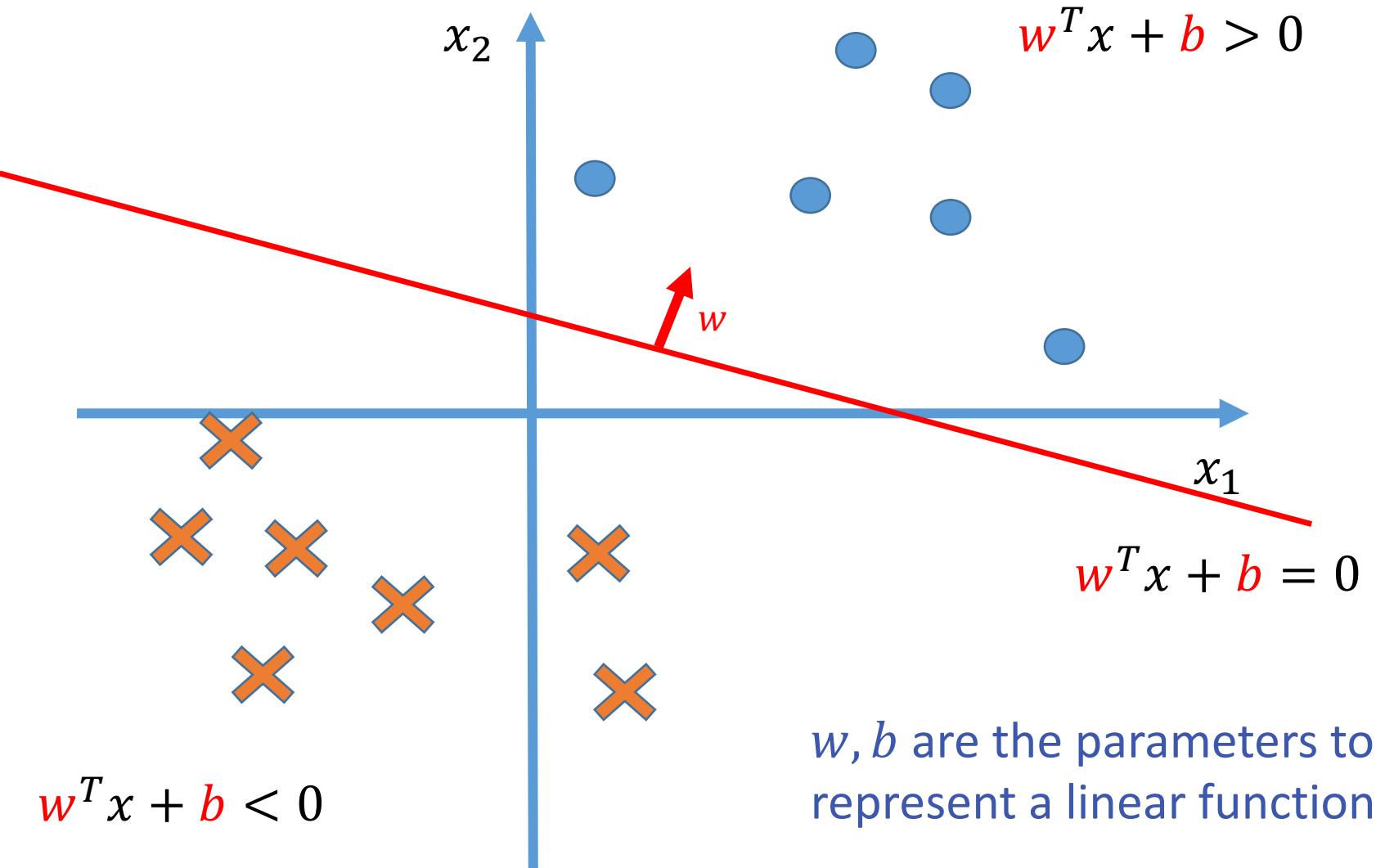
Hypothesis space:



Hypothesis space: linear model



Hypothesis space: linear model



How to learn?

How can we find a good model from the hypothesis space?

Recap: rule-out

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"

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

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

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

Learning is the removal of our
remaining uncertainty

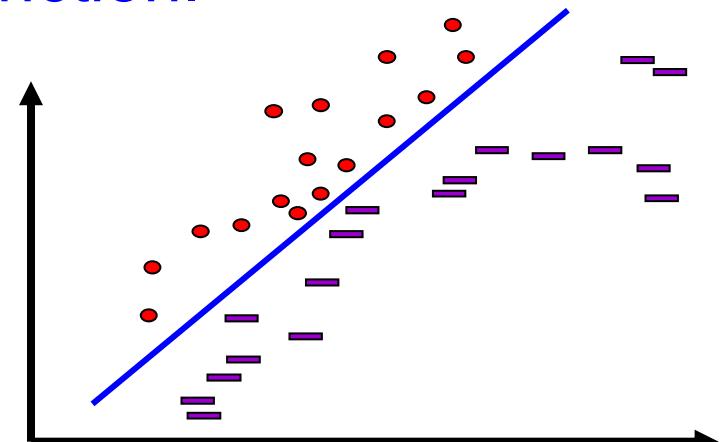
How about linear function?

- ❖ Challenges

- ❖ The hypothesis space contains infinite # functions
- ❖ Several functions are consistent with the data

- ❖ A possibility: Local search

- ❖ Start with a linear threshold function.
- ❖ See how well you are doing.
- ❖ Correct
- ❖ Repeat until you converge.



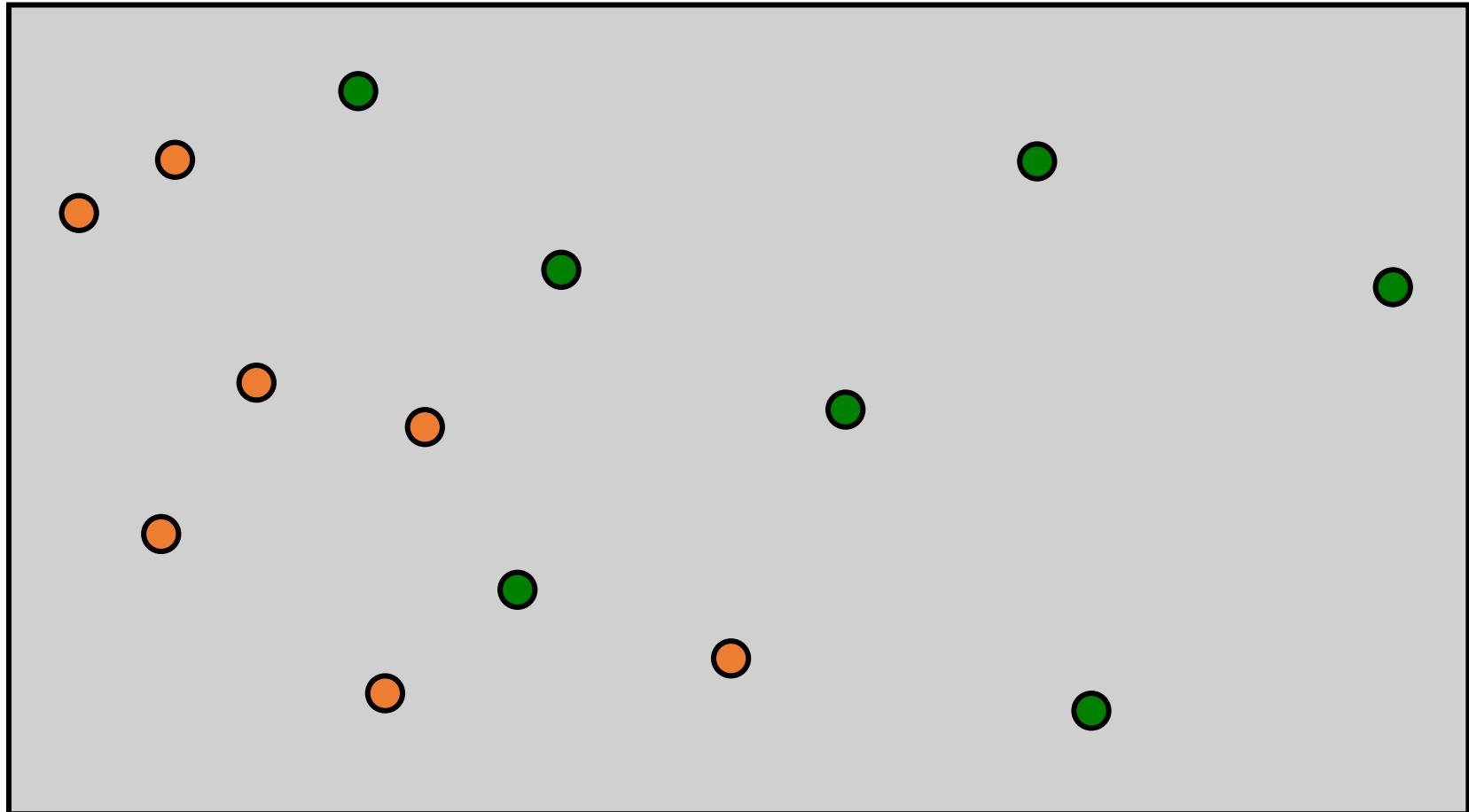
Search can be done in a smarter way

- ❖ There are other ways that do not search directly in the hypotheses space
- ❖ Directly compute the hypothesis by **optimizing** an **objective**
- ❖ Key question: **what is a good classifier?**

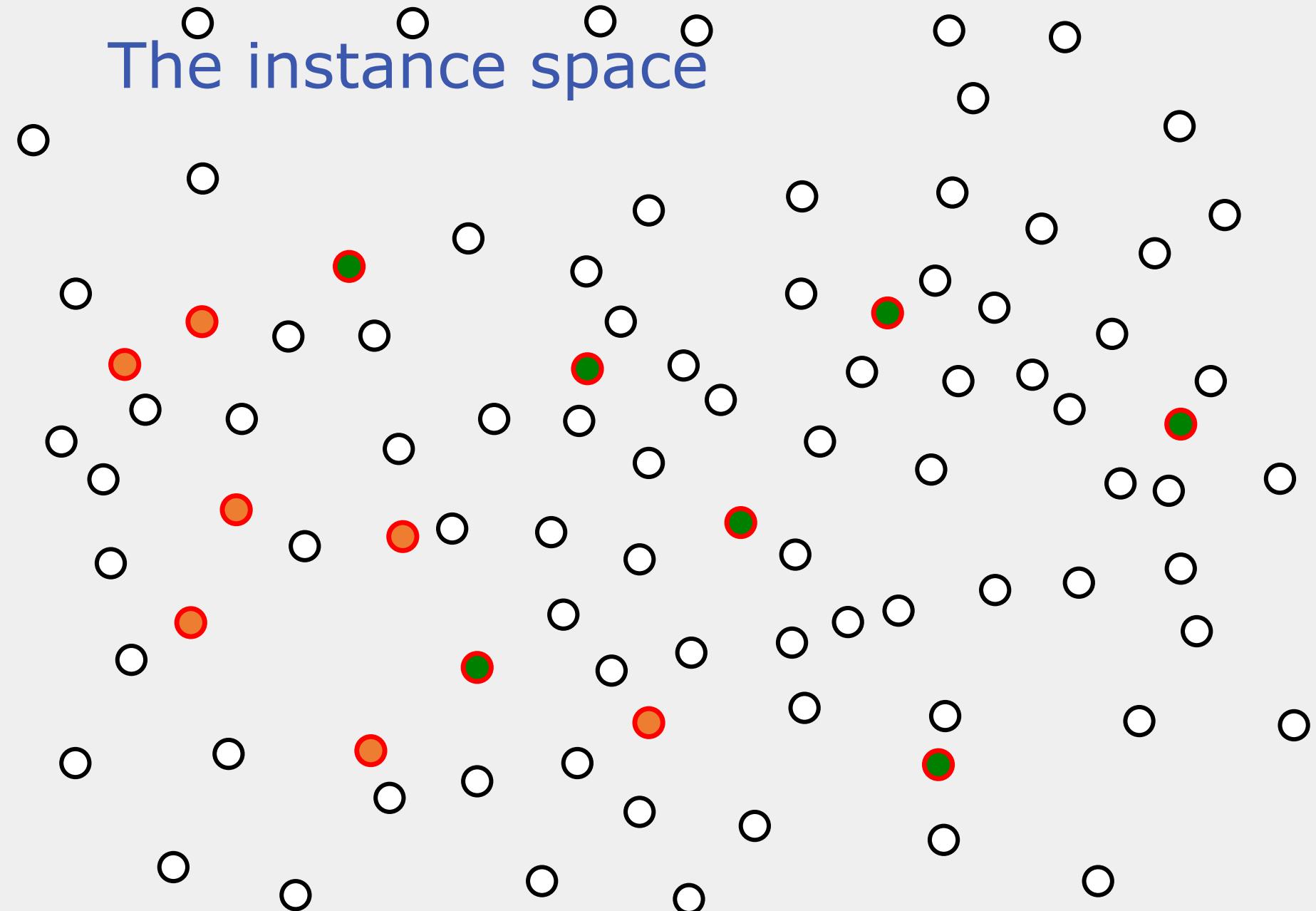


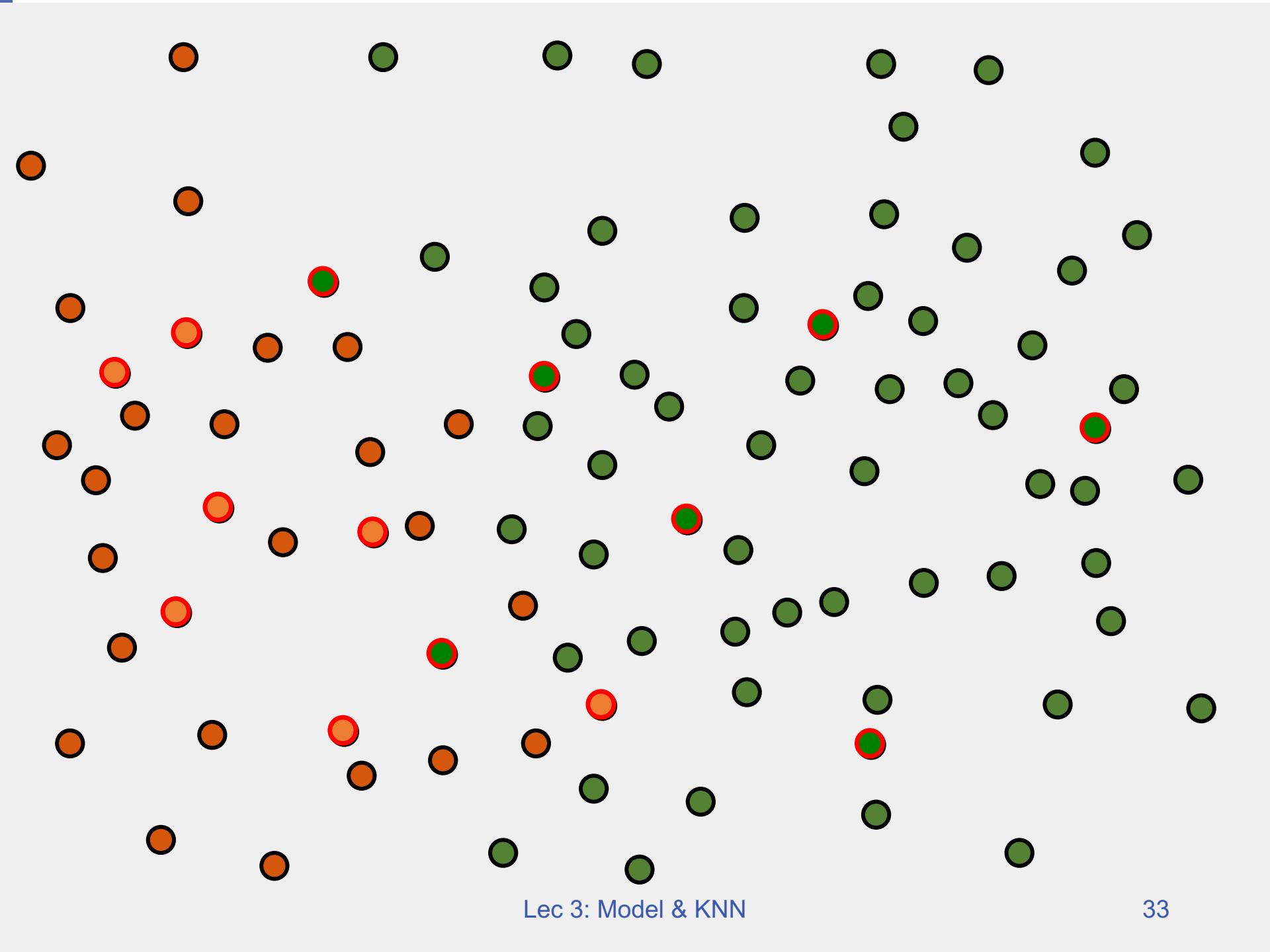
Choose Hypothesis Space

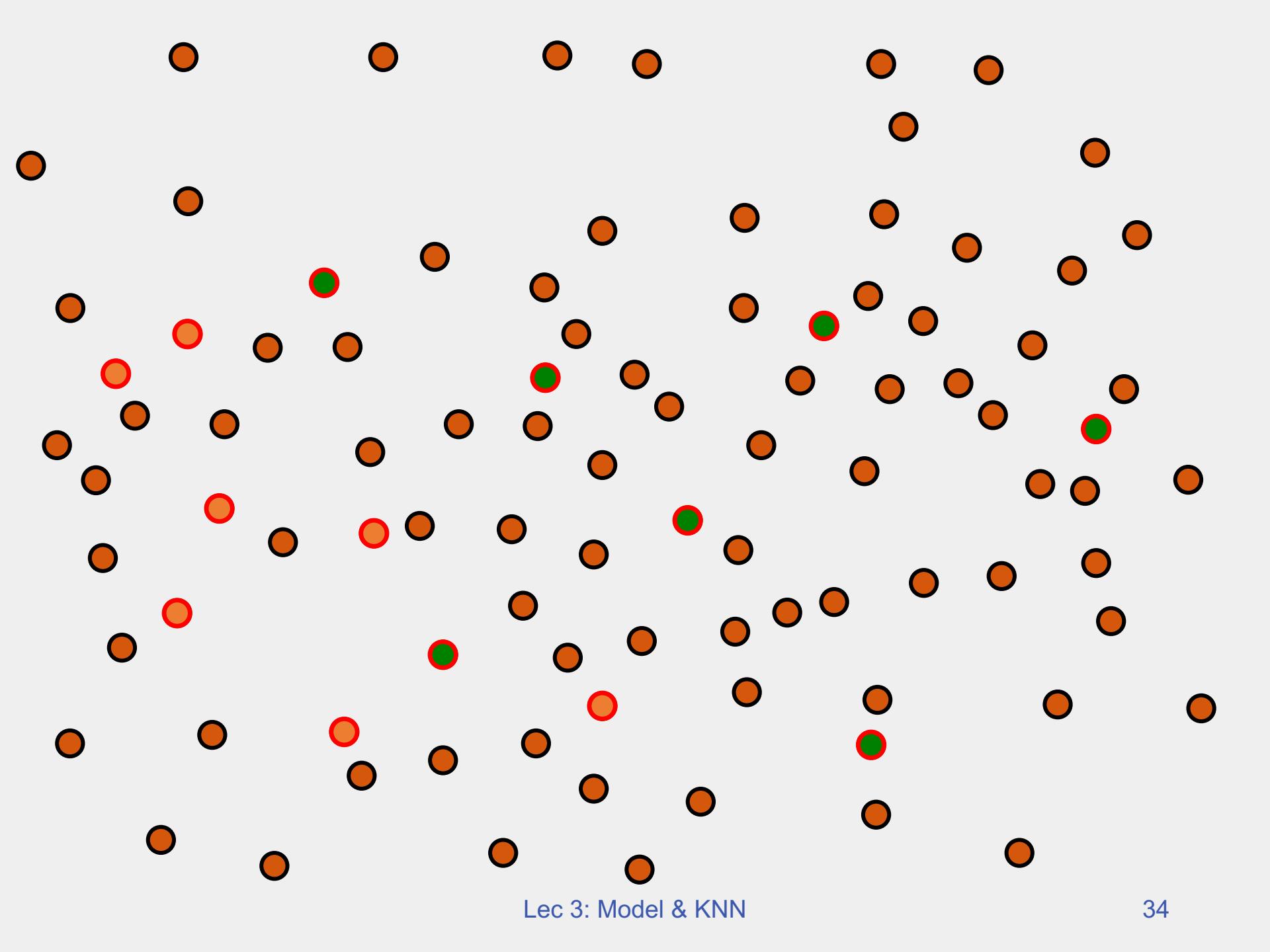
Our training data



The instance space

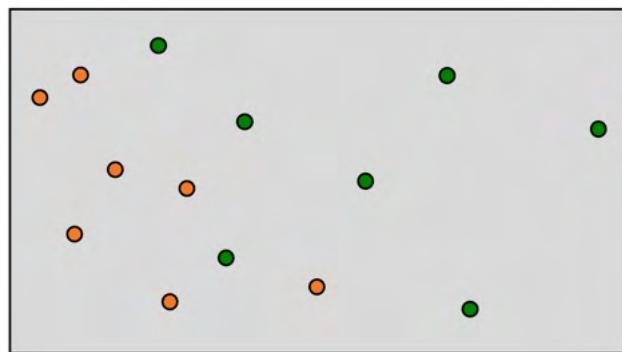




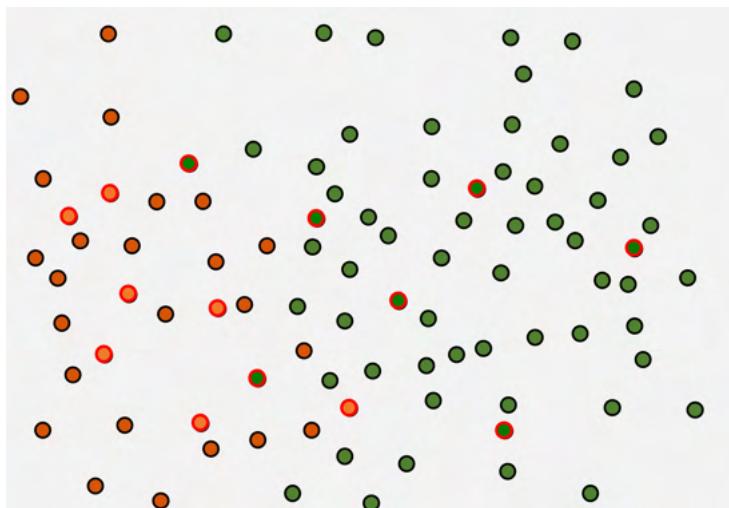


Which one is more likely?

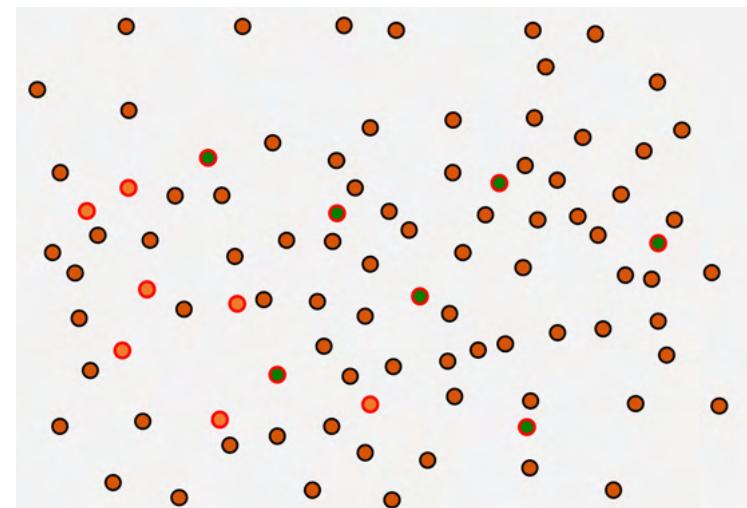
Training set



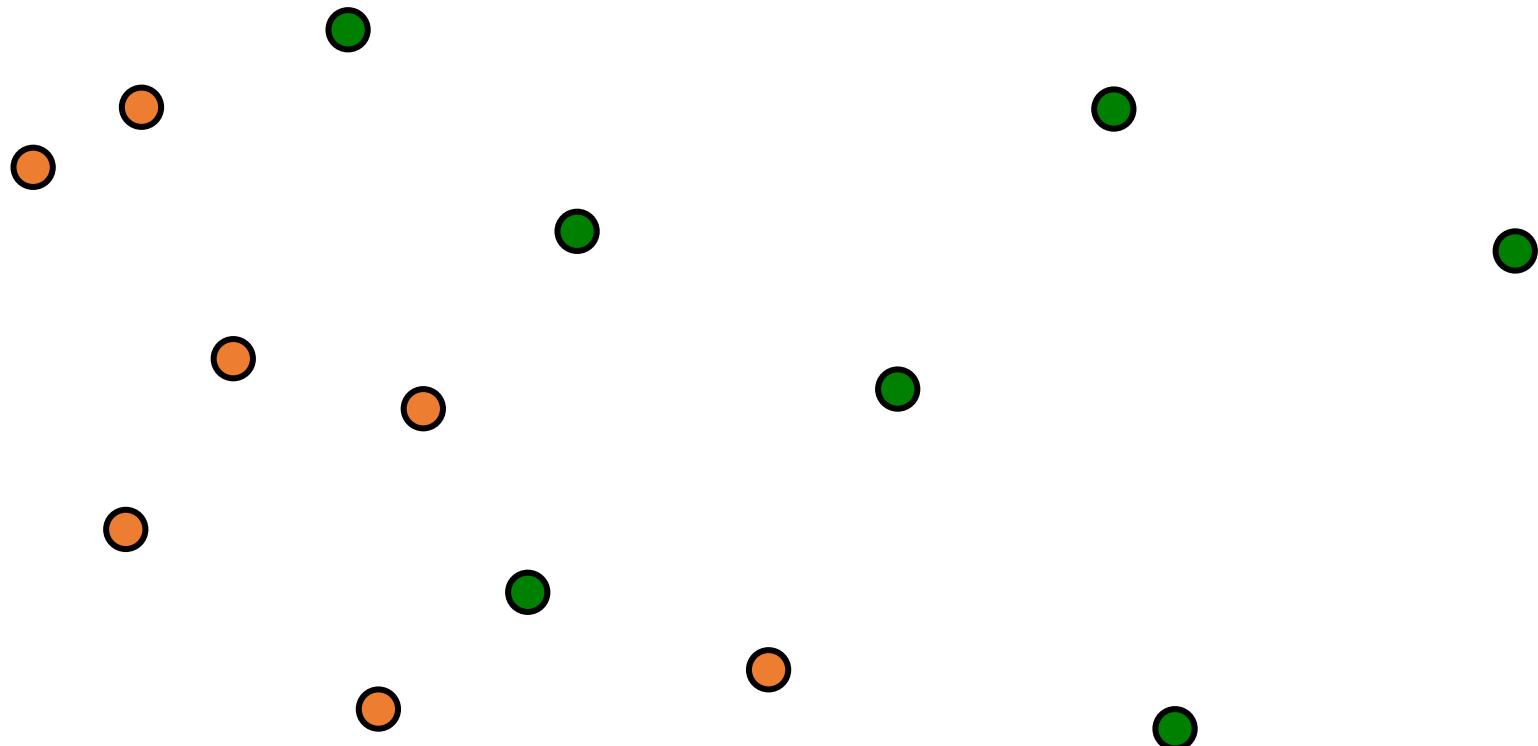
Data distribution A



Data distribution B

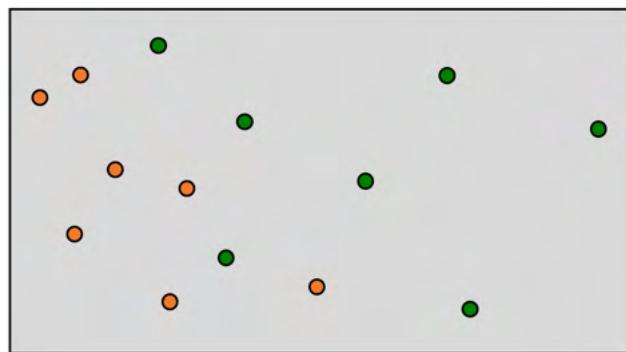


Our training data



Which one is more likely to be a good hypothesis?

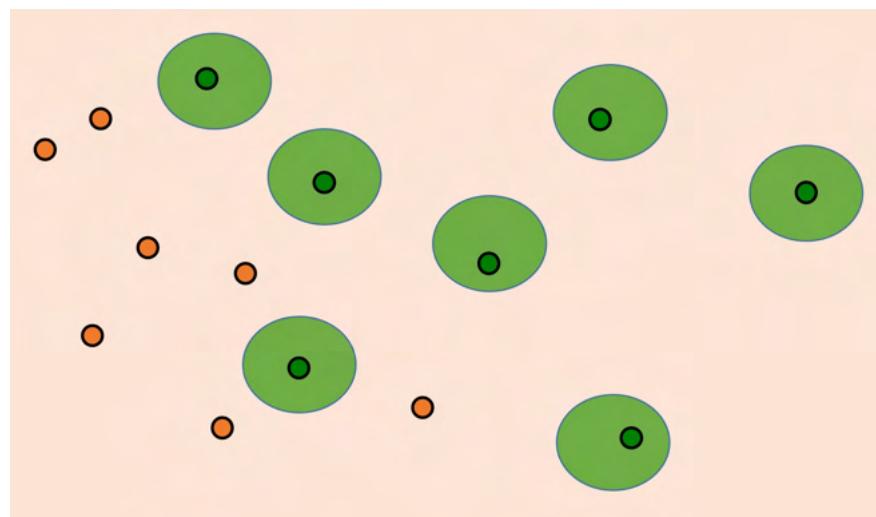
Training set



Hypothesis A

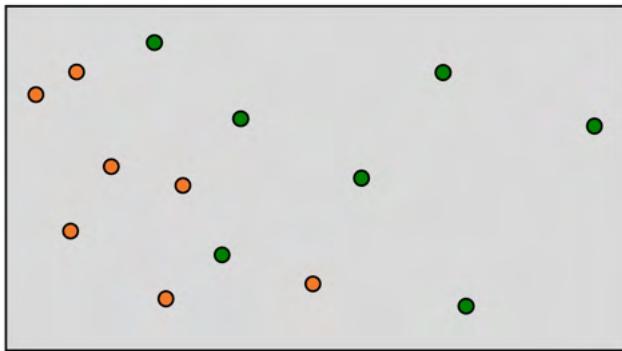


Hypothesis B

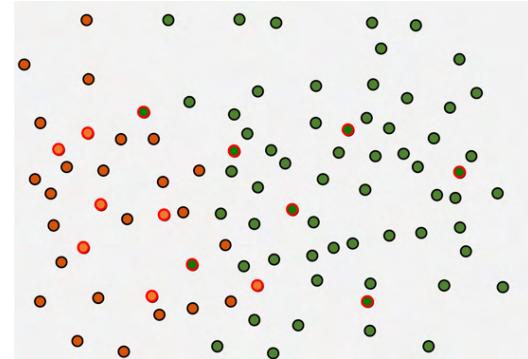


Which one is more likely to be a good hypothesis?

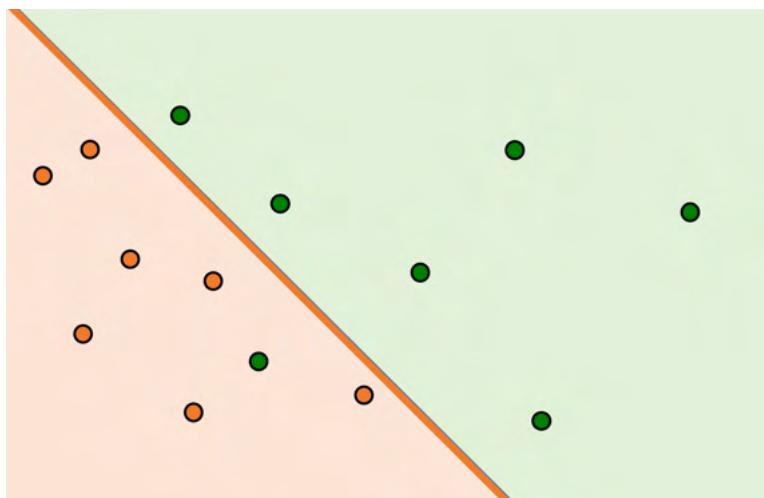
Training set



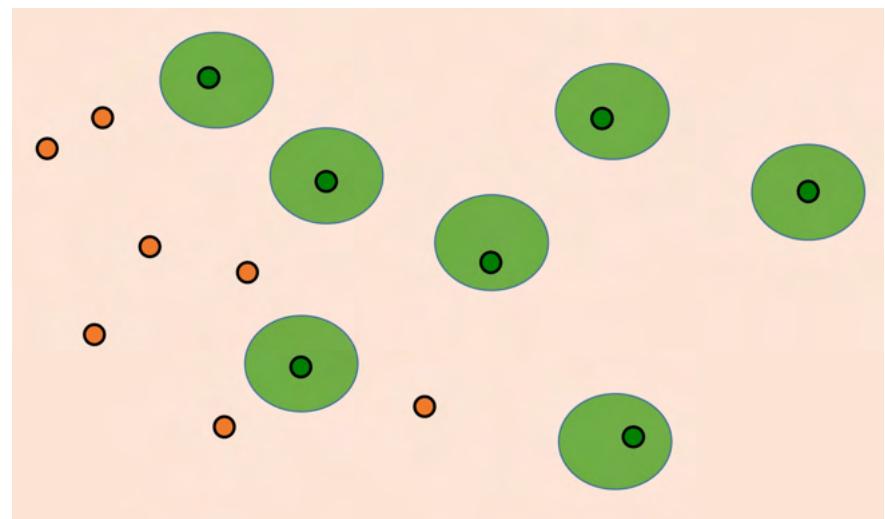
(Likely) Data Distribution



Hypothesis A

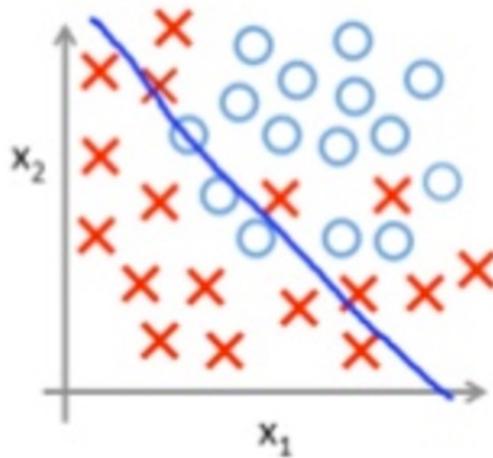


Hypothesis B

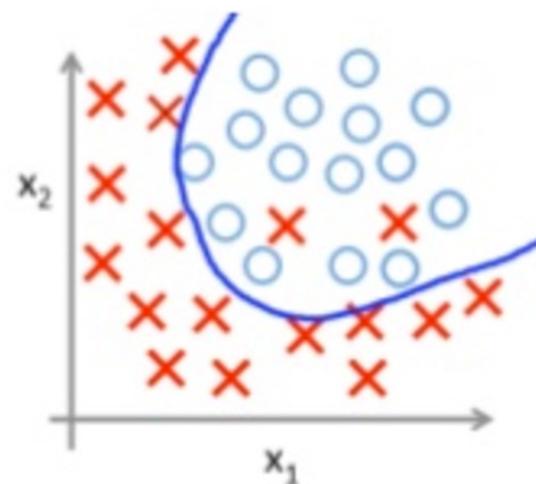


Under-fitting and over-fitting

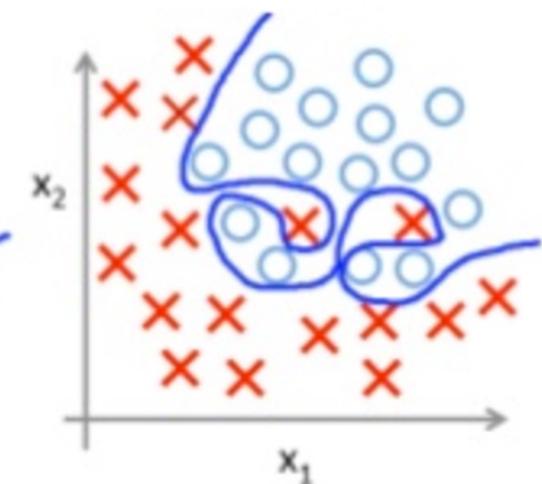
- ❖ Which classifier (blue line) is the best one?



(A)



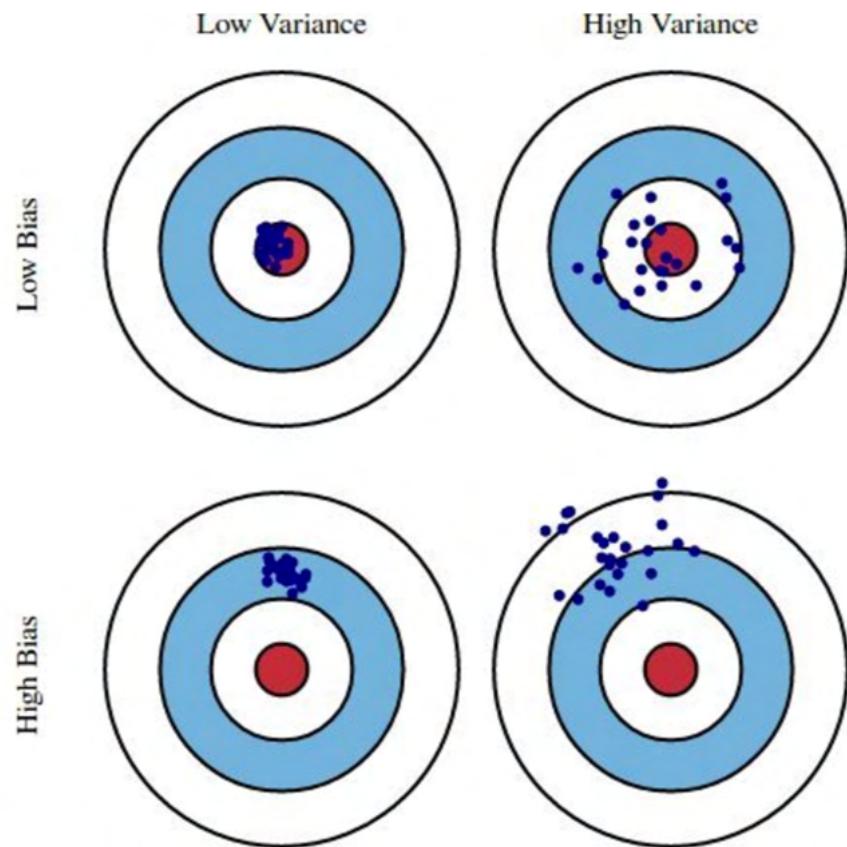
(B)



(C)

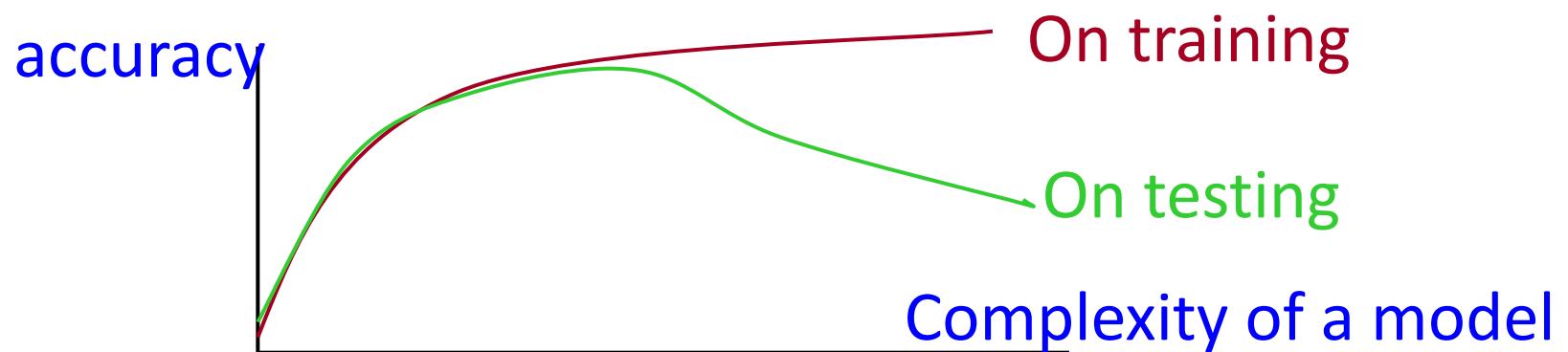
Bias V.S. Variance

- ❖ 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



Overfitting the Data

- ❖ A classifier perform perfectly on the training data may not lead to the best generalization performance.
 - ❖ There may be noise in the training data
 - ❖ The algorithm might be making decisions based on very little data



Prevent overfitting

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

More discussion in later lectures

K-Nearest Neighbor

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Motivation: spam mail

hi Spam x

! marina <marina0279@static.vnpt.vn> to kw Jan 16 (7 days ago) ★ ↗

This message has a from address in static.vnpt.vn but has failed static.vnpt.vn's required tests for authentication. [Learn more](#)

You seem like my type and I would like to know you more! Write me if you are interested, here is my email denisavafursula@rambler.ru and, if you want, I will send some of my photos. Hugs, marina

□ ★ Victoria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email vanesammrenate@	Jan 17
□ ★ Oksana	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elenarzpilse@rambl	Jan 16
□ ★ Oksana	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email venhelgarjxq@raml	Jan 16
□ ★ Victoria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elipetrayk@rambler	Jan 16
□ ★ Natasha	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email lucietmsabine@rar	Jan 16
□ ★ Daria	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email rokcmingrid@rambl	Jan 16
□ ★ Katya	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email michaelavat62@rar	Jan 16
□ ★ Yulia	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email elisa4inge@ramble	Jan 16
□ ★ Veronika	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email venlsajutta@ramble	Jan 16
□ ★ Tanya	hi - You seem like my type and I would like to know you more! Write me if you are interested, here is my email ellairxt6t@rambler.r	Jan 16

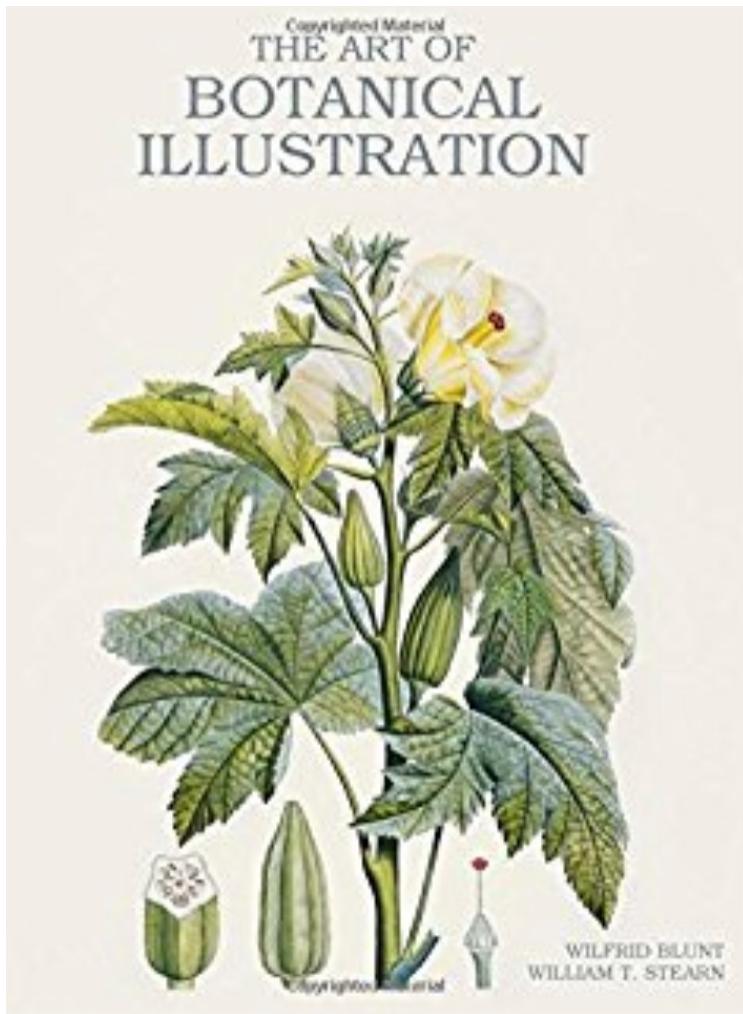
Motivation: Learning from memorization

Recognizing flowers

- ❖ What is this flower called?



Look up at Botanical Illustration Books



Motivation: Learning from memorization

Recognizing flowers

Types of Iris: *setosa*, *versicolor*, and *virginica*



(A)



(B)



(C)



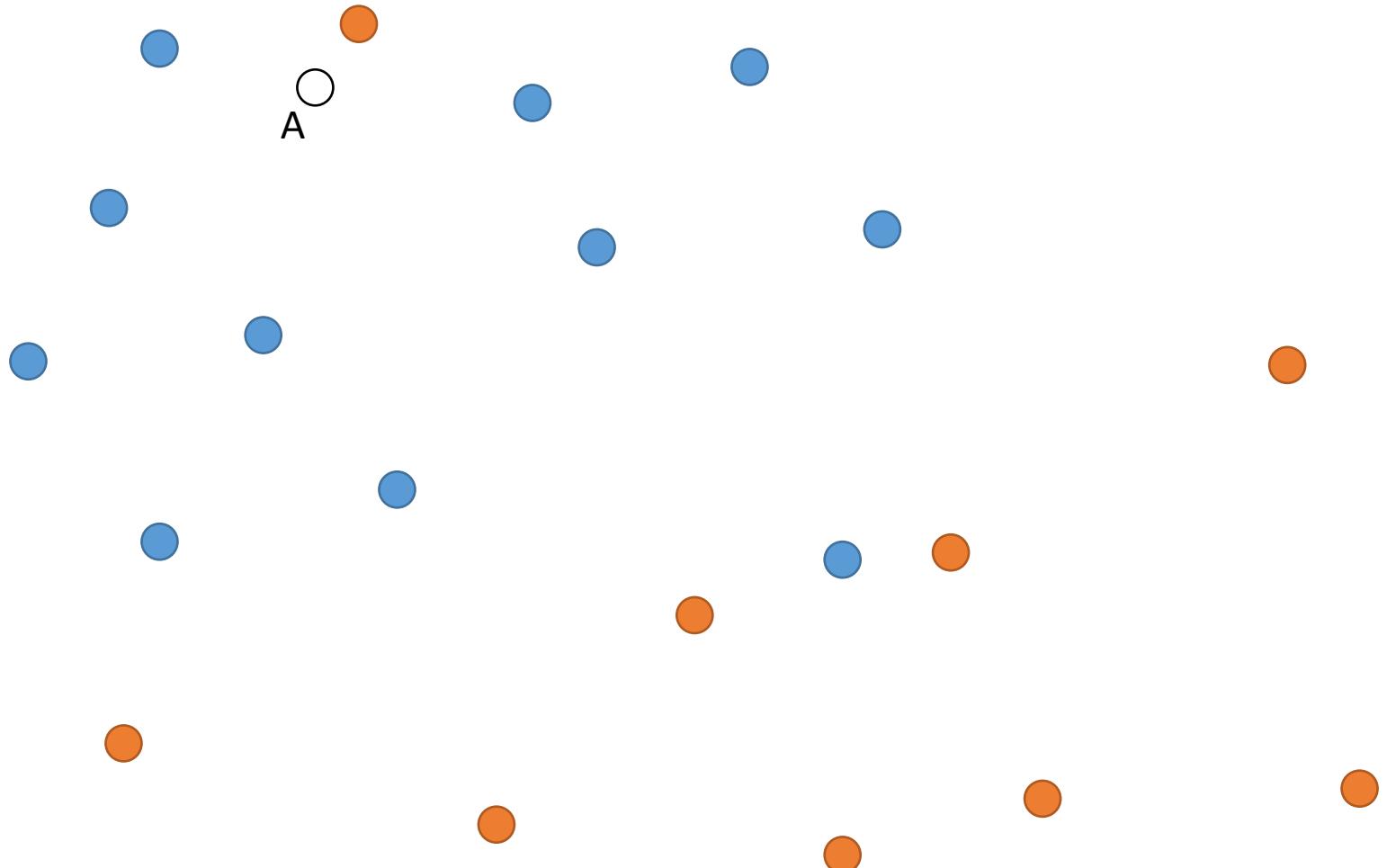
What you will learn in this lecture

- ❖ k-NN algorithm
- ❖ Distance between in the vector space
- ❖ Parameter tuning
- ❖ Decision boundary
- ❖ Curse of the Dimensionality
(Theoretical Limitation)

Nearest Neighbors: The basic version

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction: for a new example \mathbf{x}
 - ❖ Find the training example \mathbf{x}_i that is *closest* to \mathbf{x}
 - ❖ Predict the label of \mathbf{x} to the label y_i associated with \mathbf{x}_i

Example:
How would you color the blank circles?



K-Nearest Neighbors

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to define distance?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Distance between instances

- ❖ How do we measure distances between instances in vector space?
- ❖ In general, a good place to inject knowledge about the domain
- ❖ Behavior of this approach can depend on this

Distance between instances

Numeric features, represented as n dimensional vectors

Distance between instances

Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

Distance between instances

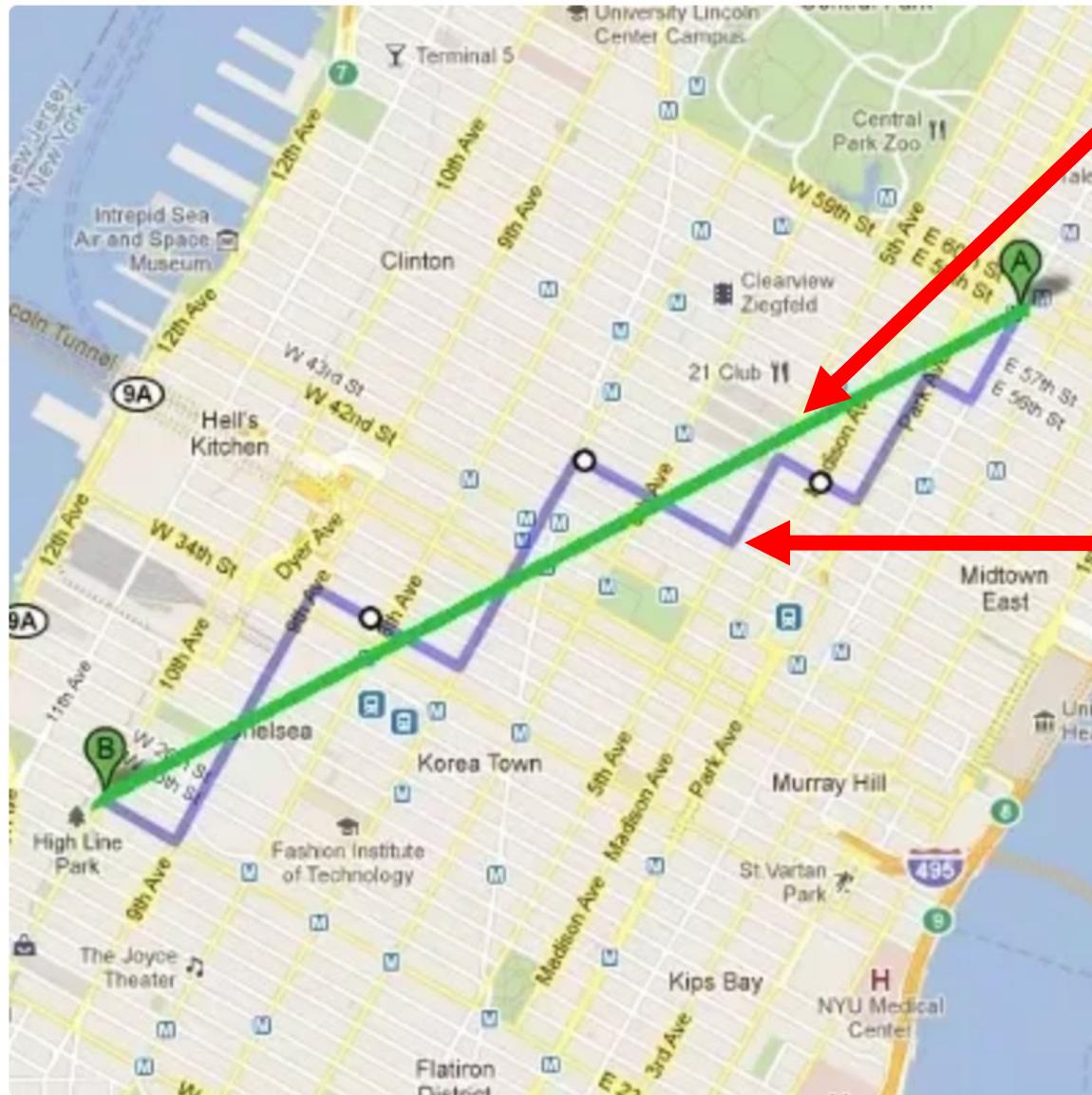
Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$\| \mathbf{x}_1 - \mathbf{x}_2 \|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- ❖ Manhattan distance

$$\| \mathbf{x}_1 - \mathbf{x}_2 \|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$



Euclidean distance

Manhattan distance

Distance between instances

Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- ❖ Manhattan distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

- ❖ L_p -norm

- ❖ Euclidean = L_2

- ❖ Manhattan = L_1

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

$P > 0$

Distance between instances

What about symbolic/categorical features?

Distance between instances

Symbolic/categorical features

Most common distance is the *Hamming distance*

- ❖ Number of bits that are different
- ❖ Or: Number of features that have a different value
- ❖ Example:

\mathbf{X}_1 : {Shape=Triangle, Color=Red, Location=Left, Orientation=Up}

\mathbf{X}_2 : {Shape=Triangle, Color=Blue, Location=Left, Orientation=Down}

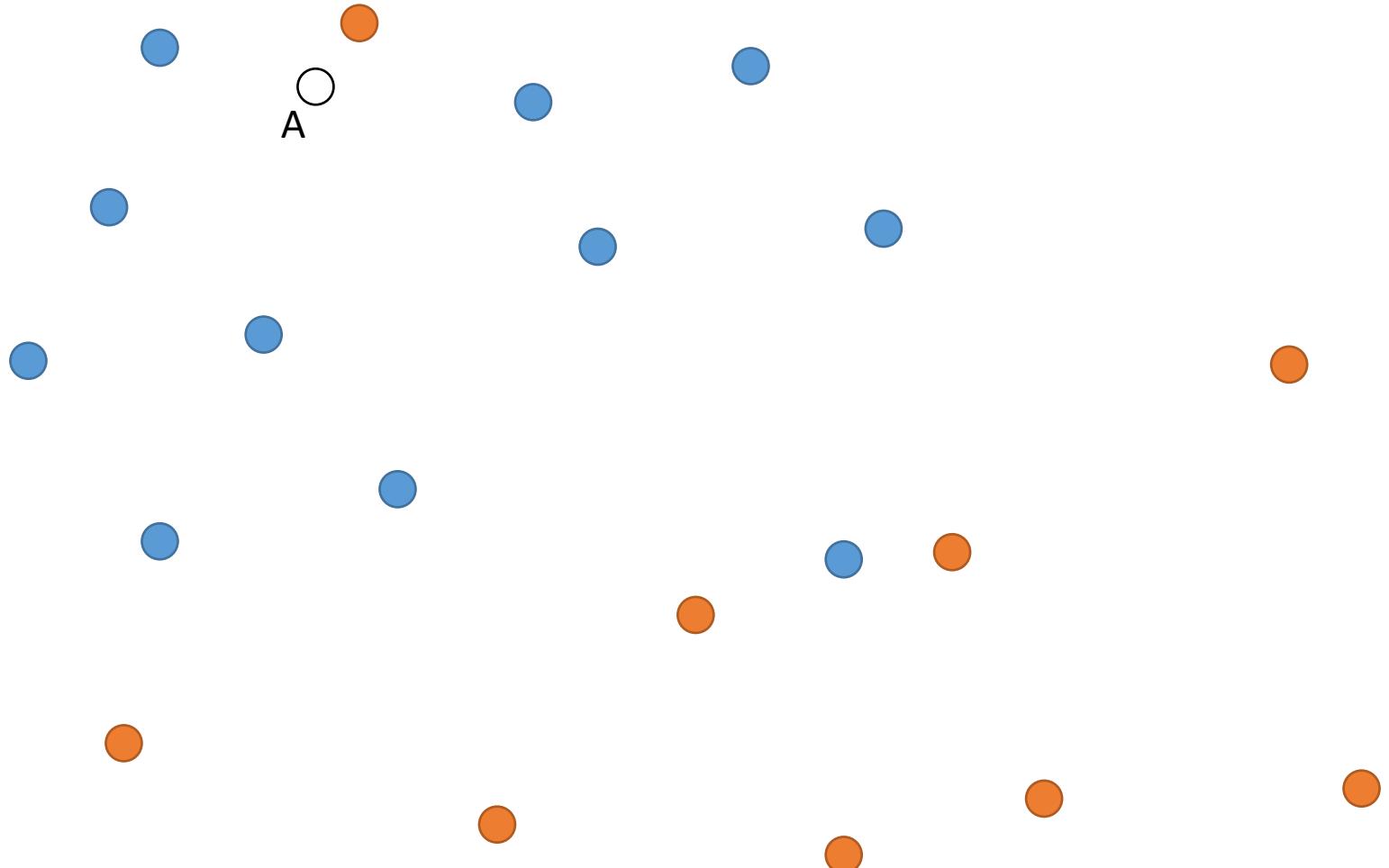
Hamming distance = 2

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to choose k and the distance measure?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Parameter K

What if K is too small?
What if K is too large?



Hyper-parameters in KNN

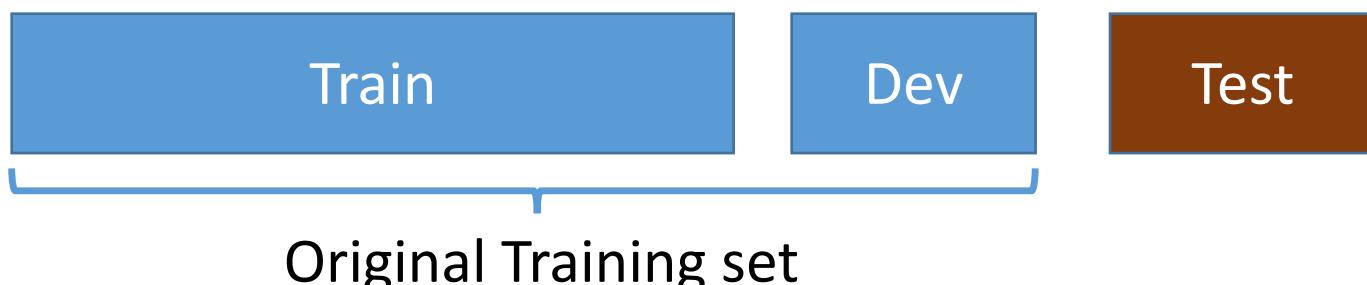
- ❖ (Hyper-)Parameters:
 - ❖ Choosing K (# nearest neighbors)
 - ❖ Distance measurement (e.g., p in the L_p -norm)

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

- ❖ Those are not specified by the algorithm itself
 - ❖ Require **empirical** studies
 - ❖ The best parameter set is **task/dataset-specific**.

Train/Dev/Test splits

- ❖ You are not allowed to look at Text in parameter tuning. Why?
- ❖ Split your training data into two sets:
 - ❖ Train: Training data (often 80-90%)
 - ❖ Dev: Development data (10-20%)
- ❖ Use Dev set (a.k.a. validation set) to find the best parameters



Recipe of train/dev/test

- ❖ For each possible value of the hyper-parameter (e.g., $M = 1, 2, 3, \dots, 10$)
 - ❖ Train a model using D^{TRAIN}
 - ❖ Evaluate the performance on D^{DEV}
- ❖ Choose the model parameter with the best performance on D^{DEV}
- ❖ (optional) Re-train the model on $D^{TRAIN} \cup D^{DEV}$ with the best parameter set
- ❖ Evaluate the model on D^{TEST}

Tradeoff between Train v.s. Dev Size

- ❖ Consider having 120 data points; 20 data points are reserved for testing
 - ❖ What is the best way to split the remainder?
 - ❖ (A) # instances: Train: 95 Dev: 5 ?
 - ❖ (B) # instances: Train: 60 Dev: 40 ?

Trade off in Train/Dev splits

- ❖ Large Train, small Dev
(e.g., #train = 95, #dev = 5)



Result on dev is not represented

- ❖ Small Train, Large Dev
(e.g., #train = 60, #dev = 40)



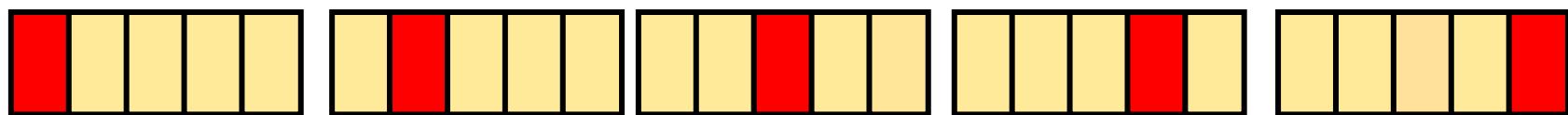
No enough data to train a model

N-fold cross validation

- ❖ Instead of a single training-dev split:



- ❖ Split data into N equal-sized parts

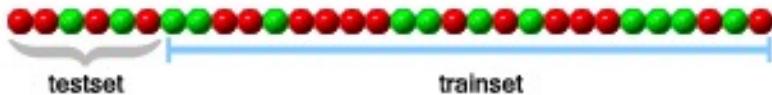


- ❖ Train and test N different classifiers
- ❖ Report average accuracy and standard deviation of the accuracy

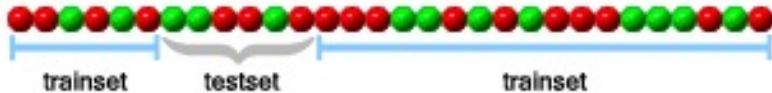
Example

ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:

1-ST FOLD:



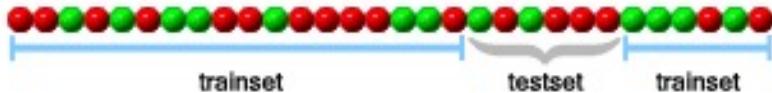
2-ND FOLD:



3-RD FOLD:



4-TH FOLD:



5-TH FOLD:



Parameter 1

Accuracy: 100%

Accuracy: 50%

Accuracy: 50%

Accuracy: 100%

Accuracy: 100%

Parameter 2

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 50%

Avg Accuracy: 80%

Avg Accuracy: 90%

<https://genome.tugraz.at/proclassify/help/pages/XV.html>

Finding parameters based on cross validation

- ❖ Given D^{TRAIN} and D^{TEST} , for each possible value of the hyper-parameter (e.g., $K = 1, 2, 3, \dots, 10$)
 - ❖ Conduct cross validation on D^{TRAIN} with parameter K
- ❖ Choose the model parameter with the best cross validation performance
- ❖ (Optional) Re-train the model on D^{TRAIN} with the best parameter set
- ❖ Evaluate the model on D^{TEST}

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to aggregate the information and make the prediction?

K-Nearest Neighbors

- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.
 - ❖ For classification: Every neighbor votes on the label.
Predict the most frequent label among the neighbors.
 - ❖ For regression:
Predict the mean value



Q: other alternatives?

K-Nearest Neighbors

- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.
 - ❖ For classification: Every neighbor votes on the label.
Predict the most frequent label among the neighbors.
 - ❖ For regression:
Predict the mean value



Neighbors' labels could be weighted by their distance
Related to Kernel method

Issues in designing KNN algorithm (computation/algorithim)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to store data?

How to find the closest points?

Issues in designing KNN algorithm (computation/algorithms)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the

How to store data?

How to find the closest points?

This is an important research topic, but I will not cover it in this class.
Reference: e.g. K-d tree (https://en.wikipedia.org/wiki/K-d_tree)

K-Nearest Neighbor

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to define distance?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Distance between instances

Numeric features, represented as n dimensional vectors

- ❖ Euclidean distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{i=1}^n (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- ❖ Manhattan distance

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 = \sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$

- ❖ L_p -norm

- ❖ Euclidean = L_2

- ❖ Manhattan = L_1

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

$p > 0$

Distance between instances

How about symbolic/categorical features?

Distance between instances

Symbolic/categorical features

Most common distance is the *Hamming distance*

- ❖ Number of bits that are different
- ❖ Or: Number of features that have a different value
- ❖ Example:

\mathbf{X}_1 : {Shape=Triangle, Color=Red, Location=Left,
Orientation=Up}

\mathbf{X}_2 : {Shape=Triangle, Color=Blue, Location=Left,
Orientation=Down}

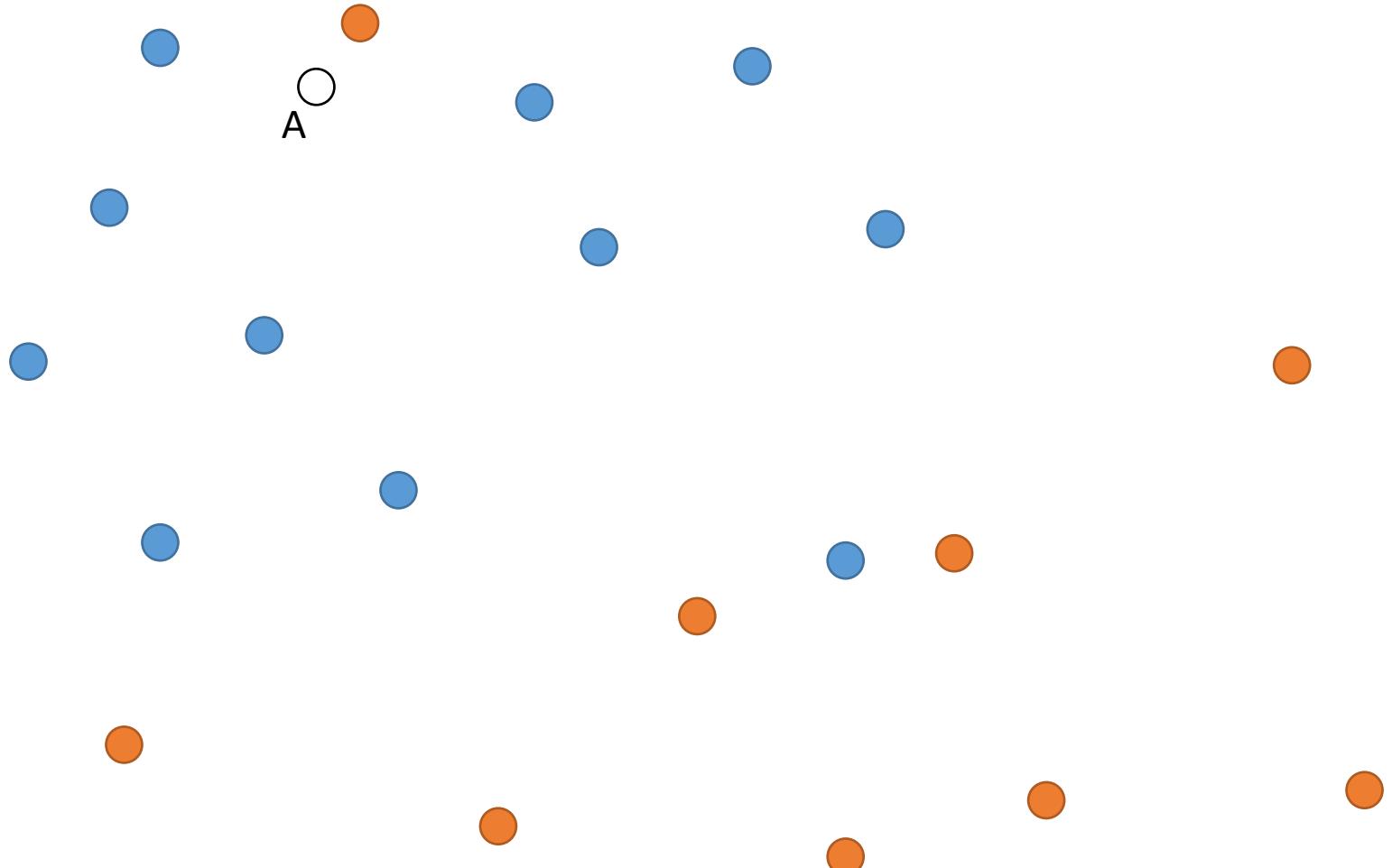
Hamming distance = 2

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to choose k and the distance measure?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

Parameter K

What if K is too small?
What if K is too large?



Hyper-parameters in KNN

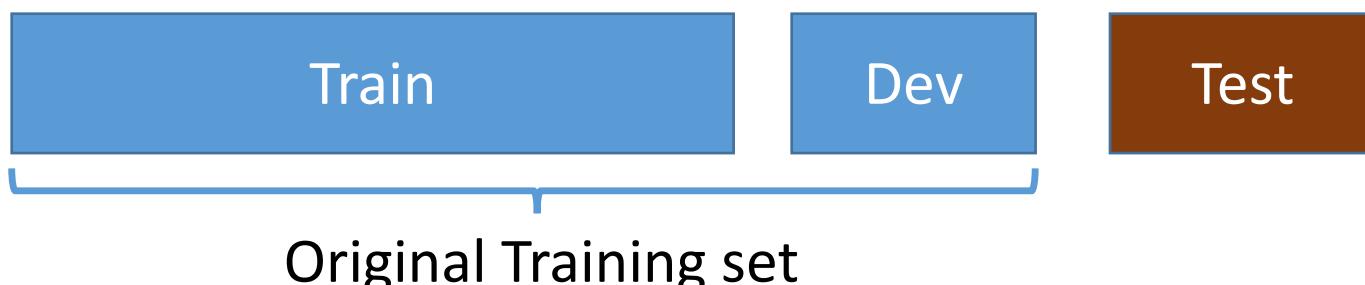
- ❖ (Hyper-)Parameters:
 - ❖ Choosing K (# nearest neighbors)
 - ❖ Distance measurement (e.g., p in the L_p -norm)

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_p = \left(\sum_{i=1}^n |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p \right)^{\frac{1}{p}}$$

- ❖ Those are not specified by the algorithm itself
 - ❖ Require **empirical** studies
 - ❖ The best parameter set is **task/dataset-specific**.

Train/Dev/Test splits

- ❖ You are not allowed to look at Test in parameter tuning. Why?
- ❖ Split your training data into two sets:
 - ❖ Train: Training data (often 80-90%)
 - ❖ Dev: Development data (10-20%)
- ❖ Use Dev set (a.k.a. validation set) to find the best parameters



Recipe of train/dev/test

- ❖ For each possible value of the hyper-parameter (e.g., $M = 1, 2, 3, \dots, 10$)
 - ❖ Train a model using D^{TRAIN}
 - ❖ Evaluate the performance on D^{DEV}
- ❖ Choose the model parameter with the best performance on D^{DEV}
- ❖ (optional) Re-train the model on $D^{TRAIN} \cup D^{DEV}$ with the best parameter set
- ❖ Evaluate the model on D^{TEST}

Tradeoff between Train v.s. Dev Size

- ❖ Consider having 120 data points; 20 data points are reserved for testing
 - ❖ What is the best way to split the remainder?
 - ❖ (A) # instances: Train: 95 Dev: 5 ?
 - ❖ (B) # instances: Train: 60 Dev: 40 ?

Trade off in Train/Dev splits

- ❖ Large Train, small Dev
(e.g., #train = 95, #dev = 5)



Result on dev is not represented

- ❖ Small Train, Large Dev
(e.g., #train = 60, #dev = 40)



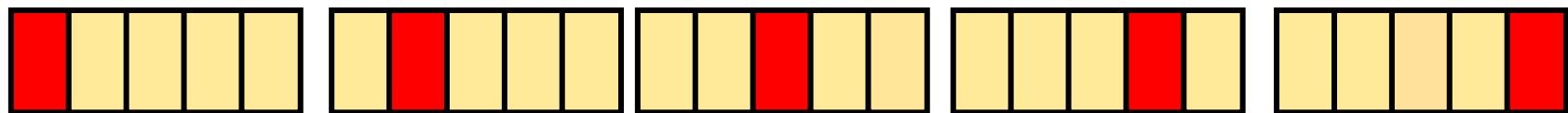
No enough data to train a model

N-fold cross validation

- ❖ Instead of a single training-dev split:



- ❖ Split data into N equal-sized parts

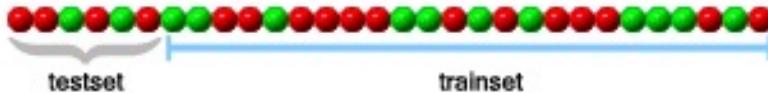


- ❖ Train and test N different classifiers
- ❖ Report average accuracy and standard deviation of the accuracy

Example

ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:

1-ST FOLD:



2-ND FOLD:



3-RD FOLD:



4-TH FOLD:



5-TH FOLD:



Parameter 1

Accuracy: 100%

Accuracy: 50%

Accuracy: 50%

Accuracy: 100%

Accuracy: 100%

Parameter 2

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 100%

Accuracy: 50%

<https://genome.tugraz.at/proclassify/help/pages/XV.html>

Avg Accuracy: 80%

Avg Accuracy: 90%

Finding parameters based on cross validation

- ❖ Given D^{TRAIN} and D^{TEST} , for each possible value of the hyper-parameter (e.g., $K = 1, 2, 3, \dots, 10$)
 - ❖ Conduct cross validation on D^{TRAIN} with parameter K
- ❖ Choose the model parameter with the best cross validation performance
- ❖ (Optional) Re-train the model on D^{TRAIN} with the best parameter set
- ❖ Evaluate the model on D^{TEST}

Issues in designing KNN algorithm (Modeling)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to aggregate the information and make the prediction?

K-Nearest Neighbors

- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.
 - ❖ For classification: Every neighbor votes on the label.
Predict the most frequent label among the neighbors.
 - ❖ For regression:
Predict the mean value



Q: other alternatives?

K-Nearest Neighbors

- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k *closest* training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.
 - ❖ For classification: Every neighbor votes on the label.
Predict the most frequent label among the neighbors.
 - ❖ For regression:
Predict the mean value



Neighbors' labels could be weighted by their distance
Related to Kernel method

Issues in designing KNN algorithm (computation/algorithim)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to store data?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the label of \mathbf{x} using these k points.

How to find the closest points?

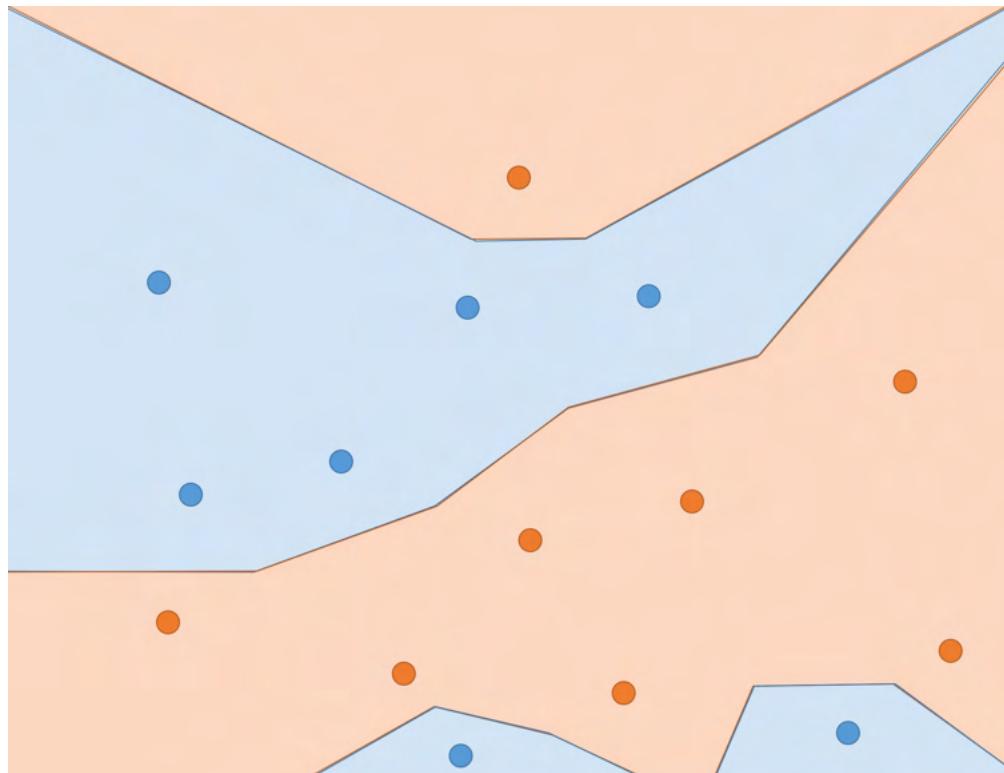
Issues in designing KNN algorithm (computation/algorithms)

- ❖ Training examples are vectors \mathbf{x}_i associated with a label y_i
 - ❖ E.g. \mathbf{x}_i = a feature vector for an email, y_i = SPAM
- ❖ Learning: Just store all the training examples
 - How to store data?
- ❖ Prediction for a new example \mathbf{x}
 - ❖ Find the k closest training examples to \mathbf{x}
 - ❖ Construct the

How to find the closest points?

This is an important research topic, but I will not cover it in this class.
Reference: e.g. K-d tree (https://en.wikipedia.org/wiki/K-d_tree)

Decision Boundary



The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

The decision boundary for KNN

Is the K nearest neighbors algorithm explicitly building a function?

- ❖ **No**, it never forms an explicit hypothesis

Given a training set what is the implicit function that is being computed

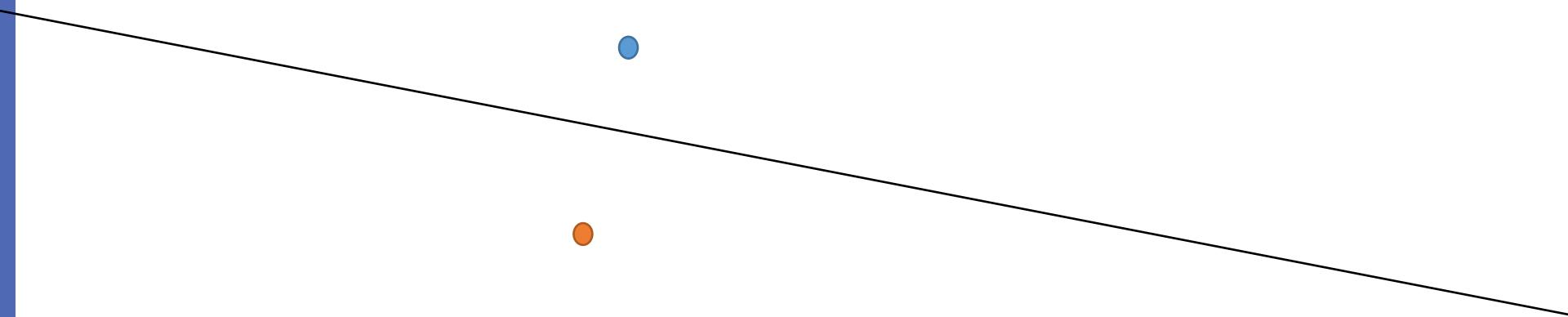
Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



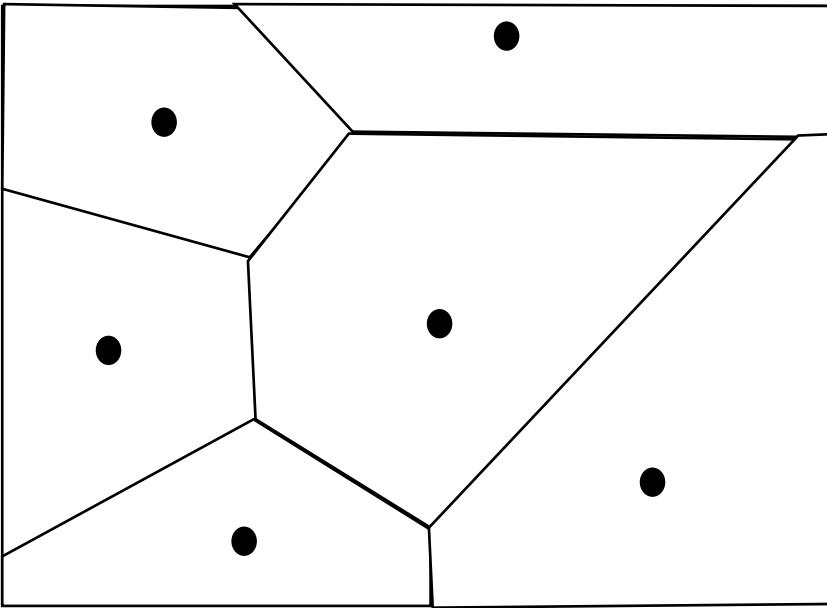
Exercise

If you have only two training points, what will the decision boundary for 1-nearest neighbor be?



- ❖ A line bisecting the two points

The Voronoi Diagram (w/ Euclidean distance)

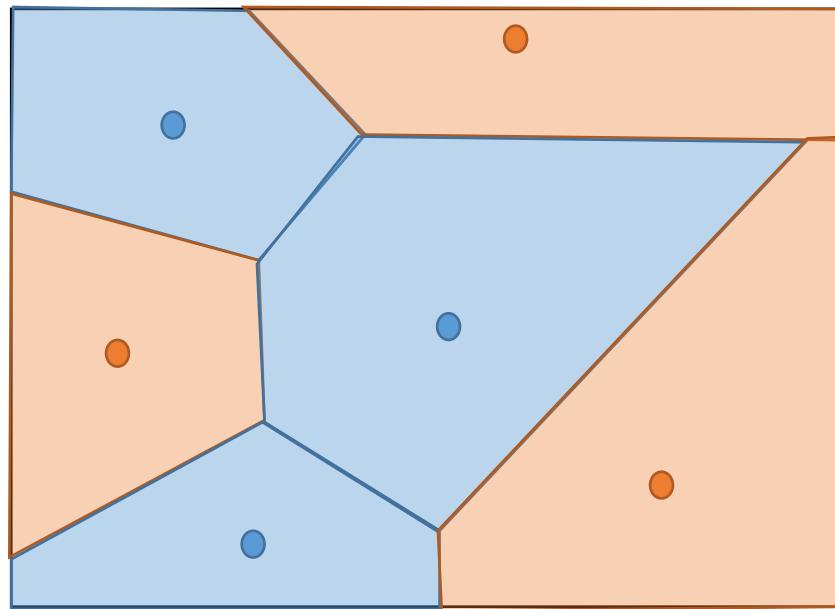


For any point x in a training set S ,
the **Voronoi Cell** of x is a polytope consisting of all points closer to x than any other points in S

The **Voronoi diagram** is the union of all Voronoi cells

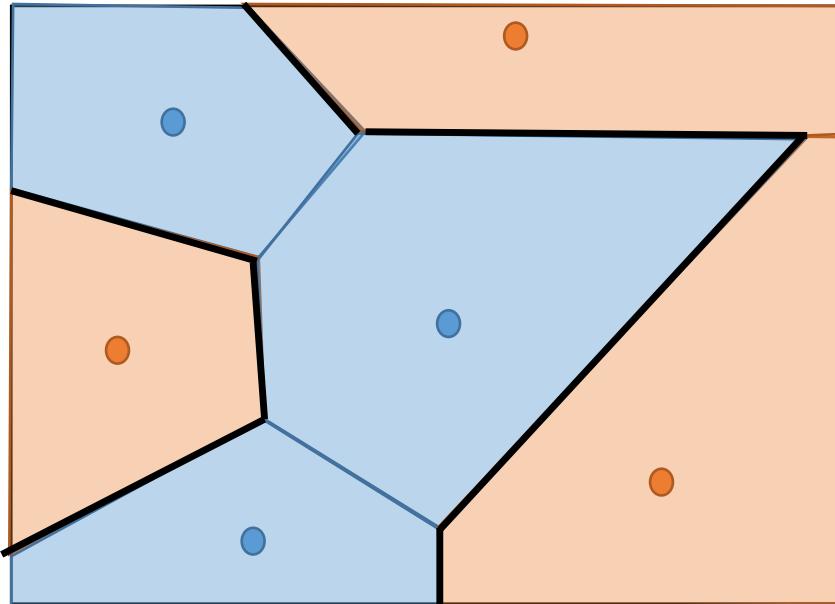
- Covers the entire space

Voronoi diagrams of training examples



Voronoi diagrams colored with the output label
Picture uses Euclidean distance with 1-nearest neighbor.

Decision boundary of 1-NN



What about K-nearest neighbors?

Also partitions the space, but much more complex
decision boundary

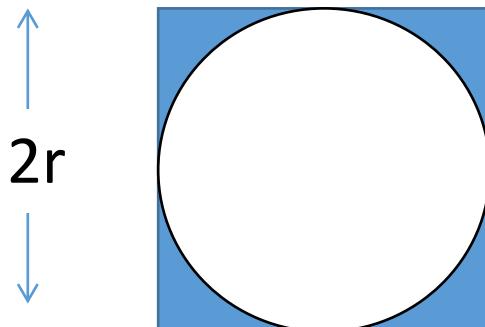


<https://erikbern.com/2015/10/20/nearest-neighbors-and-vector-models-epilogue-curse-of-dimensionality.html>

The Curse of Dimensionality

Example: What fraction of the points in a cube lie outside the sphere inscribed in it?

In two dimensions

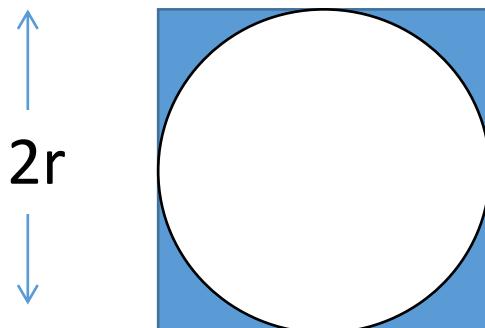


What fraction of the square (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

The Curse of Dimensionality

Example: What fraction of the points in a cube lie outside the sphere inscribed in it?

In two dimensions



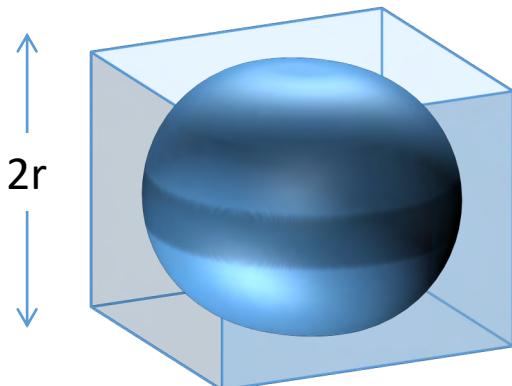
What fraction of the square (i.e the cube) is outside the inscribed circle (i.e the sphere) in two dimensions?

$$1 - \frac{\pi r^2}{4r^2} = 1 - \frac{\pi}{4} \approx 0.2$$

The Curse of Dimensionality

Example: What fraction of the points in a cube lie outside the sphere inscribed in it?

In three dimensions



What fraction of the square
(i.e the cube) is outside the
inscribed circle (i.e the sphere)
in two dimensions?

$$1 - \frac{\frac{4}{3}\pi r^3}{8r^3} = 1 - \frac{\pi}{6} \approx 0.47$$

General form:

https://en.wikipedia.org/wiki/Volume_of_an_n-ball Lec 4: KNN & Decision Tree

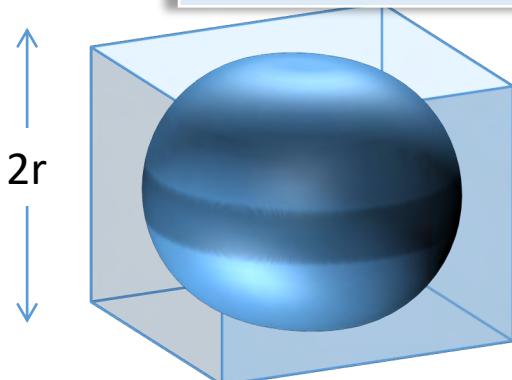
The Curse of Dimensionality

Example: What fraction of the points in a cube lie outside the sphere inscribed in it?

In three

As the dimensionality increases,
this fraction approaches 1!!

are

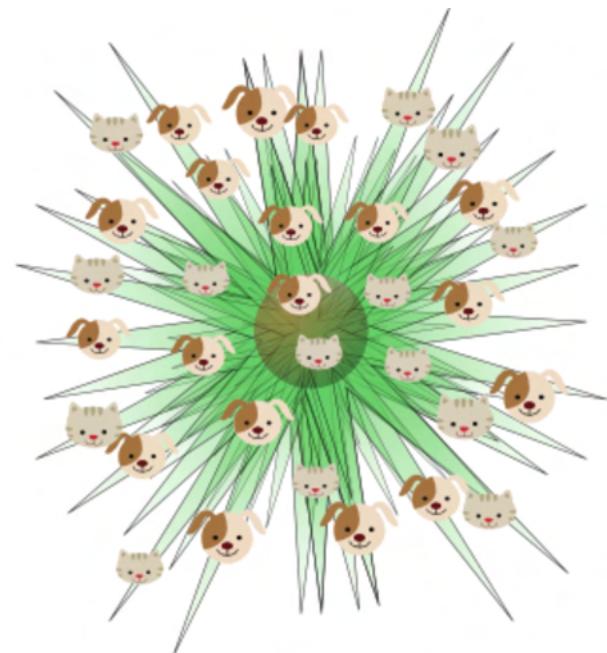
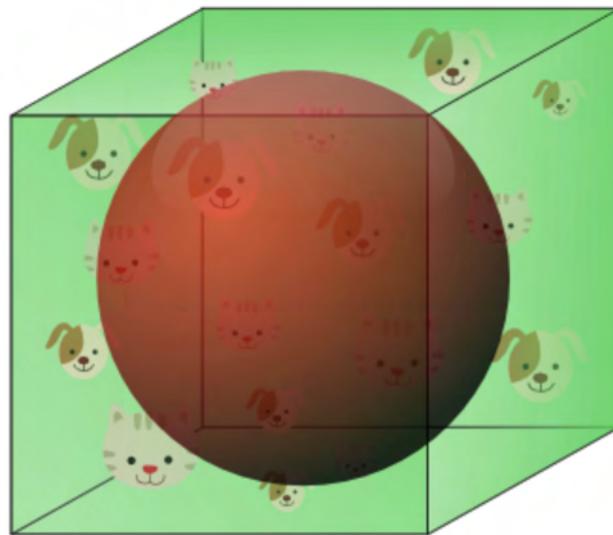
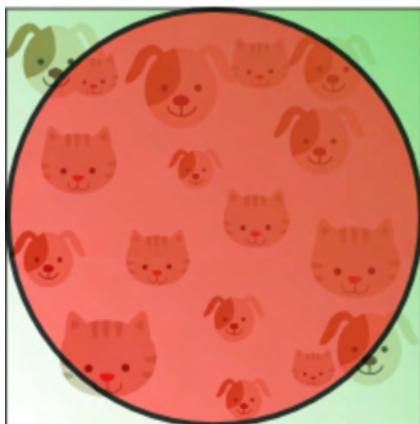


(i.e the cube) is outside the
inscribed circle (i.e the sphere)
in two dimensions?

$$1 - \frac{\frac{4}{3}\pi r^3}{8r^3} = 1 - \frac{\pi}{6}$$

Indication

- ❖ In high dimensions, most of the volume of the cube is far away from the center!



[http://www.woforo.com/blogs/HosseinAbedi/
curse-of-dimensionality](http://www.woforo.com/blogs/HosseinAbedi/curse-of-dimensionality)

The Curse of Dimensionality

- ❖ Most of the points in high dimensional spaces are far away from the origin!
- ❖ In 2 or 3 dimensions, most points are near the center
 - ❖ Need more data to “fill up the space”
- ❖ Bad news for k-NN in high dimensional spaces
 - ❖ Even if most/all features are relevant, in high dimensional spaces, most points are equally far from each other!
 - ❖ “Neighborhood” becomes very large

Dealing with the curse of dimensionality

- ❖ Most “*real-world*” data is not uniformly distributed in the high dimensional space
 - ❖ Capturing the *underlying dimensionality* of the space -- dimensionality reduction
- ❖ Feature selection (**e.g., by information gain**)
- ❖ Prior knowledge or preferences about the hypotheses can also help

Lecture 4: Decision Tree Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Key issues in machine learning

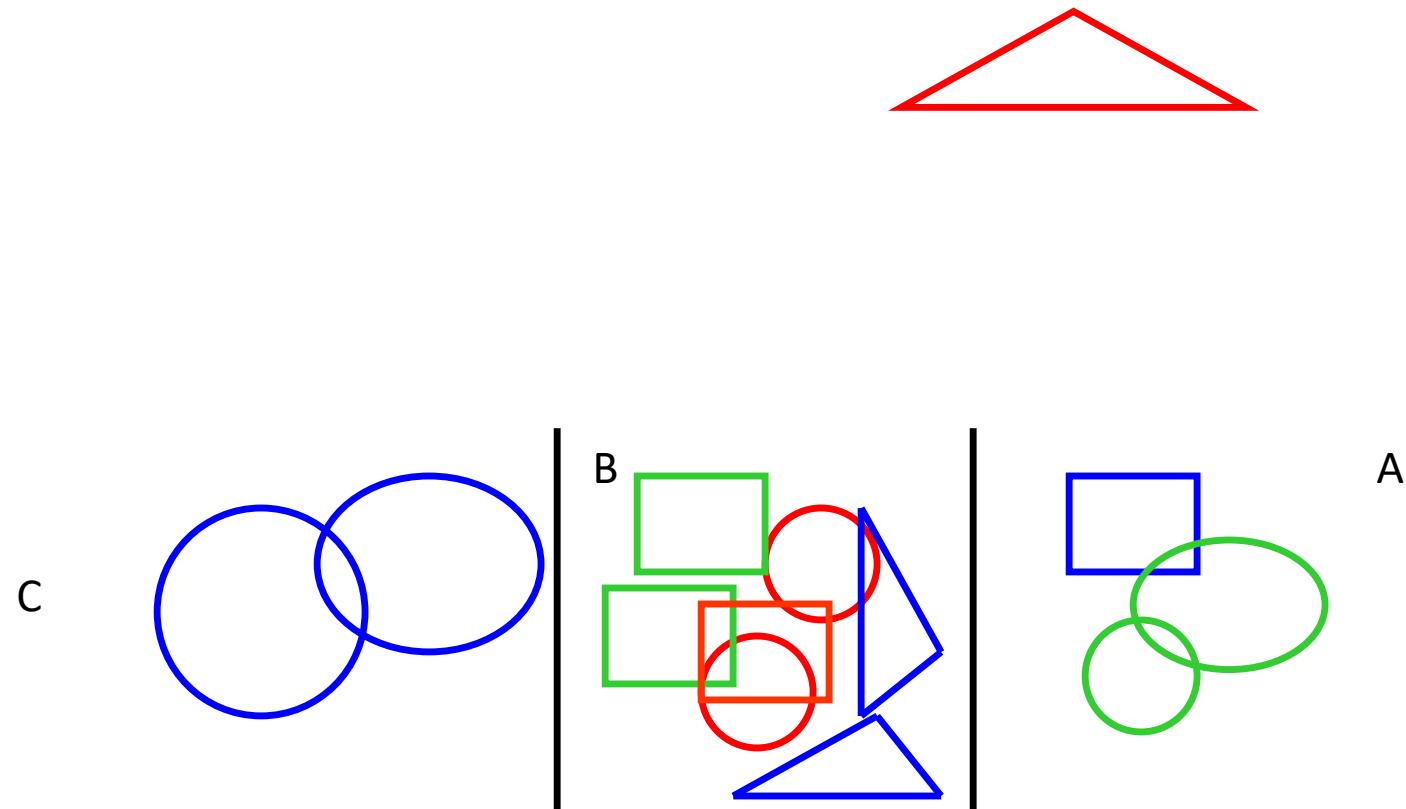
- ❖ Modeling
 - ❖ How to formulate your problem?
- ❖ Representation
 - ❖ What is the input/output space?
 - ❖ What is the hypothesis space?
- ❖ Algorithms
 - ❖ How to find the best hypothesis?

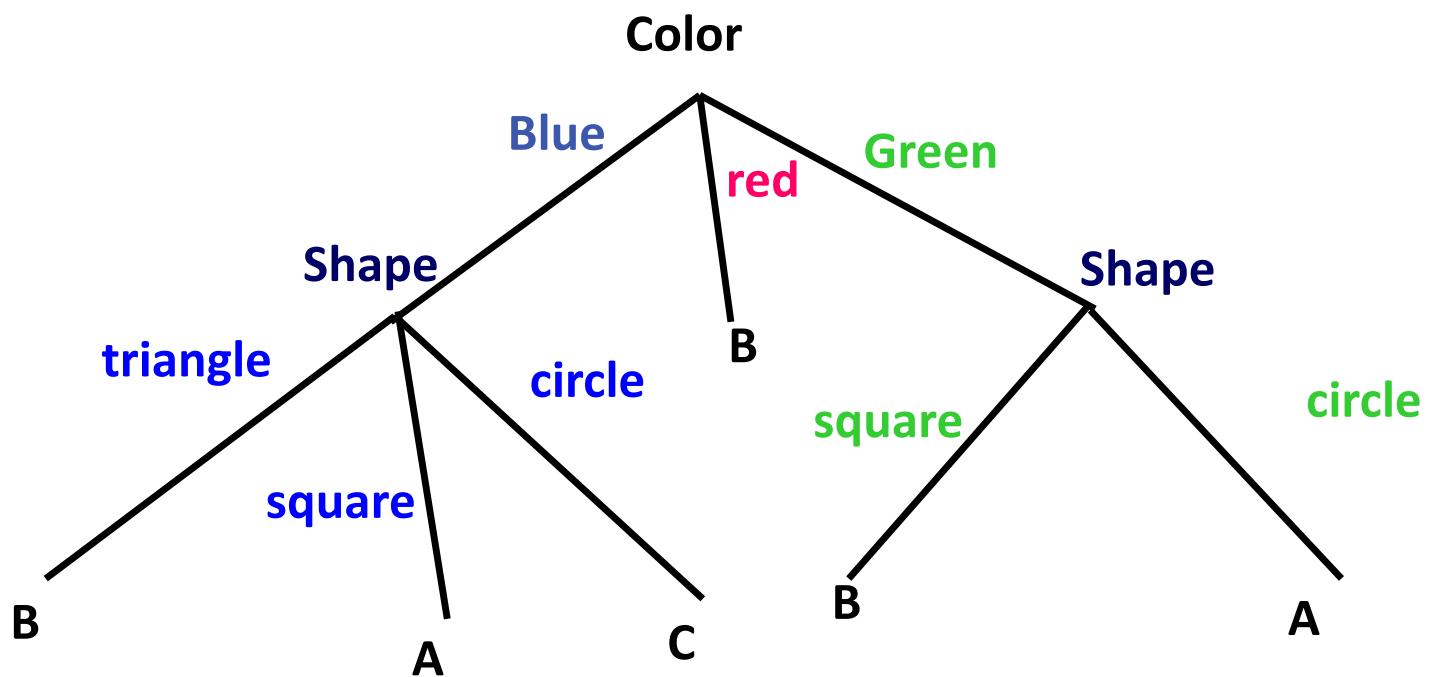
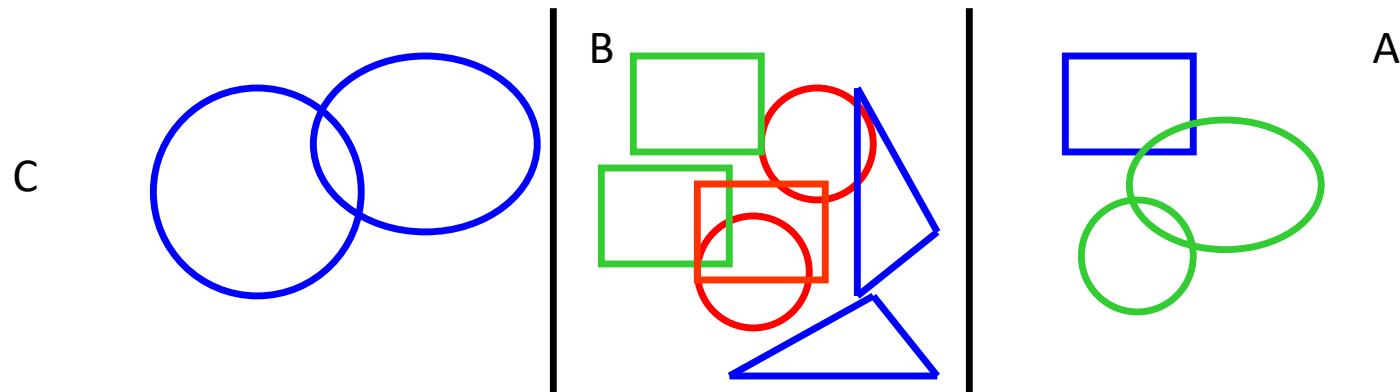
This Lecture

- ❖ Model/Representation: Decision trees
- ❖ Algorithm: Learning decision trees
 - (ID3 algorithm)
 - ❖ Information theory
 - ❖ Greedy heuristic (based on information gain)
Originally developed for discrete features

Sample dataset

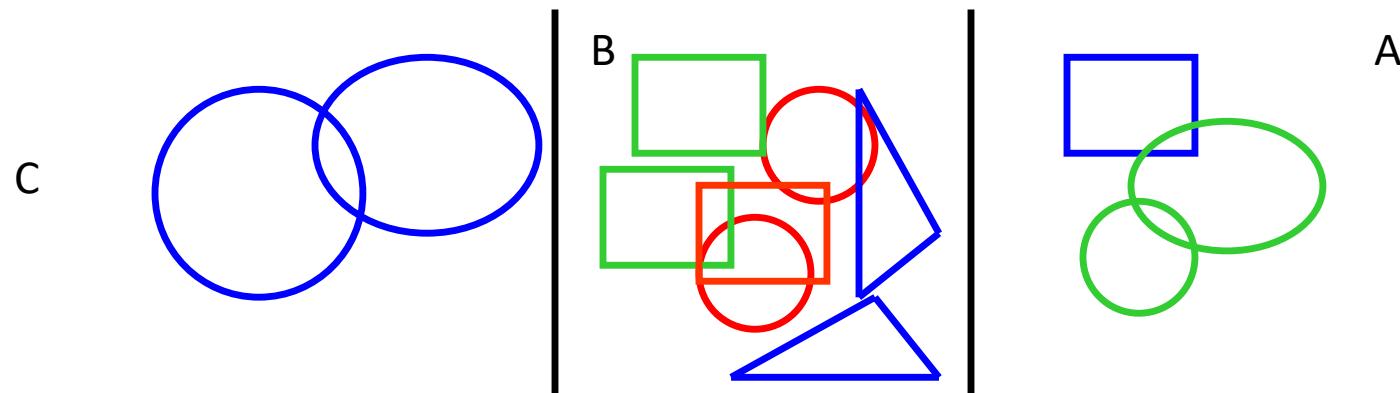
- ❖ What is the label for a red triangle?





Decision Trees

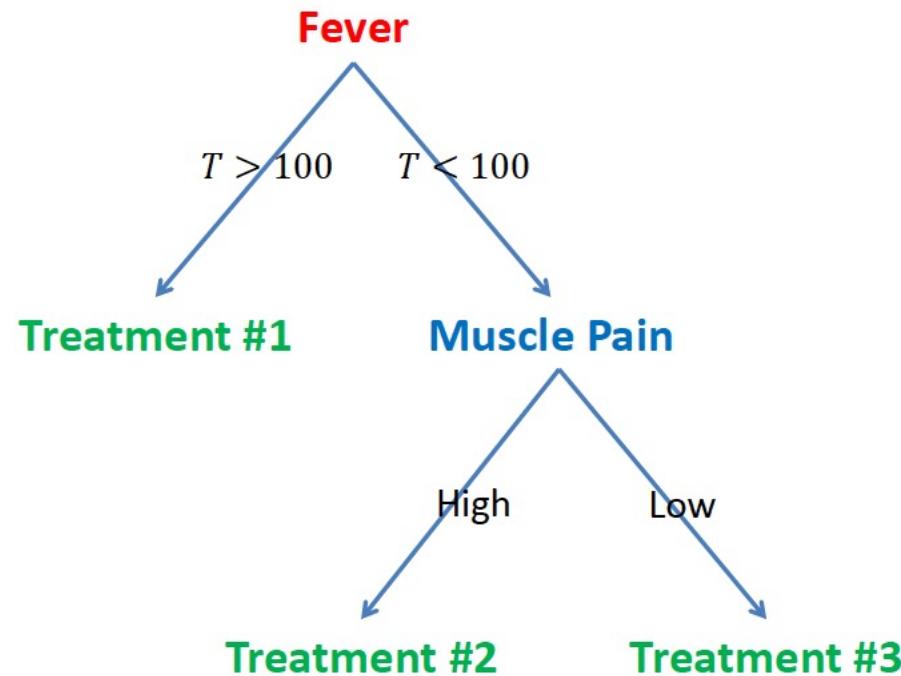
- ❖ A hierarchical data structure that represents data



Motivations:

Many decisions are tree structures

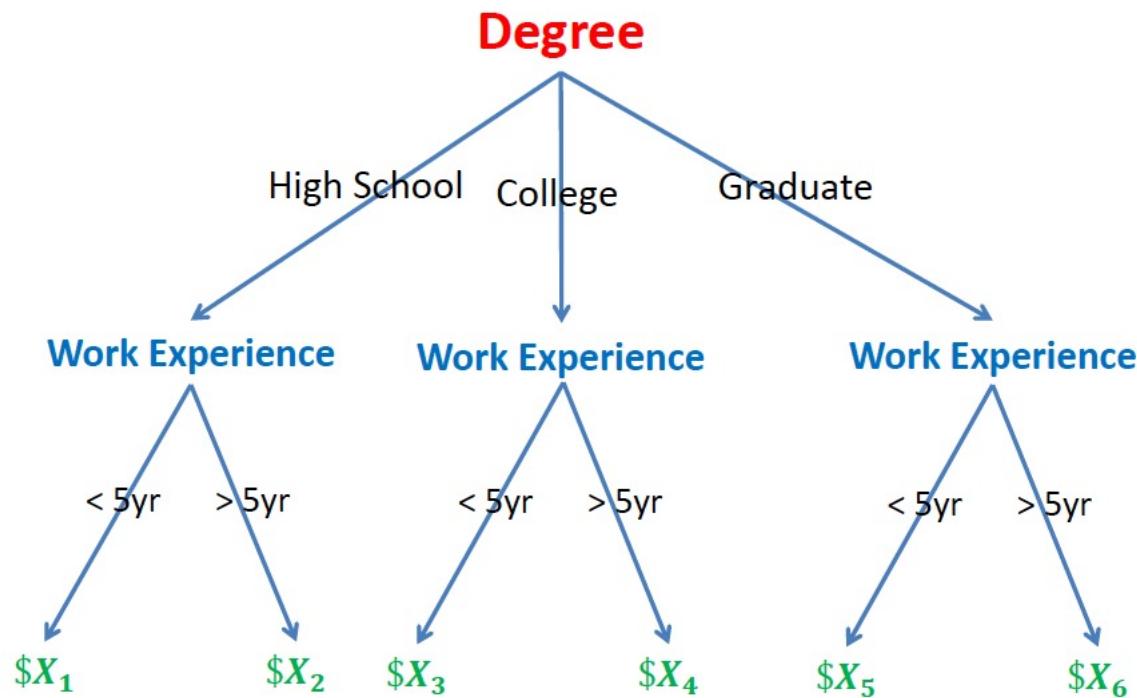
Medical treatment



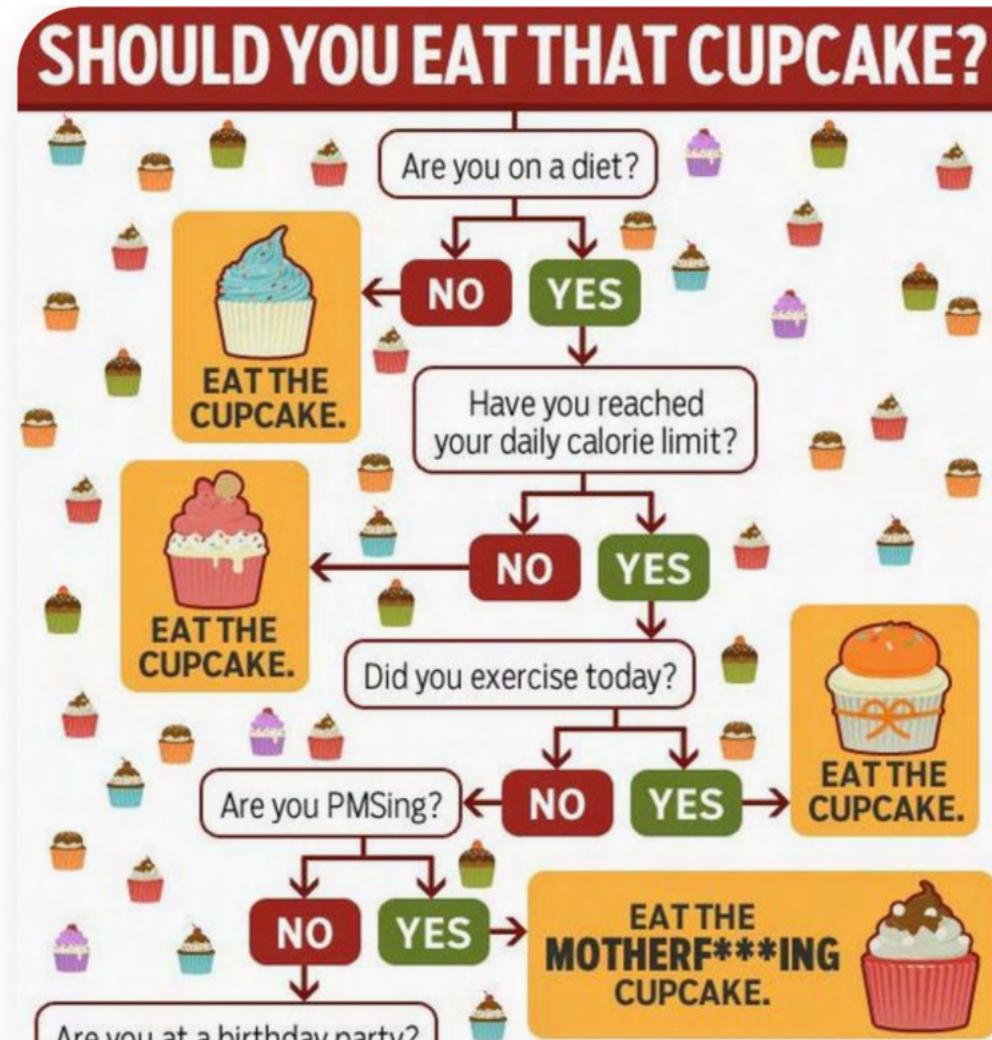
Motivations:

Many decisions are tree structures

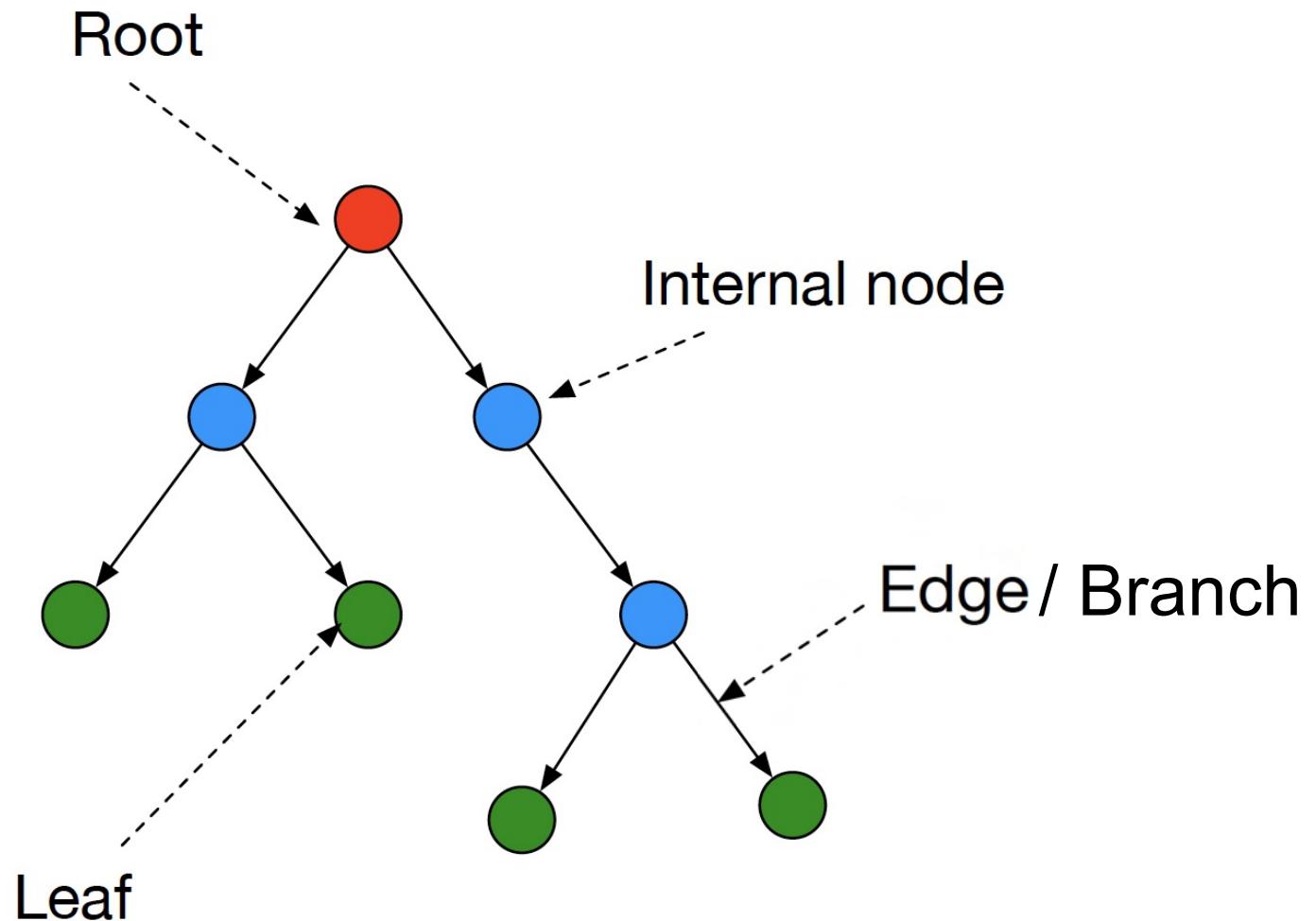
Salary in a company



Motivations: Many decisions are tree structures



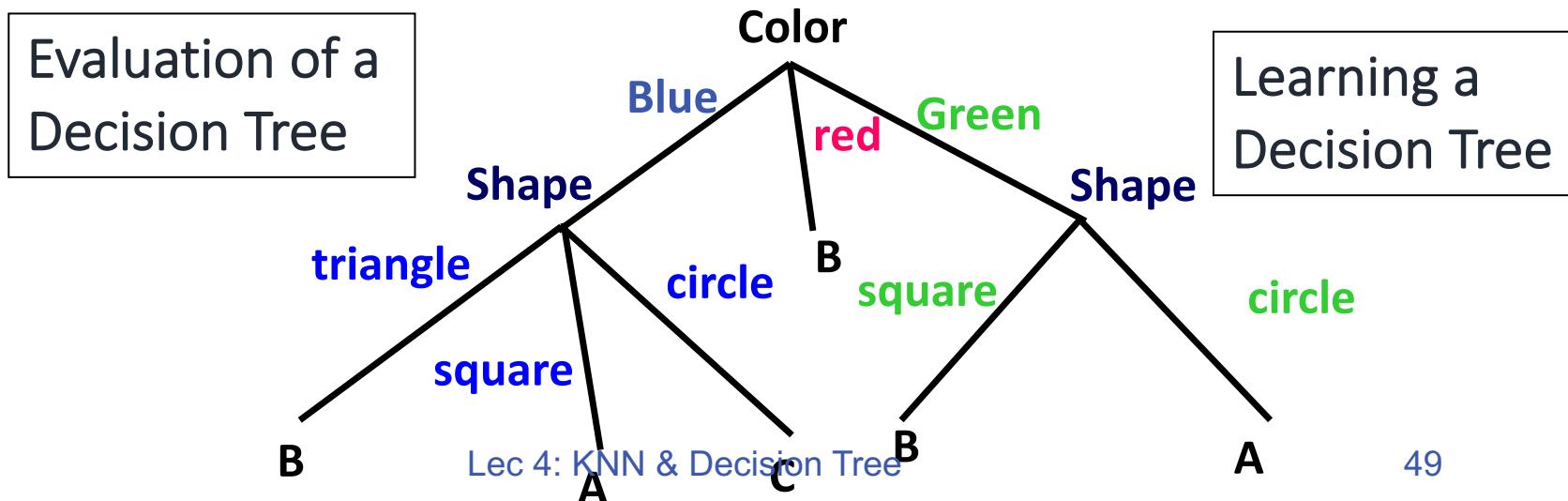
Terminology



Will sometimes drop the arrows on the edges

The Representation

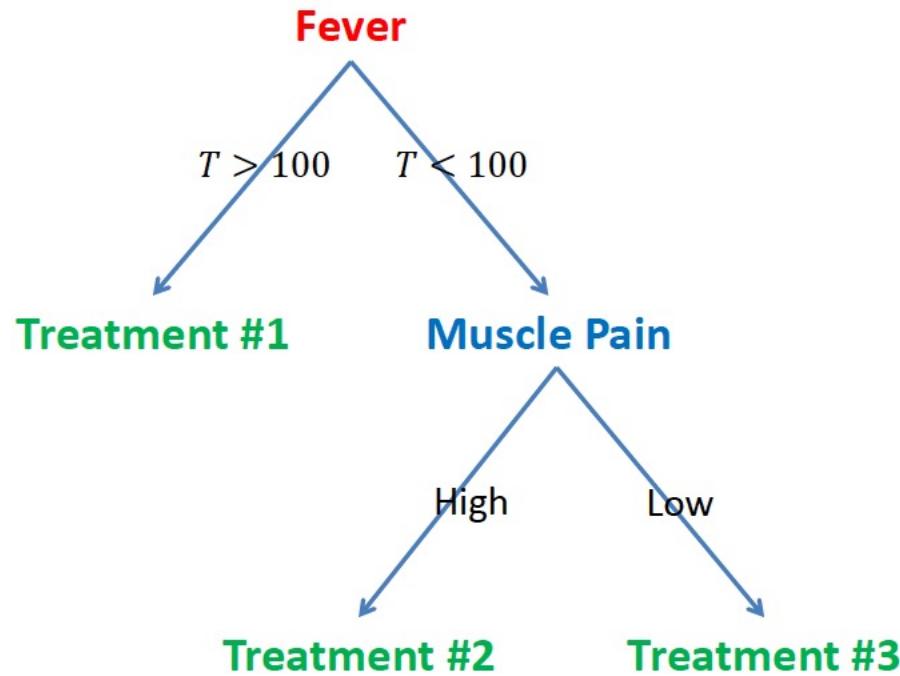
- ❖ Decision Trees are classifiers for instances represented as feature vectors (color= ; shape= ; label=)
- ❖ Nodes are **tests** for feature values
- ❖ Edges: There is one branch for each value of the feature
- ❖ Leaves specify the category (labels)
- ❖ Can categorize instances into multiple disjoint categories



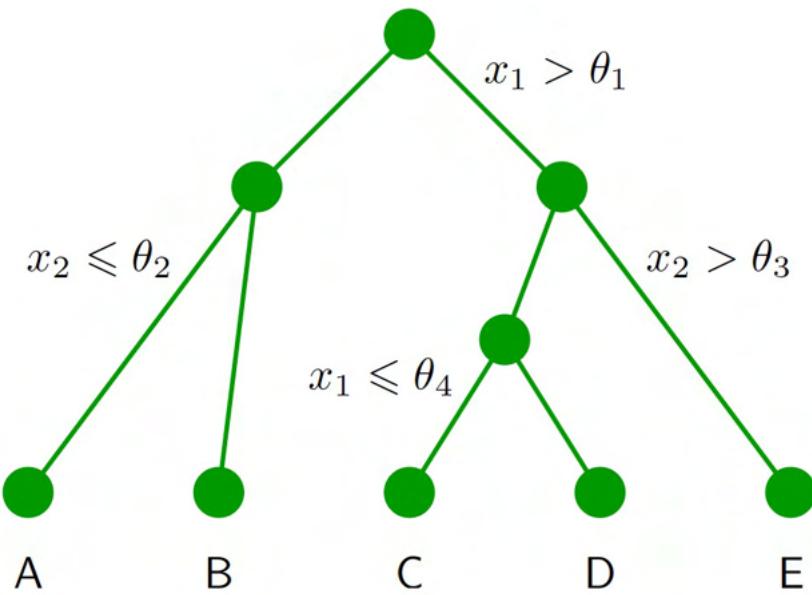
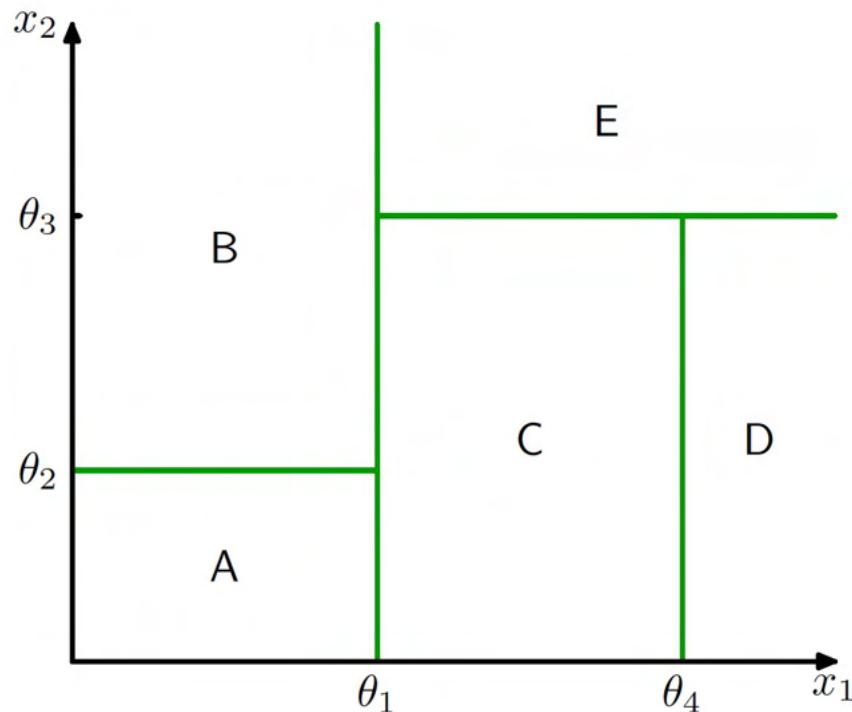
Handling real-valued features

- ❖ Usually, instances are represented as attribute-value pairs (color=blue, shape = square, +)
- ❖ Numerical values can be used either by discretizing or by using thresholds for splitting nodes

Example

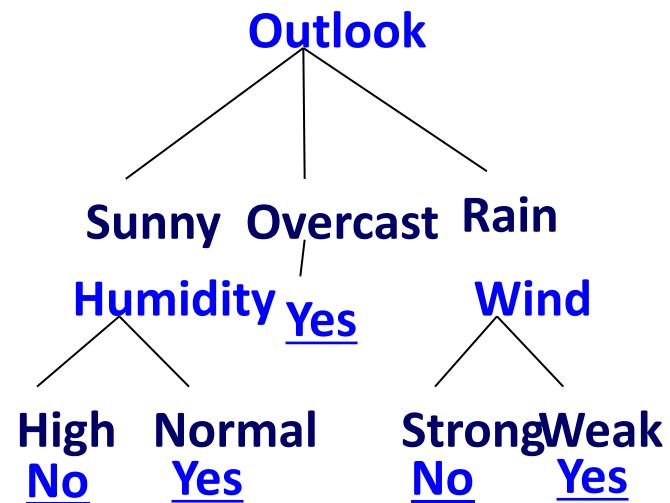


A tree partitions the feature space



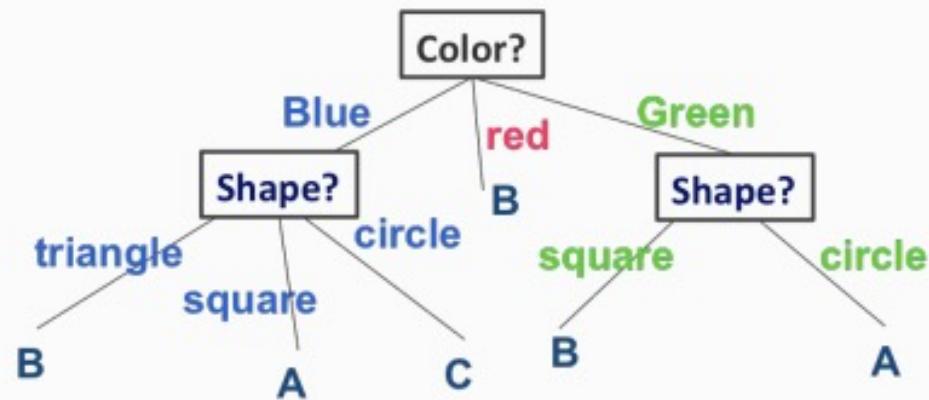
Advantages of Decision tree

- ❖ It is expressive -- can represent any Boolean Function
- ❖ It is interpretable



Expressivity of Decision Trees

- ❖ What Boolean functions can decision trees represent?
 - any Boolean function



(Color=blue AND Shape=triangle \Rightarrow Label=B) AND
(Color=blue AND Shape=square \Rightarrow Label=A) AND
(Color=blue AND Shape=circle \Rightarrow Label=C) AND....

Learning a decision tree

This Lecture

- ❖ Model/Representation: Decision trees
- ❖ Algorithm: Learning decision trees (ID3 algorithm)
 - ❖ Greedy heuristic (based on information gain)
Originally developed for discrete features

Will I play tennis today?

❖ Features

- ❖ Outlook: {Sun, Overcast, Rain}
- ❖ Temperature: {Hot, Mild, Cool}
- ❖ Humidity: {High, Normal, Low}
- ❖ Wind: {Strong, Weak}

❖ Labels

- ❖ Binary classification task: $Y = \{+, -\}$

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

Temperature: H(ot),
M(edium),
C(ool)

Humidity: H(igh),
N(ormal),
L(ow)

Wind: S(trong),
W(eak)

Discussion: Draw a DT

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

Temperature: H(ot),
M(edium),
C(ool)

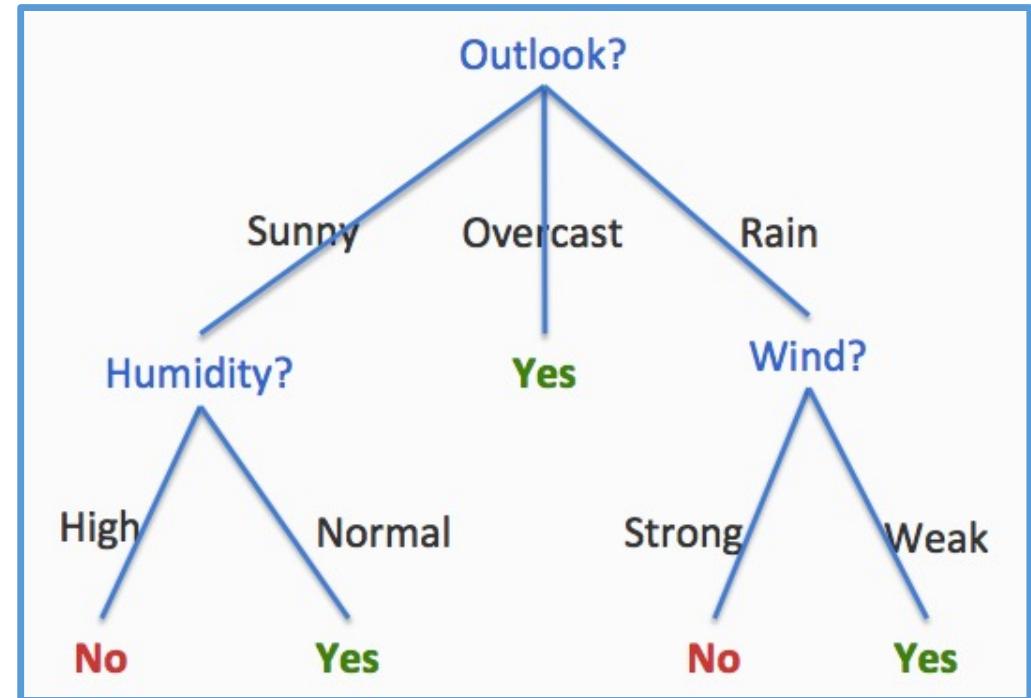
Humidity: H(igh),
N(ormal),
L(ow)

Wind: S(trong),
W(eak)

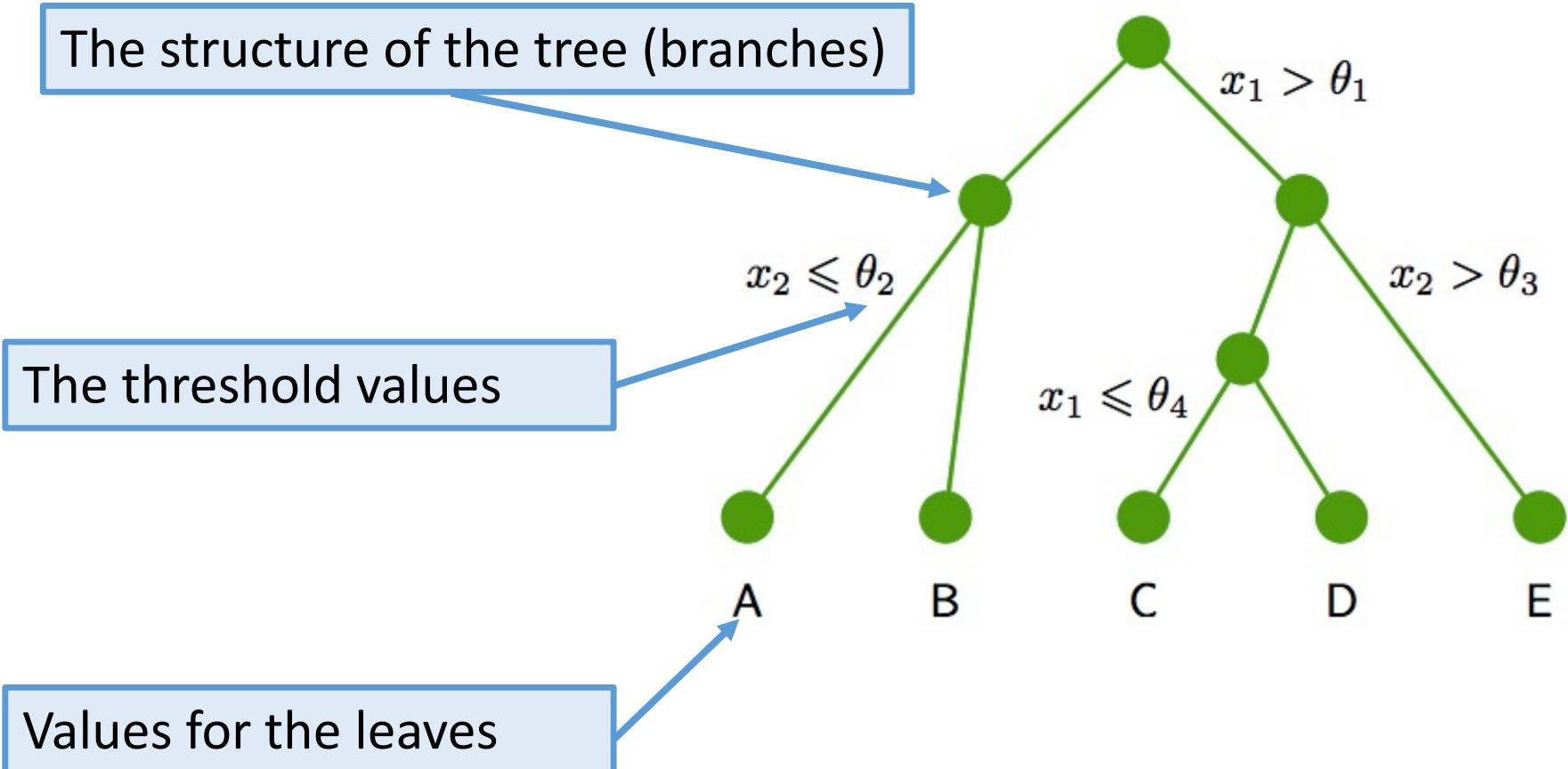
Basic Decision Trees Learning Algorithm

- ❖ Data is processed in Batch
(i.e. all the data available)
- ❖ Recursively build a decision tree top down.

O	T	H	W	Play?	
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-



What do we need to learn?



Decision Tree

Fall 2021

Kai-Wei Chang
CS @ UCLA

kw+cm146@kwchang.net

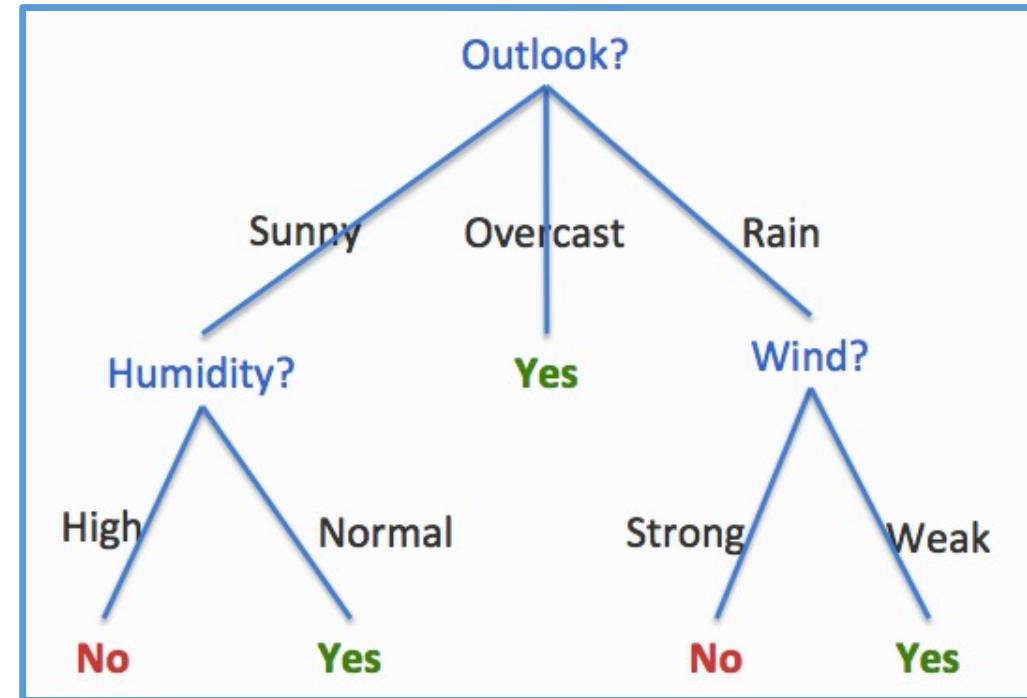
The instructor gratefully acknowledges Dan Roth, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Learning a decision tree

Basic Decision Trees Learning Algorithm

- ❖ Data is processed in Batch
(i.e. all the data available)
- ❖ Recursively build a decision tree top down.

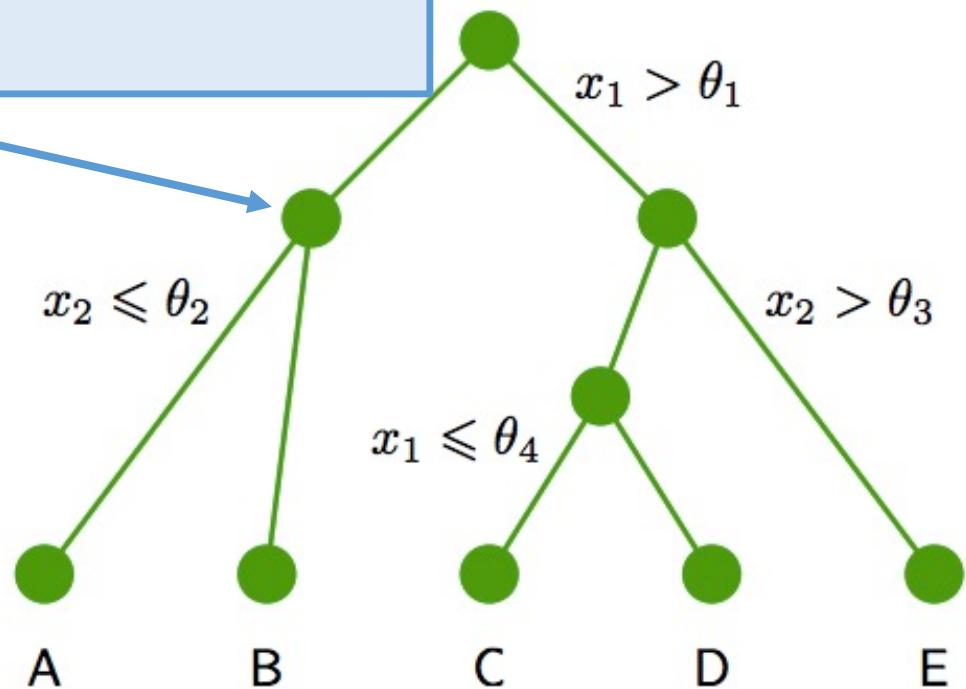
O	T	H	W	Play?	
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-



What do we need to learn?

Pick one feature that **best** classifier the data

How?



DT algorithm: ID3(S , Attributes, Label)

- ❖ A recursive algorithm
- ❖ Recursively build a decision tree top down.
- ❖ Base case:
 - If all examples are labeled the same
Return a single node tree with Label
 - Otherwise
 - Recursive decision tree algorithm
(see next slide)

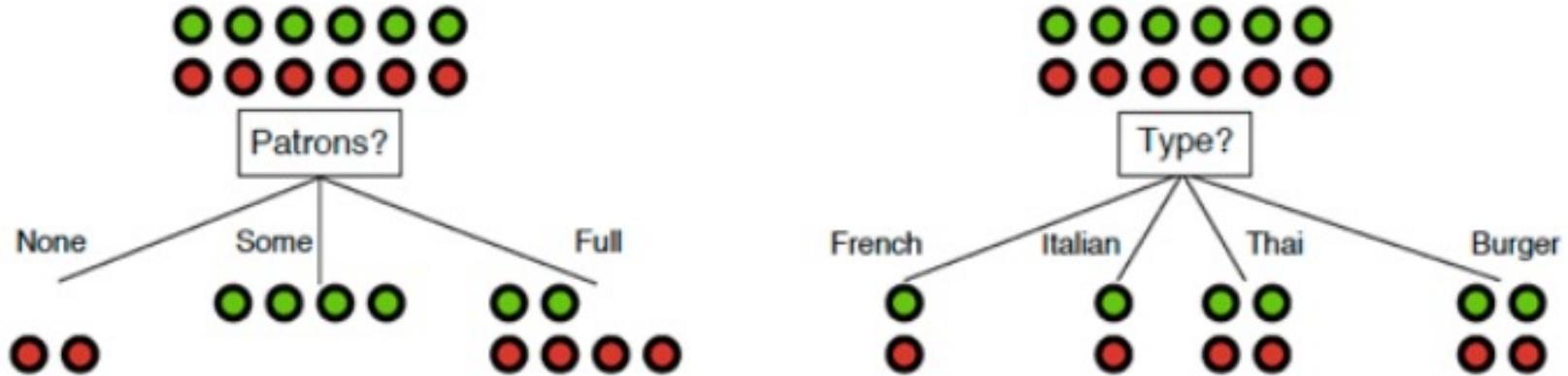
DT algorithm: ID3(S , Attributes, Label)

1. If all examples have a same label
return a single node tree with LabelBase case
2. $A = \text{attribute in Attributes that } \underline{\text{best}} \text{ classifies } S$
3. For each possible value v of A
 1. Add a new tree branch corresponding to $A=v$
 2. Let S_v be the subset of examples in S with $A=v$
 3. if S_v is empty:
add leaf node with the common value of Label in S
else: below this branch add the subtree
 $\text{ID3}(S_v, \text{Attributes} - \{A\}, \text{Label})$

Which attribute to split?

- ❖ The goal is to have the resulting decision tree as small as possible
- ❖ Finding the minimal decision tree consistent with the data is NP-hard
- ❖ A greedy heuristic search for a simple tree (cannot guarantee optimality)

Which attribute to split?



Patrons? is a better choice—gives **information** about the classification

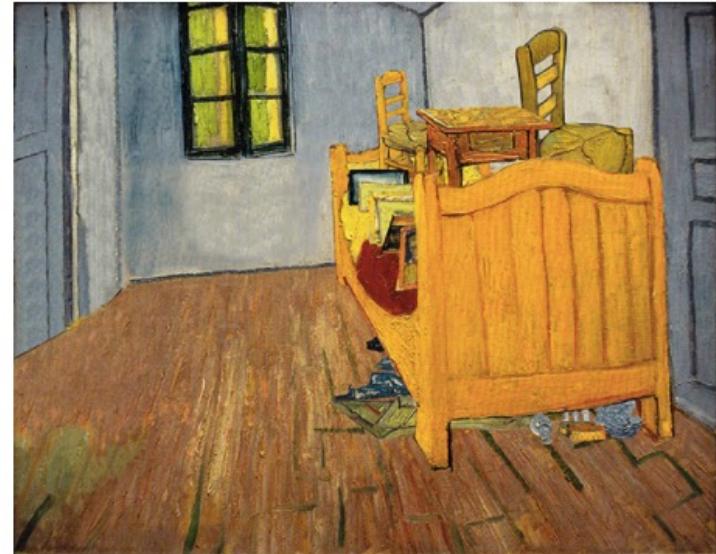
Need a way to quantify it

Which attribute to split?

- ❖ We want attributes that split the examples to sets that are **relatively pure in one label**; this way we are closer to a leaf node.
- ❖ The most popular heuristics is based on **information gain**

How to measure information gain?

- ❖ Idea: Gaining information reduces uncertainty
- ❖ Uncertainty can be measured by Entropy



Vincent Van Gogh: Bedroom in Arles

High entropy

By Ursus Wehrli

Low entropy

How to measure information gain?

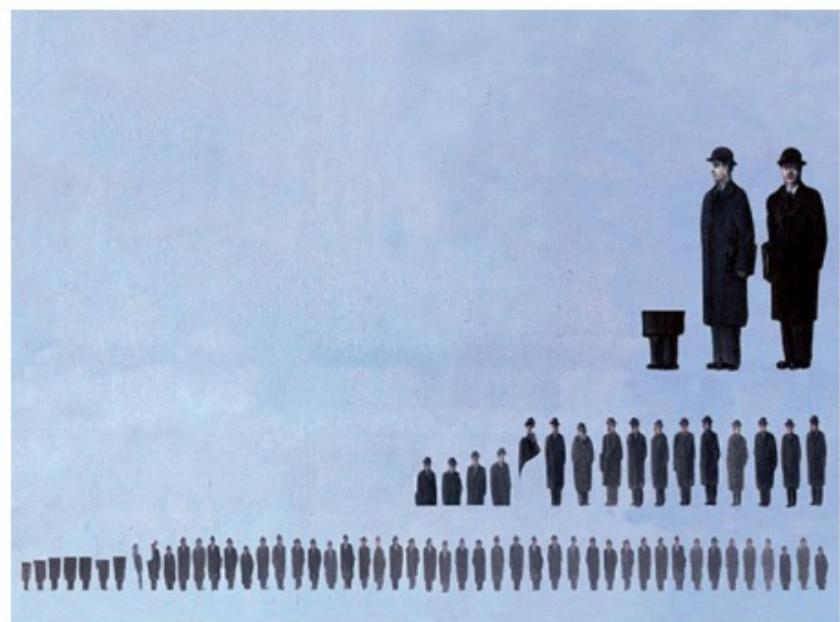
- ❖ Idea: Gaining information reduces uncertainty
- ❖ Uncertainty can be measured by Entropy

René Magritte "Golconda"



High entropy

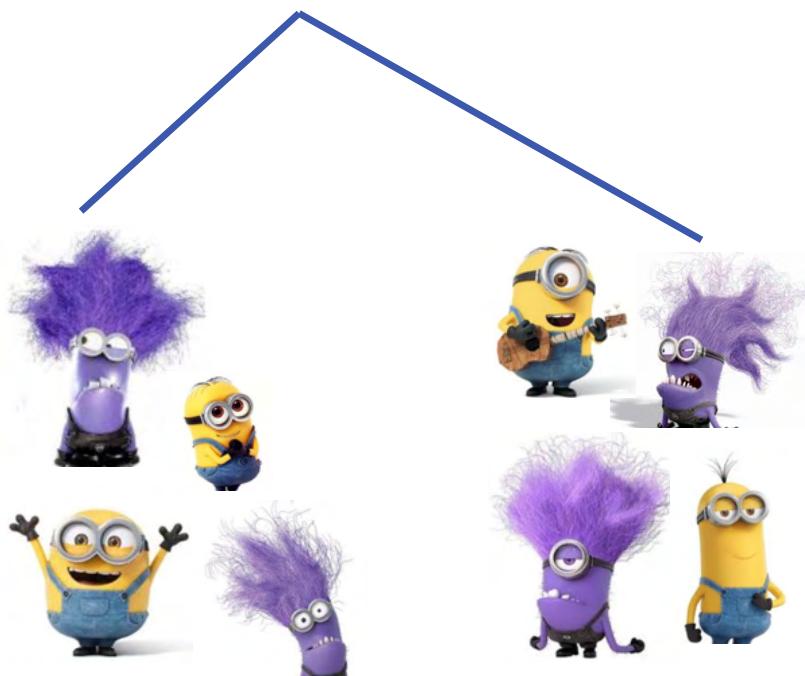
Bv Ursus Wehrli



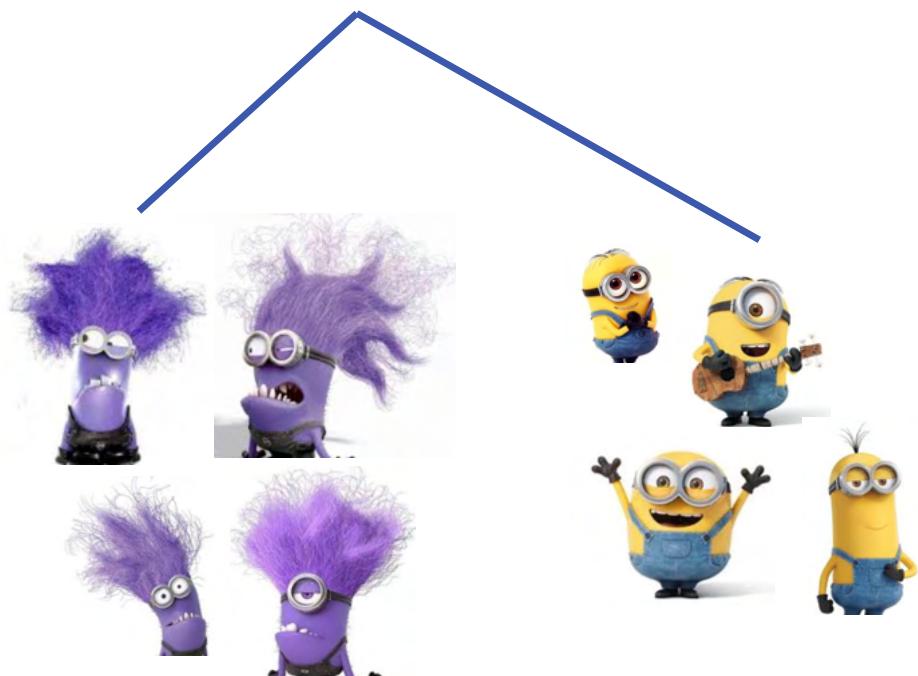
Low entropy

How to measure information gain?

- ❖ Idea: Gaining information reduces uncertainty
- ❖ Uncertainty can be measured by Entropy



High entropy



Low entropy

Entropy

- ❖ Entropy (impurity, disorder) of a set of examples, S , relative to a binary classification is:

$$H[S] = -P_+ \log_2(P_+) - P_- \log_2(P_-)$$

- ❖ where P_+ is the proportion of positive examples in S and P_- is the proportion of negatives.

Here we define $0 \log 0 = 0$

Entropy (formal definition)

- ❖ If a random variable S has K different values, a_1, a_2, \dots, a_K , its entropy is given by

$$H[S] = - \sum_{v=1}^K P(S = a_v) \log_2 P(S = a_v)$$

- ❖ Measures the amount of uncertainty of a random variable with a specific probability distribution. Higher it is, less confident we are in its outcome

Entropy (intuition)

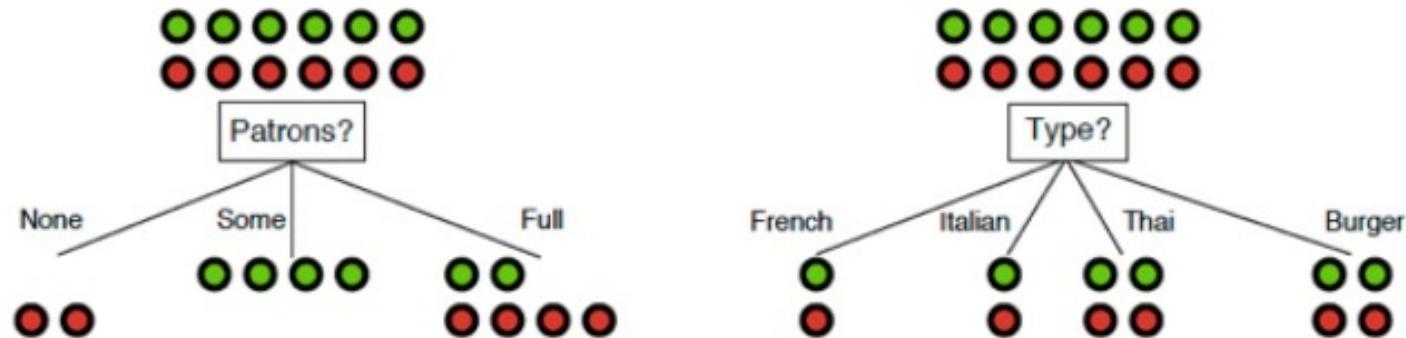
- ❖ In average, how many bits do we need to send the message (#bits/#length of message)



Entropy (intuition)

- ❖ In average, how many bits do we need to send the message (#bits/#length of message)
- ❖ Consider you have four possible tokens (a,b,c,d). What is the best way to encode them?
- ❖ All examples belong to the same category
 - e.g., aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 - no need to communicate (or just 1 bit)
- ❖ If all the examples are equally mixed (0.25, 0.25, 0.25, 0.25):
 - e.g., abbacaccddd.....
 - two bits for each token: (a:00, b:01, c:10, d:11)
- ❖ If $\frac{1}{4}$ of message is a, and $\frac{1}{2}$ is b and $\frac{1}{4}$ is c in average:
 - e.g., abbbbacc.....
 - (a:00, b:1, c:01, d:--)

Which attribute to split



Patrons? is a better choice—gives **information** about the classification

Patron vs. Type?

By choosing Patron, we end up with a partition (3 branches) with smaller entropy, ie, smaller uncertainty (0.45 bit)

By choosing Type, we end up with uncertainty of 1 bit.

Thus, we choose Patron over Type.

Uncertainty if we go with “Patron”

For “None” branch

$$-\left(\frac{0}{0+2} \log_2 \frac{0}{0+2} + \frac{2}{0+2} \log_2 \frac{2}{0+2}\right) = 0$$

For “Some” branch

$$-\left(\frac{4}{4+0} \log_2 \frac{4}{4+0} + \frac{0}{4+0} \log_2 \frac{0}{4+0}\right) = 0$$

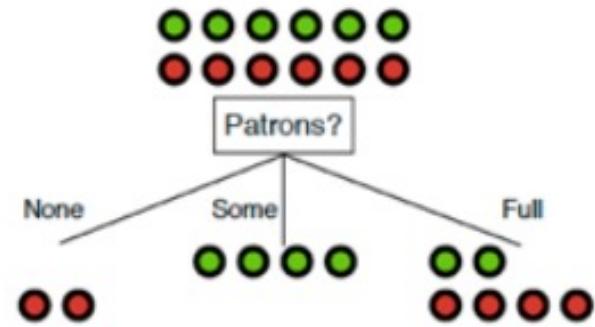
For “Full” branch

$$-\left(\frac{2}{2+4} \log_2 \frac{2}{2+4} + \frac{4}{2+4} \log_2 \frac{4}{2+4}\right) \approx 0.9$$

For choosing “Patrons”

weighted average of each branch: this quantity is called **conditional entropy**

$$\frac{2}{12} * 0 + \frac{4}{12} * 0 + \frac{6}{12} * 0.9 = 0.45$$



Information Gain

- ❖ The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- ❖ S_v is the subset of S for which attribute a has value v .
- ❖ The entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

Temperature: H(ot),
M(edium),
C(ool)

Humidity: H(igh),
N(ormal),
L(ow)

Wind: S(strong),
W(eak)

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Current entropy:

$$p = 9/14$$

$$n = 5/14$$

$$H(\text{Play?}) = -(9/14) \log_2(9/14) \\ -(5/14) \log_2(5/14)$$

$$\approx 0.94$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny: 5 of 14 examples

$$p = 2/5 \quad n = 3/5 \quad H_s = 0.971$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny: 5 of 14 examples

$$p = 2/5 \quad n = 3/5 \quad H_s = 0.971$$

Outlook = overcast: 4 of 14 examples

$$p = 4/4 \quad n = 0 \quad H_o = 0$$

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny: 5 of 14 examples

$$p = 2/5 \quad n = 3/5 \quad H_s = 0.971$$

Outlook = overcast: 4 of 14 examples

$$p = 4/4 \quad n = 0 \quad H_o = 0$$

Outlook = rainy: 5 of 14 examples

$$p = 3/5 \quad n = 2/5 \quad H_R = 0.971$$

Expected entropy:

$$(5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971 \\ = 0.694$$

Information gain:

$$0.940 - 0.694 = 0.246$$

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	W	-
2	S	H	S	-
3	O	H	W	+
4	R	M	W	+
5	R	C	W	+
6	R	C	S	-
7	O	C	S	+
8	S	M	W	-
9	S	C	W	+
10	R	M	W	+
11	S	M	S	+
12	O	M	S	+
13	O	H	W	+
14	R	M	S	-

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	W	-
2	S	H	S	-
3	O	H	W	+
4	R	M	W	+
5	R	C	W	+
6	R	C	S	-
7	O	C	S	+
8	S	M	W	-
9	S	C	W	+
10	R	M	W	+
11	S	M	S	+
12	O	M	S	+
13	O	H	W	+
14	R	M	S	-

Humidity = High:

$$p = 3/7 \quad n = 4/7$$

$$H_h = 0.985$$

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	W	-
2	S	H	S	-
3	O	H	W	+
4	R	M	W	+
5	R	C	W	+
6	R	C	S	-
7	O	C	S	+
8	S	M	W	-
9	S	C	W	+
10	R	M	W	+
11	S	M	S	+
12	O	M	S	+
13	O	H	W	+
14	R	M	S	-

Humidity = High:

$$p = 3/7 \quad n = 4/7 \quad H_h = 0.985$$

Humidity = Normal:

$$p = 6/7 \quad n = 1/7 \quad H_o = 0.592$$

Expected entropy:

$$(7/14) \times 0.985 + (7/14) \times 0.592 = 0.7885$$

Information Gain: Humidity

O	T	H	W	Play?
1	S	H	W	-
2	S	H	S	-
3	O	H	W	+
4	R	M	W	+
5	R	C	W	+
6	R	C	S	-
7	O	C	S	+
8	S	M	W	-
9	S	C	W	+
10	R	M	W	+
11	S	M	S	+
12	O	M	S	+
13	O	H	W	+
14	R	M	S	-

Humidity = High:

$$p = 3/7 \quad n = 4/7 \quad H_h = 0.985$$

Humidity = Normal:

$$p = 6/7 \quad n = 1/7 \quad H_o = 0.592$$

Expected entropy:

$$(7/14) \times 0.985 + (7/14) \times 0.592 = 0.7885$$

Information gain:

$$0.940 - 0.7885 = 0.1515$$

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246

Humidity: 0.151

Wind: 0.048

Temperature: 0.029

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246

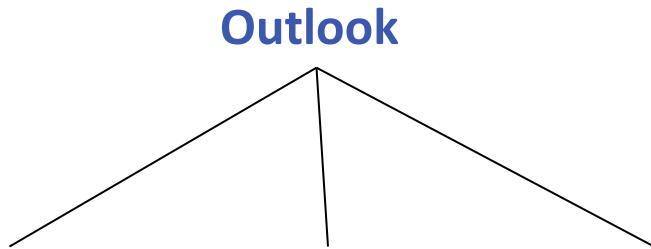
Humidity: 0.151

Wind: 0.048

Temperature: 0.029

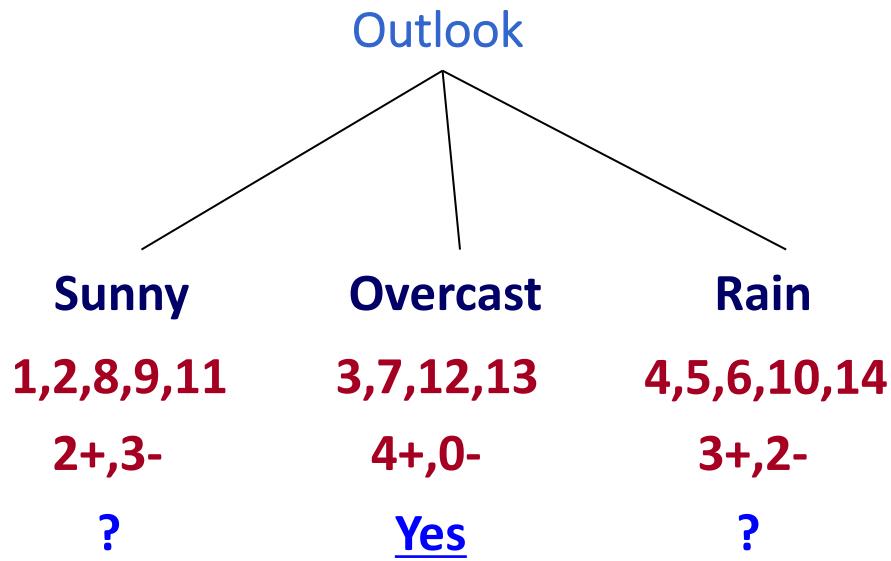
→ Split on Outlook

An Illustrative Example



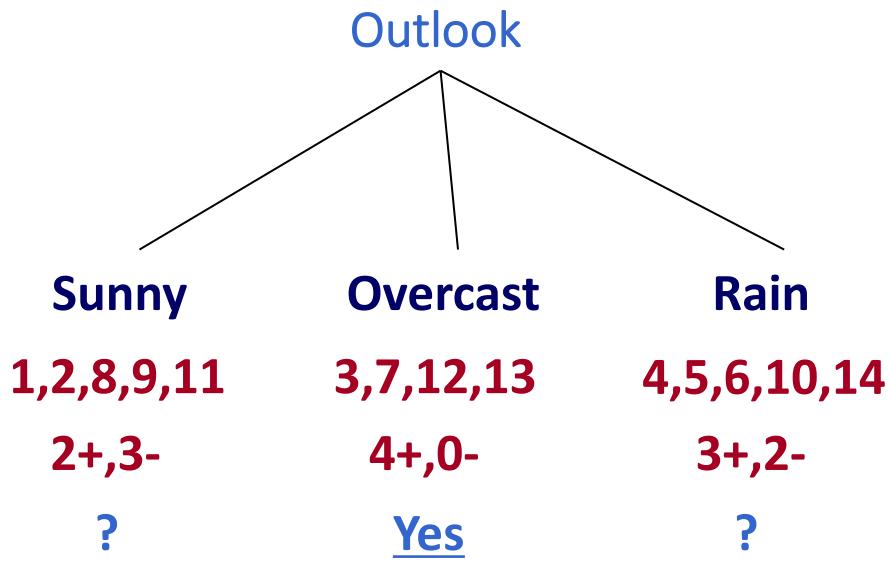
Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An Illustrative Example



	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example

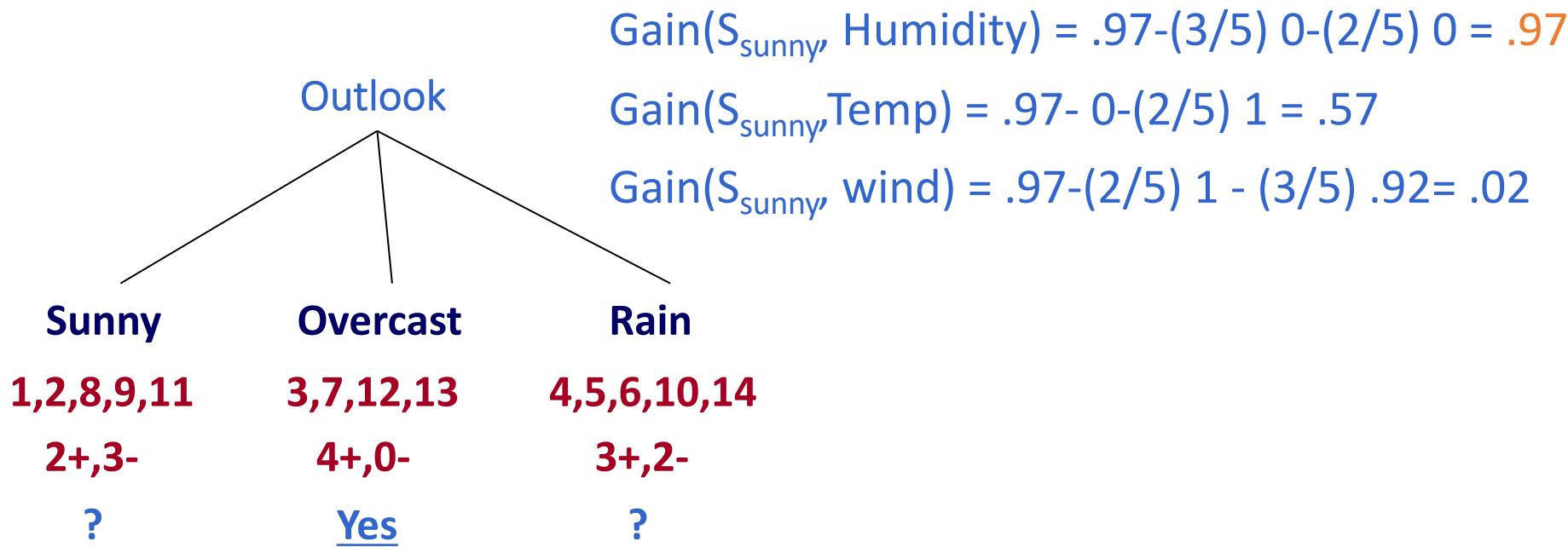


Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

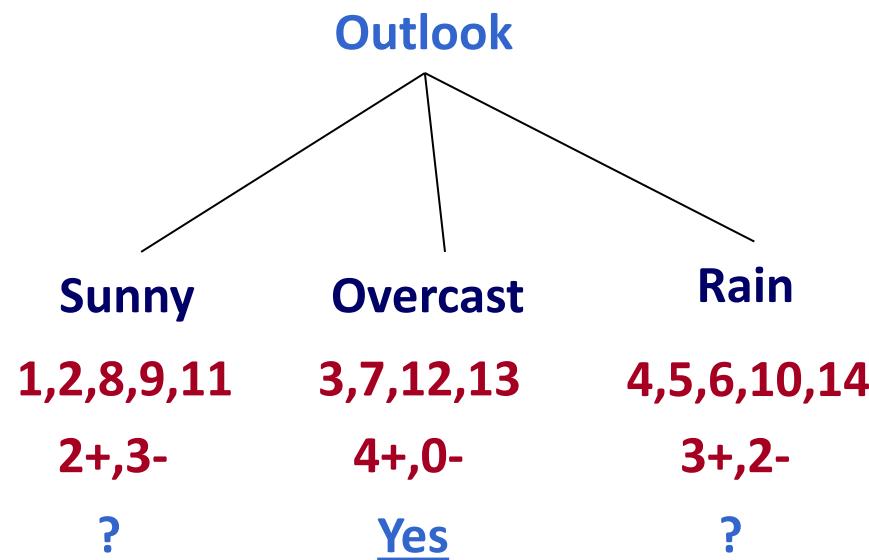
	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example

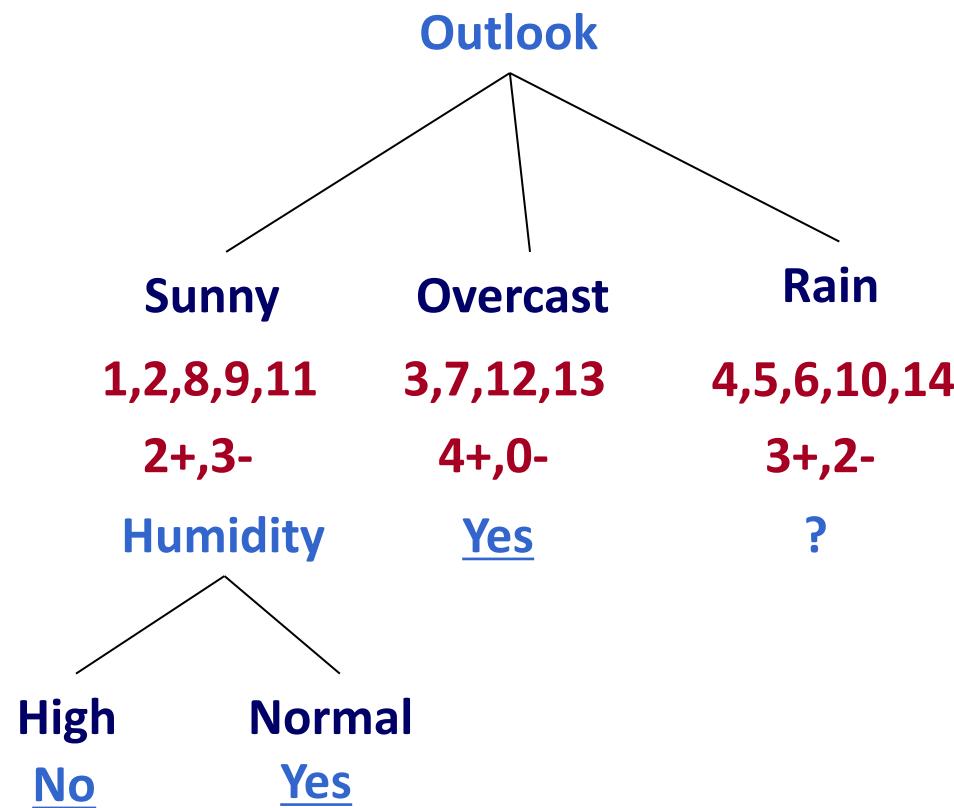


Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

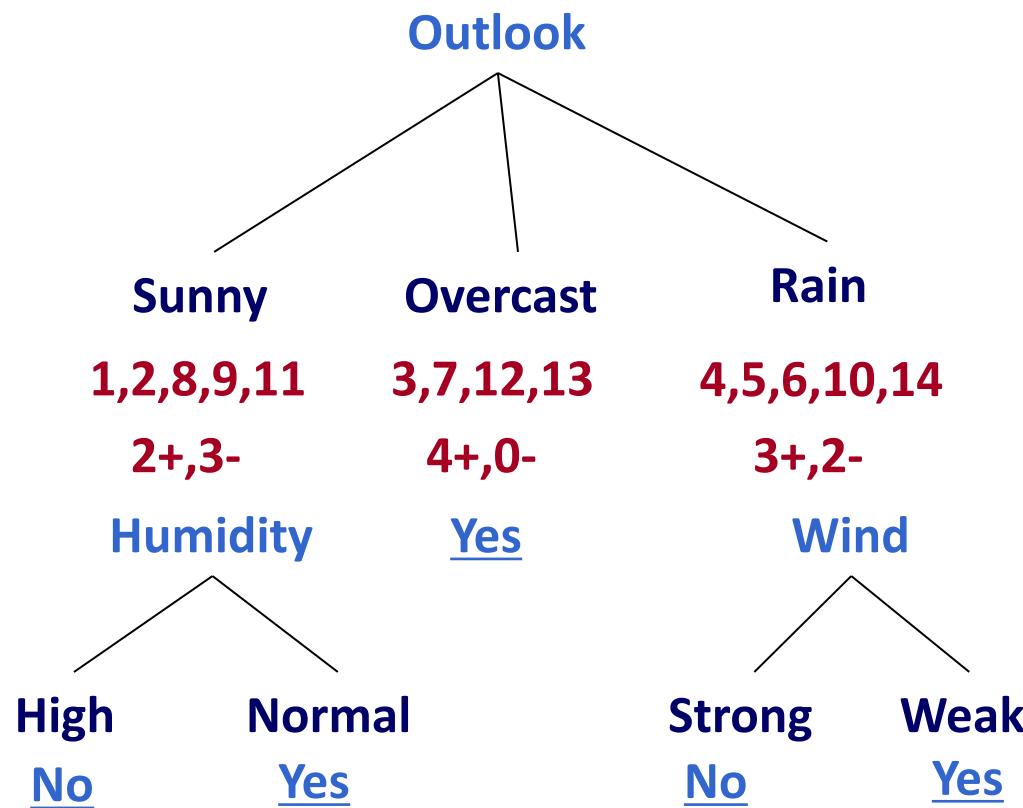
An Illustrative Example



An Illustrative Example



An Illustrative Example



Summary: Learning Decision Trees

1. **Representation**: What are decision trees?

- ❖ A hierarchical data structure that represents data

2. **Algorithm**: Learning decision trees

The ID3 algorithm: A greedy heuristic

- ❖ If all the examples have the same label, create a leaf with that label
- ❖ Otherwise, find the “most informative” attribute and split the data for different values of that attribute
- ❖ Recurse on the splits

Lecture 5: Perceptron Fall 2019

Kai-Wei Chang

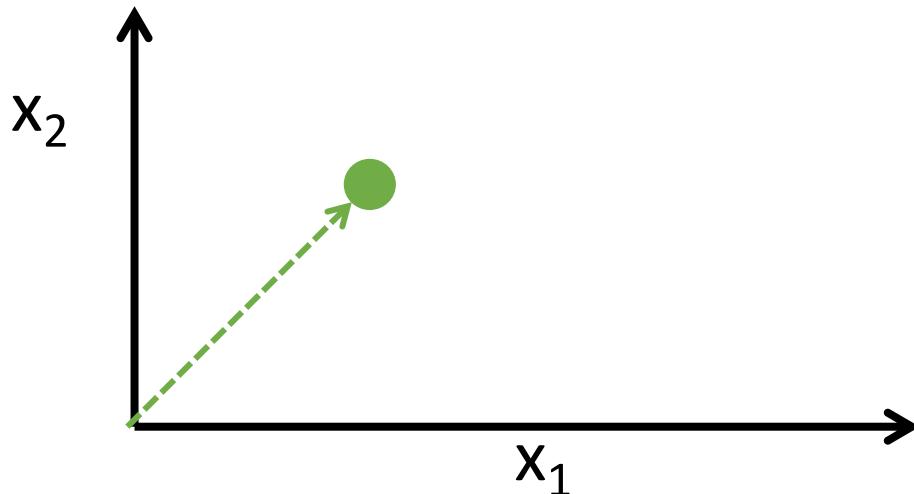
CS @ UCLA

kw+cm146@kwchang.net

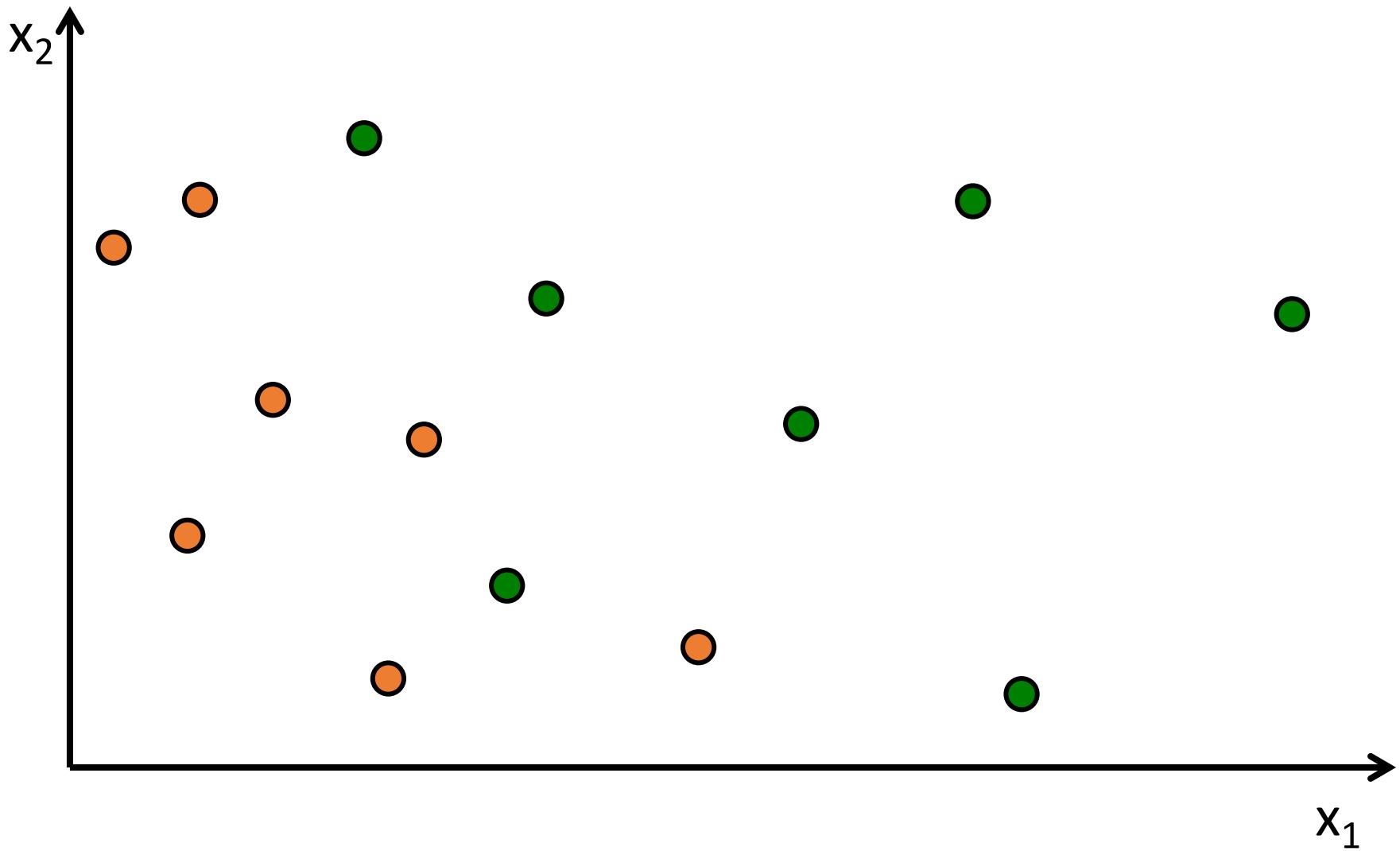
The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Recap: \mathcal{X} as a vector space

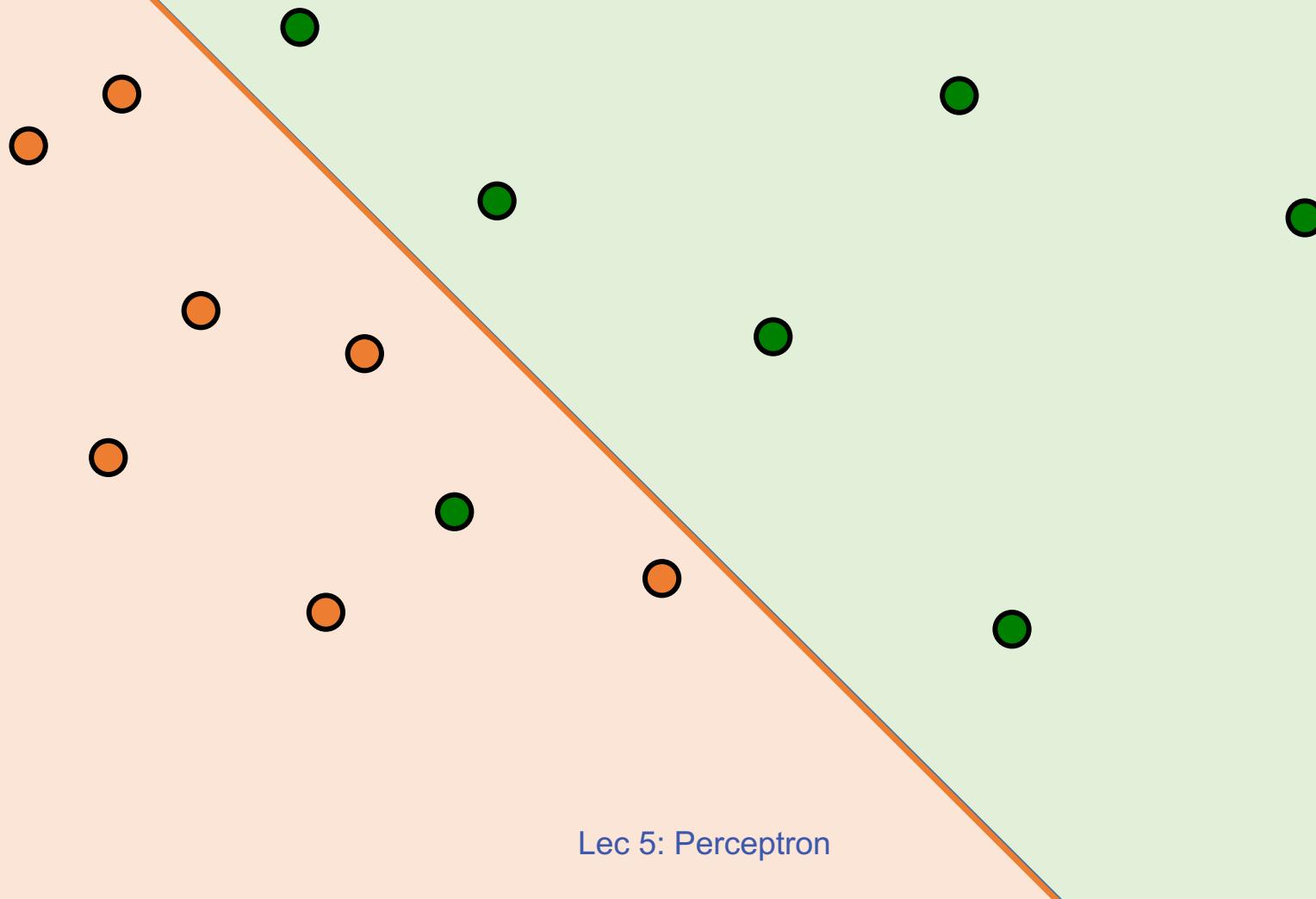
- ❖ \mathcal{X} is an N-dimensional vector space (e.g. \mathbb{R}^N)
 - ❖ Each dimension = one feature.
- ❖ Each \mathbf{x} is a **feature vector** (hence the boldface \mathbf{x}).
- ❖ Think of $\mathbf{x} = [x_1 \dots x_N]$ as a point in \mathcal{X} :



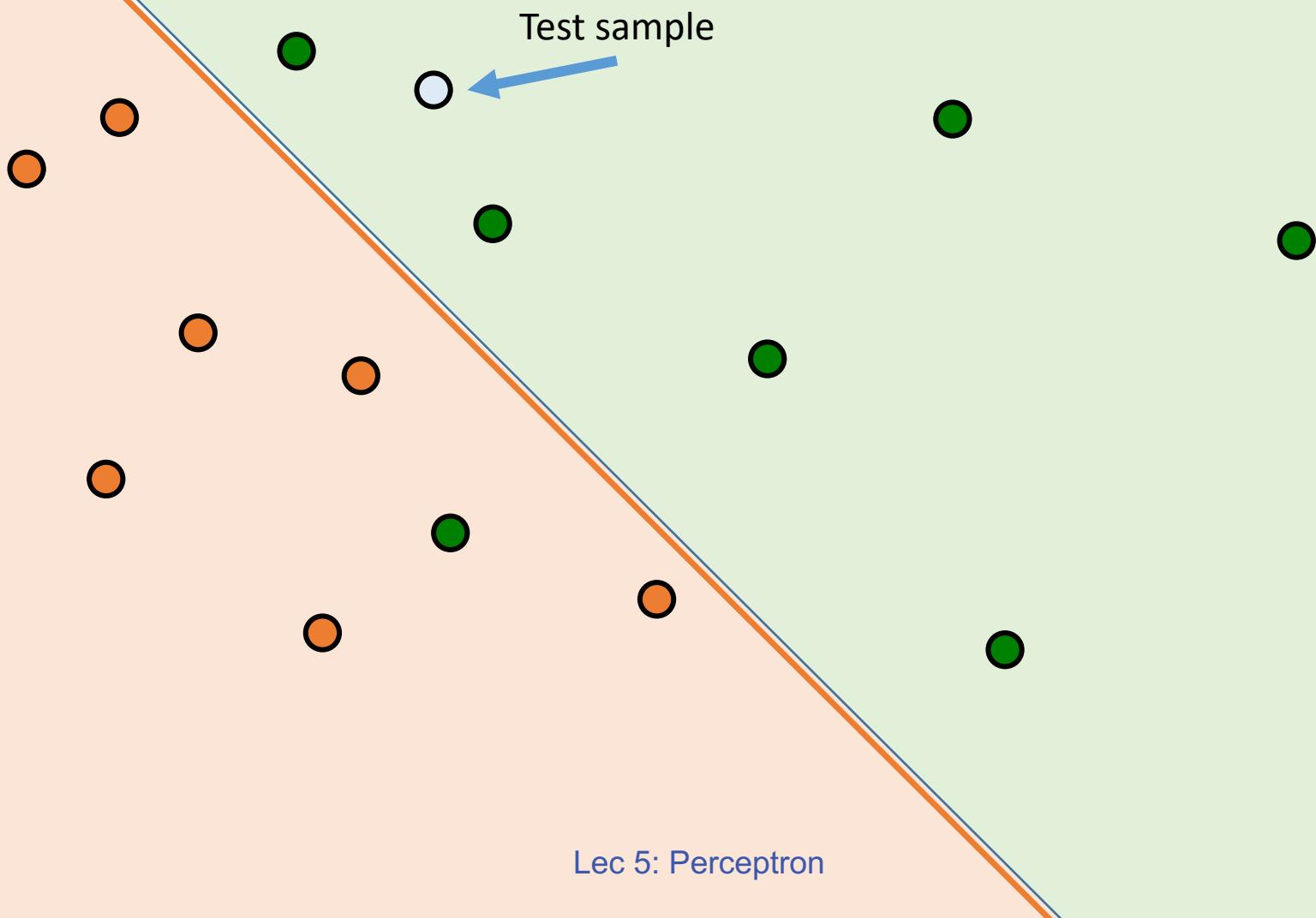
Training data



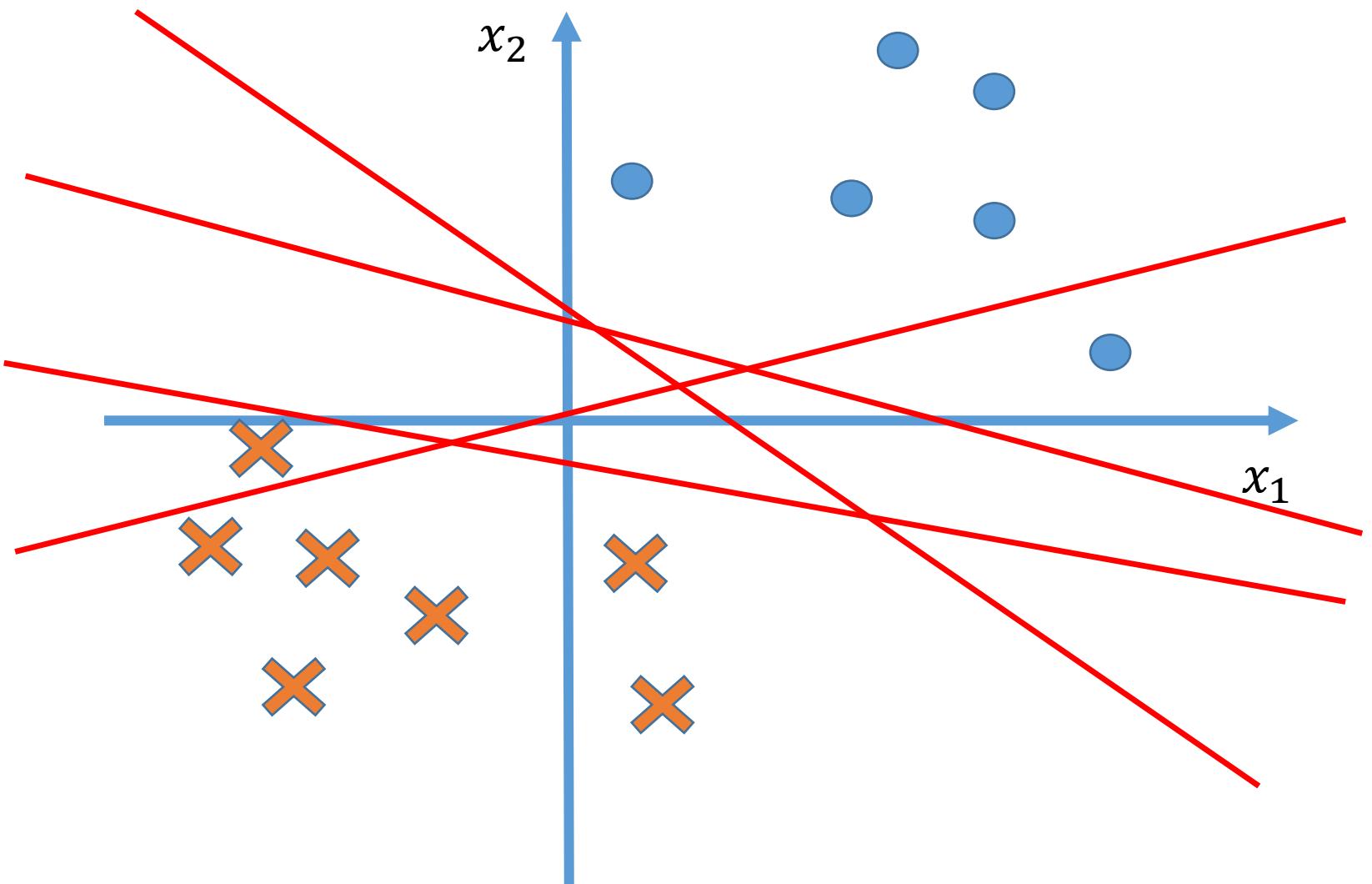
Hyperplane Separates the Space



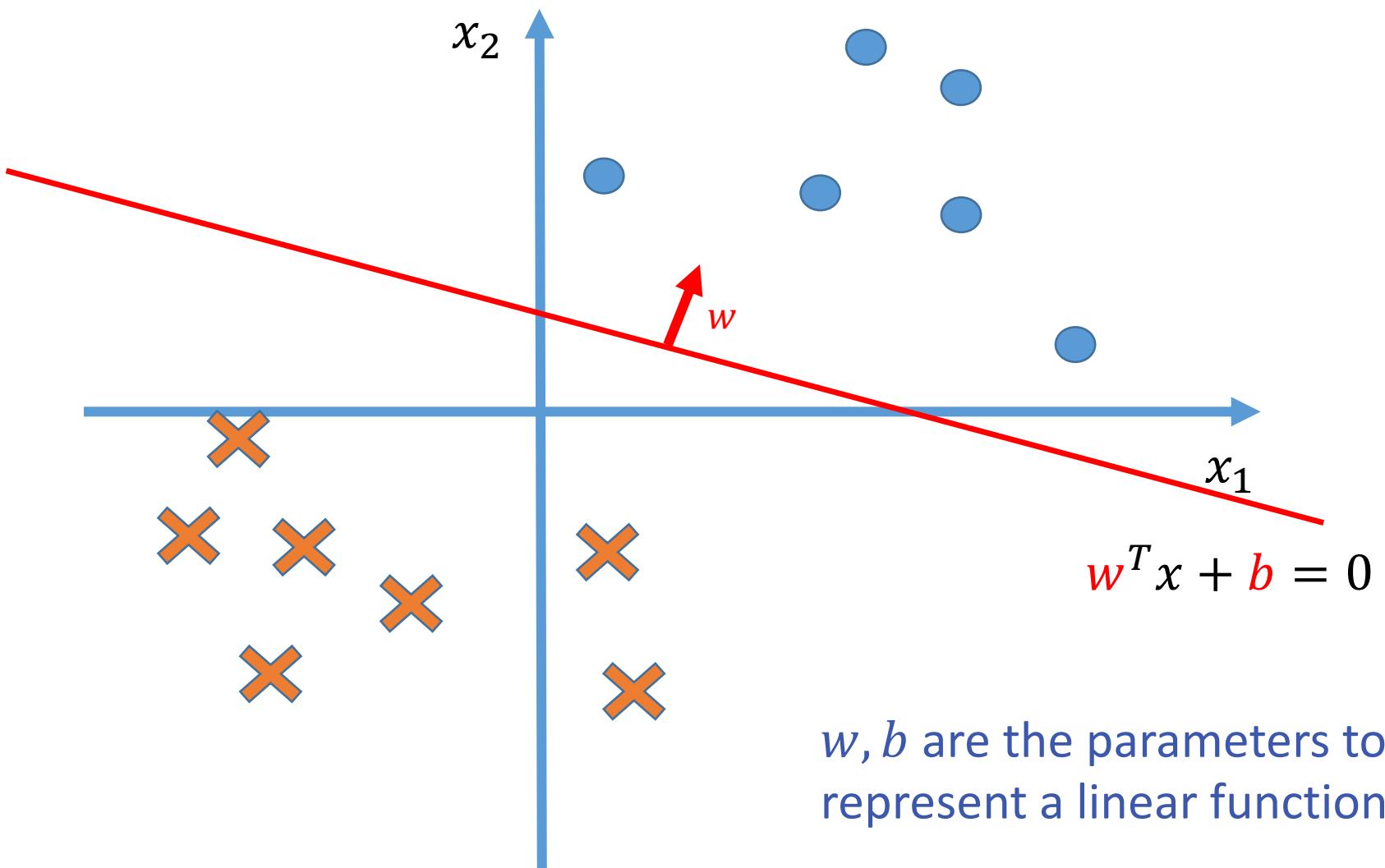
Test Phase



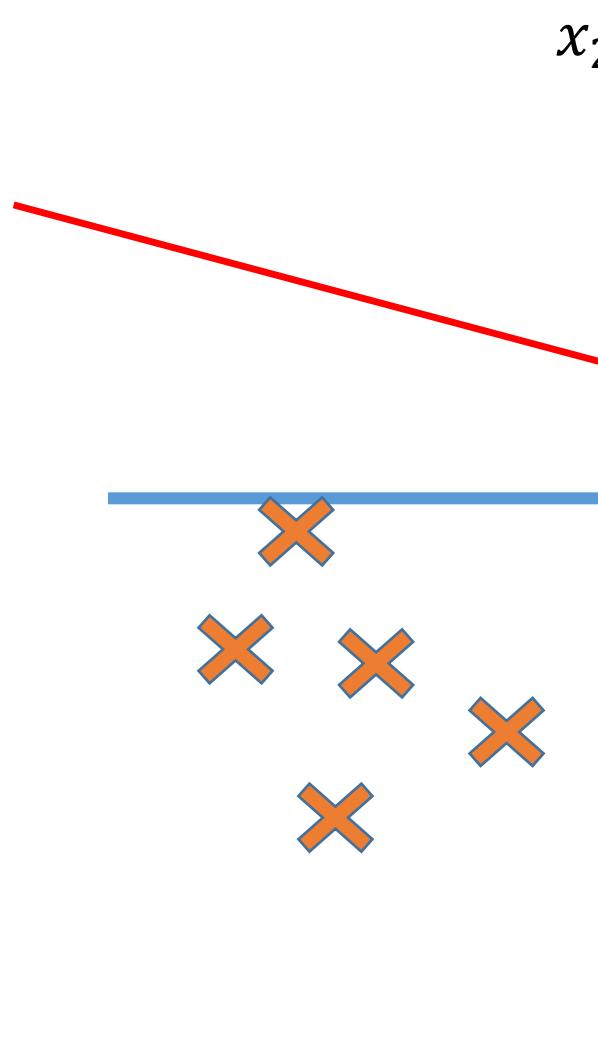
Hypothesis space: linear model



Hypothesis space: linear model



Hypothesis space: linear model



$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

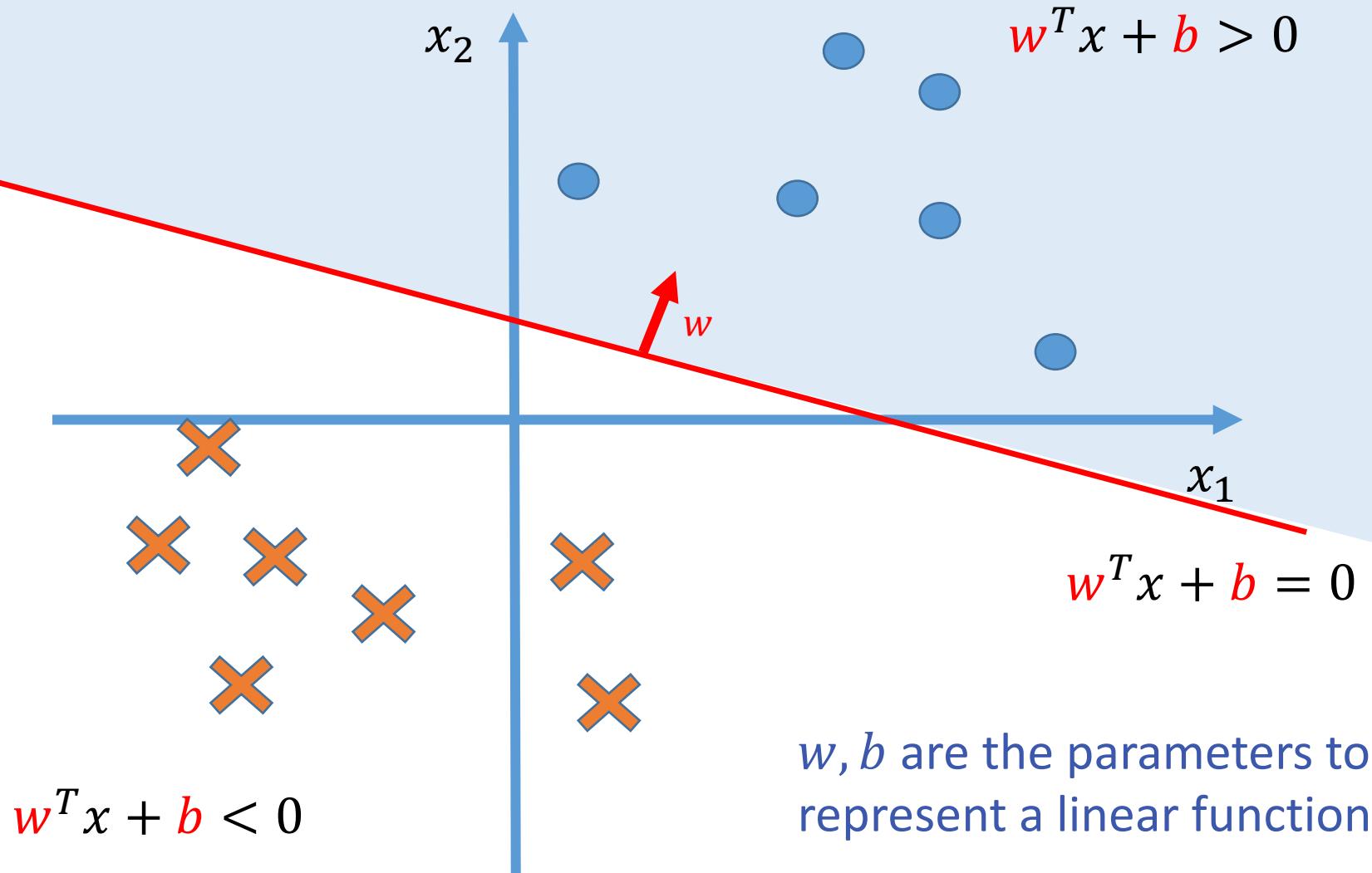
$$w^T x = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

$w^T x$ is the inner product between w and x

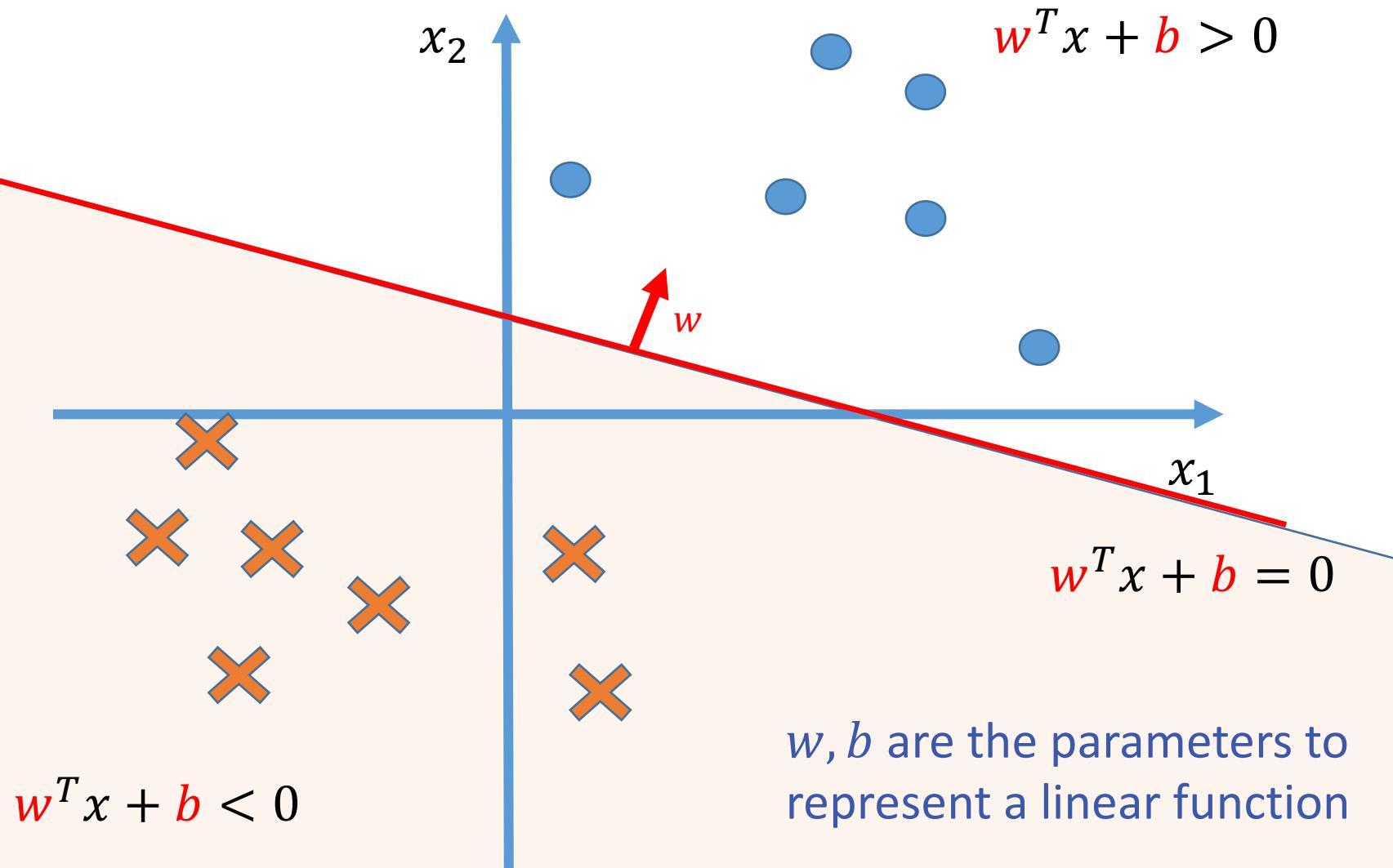
$$w^T x + b = 0$$

w, b are the parameters to represent a linear function

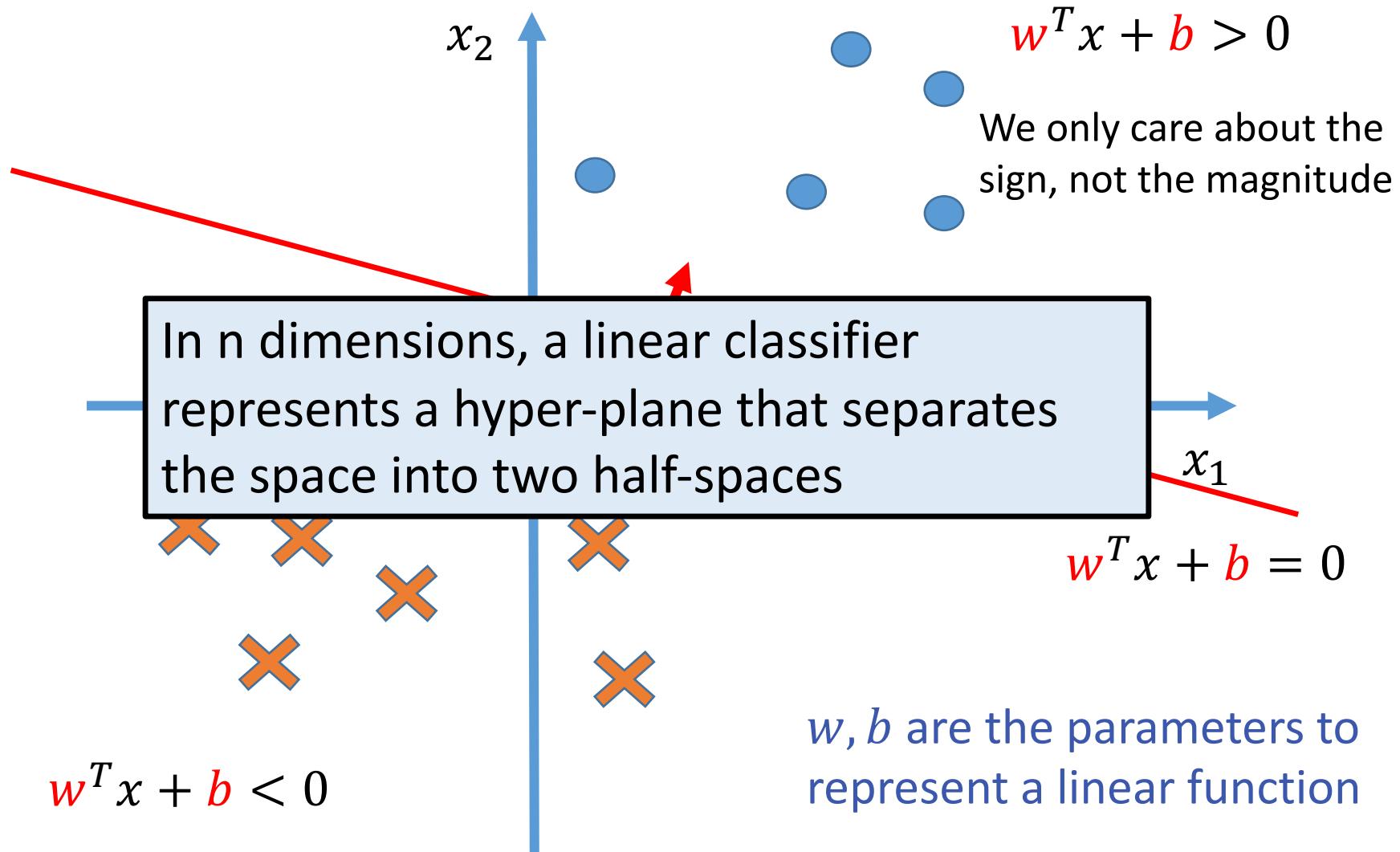
Hypothesis space: linear model



Hypothesis space: linear model

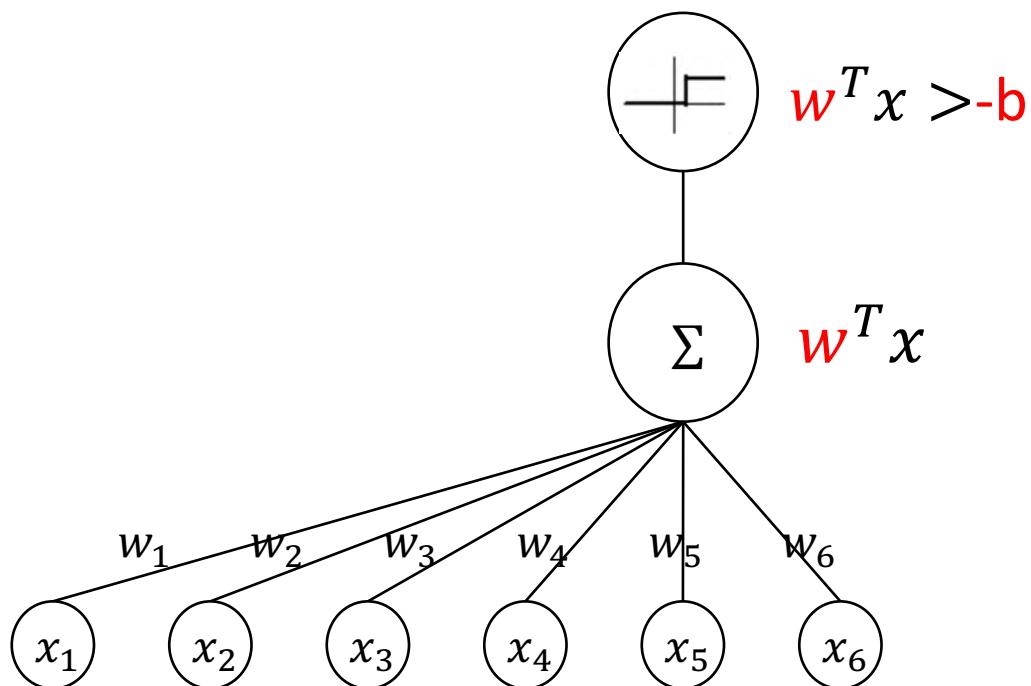


Hypothesis space: linear model



Recall: Linear Classifiers

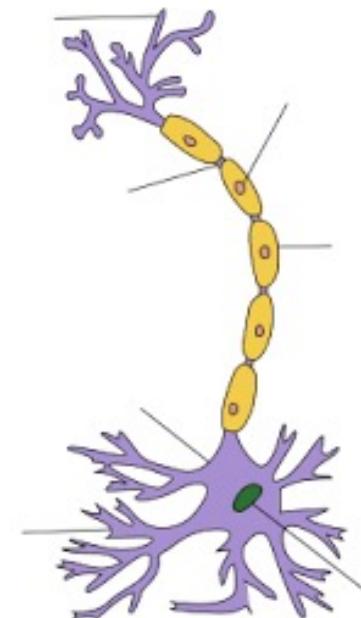
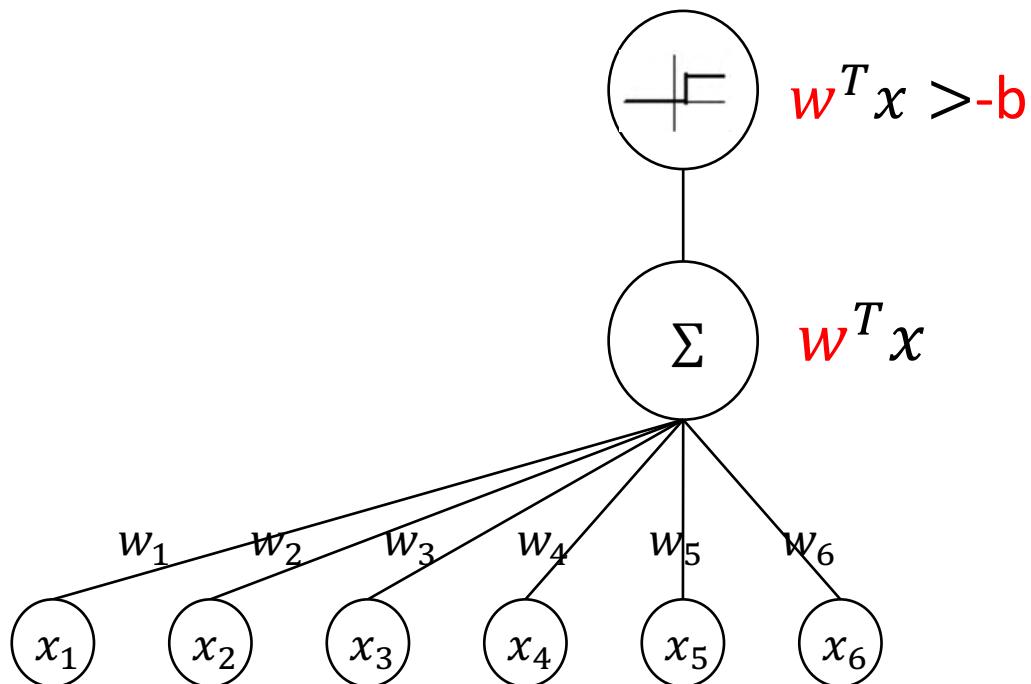
- ❖ *Linear Threshold Units* classify an example \mathbf{x} using the following classification rule



E.g., $0.3 * [\text{first char=a}] + 0.2 * [\text{first char b}] + 2 * [\text{word length}] + \dots - 0.8 > 0$

Recall: Linear Classifiers

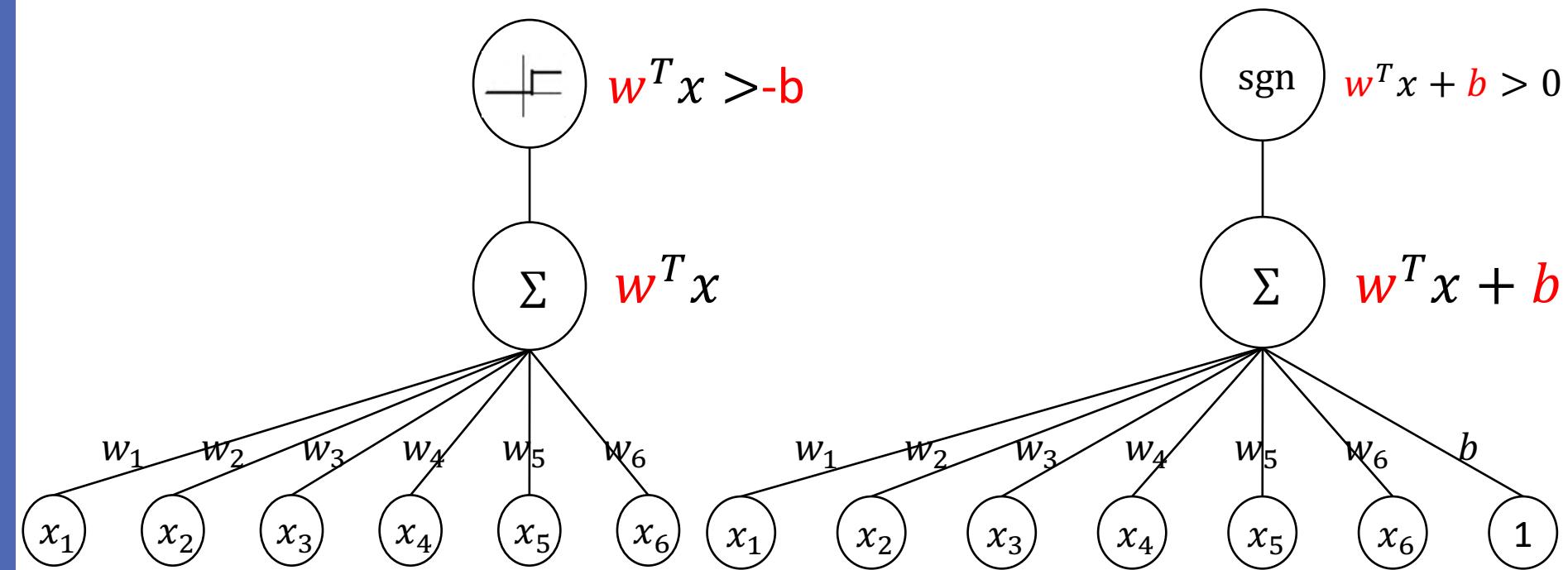
- ❖ *Linear Threshold Units* classify an example \mathbf{x} using the following classification rule



E.g., $0.3 * [\text{first char=a}] + 0.2 * [\text{first char b}] + 2 * [\text{word length}] + \dots - 0.8 > 0$

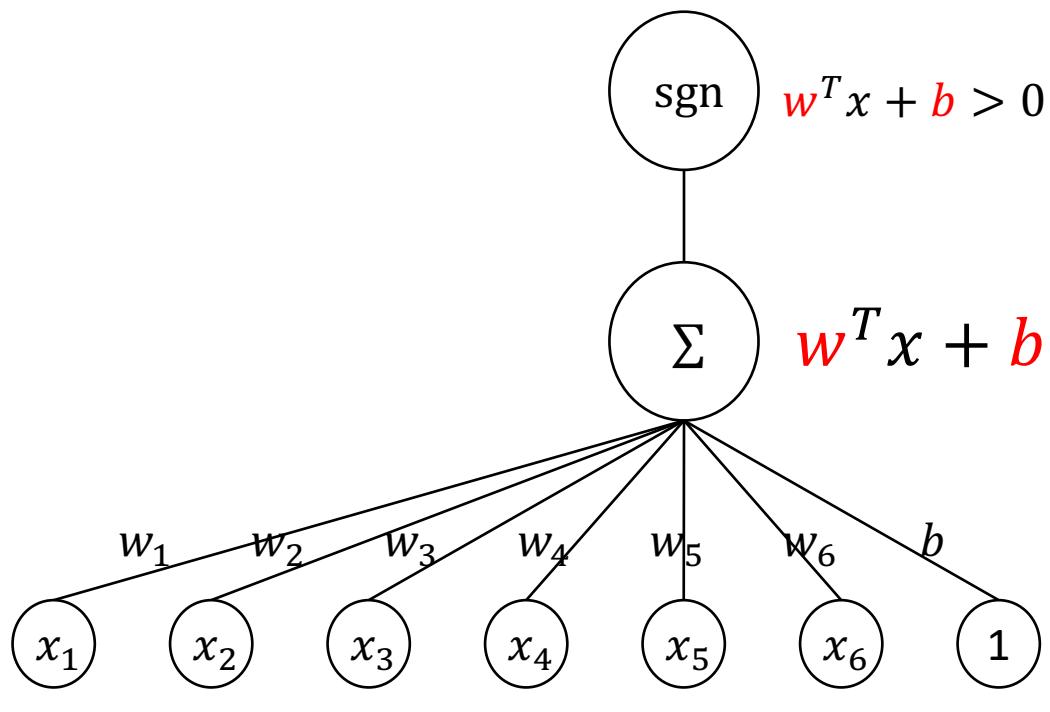
Recall: Linear Classifiers

- ❖ *Linear Threshold Units* classify an example \mathbf{x} using the following classification rule



E.g., $0.3 * [\text{first char=a}] + 0.2 * [\text{first char b}] + 2 * [\text{word length}] + \dots - 0.8 > 0$

A simple trick to remove the bias term b

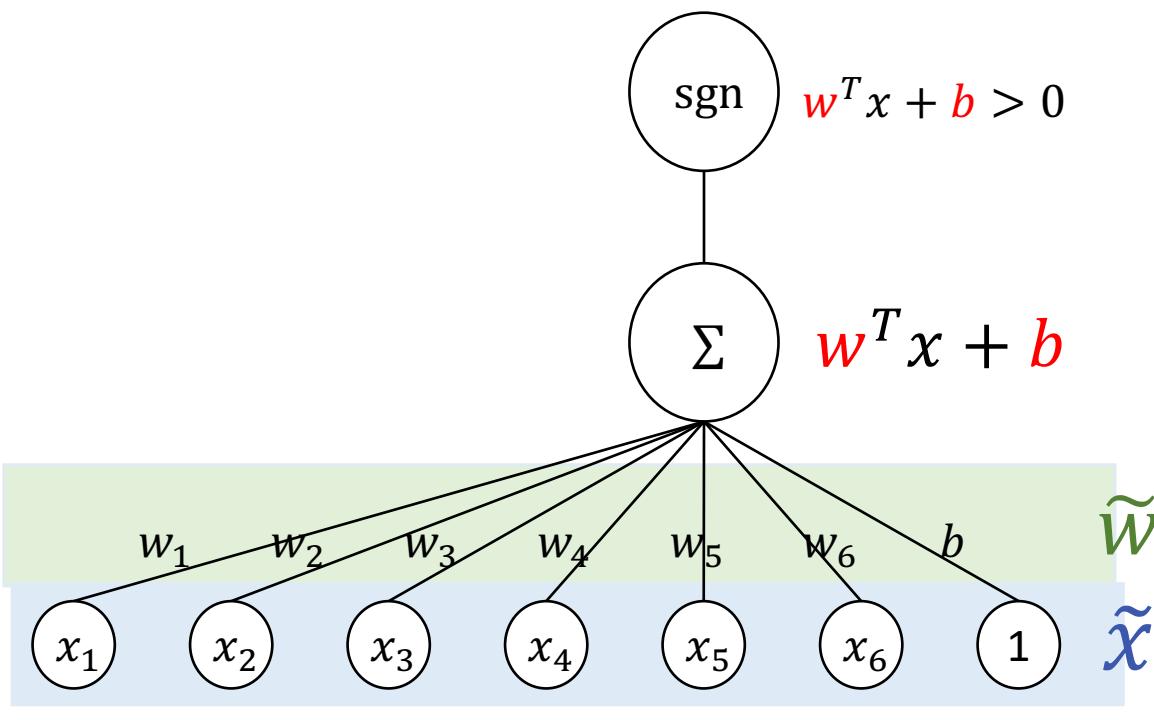


$$\begin{aligned}w^T x + b \\= [w^T \ b] \cdot \begin{bmatrix}x \\ 1\end{bmatrix} \\= \tilde{w} \cdot \tilde{x}\end{aligned}$$

$$\tilde{w} = \begin{bmatrix}w_1 \\ w_2 \\ \vdots \\ w_n \\ b\end{bmatrix}, \tilde{x} = \begin{bmatrix}x_1 \\ x_2 \\ \vdots \\ x_n \\ 1\end{bmatrix}$$

For simplicity, I may write \tilde{w} and \tilde{x} as w and x when there is no confusion

A simple trick to remove the bias term b

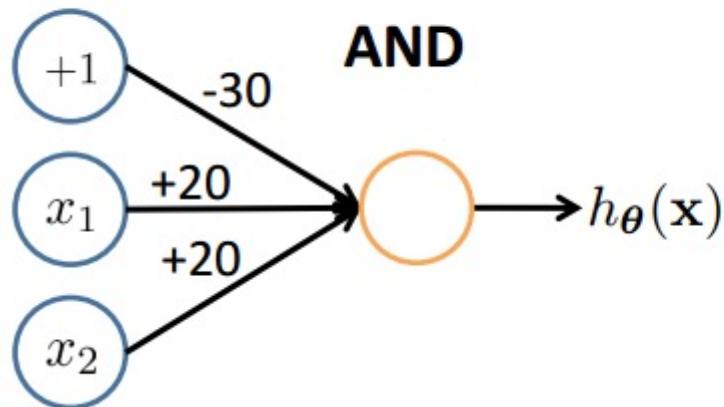


$$\begin{aligned}w^T x + b \\= [w^T \ b] \cdot \begin{bmatrix}x \\ 1\end{bmatrix} \\= \tilde{w} \cdot \tilde{x}\end{aligned}$$

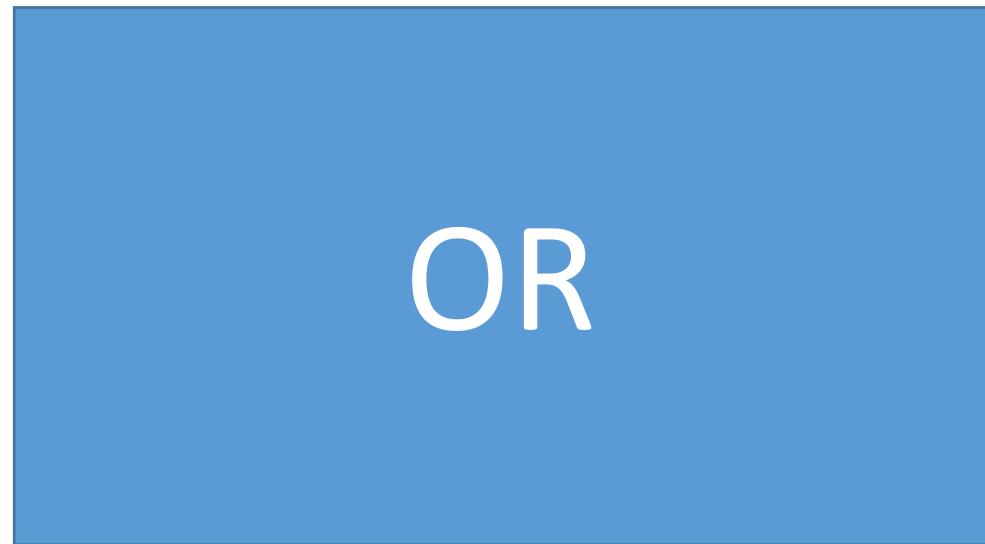
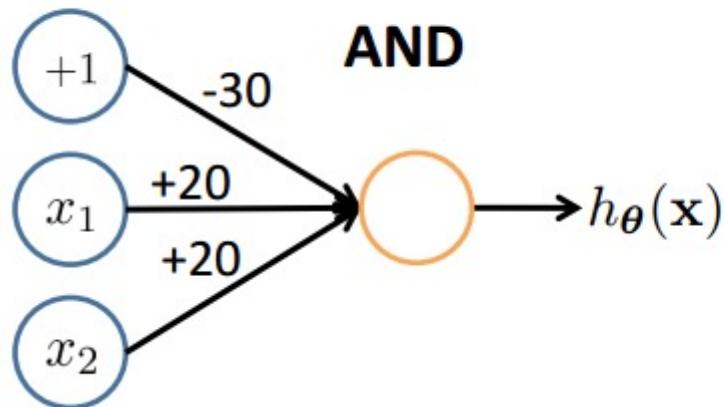
$$\tilde{w} = \begin{bmatrix}w_1 \\ w_2 \\ \vdots \\ w_n \\ b\end{bmatrix}, \tilde{x} = \begin{bmatrix}x_1 \\ x_2 \\ \vdots \\ x_n \\ 1\end{bmatrix}$$

For simplicity, I may write \tilde{w} and \tilde{x} as w and x when there is no confusion

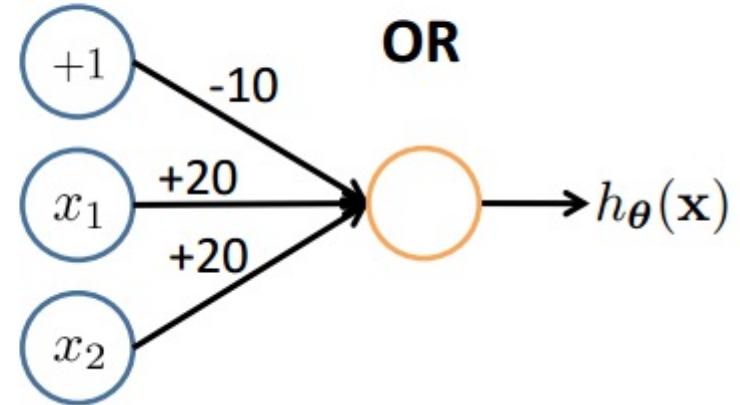
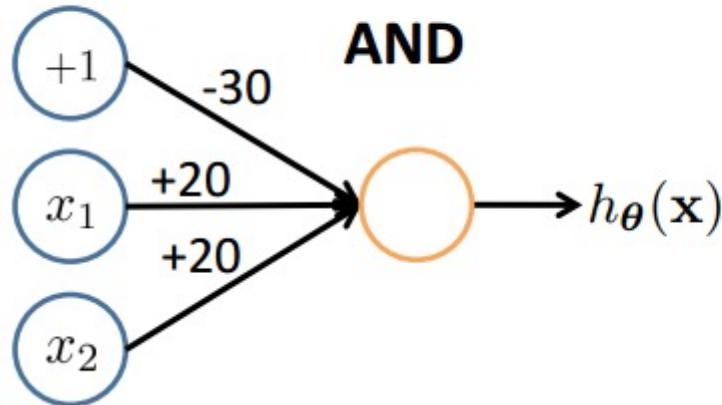
Representing Boolean Functions



Representing Boolean Functions



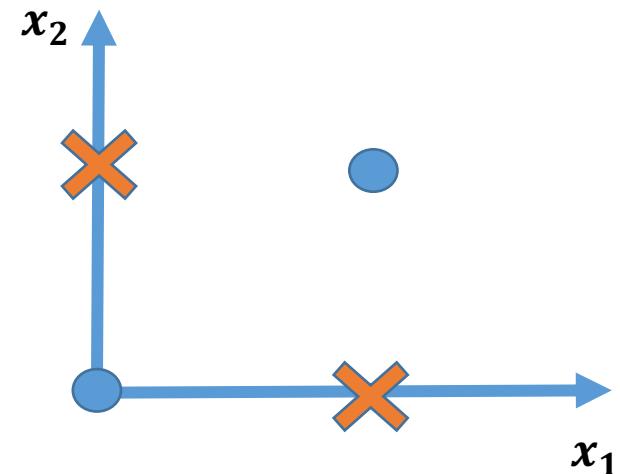
Representing Boolean Functions



Limitation

- ❖ Can linear model represent XNOR ?

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1

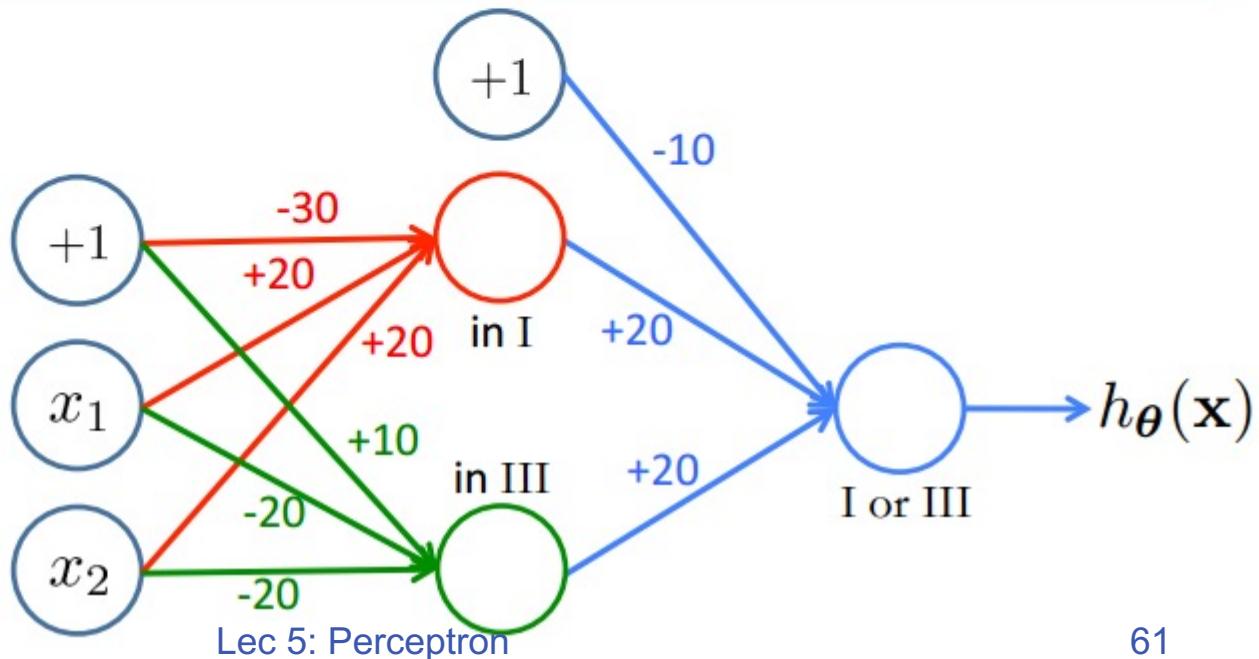
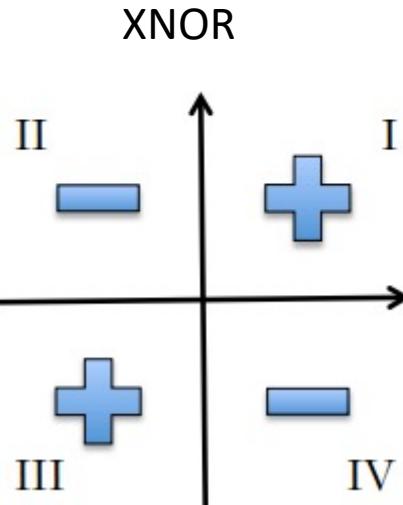
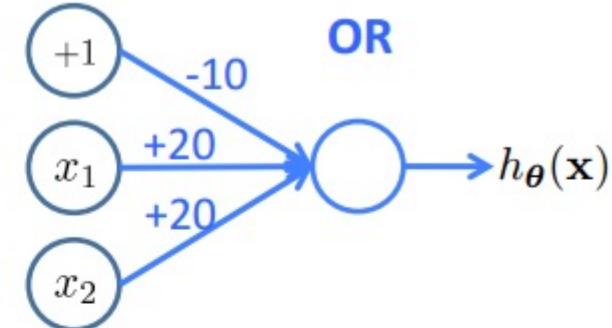
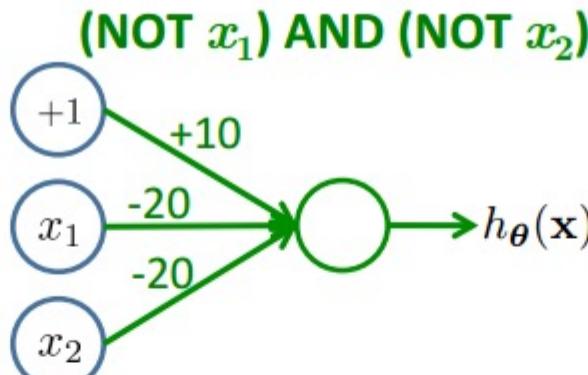
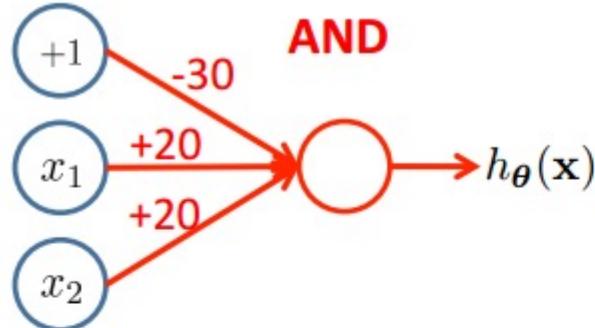


Assume the separating hyper plane is $w_1x_1 + w_2x_2 + b = 0$

From the four points we have:

$$\begin{aligned} w_1 + b &< 0 \\ w_2 + b &< 0 \\ b \geq 0 \\ w_1 + w_2 + b &\geq 0 \end{aligned} \quad \left. \begin{array}{l} w_1 + b < 0 \\ w_2 + b < 0 \\ w_1 + w_2 + b \geq 0 \end{array} \right\} \quad \left. \begin{array}{l} w_1 + b < 0 \\ w_2 + b < 0 \end{array} \right\} \quad w_1 + w_2 + b < 0$$

Multi-layer Perceptron (NN)



Learning a Linear Classifier

- ❖ There are several algorithms/models
 - ❖ Perceptron
 - ❖ Logistic Regression
 - ❖ (Linear) Support Vector Machines
 - ❖ ...
- ❖ Based on different assumptions, you get different linear models

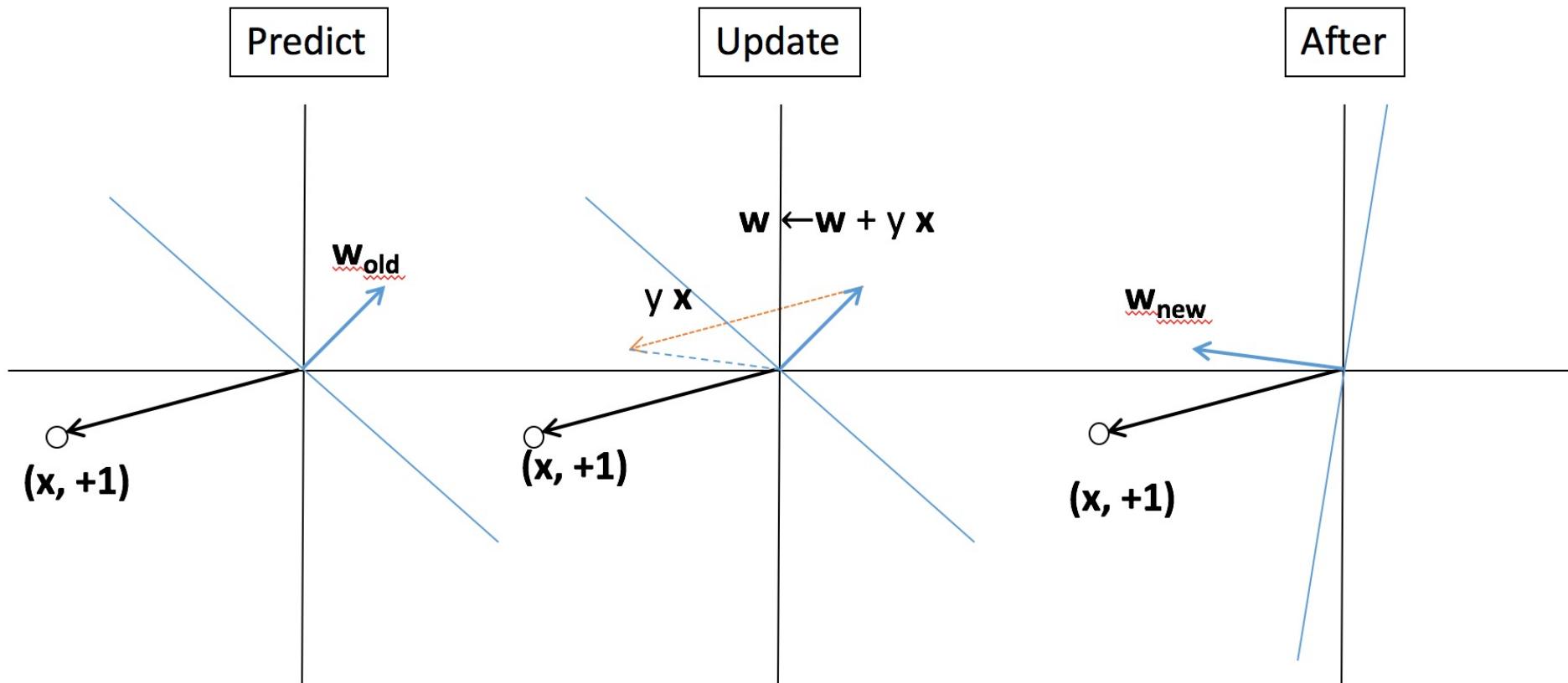
The Perceptron Algorithm

mistake
+
correction
=

learning

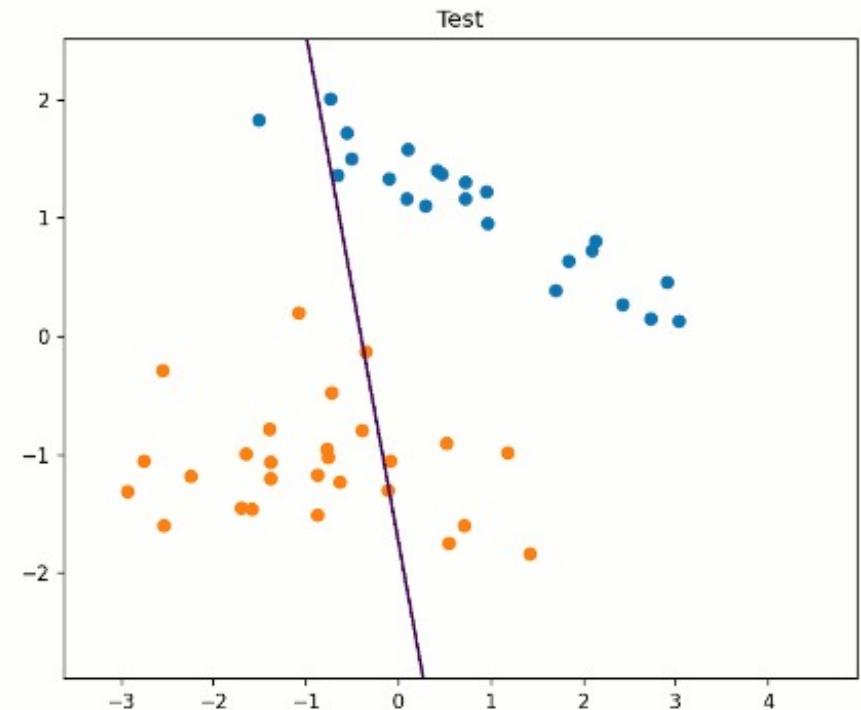
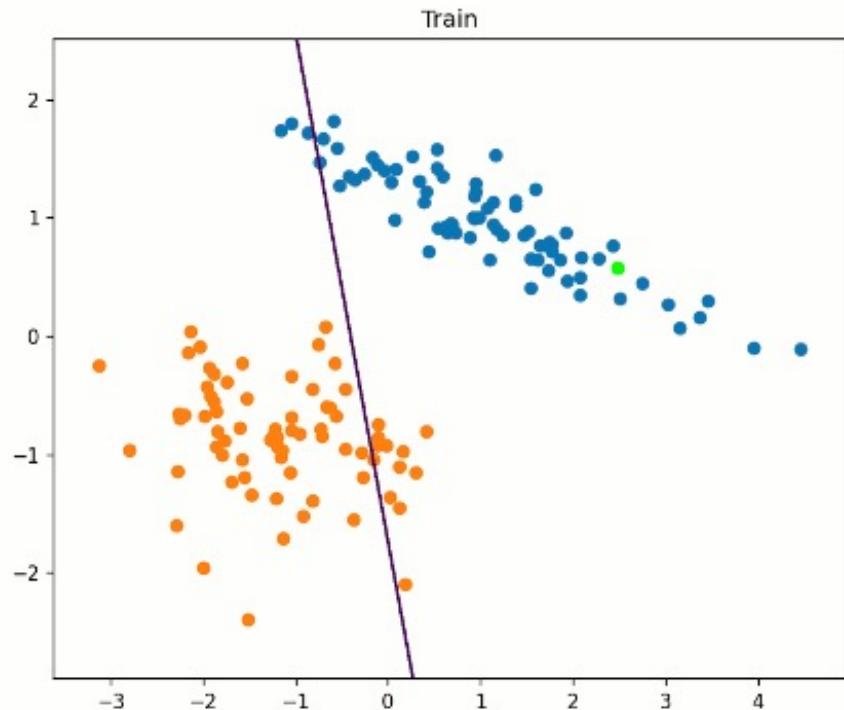
Today: Perceptron

- ❖ Learning by making mistakes



Perceptron in Action

Iteration: 1/2; Point: 1/150



<https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-3c8e76b4e2d1>

The Perceptron

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN¹

F. ROSENBLATT

Cornell Aeronautical Laboratory

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44

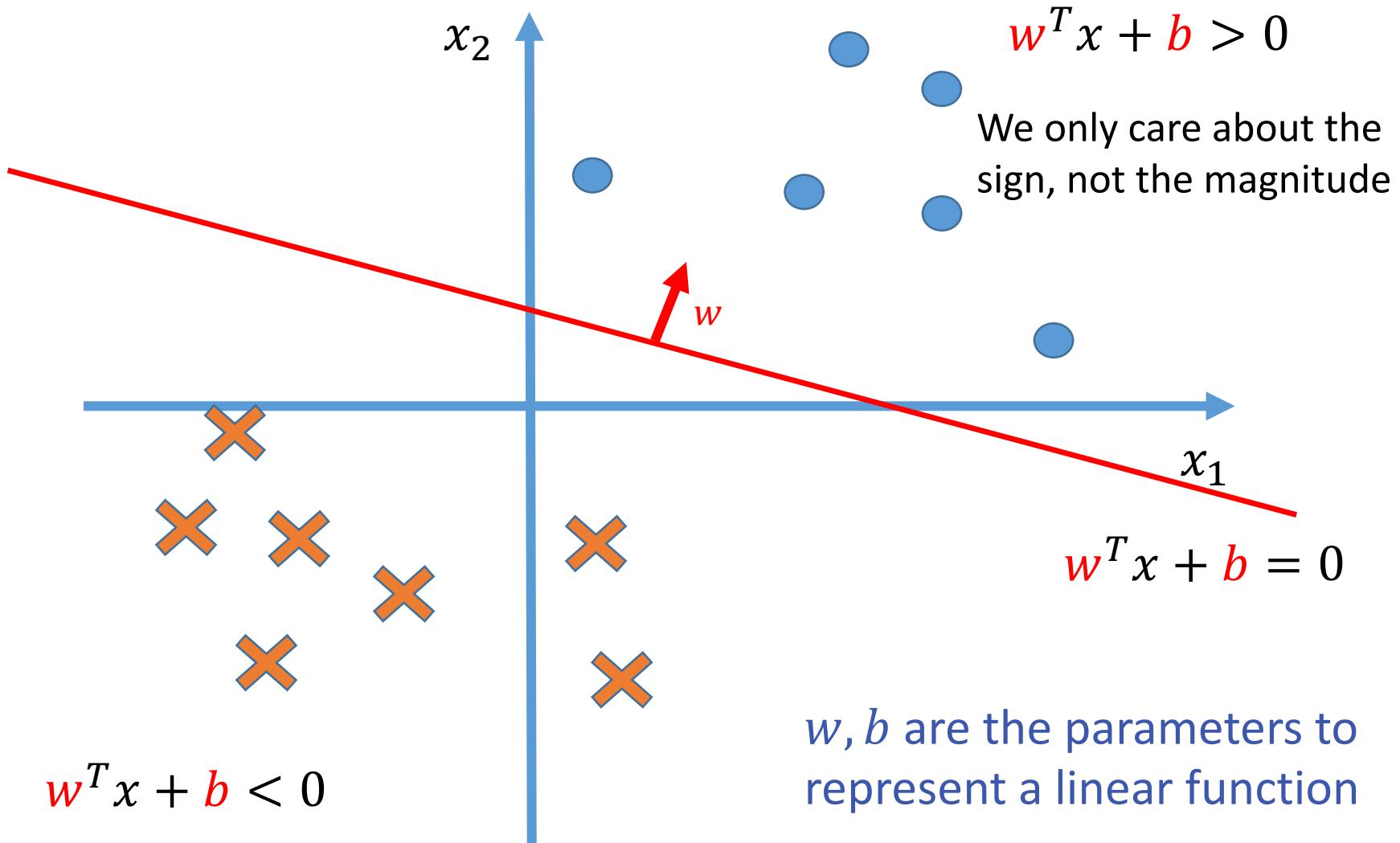




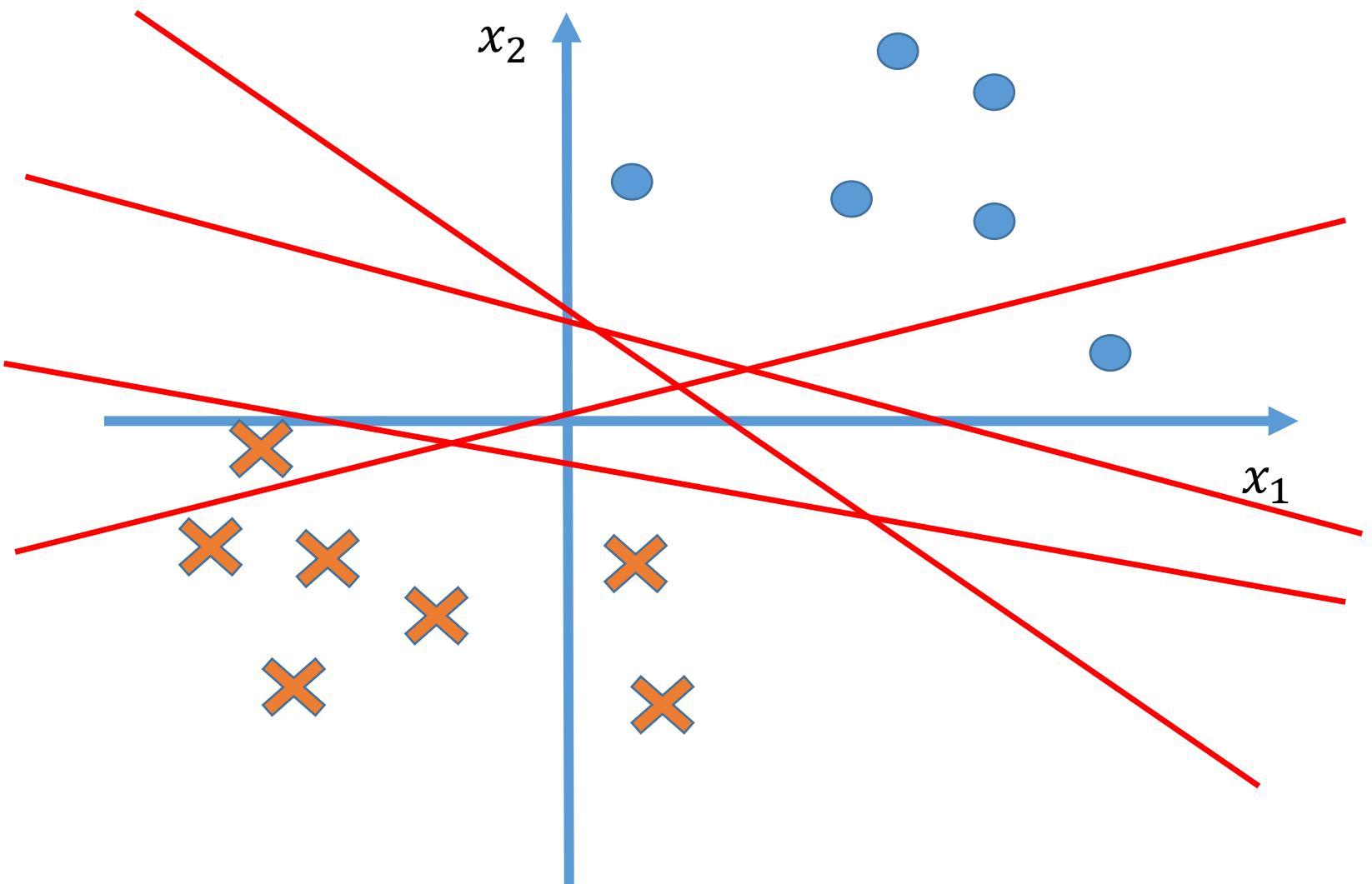
The Perceptron algorithm

- ❖ Rosenblatt 1958
- ❖ The goal is to find a separating hyperplane
 - ❖ For separable data, guaranteed to find one
- ❖ An **online** algorithm
 - ❖ Processes one example at a time
- ❖ Converges if data is separable
 - mistake bound

Hypothesis space: linear model



How to find the best linear model?

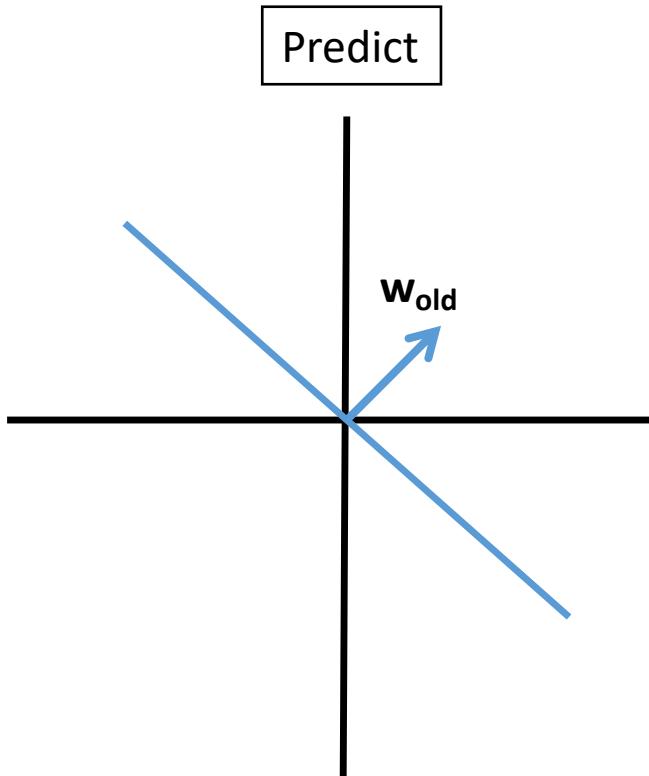


The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

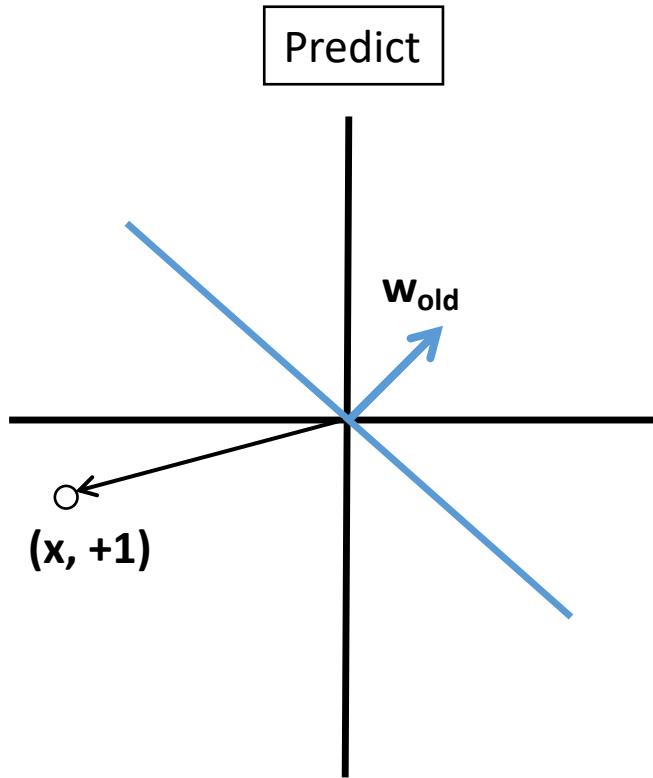
Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Geometry of the perceptron update



Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

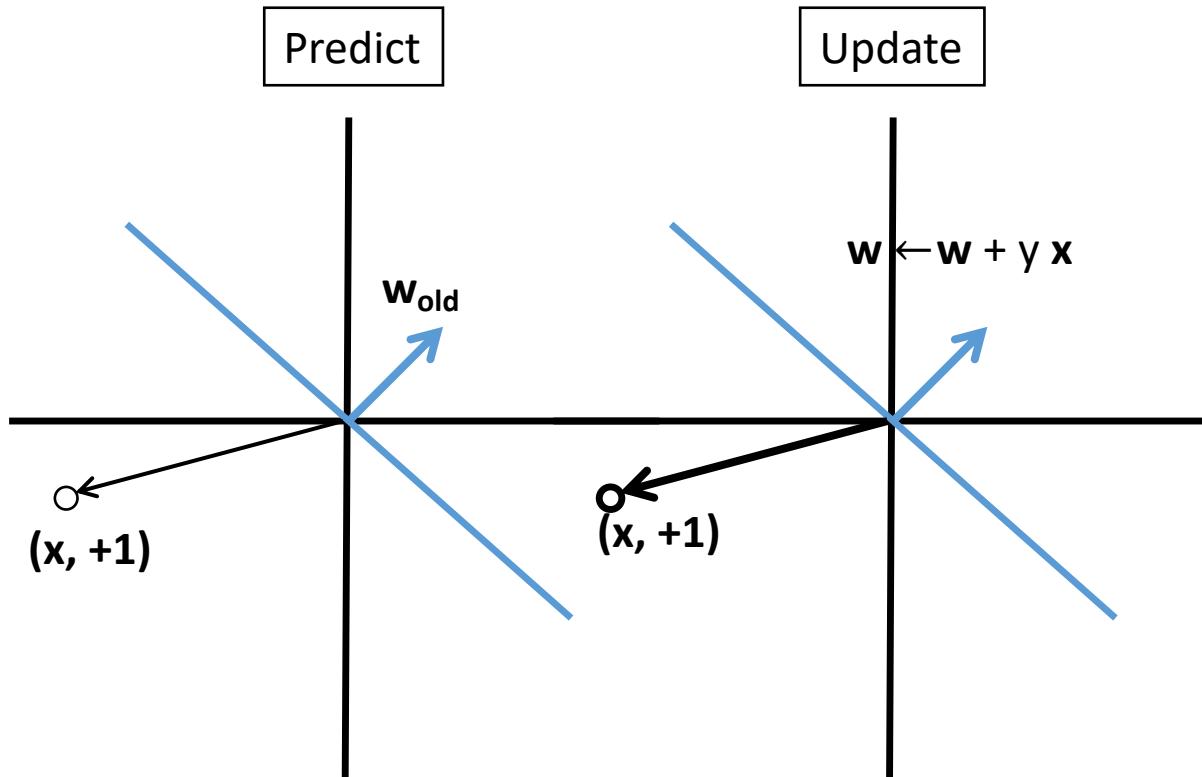
Geometry of the perceptron update



Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

Geometry of the

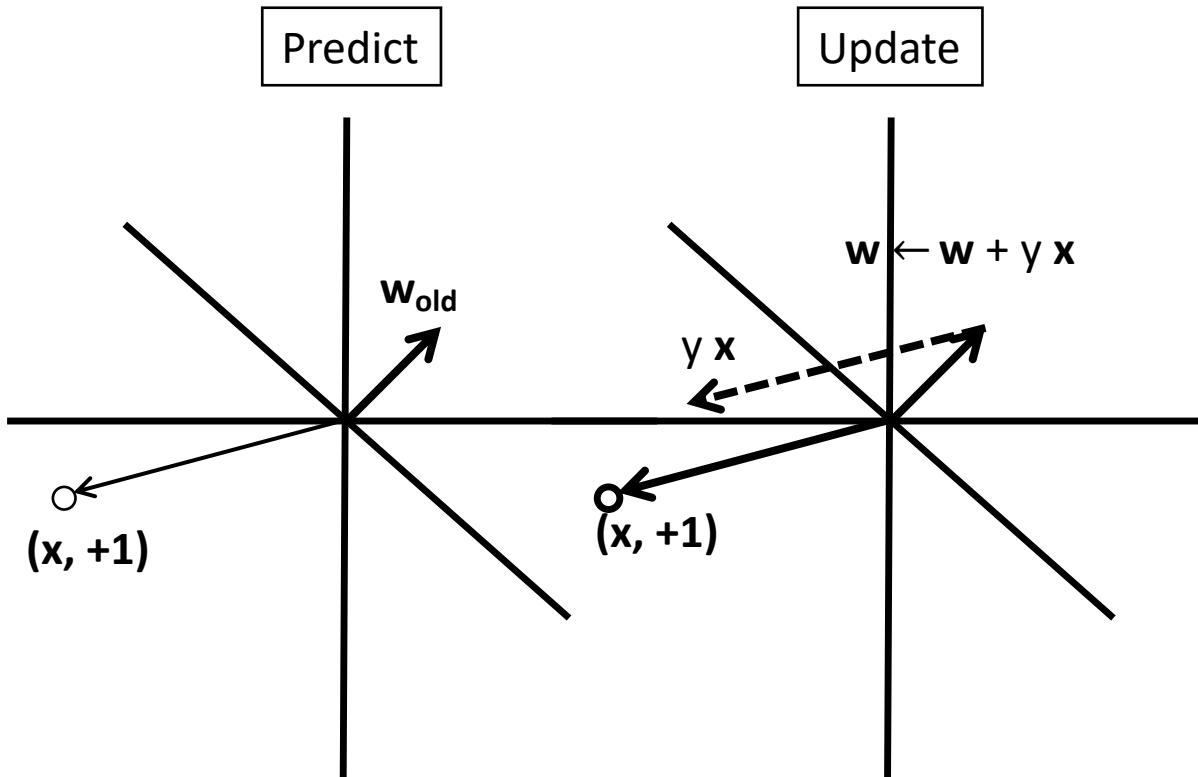
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

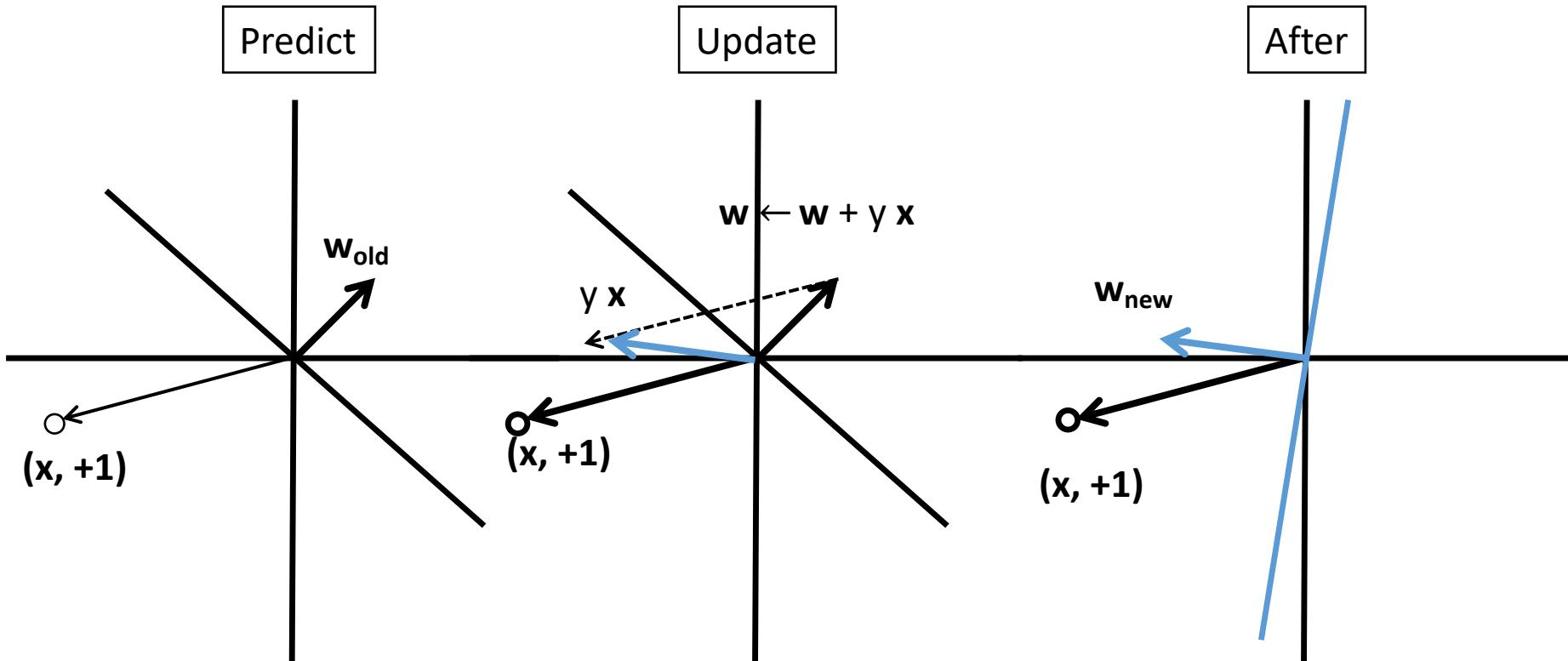
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

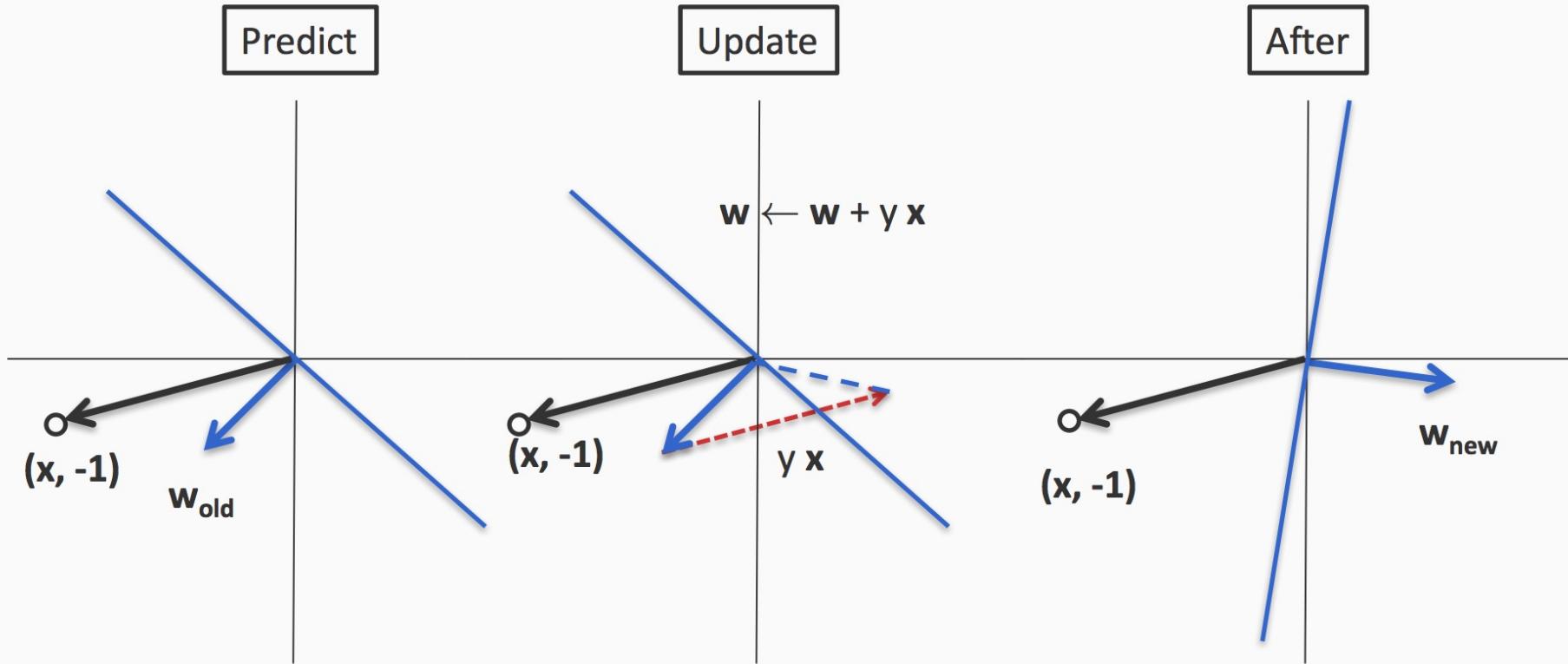
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **negative** example

Intuition behind the update

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Suppose we have made a mistake on a positive example

That is, $y = +1$ and $\mathbf{w}_t^T \mathbf{x} \leq 0$

Call the new weight vector $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$

The new dot product will be

$$\mathbf{w}_{t+1}^T \mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T \mathbf{x} = \mathbf{w}_t^T \mathbf{x} + \mathbf{x}^T \mathbf{x}$$

For a positive example, the Perceptron update will increase the score assigned to the same input

Similar reasoning for negative examples

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
 2. For (\mathbf{x}, y) in \mathcal{D} :
 3. $\hat{y} = \text{sgn}(\mathbf{w}^\top \mathbf{x})$
 4. if $\hat{y} \neq y$, $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
 - 5.
 6. Return \mathbf{w}

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (\mathbf{x}, y) in \mathcal{D} :
3. if $y(\mathbf{w}^\top \mathbf{x}) \leq 0$
4. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- 5.
6. Return \mathbf{w}

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Moral quotes for life



Check point: What you need to know

- ❖ The Perceptron algorithm
- ❖ The geometry of the update

Lecture 6: Perceptron Logistic Regression

Fall 2021

Kai-Wei Chang
CS @ UCLA

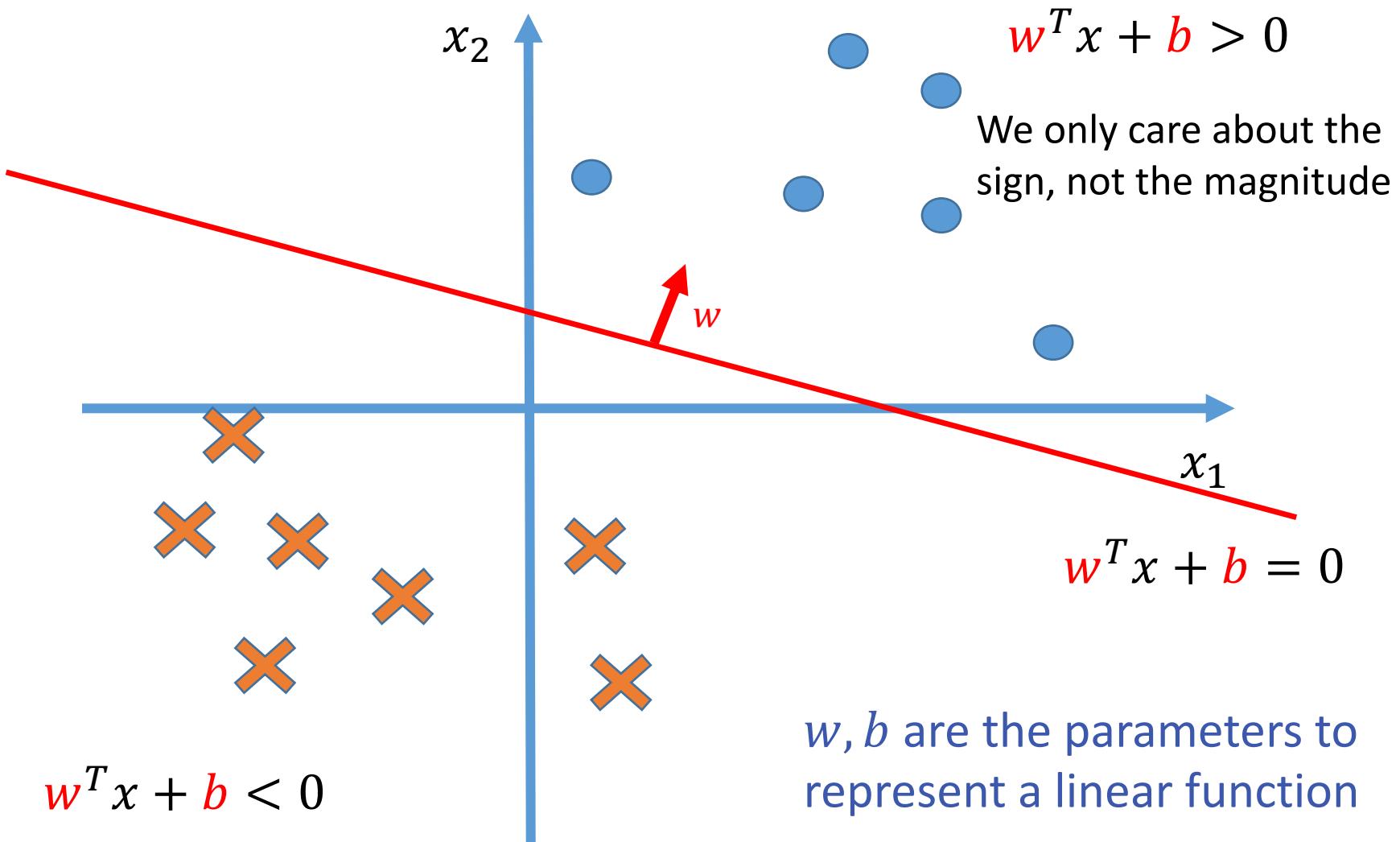
kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

- ❖ Hw1 will be due 10/25 (11:59 PM)
- ❖ Quiz will be due next Mon (11:59 PM)
- ❖ Hw2 will be announced next Thu

Hypothesis space: linear model



The Perceptron

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN¹

F. ROSENBLATT

Cornell Aeronautical Laboratory

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

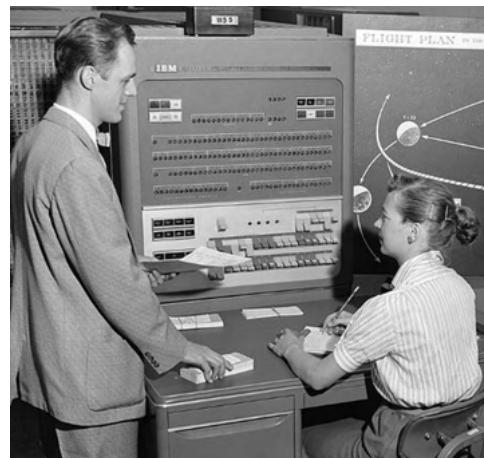
WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44



Lec 6: Perceptron & Logistic

Regression

The IBM 704 computer

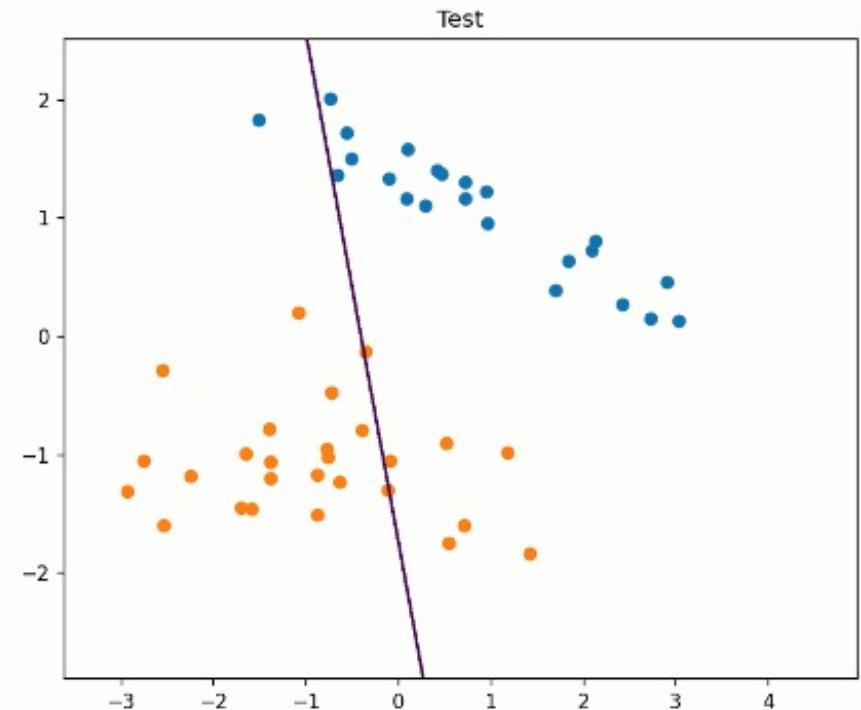
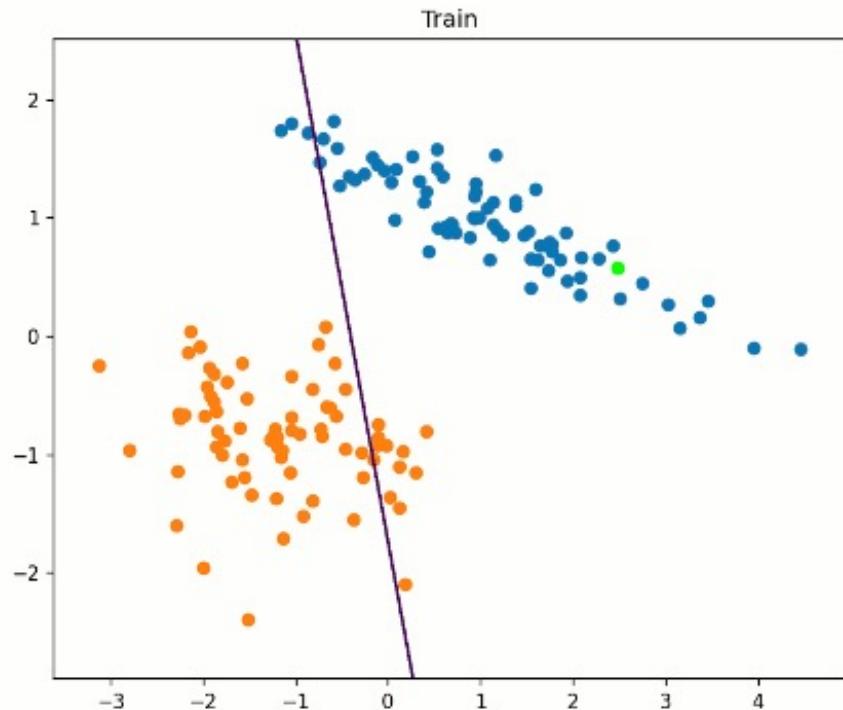


Note: The application scenario discussed in the video is not ideal.

mistake
+
correction
=
learning

Perceptron in Action

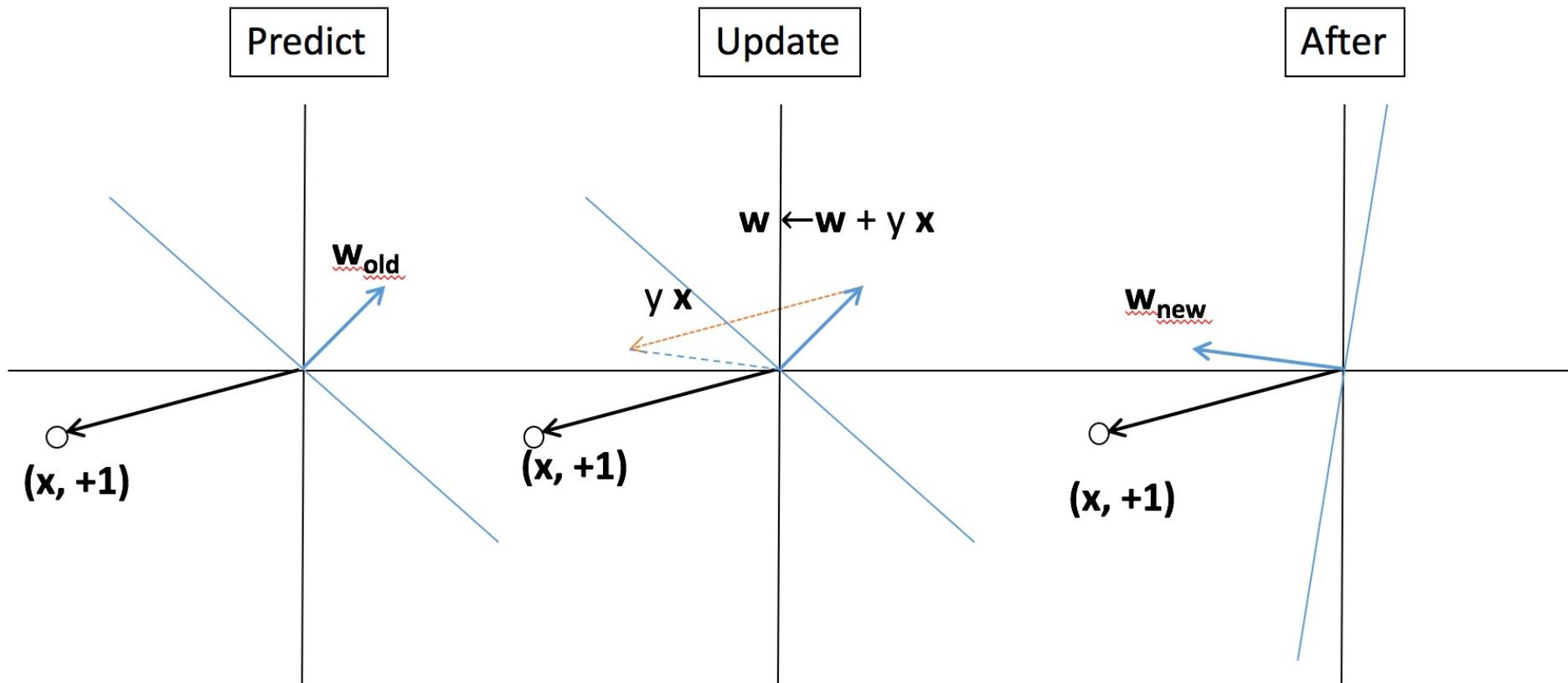
Iteration: 1/2; Point: 1/150



<https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-3c8e76b4e2d1>

Perceptron

- ❖ Learning by making mistakes

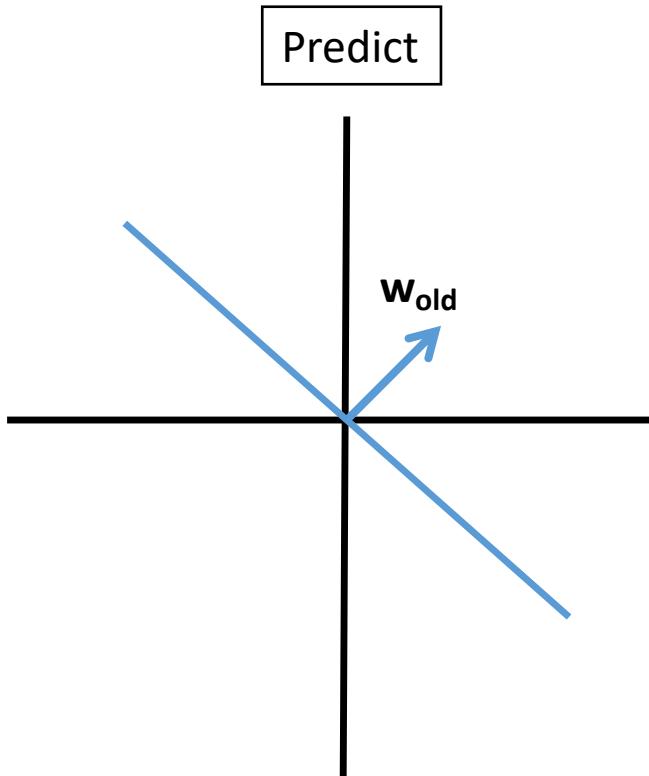


The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

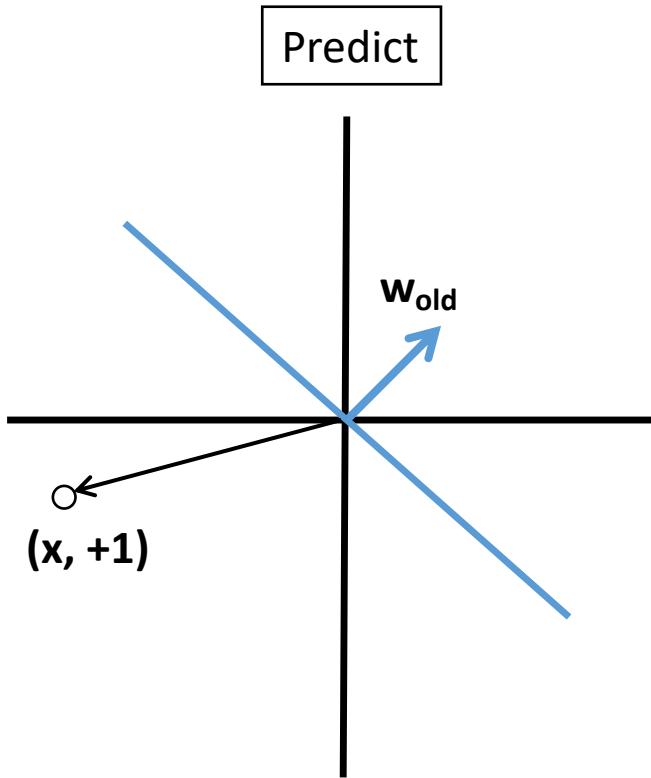
Geometry of the perceptron update



Predict

Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

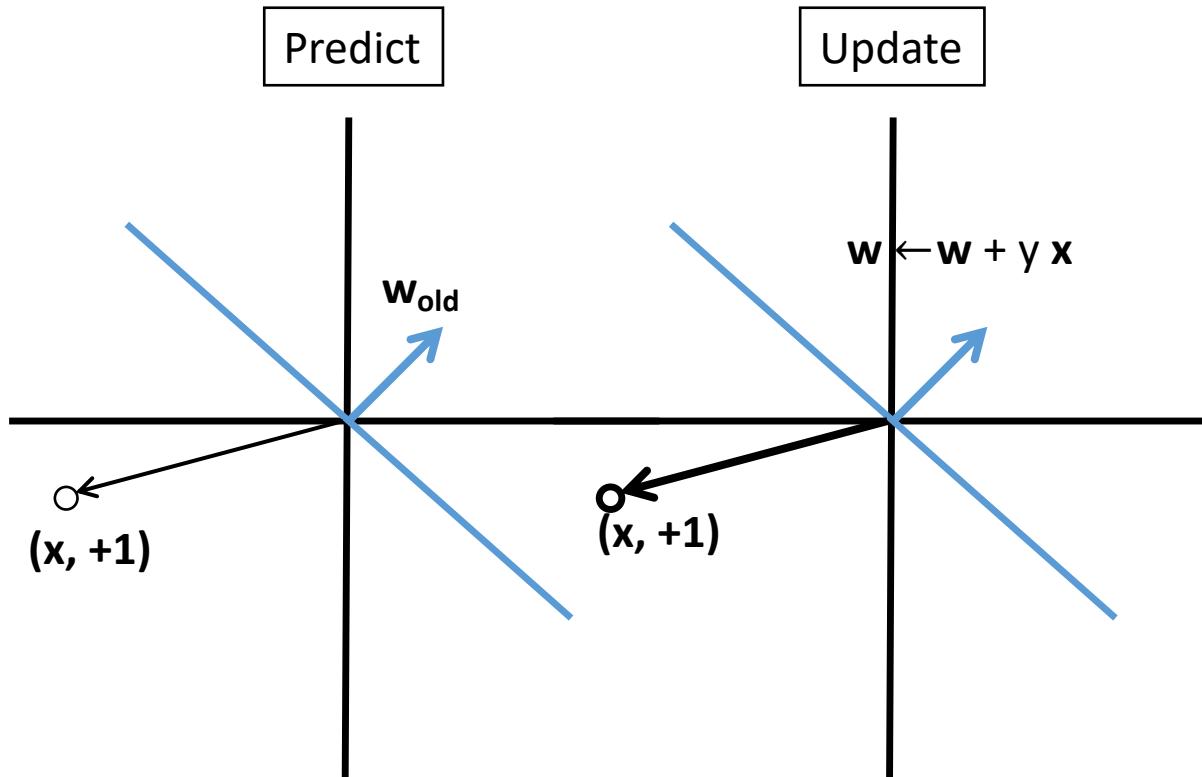
Geometry of the perceptron update



Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

Geometry of the

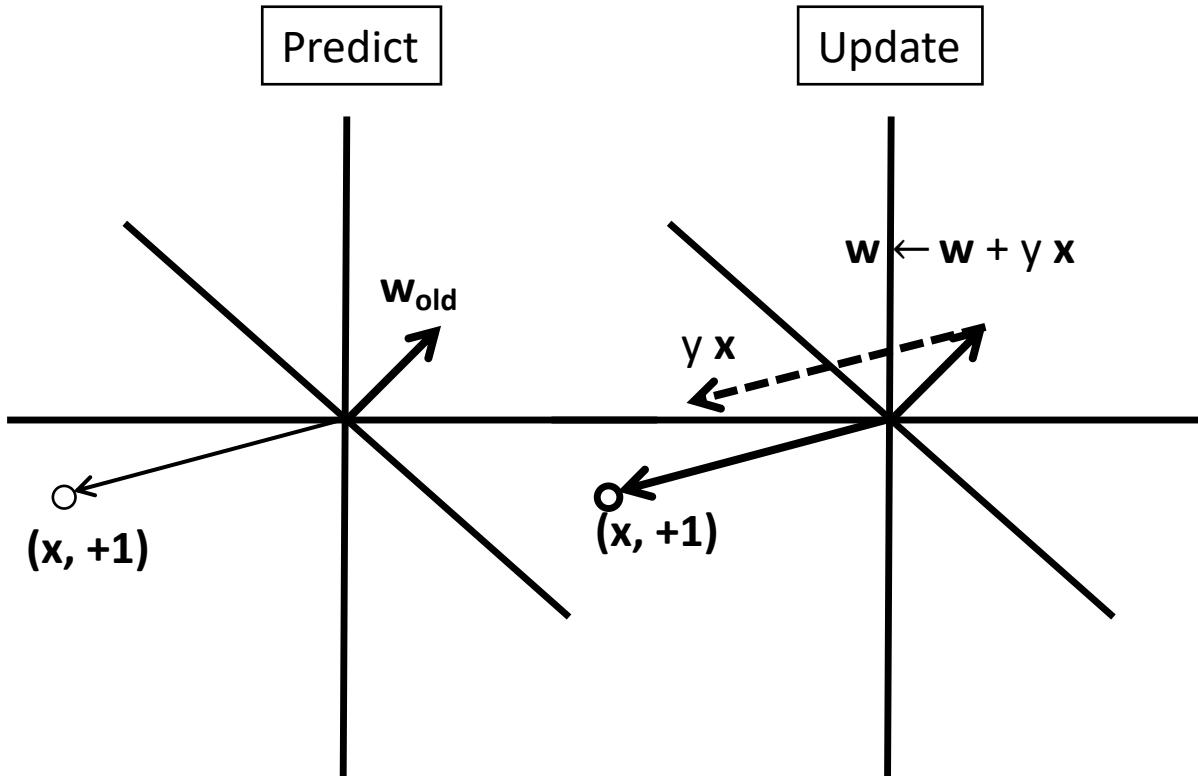
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

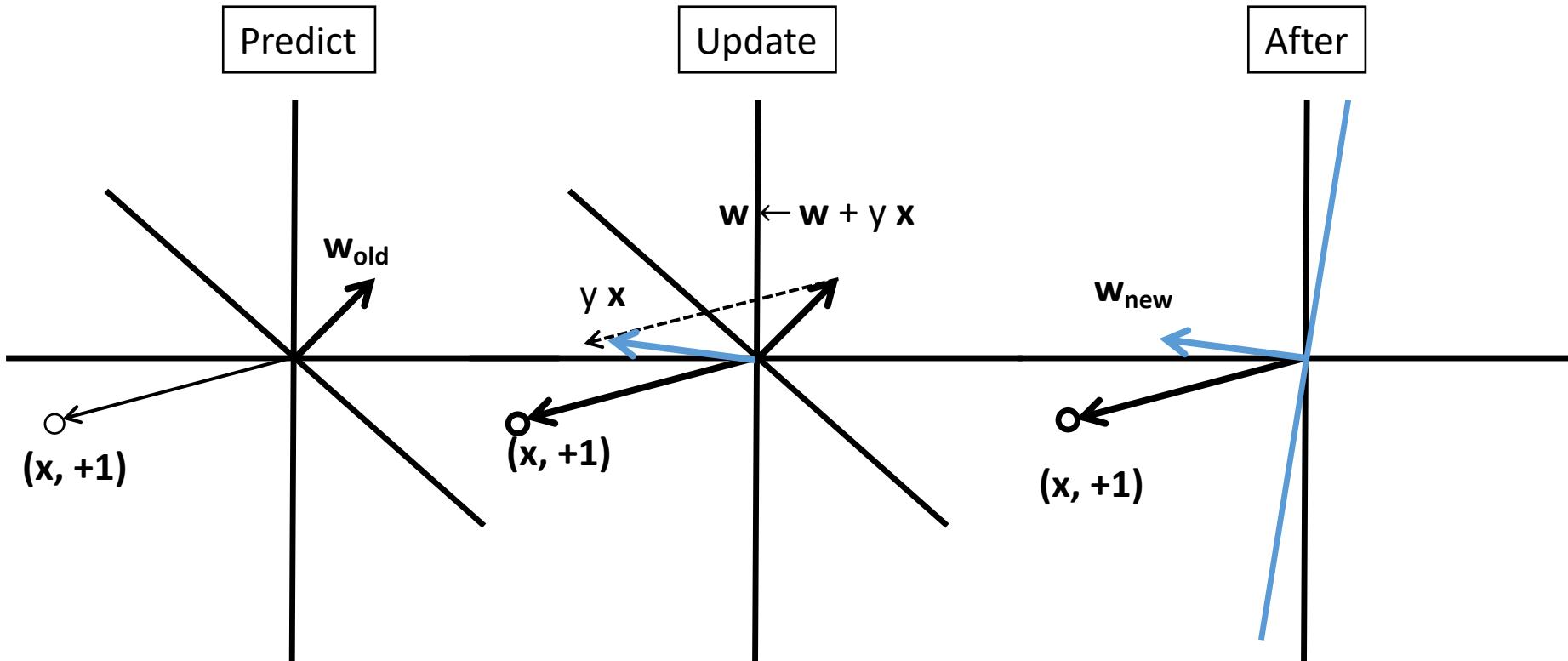
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

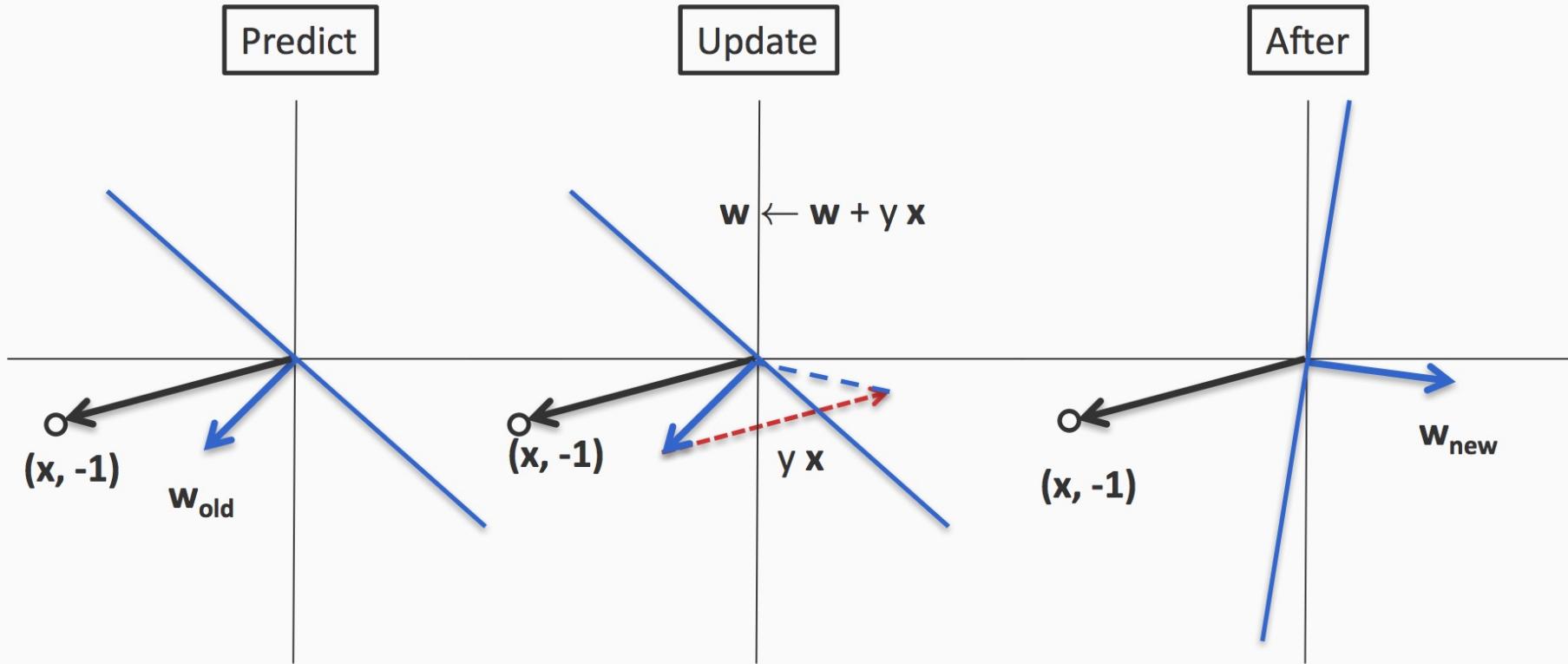
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **positive** example

Geometry of the

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a **negative** example

Intuition behind the update

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Suppose we have made a mistake on a positive example

That is, $y = +1$ and $\mathbf{w}_t^T \mathbf{x} \leq 0$

Call the new weight vector $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$

The new dot product will be

$$\mathbf{w}_{t+1}^T \mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T \mathbf{x} = \mathbf{w}_t^T \mathbf{x} + \mathbf{x}^T \mathbf{x}$$

For a positive example, the Perceptron update will increase the score assigned to the same input

Similar reasoning for negative examples

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
 2. For (x, y) in \mathcal{D} :
 3. $\hat{y} = \text{sgn}(w^\top x)$
 4. if $\hat{y} \neq y$, $w \leftarrow w + yx$
 - 5.
 6. Return w

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (\mathbf{x}, y) in \mathcal{D} :
3. if $y(\mathbf{w}^\top \mathbf{x}) \leq 0$
4. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- 5.
6. Return \mathbf{w}

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

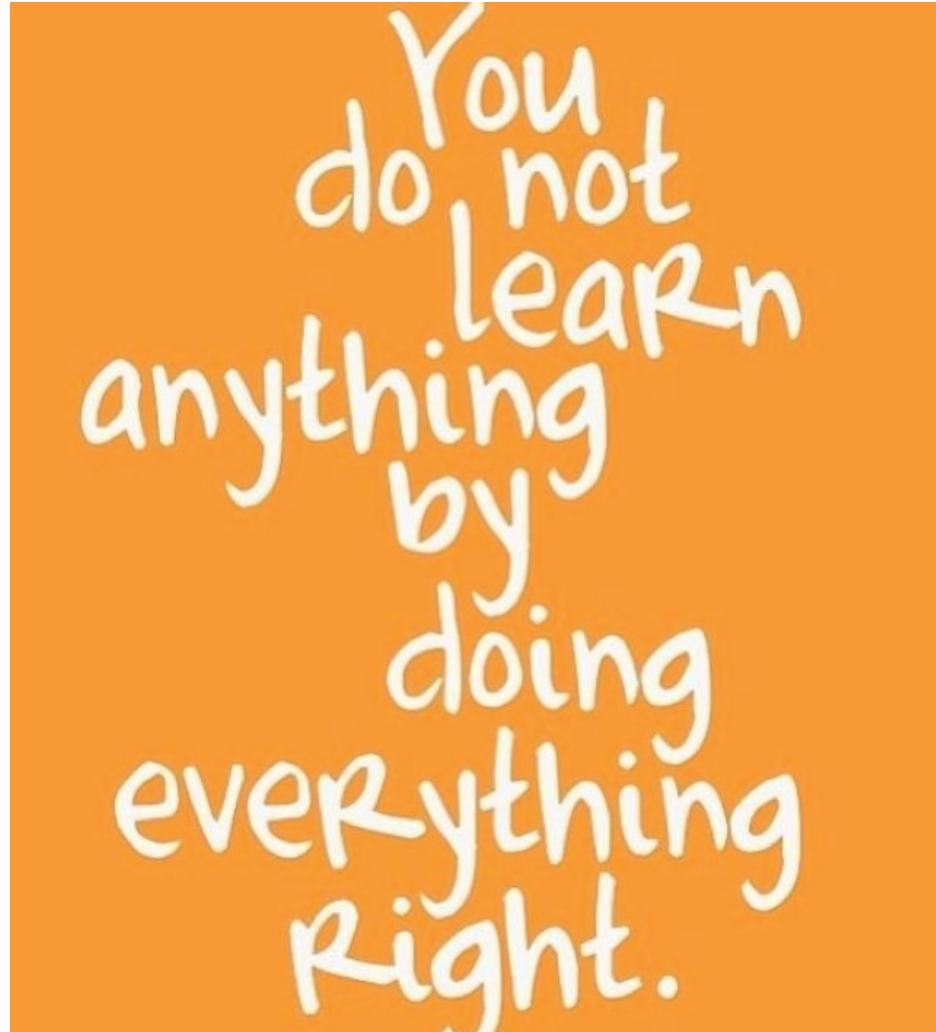
1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Moral quotes for life



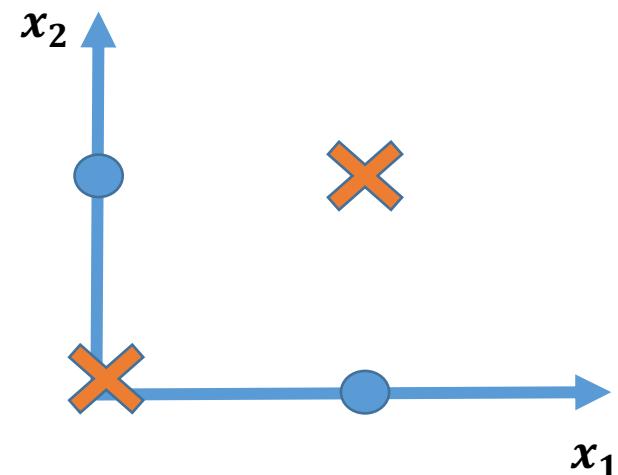
Check point: What you need to know

- ❖ The Perceptron algorithm
- ❖ The geometry of the update

Perceptron Learnability

- ❖ Perceptron cannot learn what it cannot represent
 - ❖ Only linearly separable functions
 - Minsky and Papert (1969)
 - ❖ E.g., Parity functions can't be learned (XOR)

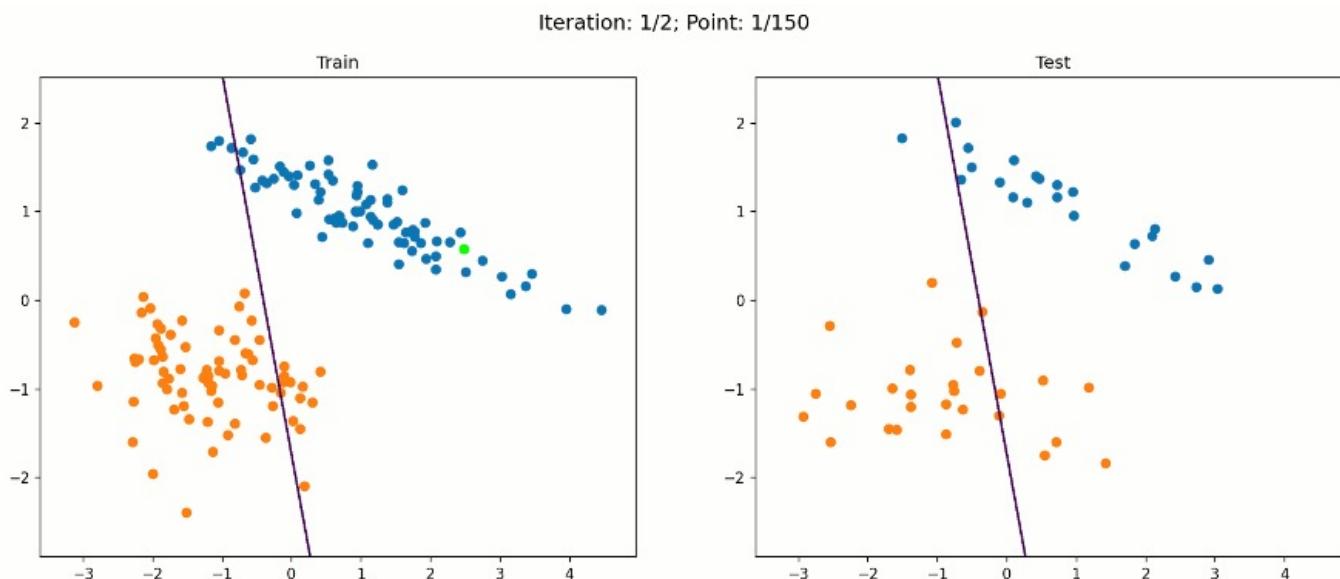
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



Convergence

Convergence theorem

- ❖ If there exist a set of weights that are consistent with the data (i.e. the data is **linearly separable**), the perceptron algorithm will converge.



Convergence

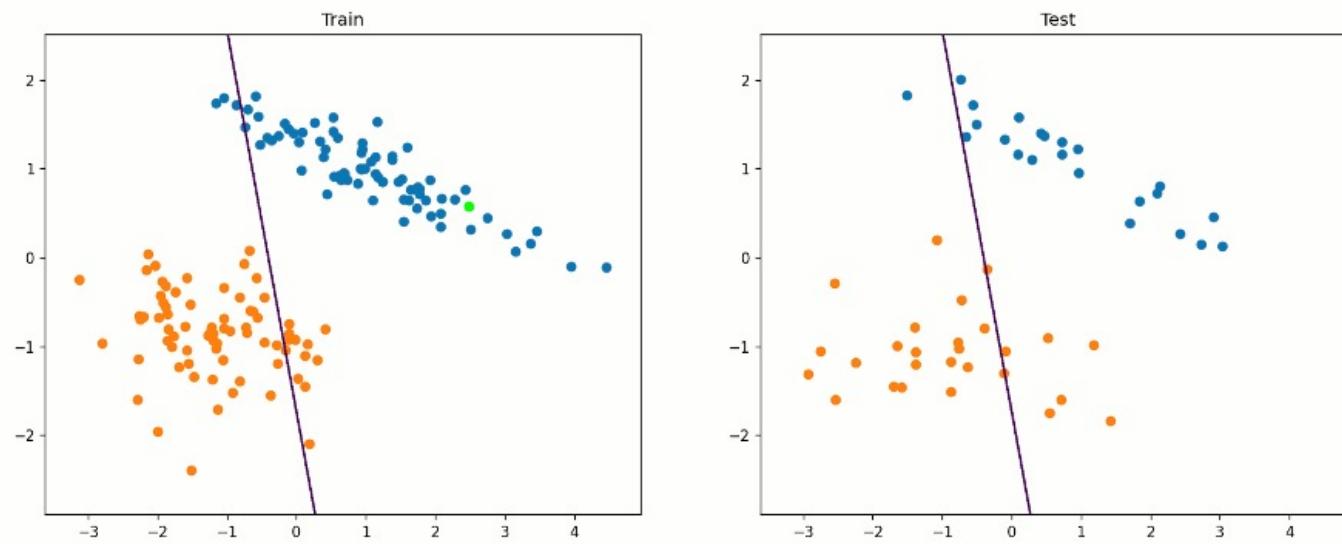
Convergence theorem

- ❖ If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge.

Cycling theorem

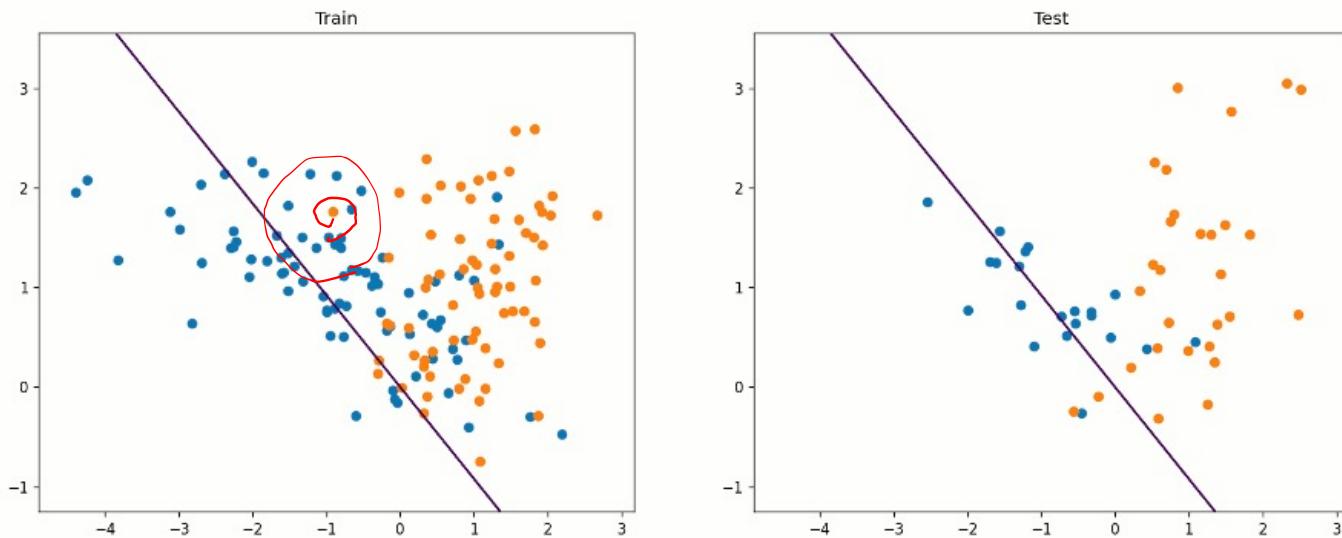
- ❖ If the training data **is not** linearly separable, then the learning algorithm will eventually repeat the same set of weights and **enter an infinite loop**

Iteration: 1/2; Point: 1/150



Linearly Separable

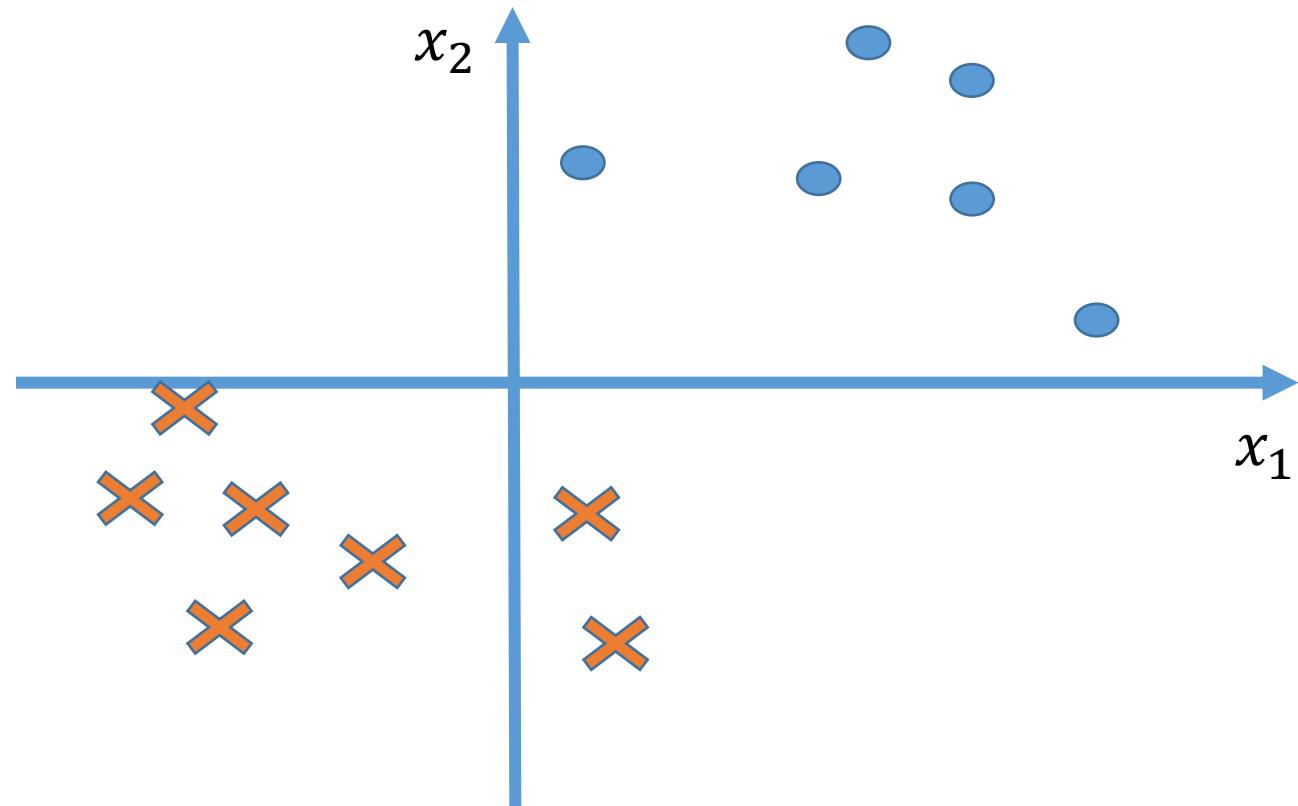
Iteration: 1/100



Lec 6: Perceptron & Logistic
Not Linearly Separable
Regression

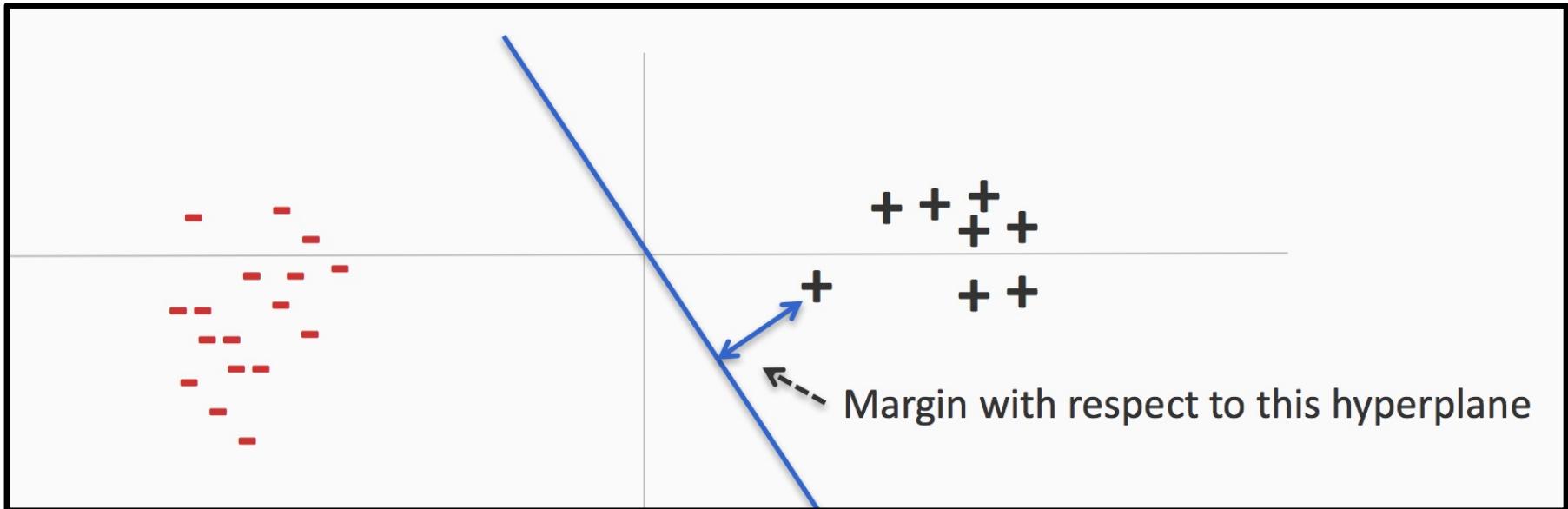
Linear separable

There exists a hyper-plane that can separate the data
(we may not know what the hyperplane is)



Margin

The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

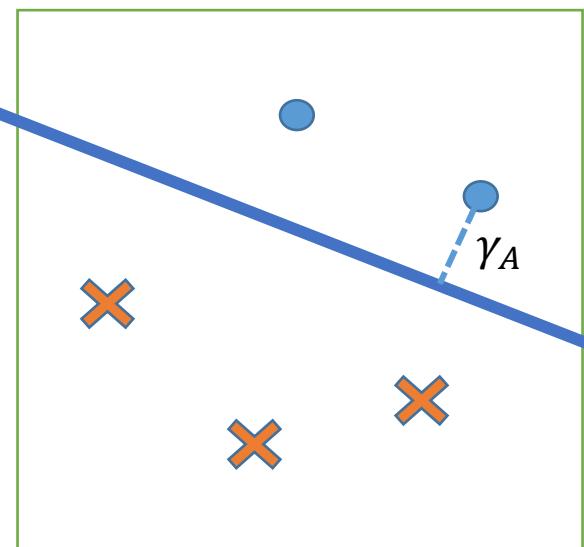


Margin

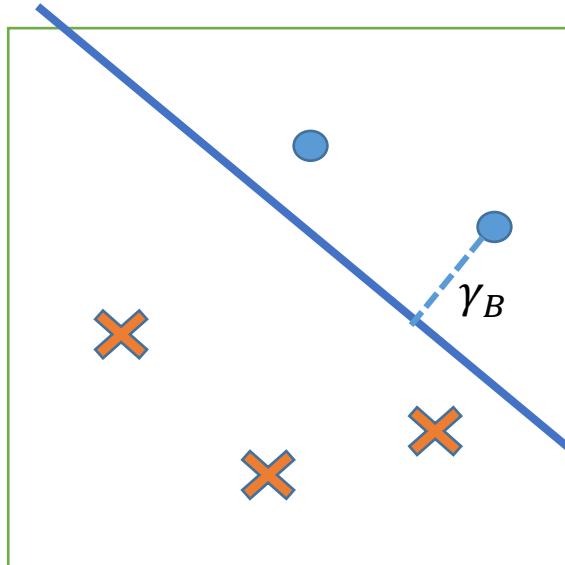
- ❖ The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- ❖ The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector.

Check Point

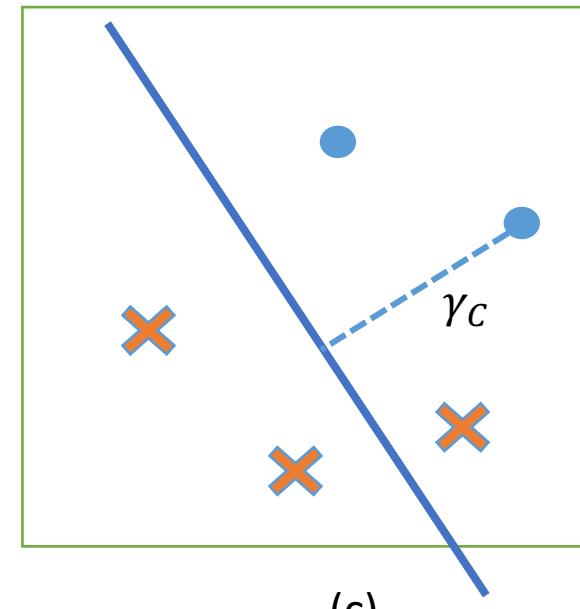
❖ Which γ is the margin of data?



(a)



(b)



(c)

Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ be a sequence of training examples such that for all i , the feature vector $x_i \in R^n$, $\|x_i\| \leq R$ and the label $y_i \in \{-1, +1\}$.

Suppose we have a binary classification dataset with n dimensional inputs.

Suppose there exists a unit vector $u \in R^n$ (i.e $\|u\| = 1$) such that for some $\gamma \in R^n$ and $\gamma > 0$ we have $y_i (u^\top x_i) \geq \gamma$.

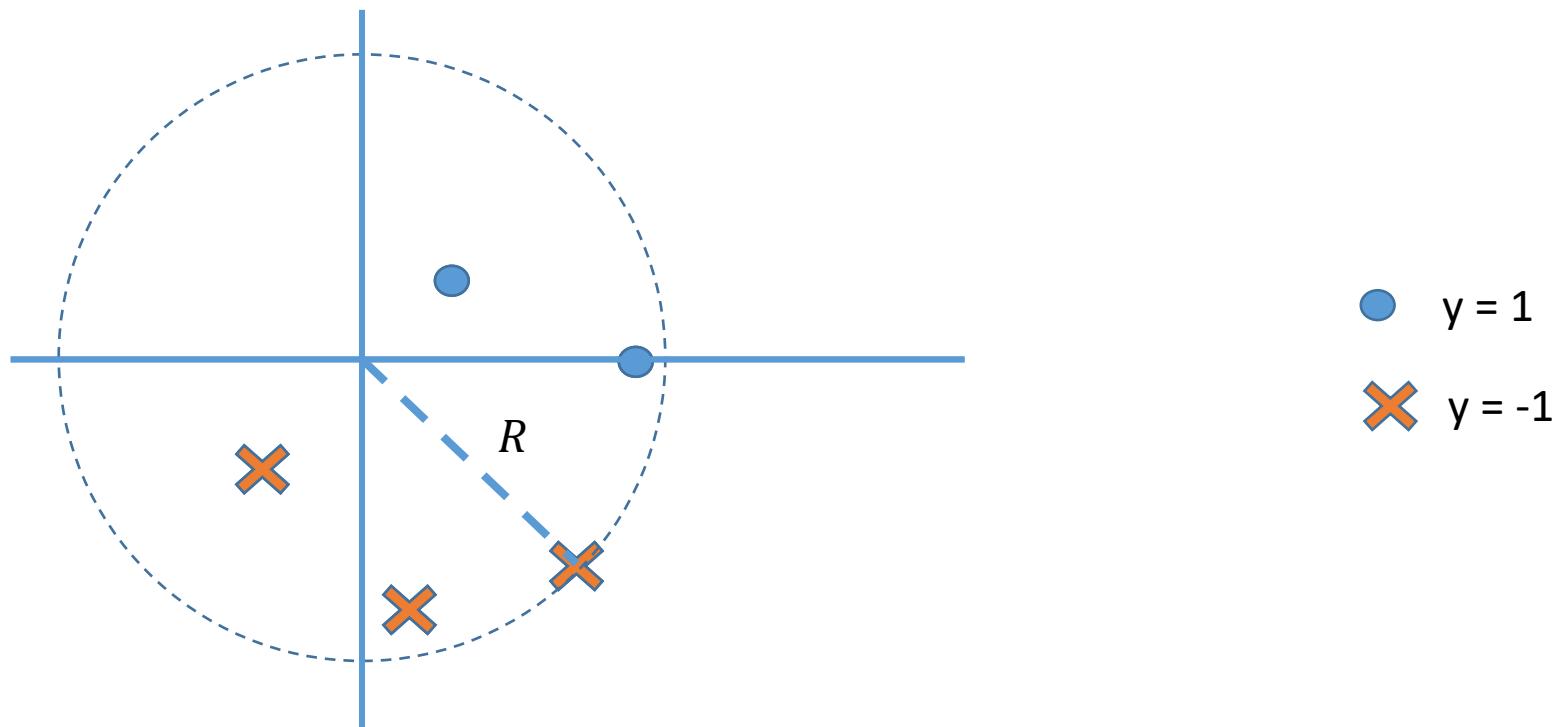
If the data is separable,...

Then, the perceptron algorithm will make at most $(R/\gamma)^2$ mistakes on the training sequence.

...then the Perceptron algorithm will find a separating hyperplane after making a finite number of mistakes

Mistake Bound Theorem [Novikoff 1962, Block 1962]

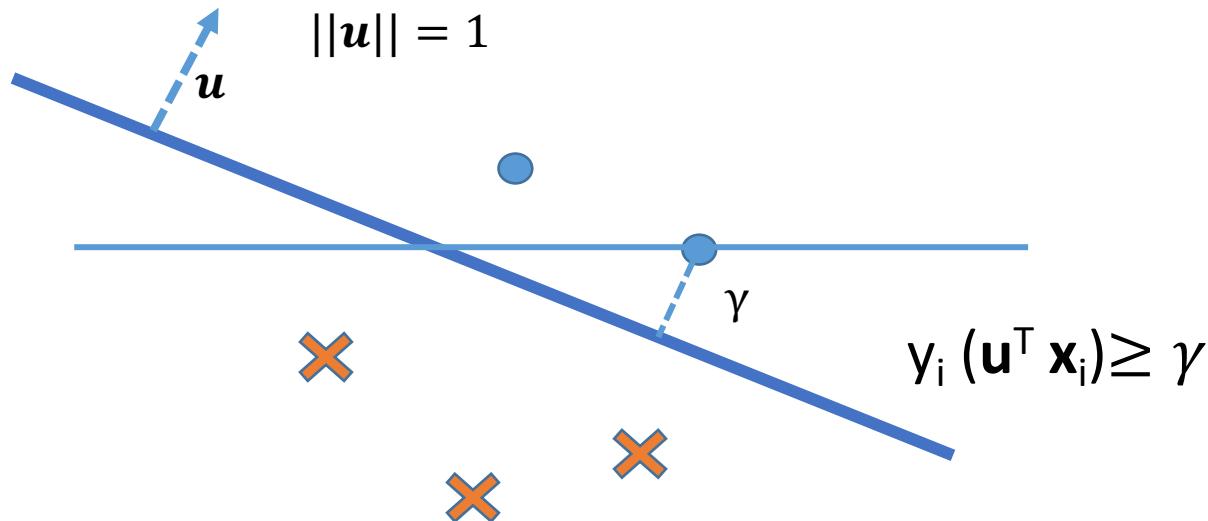
Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ be a sequence of training examples such that for all i , the feature vector $\mathbf{x}_i \in R^n$, $\|\mathbf{x}_i\| \leq R$ and the label $y_i \in \{-1, +1\}$.



Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ be a sequence of training examples such that for all i , the feature vector $\mathbf{x}_i \in R^n$, $\|\mathbf{x}_i\| \leq R$ and the label $y_i \in \{-1, +1\}$.

Suppose there exists a unit vector $\mathbf{u} \in R^n$ (i.e $\|\mathbf{u}\| = 1$) such that for some $\gamma > 0$ we have $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$.



Note that in reality, \mathbf{u} and γ are in the hindsight.

They are the property of the data and we don't know their values

Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ be a sequence of training examples such that for all i , the feature vector $x_i \in R^n$, $\|x_i\| \leq R$ and the label $y_i \in \{-1, +1\}$.

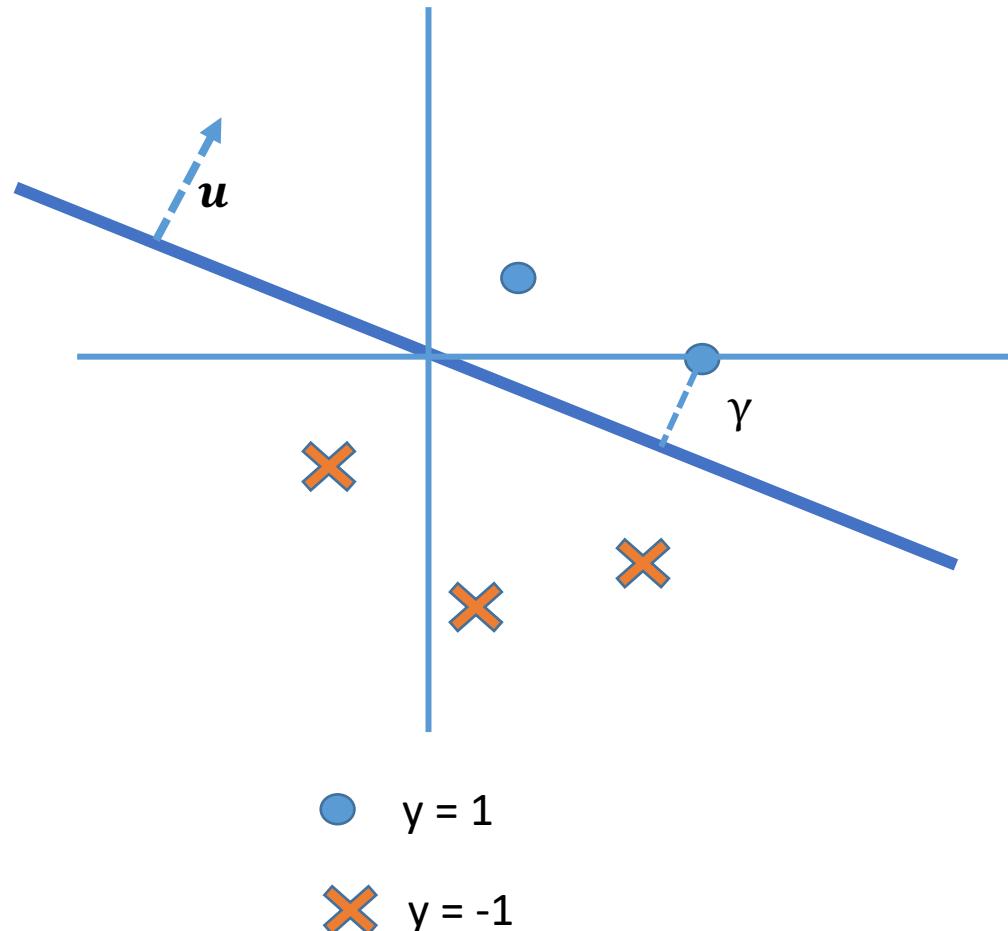
Suppose there exists a unit vector $u \in R^n$ (i.e $\|u\| = 1$) such that for some $\gamma > 0$ we have $y_i (u^\top x_i) \geq \gamma$.

Then, the perceptron algorithm will make at most $(R/\gamma)^2$ mistakes on the training sequence.

Proof will be covered in TA session and will not be in final exam

Intuition

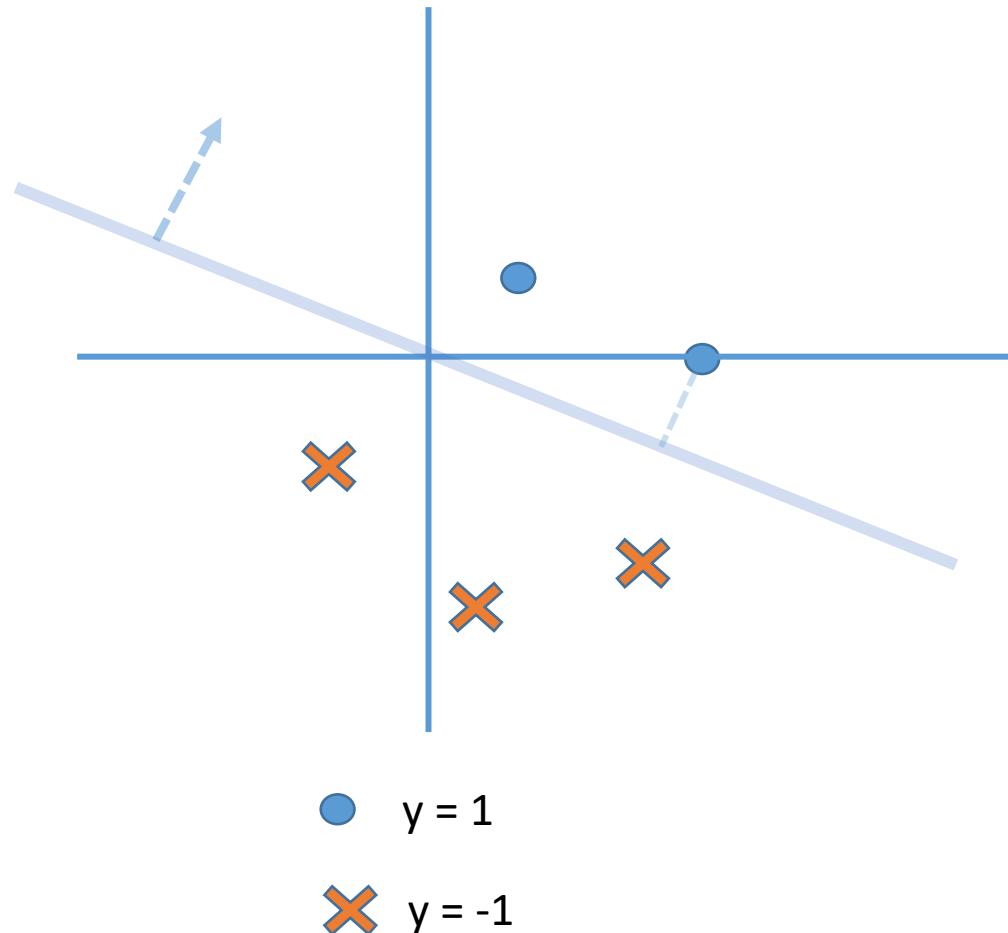
$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



Proof will be covered in TA session and will not be in final exam

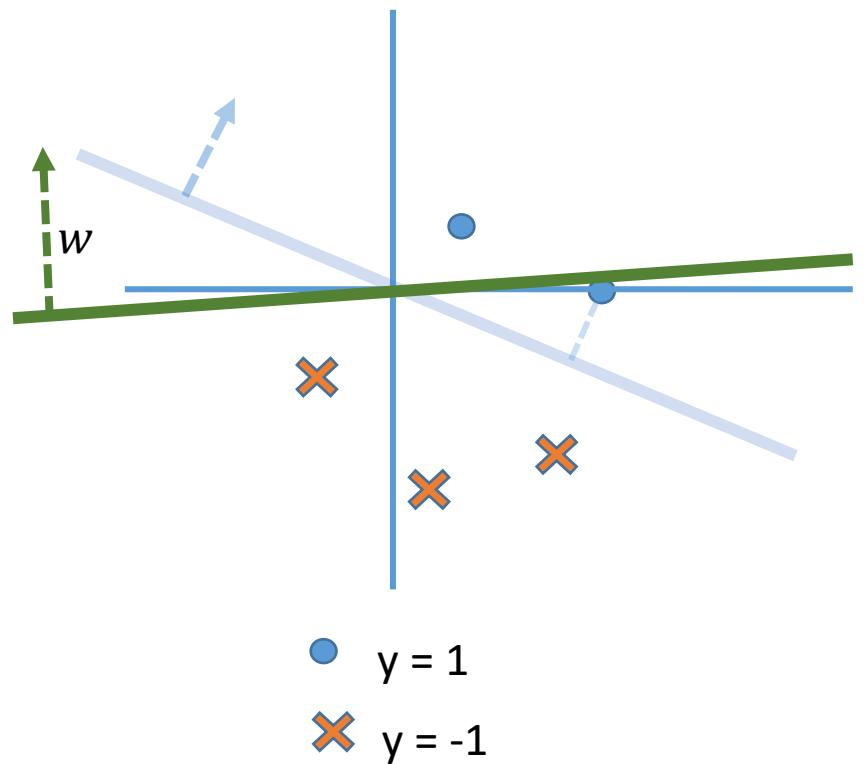
Intuition

$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



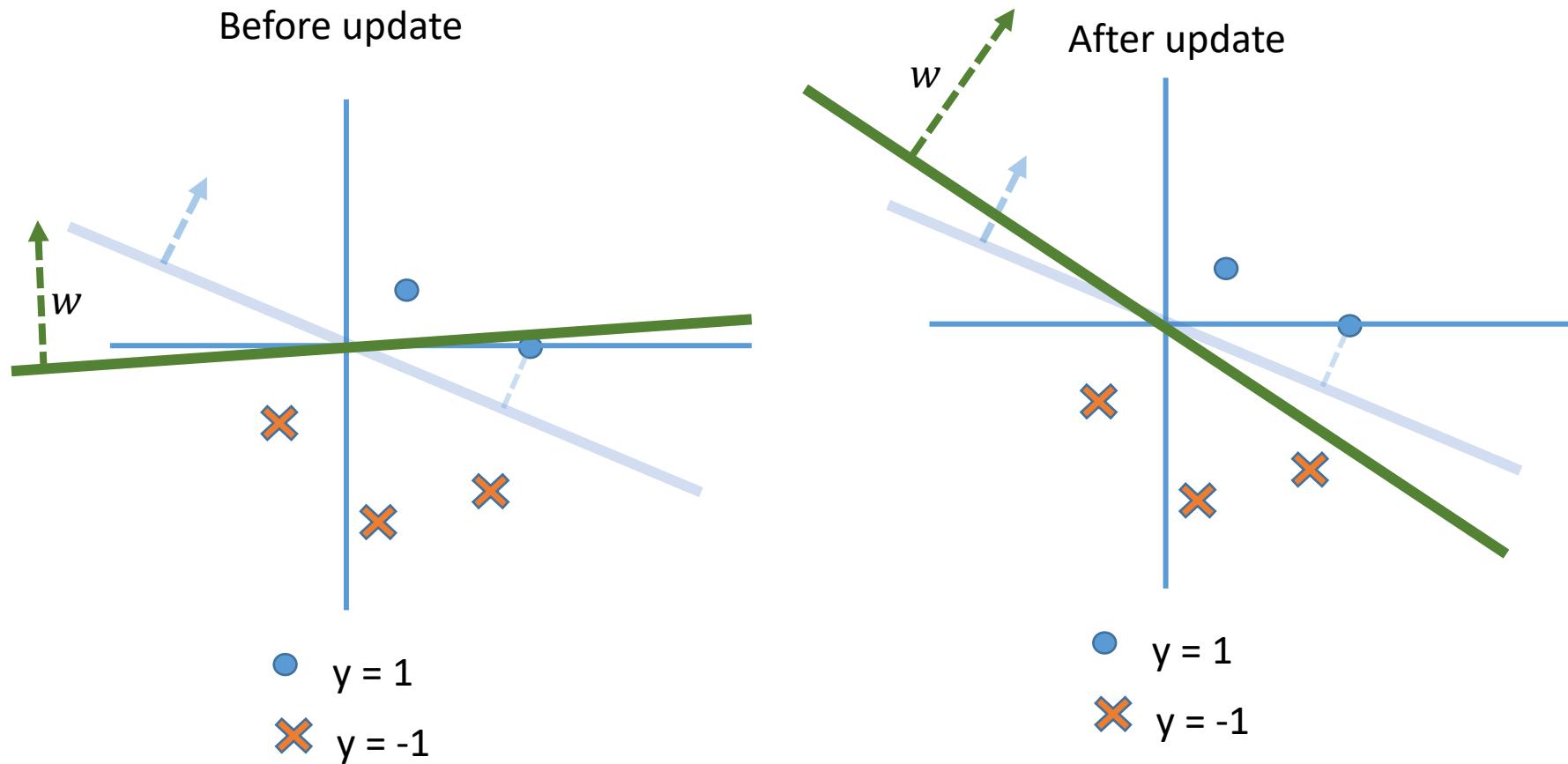
Intuition

$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



Intuition

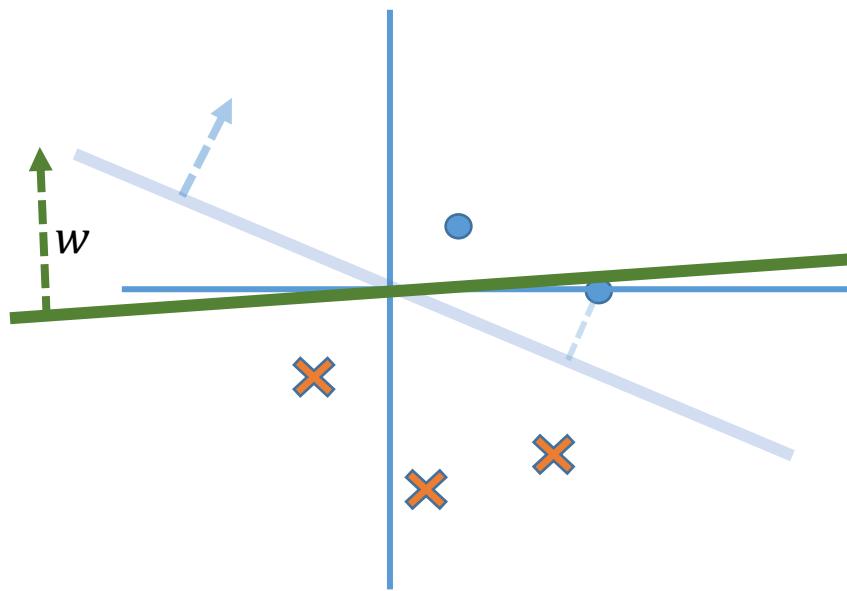
$$\|u\| = 1 \quad y_i (u^\top x_i) \geq \gamma$$



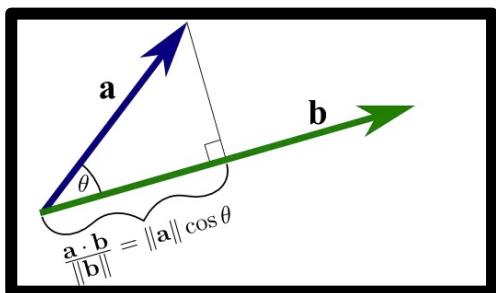
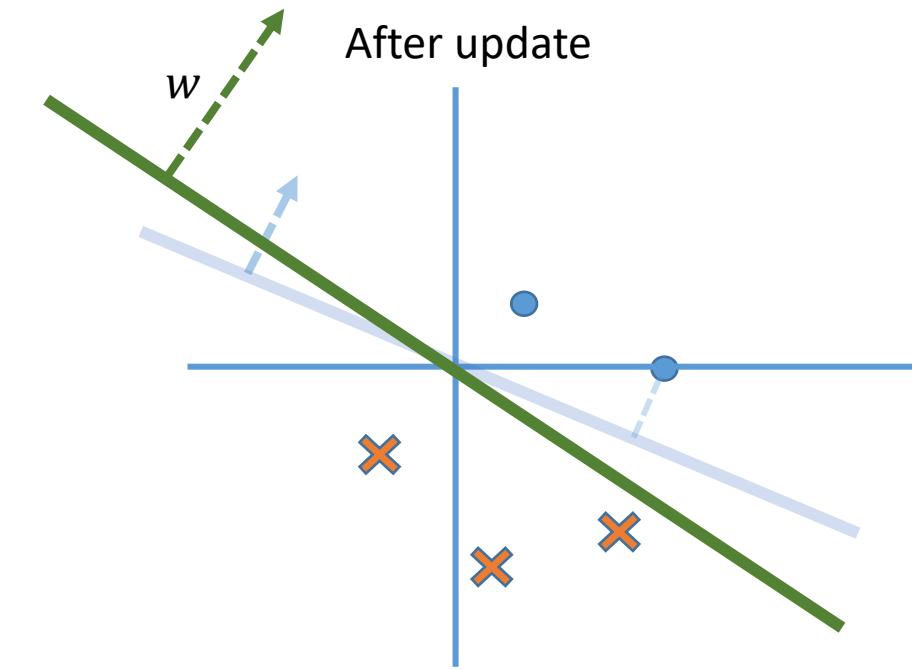
Intuition

$$\|w\| = 1 \quad y_i (w^T x_i) \geq \gamma$$

Before update



After update



Before update

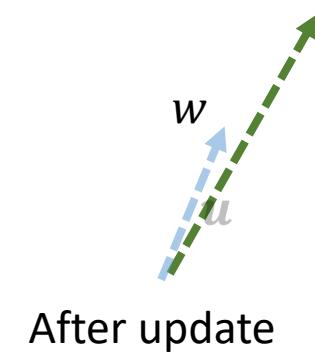
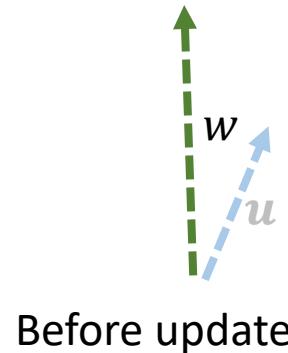
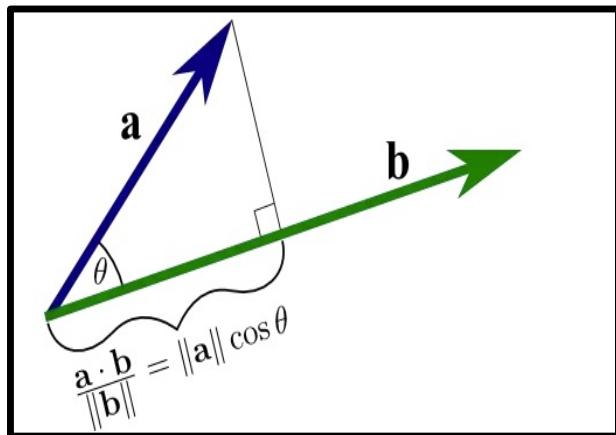


After update



Intuition

1. After update, $\mathbf{u}^T \mathbf{w}_{t+1}$ is larger than $\mathbf{u}^T \mathbf{w}_t$
After t mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t \gamma$
2. The size of $\|\mathbf{w}_{t+1}\|$ may increase, but not much
After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$



Proof (preliminaries)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

The setting

- ❖ Initial weight vector \mathbf{w} is all zeros
- ❖ All training examples are contained in a ball of size R
 - ❖ $\|\mathbf{x}_i\| \leq R$
- ❖ The training data is separable by margin γ using a unit vector \mathbf{u}
 - ❖ $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$

Proof will be covered in TA session and will not be in final exam

Proof (1/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

1. Claim: After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t \gamma$

$$\begin{aligned}\mathbf{u}^T \mathbf{w}_{t+1} &= \mathbf{u}^T \mathbf{w}_t + y_i \mathbf{u}^T \mathbf{x}_i \\ &\geq \mathbf{u}^T \mathbf{w}_t + \gamma\end{aligned}$$

Because the data is
separable by a
margin γ
 $y_i (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^\top \mathbf{w}_0 = 0$), straightforward induction gives us

$$\mathbf{u}^\top \mathbf{w}_t \geq t \gamma$$

Intuition: the inner product between the underlying true model and the current model is non-decreasing after each update
1). The directions of u, w align or 2). $\|\mathbf{w}\|$ increases

Proof will be covered in TA session and will not be in final exam

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^T \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

Proof (2/3)

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^T \mathbf{x}_i) + \|\mathbf{x}_i\|^2\end{aligned}$$

Proof will be covered in TA session and will not be in final exam

Proof (2/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^\top \mathbf{x}_i) + \|\mathbf{x}_i\|^2\end{aligned}$$

The weight is updated only when there is a mistake.
That is when $y_i \mathbf{w}_t^\top \mathbf{x}_i < 0$.

$\|\mathbf{x}_i\| \cdot R$, by definition of R

Proof will be covered in TA session and will not be in final exam

Proof (2/3)

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^T \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^T \mathbf{x}_i) + \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + R^2\end{aligned}$$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^T \mathbf{w}_0 = 0$),
straightforward induction gives us $\|\mathbf{w}_t\|^2 \leq tR^2$

Proof will be covered in TA session and will not be in final exam

- Receive an input (\mathbf{x}_i, y_i)
- if $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_i) \neq y_i$:
Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

Proof (2/3)

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

Intuition: $\|w\|$ is bounded

$$\begin{aligned}\|\mathbf{w}_{t+1}\| &= \|\mathbf{w}_t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2y_i (\mathbf{w}_t^\top \mathbf{x}_i) + \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + R^2\end{aligned}$$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^\top \mathbf{w}_0 = 0$),
straightforward induction gives us $\|\mathbf{w}_t\|^2 \leq tR^2$

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

Intuition 1: the inner product between the underlying true model and the current model is non-decreasing after each update
1). The directions of u, w align or 2). $\|w\|$ increases

Intuition 2: $\|w\|$ is bounded

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\|$$

From (2)

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t$$

From (2)



$$\mathbf{u}^T \mathbf{w}_t = \|\mathbf{u}\| \|\mathbf{w}_t\| \cos(\text{angle between them})$$

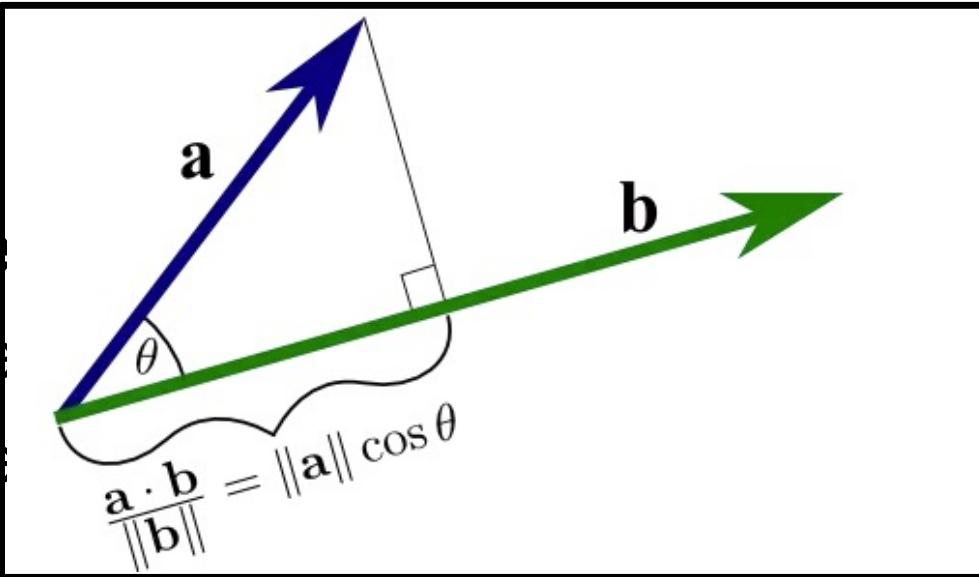
But $\|\mathbf{u}\| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq \|\mathbf{w}_t\|$ (Cauchy-Schwarz inequality)
Lec 9: Perceptron & Logistic Regression

Proof (

What we

1. After
2. After



$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t$$

From (2)



$$\mathbf{u}^T \mathbf{w}_t = \|\mathbf{u}\| \|\mathbf{w}_t\| \cos(\text{angle between them})$$

But $\|\mathbf{u}\| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq \|\mathbf{w}_t\|$ (Cauchy-Schwarz inequality)
Lec 9: Perceptron & Logistic Regression

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^\top \mathbf{w}_t \geq t\gamma$$

From (2)

From (1)

Proof (3/3)

mistakes != # seen data points

What we know:

1. After t mistakes, $\mathbf{u}^T \mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

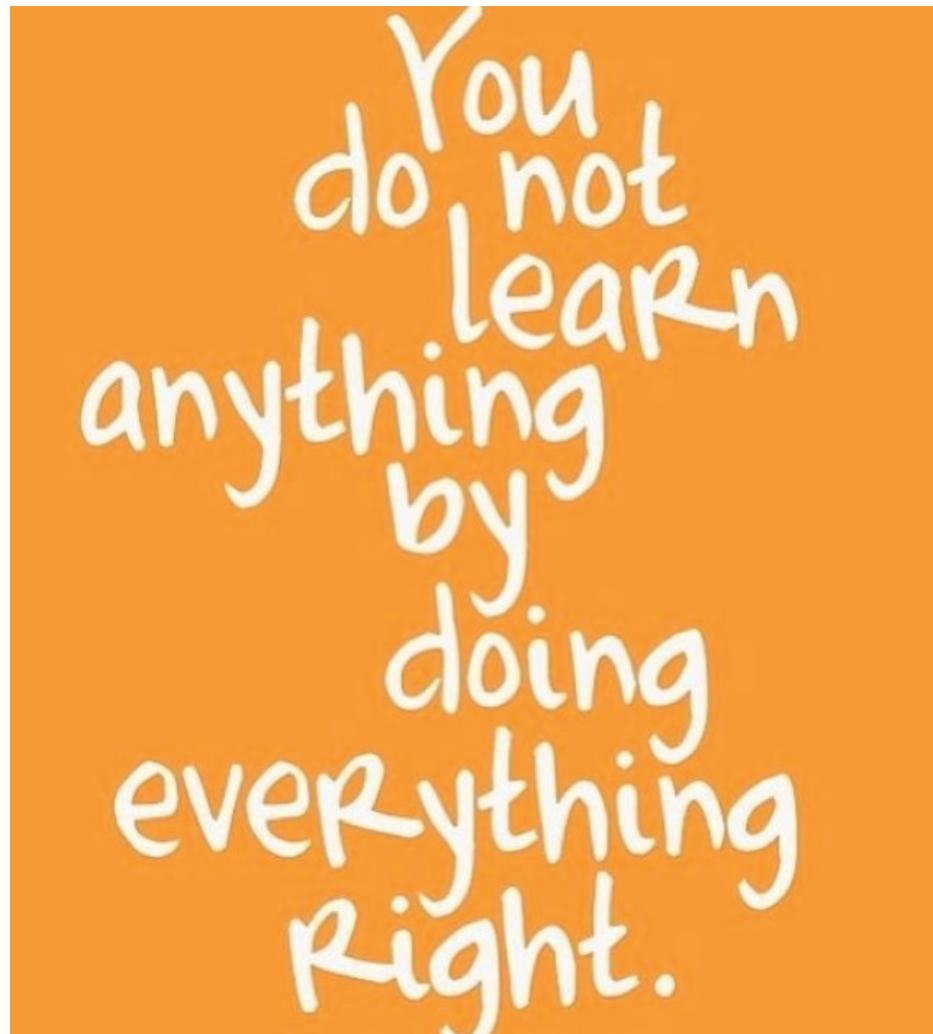


$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t \geq t\gamma$$

Number of mistakes $t \leq \frac{R^2}{\gamma^2}$

Bounds the total number of mistakes!

NOTE!! # mistakes!= # seen data points



Beyond the separable case

- ❖ Good news
 - ❖ Perceptron makes no assumption about data distribution, could be **even adversarial**
 - ❖ After a fixed number of **mistakes**, you are done.
Don't even need to see any more data
- ❖ Bad news:
 - ❖ Real world is often **not** linearly separable
 - ❖ Can't expect to *never* make mistakes again

What you will learn today

- ❖ Linear models
- ❖ The Perceptron Algorithm
- ❖ Perceptron Mistake Bound
- ❖ Variants of Perceptron
- ❖ Online v.s. Batch learning

Practical use of the Perceptron algorithm

1. Using the Perceptron algorithm with a finite dataset
2. Margin Perceptron

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$

2. For (x, y) in \mathcal{D} :

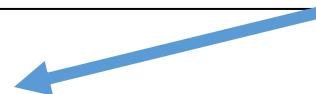
3. if $y(w^\top x) \leq 0$

4. $w \leftarrow w + yx$

5.

6. Return w

Update only on error. A
mistake-driven algorithm



Mistake on positive: $w_{t+1} \leftarrow w_t + x_i$
Mistake on negative: $w_{t+1} \leftarrow w_t - x_i$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

The Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
 For (x, y) in \mathcal{D} :
 if $y(w^T x) \leq 0$
 $w \leftarrow w + \eta yx$
3. Return w



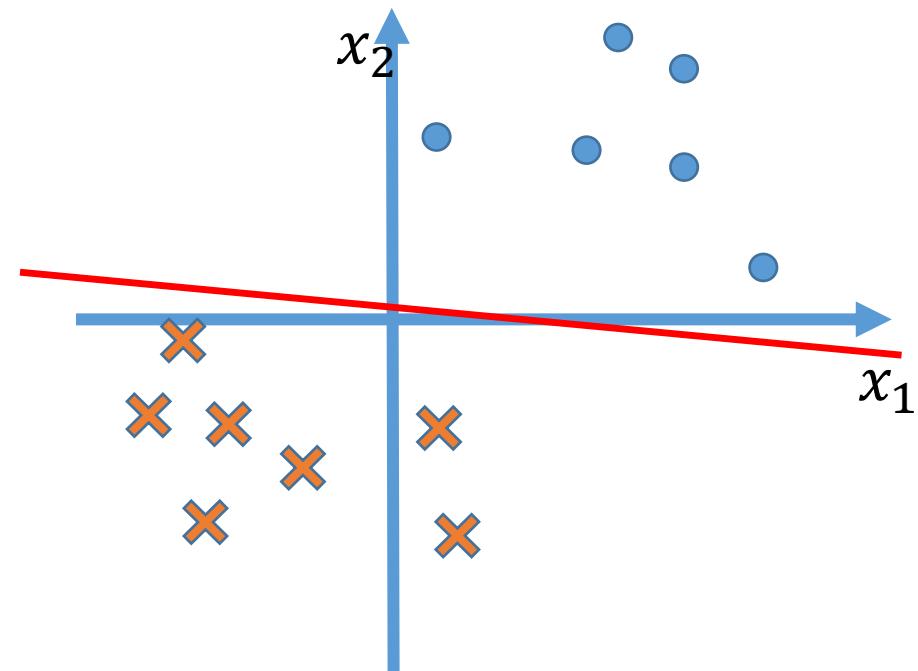
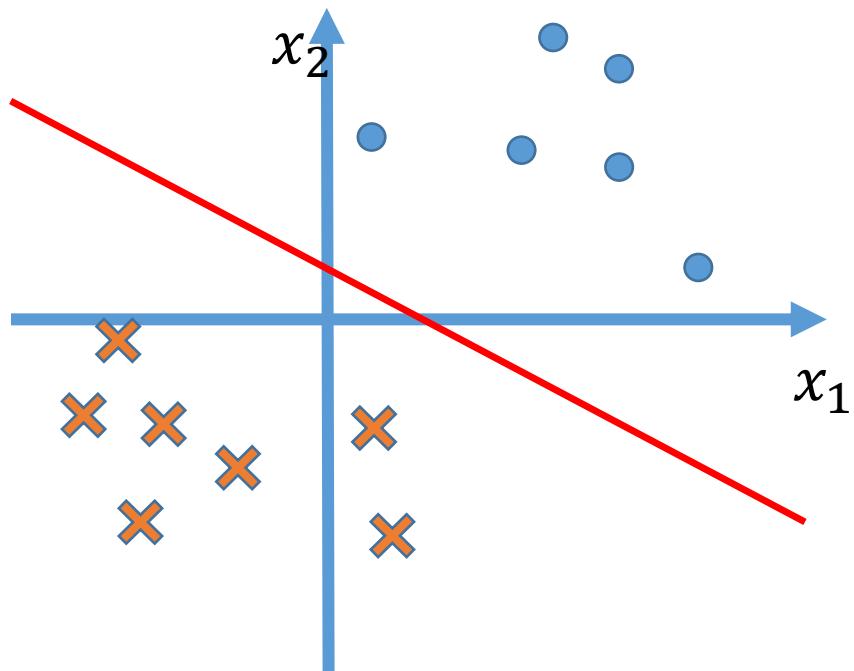
η is a hyper-parameter to the algorithm

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Marginal Perceptron

- ❖ Which one is better



Marginal Perceptron

- ❖ Perceptron makes updates only when the prediction is incorrect

$$y_i \mathbf{w}^T \mathbf{x}_i \leq 0$$

- ❖ What if the prediction is close to being incorrect?
That is, Pick a small positive ϵ and update when

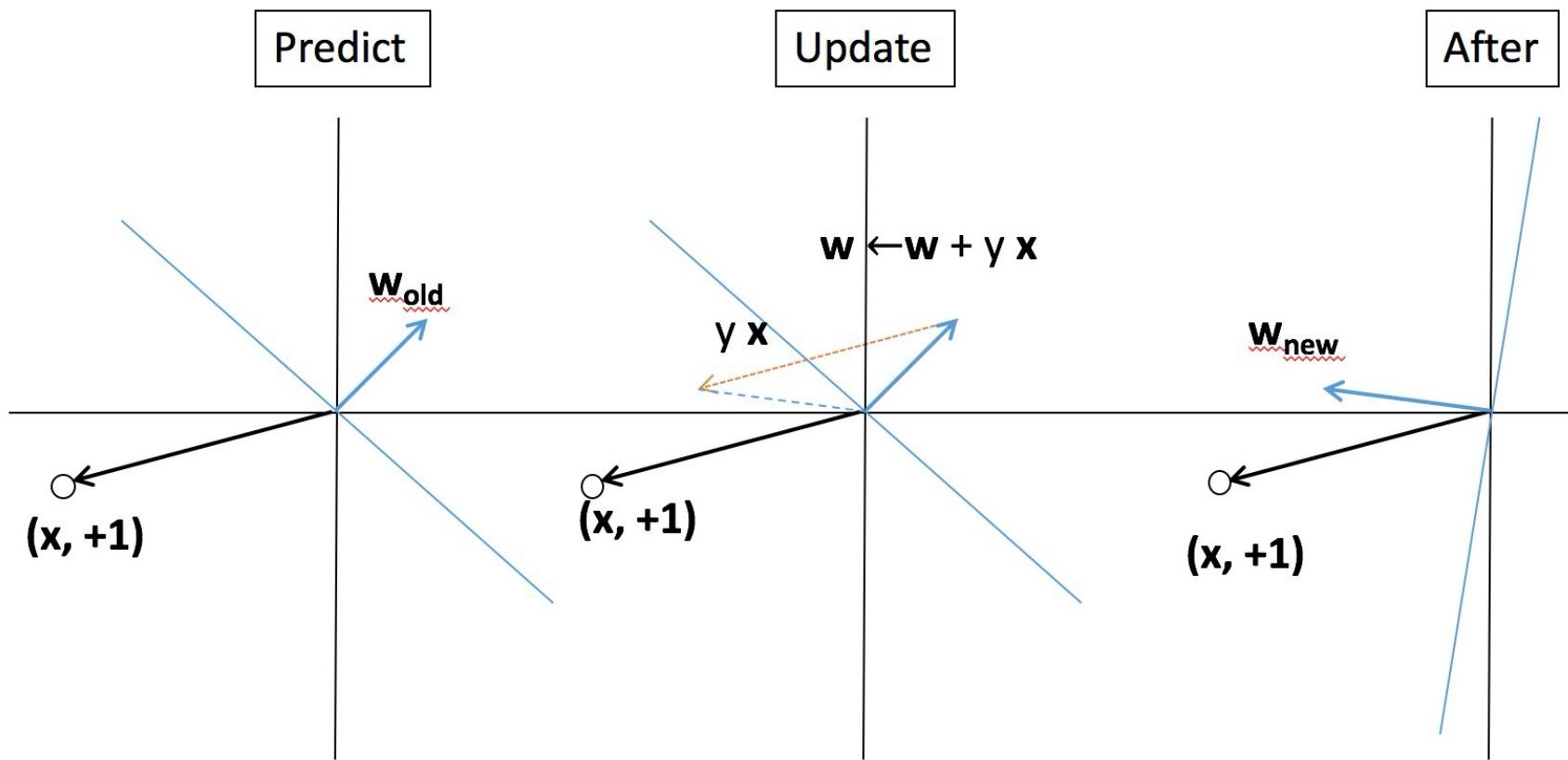
$$y_i \mathbf{w}^T \mathbf{x}_i \leq \epsilon$$

- ❖ Can generalize better, but need to choose ϵ
- ❖ **Exercise:** Why is this a good idea?

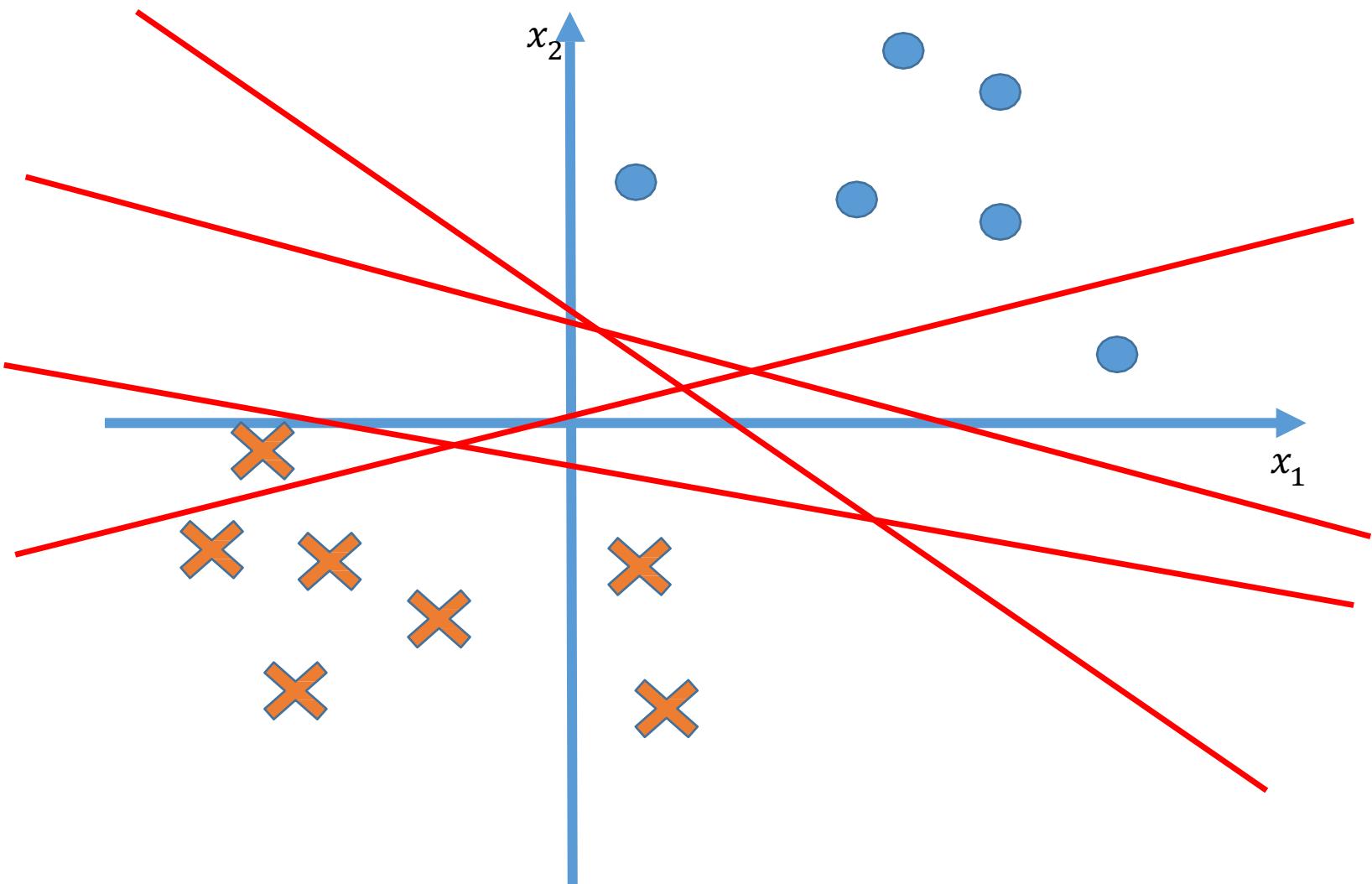
Logistic regression

Recap: Perceptron

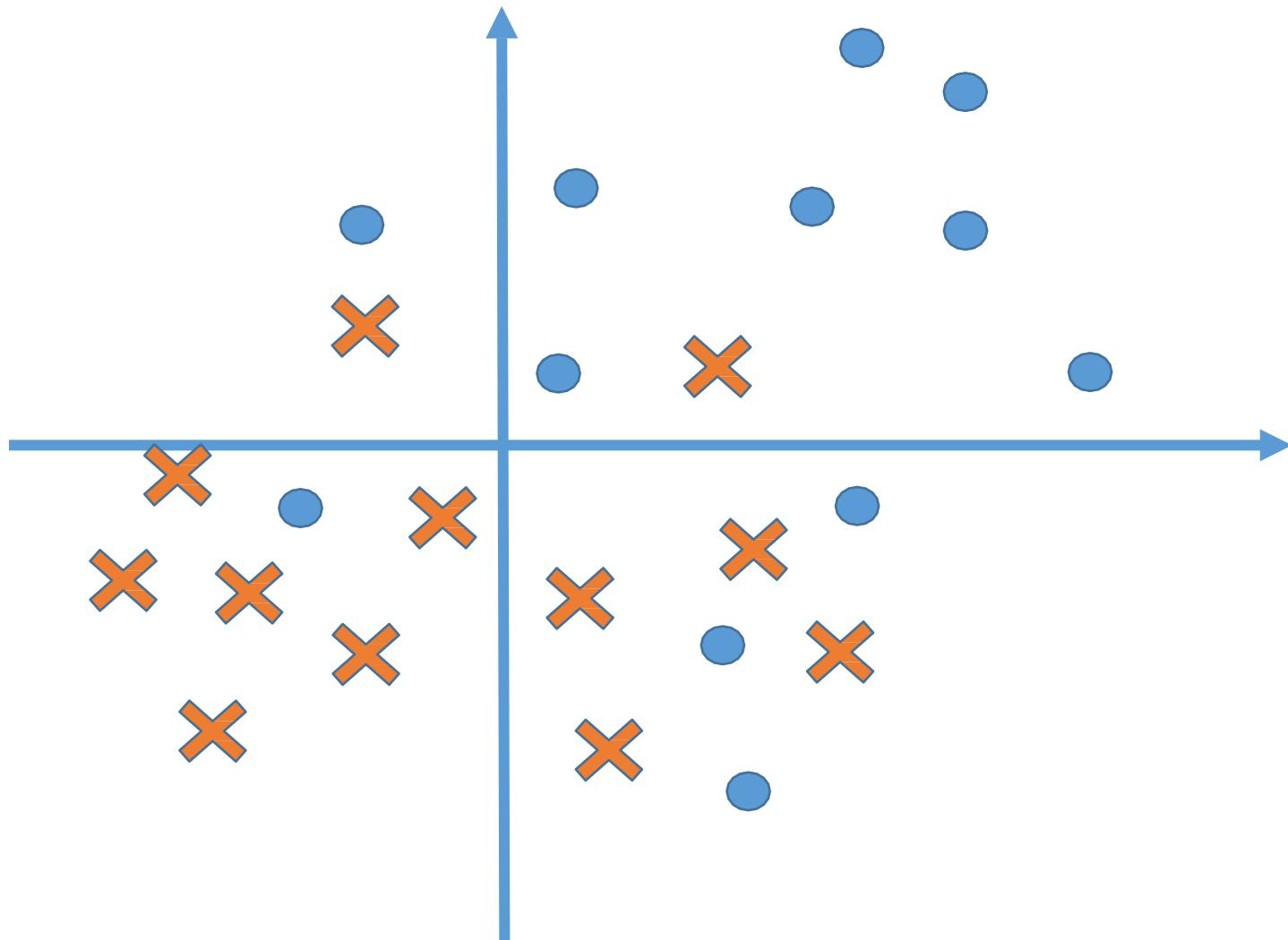
Learning a linear separator when data is linearly separable.



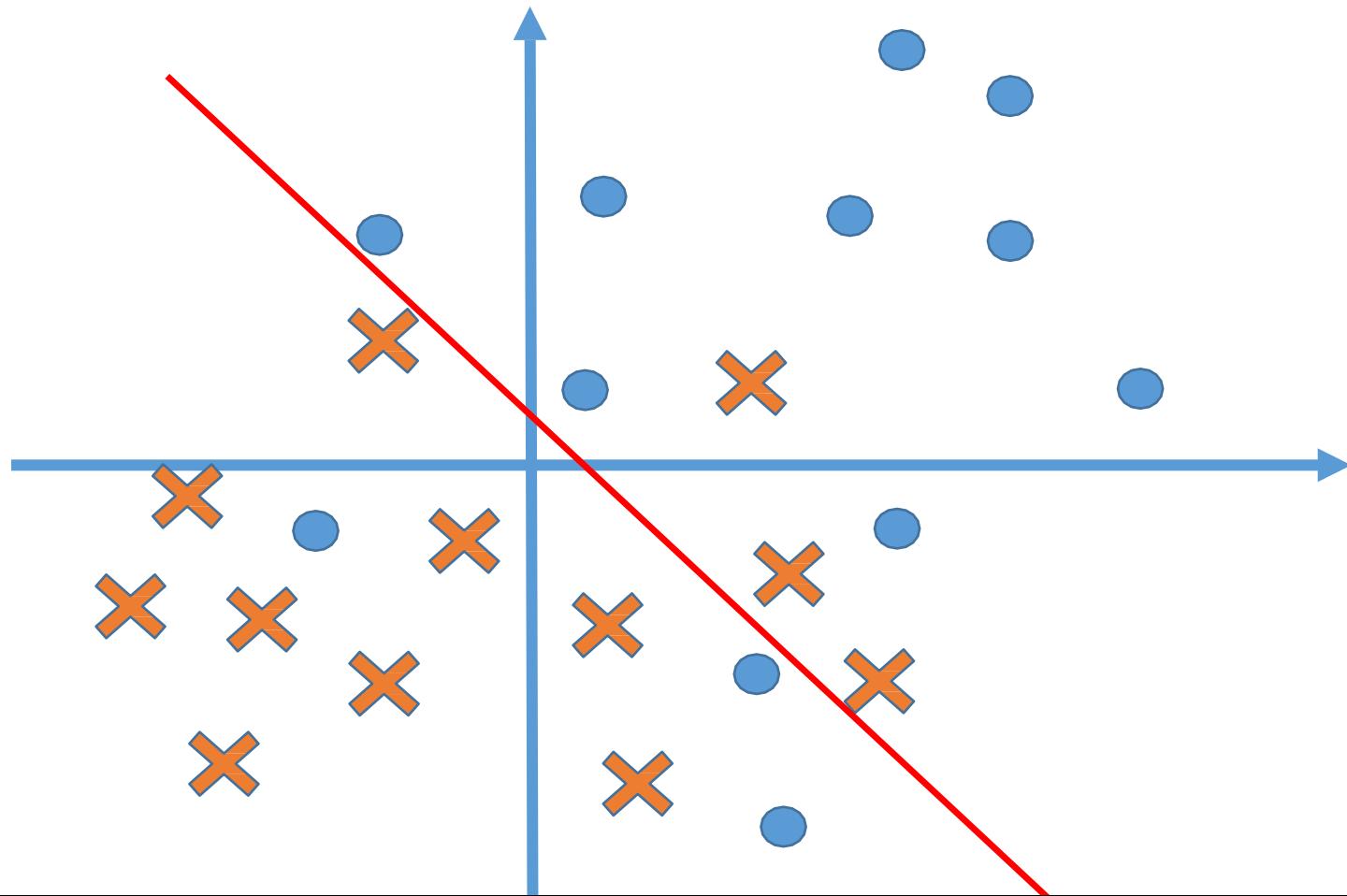
Hypothesis space: linear model



What if data is not linearly separable?

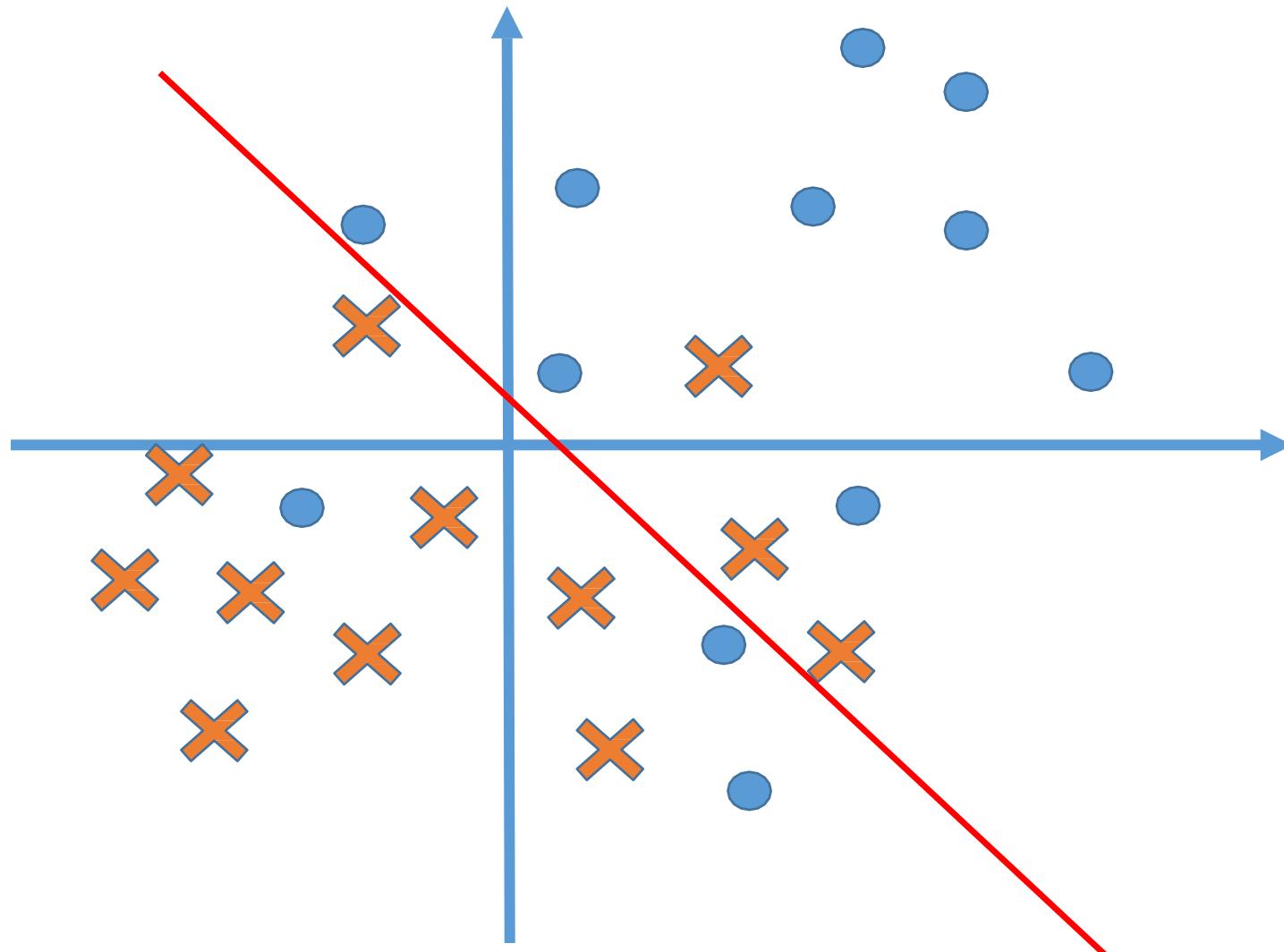


What if dataset is not linearly separable?

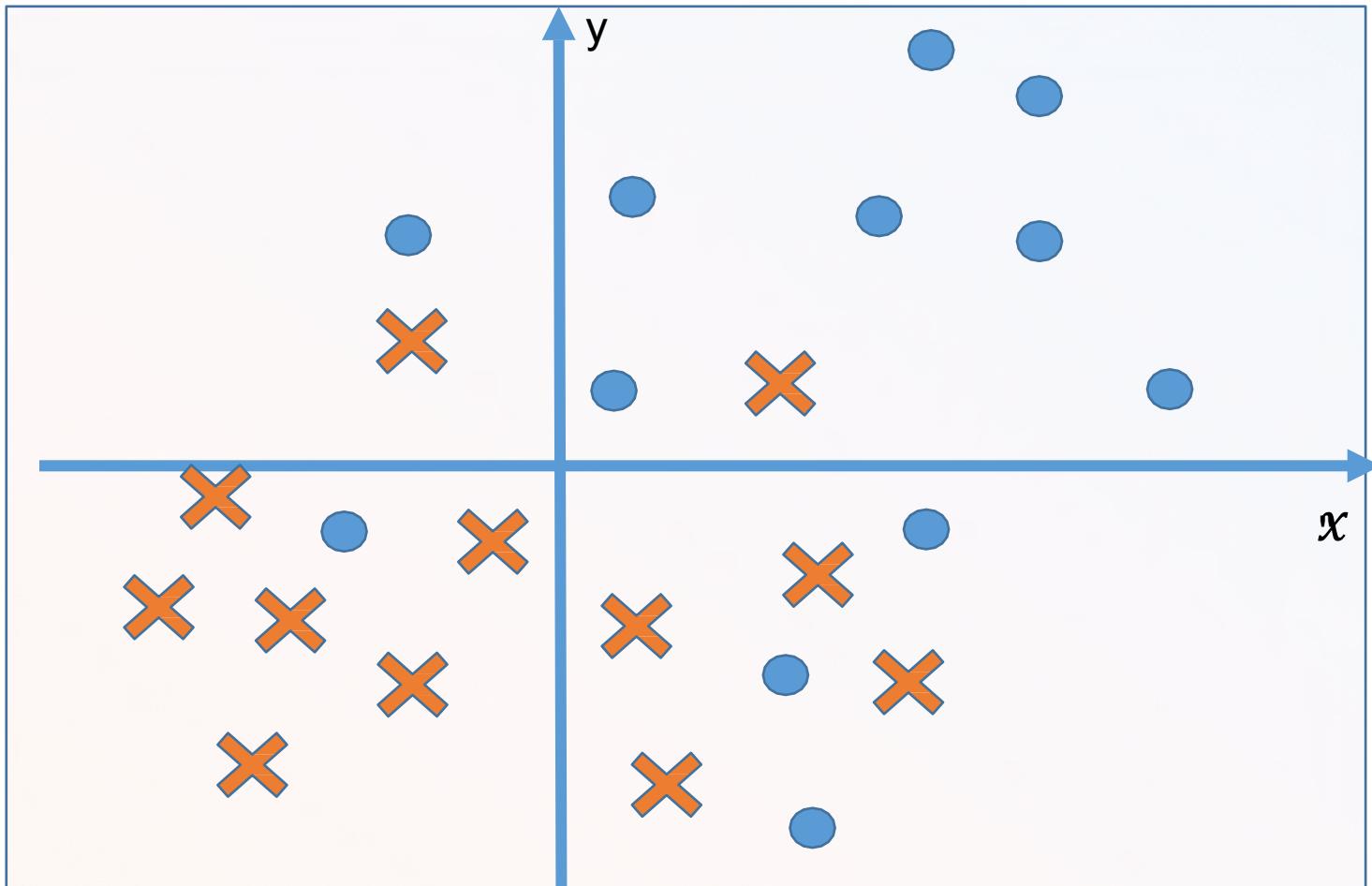


There is no linear model can separate all the data well

What Making data not linearly separable? [Discussion]



What if data is not linearly separable?

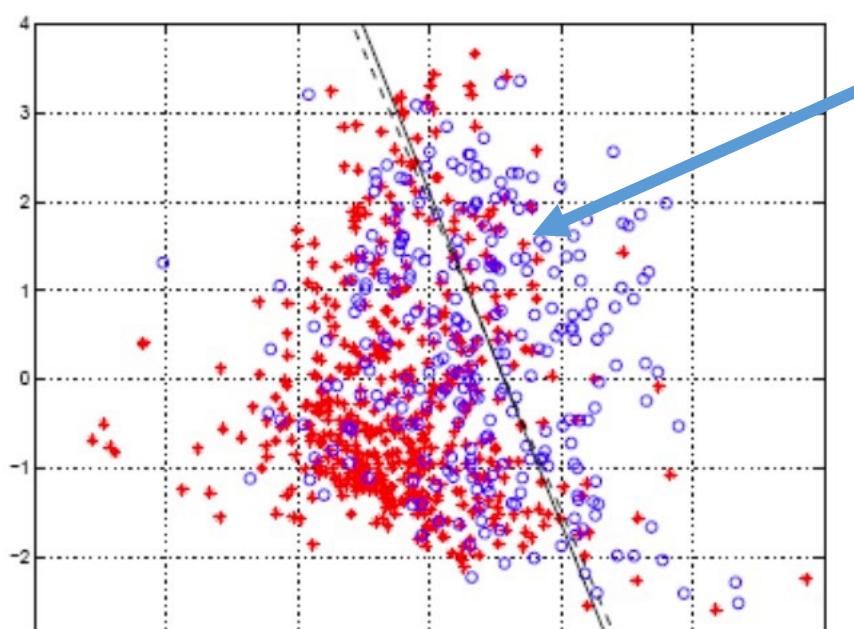


We can consider a probabilistic model (today's lecture)

Modeling the Probability

Classification, but...

- ❖ The output y is discrete valued
- ❖ Instead of predicting the output label,
let us predict $P(y = 1 | \mathbf{x})$

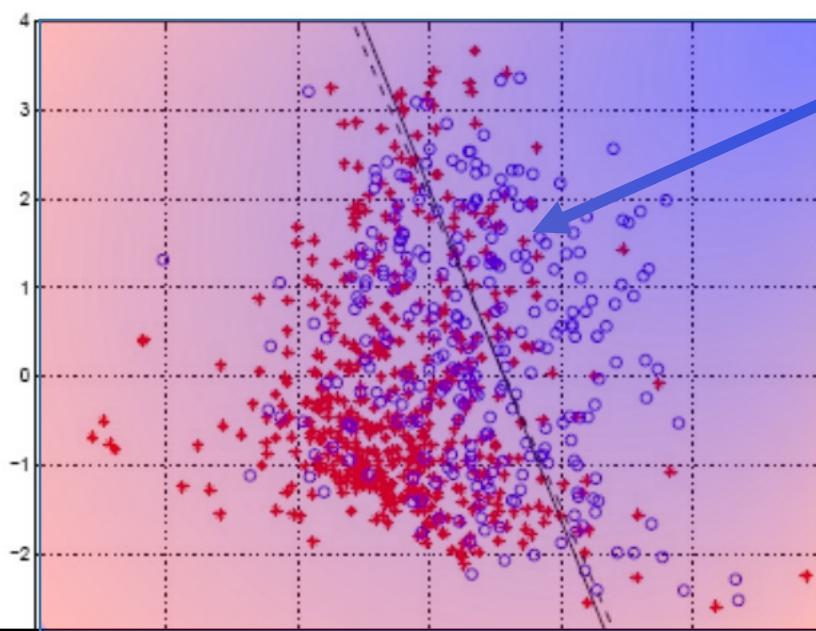


How likely the label $y = 1$
if my feature vector is
in this region

Perceptron does not produce probability estimates

Classification, but...

- ❖ The output y is discrete valued
- ❖ Instead of predicting the output label, let us predict $P(y = 1 | \mathbf{x})$



How likely the label $y = 1$
if my feature vector is
in this region

Perceptron does not produce probability estimates

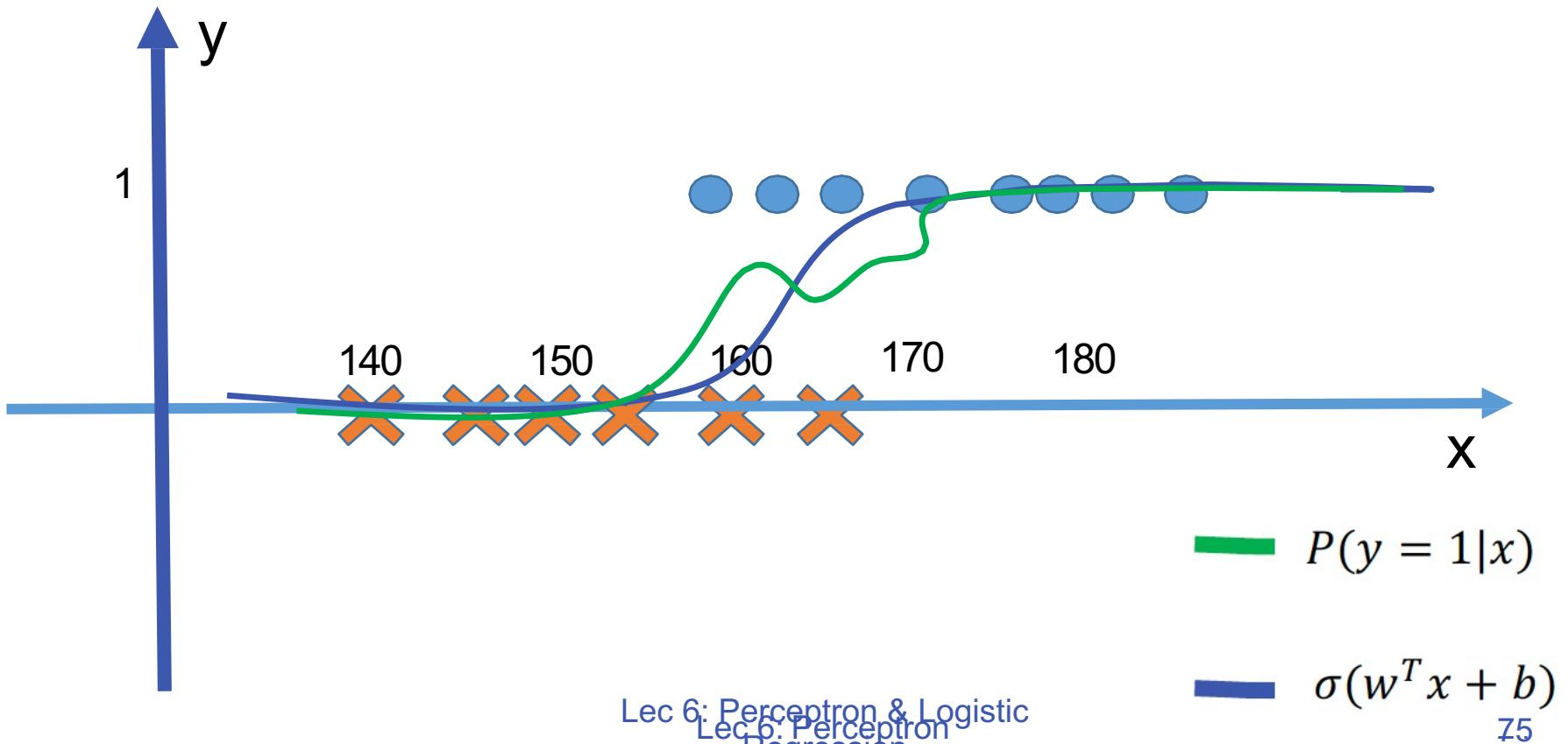
Predict $P(y = 1 | \mathbf{x})$

- Expand hypothesis space to functions whose output is [0-1]
- Original problem: $y \in \{+, -\}$
- Build a model $h(x)$ such that
$$h(x) = \sigma(w^T x + b) \approx P(y = 1|x)$$
- Effectively make the problem a regression problem

Define a model to fit the curve

Define a transformation function such that

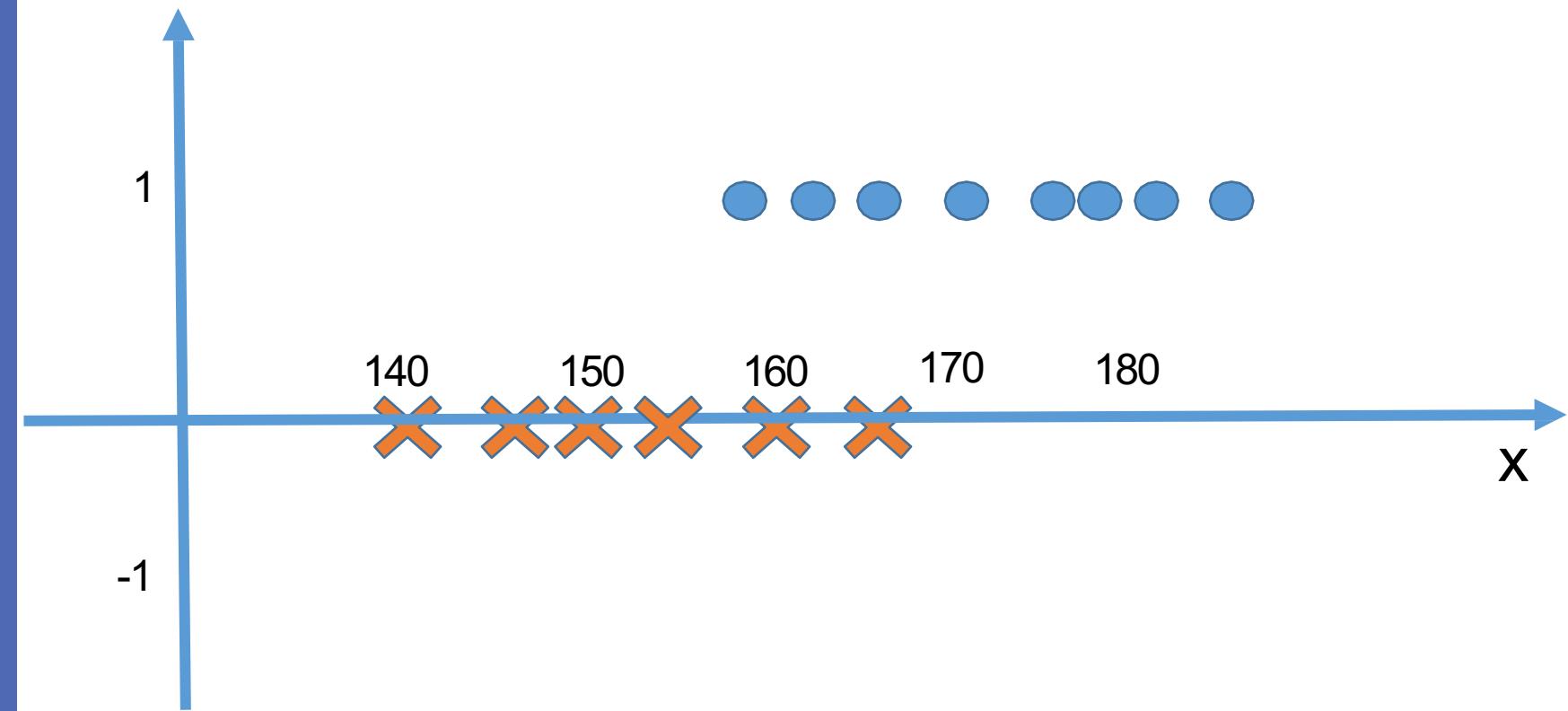
$$\sigma(w^T x + b) \approx P(y = 1|x)$$



— $P(y = 1|x)$

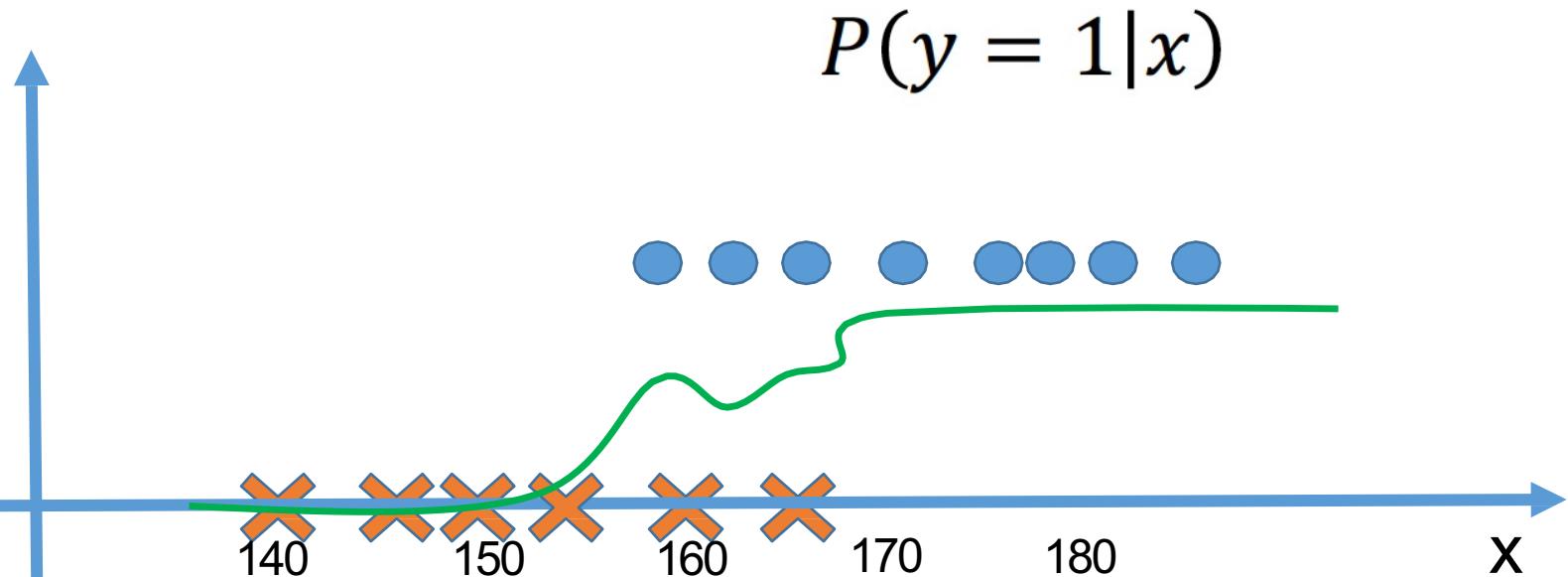
— $\sigma(w^T x + b)$

What is the σ function



What is the σ function

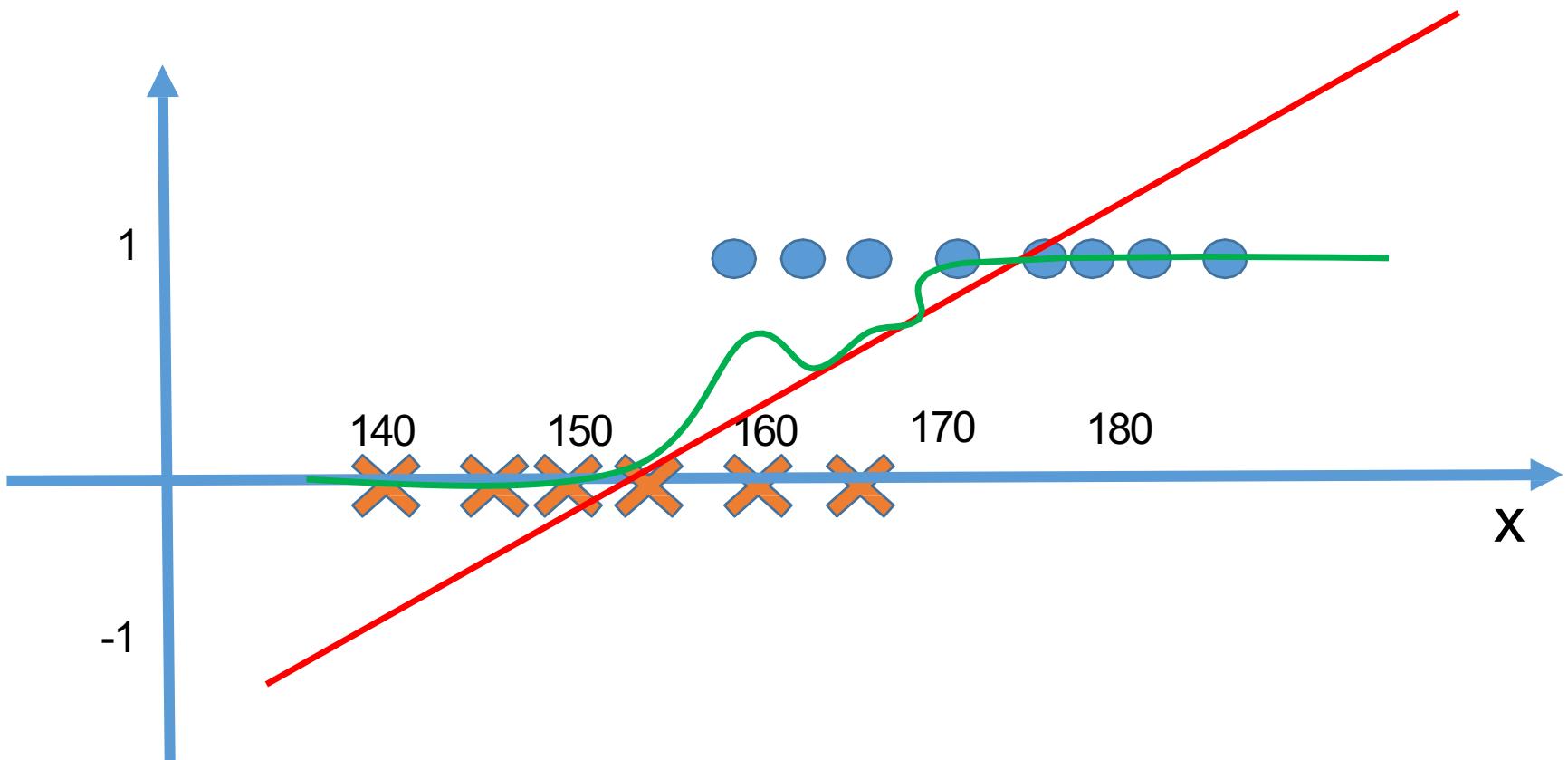
What is the underlying target function?



How to fit $P(y = 1|x)$

Can we fit it with a linear function?

$$y = \mathbf{w}^T \mathbf{x} + b$$



How to fit $P(y = 1|x)$

Probability cannot be larger than 1

1

140

150

160

170

180

x

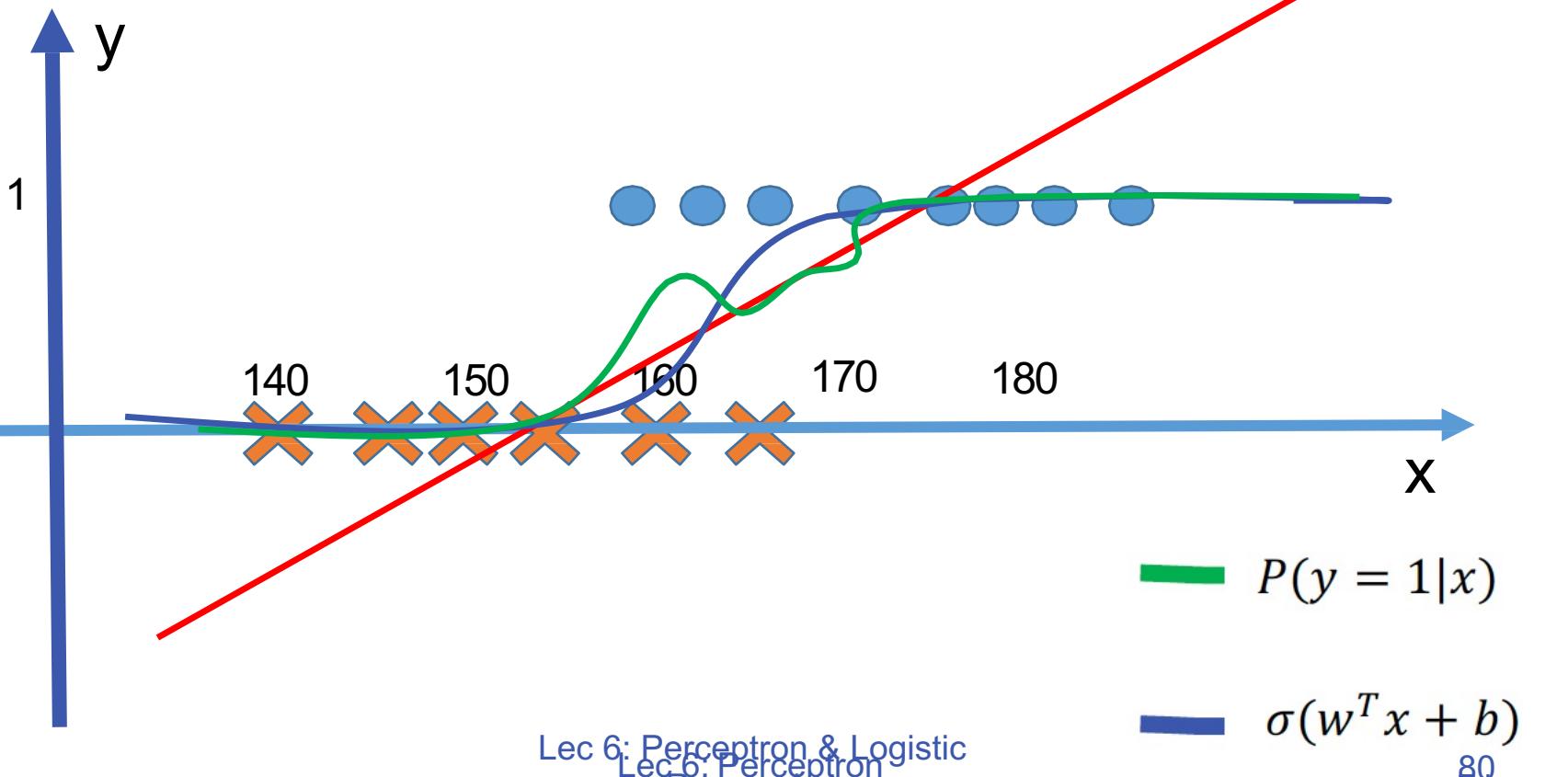
-1

Probability cannot be negative

Transform the linear function

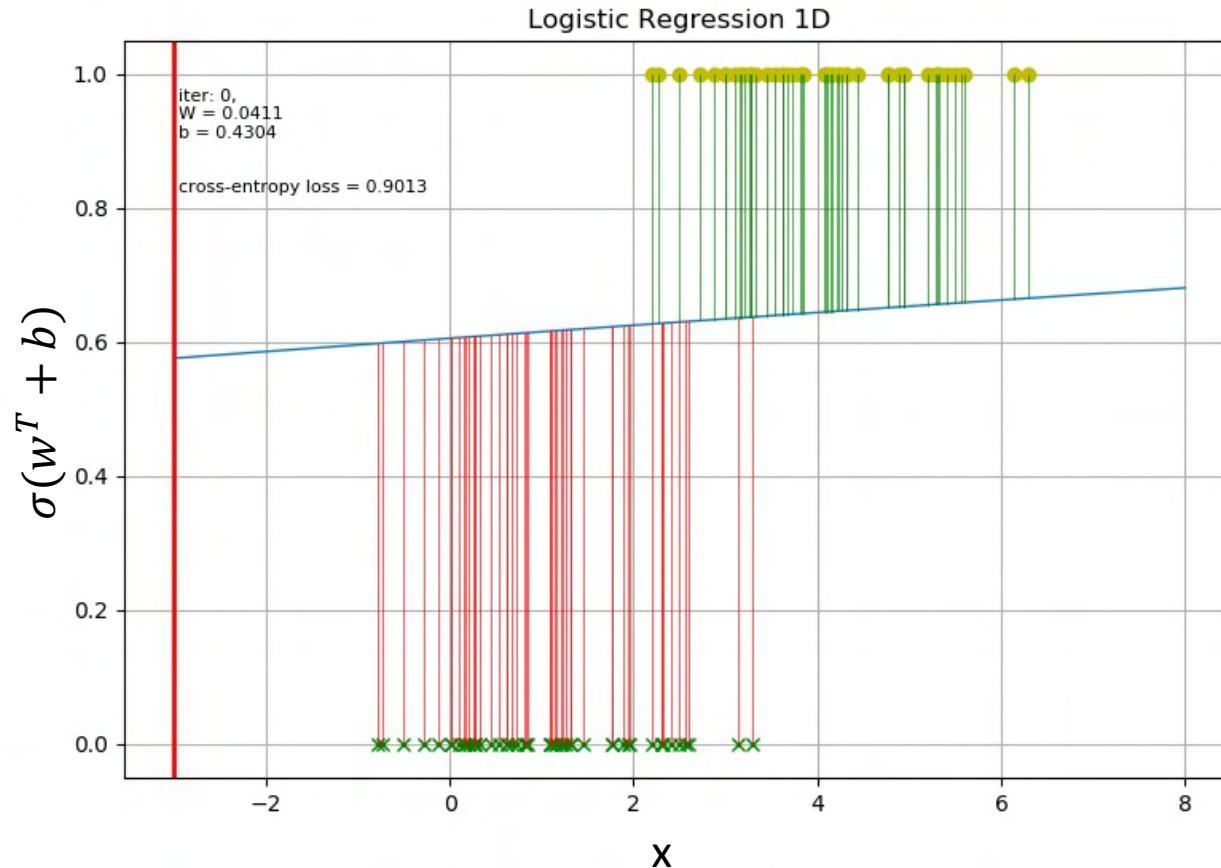
Define a transformation function such that

$$\sigma(w^T x + b) = P(y = 1|x)$$



Logistic Regression in Action

$$\sigma(w^T x + b) \approx P(y = 1|x)$$

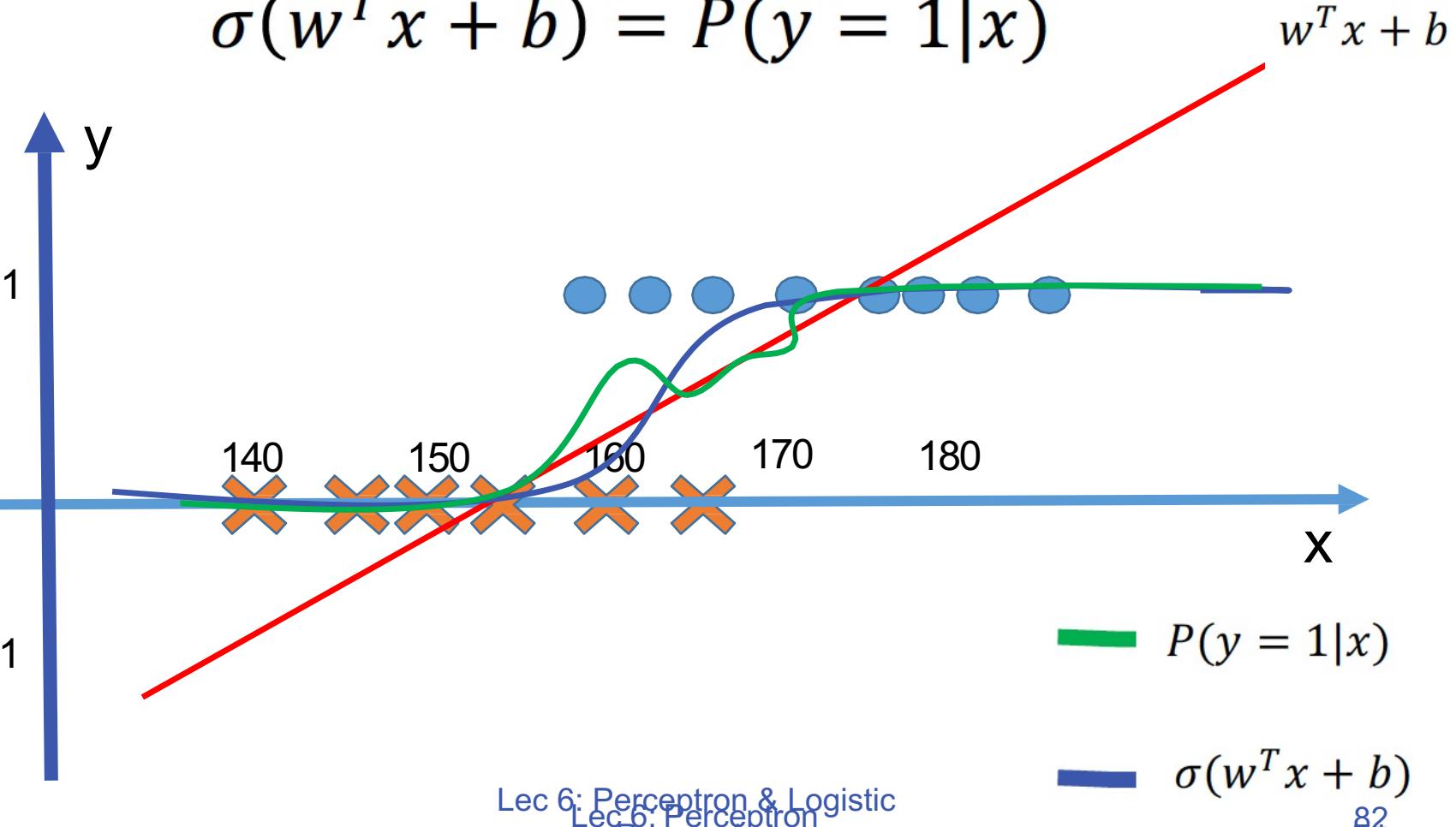


<https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf>

Transform the linear function

Define a transformation function such that

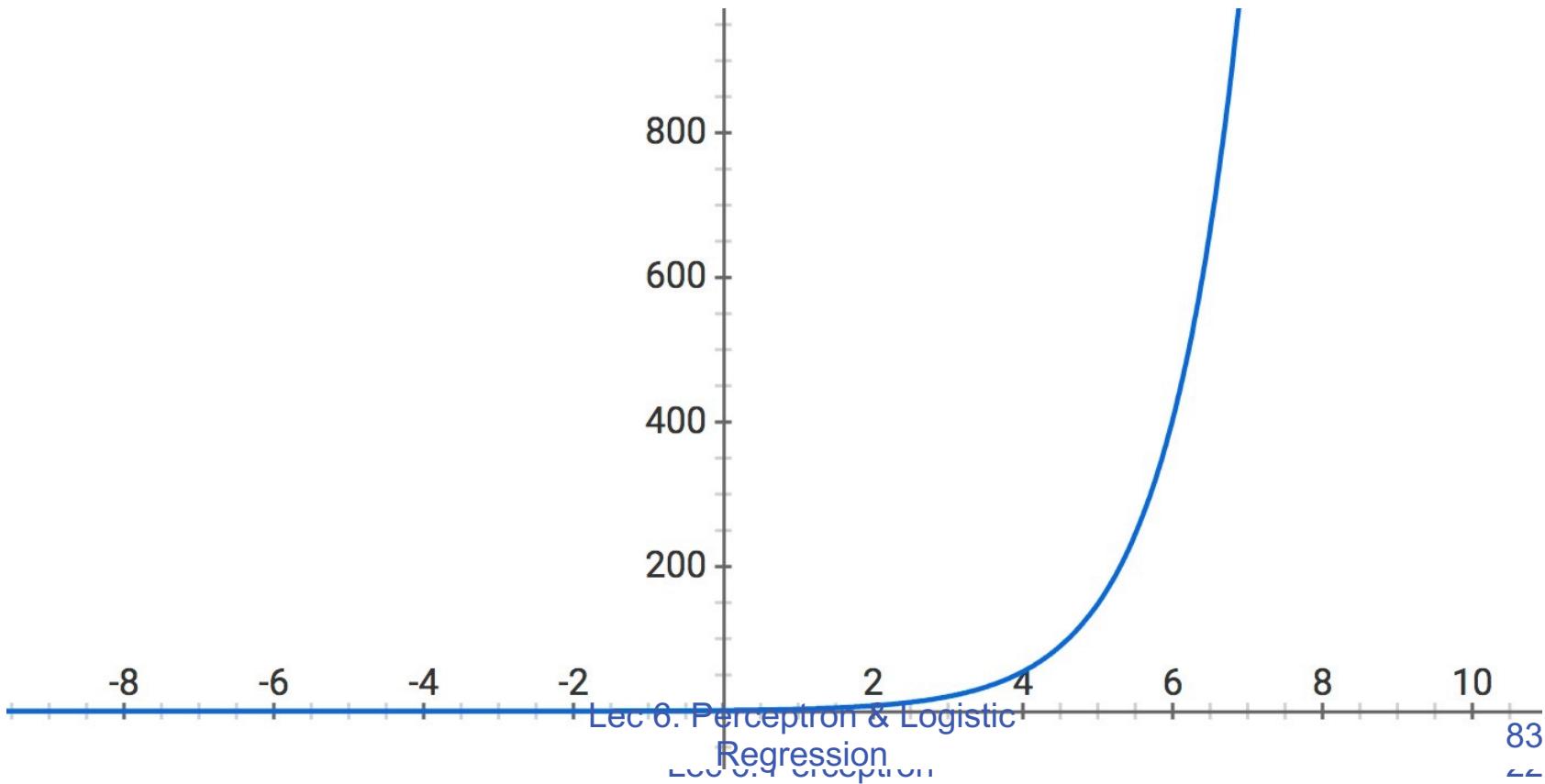
$$\sigma(w^T x + b) = P(y = 1|x)$$



How can we design such a transformation function?

Idea 1: function always output positive value

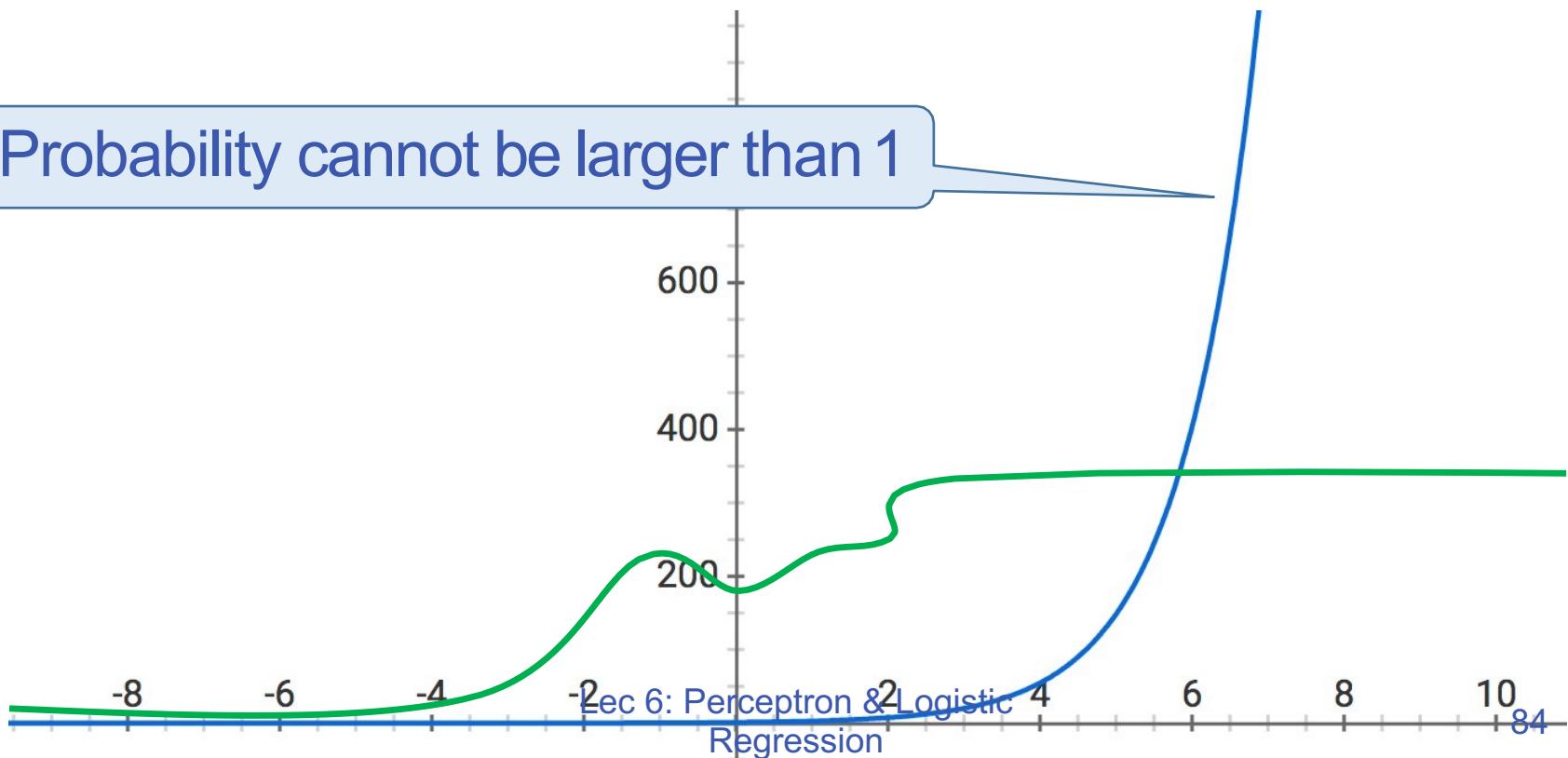
$\exp(\cdot)$ is always positive



How can we design such a transformation function?

- ❖ Idea 1: function always output positive value
 - ❖ $\exp(\cdot)$ is always positive
 - ❖ $\exp(w^T x + b)$ always return positive value

Probability cannot be larger than 1



How can we design such a transformation function?

Idea 2: normalize the value such that it is less than 1

- ❖ $\exp(w^T x + b) \in (0, \infty)$ grows very fast, so we need to use exp to normalize itself
- ❖ Let's use

$$\sigma(w^T x + b) = \frac{\exp(w^T x + b)}{1 + \exp(w^T x + b)}$$

- ❖ When $w^T x + b \rightarrow \infty$,
$$\sigma(w^T x + b) \rightarrow 1$$

- ❖ When $w^T x + b \rightarrow -\infty$,
$$\sigma(w^T x + b) \rightarrow 0$$

The Sigmoid function

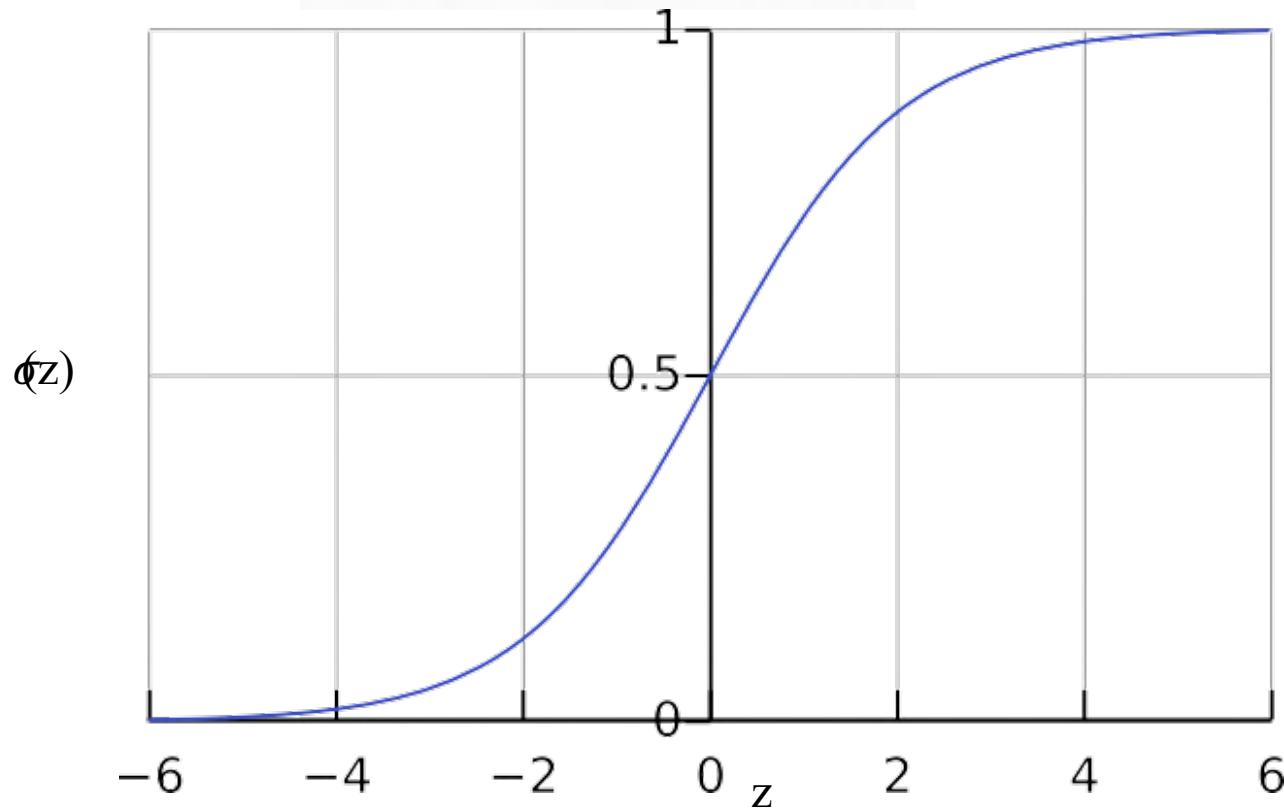
The $\sigma(z)$ function is called sigmoid function
(or logistic function)

$$\begin{aligned}\sigma(z) &= \frac{\exp(z)}{1 + \exp(z)} \\ &= \frac{1}{1 + \exp(-z)}\end{aligned}$$

What is the domain and the range of the sigmoid function?

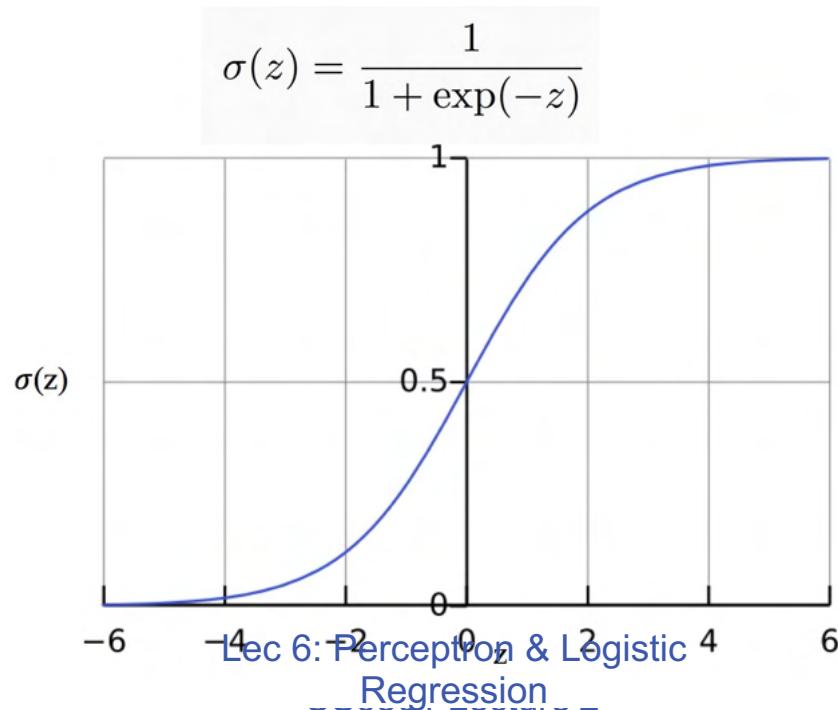
The Sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



logistic function or sigmoid function

- ❖ When $z \rightarrow \infty$ what is $\sigma(z)$?
- ❖ When $z \rightarrow -\infty$ what is $\sigma(z)$?
- ❖ When $z = 0$ what is $\sigma(z)$?



The hypothesis space for logistic regression

Let's ignore b from now

All functions of the form

$$h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

That is, a linear function, composed with a sigmoid function (the logistic function)

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

We want to find $h_Q(x)$ such that

$$h_w(x) \approx P(y = 1 | x)$$

Summary (Modeling)

- ❖ What is the goal of logistic regression?
 - ❖ Model $P(y = 1|x)$
- ❖ What is the hypothesis space?

$$H = \{ h \mid h : X \rightarrow P(Y \mid X), h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Prediction in Logistic Regression

Predicting a label

$$P(y = 1 | \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

Compute $\sigma(w^T x)$; If this is greater than half, predict 1 else predict -1

What does this correspond to in terms of $\mathbf{w}^T \mathbf{x}$?

Decision Boundary

$$P(y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$P(y = -1 | \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

$$\log \frac{P(Y = 1 | x)}{P(Y = -1 | x)} = \mathbf{w}^T \mathbf{x}$$

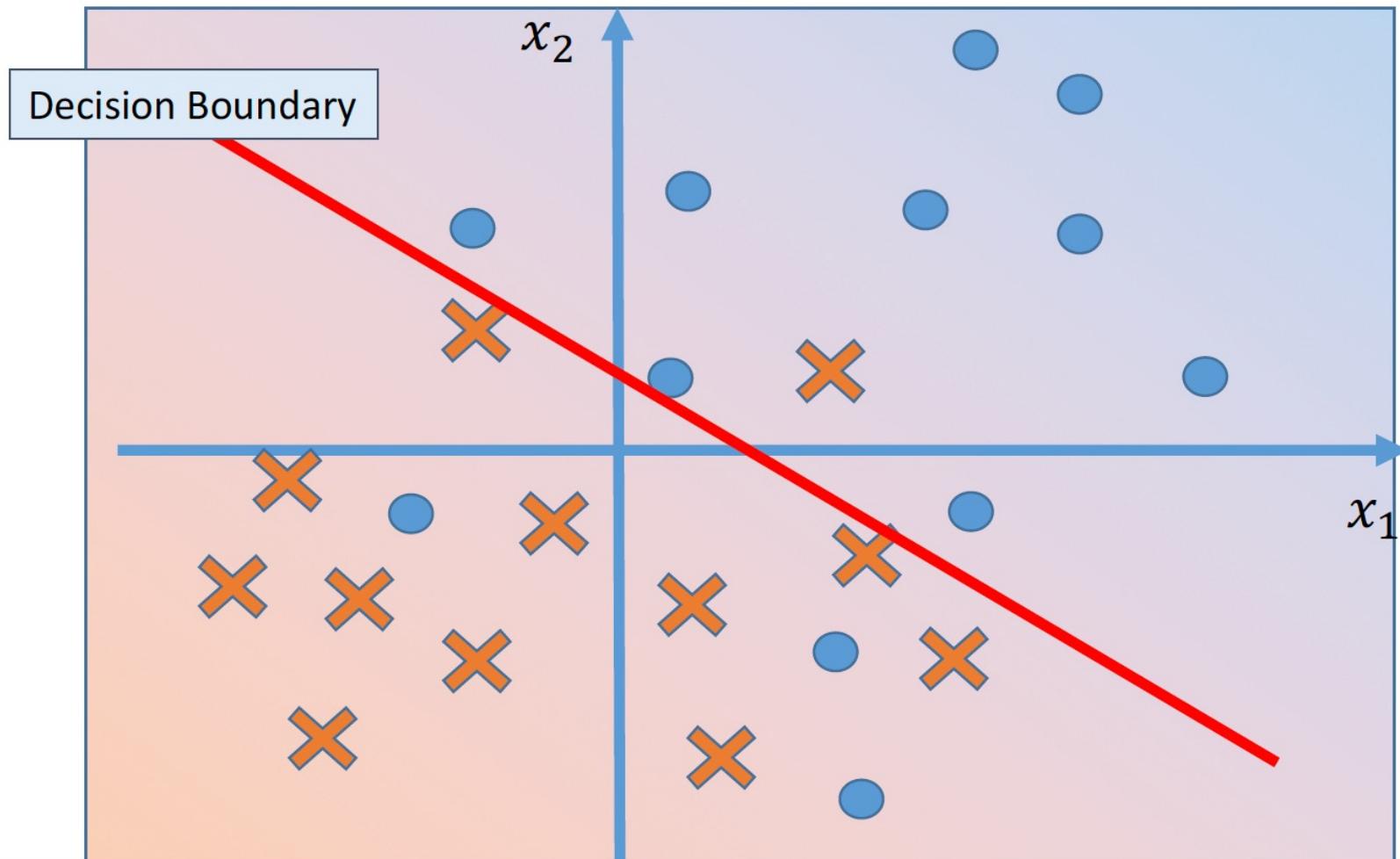
$$\log(z) \geq 0 \quad \text{iff } z \geq 1$$

❖ The decision boundary?

$$\mathbf{w}^T \mathbf{x} = 0$$

❖ $\sigma(\mathbf{w}^T \mathbf{x})$ is non-linear but the decision boundary is linear!

Prediction by logistic regression



A probabilistic model of modeling $\sigma(w^T x + b) = P(y = 1|x)$

Lecture 7: Logistic Regression Fall 2021

Kai-Wei Chang

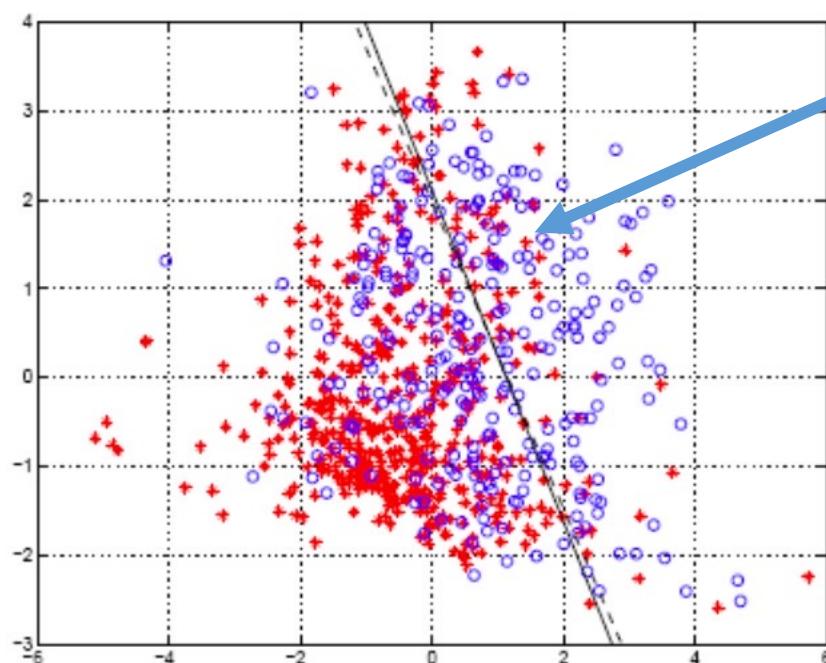
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Classification, but...

- ❖ The output y is discrete valued (-1 or 1)
- ❖ Instead of predicting the output label, let us predict $P(y = 1 | x)$

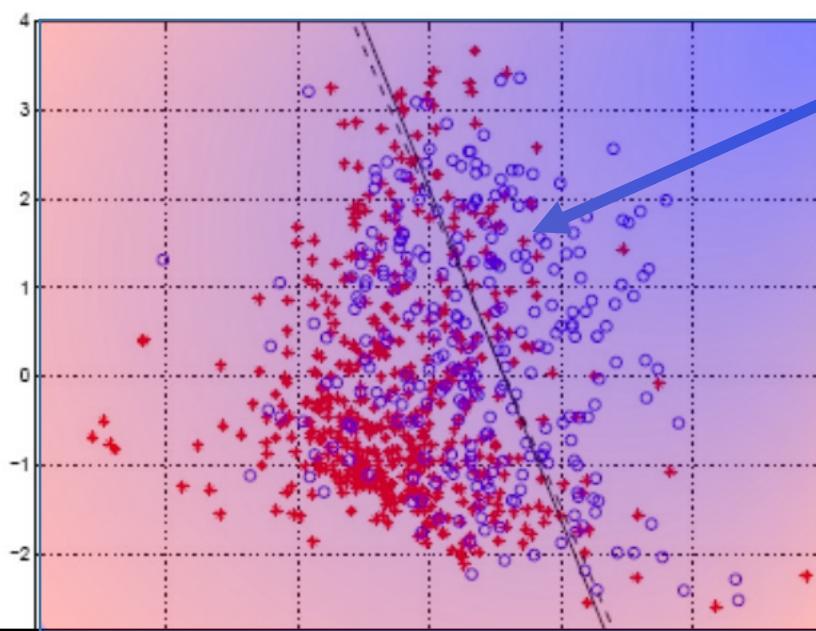


How likely the label $y = 1$
if my feature vector is
in this region

Perceptron does not produce probability estimates

Classification, but...

- ❖ The output y is discrete valued (-1 or 1)
- ❖ Instead of predicting the output label, let us predict $P(y = 1 | x)$

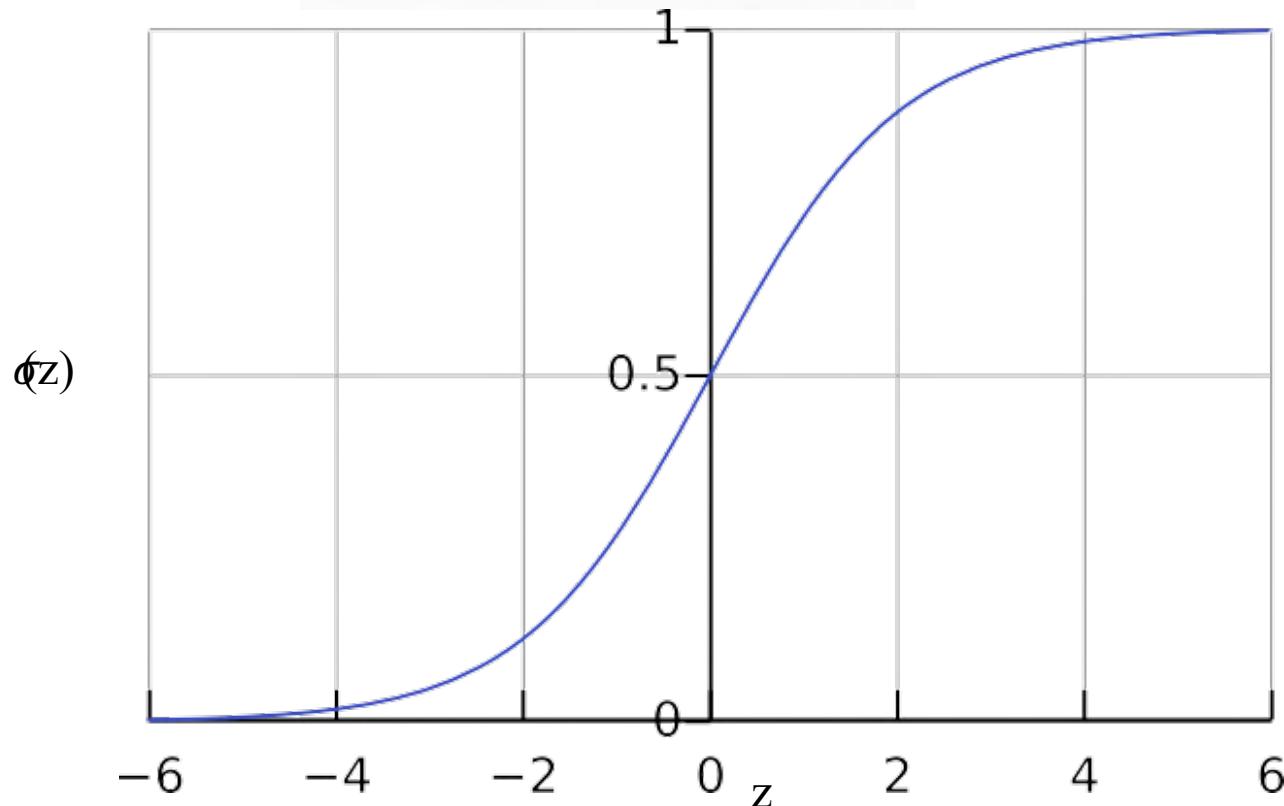


How likely the label $y = 1$ if my feature vector is in this region

Perceptron does not produce probability estimates

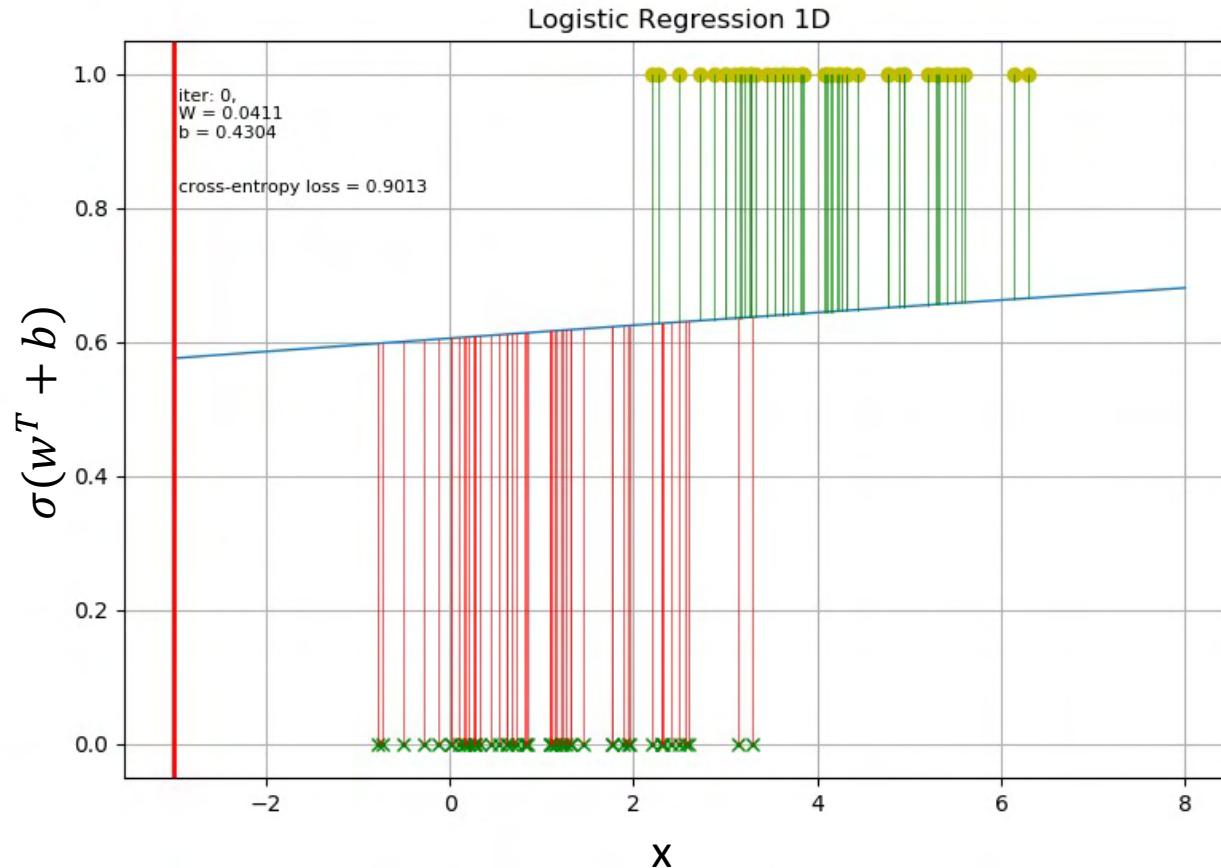
The Sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



Logistic Regression in Action

$$\sigma(w^T x + b) \approx P(y = 1|x)$$



<https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf>

Summary (Modeling)

- ❖ What is the goal of logistic regression?
 - ❖ Model $P(y = 1|x)$
- ❖ What is the hypothesis space?

$$H = \{ h \mid h : X \rightarrow P(Y \mid X), h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Prediction in Logistic Regression

Predicting a label

$$P(y = 1 | \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

Compute $\sigma(w^T x)$; If this is greater than half, predict 1 else predict -1

What does this correspond to in terms of $\mathbf{w}^T \mathbf{x}$?

Decision Boundary

$$P(y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$P(y = -1 | \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

$$\log \frac{P(Y = 1 | x)}{P(Y = -1 | x)} = \mathbf{w}^T \mathbf{x}$$

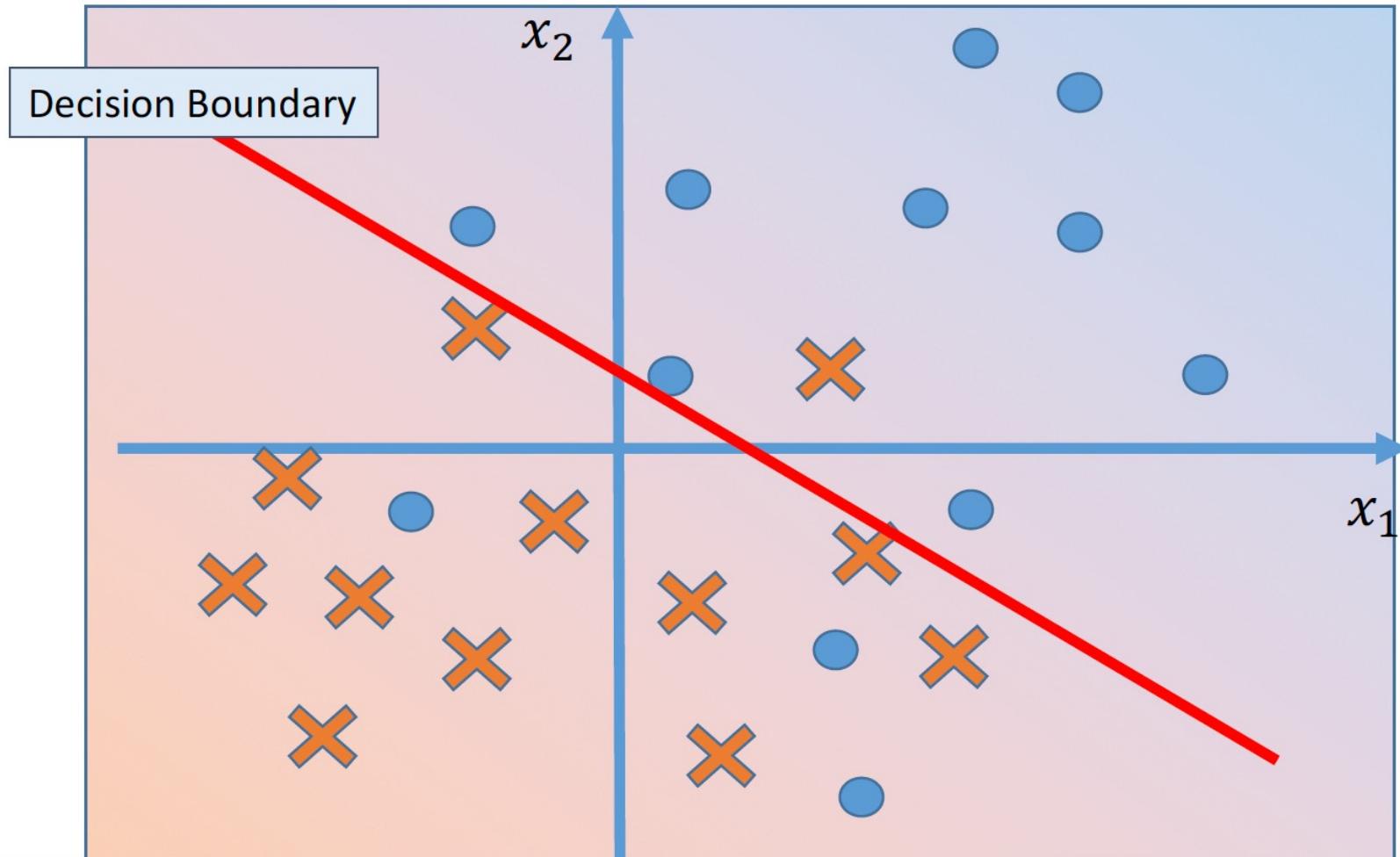
$$\log(z) \geq 0 \quad \text{iff } z \geq 1$$

❖ The decision boundary?

$$\mathbf{w}^T \mathbf{x} = 0$$

❖ $\sigma(\mathbf{w}^T \mathbf{x})$ is non-linear but the decision boundary is linear!

Prediction by logistic regression



A probabilistic model of modeling $\sigma(w^T x + b) = P(y = 1|x)$

How to Train a Logistic Regression Model?

Logistic Regression: Setup

- ❖ The setting
 - ❖ Binary classification
 - ❖ Inputs: Feature vectors $x \in R^N$
 - ❖ Labels: $y \in \{-1, +1\}$
- ❖ Training data
 - ❖ $S = \{(x_i, y_i)\}$, m examples
- ❖ Hypothesis space

$$H = \{ h \mid h : X \rightarrow P(Y \mid X), h(x) = \sigma(w^T x + b) \}$$
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Logistic Regression: Setup

- ❖ Training data

- ❖ $S = \{(x_i, y_i)\}$, m examples

- ❖ Hypothesis space

$$H = \{ h \mid h : X \rightarrow Y, h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- ❖ Learning Goal:

- ❖ Find an $h \in H$, such that $h(x) \approx P(y = 1 \mid x)$

- ❖ How to?

Maximum Likelihood

Which bag of words more likely generate:

aDaaa



Maximum Likelihood

Which bag of words more likely generate:

aDaaa

$$0.7 \times 0.1 \times 0.7 \times 0.7 \times 0.7 \\ = 2.401 \times 10^{-2}$$



$$0.2 \times 0.1 \times 0.2 \times 0.2 \times 0.2 \\ = 1.6 \times 10^{-4}$$



Drawing color cards from the envelope

- ❖ Let say we have several cards in the envelope
- ❖ Assume



$$P(\text{card} = \text{yellow}) = \theta$$

$$P(\text{card} = \text{purple}) = 1 - \theta$$

Drawing color cards from the envelope

- ❖ Sample with replacement n times
- ❖ k times we get yellow card, and n-k times, we get purple card
- ❖ The joint probability (likelihood)
$$C_K^N \theta^k (1 - \theta)^{n-k}$$
- ❖ What is the best θ making the joint probability maximal?

Drawing color cards from the envelope

$$\cancel{C_K^N} \theta^k (1 - \theta)^{n-k}$$

- ❖ Solving $\max_{\theta} \theta^k (1 - \theta)^{n-k}$
- ❖ Equivalently, we can solve

$$\max_{\theta} \log(\theta^k (1 - \theta)^{n-k})$$

$$\max_{\theta} k \log \theta + (n - k) \log(1 - \theta)$$

- ❖ At the optimum,

$$\frac{d(k \log \theta + (n - k) \log(1 - \theta))}{d\theta} = 0$$

The usual trick: Convert products to sums by taking log

Recall that this works only because log is an increasing function and the maximizer will not change

Drawing color cards from the envelope

$$\frac{d(k \log \theta + (n-k) \log(1-\theta))}{d\theta} = 0$$

$$\Rightarrow \frac{k}{\theta} - \frac{n-k}{1-\theta} = 0$$

For this simple problem, we have a closed-form solution.
We are not always lucky like this

$$\Rightarrow \theta(n - k) = (1 - \theta)k$$

$$\Rightarrow \theta = \frac{k}{n}$$



Maximum Likelihood Estimator (formal definition)

Likelihood function of parameters

Let X_1, \dots, X_N be **IID** (independent and identically distributed) with PDF $p(x|\theta)$ (also written as $p(x; \theta)$). The *likelihood function* is defined by $L(\theta)$,

$$L(\theta) = p(X_1, \dots, X_N; \theta). \quad = \prod_{i=1}^N p(X_i; \theta).$$

Notes The likelihood function is just the joint density of the data, except that we treat it as a function of the parameter θ .

Maximum Likelihood Estimation

Maximum Likelihood Estimator

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

Back to Logistic regression

Training data

$S = \{(x_i, y_i)\}$, N examples

Hypothesis space

$$H = \{ h \mid h : X \rightarrow Y, h(x) = \sigma(w^T x + b) \}$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Find an $h \in H$, such that $h(x) \approx P(y = 1 \mid x)$

How to? **Maximum Likelihood Estimator**

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i, \mathbf{w})$$

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^N \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^N \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

By definition, we know that

$$P(y|\mathbf{w}, \mathbf{x}) = \sigma(y_i \mathbf{w}^T \mathbf{x}_i) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

Maximum likelihood estimator for logistic regression

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^N \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^N -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$P(y|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-y \mathbf{w}^T \mathbf{x})}$$

Minimizing Negative Log-likelihood

- ❖ It is convenient to work with its negation termed **negative log likelihood**

$$\max_{\mathbf{w}} \sum_i^N -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$\min_{\mathbf{w}} \sum_i^N \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$



Alternative Formulation

- ❖ Sometimes, we define
 - ❖ Inputs: Feature vectors $x \in R^N$
 - ❖ Labels: $y \in \{0, 1\}$
- ❖ Training data
 - ❖ $S = \{(x_i, y_i)\}$, N examples

Alternative representation

$$P(y|\mathbf{w}, \mathbf{x}) = \sigma(y\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-y\mathbf{w}^T \mathbf{x})}$$

Maximum Likelihood estimator:

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$P(y_i|\mathbf{x}_i, \mathbf{w}) = \begin{cases} \sigma(w^T x) & \text{if } y_i = 1 \\ 1 - \sigma(w^T x) & \text{if } y_i = 0 \end{cases}$$

Alternative representation

$$P(y|\mathbf{w}, \mathbf{x}) = \sigma(y\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-y\mathbf{w}^T \mathbf{x})}$$

Maximum Likelihood estimator:

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$P(y|\mathbf{x}_i, \mathbf{w}) = \begin{cases} \sigma(\mathbf{w}^T \mathbf{x}) & \text{if } y = 1 \\ 1 - \sigma(\mathbf{w}^T \mathbf{x}) & \text{if } y = 0 \end{cases}$$

$$\operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y = 1|\mathbf{x}_i, \mathbf{w})^{y_i} P(y = 0|\mathbf{x}_i, \mathbf{w})^{1-y_i}$$

Alternative representation

$$P(y|\mathbf{w}, \mathbf{x}) = \sigma(y\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-y\mathbf{w}^T \mathbf{x})}$$

Maximum Log-Likelihood estimator:

$$\operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y=1|\mathbf{x}_i, \mathbf{w})^{y_i} P(y=0|\mathbf{x}_i, \mathbf{w})^{1-y_i}$$

Equivalent to solving

$$P(y|\mathbf{x}_i, \mathbf{w}) = \begin{cases} \sigma(\mathbf{w}^T \mathbf{x}) & \text{if } y = 1 \\ 1 - \sigma(\mathbf{w}^T \mathbf{x}) & \text{if } y = 0 \end{cases}$$

$$\operatorname{argmax}_{\mathbf{w}} \sum_{i=1}^N y_i \log \sigma(\mathbf{w}^T \mathbf{x}) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}))$$

How to Optimize the Loss?

How to minimizing the loss

- ❖ Optimization methods
 - ❖ Gradient Descent
 - ❖ Stochastic Gradient Descent
 - ❖ Analytic solution
- ❖ ...many other approaches

How to solve it?

$$\min_{\mathbf{w}} \sum_i^N \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

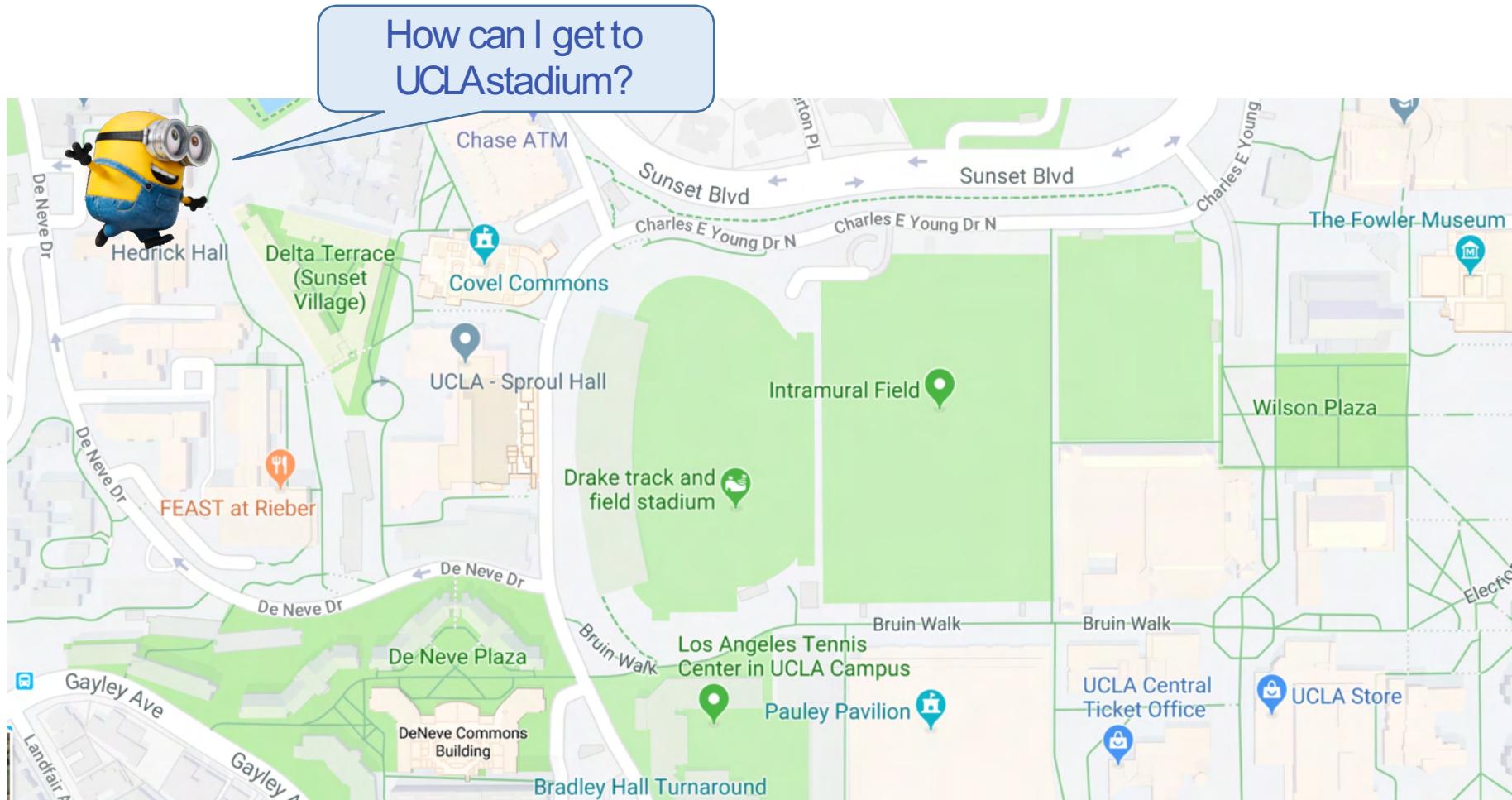
There is no closed-form solution

One way to solve it is by gradient descent

Gradient Descent

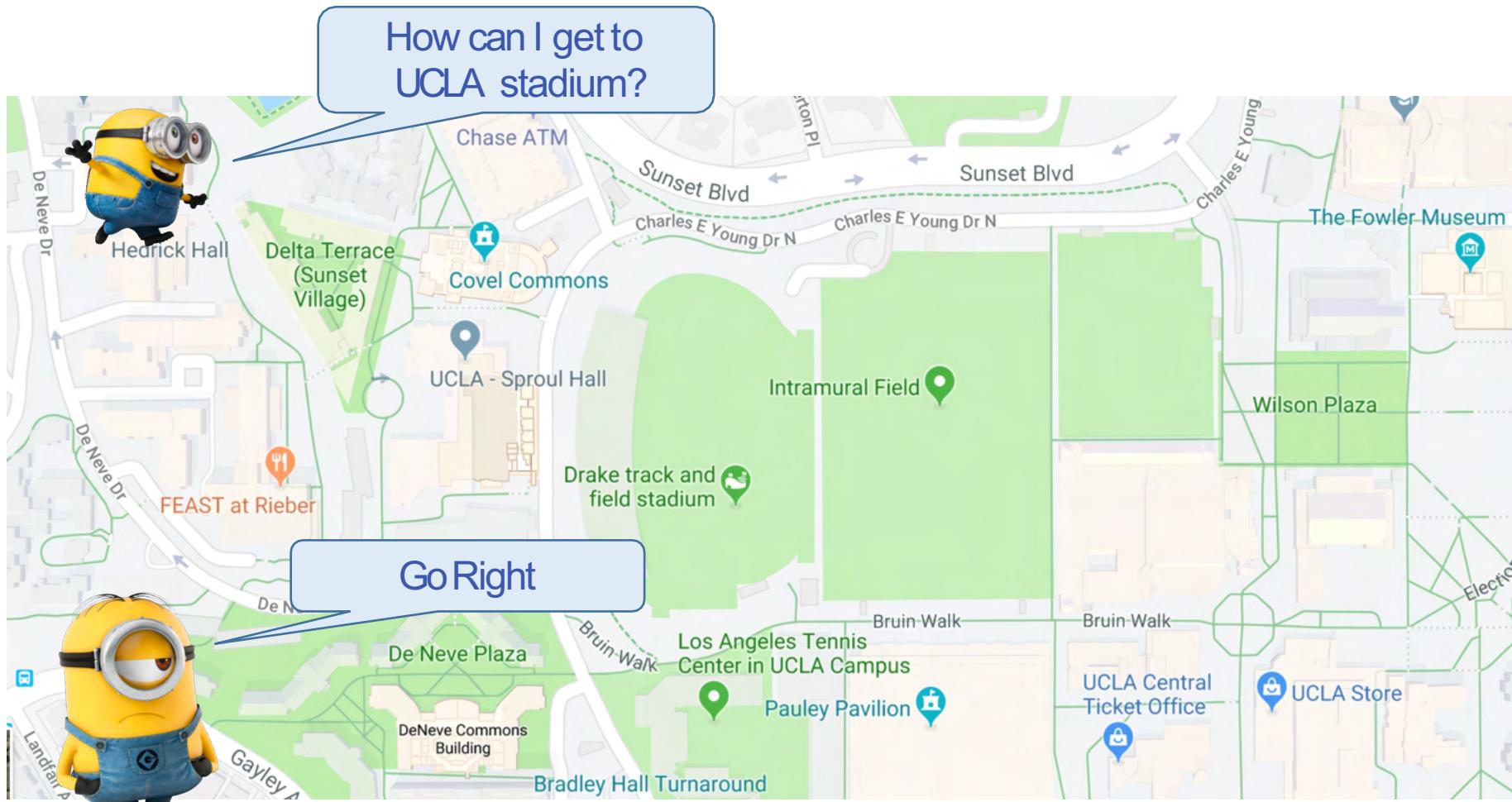
Intuition

Asking direction



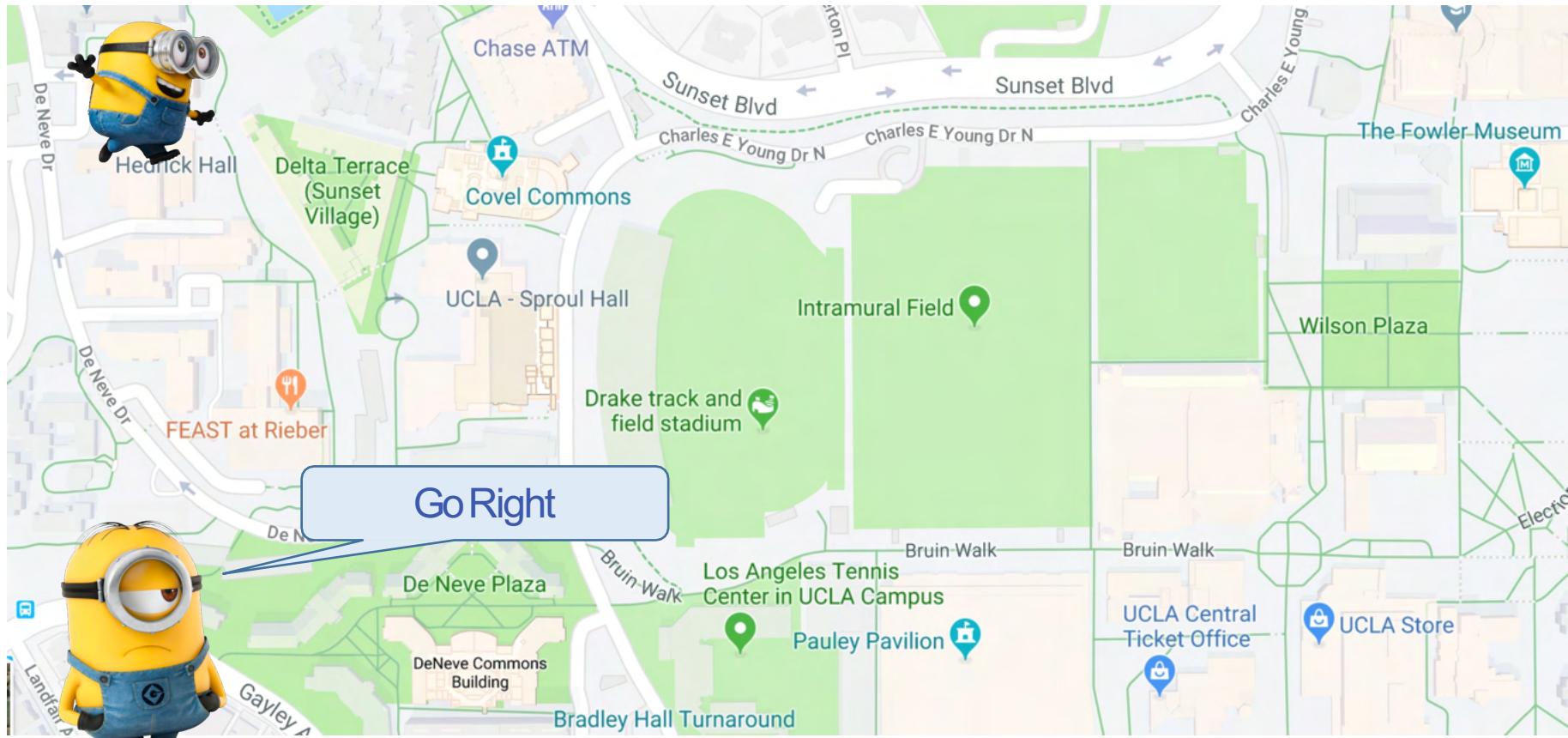
Intuition

Asking direction



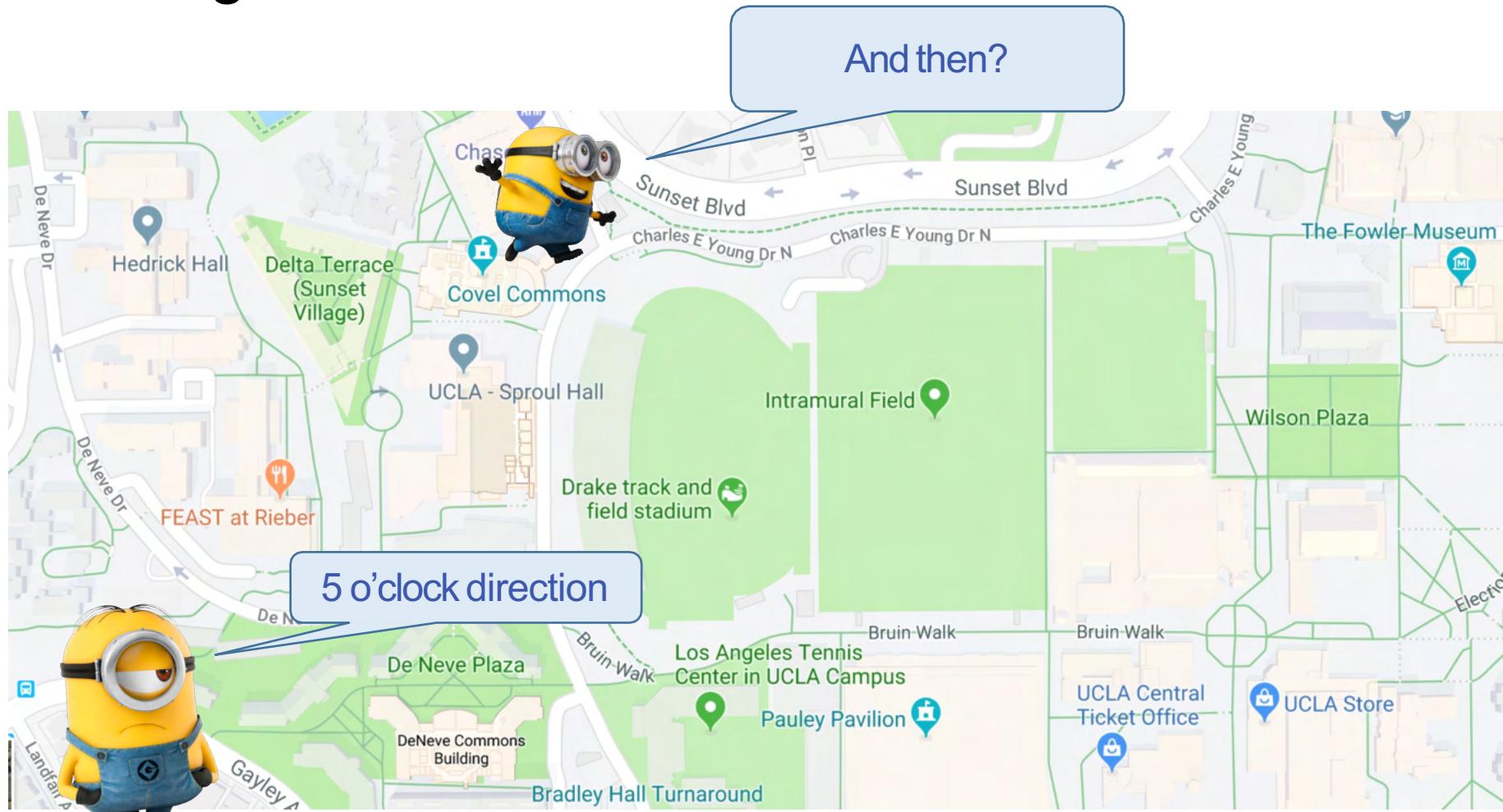
Intuition

Asking direction



Intuition

Asking direction



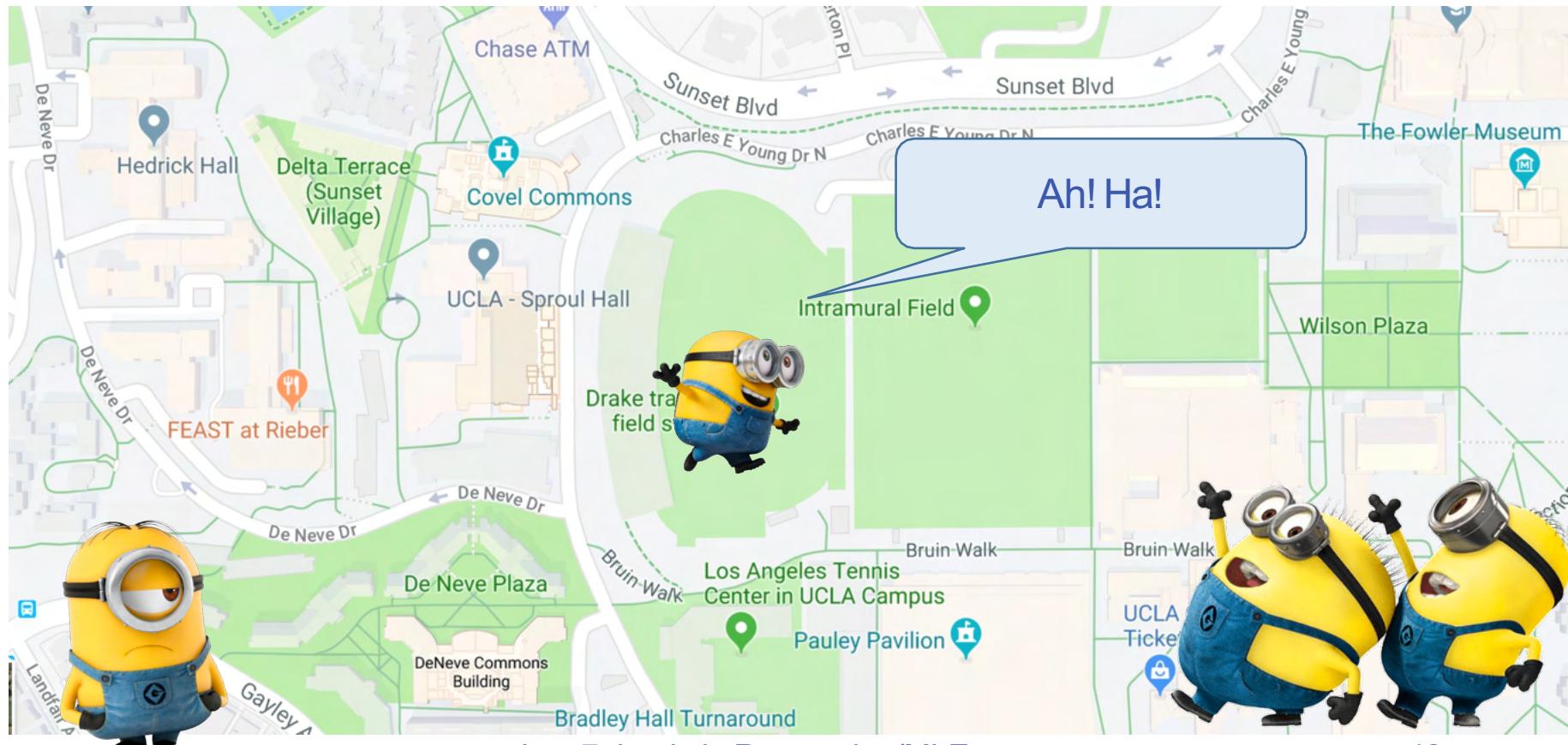
Intuition

Asking direction



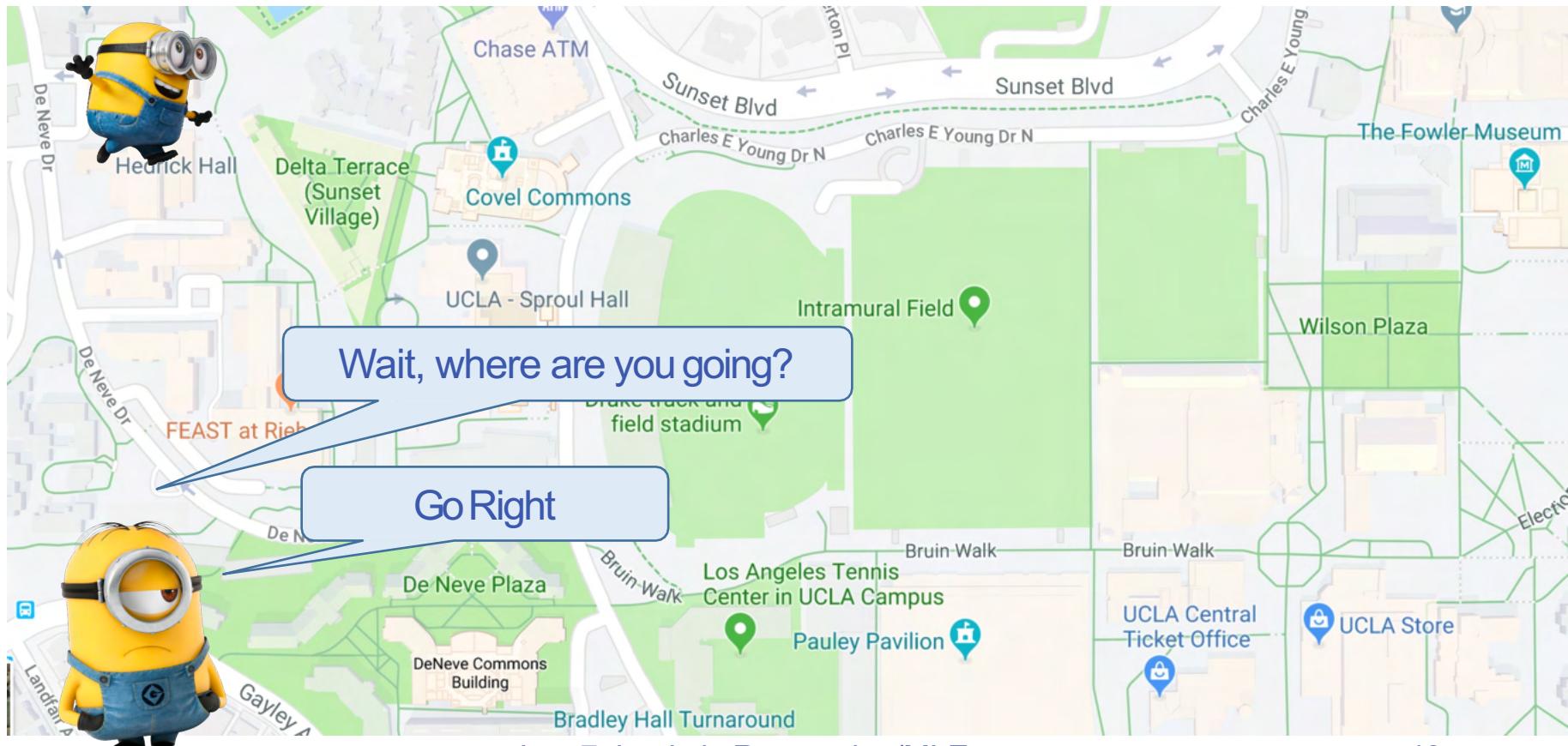
Intuition

Asking direction



Intuition

What may go wrong? Incorrect Step-size

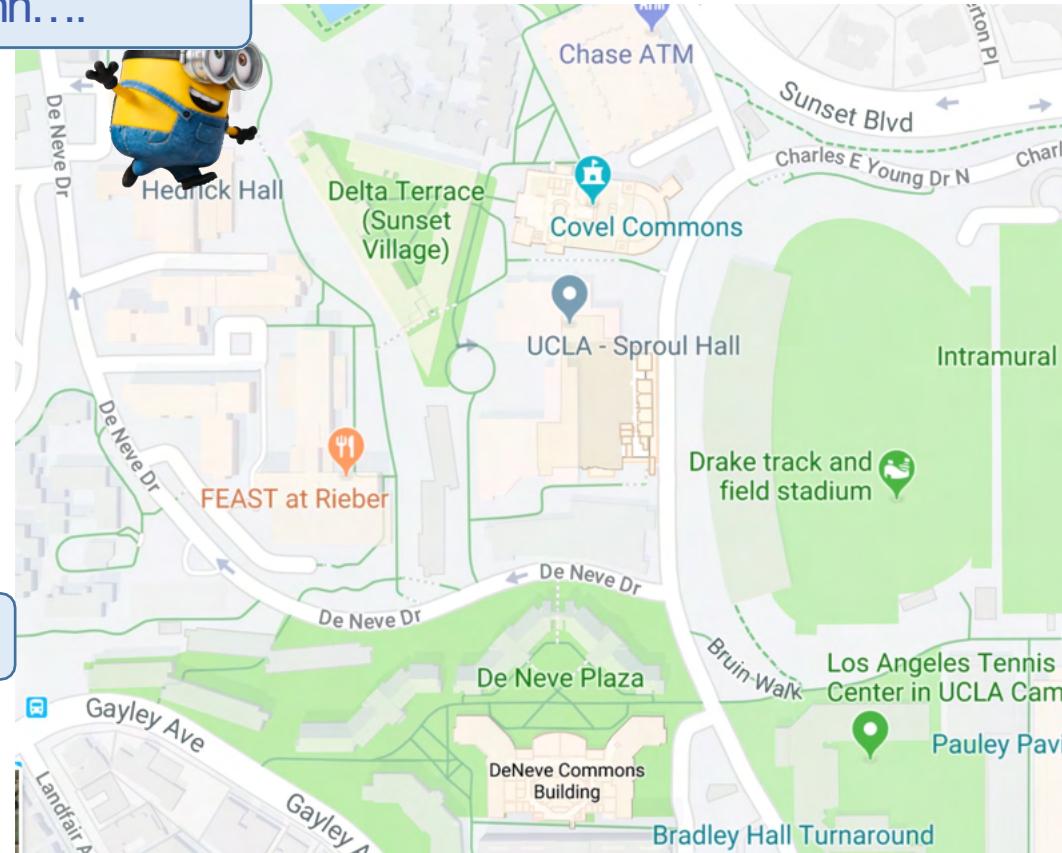


Intuition

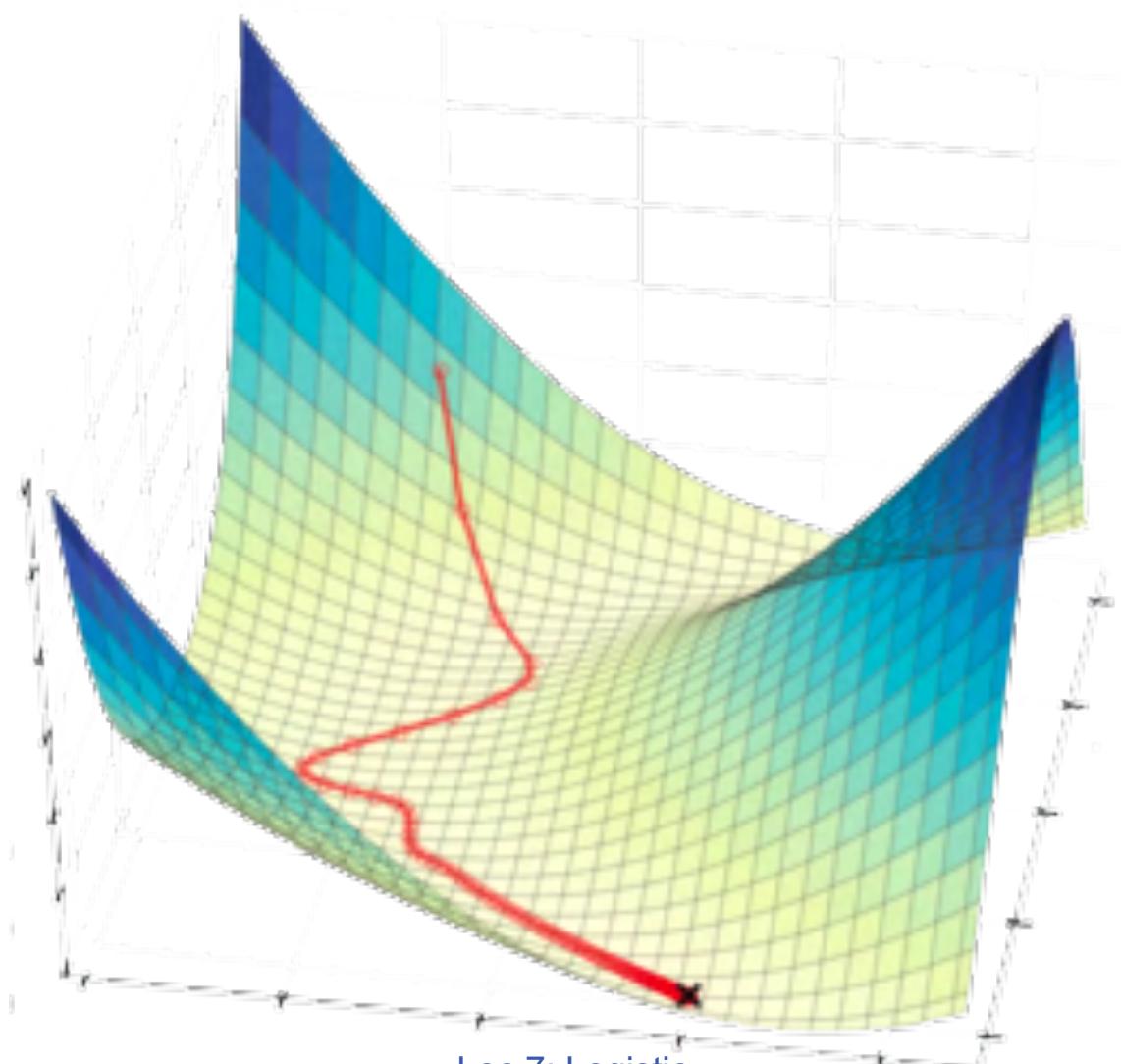
What may go wrong? Incorrect Direction



AHHhhhhhhh....



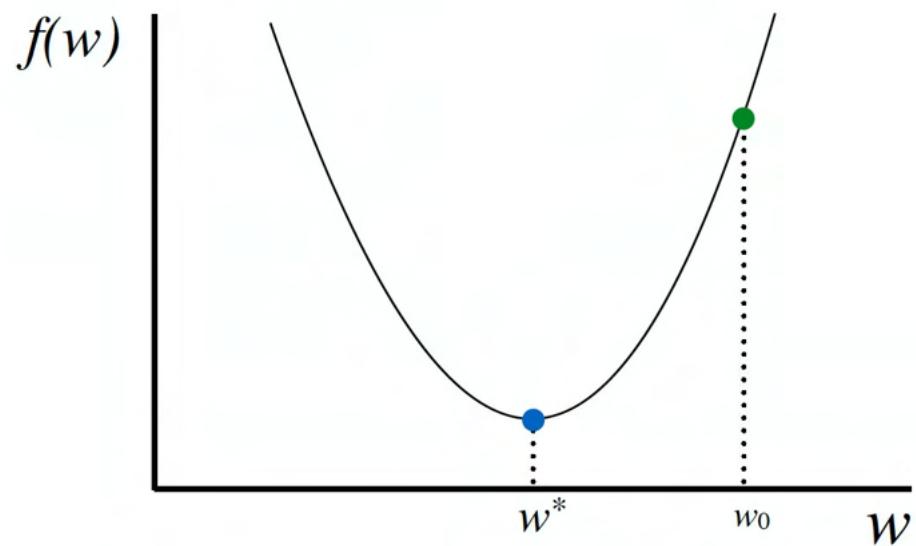
Gradient Descent



Lec 7: Logistic
Regression/MLE

Gradient descent

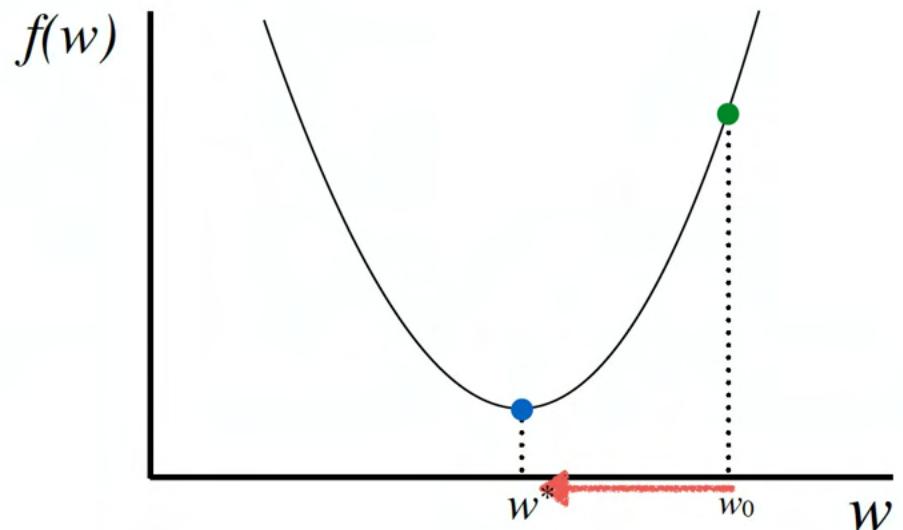
Start at a random point



Gradient descent

Start at a random point

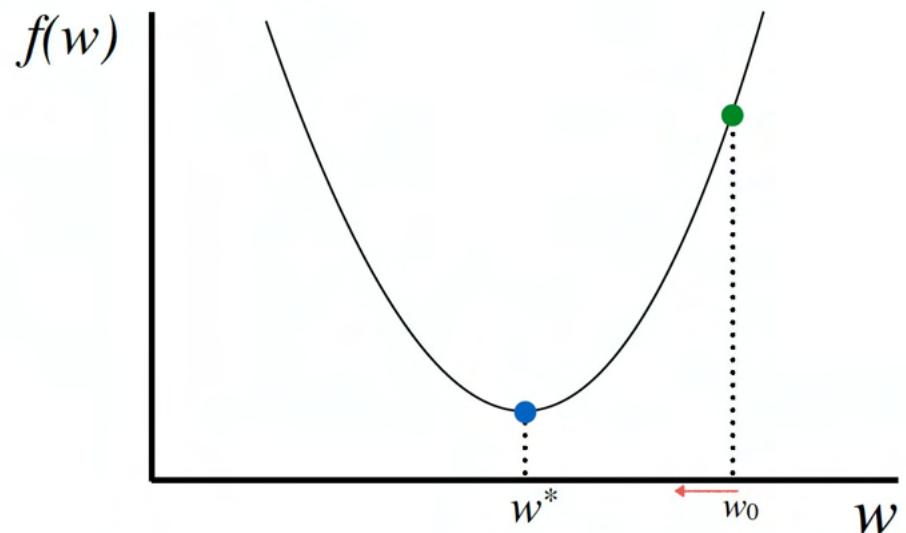
Determine a descent direction



Gradient descent

Start at a random point

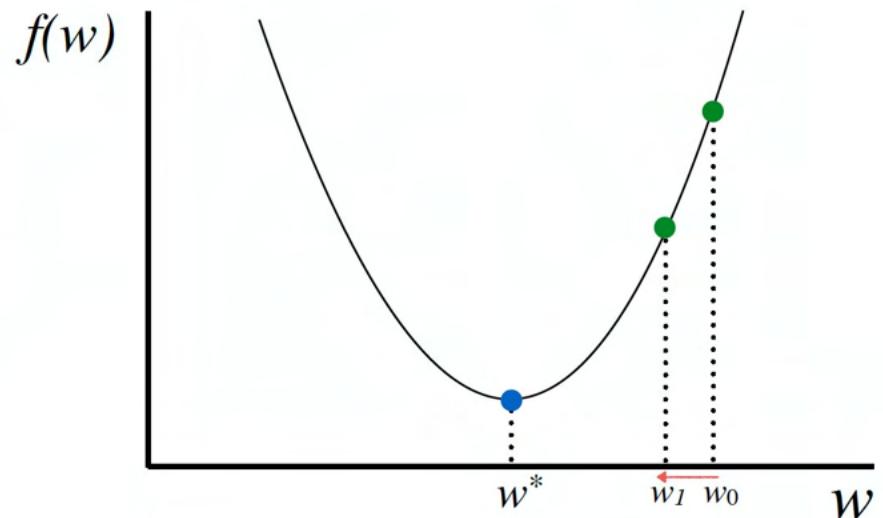
Determine a descent direction
Choose a step size



Gradient descent

Start at a random point

Determine a descent direction
Choose a step size
Update



Gradient descent

Start at a random point

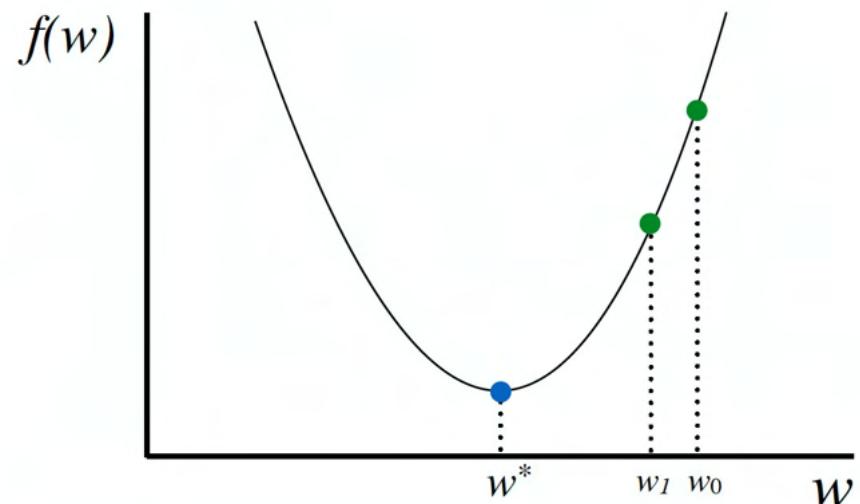
Repeat

Determine a descent direction

Choose a step size

Update

Until stopping criterion is satisfied



Gradient descent

Start at a random point

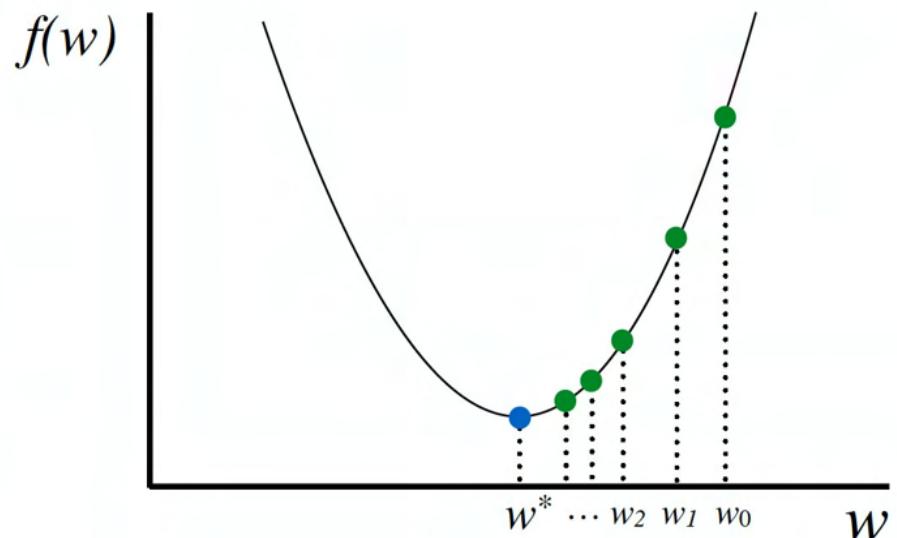
Repeat

Determine a descent direction

Choose a step size

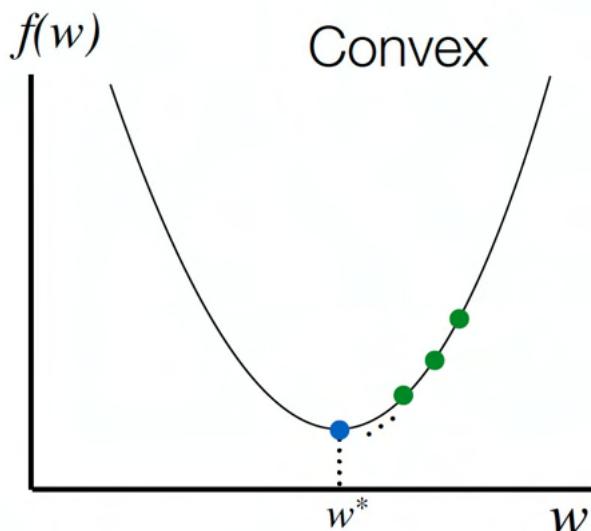
Update

Until stopping criterion is satisfied

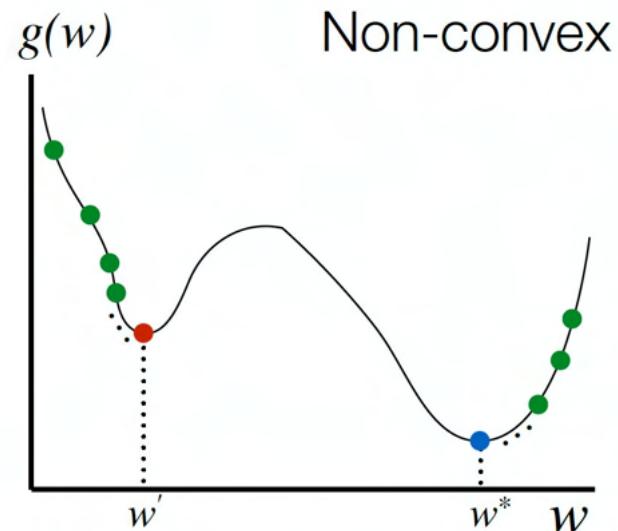


Where will we converge?

If function is convex, it converges to the global optimum (need proper choice of step-size)



Any local minimum is a global minimum



Multiple local minima may exist

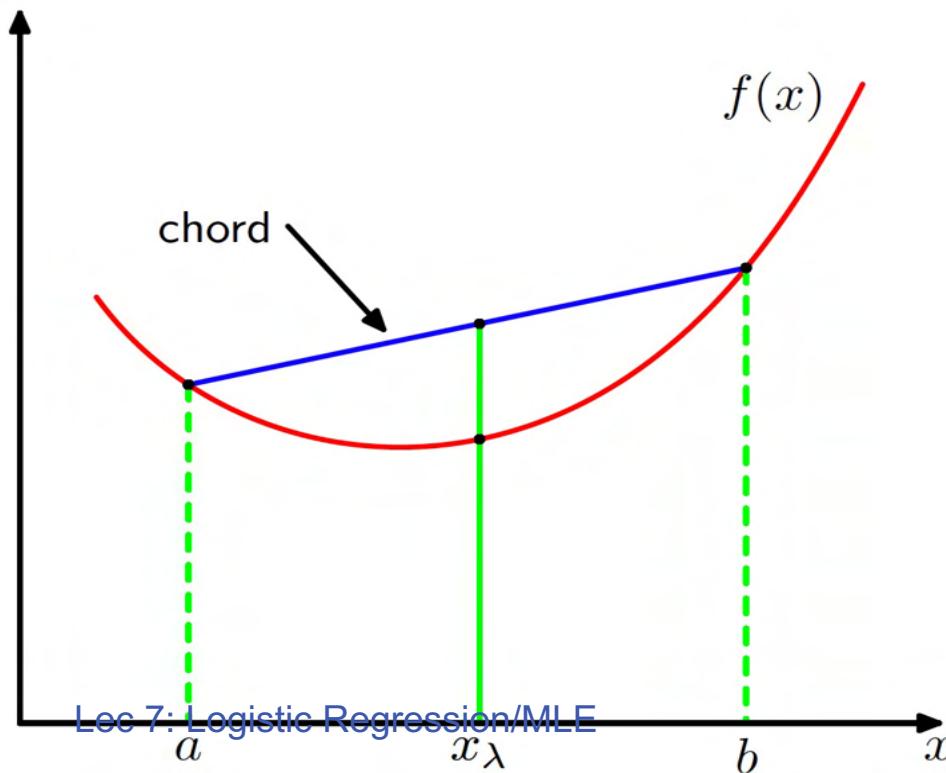
Convex function

A function $f(x)$ is convex if

$$f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b)$$

for

$$0 \leq \lambda \leq 1$$



How to determine convexity?

$f(x)$ is convex if

$$f''(x) \geq 0$$

Examples:

$$f(x) = x^2, f''(x) = 2 > 0$$

Exercise

Prove the following functions are convex

$$f(x) = ax + b$$

$$f(x) = x^2$$

$$f(x) = e^x$$

Lecture 8: Linear Regression Fall 2021

Kai-Wei Chang

CS @ UCLA

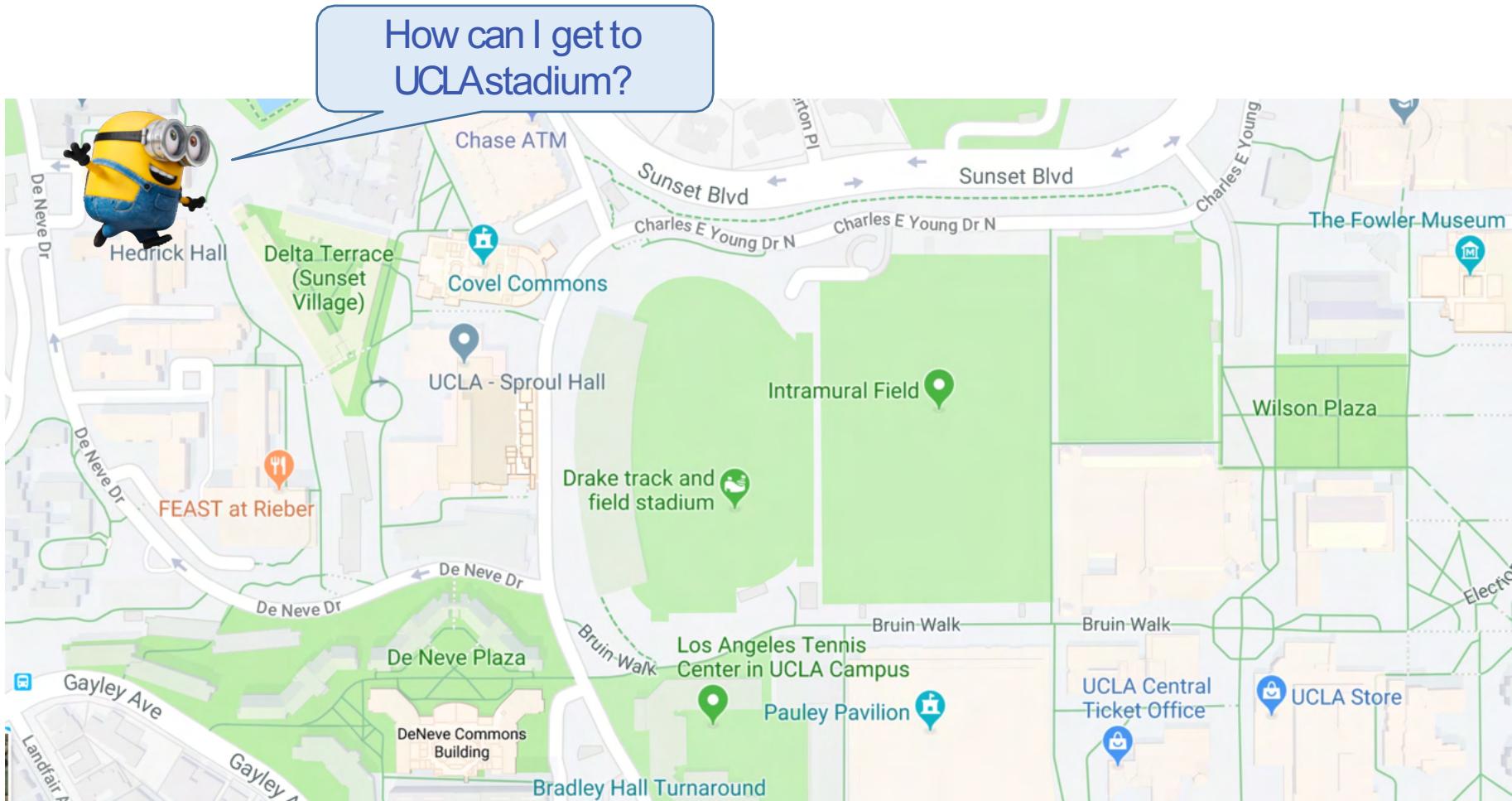
kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Gradient Descent

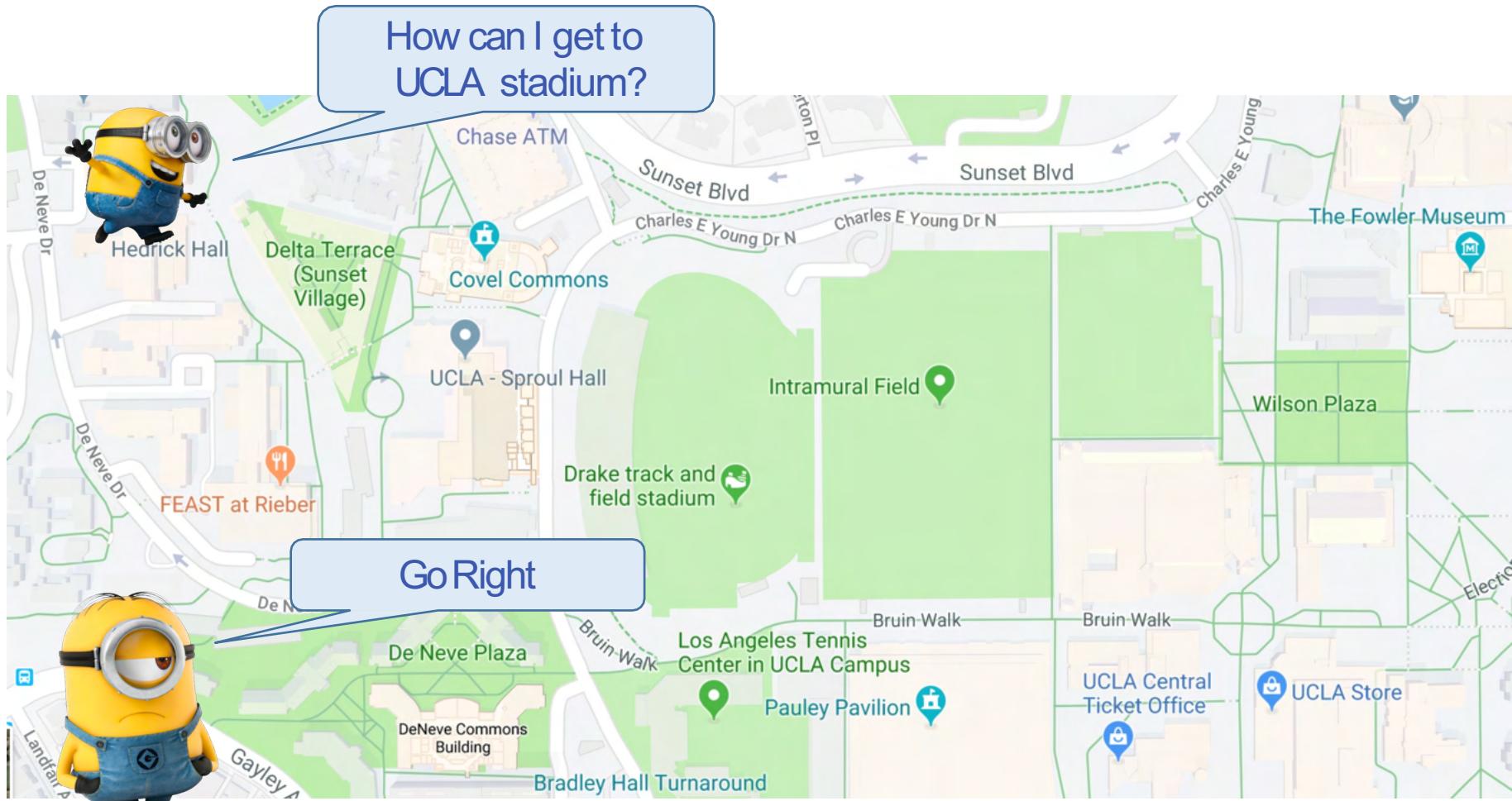
Intuition

Asking direction



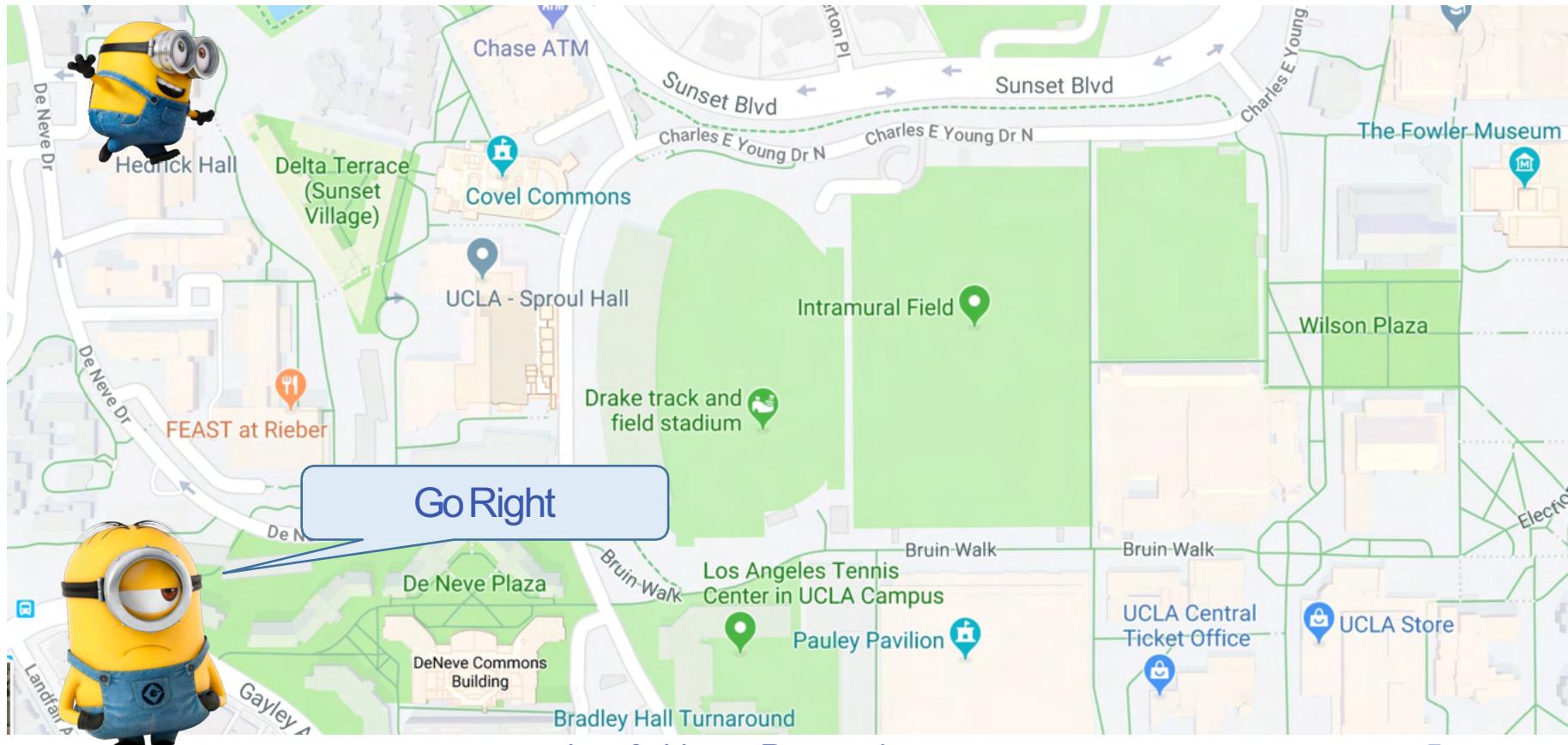
Intuition

Asking direction



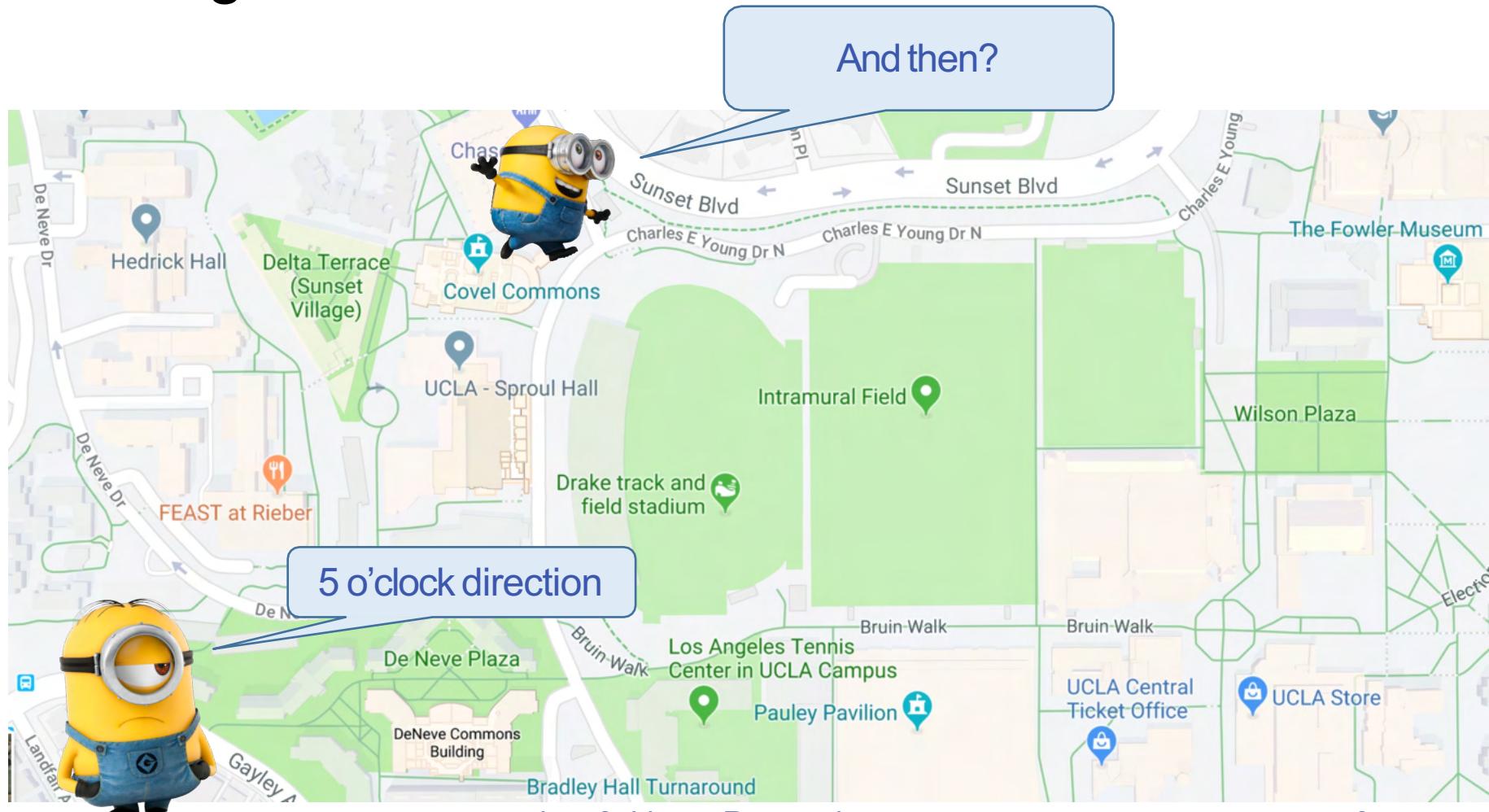
Intuition

Asking direction



Intuition

Asking direction



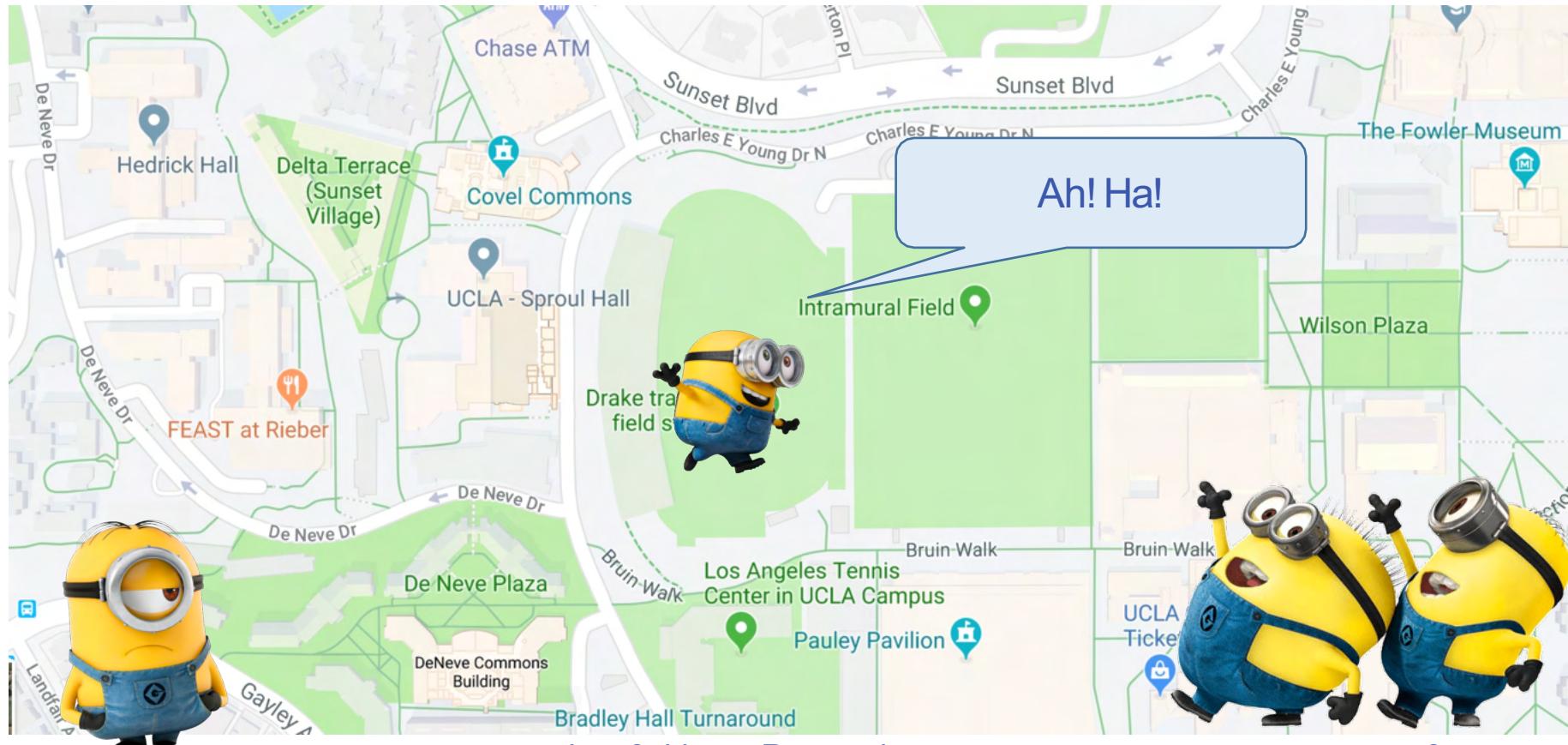
Intuition

Asking direction

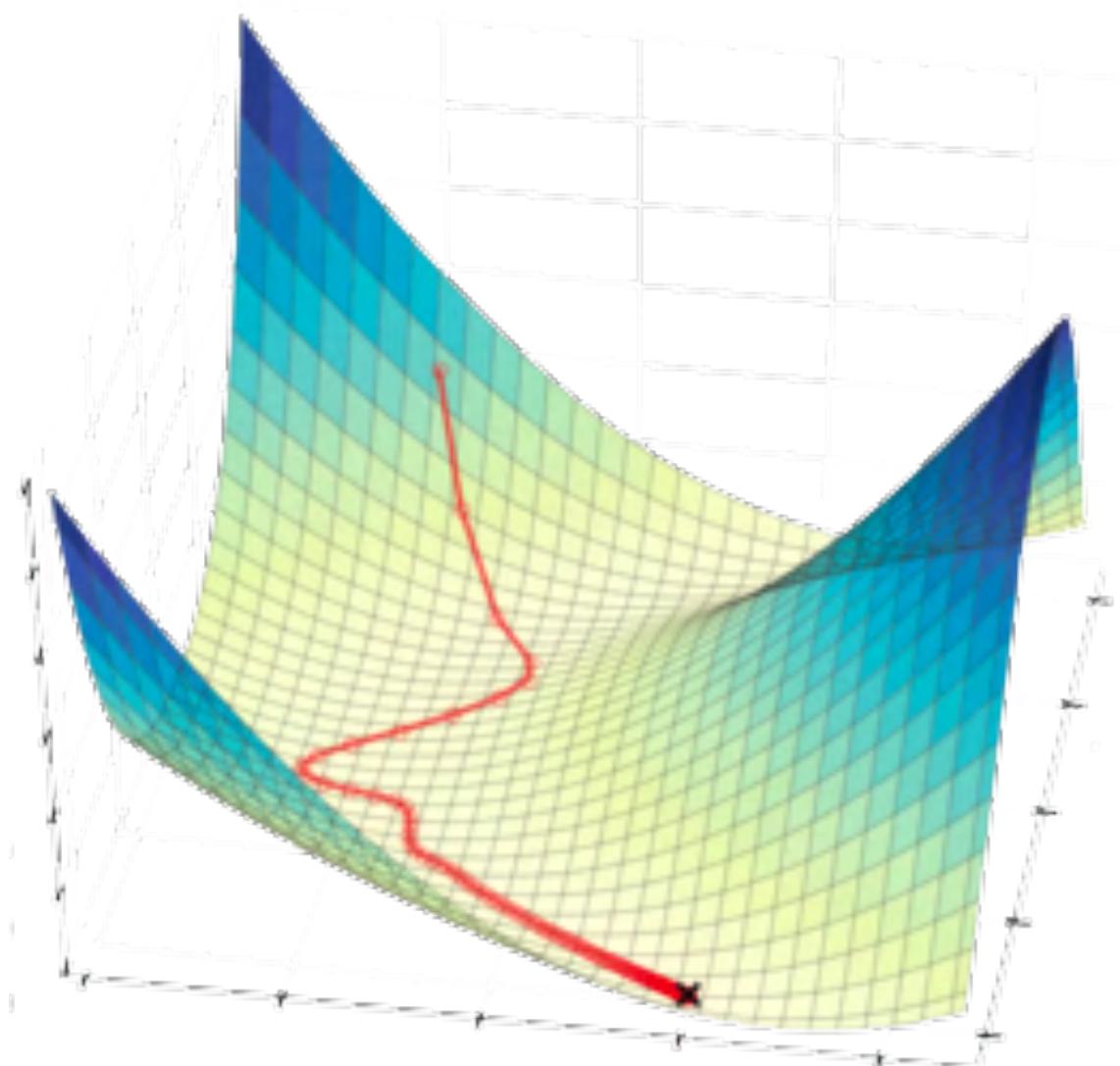


Intuition

Asking direction



Gradient Descent



Gradient Descent

Algorithm 1 Gradient Descent (J)

- 1: $t \leftarrow 0$
 - 2: Initialize $\theta^{(0)}$
 - 3: **repeat**
 - 4: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla J(\theta^{(t)})$
 - 5: $t \leftarrow t + 1$
 - 6: **until** convergence
 - 7: Return final value of θ
-

Need to compute the gradient for the negative log likelihood

Example

$$\min f(\boldsymbol{\theta}) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$$

We compute the gradients

$$\frac{\partial f}{\partial \theta_1} = 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1$$

$$\frac{\partial f}{\partial \theta_2} = -(\theta_1^2 - \theta_2)$$

Example $\min f(\theta) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2$

- ❖ Use the following iterative procedure for gradient descent

- ➊ Initialize $\theta_1^{(0)}$ and $\theta_2^{(0)}$, and $t = 0$
- ➋ do

Type equation here.

$$\nabla f(\theta) = \begin{bmatrix} 2(\theta_1^2 - \theta_2)\theta_1 + \theta_1 - 1 \\ -(\theta_1^2 - \theta_2) \end{bmatrix}$$

$$\theta_1^{(t+1)} \leftarrow \theta_1^{(t)} - \eta \left[2(\theta_1^{(t)})^2 - \theta_2^{(t)} \right] \theta_1^{(t)} + \theta_1^{(t)} - 1$$

$$\theta_2^{(t+1)} \leftarrow \theta_2^{(t)} - \eta \left[-(\theta_1^{(t)})^2 - \theta_2^{(t)} \right]$$

$$t \leftarrow t + 1$$

- ➌ until $f(\theta^{(t)})$ does not change much

Remarks

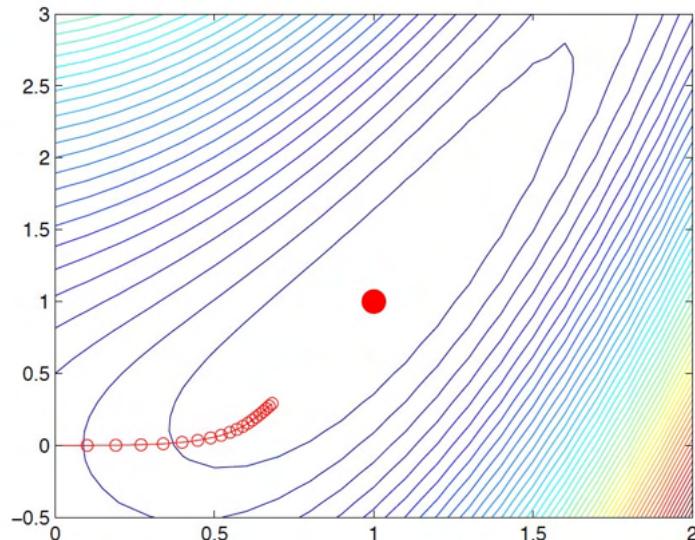
- ❖ η is often called step size or learning rate -- how far our update will go along the the direction of the negative gradient
- ❖ With a **suitable** choice of η , the iterative procedure converges to a stationary point where

$$\frac{\partial f}{\partial \theta} = 0$$

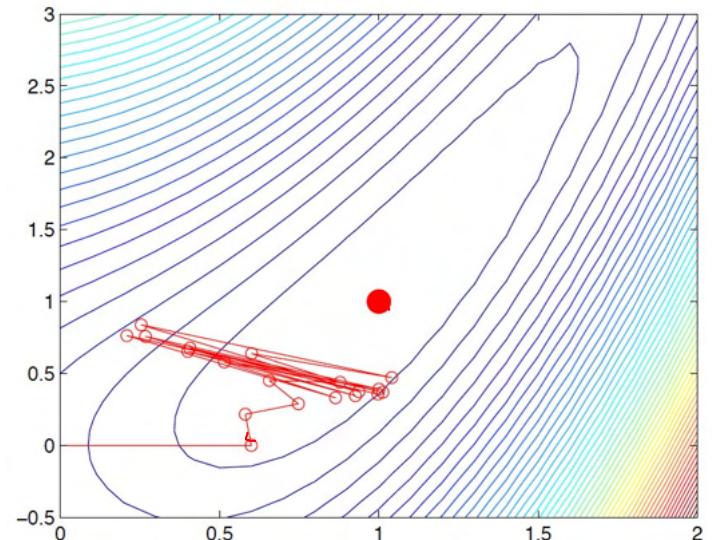
- ❖ A stationary point is only necessary for being the minimum

Choosing the right η is important

small η is too slow?

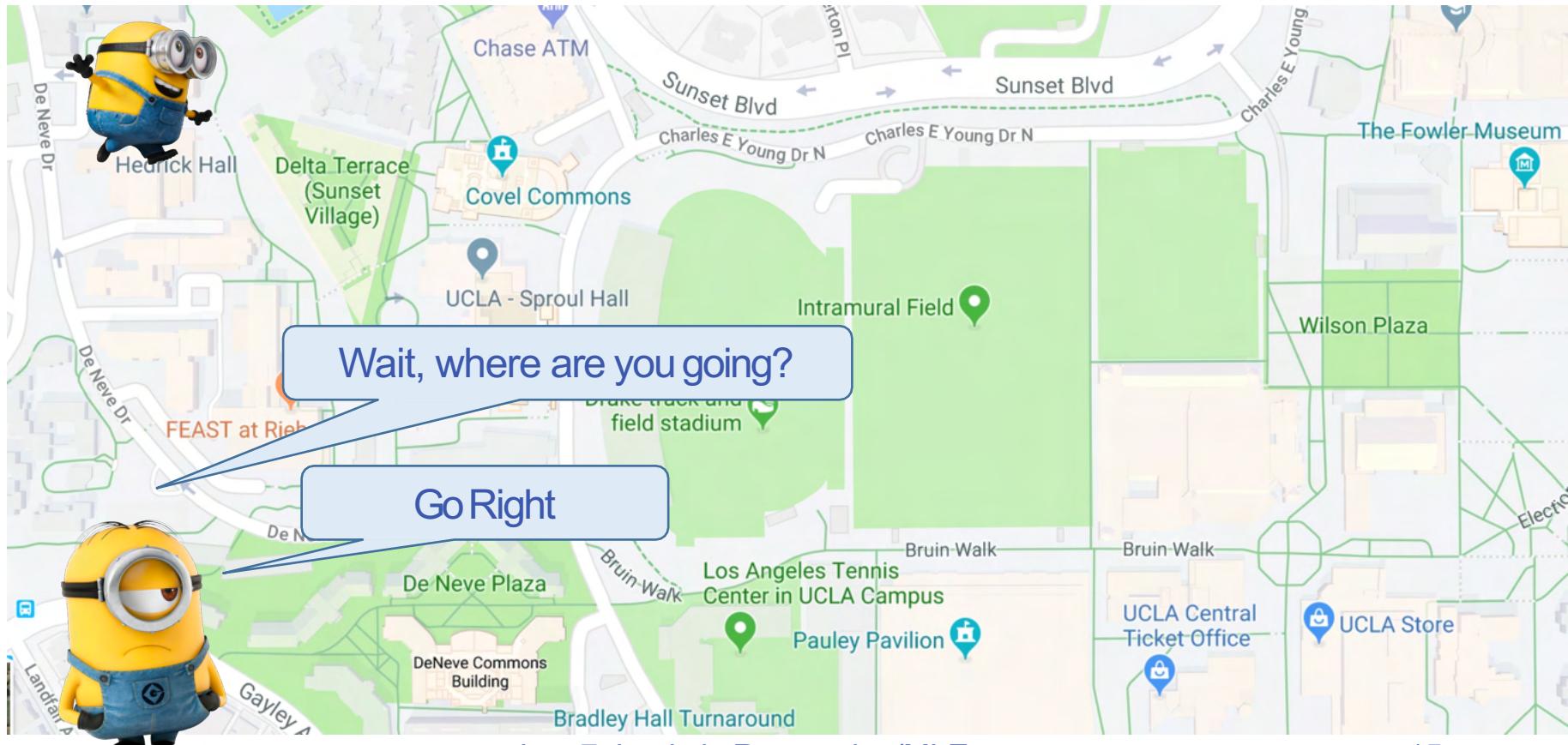


large η is too unstable?



Intuition

What may go wrong? Incorrect Step-size



Global vs Local Optimum

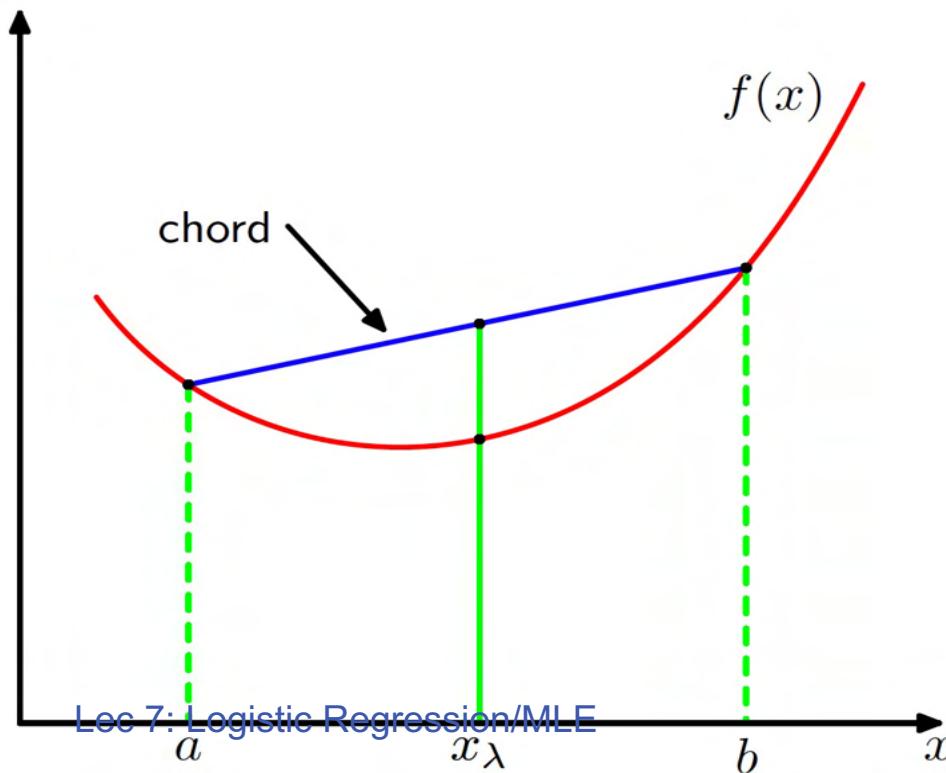
Convex function

A function $f(x)$ is convex if

$$f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b)$$

for

$$0 \leq \lambda \leq 1$$



How to determine convexity?

$f(x)$ is convex if

$$f''(x) \geq 0$$

Examples:

$$f(x) = x^2, f''(x) = 2 > 0$$

Exercise

Prove the following functions are convex

$$f(x) = ax + b$$

$$f(x) = x^2$$

$$f(x) = e^x$$

Multi-variate function

$f(\mathbf{x})$ is convex

$$f(\lambda \mathbf{a} + (1 - \lambda) \mathbf{b}) \leq \lambda f(\mathbf{a}) + (1 - \lambda) f(\mathbf{b})$$

for all \mathbf{a}, \mathbf{b} , $0 \leq \lambda \leq 1$

Multi-variate functions

How to determine convexity in this case?

Matrix of second-order derivatives (Hessian)

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_D} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_D} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_D} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_D^2} \end{pmatrix}$$

Multi-variate functions

How to determine convexity in this case?

If the Hessian is positive semi-definite $\mathbf{H} \succeq 0$, then f is convex.

A matrix \mathbf{H} is positive semi-definite if and only if

$$\mathbf{z}^T \mathbf{H} \mathbf{z} = \sum_{j,k} H_{j,k} z_j z_k \geq 0$$

for all \mathbf{z} .

Multi-variate functions

Example

$$f(\mathbf{x}) = x_1^2 + 2x_2^2$$

$$\mathbf{H} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

$$\mathbf{z}^T \mathbf{H} \mathbf{z} = 2z_1^2 + 4z_2^2 \geq 0$$

Stochastic Gradient Descent

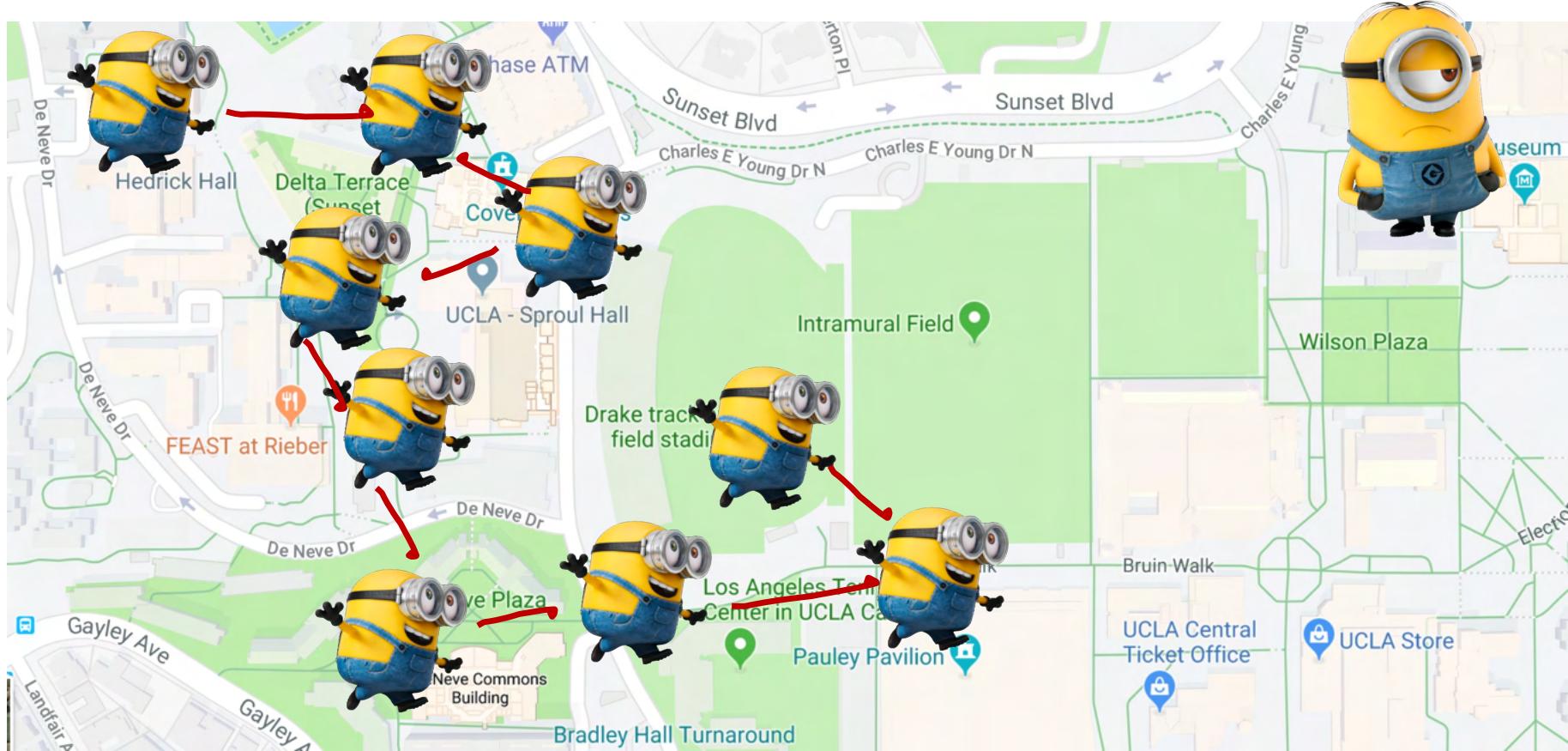
Intuition

Asking direction. Gradient descent:
compute gradient of all instances.



Intuition

Asking direction. Stochastic Gradient descent:
compute approximate gradient by one instance



Incremental/Stochastic gradient descent

Repeat for each example (\mathbf{x}_i, y_i)

Use this example to calculate the
gradient and update the model

Contrast with *batch gradient descent* which
makes one update to the weight vector for
every pass over the data

Stochastic gradient descent

If $f(w) = \frac{1}{|D|} \sum_i^{|D|} f_i(w)$

$$\min_w \sum_i^N \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$\nabla f(w) = \frac{1}{|D|} \sum_i \nabla f_i(w) = E_{i \sim D} \nabla f_i(w)$$

- ❖ Approximate the true gradient by a gradient at a single example at a time

Repeat until converge:

Randomly pick one sample (x_i, y_i)

Update $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla f_i(\mathbf{w})$

Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
 3. For (x, y) in \mathcal{D} :
 4. Update $w \leftarrow w - \eta \nabla f(w)$
 5. Return w

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
2. For epoch 1 ... T :
 3. For (x, y) in \mathcal{D} :
 4. if $y(w^\top x) < 0$
 5. $w \leftarrow w + \eta yx$
 6. Return w

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch 1 ... T :
3. For (x, y) in \mathcal{D} :
4. if $y(w^\top x) < 0$
5. $w \leftarrow w + \eta yx$
6. Return w

Prediction: y^{test}

Perceptron effectively minimizing:

$$\sum_i \max(0, 1 - y_i(w^\top x_i))$$

Linear Regression

Outline

- ❖ Least Squares Method for regression
 - ❖ Examples
 - ❖ The LMS objective
 - ❖ Gradient descent
 - ❖ Incremental/stochastic gradient descent

Learning Goals:

1. What is regression?
2. Examples of GD/SGD?
3. Data science practices

Regression

- Predicting a continuous outcome variable
- Predicting a company's future stock price using its profit and other financial info
- Predicting song year from audio features

Regression v.s. Classification

- We can measure 'closeness' of prediction and labels, leading to different ways to evaluate prediction errors.
 - E.g., predicting song year: better to be off by one year than by 20 years
- This will lead to different learning models and algorithms

Application– House Price Prediction

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview>

<https://www.kaggle.com/paultimothymooney/zillow-house-price-data>

<https://www.kaggle.com/c/zillow-prize-1/>

**111 Archer Ave,
New York, NY 10031**
4 beds • 3 baths • 3,410 sqft

Built in 2009, perfectly blending elegance with functional living space. Excellent floor plan with 3 beds up and 1 on main. Open living, kitchen & dining w/ huge fireplace & Sound views. Spacious kitchen w/ slab granite surfaces & center island. Huge master suite with Jacuzzi tub & separate shower. Features; hdwd floors, all

● FOR SALE
\$1,175,000
Zestimate®: \$1,275,448

EST. MORTGAGE
\$4,461/mo

CONTACT
Your name
Phone
Email
I am interested in NY 10031.
 I want f

Feature can be used

3620 South BUDLONG
Los Angeles, CA 90007
Status: Closed

\$1,510,000 | **14** Beds | **6** Baths | **4,418** Sq. Ft.
Last Sold Price | Built: 1956 | Lot Size: 9,649 Sq. Ft. | Sold On: Jul 26, 2013

Overview Property Details Tour Insights Property History Public Records Activity Schools



1 of 12 

Five unit apartment complex within 2 blocks of USC campus, Gate #6. Great for students (most student leases have parents as guarantors). Most USC students live off campus, so housing units like this are always fully leased. Situated on a gated, corner lot, and across from an elementary school, this complex was recently renovated, and has in-unit laundry hook ups, wall-unit AC, and 12 parking spaces. It is within a DPS (Department of Public Safety) and Campus Cruiser patrolled area. This is a great income generating property, not to be missed!

Property Type: Multi-Family | Style: Two Level, Low Rise
Community: Downtown Los Angeles | County: Los Angeles
MLS#: 22176741

Property Details for 3620 South BUDLONG, Los Angeles, CA 90007

Details provided by i-Tech MLS and may not match the public record. [Learn More](#)

Interior Features

Kitchen Information

- Remodeled
- Oven, Range

Laundry Information

- Inside Laundry

Heating & Cooling

- Wall Cooling Unit(s)

Multi-Unit Information

Community Features

- Units in Complex (Total): 5

Multi-Family Information

- # Leased: 5
- # of Buildings: 1
- Owner Pays Water
- Tenant Pays Electricity, Tenant Pays Gas

Unit 2 Information

- # of Beds: 3
- # of Baths: 1
- Unfurnished
- Monthly Rent: \$2,250

- Monthly Rent: \$2,350

Unit 3 Information

- Unfurnished

Unit 5 Information

- # of Beds: 3
- # of Baths: 2
- Unfurnished
- Monthly Rent: \$2,325

Unit 4 Information

- # of Beds: 3
- # of Baths: 1
- Unfurnished

Unit 6 Information

- # of Beds: 3
- # of Baths: 1
- Monthly Rent: \$2,250

Property / Lot Details

Property Features

- Automatic Gate, Card/Code Access

- Automatic Gate, Lawn, Sidewalks
- Corner Lot, Near Public Transit

- Tax Parcel Number: 5040017019

Lot Information

- Lot Size (Sq. Ft.): 9,649
- Lot Size (Acres): 0.2215
- Lot Size Source: Public Records

Property Information

- Updated/Remodeled
- Square Footage Source: Public Records

Parking / Garage, Exterior Features, Utilities & Financing

Parking Information

- # of Parking Spaces (Total): 12
- Parking Space
- Gated

Utility Information

- Green Certification Rating: 0.00
- Green Location: Transportation, Walkability
- Green Walk Score: 0
- Green Year Certified: 0

Financial Information

- Capitalization Rate (%): 6.25
- Actual Annual Gross Rent: \$128,331
- Gross Rent Multiplier: 11.29

Location Details, Misc. Information & Listing Information

Location Information

- Cross Streets: W 36th Pl

Expense Information

- Operating: \$37,664

Listing Information

- Listing Terms: Cash, Cash To Existing Loan
- Buyer Financing: Cash

See analysis at:

<https://www.kaggle.com/sudalairajkumar/simple-exploration-notebook-zillow-prize>

How to learn a regression model?

Training data (past sales record)

sqft	sale price
2000	800K
2100	907K
1100	312K
5500	2,600K
...	...

What is our model?

Price_per_sqft

- ❖ Sale Price = square_footage $\times w_1 + w_b$

Unexplainable _stuff

How to measure prediction error

The classification error (hit or miss) is not appropriate for continuous outcomes

How should we evaluate quality of a prediction?

- ▶ *absolute* difference: $| \text{prediction} - \text{sale price}|$
- ▶ *squared* difference: $(\text{prediction} - \text{sale price})^2$

sqft	sale price	prediction	error	squared error
2000	810K	720K	90K	90^2
2100	907K	800K	107K	107^2
1100	312K	350K	-38K	38^2
5500	2,600K	2,600K	0	0
...	...			

Minimize squared errors

- ❖ Our model:

Sale Price = square_footage $\times w_1 + w_b$

- ❖ Learning: Using the training data to find the *best* possible value of w

- ❖ Prediction: Predict the sale price given the square_footage

Minimize squared errors

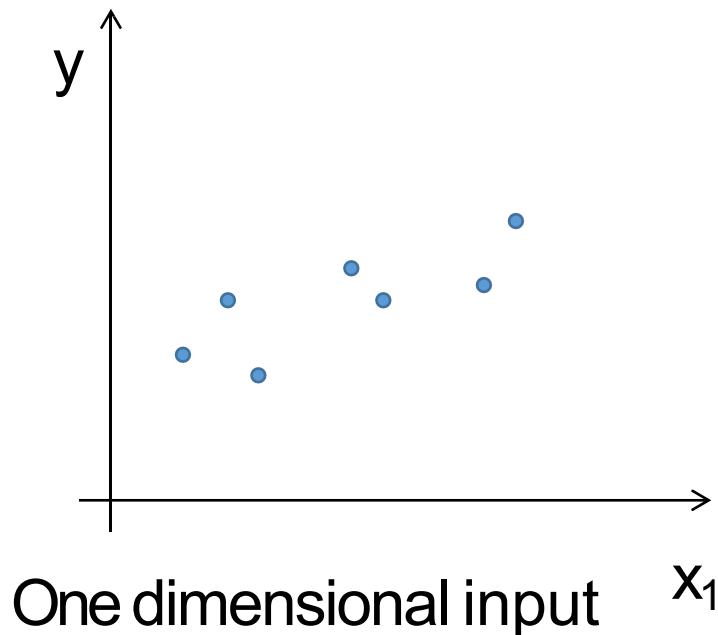
sqft	sale price	prediction	error	squared error
2000	810K	720K	90K	90^2
2100	907K	800K	107K	107^2
1100	312K	350K	38K	38^2
5500	2,600K	2,600K	0	0
...	...			
Total				$90^2 + 107^2 + 38^2 + 0 + \dots$

- ❖ Our model:
 $\text{Sale Price} = \text{square_footage} \times w_1 + w_b$
- ❖ Goal of training: adjust w_1, w_b such that the sum of the squared error is minimized

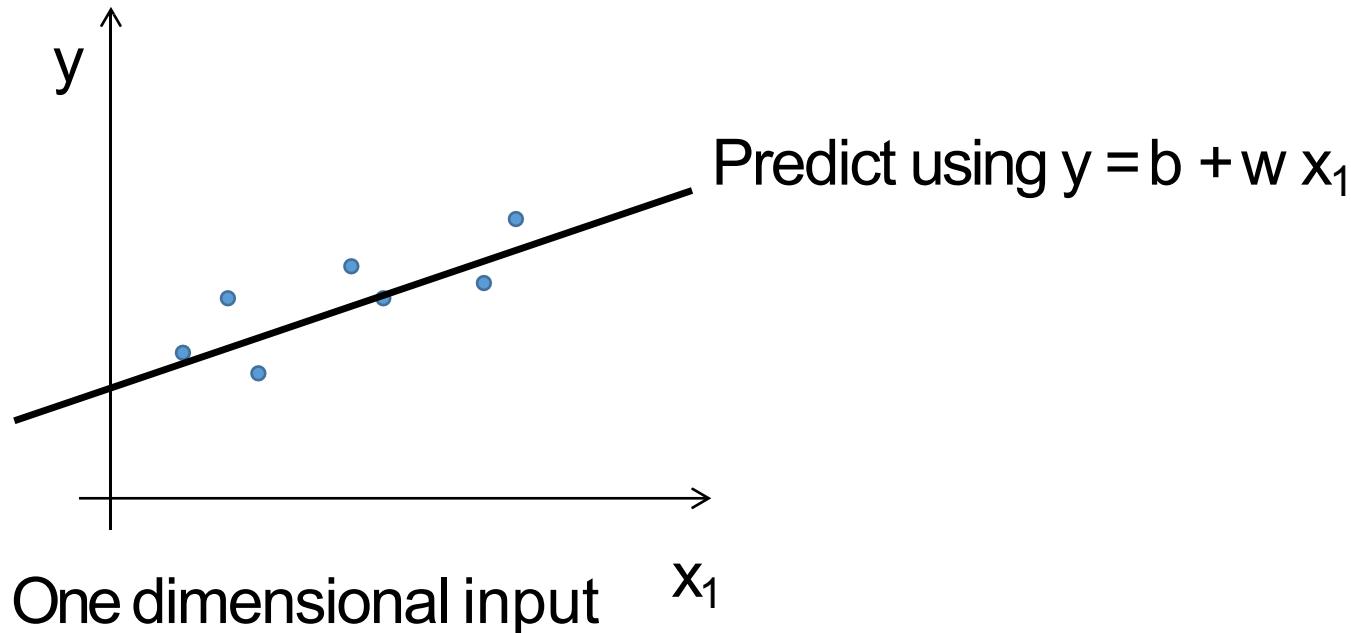
Setup – Linear Regression

- Input: $\mathbf{x} \in \mathbb{R}^D$ (covariates, predictors, features, etc)
- Output: $y \in \mathbb{R}$ (responses, targets, outcomes, outputs, etc)
- Hypotheses/Model: $h_{\mathbf{w}, b}$, with $h_{\mathbf{w}, b}(\mathbf{x}) = b + \sum_d w_d x_d = b + \mathbf{w}^T \mathbf{x}$
 $\mathbf{w} = [w_1 \ w_2 \ \cdots \ w_D]^T$: *weights*
 b is called the **bias or offset or intercept term**.
 $\boldsymbol{\theta} = [b \ w_1 \ w_2 \ \cdots \ w_D]^T$
- Training data: $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, 2, \dots, N\}$

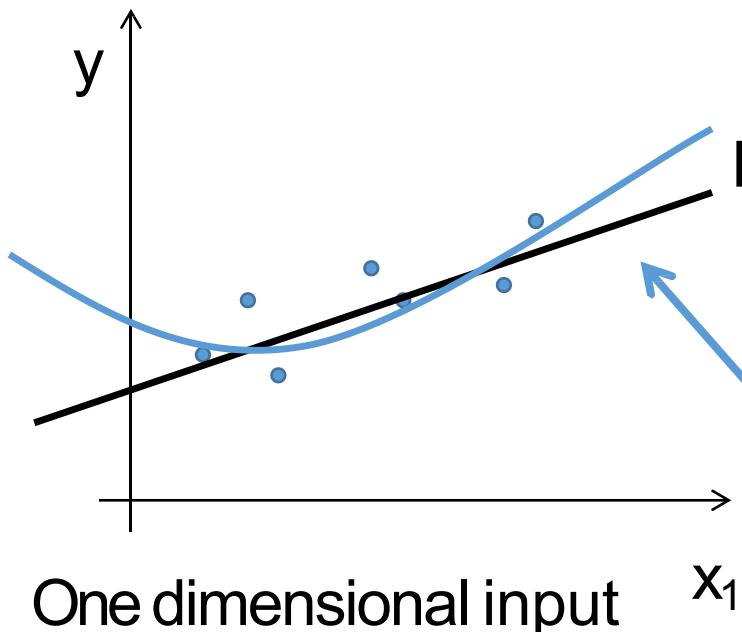
Examples



Examples

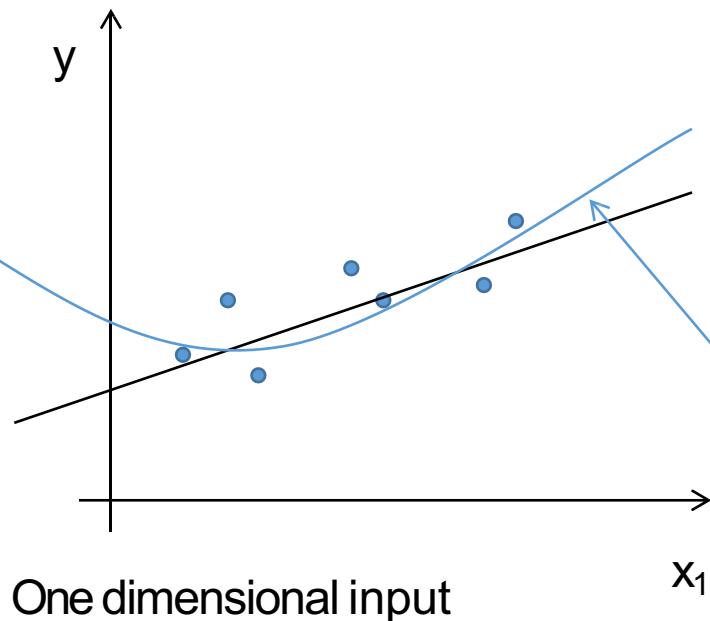


Examples

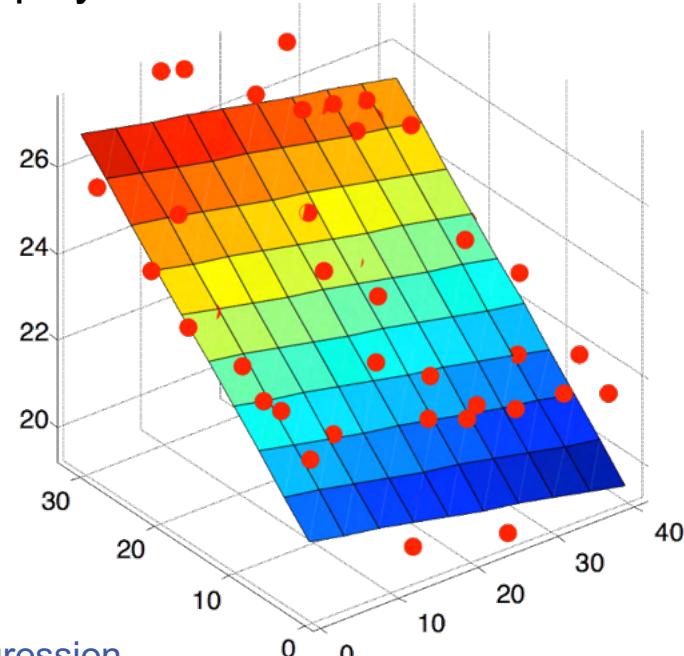


The linear function is not our only choice. We could have tried to fit the data as another polynomial

Examples



Two dimensional input
Predict using $y = w_1x_1 + w_2x_2 + b$



What is the best weight vector?

Question: How do we know which weight vector is the *best* one for a training set?

For an input (\mathbf{x}_i, y_i) in the training set, the **cost** of a mistake is

$$|y_i - \mathbf{w}^T \mathbf{x}_i|$$

Define the cost (or **loss**) for a particular weight vector \mathbf{w} to be

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Sum of squared costs over the training set

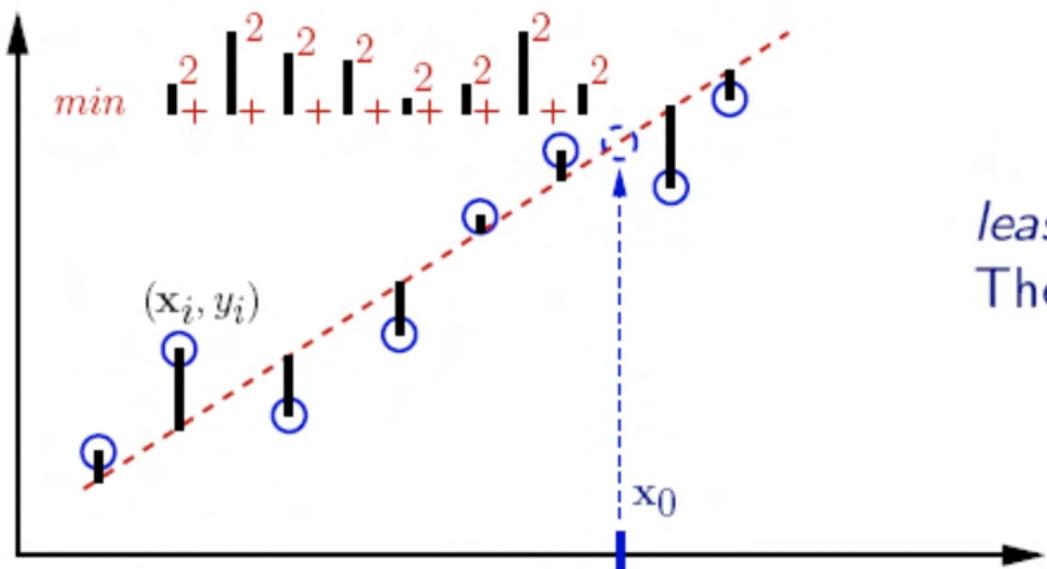
One strategy for learning: *Find the \mathbf{w} with least cost on this data*

Least Mean Squares (LMS) Regression

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Learning: minimizing mean squared error

Example



least squares (LSQ)
The fitted line is used as a predictor

Least Mean Squares (LMS) Regression

Learning: minimizing mean squared error

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Some people write this:

$$\min_{\mathbf{w}} \frac{1}{2m} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Different strategies exist for *learning by optimization*

Gradient descent is a popular algorithm

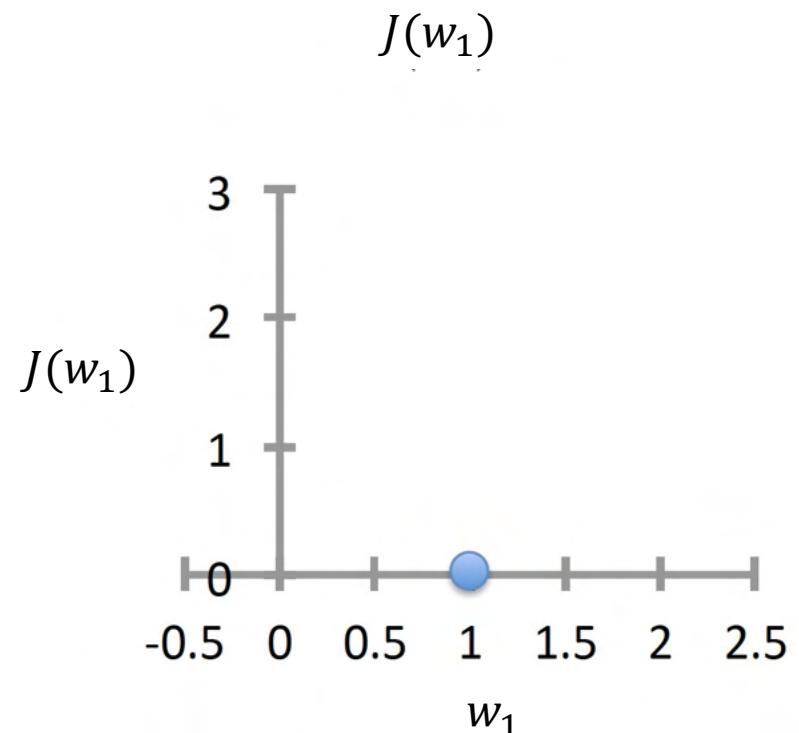
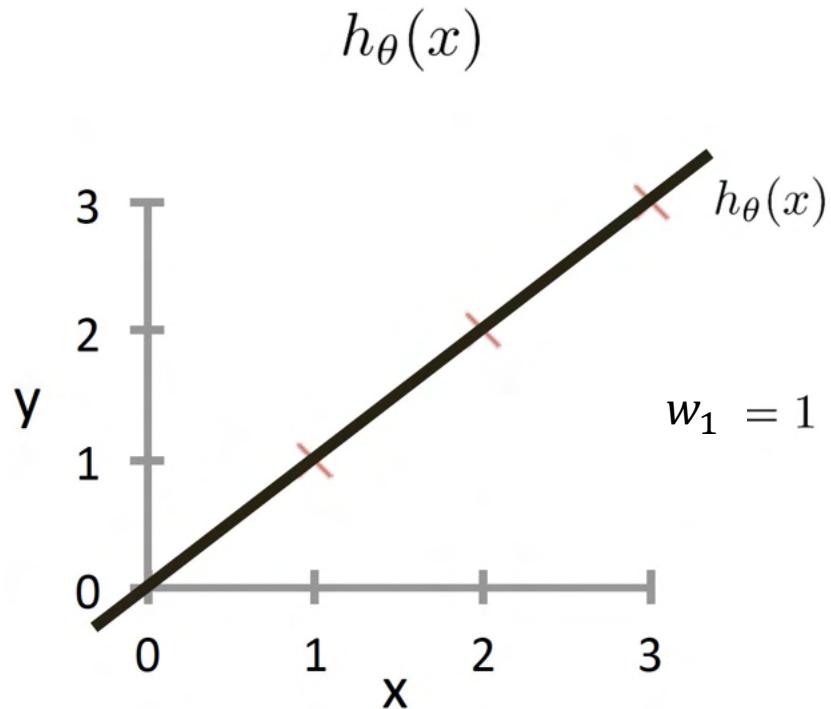
(For this minimization objective, there is also an analytical solution)

Intuition behind the cost function

Assume $x \in \mathbb{R}$

$$h_{\theta}(x) = w_1 x$$

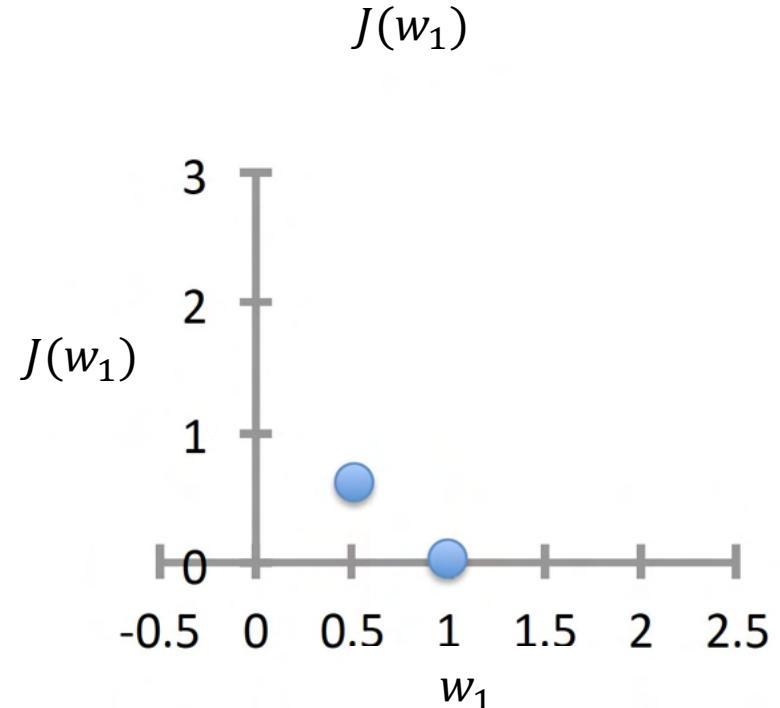
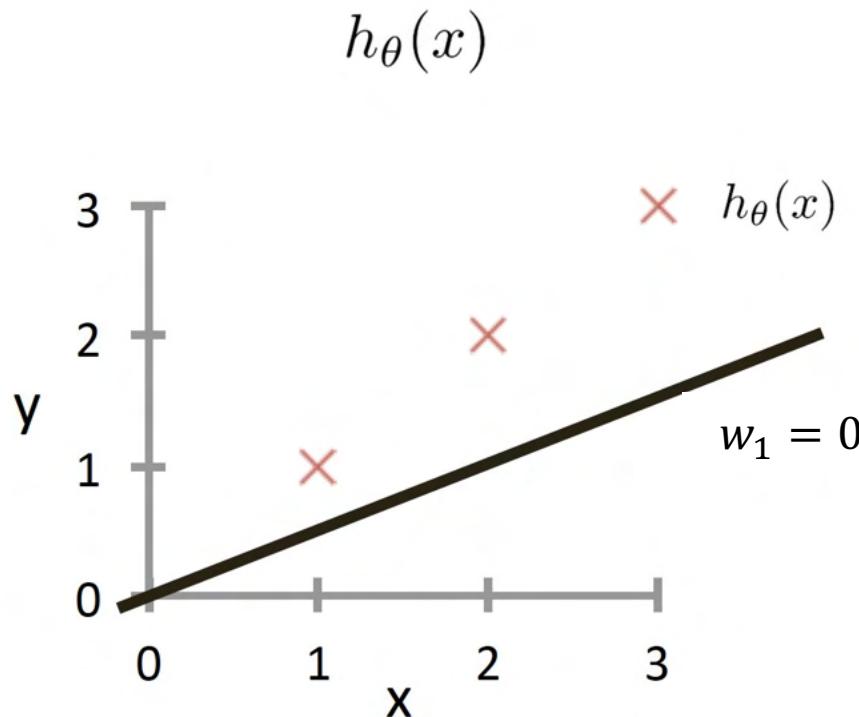
$$J(w_1) = \frac{1}{2m} \sum_{i=1}^m (y_i - h_{\theta}(x_i))^2$$



Intuition behind the cost function

Assume $x \in \mathbb{R}$,
 $h_{\theta}(x) = w_1 x$

$$J(w_1) = \frac{1}{2m} \sum_{i=1}^m (y_i - h_{\theta}(x_i))^2$$

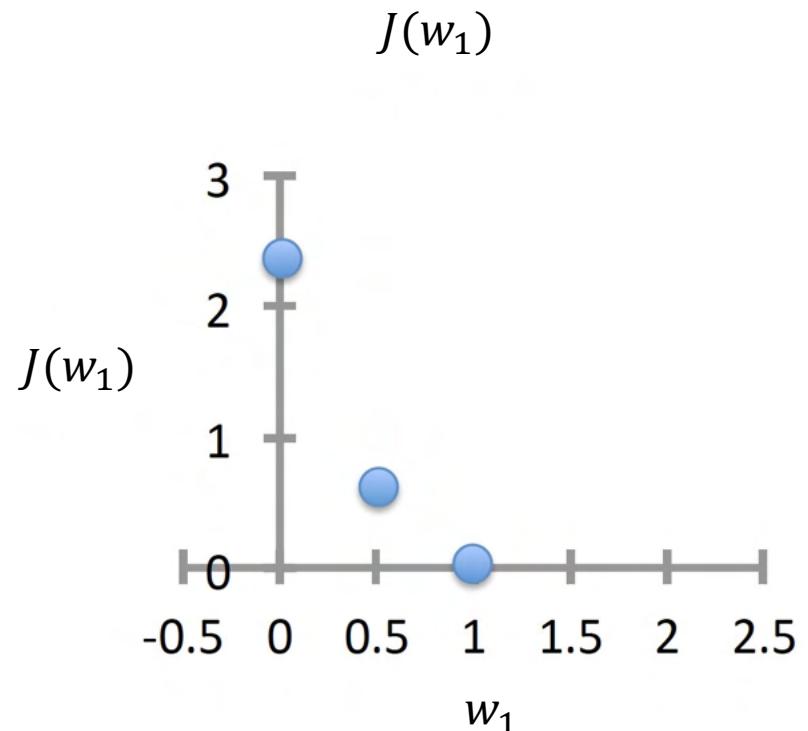
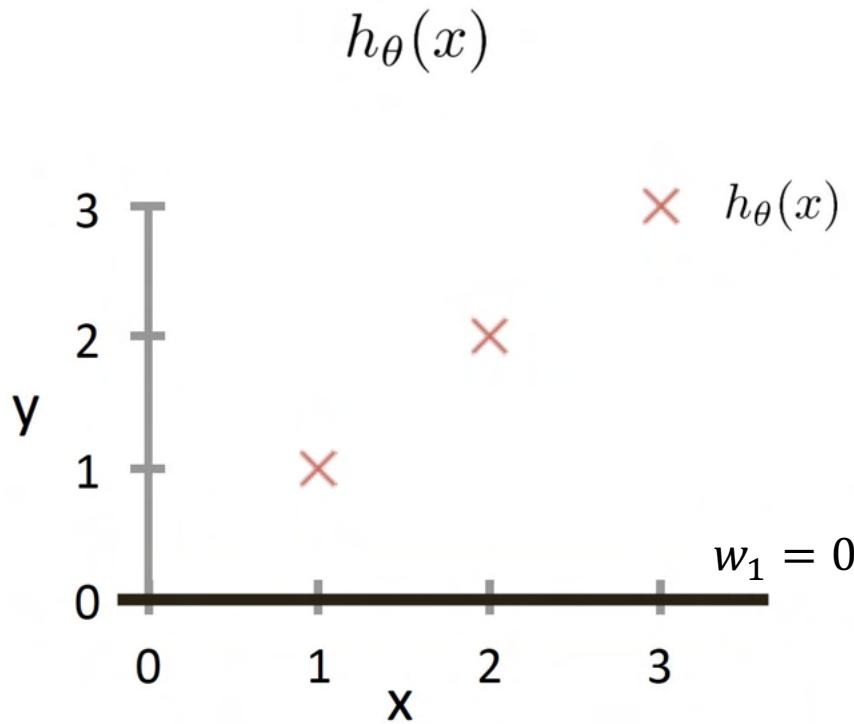


Intuition behind the cost function

Assume $x \in \mathbb{R}$,

$$h_{\theta}(x) = w_1 x_1$$

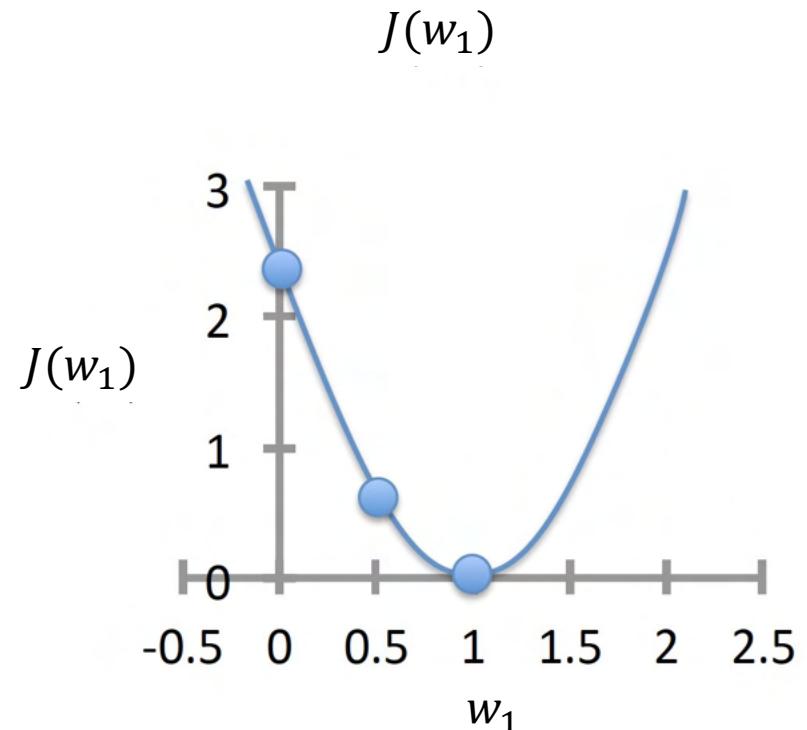
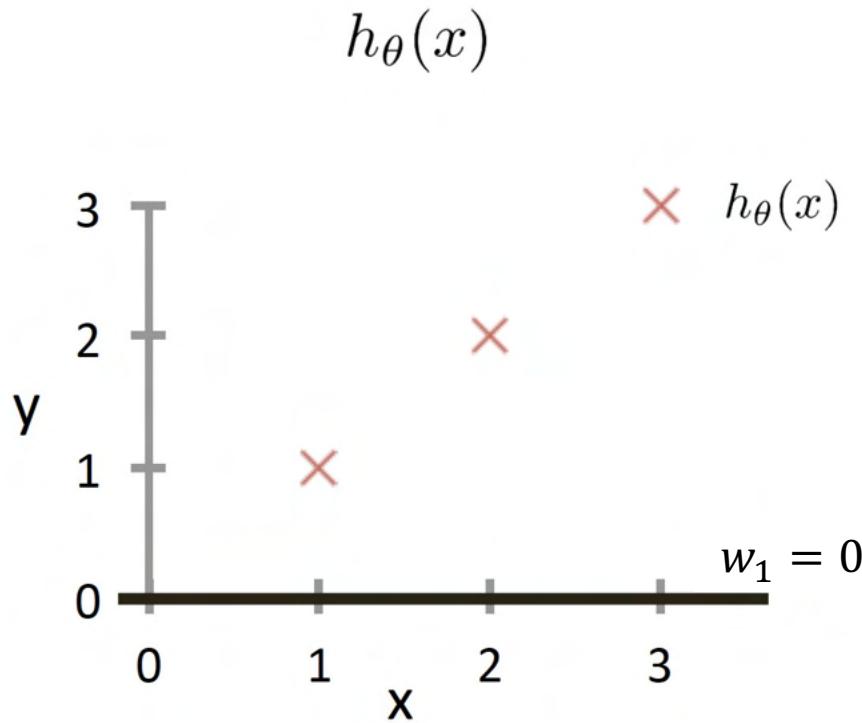
$$J(w_1) = \frac{1}{2m} \sum_{i=1}^m (y_i - h_{\theta}(x_i))^2$$



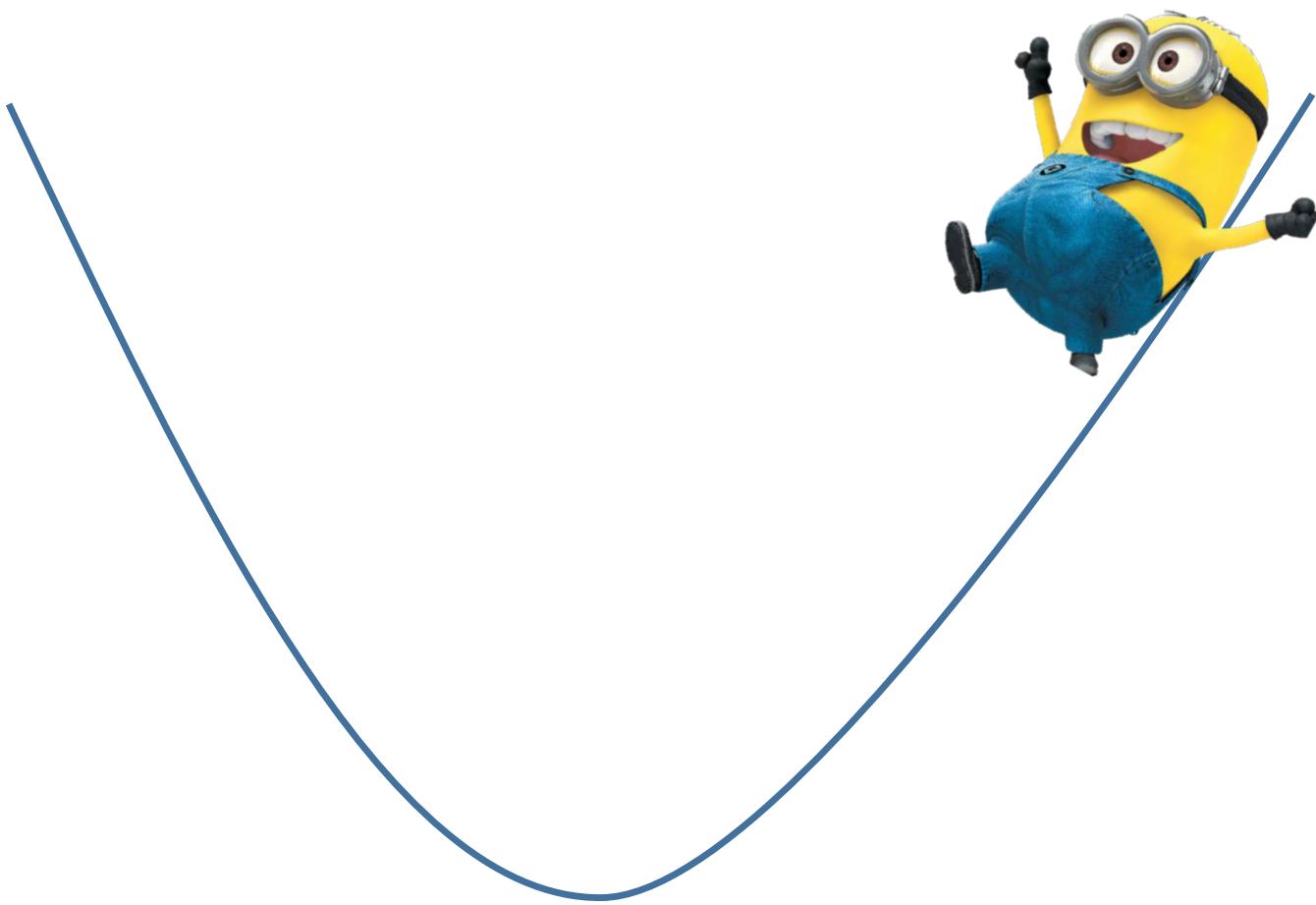
Intuition behind the cost function

Assume $x \in \mathbb{R}$,
 $h_{\theta}(x) = w_1 x_1$

$$J(w_1) = \frac{1}{2m} \sum_{i=1}^m (y_i - h_{\theta}(x_i))^2$$



Gradient Descent For LMS



Gradient Descent

Algorithm 1 Gradient Descent (J)

- 1: $t \leftarrow 0$
 - 2: Initialize $\theta^{(0)}$
 - 3: **repeat**
 - 4: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla J(\theta^{(t)})$
 - 5: $t \leftarrow t + 1$
 - 6: **until** convergence
 - 7: Return final value of θ
-

Need to compute the gradient for the negative log likelihood

Compute the gradient of $J(\theta)$

- ❖ What is our objective function?

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- ❖ Gradient of $J(\mathbf{w})$ is:

$$\nabla J(\mathbf{w}^t) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \dots, \frac{\partial J}{\partial w_d} \right]$$

Gradient of the cost

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

✓ The gradient is of the form

$$\nabla J(\mathbf{w}^t) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \dots, \frac{\partial J}{\partial w_d} \right]$$

$$\begin{aligned}\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\ &= \frac{1}{2} \sum_{i=1}^m \frac{\partial}{\partial w_j} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\ &= \frac{1}{2} \sum_{i=1}^m 2(y_i - \mathbf{w}^T \mathbf{x}_i) \frac{\partial}{\partial w_j} (y_i - w_1 x_{i1} - \dots - w_j x_{ij} - \dots) \\ &= \frac{1}{2} \sum_{i=1}^m 2(y_i - \mathbf{w}^T \mathbf{x}_i)(-x_{ij}) \\ &= -\sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)x_{ij}\end{aligned}$$

Sum of Error input

One element of the gradient vector

We are trying to minimize

Gradient descent for LMS

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

1. Initialize \mathbf{w}^0
2. For $t = 0, 1, 2, \dots$ (*until total error is below a threshold*)
 1. Compute gradient of $J(\mathbf{w})$ at \mathbf{w}^T . Call it $\nabla J(\mathbf{w}^t)$

Evaluate the function for **each** training example to compute the error and construct the gradient vector

$$\frac{\partial J}{\partial w_j} = - \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i) x_{ij}$$

One element of
 $\nabla J(\mathbf{w}^t)$



2. Update \mathbf{w} as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - r \nabla J(\mathbf{w}^t)$$

r : Called the **learning rate**

(For now, a small constant. We will get to this later)

We are trying to minimize

Gradient descent for LMS

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

1. Initialize \mathbf{w}^0

The weight vector is not updated until **all** errors are calculated

2.

Why not make early updates to the weight vector as soon as we encounter errors instead of waiting for a full pass over the data?

compute the error and construct the gradient vector

$$\frac{\partial J}{\partial w_j} = - \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i) x_{ij}$$

One element of
 $\nabla J(\mathbf{w}^t)$



2. Update \mathbf{w} as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - r \nabla J(\mathbf{w}^t)$$

r : Called the **learning rate**

(For now, a small constant. We will get to this later)

Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \dots T$:
 3. For (x, y) in \mathcal{D} :
 4. Update $w \leftarrow w - \eta \nabla f(w)$
 5. Return w

Stochastic gradient descent for LMS

1. Initialize \mathbf{w}
2. For $t = 0, 1, 2, \dots$ (until error below some threshold)

For each training example (\mathbf{x}_i, y_i) :

Update \mathbf{w} . For each element of the weight vector (w_j):

$$w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T \mathbf{x}_i)x_{ij}$$

Incremental/Stochastic gradient descent

1. Initialize \mathbf{w}
2. For $t = 0, 1, 2, \dots$ (until error below some threshold)
 - ▀ For each training example (\mathbf{x}_i, y_i) :
 - ▀ Update \mathbf{w} . For each element of the weight vector (w_j):

$$w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T \mathbf{x}_i)x_{ij}$$

Contrast with the previous method, where the weights are updated only after all examples are processed once

Online/Incremental algorithms are often preferred when the training set is very large

Learning Rates and Convergence Remarks

- In the general (non-separable) case the learning rate η must decrease to approach zero to guarantee convergence
- More sophisticated algorithms choose η automatically and converge faster
- Choosing a better starting point can also have impact

Least Squares in Matrix Form

- ❖ Given a dataset $D = \{x_i, y_i\}_{i=1}^m, x_i \in R^d, y_i \in R$

$$X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \mathbf{x}_m \end{bmatrix}_{d \times m} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m \times 1}$$

- ❖ We can rewrite $J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$ into a matrix format

$$\min_{\mathbf{w}} (X^T \mathbf{w} - Y)^T (X^T \mathbf{w} - Y)$$

Least Squares in Matrix Form

$$X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \mathbf{x}_m \end{bmatrix}_{d \times m} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m \times 1}$$

We can rewrite $J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w}^T \mathbf{x}_i)^2$ into a matrix format

$$\min_{\mathbf{w}} (X^T \mathbf{w} - Y)^T (X^T \mathbf{w} - Y)$$

Analytic Solution for LMS

$$\min_{\mathbf{w}} (\mathbf{X}^T \mathbf{w} - \mathbf{Y})^T (\mathbf{X}^T \mathbf{w} - \mathbf{Y})$$

- ❖ This can be solved analytically, and the solution w^* is

$$\mathbf{w}^* = (\mathbf{X} \mathbf{X}^T)^{-1} \mathbf{X} \mathbf{Y}$$

- ❖ A useful resource for matrix operations:
<https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>

Probabilistic interpretation – LMS is MLE

(advanced topic: not covered in the exam)

- Noisy observation model

$$Y = \theta_0 + \theta_1 X + \eta$$

where $\eta \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable

- Likelihood of one training sample (x_n, y_n)

$$p(y_n|x_n; \boldsymbol{\theta}) = \mathcal{N}(\theta_0 + \theta_1 x_n, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2}}$$

[Advanced topic] **Probabilistic interpretation**

Log-likelihood of the training data \mathcal{D} (assuming i.i.d)

$$\mathcal{LL}(\boldsymbol{\theta}) = \log P(\mathcal{D})$$

$$= \log \prod_{n=1}^N p(y_n|x_n) = \sum_n \log p(y_n|x_n)$$

$$= \sum_n \left\{ -\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\}$$

$$= -\frac{1}{2\sigma^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 - \frac{N}{2} \log \sigma^2 - N \log \sqrt{2\pi}$$

$$= -\frac{1}{2} \left\{ \frac{1}{\sigma^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 + N \log \sigma^2 \right\} + \text{const}$$

What is the relationship between minimizing J and maximizing the log-likelihood?

[Advanced topic] **Probabilistic interpretation**

Estimating σ , θ_0 and θ_1 can be done in two steps

- Maximize over θ_0 and θ_1

$$\max \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 \leftarrow \text{That is } J(\boldsymbol{\theta})!$$

- Maximize over $s = \sigma^2$ (we could estimate σ directly)

$$\begin{aligned} \frac{\partial \log P(\mathcal{D})}{\partial s} &= -\frac{1}{2} \left\{ -\frac{1}{s^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 + N \frac{1}{s} \right\} = 0 \\ \rightarrow \sigma^{*2} &= s^* = \frac{1}{N} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 \end{aligned}$$

Linear regression: Summary

What we want: Predict a real valued output using a feature representation of the input

Assumption: Output is a linear function of the inputs

Learning by minimizing total cost

- Gradient descent and stochastic gradient descent to find the *best* weight vector

Lecture 9: Neural Network & Deep Learning

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Checkpoint: The bigger picture

- ❖ Supervised learning: instances, concepts, and hypotheses

- ❖ Specific learners

- ❖ Decision trees

- ❖ K-NN

- ❖ Perceptron

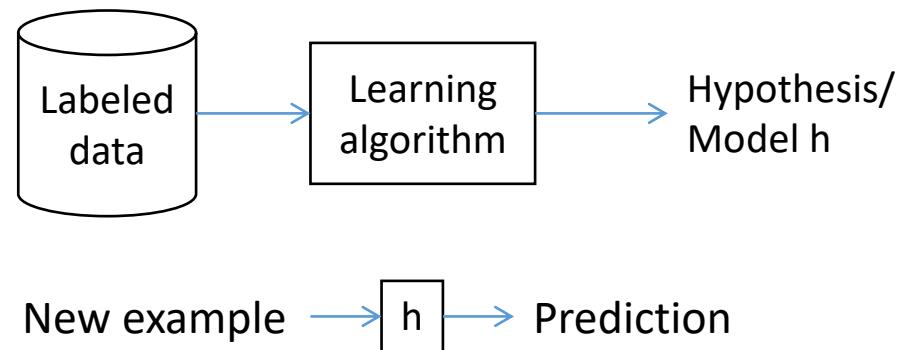
- ❖ Logistic regression

- ❖ General ML ideas

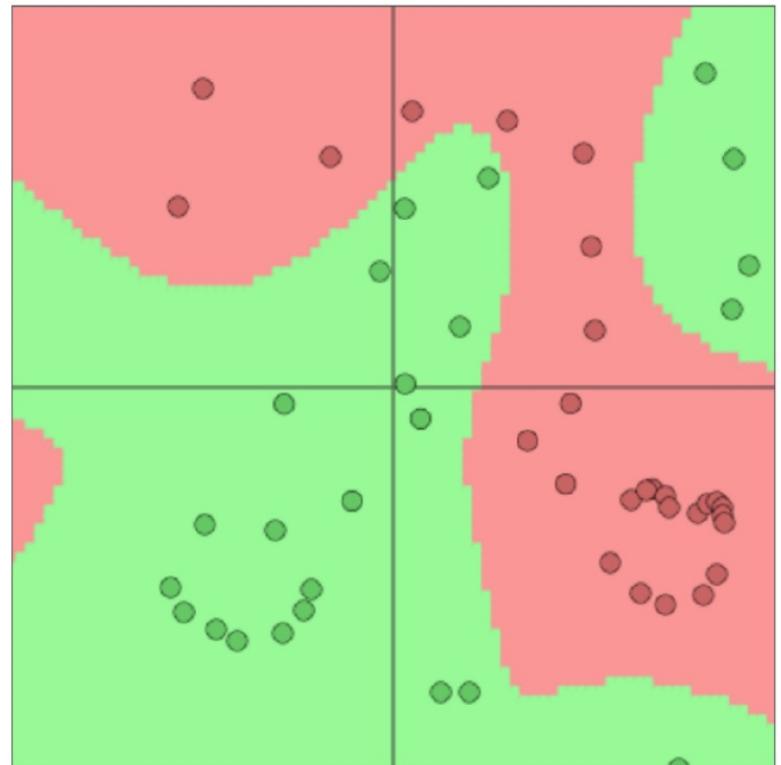
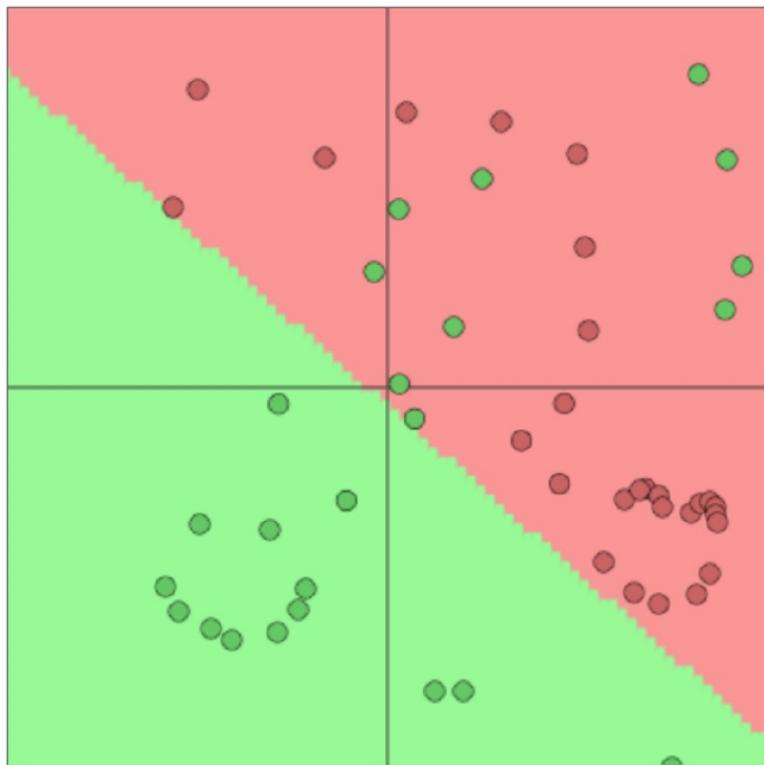
- ❖ Feature vectors

- ❖ Overfitting

- ❖ Probabilistic model

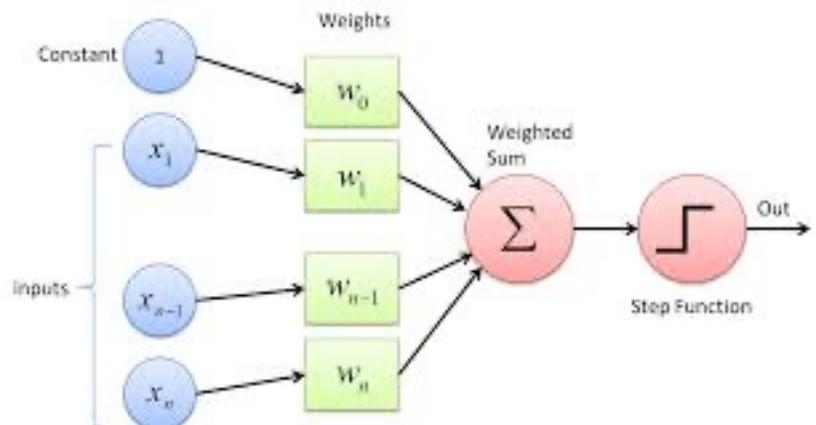
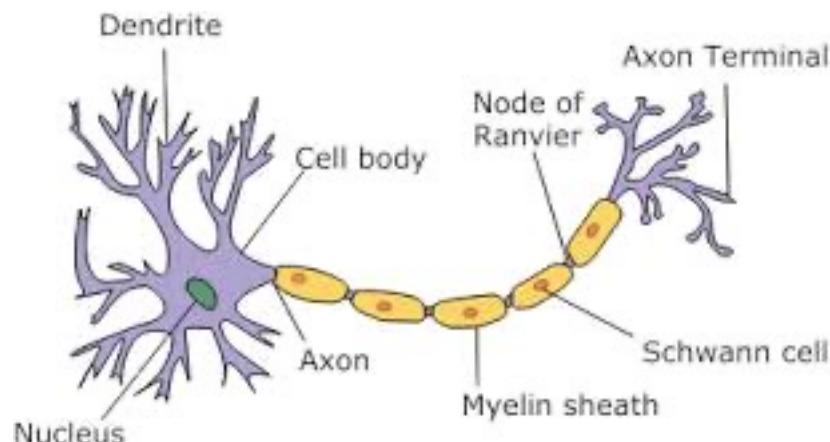


Non-Linear Decision Boundary

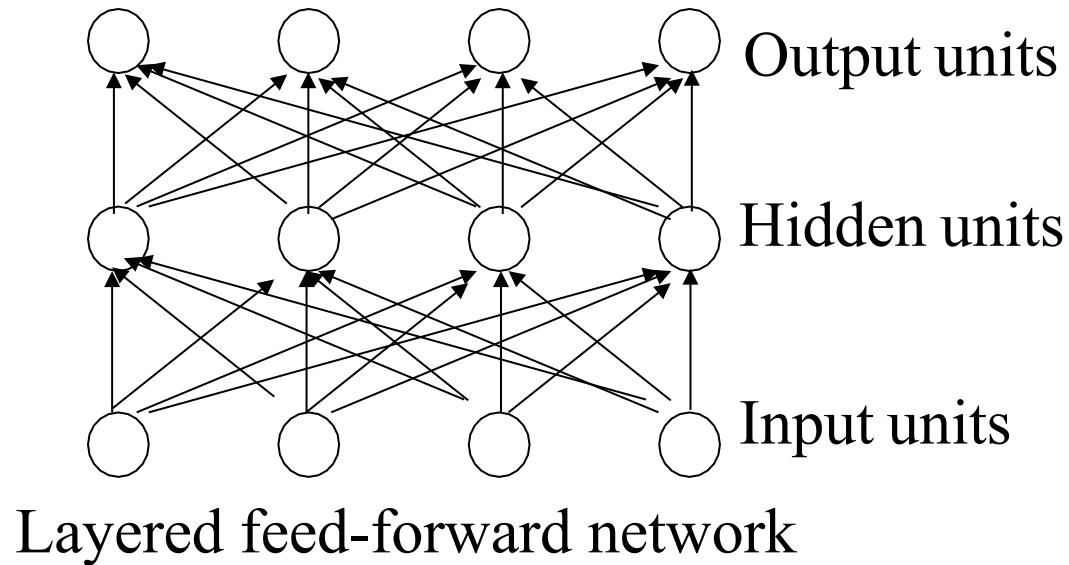


Neural Networks

- Design to mimic the brain.
- Artificial neural networks are not nearly as complex or intricate as the actual brain structure



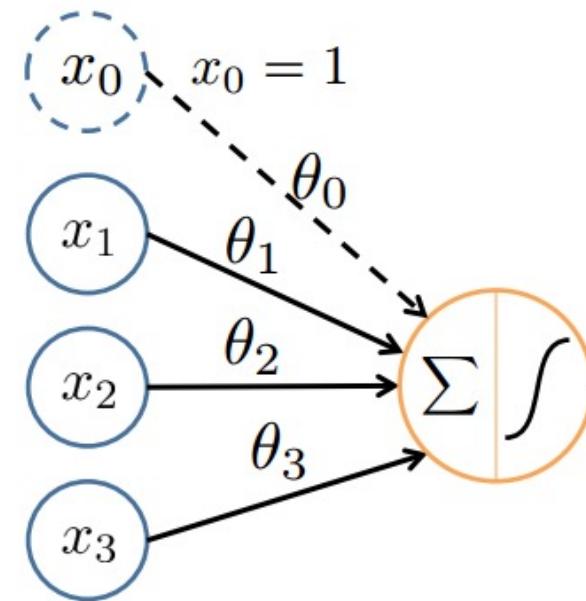
Neural networks



- Neural networks are made up of **nodes** or **units**, connected by **links**
- Each link has an associated **weight** and **activation level**
- Each node has an **input function** (typically summing over weighted inputs), an **activation function**, and an **output**

Neuron Model Example: Logistic Unit

“bias unit”

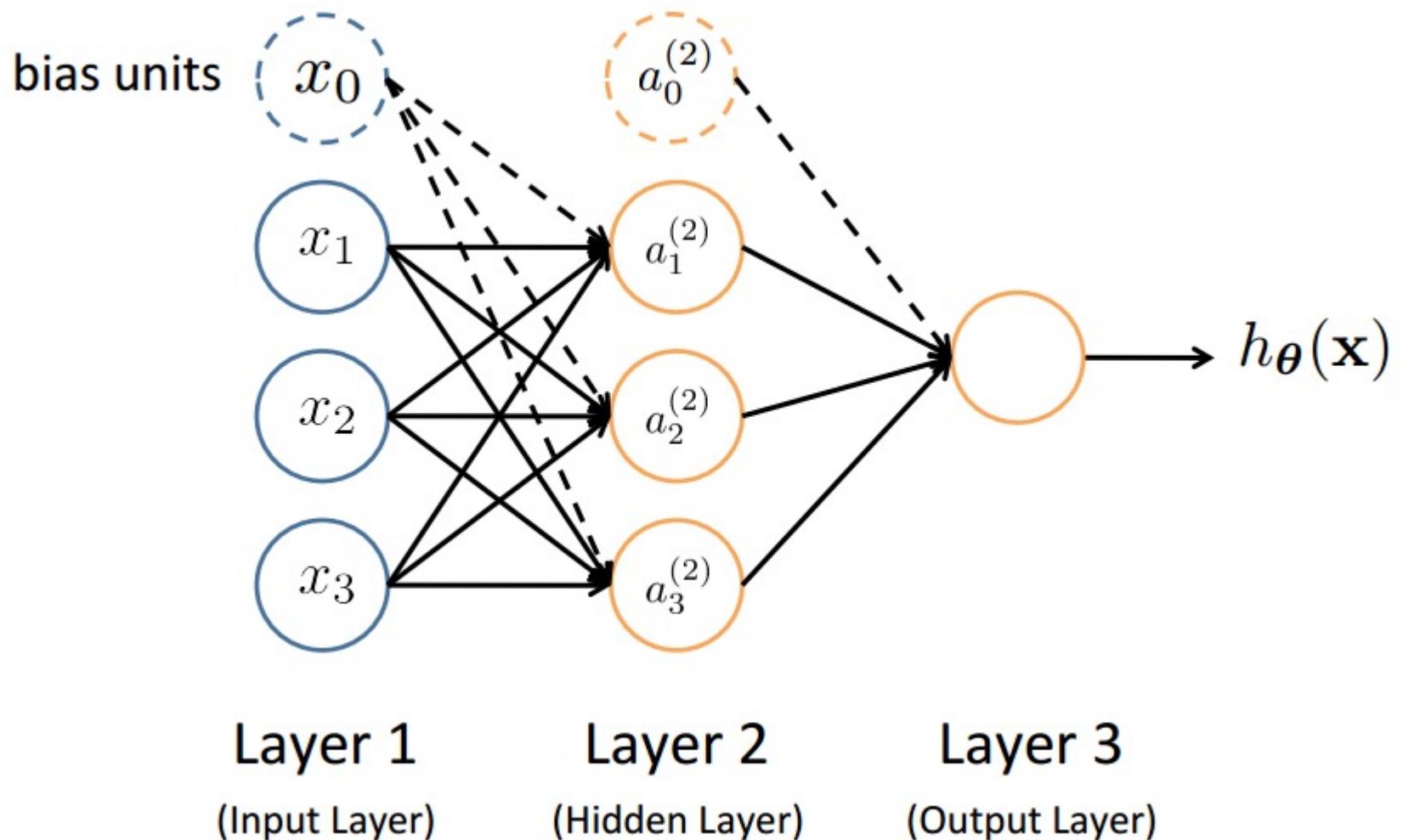


$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) \\ = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

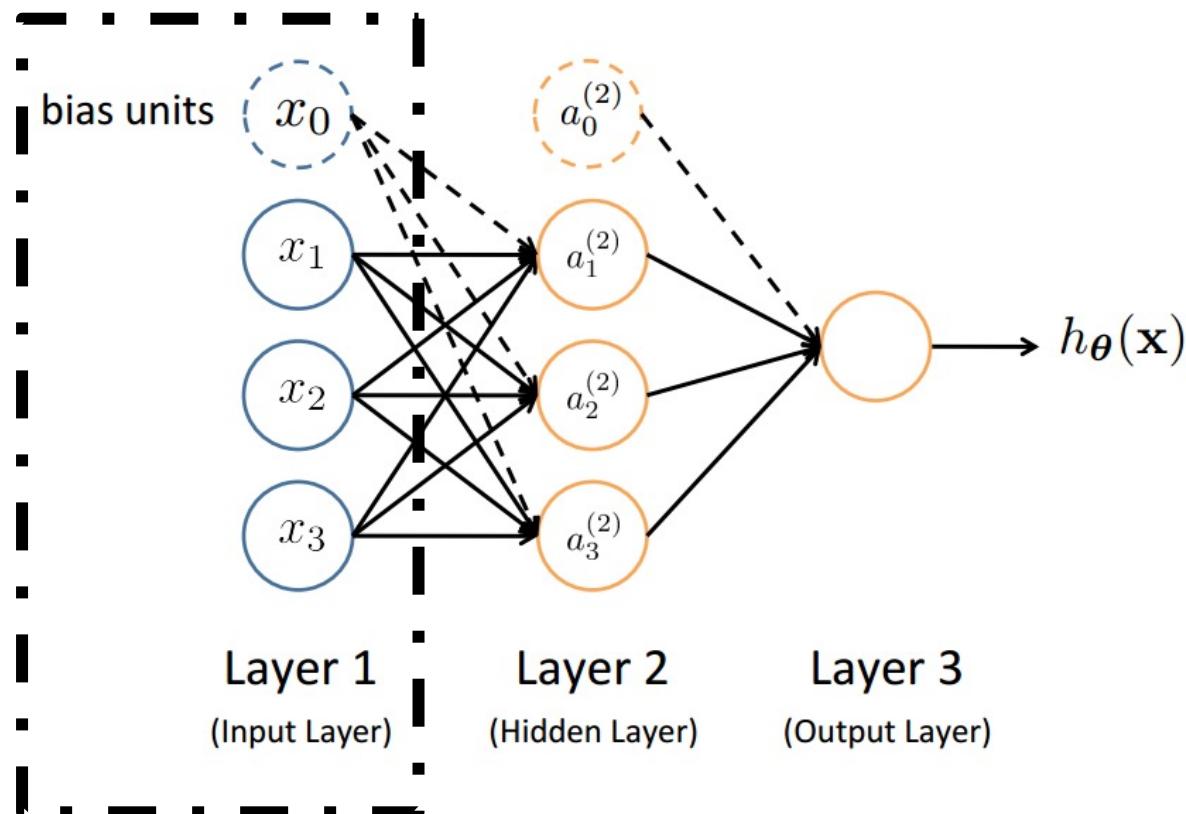
Sigmoid (logistic) activation function: $g(z) = \frac{1}{1 + e^{-z}}$

Neural Network



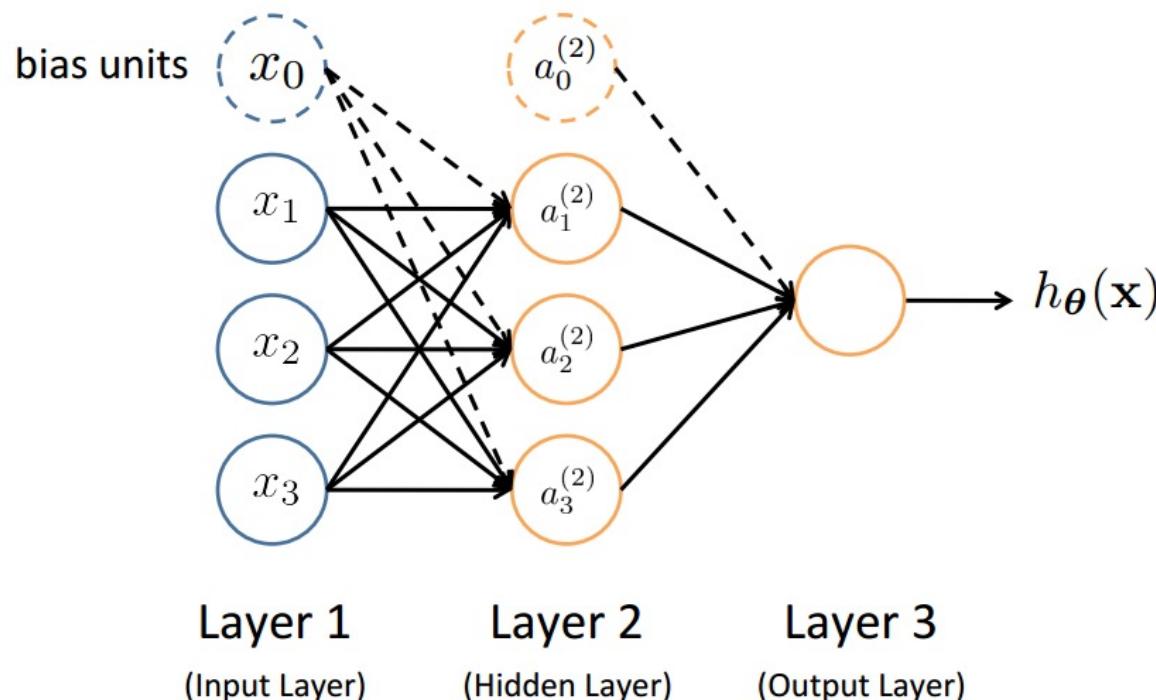
Feed-Forward Process

- Input layer units are set by some exterior function (think of these as **sensors**), which causes their output links to be **activated** at the specified level

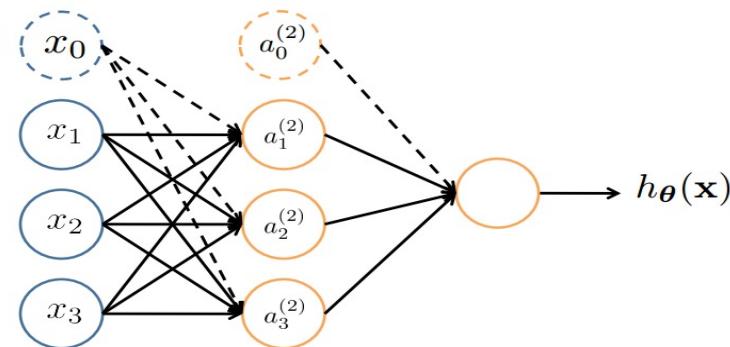


Feed-Forward Process

- Working forward through the network, the **input function** of each unit is applied to compute the input value
- The **activation function** transforms this input function into a final value



Neural Network



$a_i^{(j)}$ = “activation” of unit i in layer j

$\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

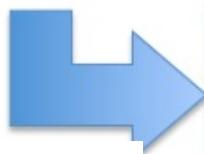
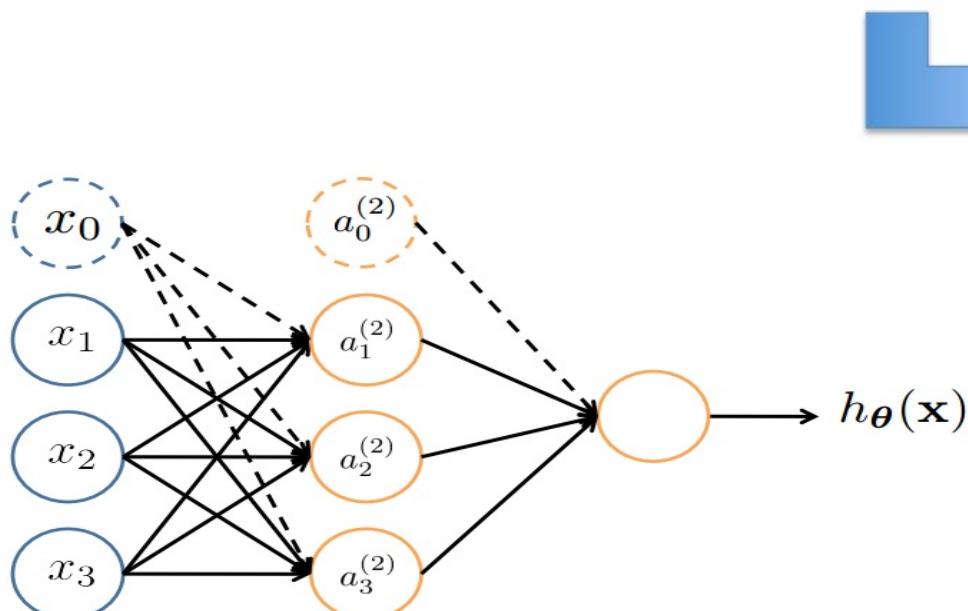
Vectorization

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g \left(z_1^{(2)} \right)$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g \left(z_2^{(2)} \right)$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g \left(z_3^{(2)} \right)$$

$$h_{\Theta}(\mathbf{x}) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g \left(z_1^{(3)} \right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

$$\text{Add } a_0^{(2)} = 1$$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

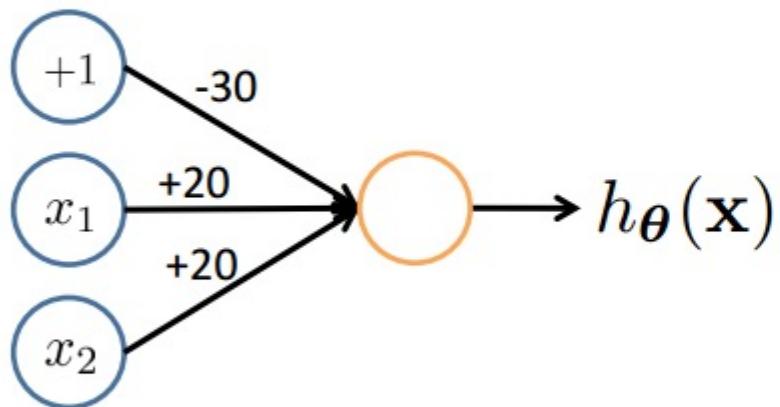
$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

Understanding Representations

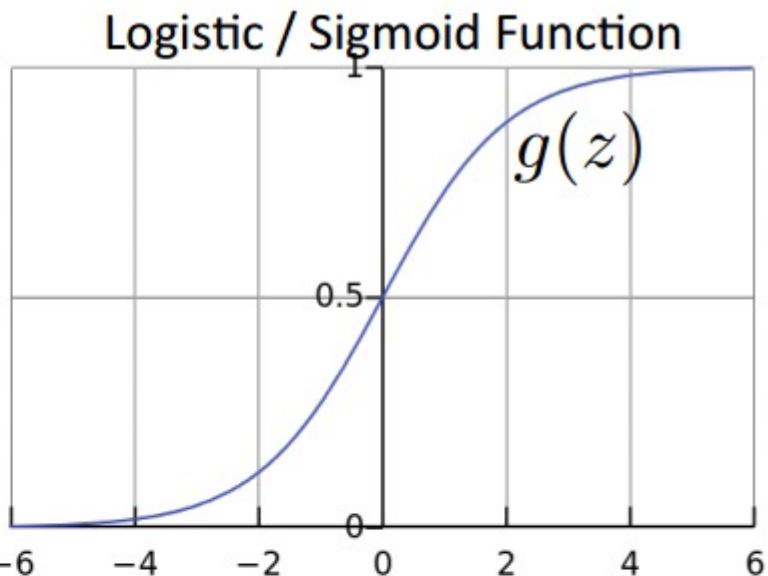
Simple example: AND

$$x_1, x_2 \in \{0, 1\}$$

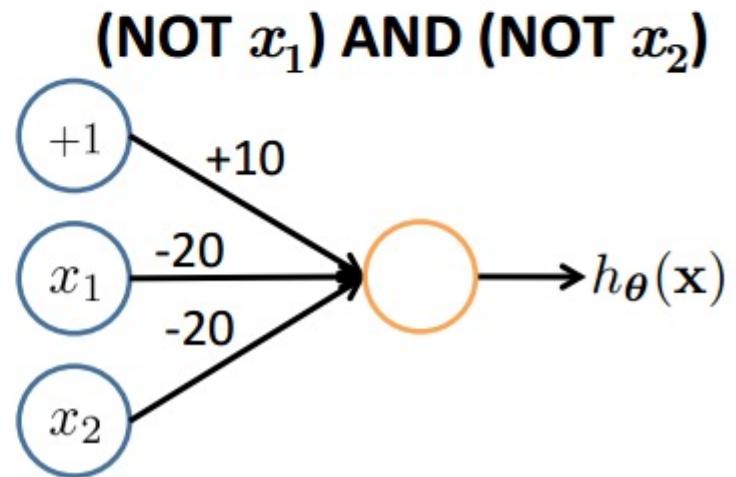
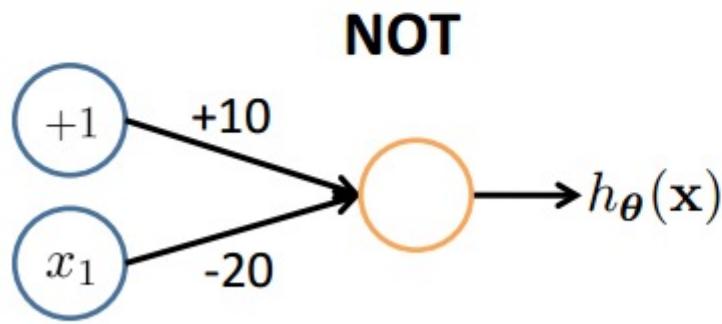
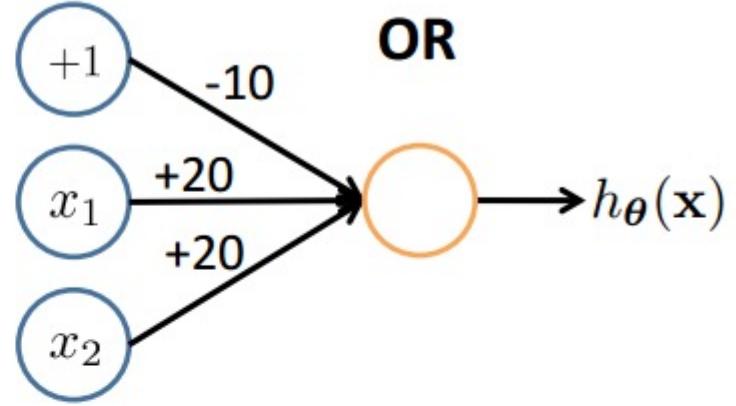
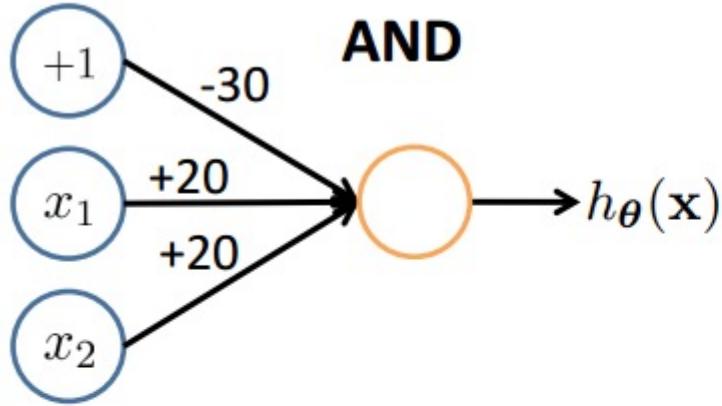
$$y = x_1 \text{ AND } x_2$$

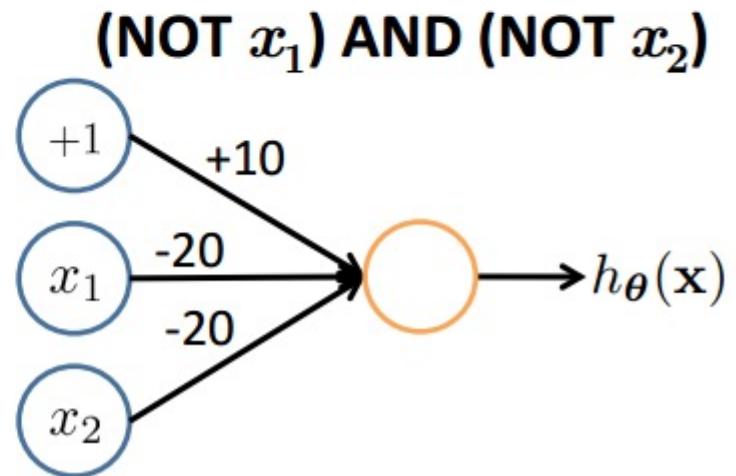
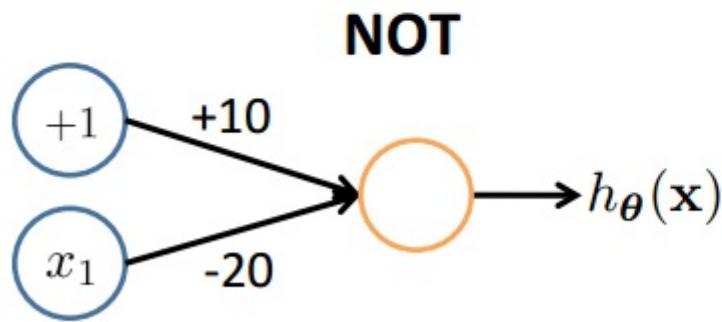
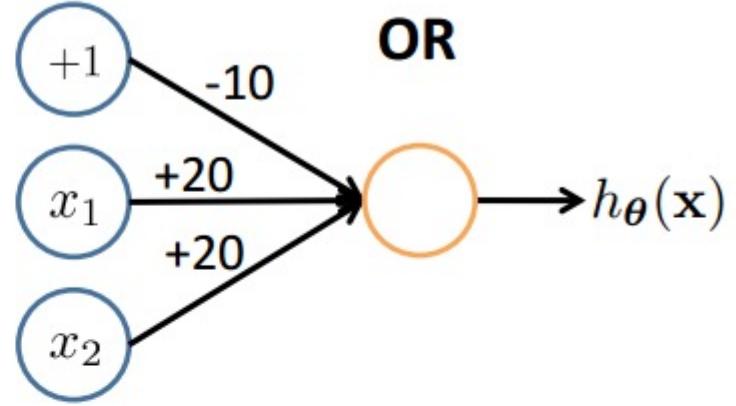
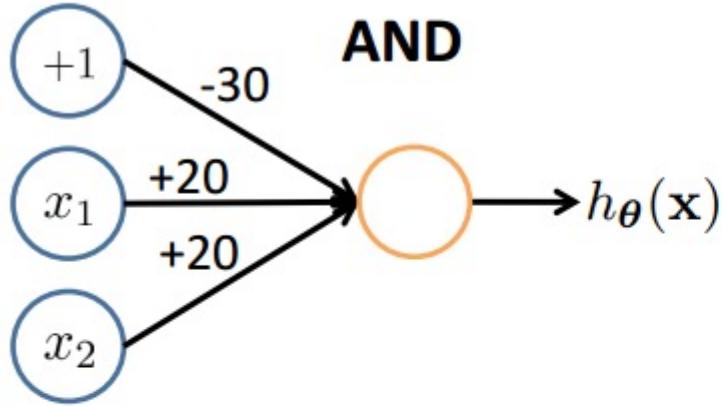


$$h_{\theta}(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

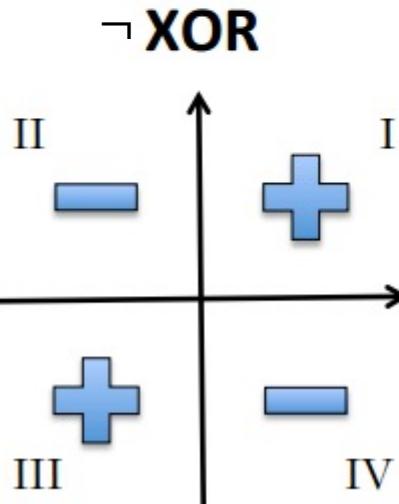
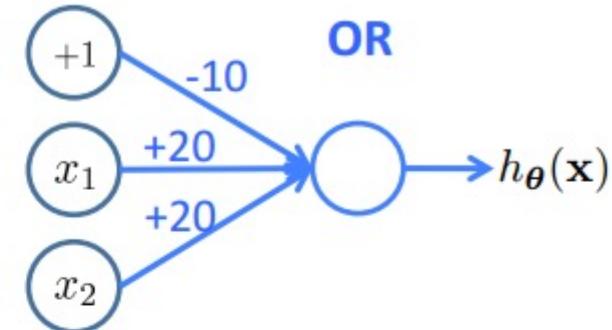
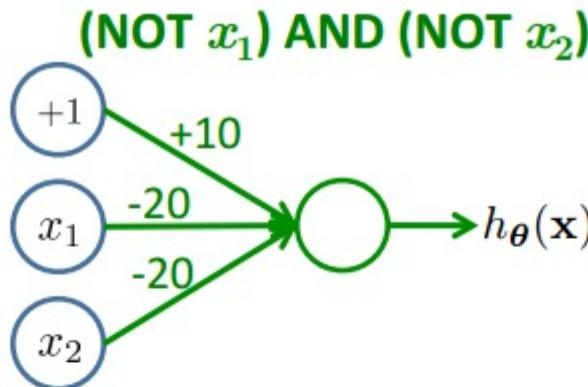
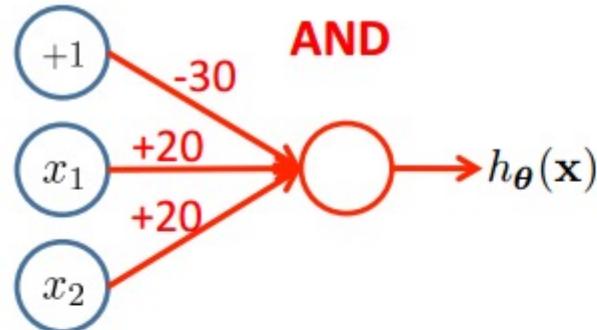


x_1	x_2	$h_{\theta}(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$



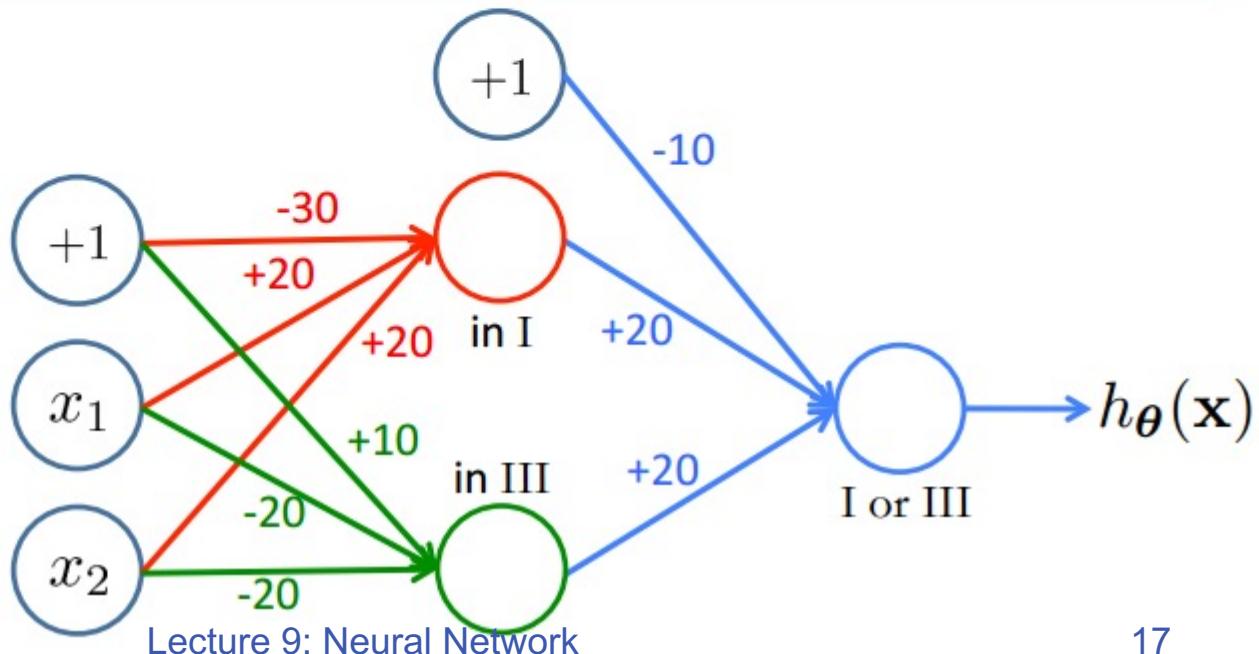
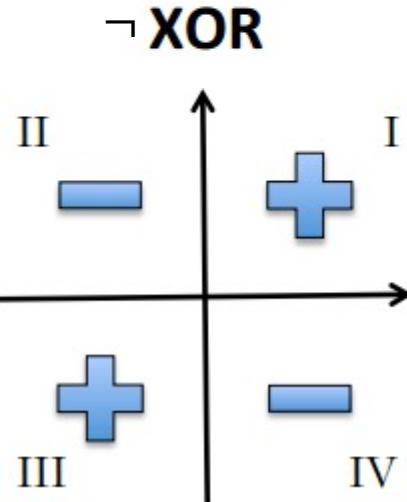
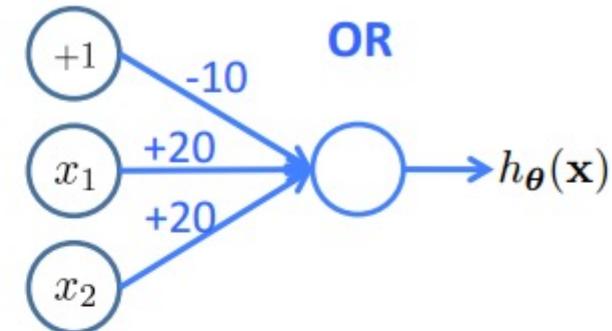
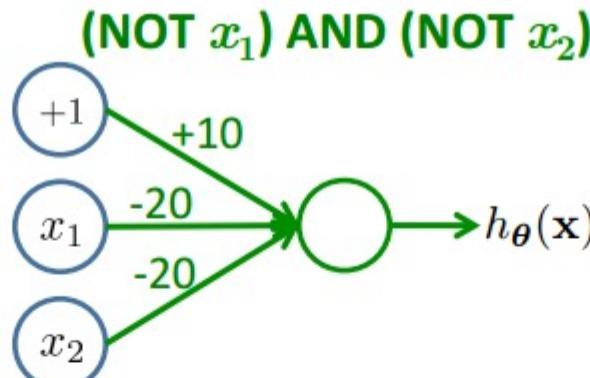
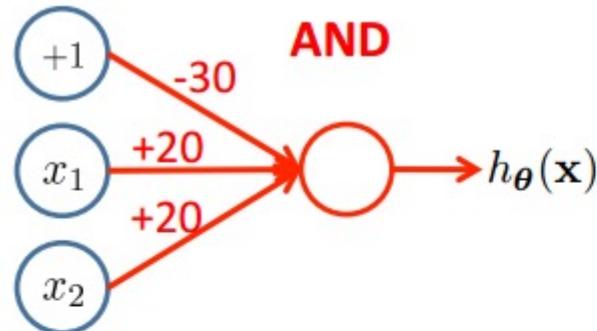


Combining Representations to Create Non-Linear Functions

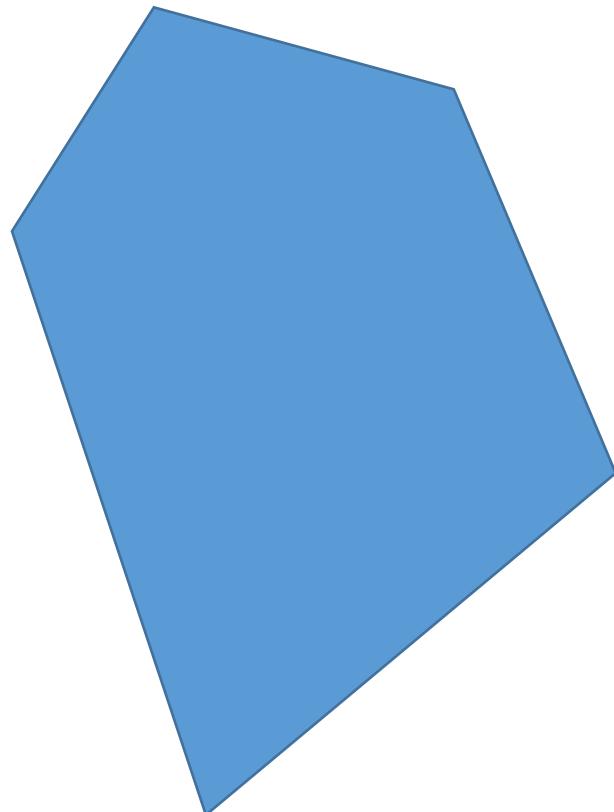


XNOR

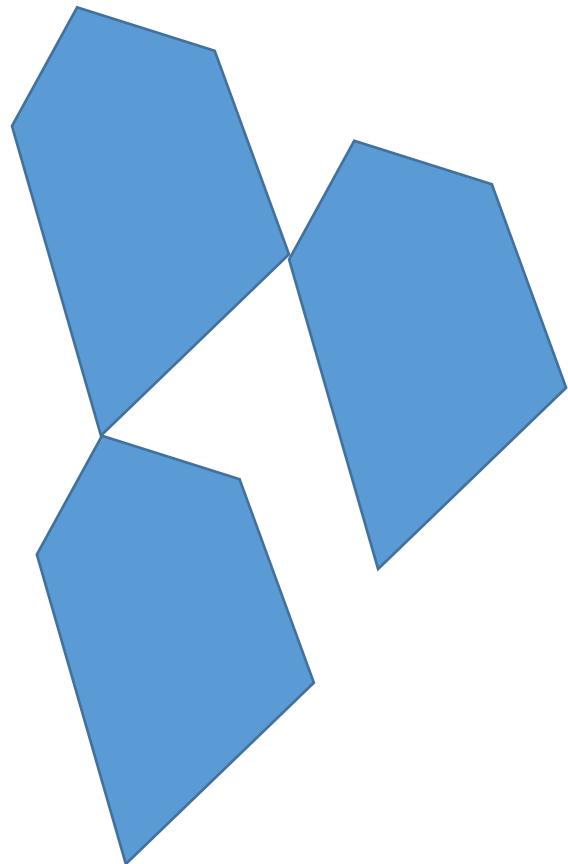
Combining Representations to Create Non-Linear Functions



Arbitrary Decision Boundary

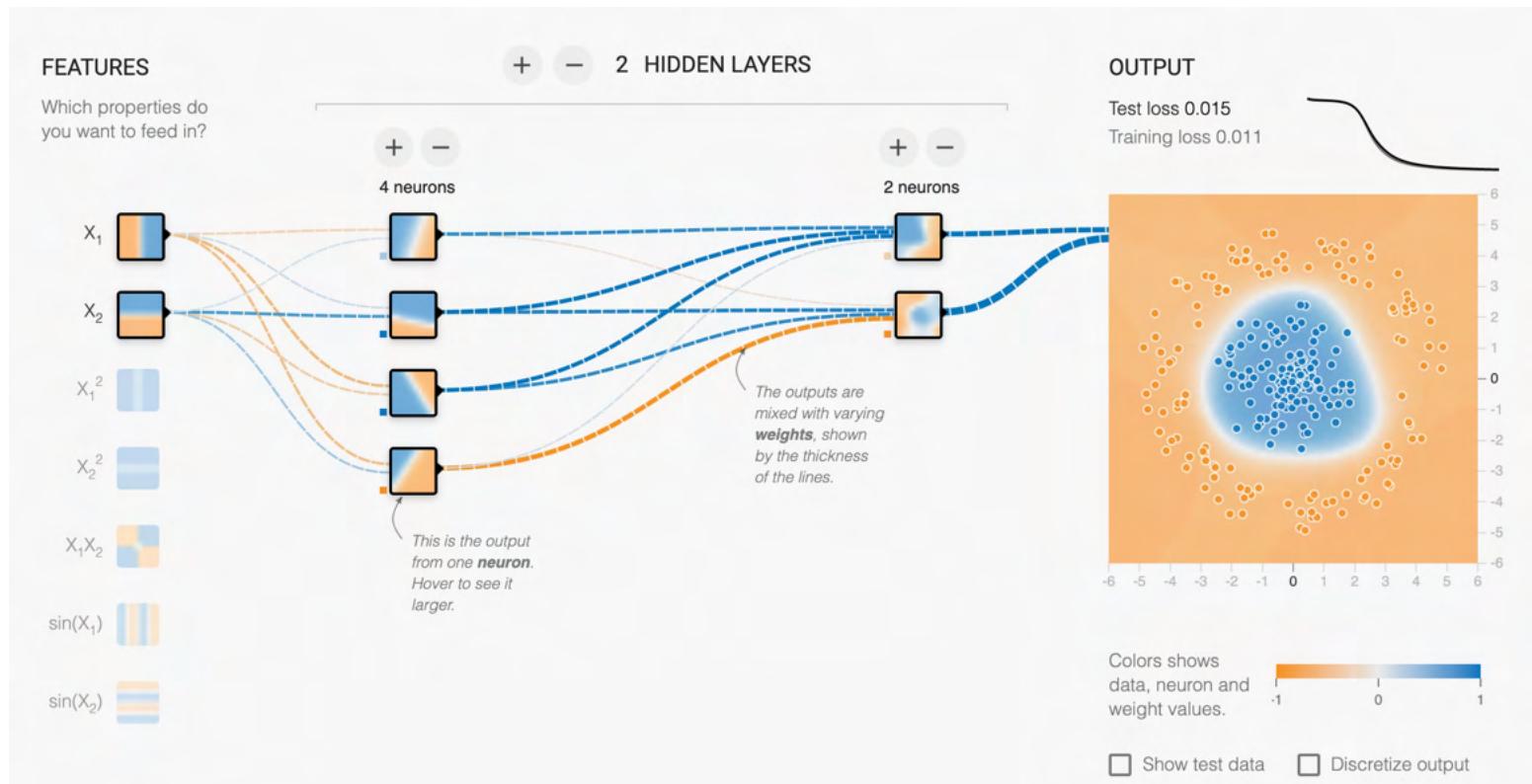


Arbitrary Decision Boundary



Neural Network Training Animation

❖ <https://playground.tensorflow.org/>



Neural Network Learning

Cost Function

Logistic Regression:

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \log h_{\theta}(\mathbf{x}_i) + (1 - y_i) \log (1 - h_{\theta}(\mathbf{x}_i))]$$

$$\begin{aligned} h_{\theta}(\mathbf{x}) &= g(\boldsymbol{\theta}^T \mathbf{x}) \\ &= \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}} \end{aligned}$$

Neural Network:

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) = g\left(z_1^{(2)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) = g\left(z_2^{(2)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) = g\left(z_3^{(2)}\right)$$

$$h_{\Theta}(\mathbf{x}) = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = g\left(z_1^{(3)}\right)$$

Stochastic gradient Descent

Given a training set $\{(\mathbf{x}_i, y_i)\}$

1. Initialize $\theta = \theta_0$

2. For epoch i from 1 to n :

3. For \mathbf{x}_i in \mathcal{D} :

4. Update $\theta_j = \theta_j - \alpha (y_i - h_{\theta}(\mathbf{x}_i)) \mathbf{x}_{ij}$

5. Return θ

(Similar to logistic regression)

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \log h_{\theta}(\mathbf{x}_i) + (1 - y_i) \log (1 - h_{\theta}(\mathbf{x}_i))]$$

Optimizing the Neural Network

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \log h_{\theta}(\mathbf{x}_i) + (1 - y_i) \log (1 - h_{\theta}(\mathbf{x}_i))]$$

Solve via: $\min_{\Theta} J(\Theta)$

$J(\Theta)$ is not convex, so GD on a neural net yields a local optimum
• But, tends to work well in practice

Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Chain Rule

- ❖ Given a function

$$f(x) = A(B(C(x)))$$

- ❖ The derivative is

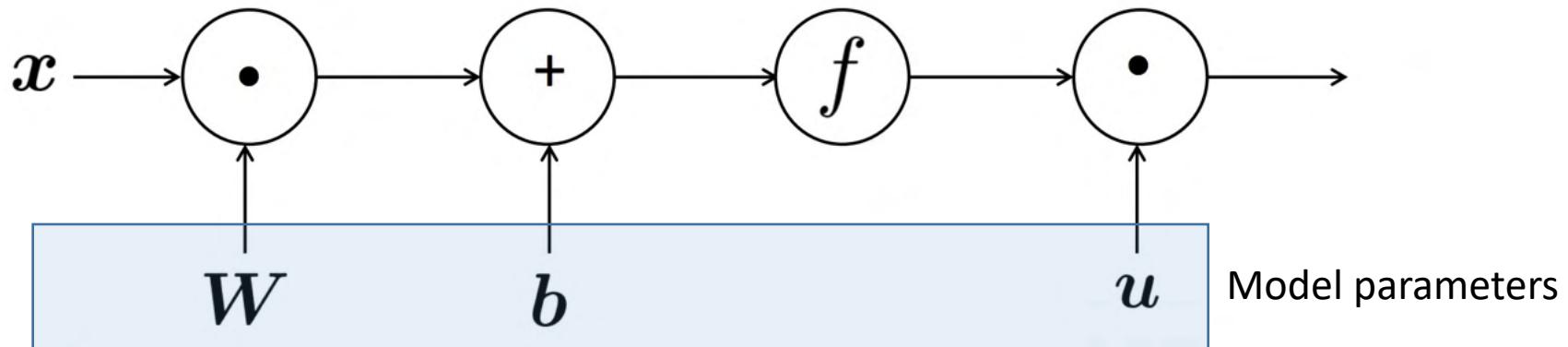
$$f'(x) = f'(A) \cdot A'(B) \cdot B'(C) \cdot C'(x)$$

Backpropagation through Computation Graphs

Computation Graphs and Backpropagation

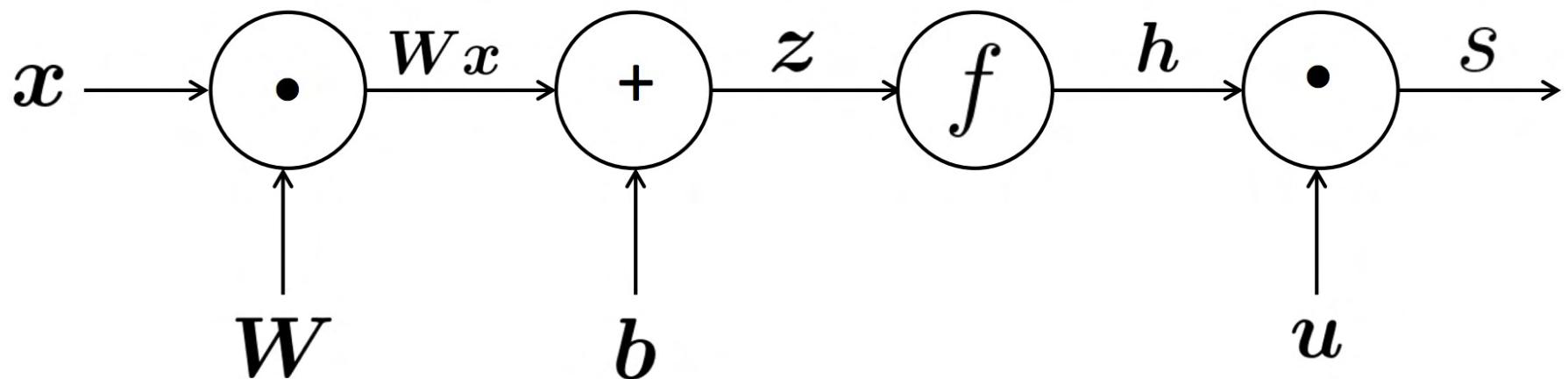
- ❖ Consider the NN on the right
- ❖ We represent NN as a graph

$$\begin{aligned}s &= \mathbf{u}^T \mathbf{h} \\ \mathbf{h} &= f(\mathbf{z}) \\ \mathbf{z} &= \mathbf{W}\mathbf{x} + \mathbf{b} \\ \mathbf{x} &\quad (\text{input})\end{aligned}$$



Forward Propagation

- ❖ Edges pass along result of the operation



$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

Back Propagation

$$s = \mathbf{u}^T \mathbf{h}$$

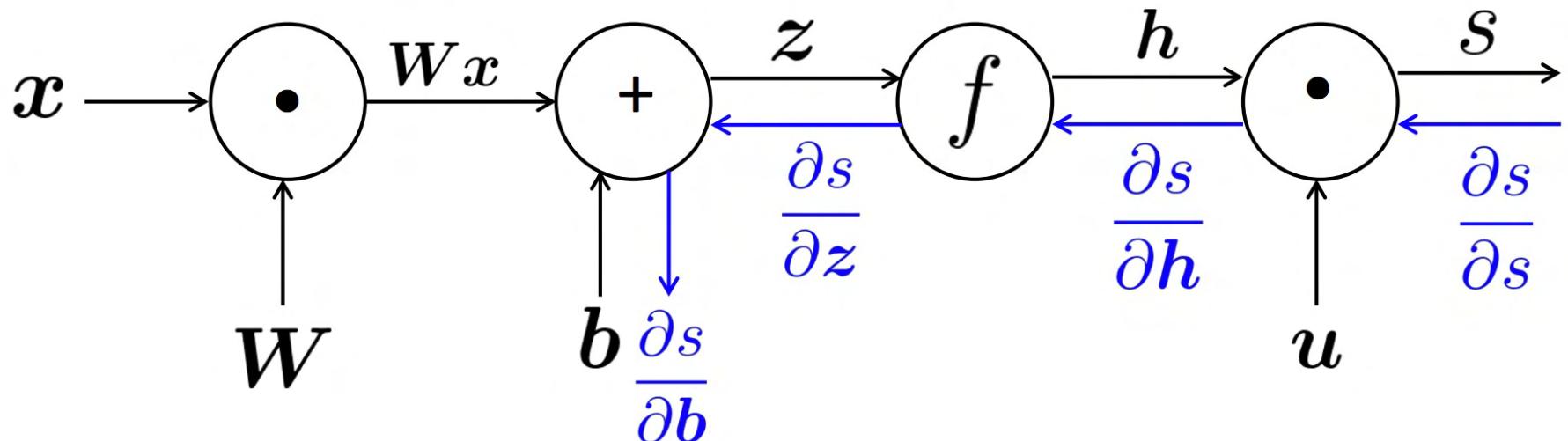
$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

\mathbf{x} (input)

❖ Compute $\frac{\partial s}{\partial b}$

$$\text{Chain Rule: } \frac{\partial s}{\partial b} = \frac{\partial s}{\partial z} \frac{\partial z}{\partial b} = \dots$$



Back Propagation

$$s = \mathbf{u}^T \mathbf{h}$$

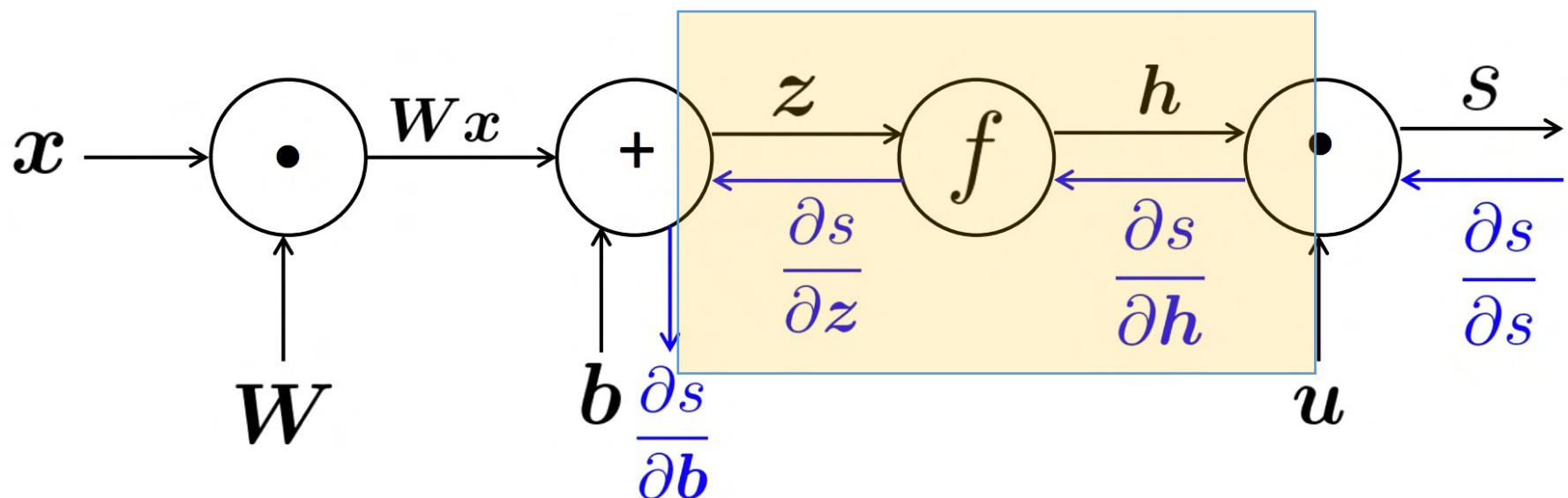
$$\mathbf{h} = f(\mathbf{z})$$

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

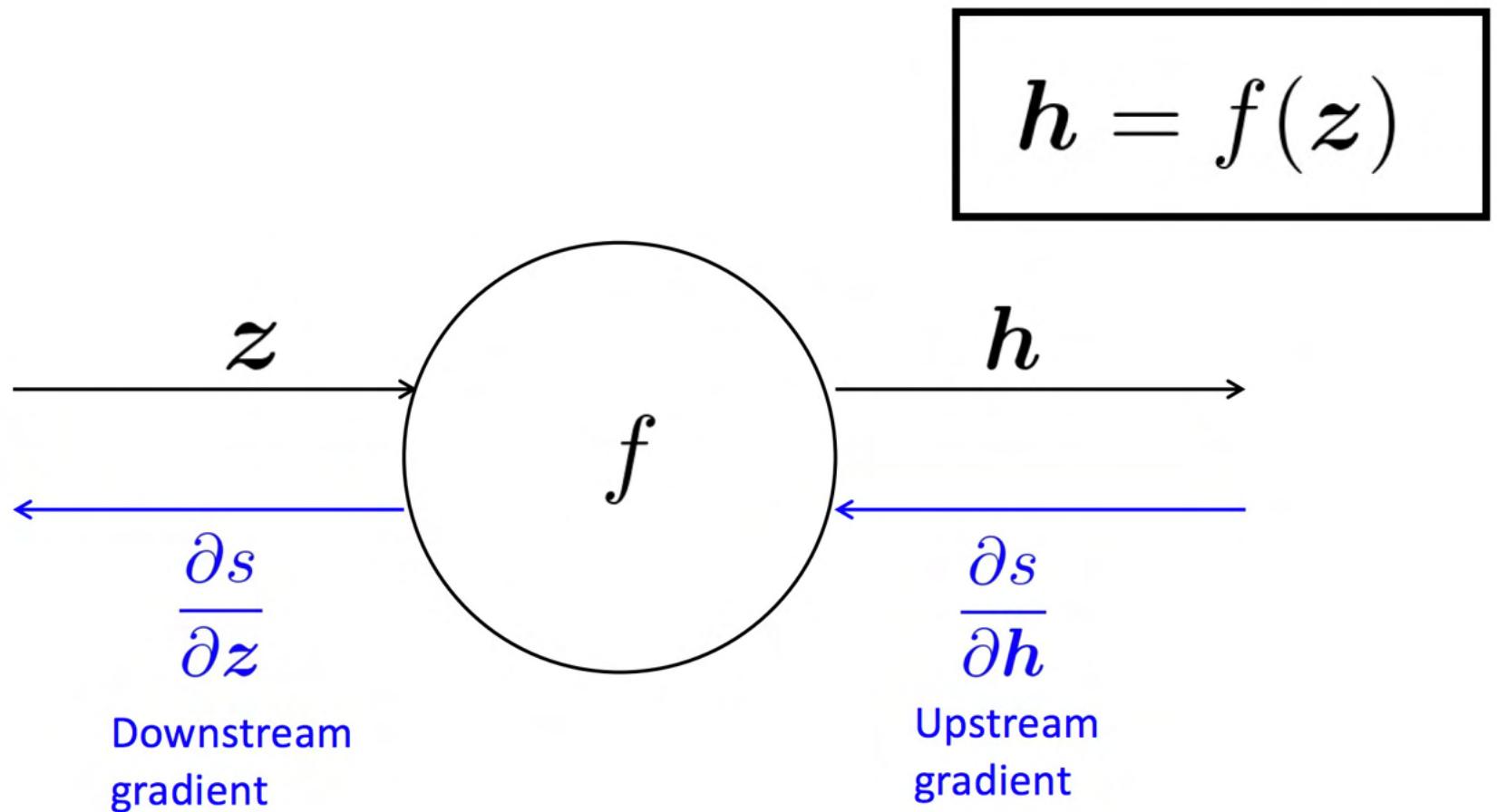
\mathbf{x} (input)

❖ Compute $\frac{\partial s}{\partial b}$

$$\text{Chain Rule: } \frac{\partial s}{\partial b} = \frac{\partial s}{\partial z} \frac{\partial z}{\partial b} = \dots$$

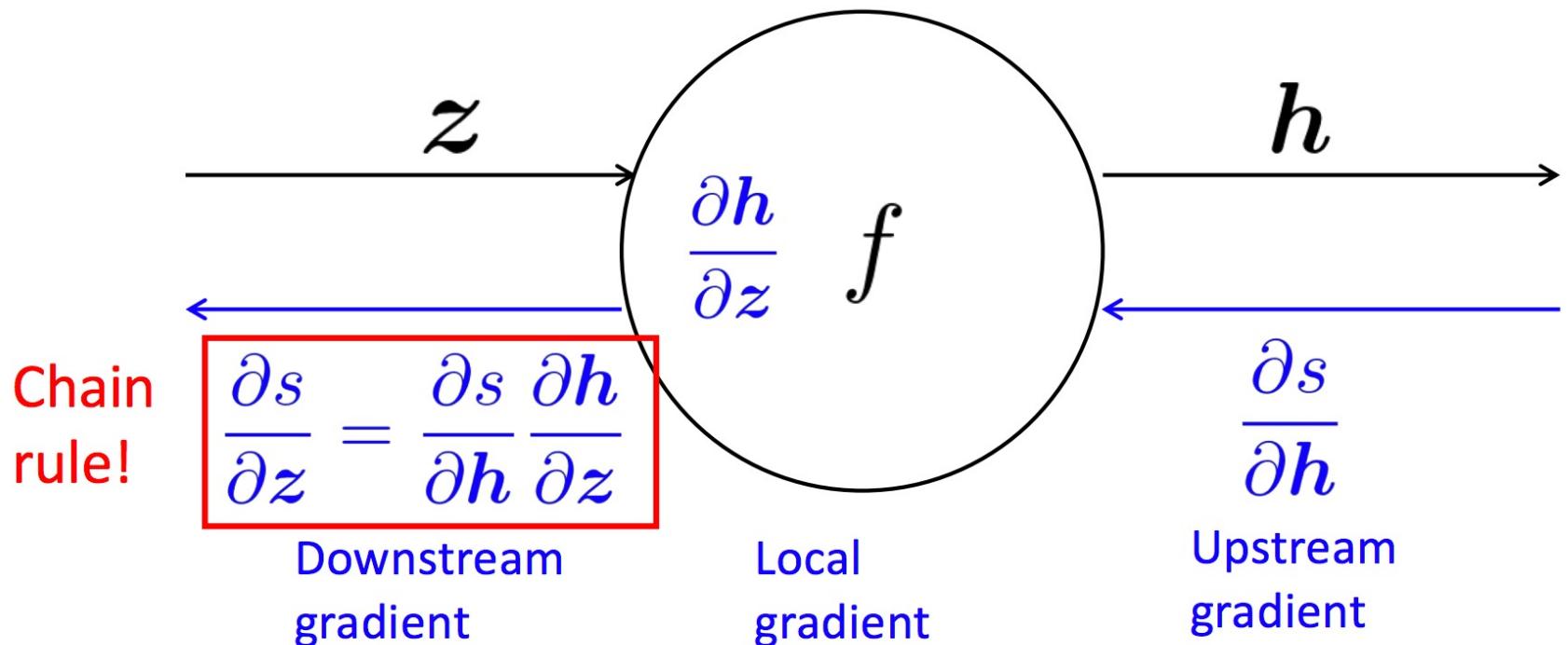


Backpropagation: Single Node



Chain Rule

$$h = f(z)$$



Back Propagation

❖ Compute $\frac{\partial s}{\partial b}$

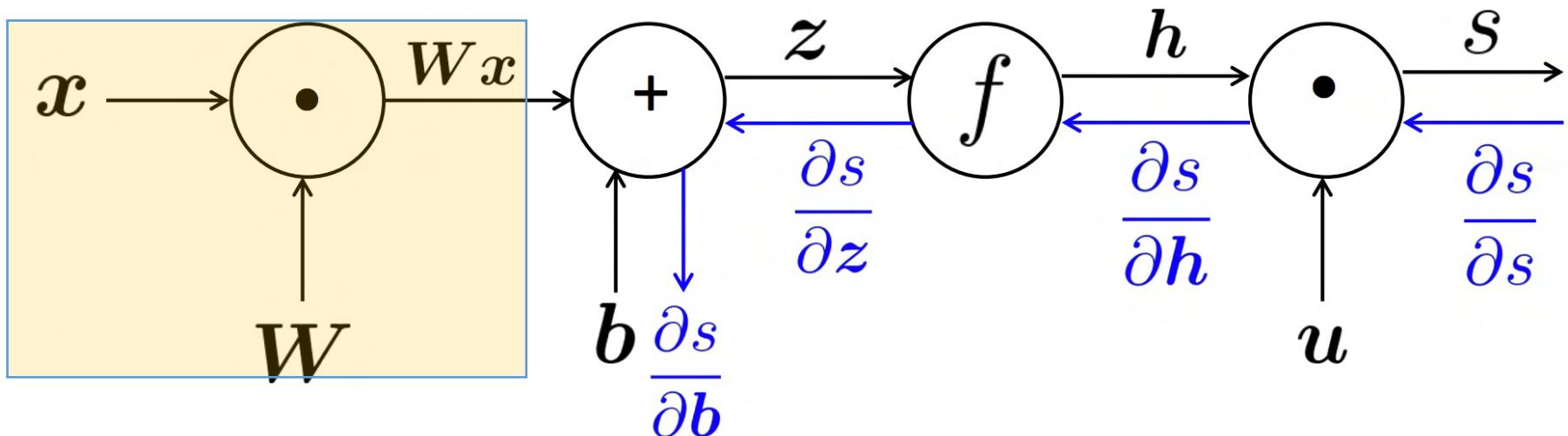
Chain Rule: $\frac{\partial s}{\partial b} = \frac{\partial s}{\partial z} \frac{\partial z}{\partial b} = \dots$

$$s = \mathbf{u}^T \mathbf{h}$$

$$\mathbf{h} = f(\mathbf{z})$$

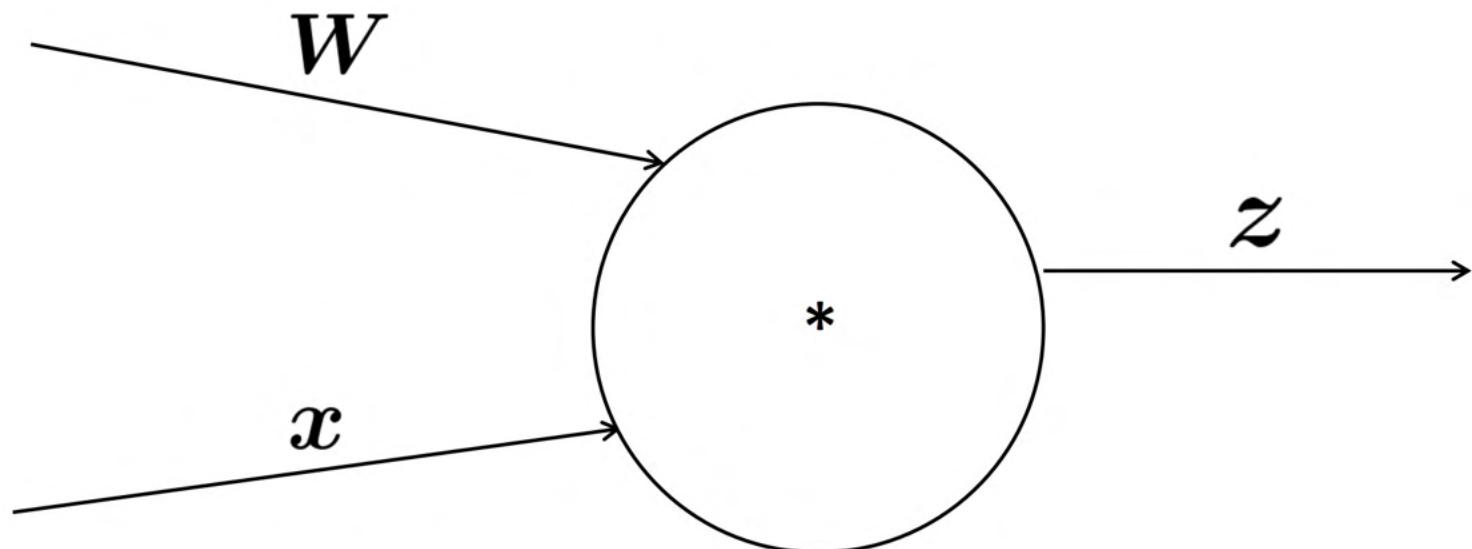
$$\mathbf{z} = \mathbf{Wx} + \mathbf{b}$$

\mathbf{x} (input)

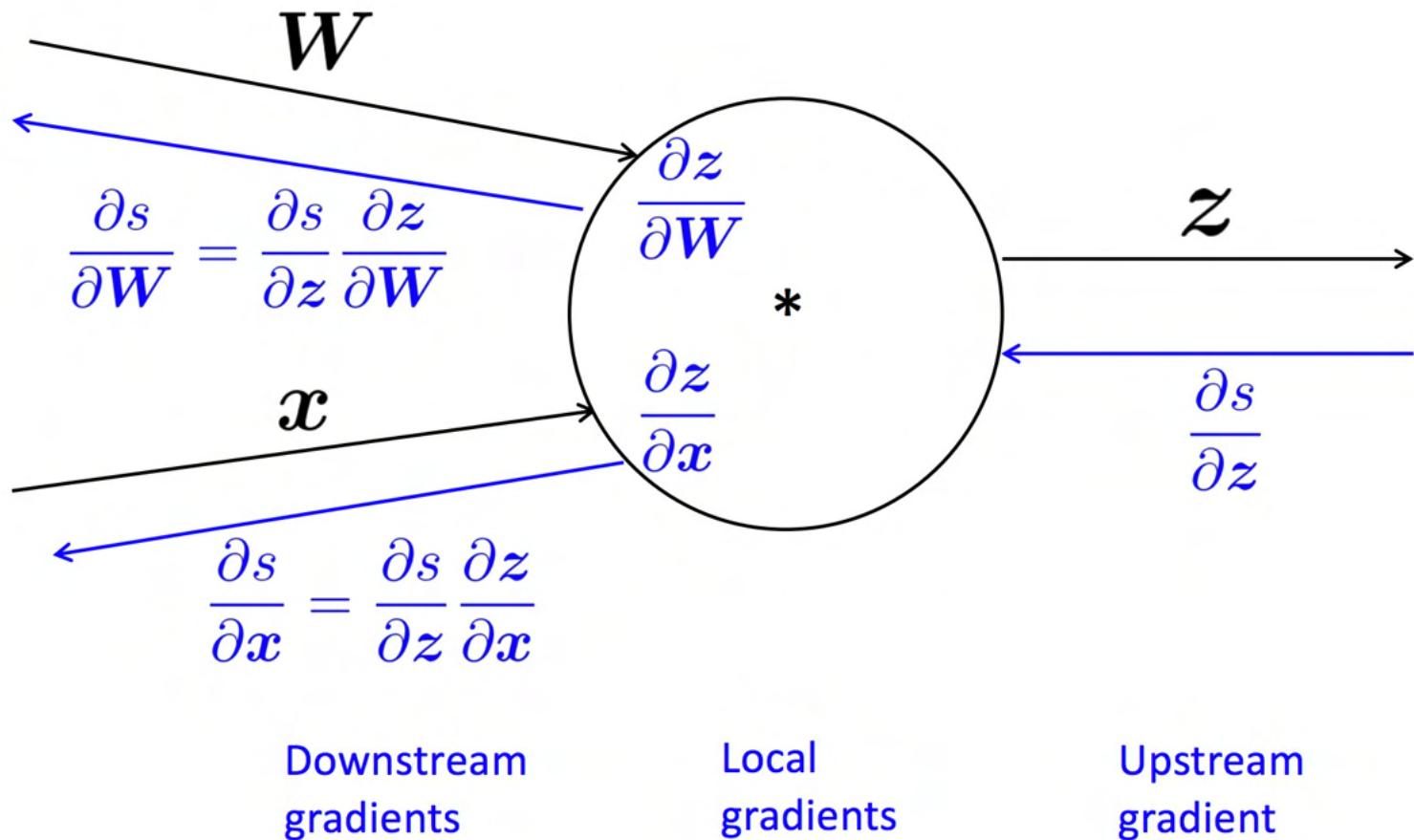


Backpropagation: Single Node

- ❖ Multiple input



Backpropagation: Single Node



Example

$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Draw the computation graph and calculate $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Example

$$f(x, y, z) = (x + y) \max(y, z)$$

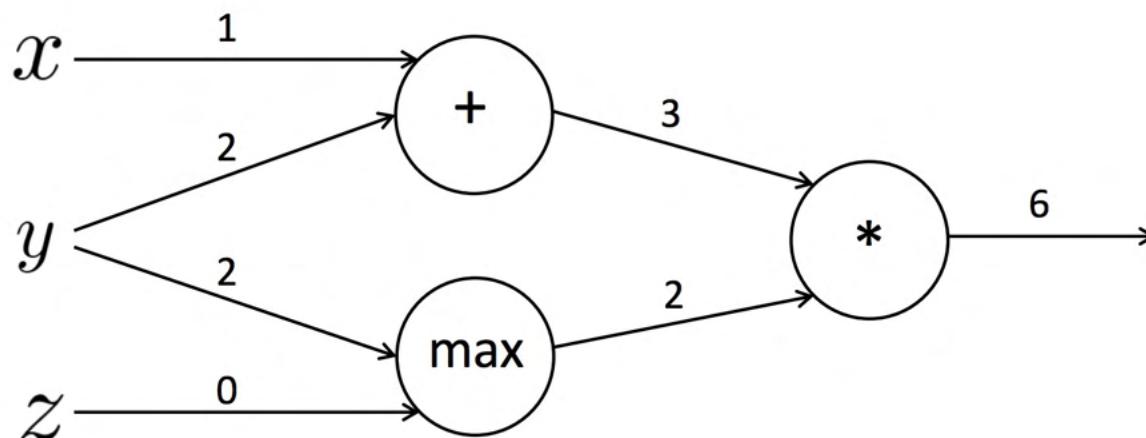
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



Example

$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

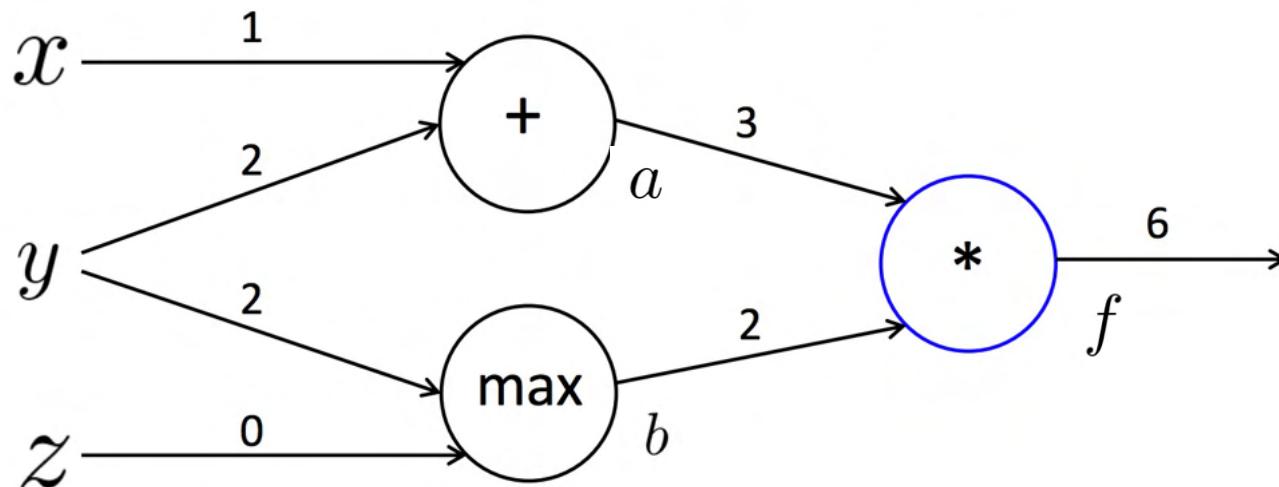
$$f = ab$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$



Example

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$

$$f(x, y, z) = (x + y) \max(y, z)$$

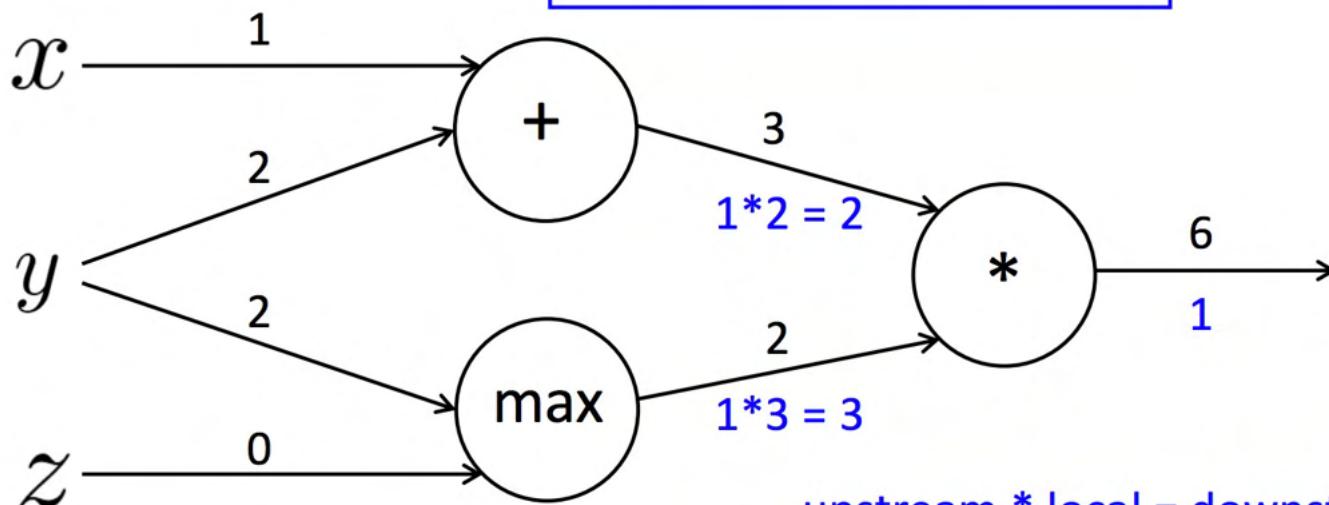
$$x = 1, y = 2, z = 0$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$



upstream * local = downstream

Example

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

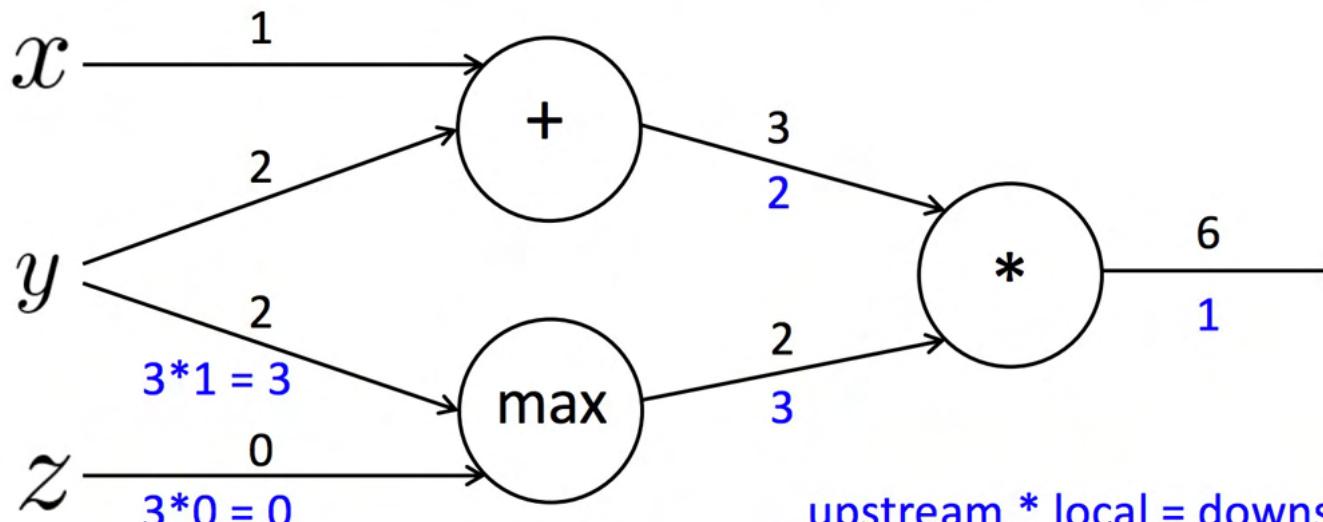
$$f = ab$$

Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$



Example

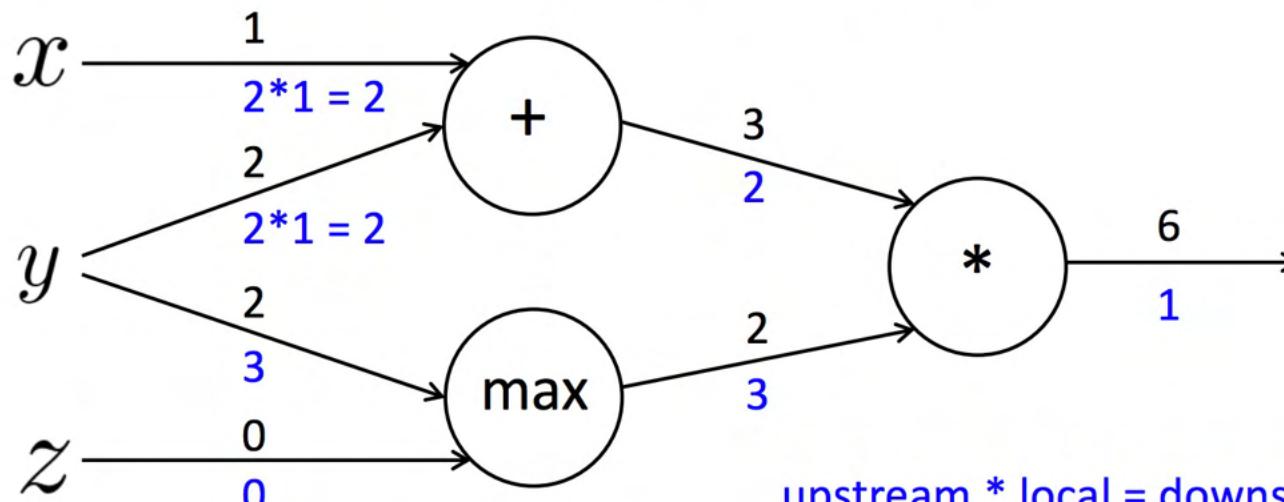
$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$



Local gradients

$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$

upstream * local = downstream

Example

$$f(x, y, z) = (x + y) \max(y, z)$$
$$x = 1, y = 2, z = 0$$

Forward prop steps

$$a = x + y$$

$$b = \max(y, z)$$

$$f = ab$$

$$\frac{\partial f}{\partial x} = 2$$

$$\frac{\partial f}{\partial y} = 3 + 2 = 5$$

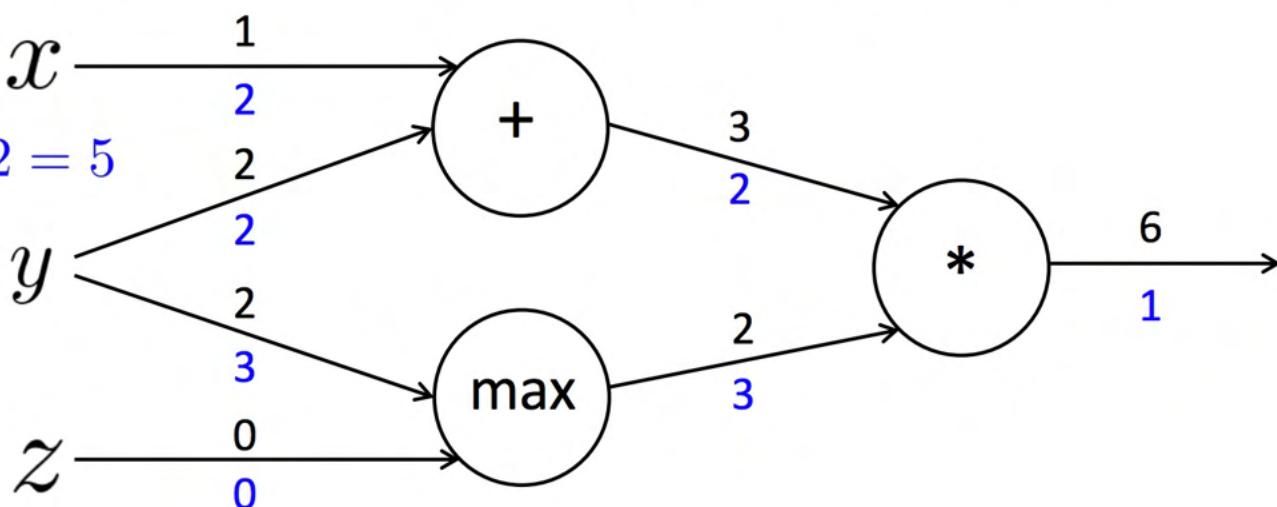
$$\frac{\partial f}{\partial z} = 0$$

Local gradients

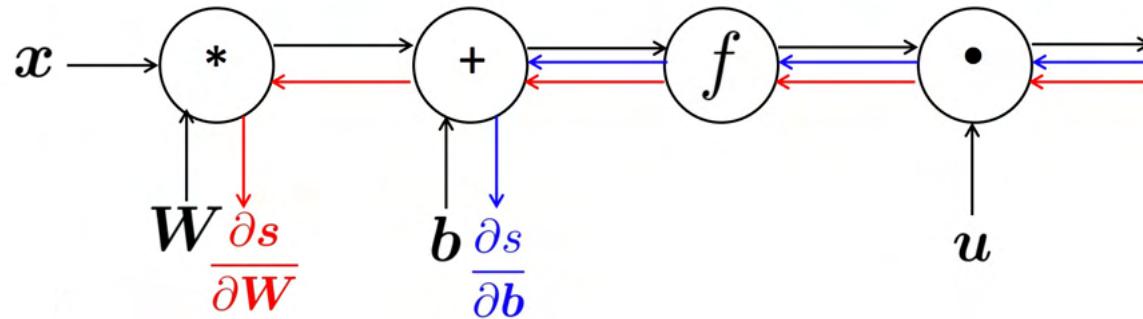
$$\frac{\partial a}{\partial x} = 1 \quad \frac{\partial a}{\partial y} = 1$$

$$\frac{\partial b}{\partial y} = \mathbf{1}(y > z) = 1 \quad \frac{\partial b}{\partial z} = \mathbf{1}(z > y) = 0$$

$$\frac{\partial f}{\partial a} = b = 2 \quad \frac{\partial f}{\partial b} = a = 3$$



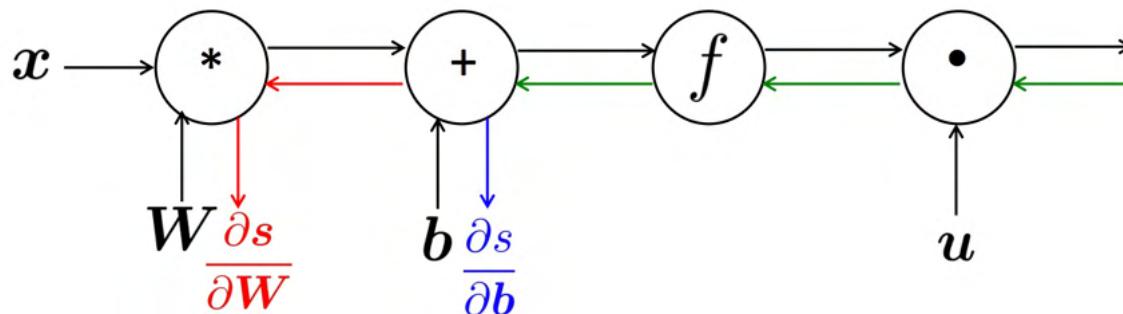
Compute all gradients at once



Naïve way to compute gradients:

Compute each component separately

⇒ Redundant computation

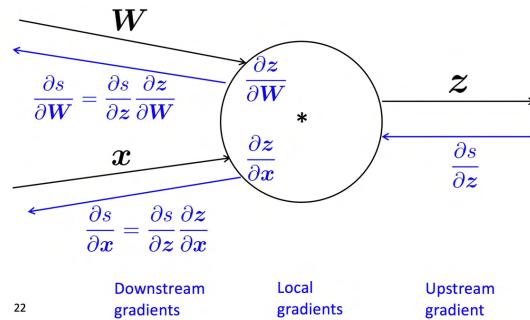


Why you should understand Backprop

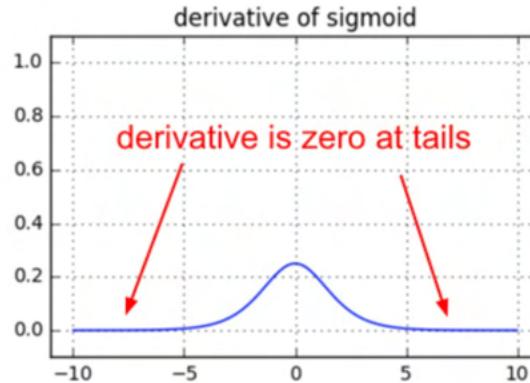
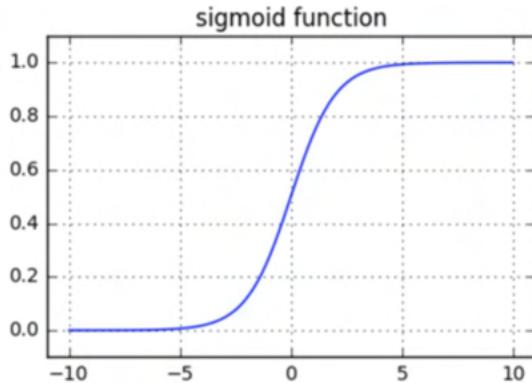
- ❖ Modern deep learning library implements backprop as a black-box for you
 - ❖ You can take a plane without knowing why it flies
 - ❖ but you're designing aircraft...
- ❖ Backpropagation doesn't always work perfectly.
 - ❖ Understanding why is crucial for debugging and improving models

<https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b>

Example: Gradient of sigmoid



```
z = 1/(1 + np.exp(-np.dot(W, x))) # forward pass  
dx = np.dot(W.T, z*(1-z)) # backward pass: local gradient for x  
dW = np.outer(z*(1-z), x) # backward pass: local gradient for W
```



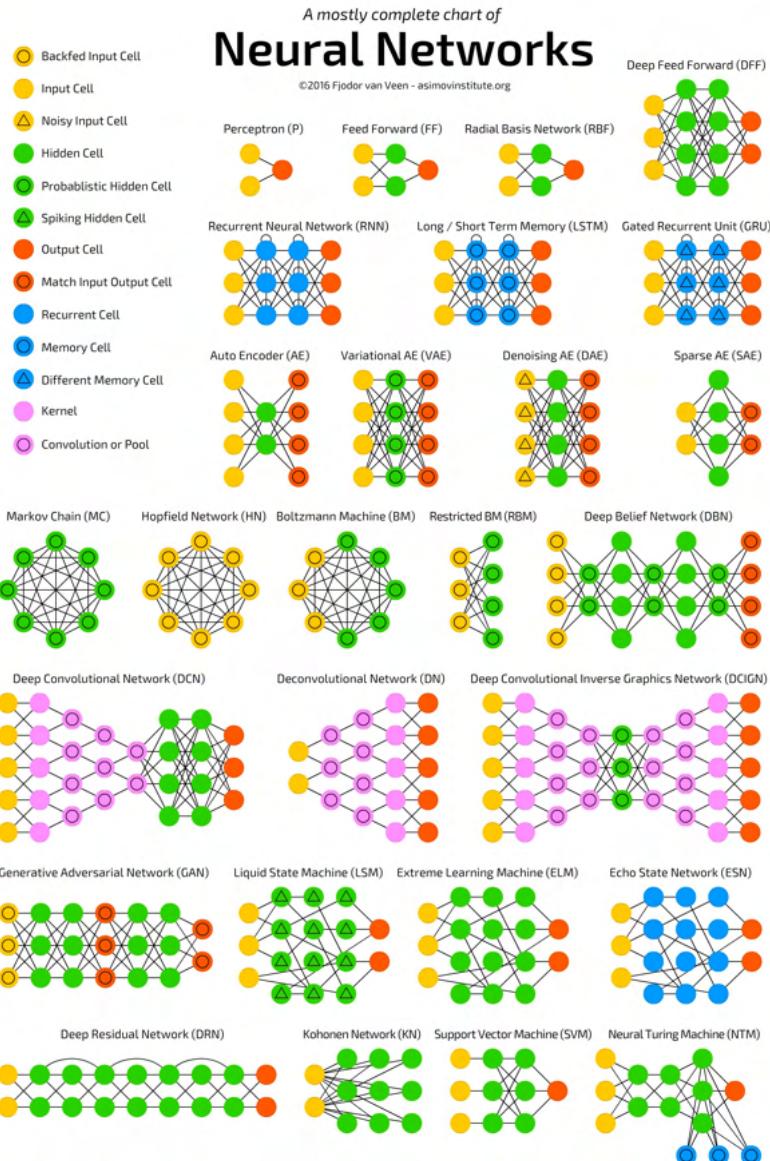
vanish gradient issue

More Details

- ❖ Parameter Initialization
 - ❖ Normally initialize weights to small random values; various designs
- ❖ Optimizer
 - ❖ Usually SGD works
 - ❖ Several SGD variants (e.g., ADAM) automatically adjust learning rate based on an accumulated gradient

A neural network zoo

- ❖ The flexibility of NN allows us to try out different ideas
- ❖ However, there is no magic



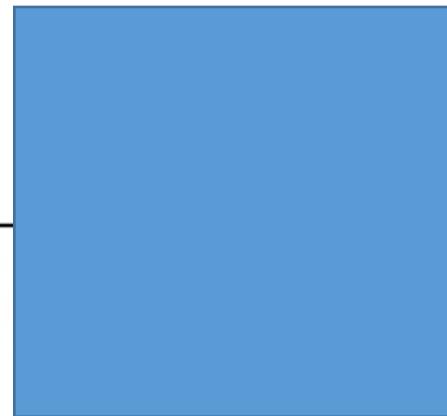
Modeling with Neural Networks

(Advanced Topic/Not Included in Final)

Example – Language Model

- ❖ Predict next word

the students opened their _____



Idea 1: A fixed-window neural Language Model

output distribution

$$\hat{y} = \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

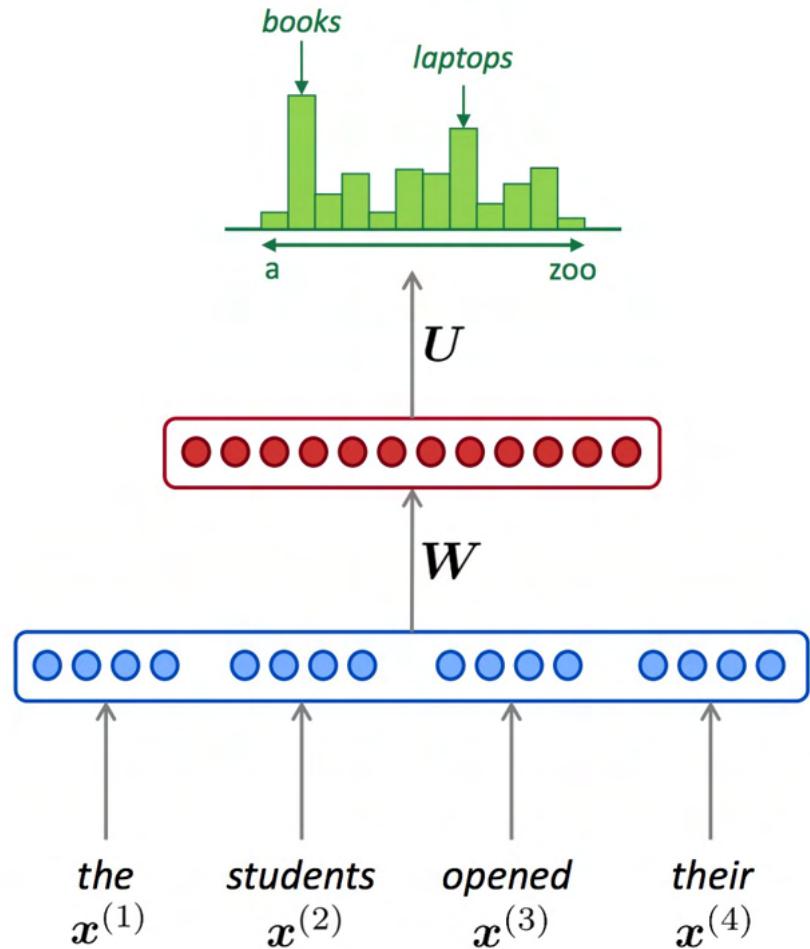
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

concatenated word embeddings

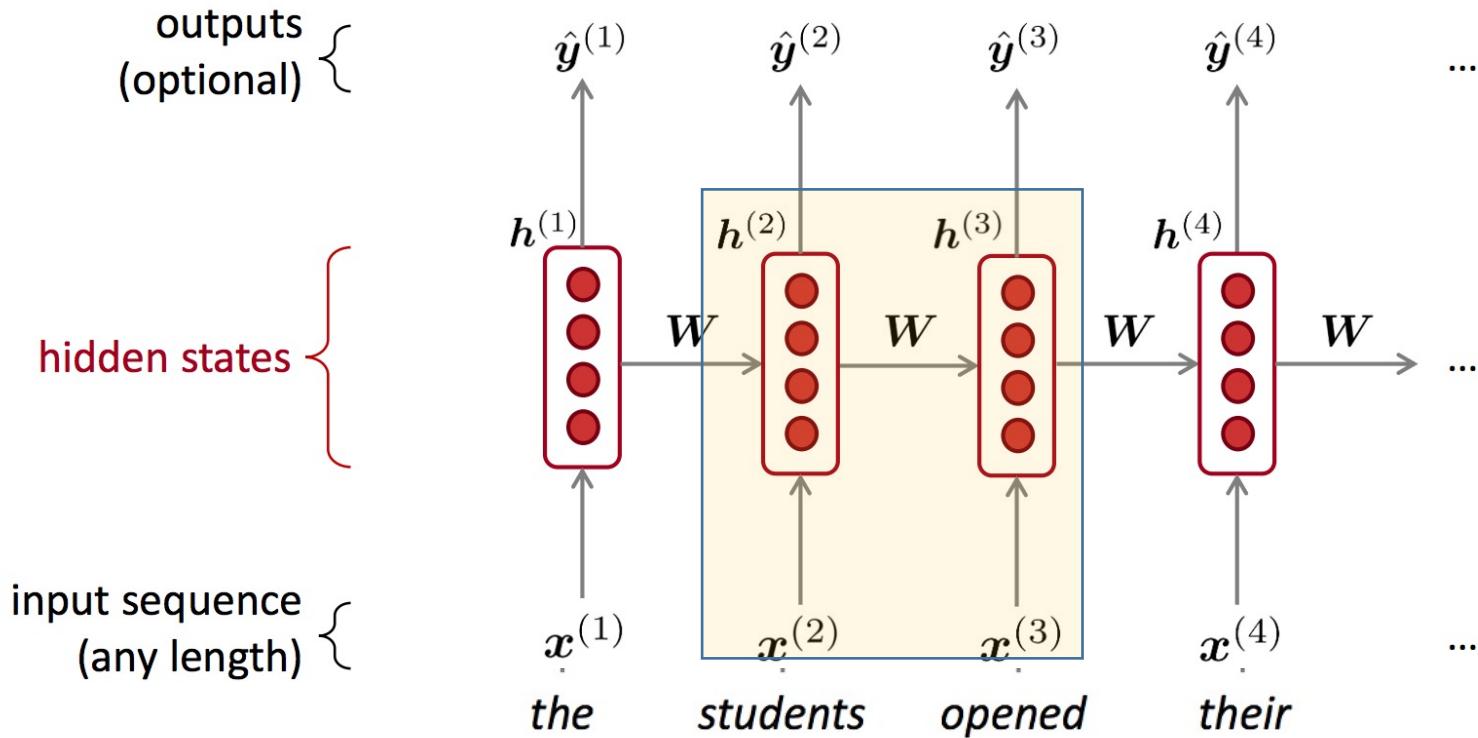
$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

words / one-hot vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$

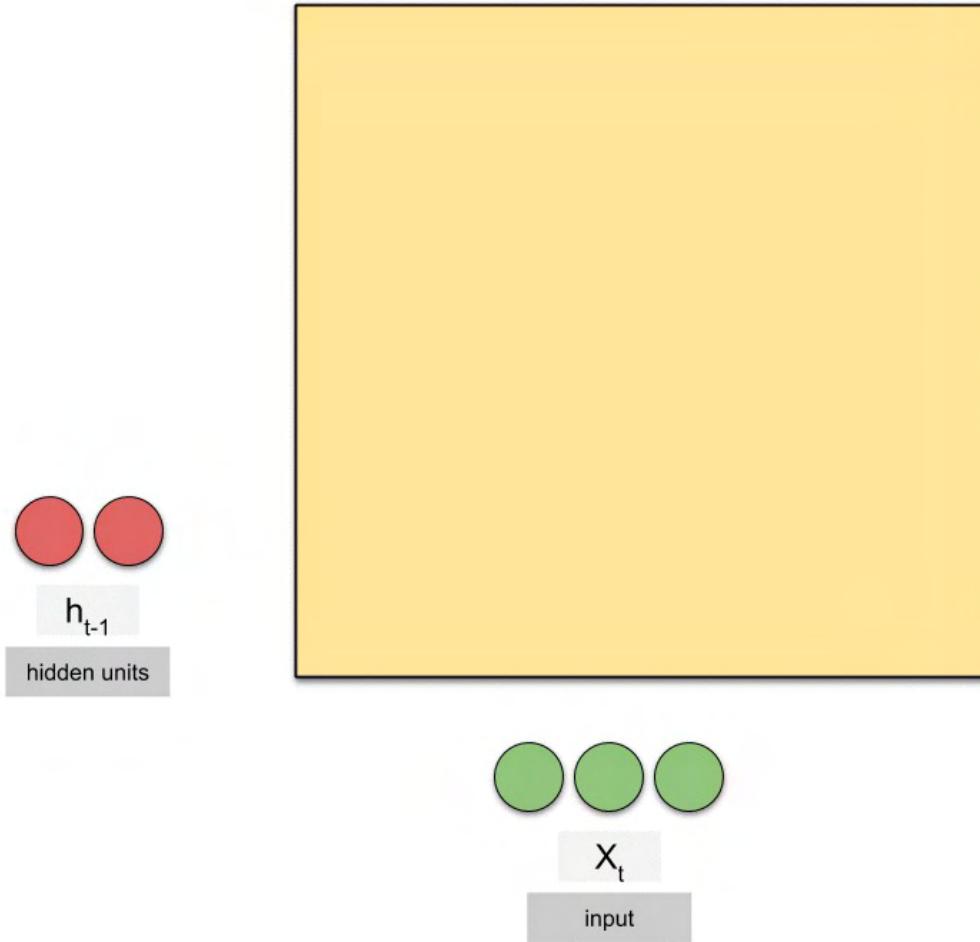


Idea 2: Recurrent Neural Networks (RNN)



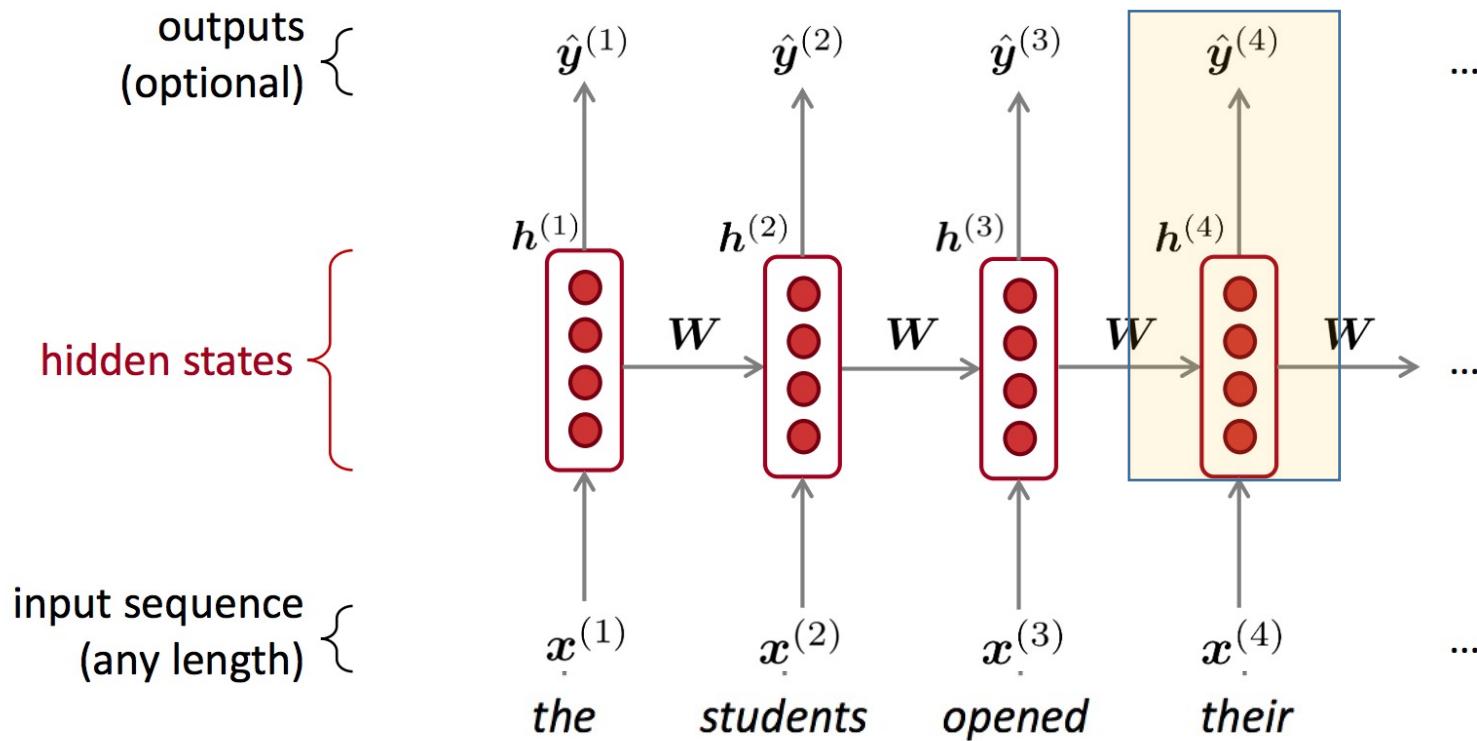
Core idea: Apply the same weights W repeatedly

Recurrent Neural Network



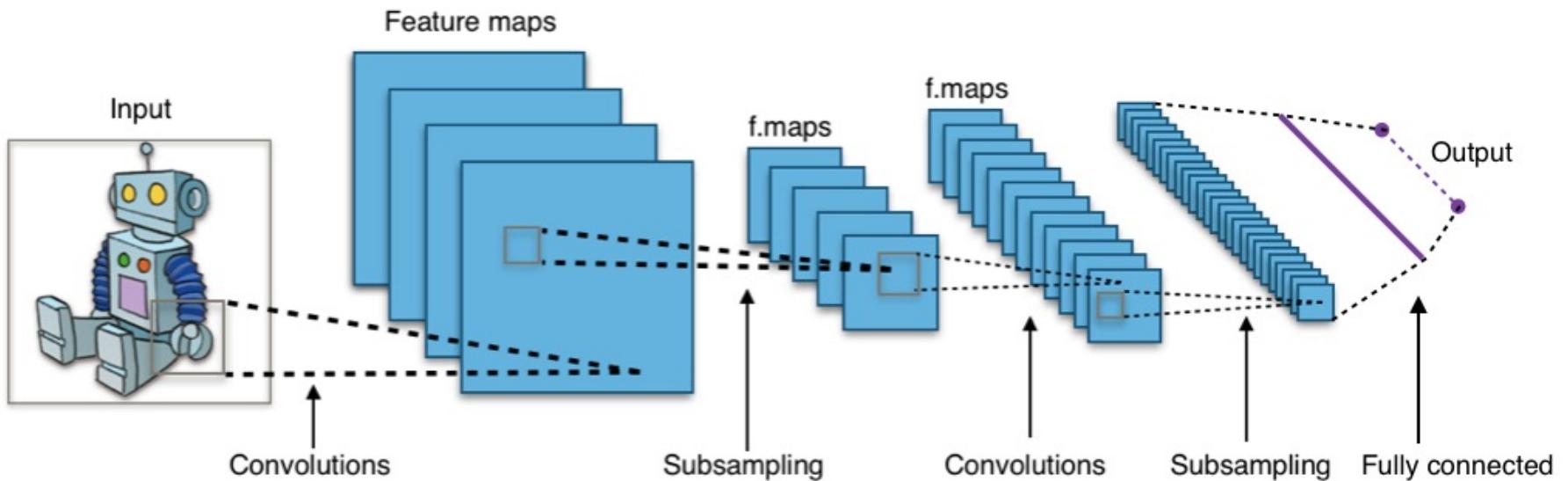
<https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45>

Prediction using Latent State

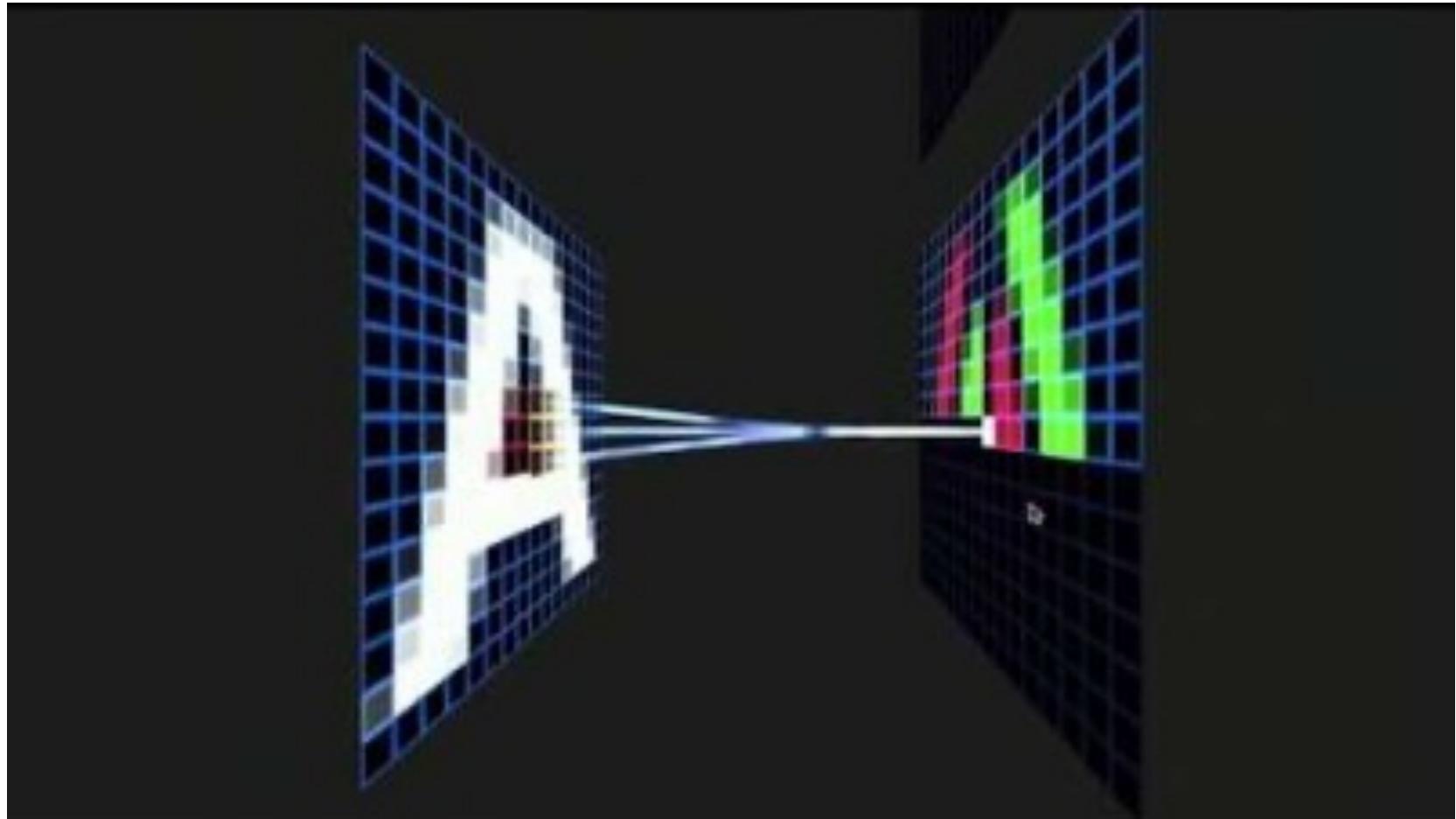


Core idea: Apply the same weights W repeatedly

Idea 3: Convolutional NN



Convolutional NN



<https://www.youtube.com/watch?v=f0t-OCG79-U>

Lecture 10: Computational Learning Theory Fall 2021

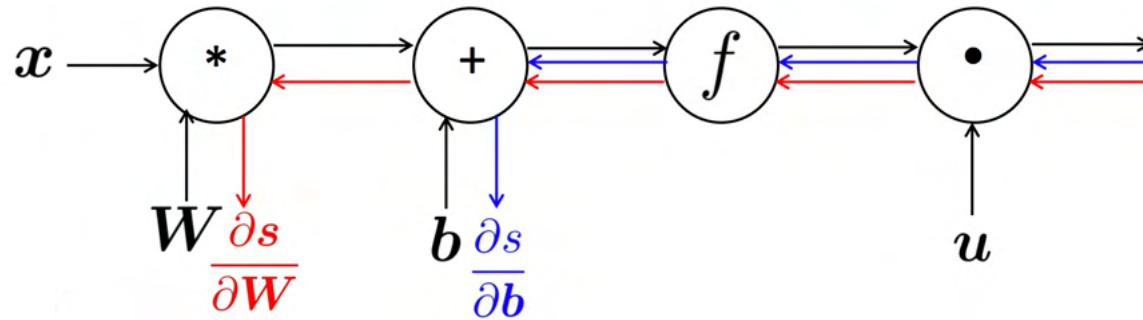
Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

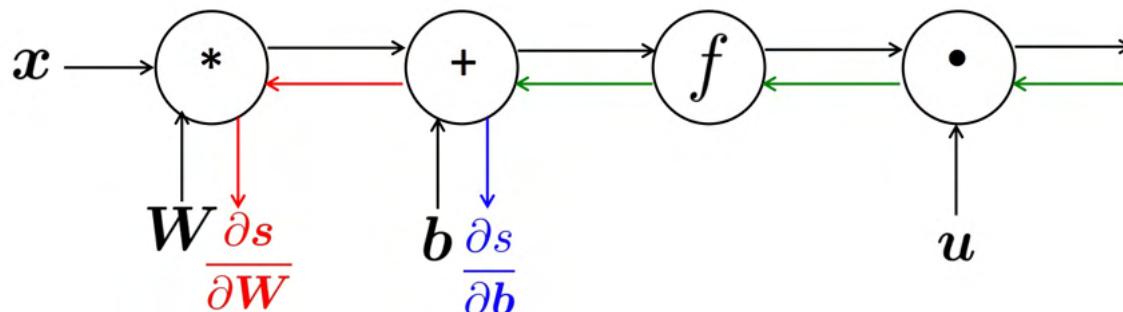
Compute all gradients at once



Naïve way to compute gradients:

Compute each component separately

⇒ Redundant computation

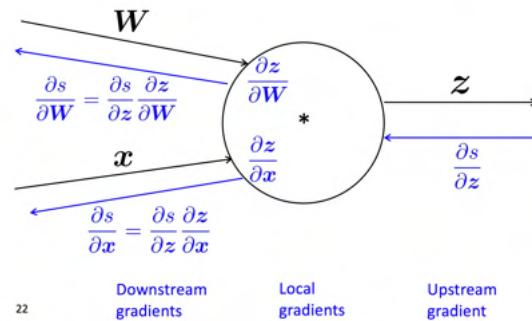


Why you should understand Backprop

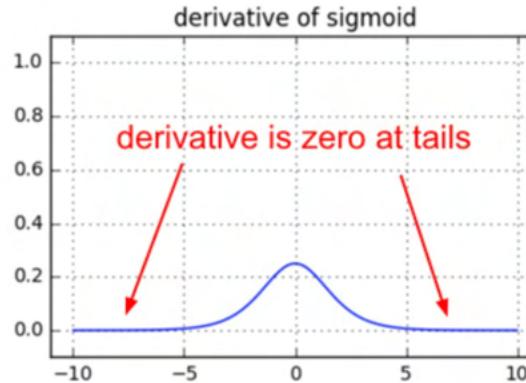
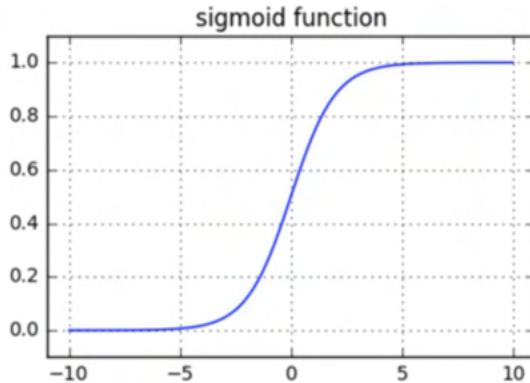
- ❖ Modern deep learning library implements backprop as a black-box for you
 - ❖ You can take a plane without knowing why it flies
 - ❖ but you're designing aircraft...
- ❖ Backpropagation doesn't always work perfectly.
 - ❖ Understanding why is crucial for debugging and improving models

<https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b>

Example: Gradient of sigmoid



```
z = 1/(1 + np.exp(-np.dot(W, x))) # forward pass  
dx = np.dot(W.T, z*(1-z)) # backward pass: local gradient for x  
dW = np.outer(z*(1-z), x) # backward pass: local gradient for W
```



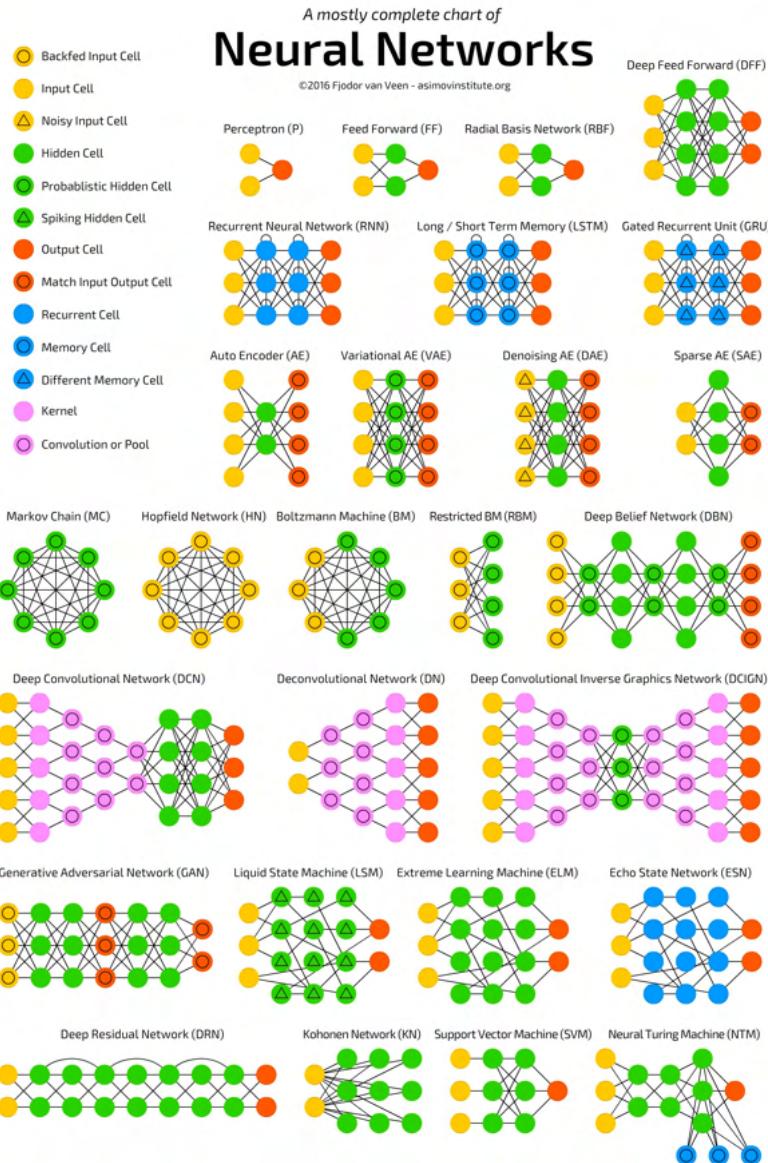
vานish gradient issue

More Details

- ❖ Parameter Initialization
 - ❖ Normally initialize weights to small random values; various designs
- ❖ Optimizer
 - ❖ Usually SGD works
 - ❖ Several SGD variants (e.g., ADAM) automatically adjust learning rate based on an accumulated gradient

A neural network zoo

- ❖ The flexibility of NN allows us to try out different ideas
- ❖ However, there is no magic



Modeling with Neural Networks

(Advanced Topic/Not Included in Final)

Example – Language Model

(Advanced Topic/Not Included in Final)

- ❖ Predict next word

the students opened their _____



Idea 1: A fixed-window neural Language Model

(Advanced Topic/Not Included in Final)

output distribution

$$\hat{y} = \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

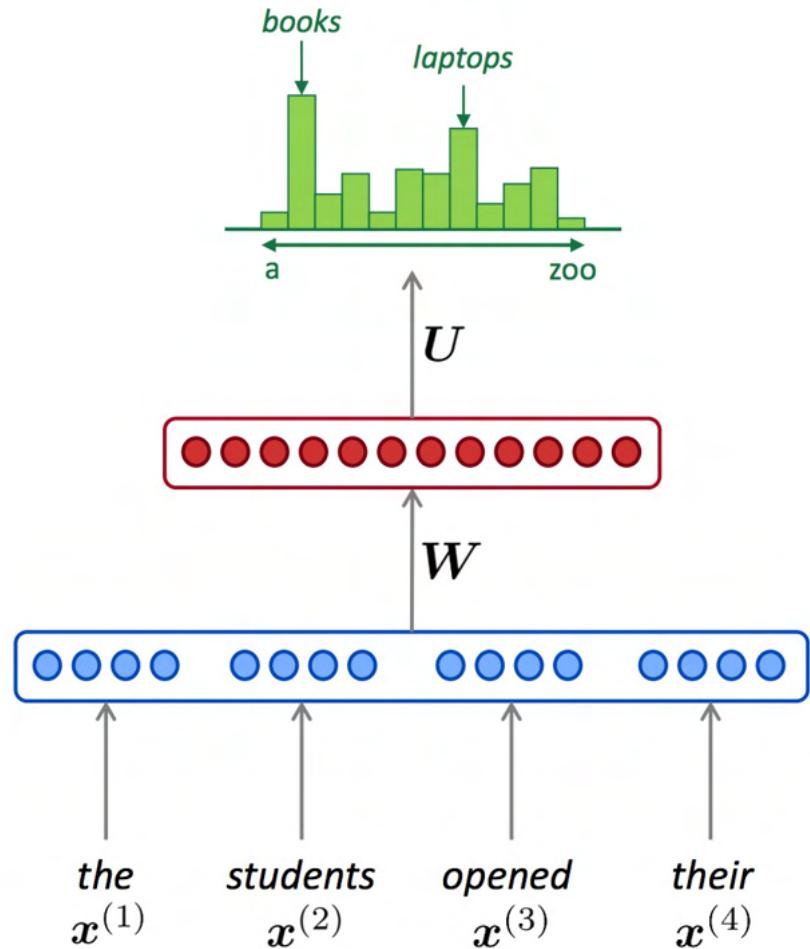
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

concatenated word embeddings

$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

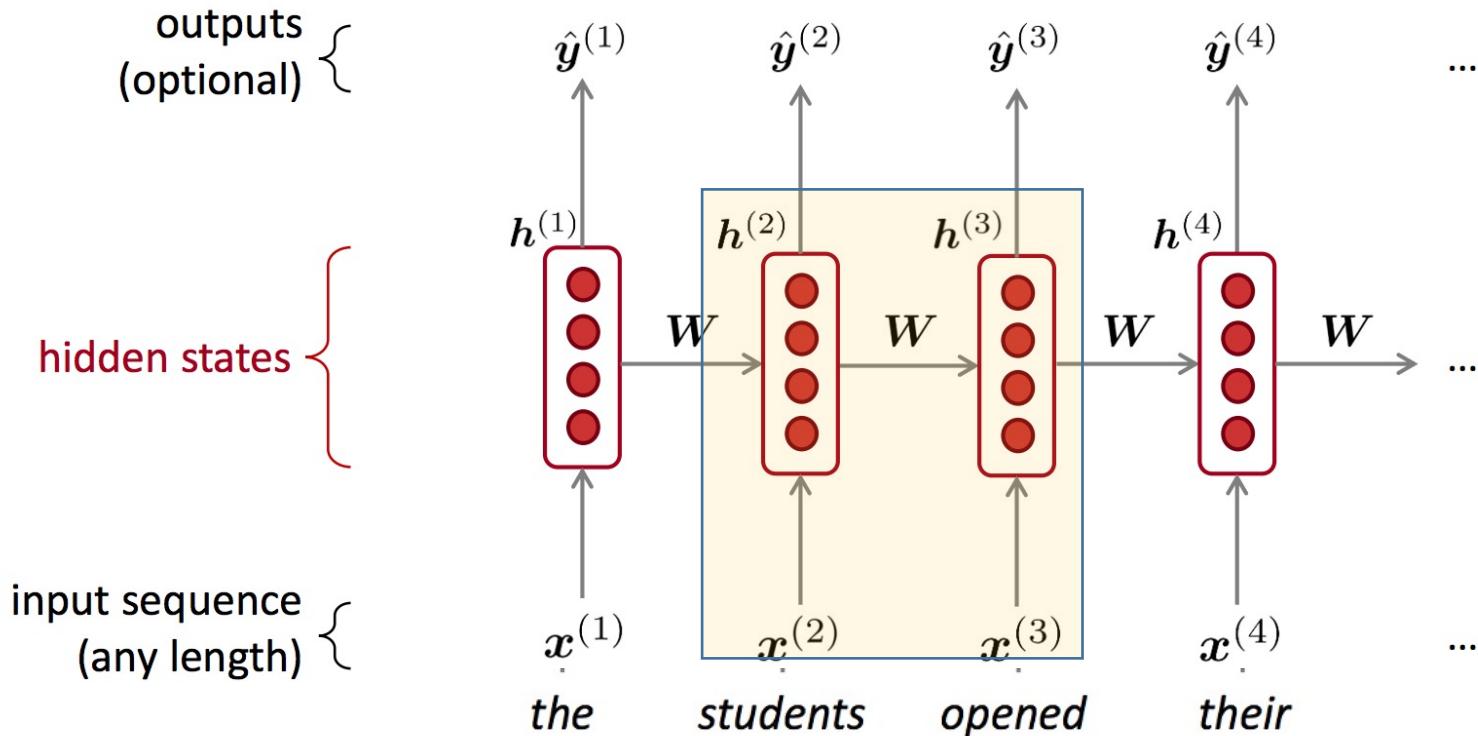
words / one-hot vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$



Idea 2: Recurrent Neural Networks (RNN)

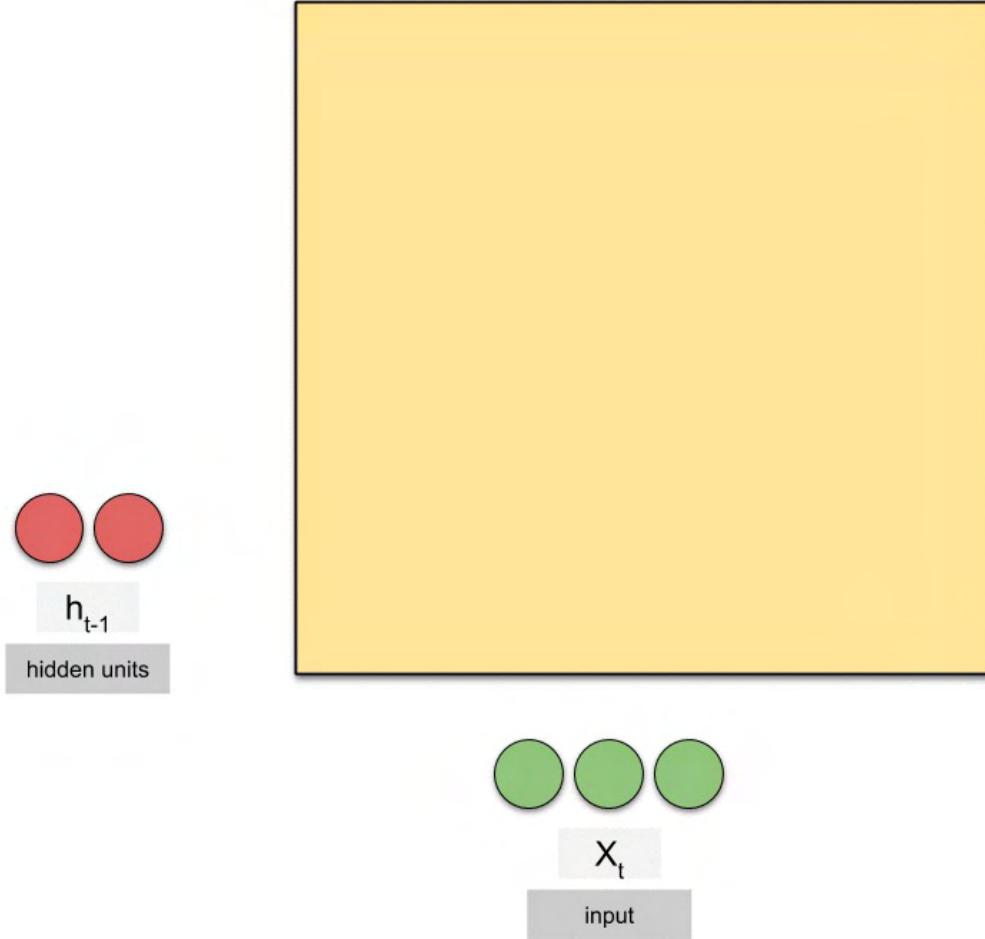
(Advanced Topic/Not Included in Final)



Core idea: Apply the same weights W repeatedly

Recurrent Neural Network

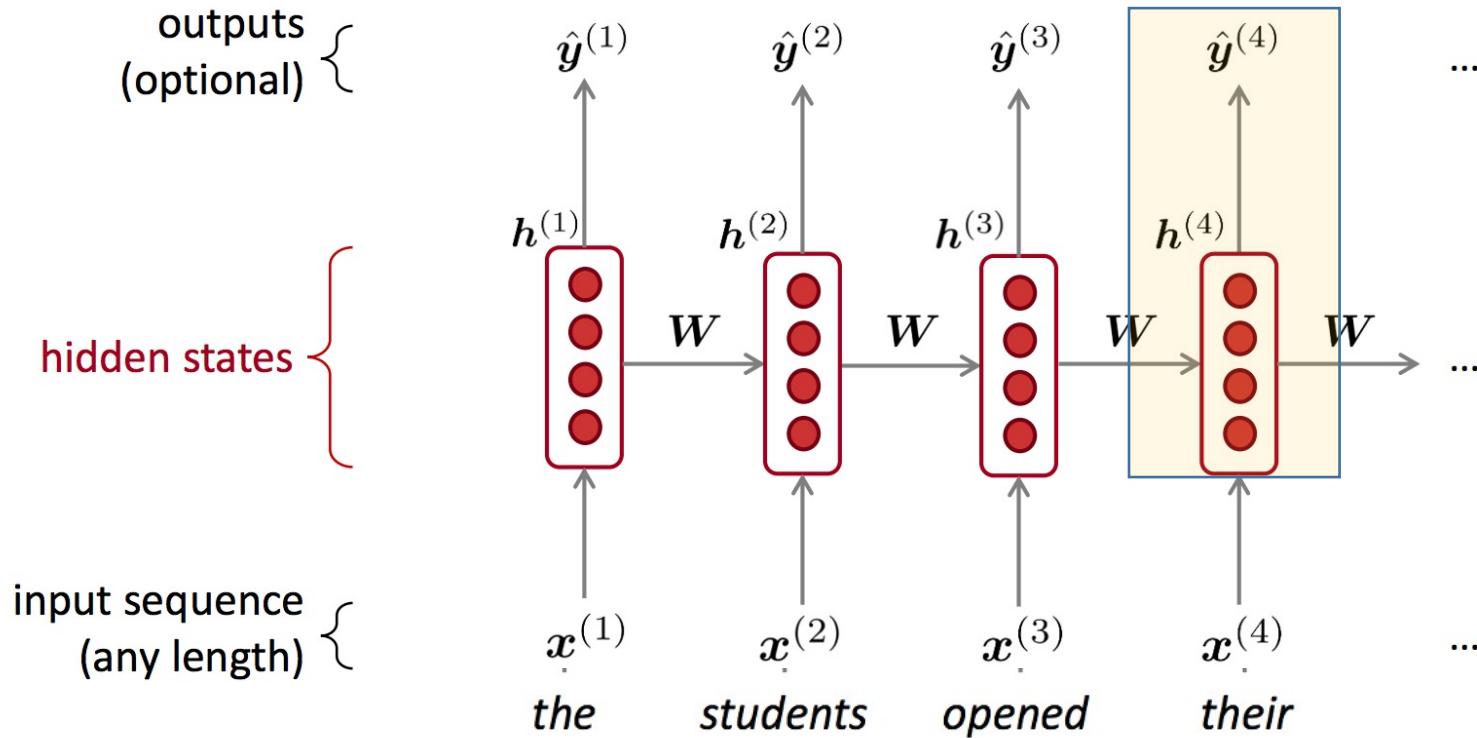
(Advanced Topic/Not Included in Final)



<https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45>

Prediction using Latent State

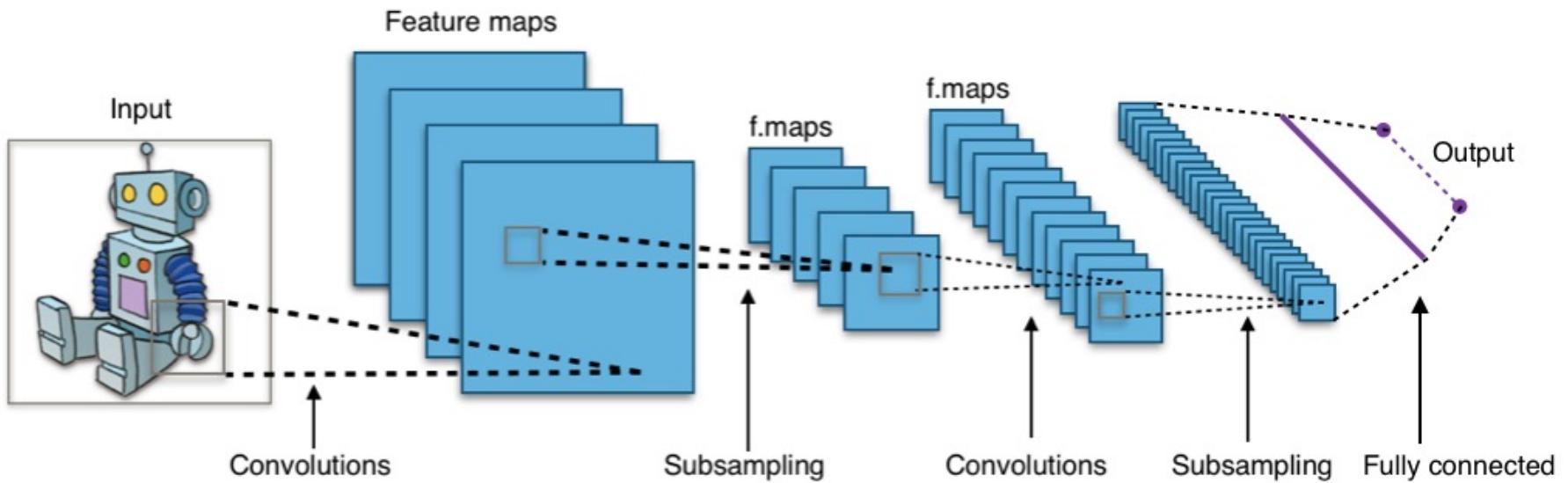
(Advanced Topic/Not Included in Final)



Core idea: Apply the same weights W repeatedly

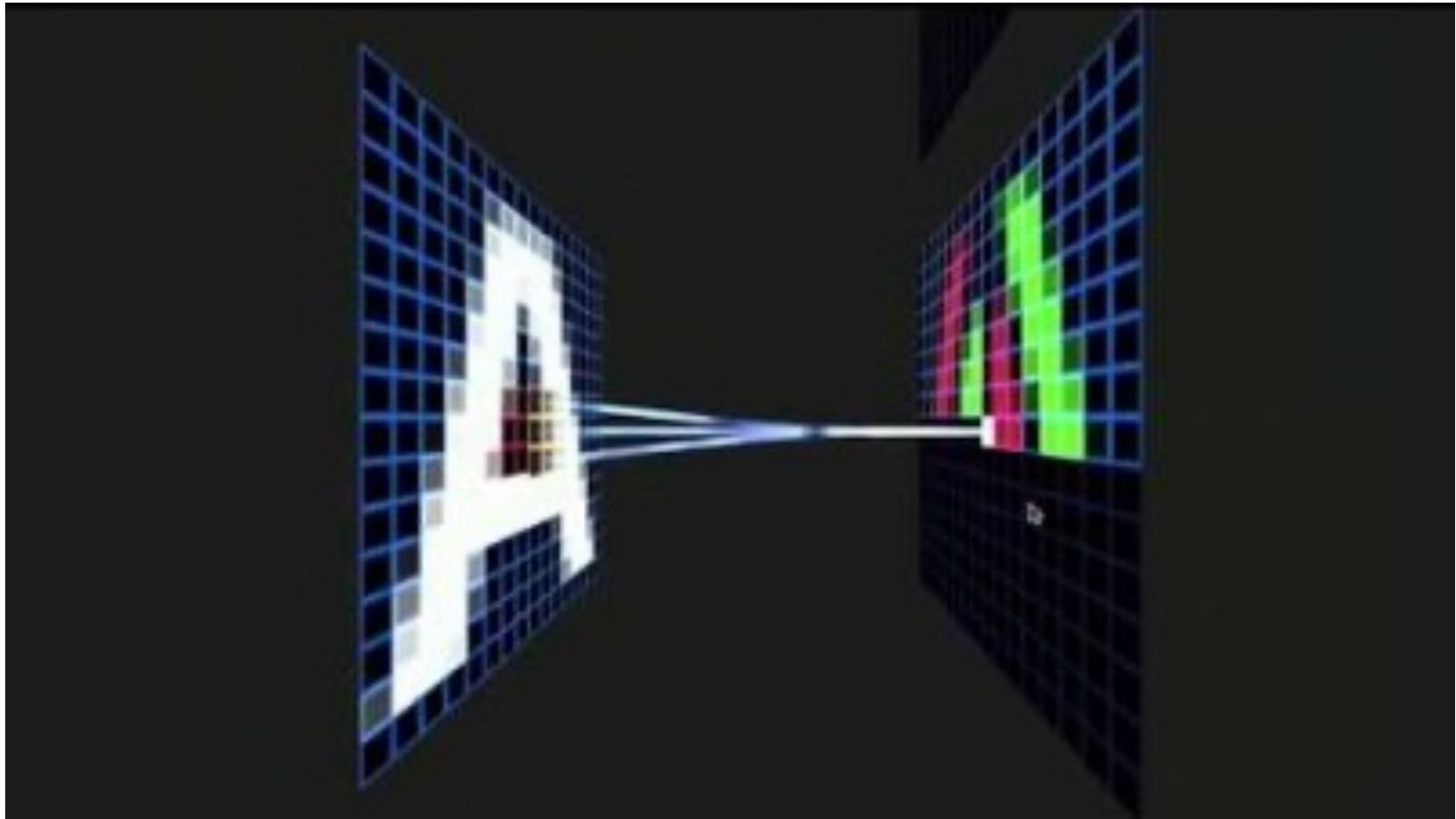
Idea 3: Convolutional NN

(Advanced Topic/Not Included in Final)



Convolutional NN

(Advanced Topic/Not Included in Final)



<https://www.youtube.com/watch?v=f0t-OCG79-U>

Lecture 10: Computational Learning Theory Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

This lecture: Computational Learning Theory

- ❖ The Theory of Generalization
- ❖ Probably Approximately Correct (PAC) learning

Learning Monotone Conjunctions

❖ Hypothesis class:

$$f = x_1 ?$$

$$f = x_2 ?$$

$$f = x_1 \wedge x_2 \wedge x_3 ?$$

$$f = x_1 \wedge x_2 ?$$

$$f = x_2 \wedge x_3 ?$$

Exercise

- ❖ Hypothesis class (3 variables):

$$f = x_1 ?$$

$$f = x_2 ?$$

$$f = x_1 \wedge x_2 \wedge x_3 ?$$

$$f = x_1 \wedge x_2 ?$$

$$f = x_2 \wedge x_3 ?$$

- ❖ Given the following data, what is the right function

- ❖ $\langle(1,1,1), 1\rangle$
- ❖ $\langle(1,0,1), 0\rangle$
- ❖ $\langle(0,1,1), 1\rangle$
- ❖ $\langle(1,1,0), 0\rangle$

Learning Monotone Conjunctions

- ❖ Hypothesis class:

$$f = x_1 ?$$

$$f = x_2 ?$$

$$f = x_1 \wedge x_2 \wedge x_3 ?$$

....

$$f = x_1 \wedge x_2 ?$$

$$f = x_2 \wedge x_3 ?$$

- ❖ Target function in the hindsight

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning protocol

Provide the learning examples

Learn the model



$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Monotone Conjunctions

❖ Supervised Learning

Teacher provides a set of example $(x, f(x))$

- ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
- ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots0,1,1), 1\rangle$
- ❖ $\langle(1,0,1,1,1,0,\dots0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots0,0,1), 1\rangle$
- ❖ $\langle(1,0,1,0,0,0,\dots0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
- ❖ $\langle(0,1,0,1,0,0,\dots0,1,1), 0\rangle$

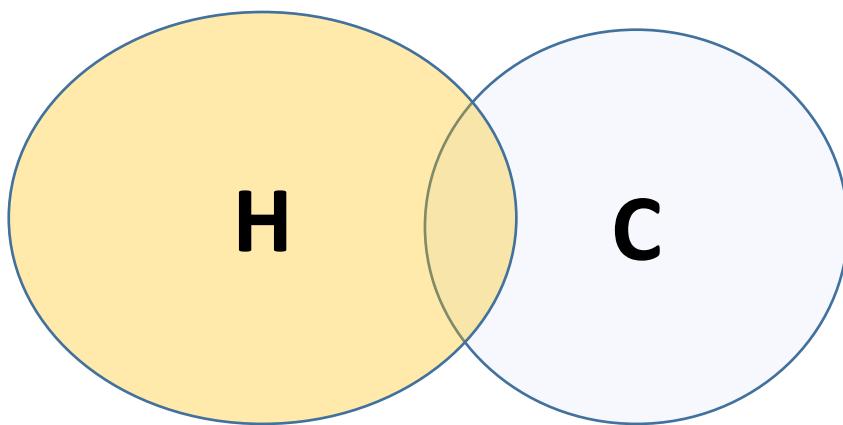
How many examples do we need to find out the right conjunction?

The setup

- ❖ **Instance Space:** X , the set of examples
- ❖ **Concept Space:** C , the set of possible target functions:
 $f \in C$ is the hidden target function
 - ❖ E.g.: all n-conjunctions; all n-dimensional linear functions, ...
- ❖ **Hypothesis Space:** H , the set of possible hypotheses
 - ❖ This is the set that the learning algorithm explores
- ❖ **Training instances:** $S \times \{-1, 1\}$: positive and negative examples of the target concept. (S is a finite subset of X)
$$\langle x_1, f(x_1) \rangle, \langle x_2, f(x_2) \rangle, \dots \langle x_n, f(x_n) \rangle$$
- ❖ **What we want:** A hypothesis $h \in H$ such that $h(x) = f(x)$
 - ❖ A hypothesis $h \in H$ such that $h(x) = f(x)$ for all $x \in S$?
 - ❖ A hypothesis $h \in H$ such that $h(x) = f(x)$ for all $x \in X$?

Concept Space v.s. Hypothesis Space

- ❖ Concept space = Hypothesis space
- ❖ Concept space \subset Hypothesis space
- ❖ Concept space $\not\subseteq$ Hypothesis space



PAC learning

- ❖ A framework for *batch learning*
 - ❖ Train on a fixed training set
 - ❖ Then deploy it in the wild
- ❖ How well will your learning algorithm do on *future* instances?

Probably Approximately Correct (PAC) learning

1. Analyze a simple algorithm for learning conjunctions
2. Define the PAC learnability

Learning monotone Conjunctions

- ❖ Assume both C, H are monotone conjunctions

- ❖ **Supervised Learning**

Teacher provides a set of example $(x, f(x))$

- ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
- ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
- ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
- ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
- ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$

Guess what would be the f ?

- ❖ Assumption: data are sample from a fixed distribution

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$
 - ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
 - ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
 - ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
 - ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
 - ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$
- ❖ Question: can the function be $f = x_1 \wedge x_2$?

Guess what would f look like?

No, 3rd instance is a violation

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$
 - ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
 - ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
 - ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
 - ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
 - ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$
- ❖ Question: can x_2 in the conjunction formulation?

Guess what would f look like?

No, again, 3rd instance is a violation

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$
 - ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
 - ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
 - ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
 - ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
 - ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$
- ❖ Question: can x_1 in the conjunction formulation?

Guess what would f look like?

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$
 - ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
 - ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
 - ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
 - ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
 - ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$
- ❖ Question: can x_1 in the conjunction formulation?

Guess what would f look like?

Yes, why? Whenever the output is 1, x_1 is present

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$
 - ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
 - ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
 - ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
 - ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
 - ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$
- ❖ How to learn a monotone conjunction consistent with these training samples

Start with having all literals in the monotone conjunctions.

Removing literal j if $x_j = 0$ in some positive instances.

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$

- ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
- ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
- ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
- ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
- ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$

Guess what would f look like?

- ❖ Question: Is $x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$ only the function consistent with the data?

No, examples can generated by the function
 $f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$

- ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
- ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
- ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
- ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
- ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$

With the given data, we only learned an *approximation* to the true concept.

- ❖ Question: Is $x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$ the right function?

Perhaps. However, it is possible we have an instance $\langle(0,1,1,1,1,1,\dots,0,1), 1\rangle$ but not in the training data.

Learning monotone Conjunctions

- ❖ Teacher provides a set of example $(x, f(x))$
 - ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
 - ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
 - ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
 - ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
 - ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
 - ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$
- ❖ Question: When the target function is $x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$, but our algorithm returns $x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$?
- ❖ How likely this would happen?

Intuitively, with more training data, it's unlikely we never see $x_1 = 0$ in positive training examples but see it in the test time if data are sampled from the same distribution

“The future will be like the past”:

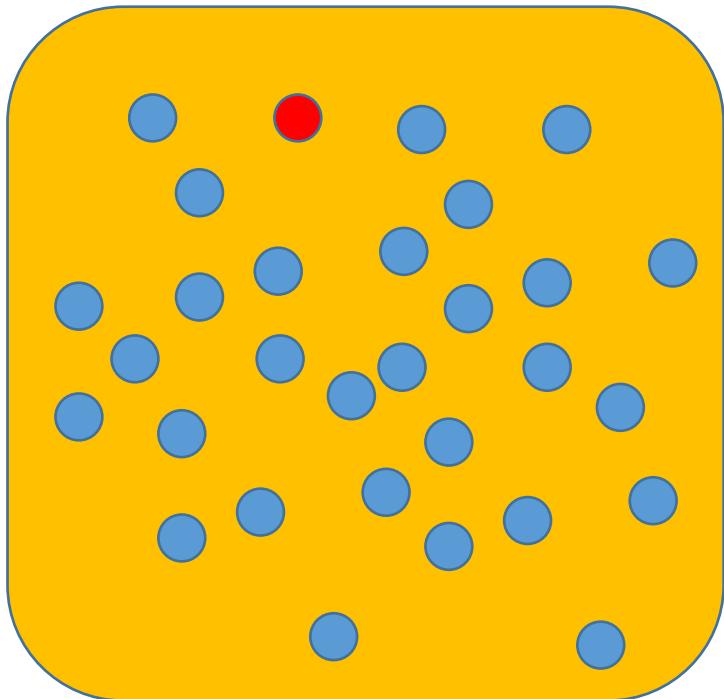
- ❖ We have seen many examples
(drawn according to the distribution D)
 - ❖ Since in all the positive examples x_1 was active,
it is very **likely** that it will be active in future positive
examples
 - ❖ Otherwise, x_1 is active only in a small percentage of
the examples so our error will be small

How likely we never see the examples like $\langle(0,1,1,1,1,1,\dots,0,1), 1\rangle$ to
filter out x_1 but have such cases in the test time

Illustrative Example

❖ Scenario 1: 1 red ball with 29 blue balls

How likely we see a red point in the test time but not training time?



Training set: 0 points

Test set: 3 point

Never see the red ball in the training:

$$\left(\frac{29}{30}\right)^{10} \cong 0.7$$

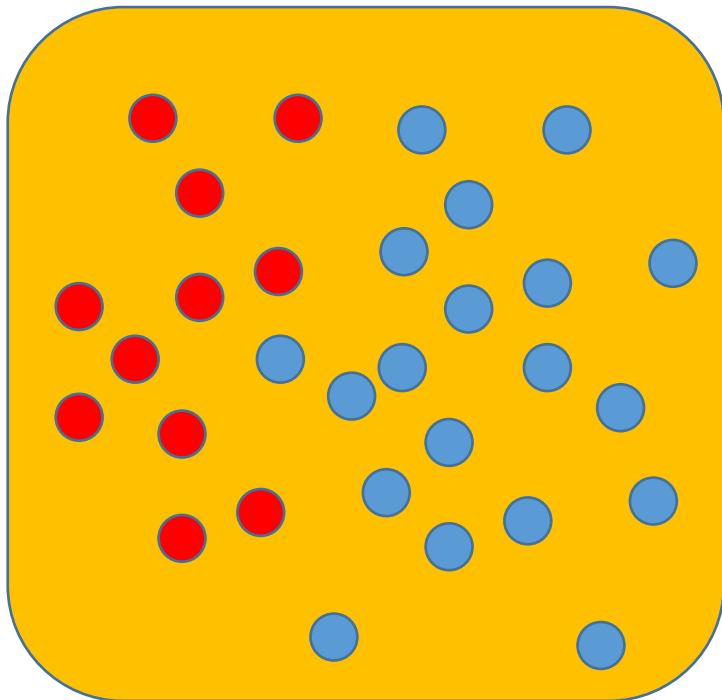
See a red ball in the test time:

$$1 - \left(\frac{29}{30}\right)^3 \cong 0.1$$

Both events happen: ~ 7%

Illustrative Example

❖ Scenario 2: 10 red balls with 20 blue balls



Training set: 10 points

Test set: 3 point

Never see the red ball in the training:

$$\left(\frac{1}{2}\right)^{10} \cong 0.001$$

See a red ball in the test time:

$$1 - \left(\frac{1}{2}\right)^3 \cong 0.875$$

Both events happen: $\sim 0.08\%$

Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

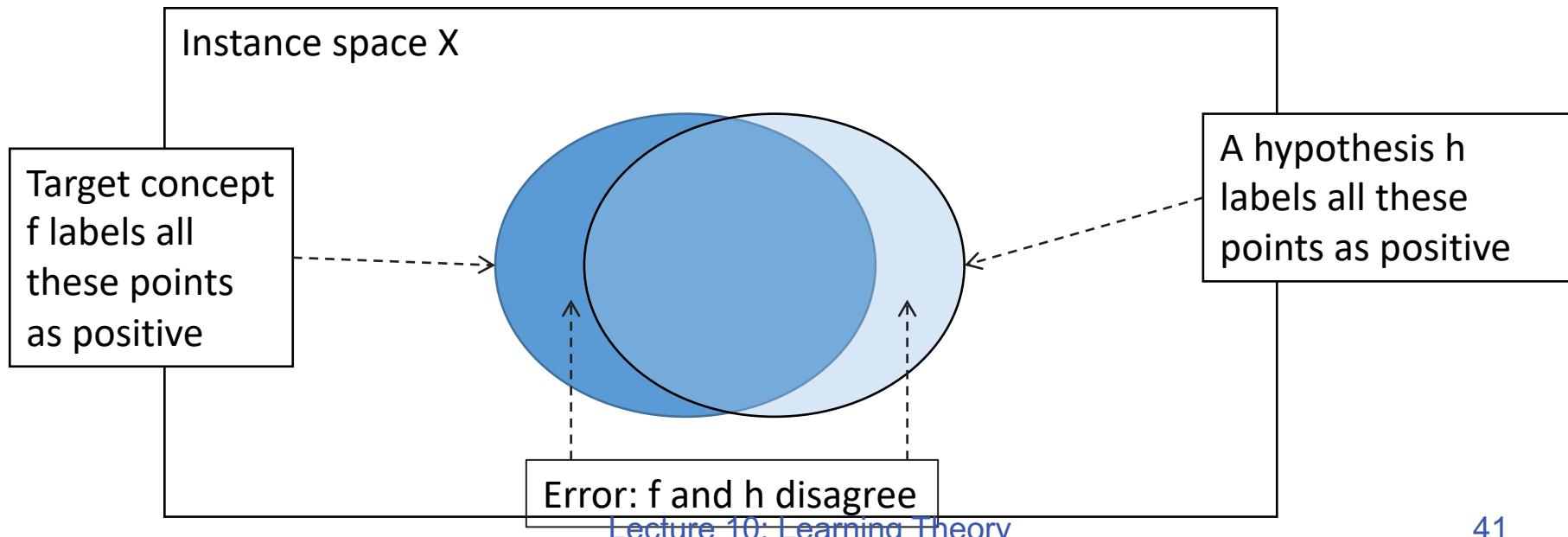
$$\text{err}_D(h) = \Pr_{x \sim D}[h(x) \neq f(x)]$$

Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

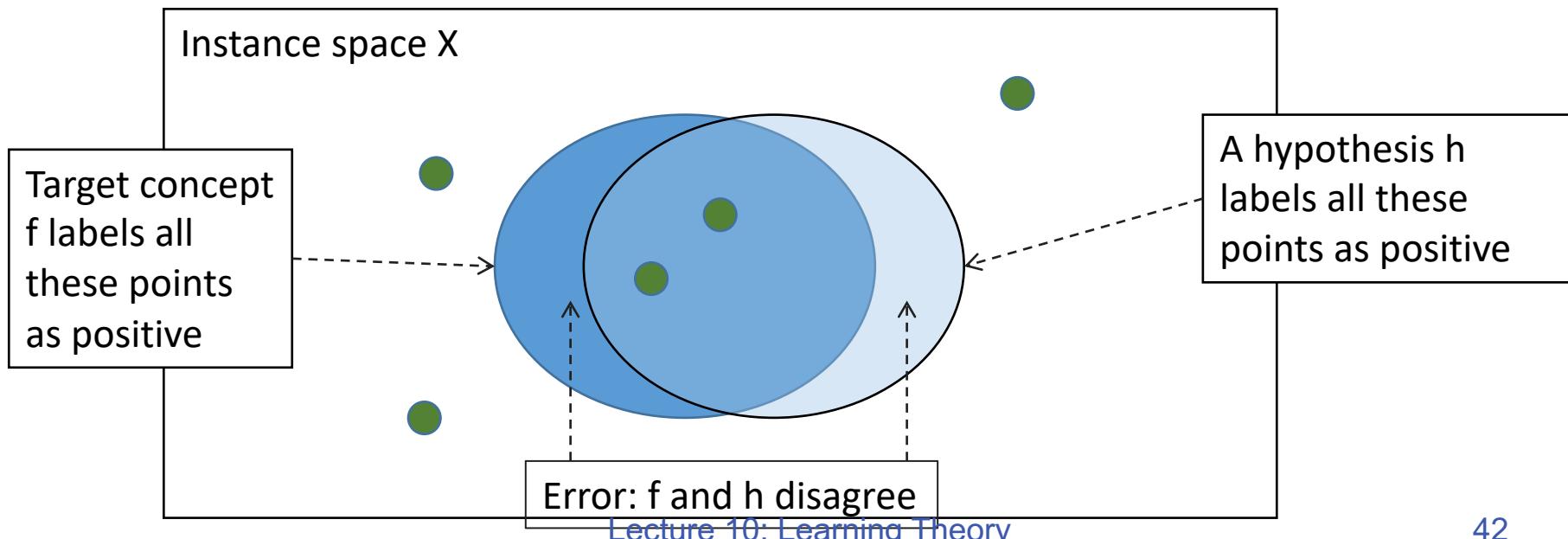
$$\text{err}_D(h) = \Pr_{x \sim D}[h(x) \neq f(x)]$$



Error of a hypothesis

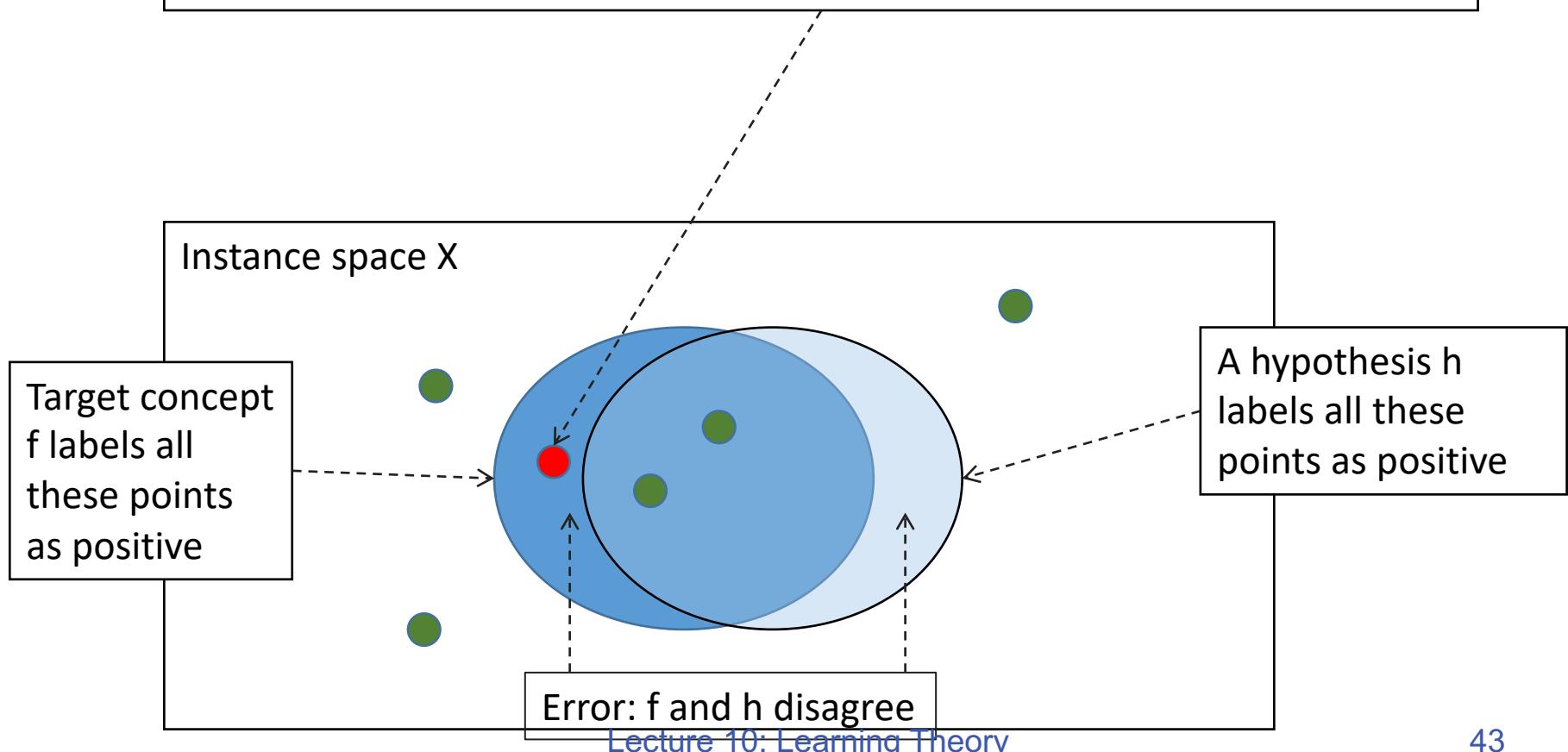
You may have a learned model that is consistent with the training data but still makes mistakes.

- Samples correctly predicted by h
- Samples incorrectly predicted by h



Error of a hypothesis

With the IID sampling assumption, we either have seen this example in the training phase, or it is unlikely to see it in the test time.



Requirements of Learning

- ❖ Cannot expect a learner to learn a concept **exactly**
- ❖ There will generally be multiple concepts consistent with the available data
- ❖ Unseen examples could *potentially* have any label
- ❖ We may misclassify *uncommon* examples that do not show up in the training set

With sufficient number of training samples under the IID assumption, the error rate of a model consistent with the data is likely to be small

What is learnable?

With sufficient number of training samples under the IID assumption, the error rate of a model consistent with the data is likely to be small

PAC Learnability

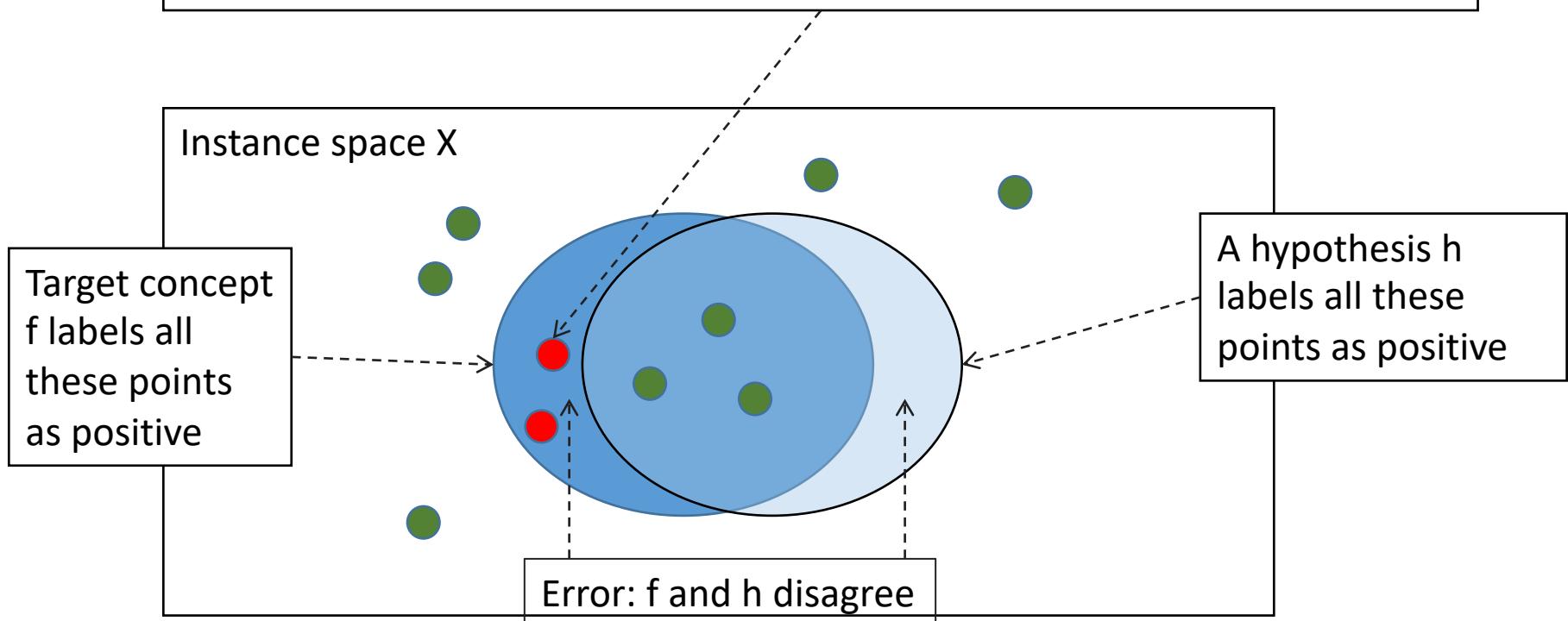
Turing Award: [Leslie Valiant](#).

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is **PAC learnable** by L using H if for all $f \in C$, for all distribution D over X , and fixed $\epsilon > 0$, $\delta < 1$, given m examples sampled i.i.d. according to D , the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ , where m is *polynomial* in $1/\epsilon$, $1/\delta$, n and $\text{size}(H)$

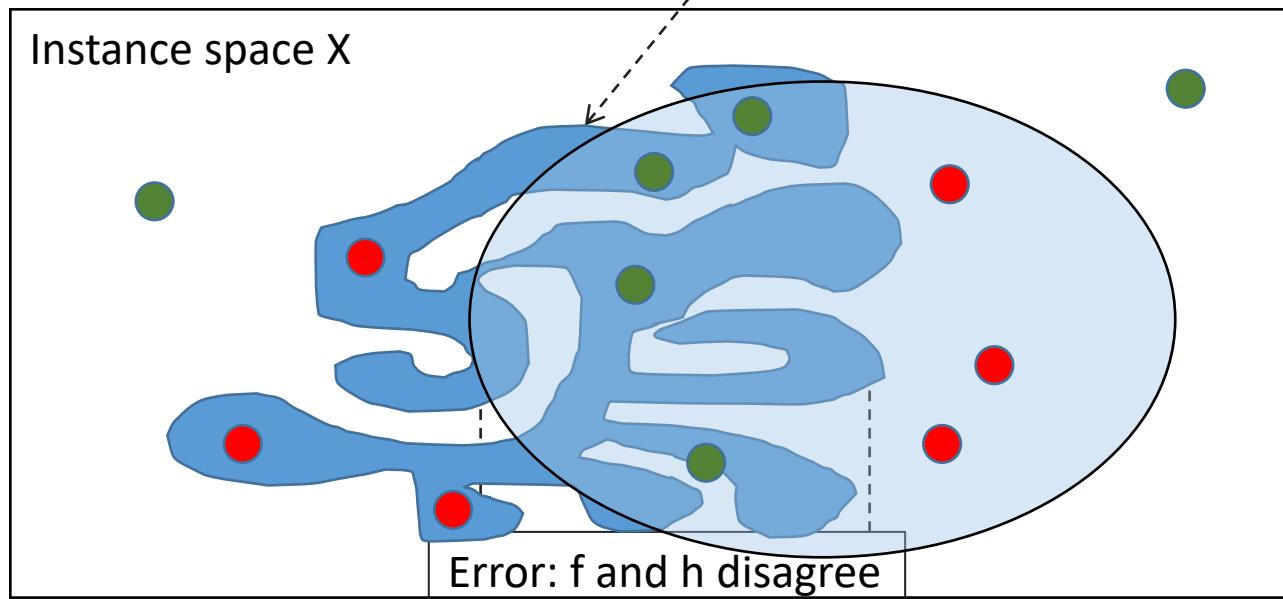
Intuition of PAC Learnability

With the IID sampling assumption, if a concept is reasonable. After, we saw enough samples, it is unlikely to have many these red points



Intuition of PAC Learnability

With the IID sampling assumption, if a concept is too complicated. We need to see exponential number of samples, such that we can rule out those red points



Example: Learning Monotone Conjunctions

The true function $f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Training data

- ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
- ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
- ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
- ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
- ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$

Learning Conjunctions

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Training data

- ❖ $\langle(1, 1, 1, 1, 1, 1, \dots, 1, 1), 1\rangle$
- ❖ $\langle(1, 1, 1, 0, 0, 0, \dots, 0, 0), 0\rangle$
- ❖ $\langle(1, 0, 1, 1, 1, 0, \dots, 0, 1, 1), 1\rangle$
- ❖ $\langle(1, 1, 1, 1, 1, 0, \dots, 0, 0, 1), 1\rangle$
- ❖ $\langle(1, 0, 1, 0, 0, 0, \dots, 0, 1, 1), 0\rangle$
- ❖ $\langle(1, 1, 1, 1, 1, 1, \dots, 0, 1), 1\rangle$
- ❖ $\langle(0, 1, 0, 1, 0, 0, \dots, 0, 1, 1), 0\rangle$

A simple learning algorithm (*Elimination*)

- Discard all negative examples
- Build a conjunction using the features that are common to all positive conjunctions

$$h = \textcolor{red}{x_1} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Clearly this algorithm produces a conjunction that is consistent with the data, that is $\text{err}_S(h) = 0$, if the target function is a monotone conjunction

Learning Conjunctions

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Training data

❖ $\langle(1, 1, 1, 1, 1, 1, 1, \dots, 1, 1), 1\rangle$

❖ $\langle(1, 1, 1, \dots, 1, 1), 1\rangle$

❖ $\langle(1, 0, 1, \dots, 1, 1), 1\rangle$

❖ $\langle(1, 1, 1, \dots, 1, 1), 1\rangle$

❖ $\langle(1, 0, 1, \dots, 1, 1), 1\rangle$

❖ $\langle(1, 1, 1, \dots, 1, 1), 1\rangle$

❖ $\langle(0, 1, 0, \dots, 0, 1), 0\rangle$

Is the true error $\text{err}_D(h)$ guaranteed to be 0?

No.

(*Elimination*)

samples
g the features
positive

100

Clearly this algorithm produces a conjunction that is consistent with the data, that is $\text{err}_S(h) = 0$, if the target function is a monotone conjunction

Learning Conjunctions: Analysis

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Claim 1: Any hypothesis consistent with the training data will only make mistakes on positive future examples

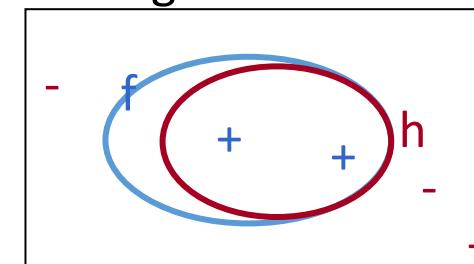
A mistake will occur only if some literal z (in our example x_1) is present in h but not in f

This mistake can cause a positive example to be predicted as negative by h

Specifically: $x_1 = 0, x_3=1, x_4=1, x_5=1, x_{100}=1 \quad y=1$

The reverse situation can never happen

For an example to be predicted as positive in the training set, every relevant literal must have been present (by our algorithm)



Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

Poly in $n, 1/\delta, 1/\epsilon$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

n : # literals

If we see these many training examples, then the algorithm will produce a conjunction that, with high probability, will make few errors

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

Poly in $n, 1/\delta, 1/\epsilon$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Let's prove this assertion

Proof Intuition

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

What kinds of examples would drive a hypothesis to make a mistake and update?

Positive examples, where $x_1 = 0$

h would say true and f would say false

None of these examples appeared during training

Otherwise x_1 would have been eliminated

If they never appeared during training, maybe their appearance in the future would also be rare!

Let's quantify our surprise at seeing such examples

Learning Conjunctions: Analysis

Let $p(z)$ be the probability that, in an example drawn from D , the feature $z = 0$ but the example has a positive label

- ❖ In the training – this is an example that can help we learn the right h
- ❖ In the test – this is an example that make an error

$\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$

$$h = x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$

$\langle(0,1,1,1,1,1,\dots,0,1), 1\rangle$

How likely we find h is wrong

Let $p(z)$ be the probability that, in an example drawn from D , the feature z is absent but the example has a positive label

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

We know that $\text{err}_D(h) \leq \sum_{z \in h} p(z)$

This is a loose bound

Via direct application of the union bound

How likely we find h is wrong

Let $p(z)$ be the probability that, in an example drawn from D , the feature z is absent but the example has a positive label

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

We know that $\text{err}_D(h) \leq \sum_{z \in h} p(z)$

This is a loose bound

Via direct application of the union bound

Union bound

For a set of events, probability that at least one of them happens < the sum of the probabilities of the individual events

Learning Conjunctions: Analysis

n = dimensionality,
 ϵ is a parameter (bound of error)

- ❖ Assume $\text{err}_D(h) > \epsilon$, by $\text{err}_D(h) \leq \sum_{z \in h} p(z)$
- ❖ We call a literal z *bad* if $p(z) > \frac{\epsilon}{n}$
- ❖ Intuitively, a **bad literal** is one that has **a significant probability** of not appearing with a positive example
 - ❖ (And, if it appears in all positive training examples, it can cause errors)
- ❖ We would like to bound the probability

$Pr(\text{Any bad literal survives } m \text{ examples})$

Learning Conjunctions: Analysis

n = dimensionality

- ❖ Call a literal z *bad* if
- ❖ Intuitively, a **bad literal** is one that has a significant probability of not appearing with a positive example
 - ❖ (And, if it appears in all positive training examples, it can cause errors)

What if there are bad literals?

Let z be a bad literal

What is the probability that it will not be eliminated by one training example?

There was one example of this kind

$\langle(1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1\rangle$

Learning Conjunctions: Analysis

$n = \text{dimensionality}$

- ❖ Call a literal z *bad* if
- ❖ Intuitively a **bad literal** is one that has a significant

! "#%& '\$ (\$

- ❖
-) * \$ # + , \$ - " " " \$. ' & \$ / % " 1 ' / % 2 \$ 0 " - 0 \$ 0 % 3 " 4 \$
Wh # + , \$ ' 1 " \$ / % 5 " / # \$ - " " " \$ 0 \$ 2 \$ 0 6 " \$ 0 1 ' % 2 % 2 7 \$ 0 % 3 "

Let z be a bad literal

What is the probability that it will not be eliminated by one training example?

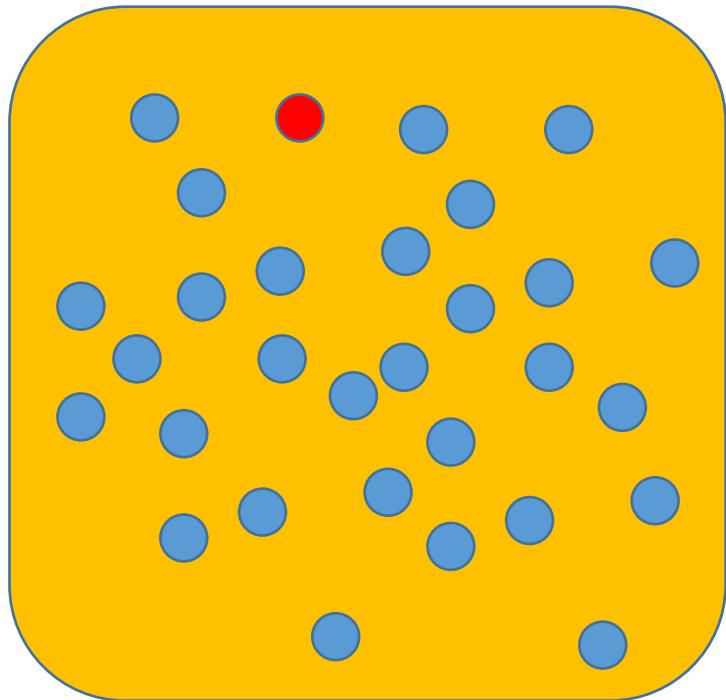
There was one example of this kind

<(1,1,1,1,1,0,...0,1,1), 1>

Illustrative Example

❖ Scenario 1: 1 red ball with 29 blue balls

How likely we see red point in the test time but not training time?



Training set: 10 points

Test set: 3 point

Learning Conjunctions: Analysis

What we know so far:

$$p(z) > \frac{\epsilon}{n} \quad n = \text{dimensionality}$$

$$\Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

Learning Conjunctions: Analysis

What we know so far:

n = dimensionality

$$Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

But say we have m training examples. Then

$$Pr(\text{A bad literal survives } m \text{ examples}) < \left(1 - \frac{\epsilon}{n}\right)^m$$

Learning Conjunctions: Analysis

What we know so far:

n = dimensionality

$$Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

But say we have m training examples. Then

$$Pr(\text{A bad literal survives } m \text{ examples}) < \left(1 - \frac{\epsilon}{n}\right)^m$$

There are at most n bad literals. So

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$$

Learning Conjunctions: Analysis

$$\Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some δ

Learning Conjunctions: Analysis

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some δ

That is, we want

$$n \left(1 - \frac{\epsilon}{n}\right)^m < \delta$$

We know that $1 - x < e^{-x}$. So it is sufficient to require

$$ne^{-\frac{m\epsilon}{n}} < \delta$$

Learning Conjunctions: Analysis

$Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some \pm

That is, we want $n \left(1 - \frac{\epsilon}{n}\right)^m < \delta$

We know that $1 - x < e^{-x}$. So it is sufficient to require $ne^{-\frac{m\epsilon}{n}} < \delta$

Or equivalently,

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Probably Approximately Correct (PAC) learning

1. Analyze a simple algorithm for learning conjunctions
2. Define the PAC model of learning

Formulating the theory of prediction

All the notation we have so far on one slide

In the general case, we have

- ❖ X : instance space, Y : output space = $\{+1, -1\}$
- ❖ D : an unknown distribution over X
- ❖ f : an unknown target function $X \rightarrow Y$, taken from a concept class C
- ❖ h : a hypothesis function $X \rightarrow Y$ that the learning algorithm selects from a hypothesis class H
- ❖ S : a set of m training examples drawn from D , labeled with f
- ❖ $\text{err}_D(h)$: The true error of any hypothesis h
- ❖ $\text{err}_S(h)$: The empirical error or training error or observed error of h

Theoretical questions

- ❖ Can we describe or bound the true error (err_D) given the empirical error (err_S)?
- ❖ Is a concept class C learnable?
- ❖ Is it possible to learn C using only the functions in H using the supervised protocol?
- ❖ How many examples does an algorithm need to guarantee good performance?

Requirements of Learning

- ❖ Cannot expect a learner to learn a concept **exactly**
 - ❖ There will generally be multiple concepts consistent with the available data
 - ❖ Unseen examples could *potentially* have any label
 - ❖ We may misclassify *uncommon* examples that do not show up in the training set

Requirements of Learning

- ❖ Cannot expect a learner to learn a concept **exactly**
 - ❖ There will generally be multiple concepts consistent with the available data
 - ❖ Unseen examples could *potentially* have any label
 - ❖ We may misclassify *uncommon* examples that do not show up in the training set
- ❖ Cannot always expect to learn a **close approximation** to the target concept
 - ❖ Sometimes the training set will not be representative

Probably approximately correctness

- ❖ The only realistic expectation of a good learner is that **with high probability** it will learn a **close approximation** to the target concept
- ❖ In Probably Approximately Correct (PAC) learning, one requires that
 - ❖ given small parameters ϵ and δ ,
 - ❖ With probability at least $1 - \epsilon$, a learner produces a hypothesis with error at most δ
- ❖ The reason we can hope for this is the ***consistent distribution assumption***

PAC Learnability

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is **PAC learnable** by L using H if for all $f \in C$, for all distribution D over X , and fixed $\epsilon > 0$, $\delta < 1$, given m examples sampled i.i.d. according to D , the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ , where m is **polynomial** in $1/\epsilon$, $1/\delta$, n and $\text{size}(H)$

example: conjunction: $m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$

efficiently learnability

- ❖ The concept class C is *efficiently learnable* if L can produce the hypothesis in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and $\text{size}(H)$

PAC Learnability

- ❖ We impose two limitations
- ❖ Polynomial *sample complexity* (information theoretic constraint)
 - ❖ Is there enough information in the sample to distinguish a hypothesis h that approximate f ?
- ❖ Polynomial *time complexity* (computational complexity)
 - ❖ Is there an efficient algorithm that can process the sample and produce a good hypothesis h ?

Worst Case definition: the algorithm must meet its accuracy

- ❖ for every distribution (The distribution free assumption)
- ❖ for every target function f in the class C

Example: Learning Conjunctions

Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

m is *polynomial* in $1/\epsilon$, $1/\delta$, n and size(H)

A general result

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is **consistent** with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$$

1. Expecting lower error increases sample complexity (i.e more examples needed for the guarantee)

3. If we want a higher confidence in the classifier we will produce, sample complexity will be higher.

2. If we have a larger hypothesis space, then we will make learning harder (i.e higher sample complexity)

A general result

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is **consistent** with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$$

It expresses a preference towards smaller hypothesis spaces

Next question: What if size(H) is infinity?

Complicated/larger hypothesis spaces are not necessarily bad. But simpler ones are unlikely to fool us by being consistent with many examples!

Lecture 11: Computational Learning Theory VC Dimension Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

What concept is learnable?

What is learnable

Given 10 variables, how many of them?

❖ Monotone Conjunctions?

$$f = x_1 ?$$

$$f = x_2 ?$$

$$f = x_1 \wedge x_2 \wedge x_3 ?$$

$$f = x_1 \wedge x_2 ?$$

$$f = x_2 \wedge x_3 ?$$

❖ Disjunctions?

$$f = x_1 \vee x_2 ?$$

$$f = \neg x_1 \vee x_2 ?$$

$$f = x_1 \vee x_3 \vee x_4 ?$$

❖ Boolean function?

$$f = x_1 \vee x_2 \wedge x_5 ?$$

Recap: Hypothesis Space (1)

Complete Ignorance:

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

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	?

Recap: Hypothesis Space (1)

To identify the right function, it needs to see all instances

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	X1	X2	X3	X4	y
0	0	0	0	0	?
0	0	0	0	1	?
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	?
1	0	0	0	0	?
1	0	0	0	1	1
1	0	1	0	0	?
1	0	1	1	1	?
1	1	0	0	0	0
1	1	0	1	1	?
1	1	1	0	1	?
1	1	1	1	0	?
1	1	1	1	1	?

Hypothesis Space (3)

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"

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

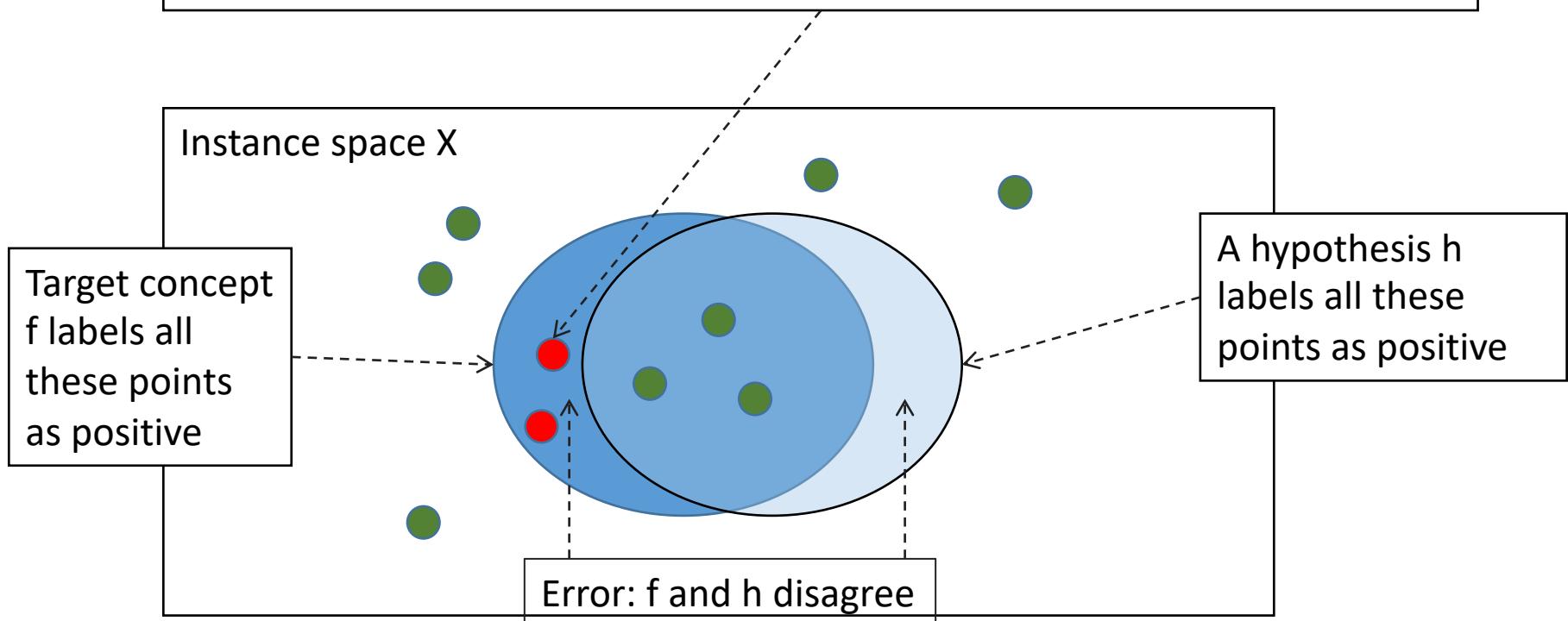
Notation: 2 variables from the set on the left. **Value:** Index of the counterexample.

variables	1-of	2-of	3-of	4-of	variables	1-of	2-of	3-of	4-of
{X1}	3	-	-	-	{X2, X4}	2	3	-	-
{X2}	2	-	-	-	{X3, X4}	4	4	-	-
{X3}	1	-	-	-	{X1, X2, X3}	1	3	3	-
{X4}	7	-	-	-	{X1, X2, X4}	2	3	3	-
{X1, X2}	2	3	-	-	{X1, X3, X4}	1	***	3	-
{X1, X3}	1	3	-	-	{X2, X3, X4}	1	5	3	-
{X1, X4}	6	3	-	-	{X1, X2, X3, X4}	1	5	3	3
{X2, X3}	2	3	-	-					

Found a consistent hypothesis.

Intuition of PAC Learnability

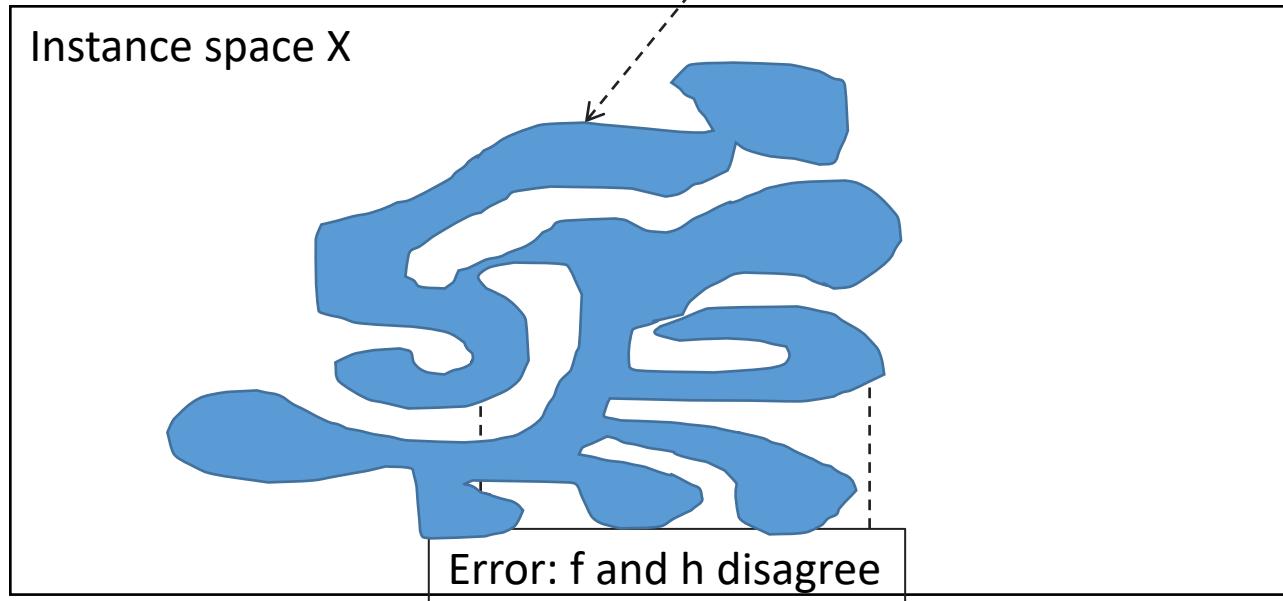
With the IID sampling assumption, if a concept is reasonable. After, we saw enough samples, it is unlikely to have many these red points



For example, considering the hypothesis space that contains all monotone conjunctions

Intuition of PAC Learnability

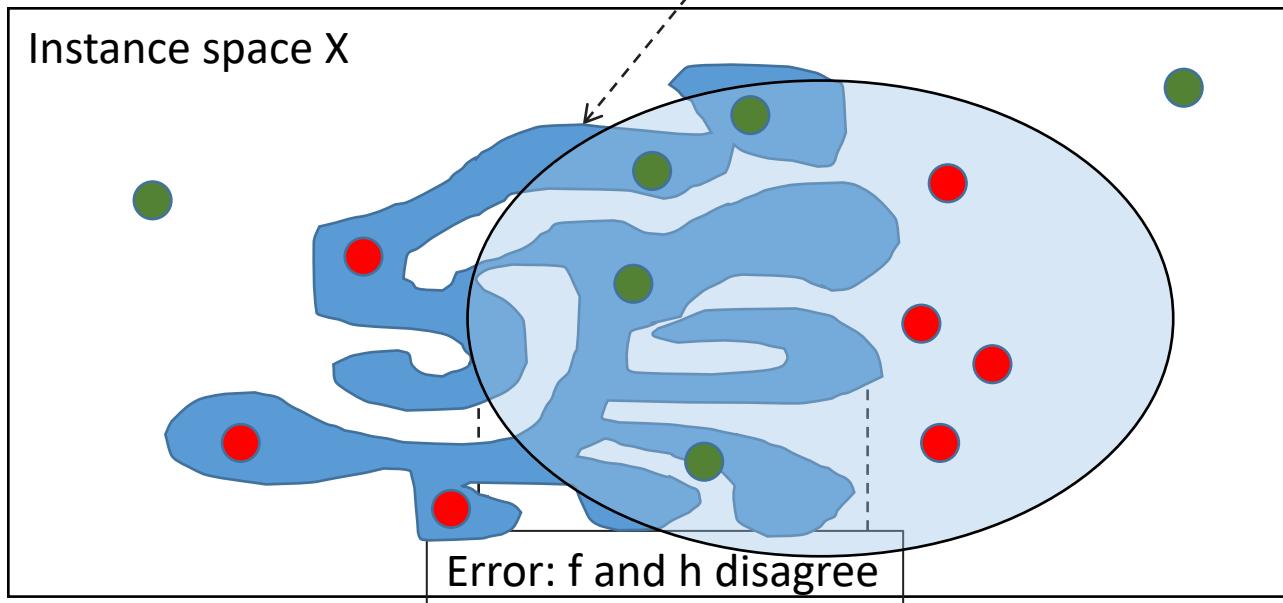
With the IID sampling assumption, if a concept is too complicated. We need to see exponential number of samples, such that we can rule out those red points



For example, considering the hypothesis space that contains all Boolean functions

Intuition of PAC Learnability

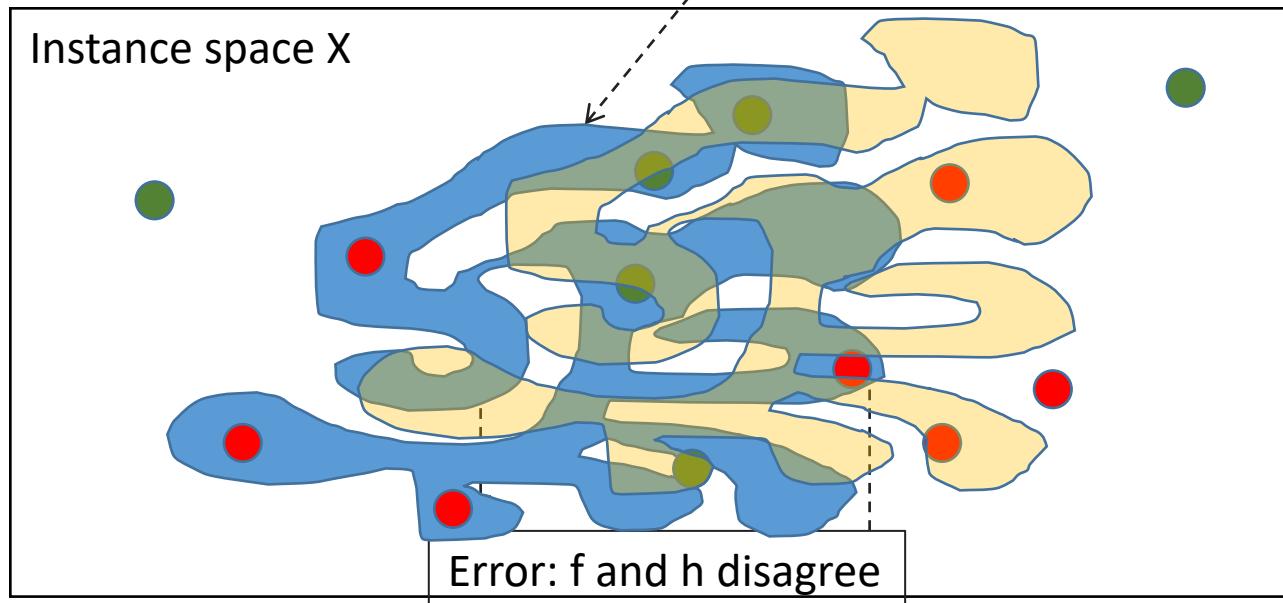
With the IID sampling assumption, if a concept is too complicated. We need to see exponential number of samples, such that we can rule out those red points



For example, considering the hypothesis space that contains all Boolean functions

Intuition of PAC Learnability

With the IID sampling assumption, if a concept is too complicated. We need to see exponential number of samples, such that we can rule out those red points



For example, considering the hypothesis space that contains all Boolean functions

What is learnable

Given 10 variables, how many of them?

❖ Concept = Monotone Conjunctions?

$$f = x_1 ?$$

$$f = x_2 ?$$

$$f = x_1 \wedge x_2 \wedge x_3 ?$$

$$f = x_1 \wedge x_2 ?$$

$$f = x_2 \wedge x_3 ?$$

❖ Concept = Disjunctions?

$$f = x_1 \vee x_2 ?$$

$$f = \neg x_1 \vee x_2 ?$$

$$f = x_1 \vee x_3 \vee x_4 ?$$

❖ Concept = All Boolean function?

$$f = x_1 \vee x_2 \wedge x_5 ?$$

Learning Conjunctions

Learning monotone Conjunctions

❖ Supervised Learning

Teacher provides a set of example $(x, f(x))$

- ❖ $\langle(1,1,1,1,1,1,\dots,1,1), 1\rangle$
- ❖ $\langle(1,1,1,0,0,0,\dots,0,0), 0\rangle$
- ❖ $\langle(1,0,1,1,1,0,\dots,0,1,1), 1\rangle$
- ❖ $\langle(1,1,1,1,1,0,\dots,0,0,1), 1\rangle$
- ❖ $\langle(1,0,1,0,0,0,\dots,0,1,1), 0\rangle$
- ❖ $\langle(1,1,1,1,1,1,\dots,0,1), 1\rangle$
- ❖ $\langle(0,1,0,1,0,0,\dots,0,1,1), 0\rangle$

Guess what would f look like?

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

❖ Assumption: data are sample from a fixed distribution

Learning Conjunctions -- Algorithm

Training data

- ❖ $\langle(1, 1, 1, 1, 1, 1, \dots, 1, 1), 1\rangle$
- ❖ $\langle(1, 1, 1, 0, 0, 0, \dots, 0, 0), 0\rangle$
- ❖ $\langle(1, 0, 1, 1, 1, 0, \dots, 0, 1, 1), 1\rangle$
- ❖ $\langle(1, 1, 1, 1, 1, 0, \dots, 0, 0, 1), 1\rangle$
- ❖ $\langle(1, 0, 1, 0, 0, 0, \dots, 0, 1, 1), 0\rangle$
- ❖ $\langle(1, 1, 1, 1, 1, 1, \dots, 0, 1), 1\rangle$
- ❖ $\langle(0, 1, 0, 1, 0, 0, \dots, 0, 1, 1), 0\rangle$

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

A simple learning algorithm (*Elimination*)

- Discard all negative examples
- Build a conjunction using the features that are common to all positive conjunctions

$$h = x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Positive examples **eliminate** irrelevant features

Learning Conjunctions: Analysis

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Claim 1: Any hypothesis consistent with the training data will only make mistakes on positive future examples

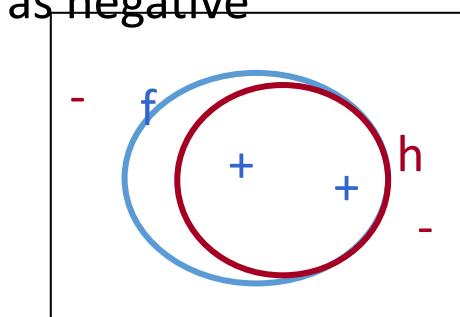
A mistake will occur only if some literal z (in our example x_1) is present in h but not in f

This mistake can cause a positive example to be predicted as negative by h

Specifically: $x_1 = 0, x_3=1, x_4=1, x_5=1, x_{100}=1 \quad y=1$

The reverse situation can never happen

For an example to be predicted as positive in the training set, every relevant literal must have been present (by our algorithm)



Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

Poly in $n, 1/\delta, 1/\epsilon$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

n : # literals

If we see these many training examples, then the algorithm will produce a conjunction that, with high probability, will make few errors

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

Poly in $n, 1/\delta, 1/\epsilon$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Let's prove this assertion

Proof Intuition

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

What kinds of examples would drive a hypothesis to make a mistake and update?

Positive examples, where $x_1 = 0$

h would say true and f would say false

None of these examples appeared during training

Otherwise x_1 would have been eliminated

If they never appeared during training, maybe their appearance in the future would also be rare!

Let's quantify our surprise at seeing such examples

Learning Conjunctions: Analysis

Let $p(z)$ be the probability that, in an example drawn from D , the feature $z = 0$ but the example has a positive label

- ❖ In the training – this is an example that can help we learn the right h
- ❖ In the test – this is an example that make an error

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100} \quad h = \textcolor{red}{x_1} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

<(1,1,1,1,1,1,...,1,1), 1>
<(1,0,1,1,1,0,...0,1,1), 1>
<(1,1,1,1,1,0,...0,0,1), 1>
<(1,1,1,1,1,1,...,0,1), 1>
<(0,1,1,1,1,1,...,0,1), 1>

Learning Conjunctions: Analysis

Let $p(z)$ be the probability that, in an example drawn from D , the feature $z=0$ but the example has a positive label

- ❖ i.e., after training is done, $p(z)$ is the probability that in a randomly drawn example, the literal z causes a mistake

$$f = x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = \textcolor{red}{x_1} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$\langle(0, 1, 1, 1, 1, 0, \dots, 0, 1, \textcolor{blue}{1}), 1\rangle$

$p(x_1)$: Probability that this situation occurs

How likely we find h is wrong

Let $p(z)$ be the probability that, in an example drawn from D , the feature z is absent but the example has a positive label

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

We know that $err_D(h) \leq \sum_{z \in h} p(z)$

This is a loose bound

Direct application of the union bound

Union bound

For a set of events, probability that at least one of them happens < the sum of the probabilities of the individual events

Learning Conjunctions: Analysis

n = dimensionality,
 ϵ is a parameter (bound of error)

- ❖ If we hope the error is less than ϵ , we hope there is no “bad literals”, where we call a literal z bad if $p(z) > \frac{\epsilon}{n}$

If there are no bad literals, then $\text{err}_D(h) \leq \epsilon$

- ❖ Because $p(z) \leq \frac{\epsilon}{n}$ and $\text{err}_D(h) \leq \sum_{z \in h} p(z)$

Intuitively, a **bad literal** is one that has a significant probability of not appearing with a positive example

That is we aim to bound

$$\Pr(\text{Any bad literal survives } m \text{ examples})$$

Learning Conjunctions: Analysis

n = dimensionality

- ❖ Call a literal z *bad* if
- ❖ Intuitively, a **bad literal** is one that has a significant probability of not appearing with a positive example
 - ❖ (And, if it appears in all positive training examples, it can cause errors)

What if there are bad literals?

Let z be a bad literal

What is the probability that it will not be eliminated by one training example?

There was one example of this kind

$\langle(1, 1, 1, 1, 1, 0, \dots, 0, 1, 1), 1\rangle$

Learning Conjunctions: Analysis

$n = \text{dimensionality}$

- ❖ Call a literal z *bad* if
- ❖ Intuitively a **bad literal** is one that has a significant

! "#%& '\$ (\$

- ❖
-) * \$ # + , \$ - " " " \$. ' & \$ / % " 1 ' / % 2 \$ 0 " - 0 \$ 0 % 3 " 4 \$
Wh # + , \$ ' 1 " \$ / % 5 " / # \$ - " " " \$ 0 \$ / % 2 \$ 0 6 " \$ 0 1 ' % 2 % 2 7 \$ 0 % 3 "

Let z be a bad literal

What is the probability that it will not be eliminated by one training example?

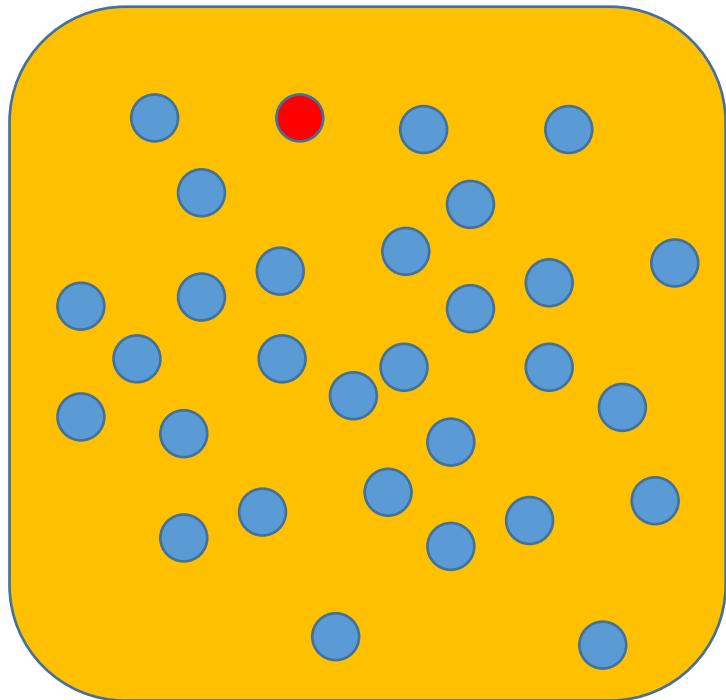
There was one example of this kind

<(1,1,1,1,1,0,...0,1,1), 1>

Illustrative Example

❖ Scenario 1: 1 red ball with 29 blue balls

How likely we see red point in the test time but not training time?



Training set: 10 points

Test set: 3 point

Learning Conjunctions: Analysis

What we know so far:

$$p(z) > \frac{\epsilon}{n} \quad n = \text{dimensionality}$$

$$\Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

Learning Conjunctions: Analysis

What we know so far: $p(z) > \frac{\epsilon}{n}$ $n = \text{dimensionality}$

$$Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

But say we have m training examples. Then

$$Pr(\text{A bad literal survives } m \text{ examples}) < \left(1 - \frac{\epsilon}{n}\right)^m$$

Learning Conjunctions: Analysis

What we know so far:

n = dimensionality

$$Pr(\text{A bad literal is not eliminated by one example}) < 1 - \frac{\epsilon}{n}$$

But say we have m training examples. Then

$$Pr(\text{A bad literal survives } m \text{ examples}) < \left(1 - \frac{\epsilon}{n}\right)^m$$

There are at most n bad literals. So

$$Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$$

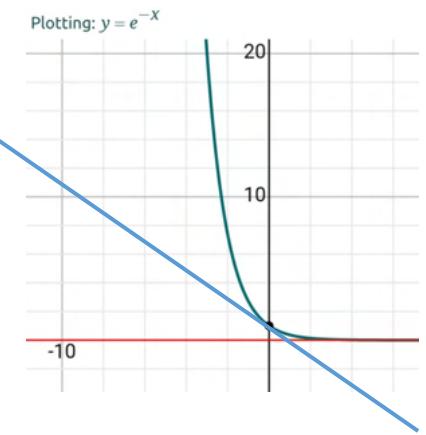
Learning Conjunctions: Analysis

$Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some δ

That is, we want $n \left(1 - \frac{\epsilon}{n}\right)^m < \delta$



We know that $1 - x < e^{-x}$. So it is sufficient to require

$$ne^{-\frac{m\epsilon}{n}} < \delta$$

Learning Conjunctions: Analysis

$Pr(\text{Any bad literal survives } m \text{ examples}) < n \left(1 - \frac{\epsilon}{n}\right)^m$

We want this probability to be small

Why? So that we can choose enough training examples so that the probability that any z survives all of them is less than some \pm

That is, we want $n \left(1 - \frac{\epsilon}{n}\right)^m < \delta$

We know that $1 - x < e^{-x}$. So it is sufficient to require $ne^{-\frac{m\epsilon}{n}} < \delta$

Or equivalently,

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

Probably Approximately Correct (PAC) learning

1. Analyze a simple algorithm for learning conjunctions
2. Define the PAC model of learning

Formulating the theory of prediction

All the notation we have so far on one slide

In the general case, we have

- ❖ X : instance space, Y : output space = $\{+1, -1\}$
- ❖ D : an unknown distribution over X
- ❖ f : an unknown target function $X \rightarrow Y$, taken from a concept class C
- ❖ h : a hypothesis function $X \rightarrow Y$ that the learning algorithm selects from a hypothesis class H
- ❖ S : a set of m training examples drawn from D , labeled with f
- ❖ $\text{err}_D(h)$: The true error of any hypothesis h
- ❖ $\text{err}_S(h)$: The empirical error or training error or observed error of h

Theoretical questions

- ❖ Can we describe or bound the true error (err_D) given the empirical error (err_S)?
- ❖ Is a concept class C learnable?
- ❖ Is it possible to learn C using only the functions in H using the supervised protocol?
- ❖ How many examples does an algorithm need to guarantee good performance?

Requirements of Learning

- ❖ Cannot expect a learner to learn a concept **exactly**
 - ❖ There will generally be multiple concepts consistent with the available data
 - ❖ Unseen examples could *potentially* have any label
 - ❖ We may misclassify *uncommon* examples that do not show up in the training set

Requirements of Learning

- ❖ Cannot expect a learner to learn a concept **exactly**
 - ❖ There will generally be multiple concepts consistent with the available data
 - ❖ Unseen examples could *potentially* have any label
 - ❖ We may misclassify *uncommon* examples that do not show up in the training set
- ❖ Cannot always expect to learn a **close approximation** to the target concept
 - ❖ Sometimes the training set will not be representative

Probably approximately correctness

- ❖ The only realistic expectation of a good learner is that **with high probability** it will learn a **close approximation** to the target concept
- ❖ In Probably Approximately Correct (PAC) learning, one requires that
 - ❖ given small parameters ϵ and δ ,
 - ❖ With probability at least $1 - \epsilon$, a learner produces a hypothesis with error at most δ
- ❖ The reason we can hope for this is the ***consistent distribution assumption***

PAC Learnability

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is **PAC learnable** by L using H if for all $f \in C$, for all distribution D over X , and fixed $\epsilon > 0$, $\delta < 1$, given m examples sampled i.i.d. according to D , the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ , where m is **polynomial** in $1/\epsilon$, $1/\delta$, n and $\text{size}(H)$

example: conjunction: $m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$

efficiently learnability

- ❖ The concept class C is *efficiently learnable* if L can produce the hypothesis in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and $\text{size}(H)$

PAC Learnability

- ❖ We impose two limitations
- ❖ Polynomial *sample complexity* (information theoretic constraint)
 - ❖ Is there enough information in the sample to distinguish a hypothesis h that approximate f ?
- ❖ Polynomial *time complexity* (computational complexity)
 - ❖ Is there an efficient algorithm that can process the sample and produce a good hypothesis h ?

Worst Case definition: the algorithm must meet its accuracy

- ❖ for every distribution (The distribution free assumption)
- ❖ for every target function f in the class C

Example: Learning Conjunctions

Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

m is *polynomial* in $1/\epsilon$, $1/\delta$, n and size(H)

A general result

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is **consistent** with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$$

1. Expecting lower error increases sample complexity (i.e more examples needed for the guarantee)

3. If we want a higher confidence in the classifier we will produce, sample complexity will be higher.

2. If we have a larger hypothesis space, then we will make learning harder (i.e higher sample complexity)

A general result

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is **consistent** with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$$

It expresses a preference towards smaller hypothesis spaces

Next question: What if $\text{size}(H)$ is infinity?

Complicated/larger hypothesis spaces are not necessarily bad. But simpler ones are unlikely to fool us by being consistent with many examples!

Lecture 12: VC Dimension Multi-Class Classification

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Learning Conjunctions

Learning Conjunctions: Analysis

Theorem: Suppose we are learning a conjunctive concept with n dimensional Boolean features using m training examples. If

$$m > \frac{n}{\epsilon} \left(\log(n) + \log\left(\frac{1}{\delta}\right) \right)$$

then, with probability $> 1 - \delta$, the error of the learned hypothesis $\text{err}_D(h)$ will be less than ϵ .

PAC Learnability

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is **PAC learnable** by L using H if for all $f \in C$, for all distribution D over X , and fixed $\epsilon > 0$, $\delta < 1$, given m examples sampled i.i.d. according to D , the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ , where m is **polynomial** in $1/\epsilon$, $1/\delta$, n and $\text{size}(H)$

example: conjunction: $m > \frac{n}{\epsilon} \left(\log(n) + \log \left(\frac{1}{\delta} \right) \right)$

A general result

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is **consistent** with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$$

It expresses a preference towards smaller hypothesis spaces.

Complicated/larger hypothesis spaces are not necessarily bad. But simpler ones are unlikely to fool us by being consistent with many examples!

Example Disjunction

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is consistent with a training set of size m will have an error $< \epsilon$ on future examples if

$$m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$$

Size of hypothesis class for disjunction class $|H| = 3^n$, so a sufficient number of example to learn the disjunction concept is

$$m > \frac{1}{\epsilon} \left(n \ln 3 + \ln \frac{1}{\delta} \right)$$

$$\delta = \epsilon = 0.05, n = 10 \Rightarrow m > 280$$

$$\delta = 0.01 \epsilon = 0.05, n = 10 \Rightarrow m > 312$$

$$\delta = \epsilon = 0.01, n = 10 \Rightarrow m > 1,625$$

$$\delta = \epsilon = 0.01, n = 50 \Rightarrow m > 5,954$$

Example Arbitrary Boolean Function

Let H be any hypothesis space.

With probability $1 - \delta$ a hypothesis $h \rightarrow H$ that is **consistent** with a training set of size m will have an error $< \epsilon$ on future examples if $m > \frac{1}{\epsilon} \left(\ln(|H|) + \ln \frac{1}{\delta} \right)$

Size of hypothesis class for disjunction class $|H| = 2^{2^n}$, so a sufficient number of example to learn the disjunction concept is

$$m > \frac{1}{\epsilon} \left(2^n \ln 2 + \ln \frac{1}{\delta} \right)$$

$$\delta = \epsilon = 0.05, n = 10 \Rightarrow m > 14,256$$

$$\delta = \epsilon = 0.05, n = 50 \Rightarrow m > 1.5 \times 10^{16}$$

This lecture: Computational Learning Theory

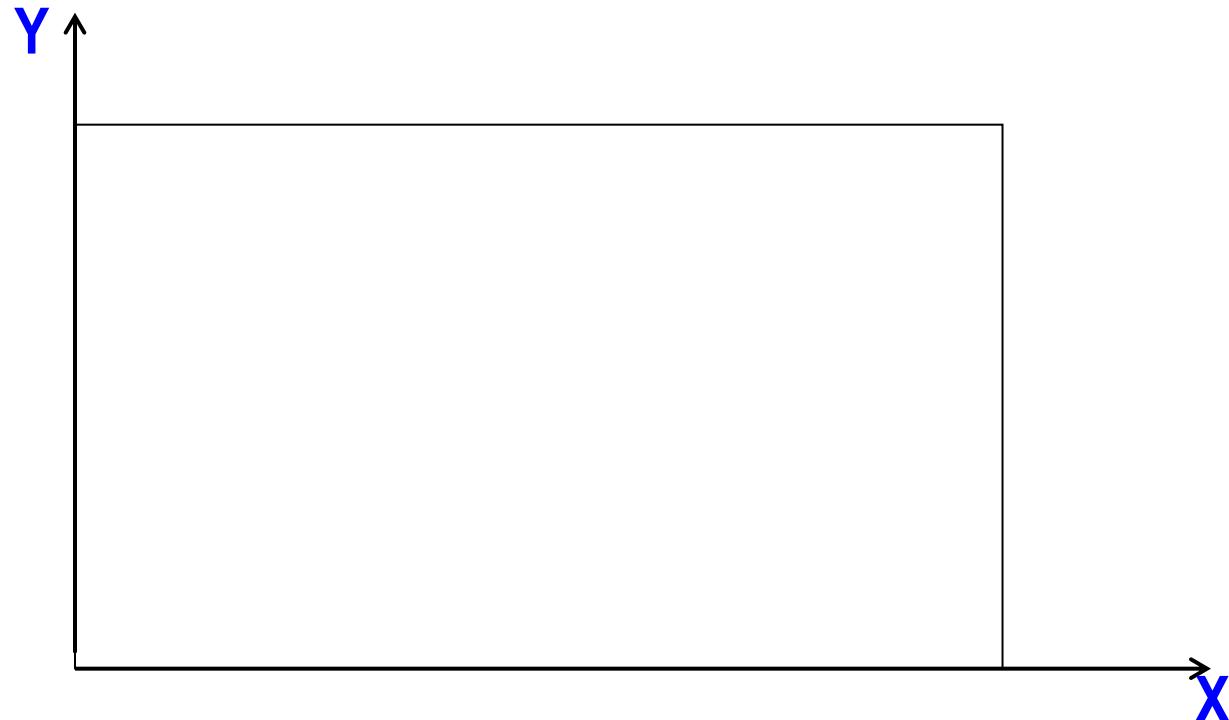
- ❖ The Theory of Generalization
- ❖ Probably Approximately Correct (PAC) learning
- ❖ Shattering and the VC dimension

Vapnik-Chervonenkis dimension

- ❖ The Vapnik-Chervonenkis dimension (**VC dimension**) provides such a measure
 - ❖ “What is the expressive *capacity* of a set of functions?”
- ❖ Analogous to $|H|$, there are bounds for sample complexity using **VC(H)**

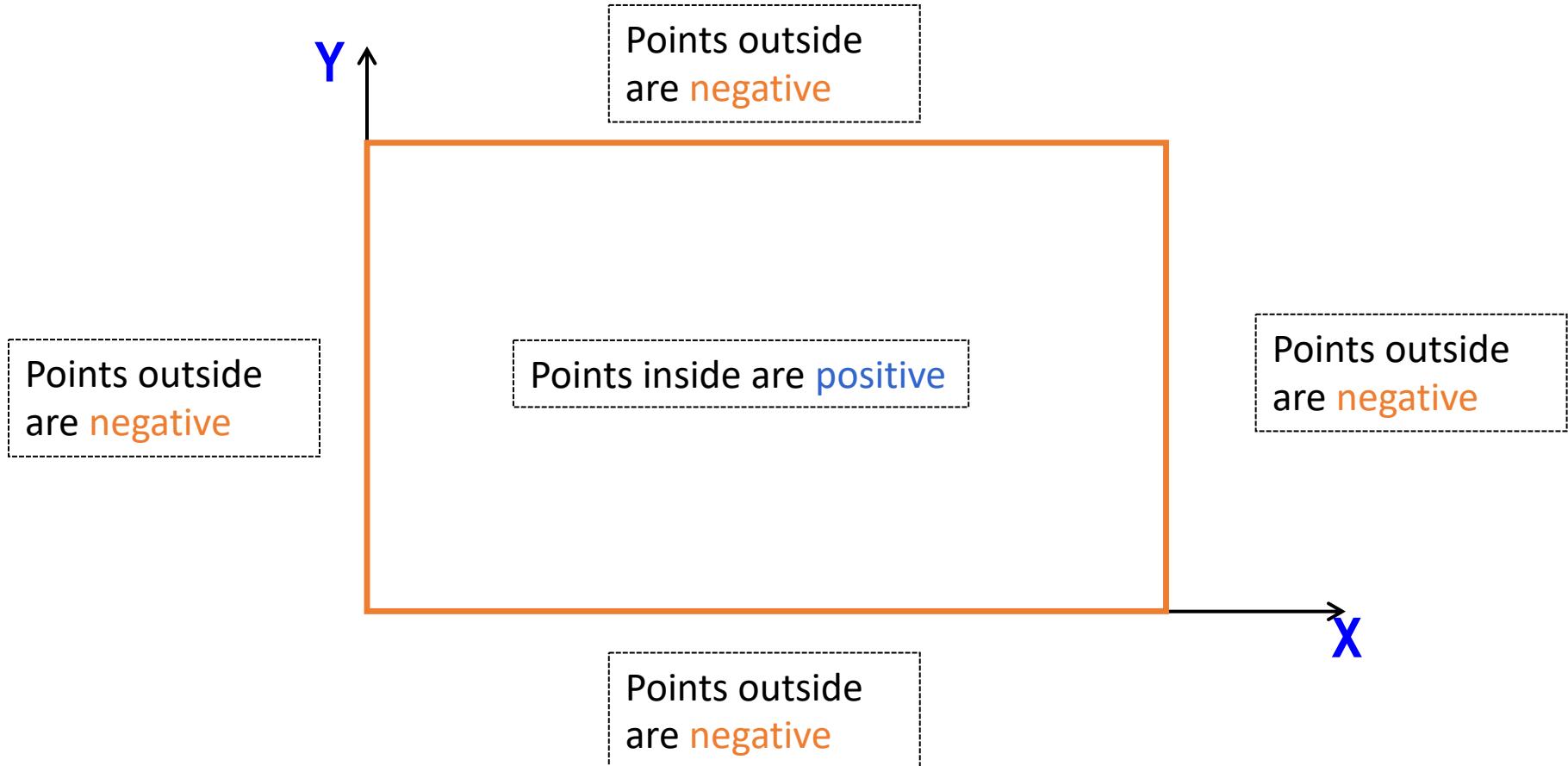
Learning Rectangles

Assume the target concept is an axis parallel rectangle



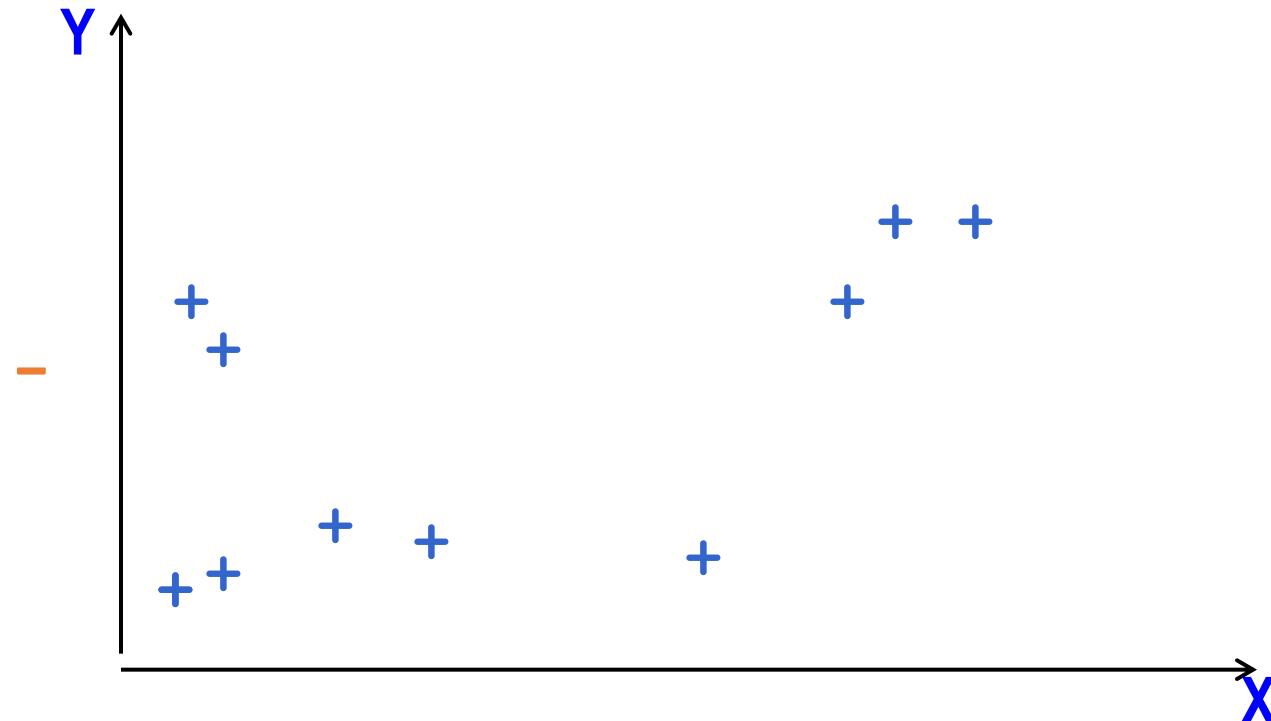
Learning Rectangles

Assume the target concept is an axis parallel rectangle



Learning Rectangles

Assume the target concept is an axis parallel rectangle

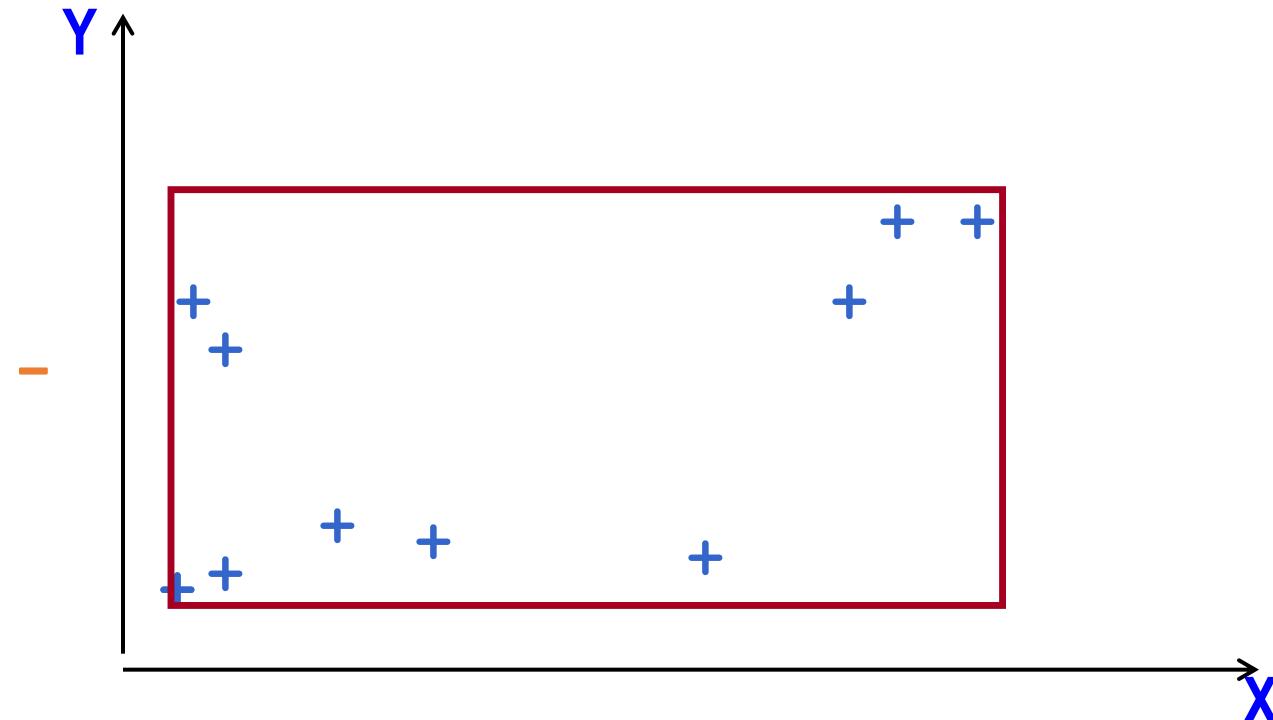


Will we be able to learn the target rectangle?

Can we come close?

Learning Rectangles

Assume the target concept is an axis parallel rectangle

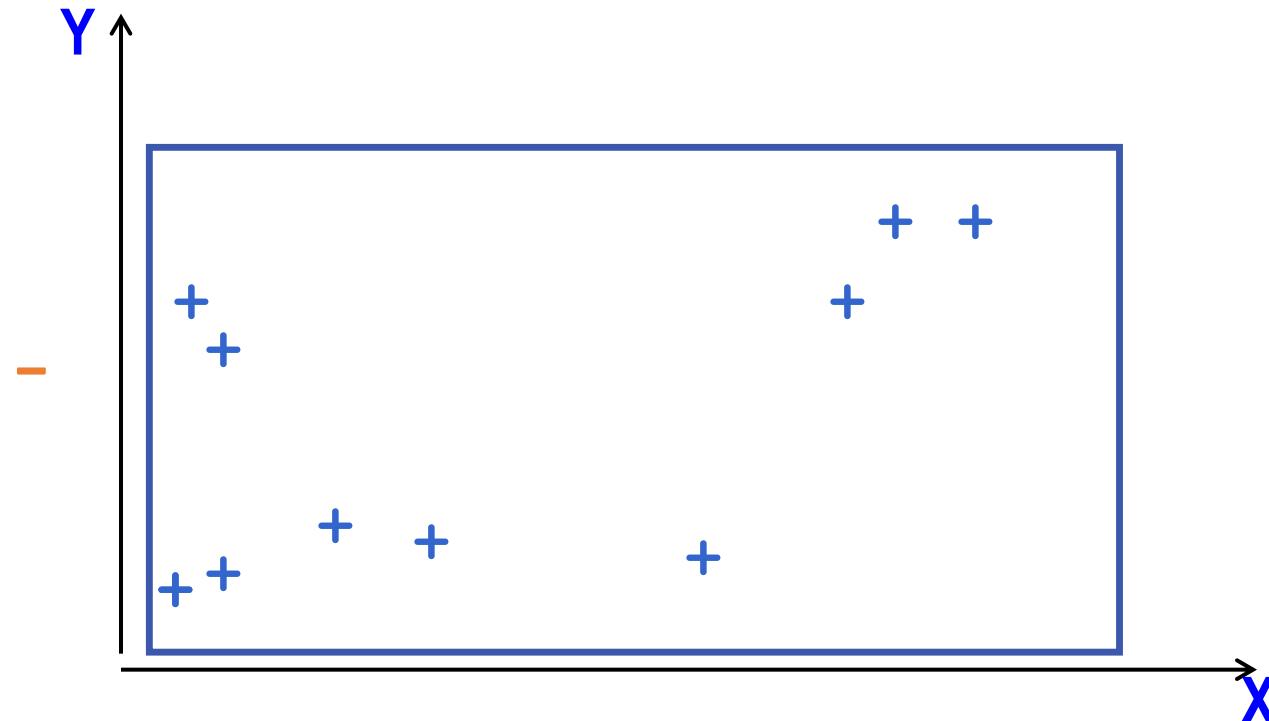


Key observation: Despite there are infinite # hypothesis
The blue & red rectangles have the same predictions

Can we come close?

Learning Rectangles

Assume the target concept is an axis parallel rectangle

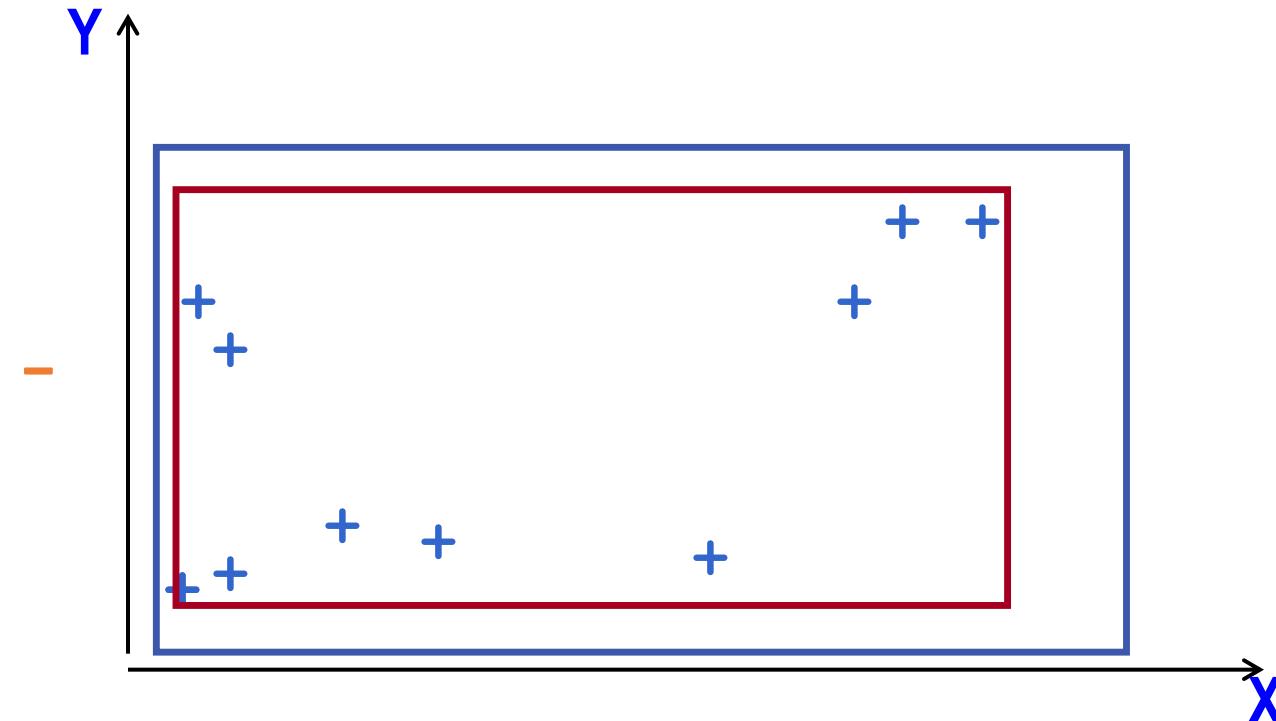


Key observation: Despite there are infinite # hypothesis
The blue & red rectangles have the same predictions

Can we come close?

Learning Rectangles

Assume the target concept is an axis parallel rectangle



Key observation: Despite there are infinite # hypothesis
The blue & red rectangles have the same predictions

Can we come close?

Let's think about expressivity of functions



Suppose we have two points.

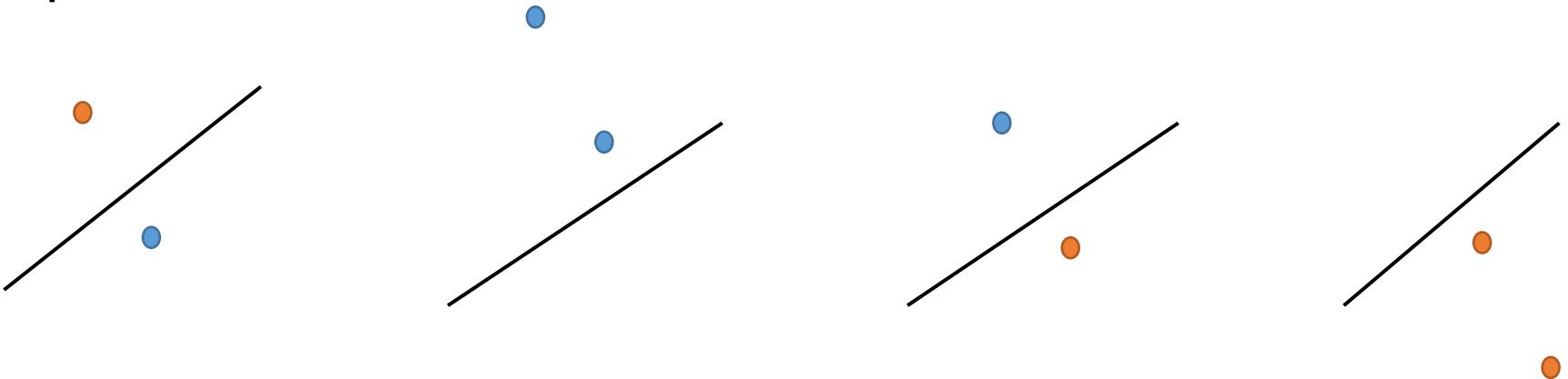
Can linear classifiers correctly classify any labeling of these points?

Linear functions are expressive enough to *shatter* 2 points

Shattering

Definition: A set S of examples is shattered by a set of functions H if **for every** partition of the examples in S into positive and negative examples **there is** a function in H that gives exactly these labels to the examples

Intuition: A rich set of functions shatters large sets of points



VC dimension of Half spaces (Not in Exam)

- ❖ In general, the VC dimension of an n -dimensional linear function is $n+1$

PAC Bound using VC-Dimension (Not in Exam)

- ❖ Give δ and m

$$err_D(h) \leq err_S(h) + \sqrt{\frac{VC(H) \left(\ln \frac{2m}{VC(H)} + 1 \right) + \ln \frac{4}{\delta}}{m}}$$

Computational Learning Theory (Not in Exam)

- ❖ The Theory of Generalization
 - ❖ Using training instance to rule out incorrect hypotheses
- ❖ Probably Approximately Correct (PAC) learning
 - ❖ How many examples you need to see to obtain a learned function with error $\leq \epsilon$
- ❖ Shattering and the VC dimension

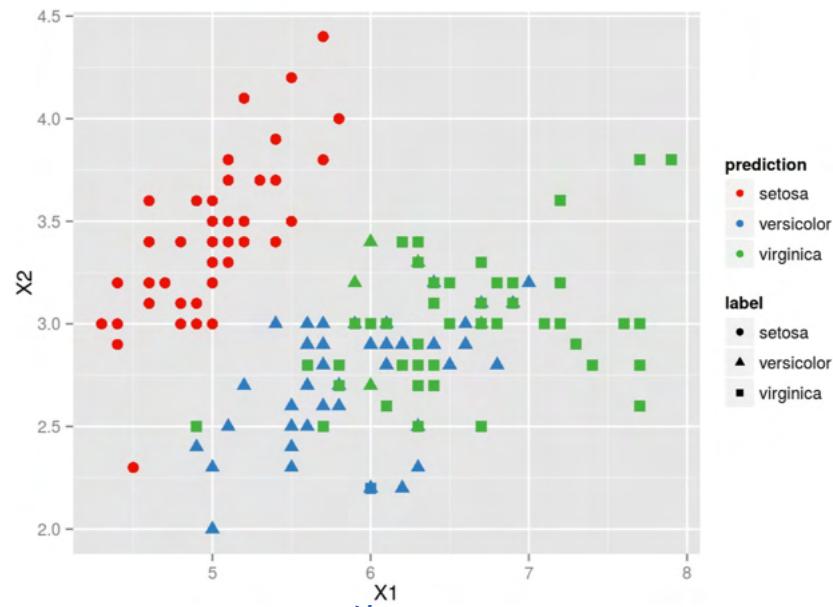
Multi-Class Classification

This Lecture

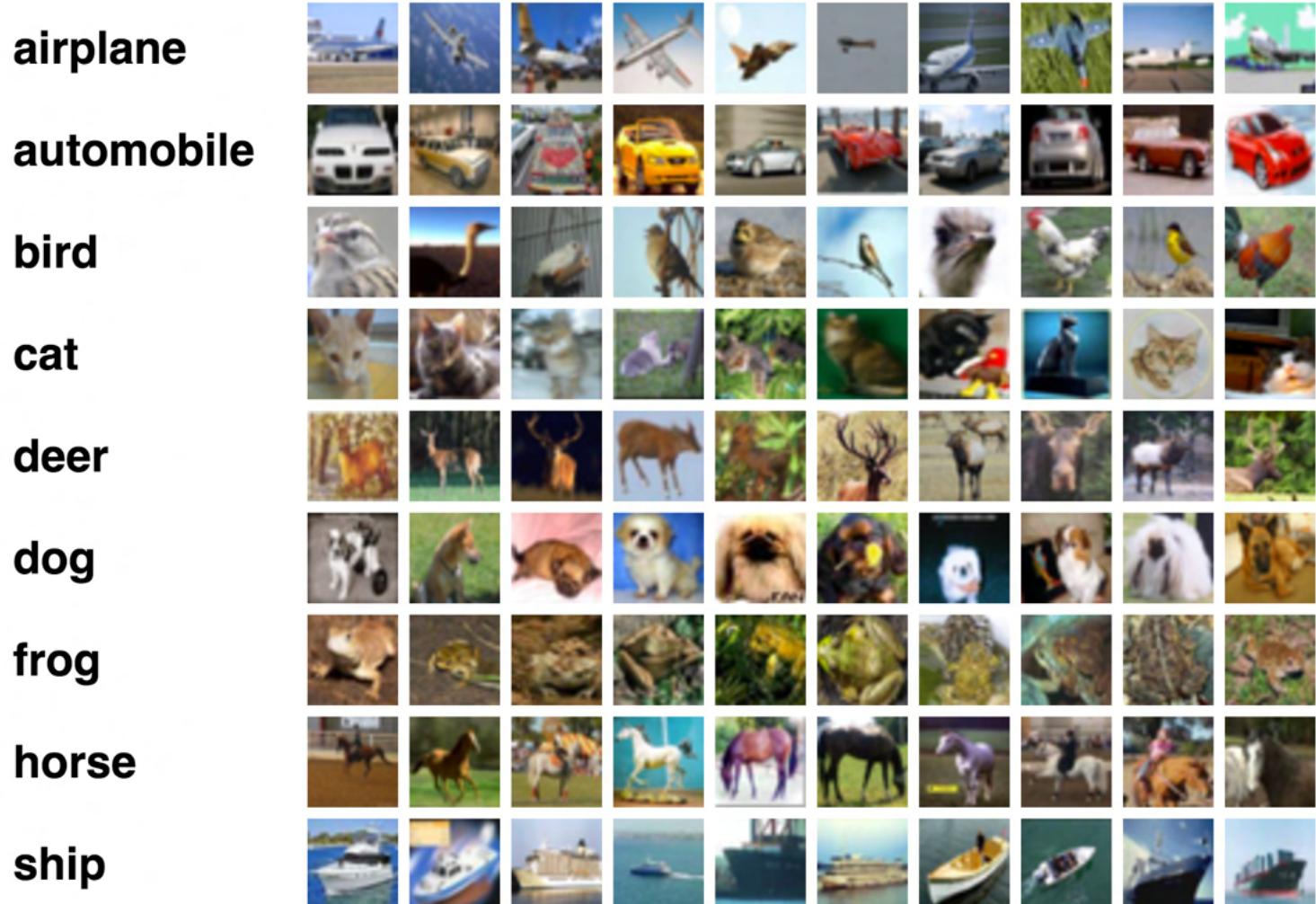
- ❖ Multiclass classification overview
- ❖ Reducing multiclass to binary
 - ❖ One-against-all & One-vs-one

What is multiclass

- ❖ Output $\in \{1, 2, 3, \dots, K\}$
 - ❖ In some cases, output space can be very large (i.e., K is very large)
- ❖ Each input belongs to exactly one class (c.f. in multilabel, input belongs to many classes)



Example applications



Two key ideas to solve multiclass

- ❖ Reducing multiclass to binary
 - ❖ Decompose the multiclass prediction into multiple binary decisions
 - ❖ Make final decision based on multiple binary classifiers
- ❖ Training a single classifier
 - ❖ Minimize the empirical risk
 - ❖ Consider all classes simultaneously

This Lecture

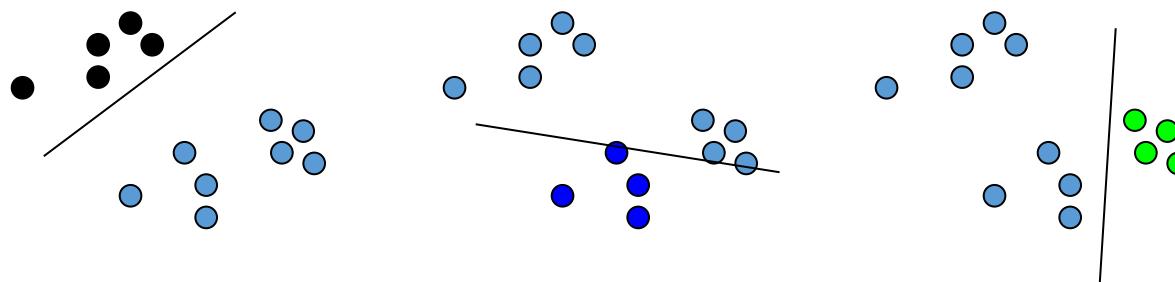
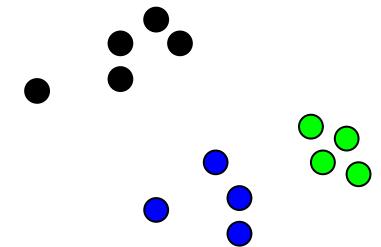
- ❖ Multiclass classification overview
- ❖ Reducing multiclass to binary
 - ❖ One-against-all & One-vs-one

One against all strategy



One against All learning

- ❖ Multiclass classifier
 - ❖ Function $f : \mathbb{R}^n \rightarrow \{1, 2, 3, \dots, k\}$
- ❖ Decompose into binary problems

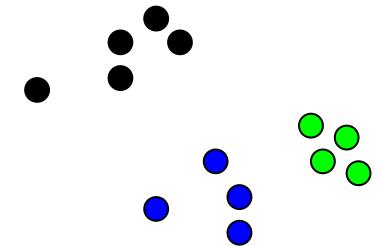


One-again-All learning algorithm

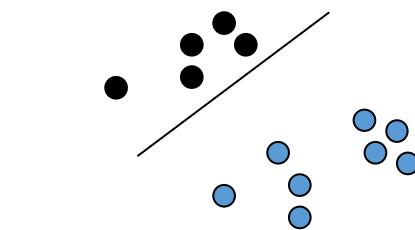
- ❖ Learning: Given a dataset $D = \{(x_i, y_i)\}$
 $x_i \in R^n, y_i \in \{1, 2, 3, \dots, K\}$
- ❖ Decompose into K binary classification tasks
 - ❖ Learn K models: $w_1, w_2, w_3, \dots, w_K$
 - ❖ For class k , construct a binary classification task as:
 - ❖ Positive examples: Elements of D with label k
 - ❖ Negative examples: All other elements of D
 - ❖ The binary classification can be solved by any algorithm we have seen

One against All learning

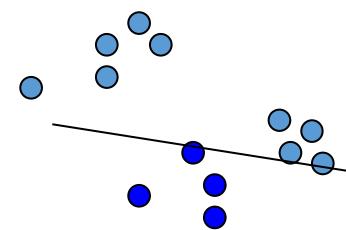
- ❖ Multiclass classifier
 - ❖ Function $f : \mathbb{R}^n \rightarrow \{1, 2, 3, \dots, k\}$
- ❖ Decompose into binary problems



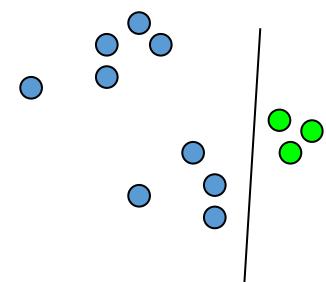
Ideal case: only the correct label will have a positive score



$$w_{black}^T x > 0$$



$$w_{blue}^T x > 0$$



$$w_{green}^T x > 0$$

One-again-All Inference

- ❖ Learning: Given a dataset $D = \{(x_i, y_i)\}$
 $x_i \in R^n, y_i \in \{1, 2, 3, \dots, K\}$
- ❖ Decompose into K binary classification tasks
 - ❖ Learn K models: $w_1, w_2, w_3, \dots, w_K$
- ❖ Inference: “Winner takes all”
 - ❖ $\hat{y} = \operatorname{argmax}_{y \in \{1, 2, \dots, K\}} w_y^T x$

For example: $y = \operatorname{argmax}(w_{black}^T x, w_{blue}^T x, w_{green}^T x)$

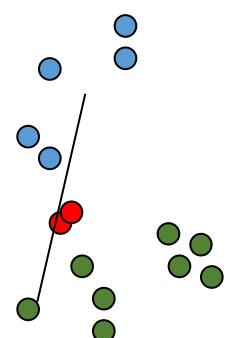
- ❖ An instance of the general form

$$\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} f(y; \mathbf{w}, \mathbf{x})$$

$$\mathbf{w} = \{w_1, w_2, \dots, w_K\}, f(\mathbf{y}; \mathbf{w}, \mathbf{x}) = w_y^T x$$

One-again-All analysis

- ❖ Not always possible to learn
 - ❖ Assumption: each class individually separable from all the others
- ❖ No theoretical justification
 - ❖ Need to make sure the range of all classifiers is the same – we are comparing scores produced by K classifiers trained independently.
- ❖ Easy to implement; work well in practice

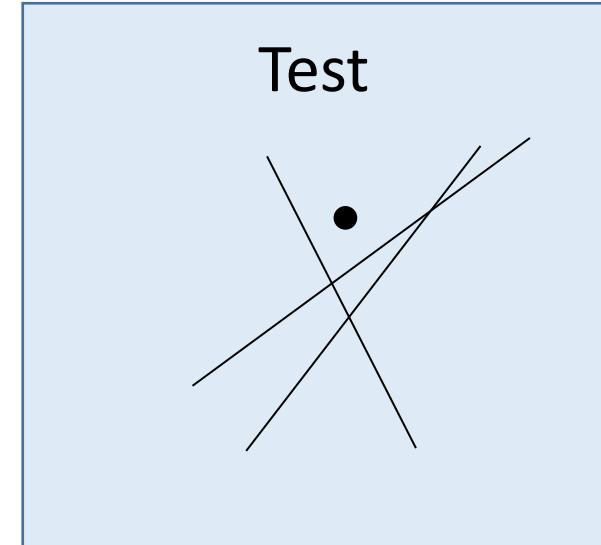
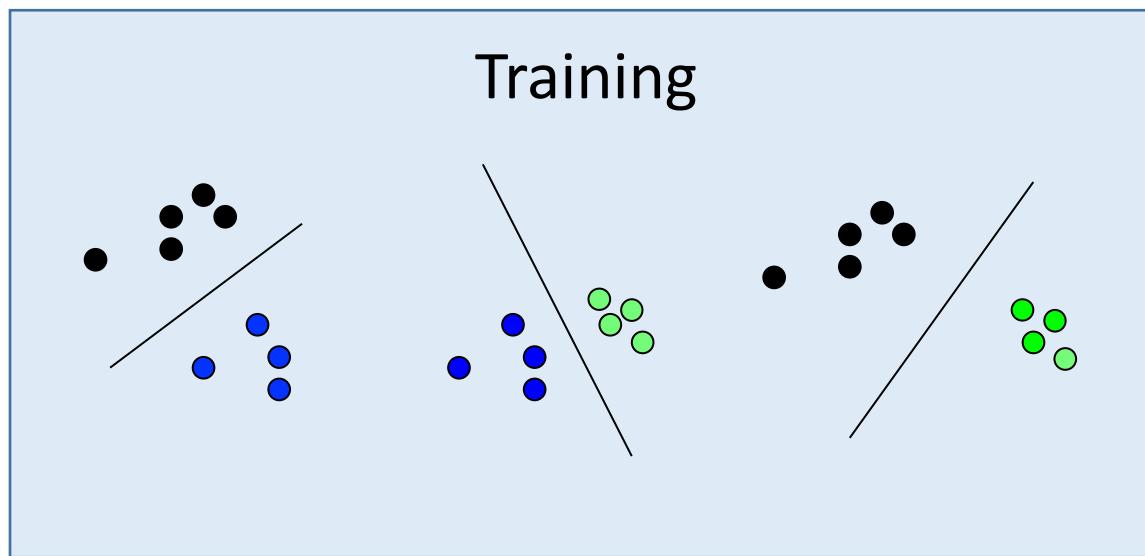
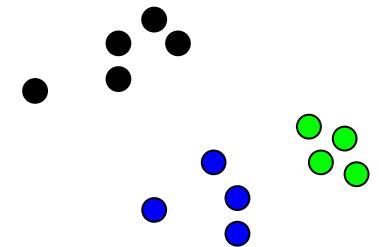


One v.s. One (All against All) strategy



One v.s. One learning

- ❖ Multiclass classifier
 - ❖ Function $f : \mathbb{R}^n \rightarrow \{1, 2, 3, \dots, k\}$
- ❖ Decompose into binary problems



One-v.s-One learning algorithm

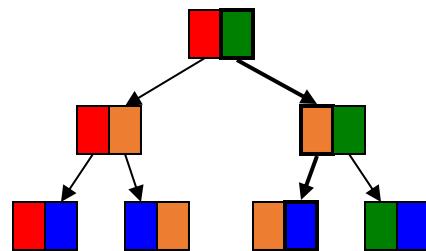
- ❖ Learning: Given a dataset $D = \{(x_i, y_i)\}$
 $x_i \in R^n, y_i \in \{1, 2, 3, \dots, K\}$
- ❖ Decompose into $C(K, 2)$ binary classification tasks
 - ❖ Learn $C(K, 2)$ models: $w_1, w_2, w_3, \dots, w_{K*(K-1)/2}$
 - ❖ For each class pair (i, j) , construct a binary classification task as:
 - ❖ Positive examples: Elements of D with label i
 - ❖ Negative examples Elements of D with label j
 - ❖ The binary classification can be solved by any algorithm we have seen

One-v.s-One Inference algorithm

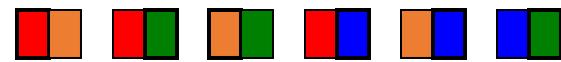
- ❖ Decision Options:
 - ❖ More complex; each label gets $k-1$ votes
 - ❖ Output of binary classifier may not cohere.
 - ❖ Majority: classify example x to take label i if i wins on x more often than j ($j=1,\dots,k$)
 - ❖ A tournament: start with $n/2$ pairs; continue with winners

Classifying with One-vs-one

Tournament



Majority Vote



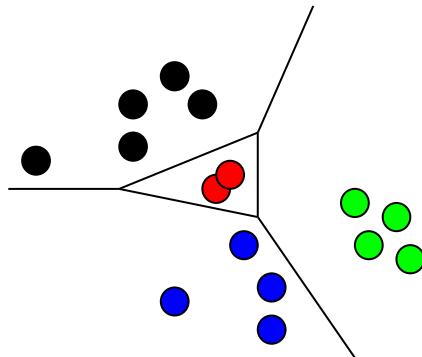
1 red, 2 yellow, 2 green

→ ?

All are post-learning and *might* cause weird stuff

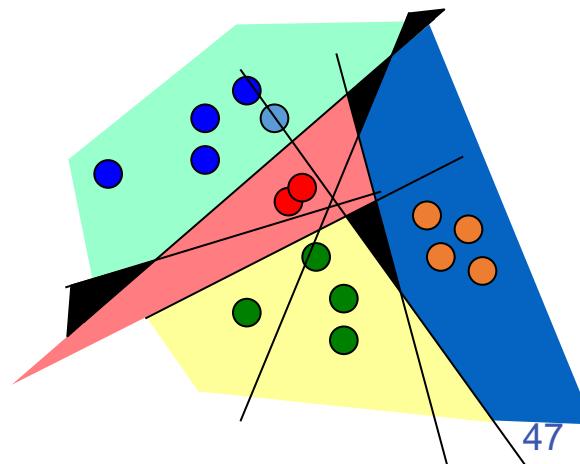
One-v.s.-one Assumption

- ❖ Every pair of classes is separable



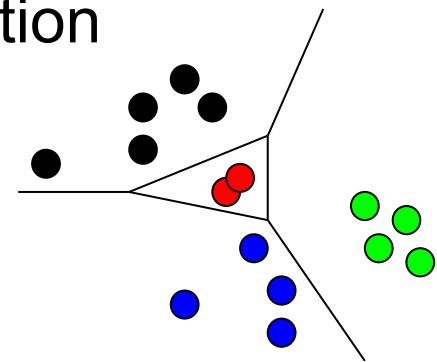
It is possible to separate all k classes with the $O(k^2)$ classifiers

Decision
Regions



Comparisons

- ❖ One against all
 - ❖ $O(K)$ weight vectors to train and store
 - ❖ Training set of the binary classifiers may unbalanced
 - ❖ Less expressive; make a strong assumption
- ❖ One v.s. One (All v.s. All)
 - ❖ $O(K^2)$ weight vectors to train and store
 - ❖ Size of training set for a pair of labels could be small
→ **overfitting** of the binary classifiers
 - ❖ Need large space to store model



Problems with Decompositions

- ❖ Learning optimizes over *local* metrics
 - ❖ Does not guarantee good *global* performance
 - ❖ We don't care about the performance of the *local* classifiers
- ❖ Poor decomposition \Rightarrow poor performance
 - ❖ Difficult local problems
 - ❖ Irrelevant local problems
- ❖ Efficiency: e.g., All vs. All vs. One vs. All
- ❖ Not clear how to generalize multi-class to problems with a very large # of output

Decomposition methods: Summary

- ❖ General Ideas:
 - ❖ Decompose the multiclass problem into many binary problems
 - ❖ Prediction depends on the decomposition
 - ❖ Constructs the multiclass label from the output of the binary classifiers
- ❖ Learning optimizes **local correctness**
 - ❖ Each binary classifier don't need to be globally correct and isn't aware of the prediction procedure

Multi-class Logistic Regression

Recall: (binary) logistic regression

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \log(1 + e^{-y_i(\mathbf{w}^T \mathbf{x}_i)})$$

Assume labels are generated using the following probability distribution:

$$P(y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$
$$P(y = -1 | \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

(multi-class) log-linear model

- ❖ Assumption:

$$P(y|x, w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)}$$

Partition function

- ❖ This is a valid probability assumption. Why?

soft-max function

Softmax

- ❖ Softmax: let $s(y)$ be the score for output y
here $s(y) = w^T \phi(x, y)$ (or $w_y^T x$) but it can be computed by other function.

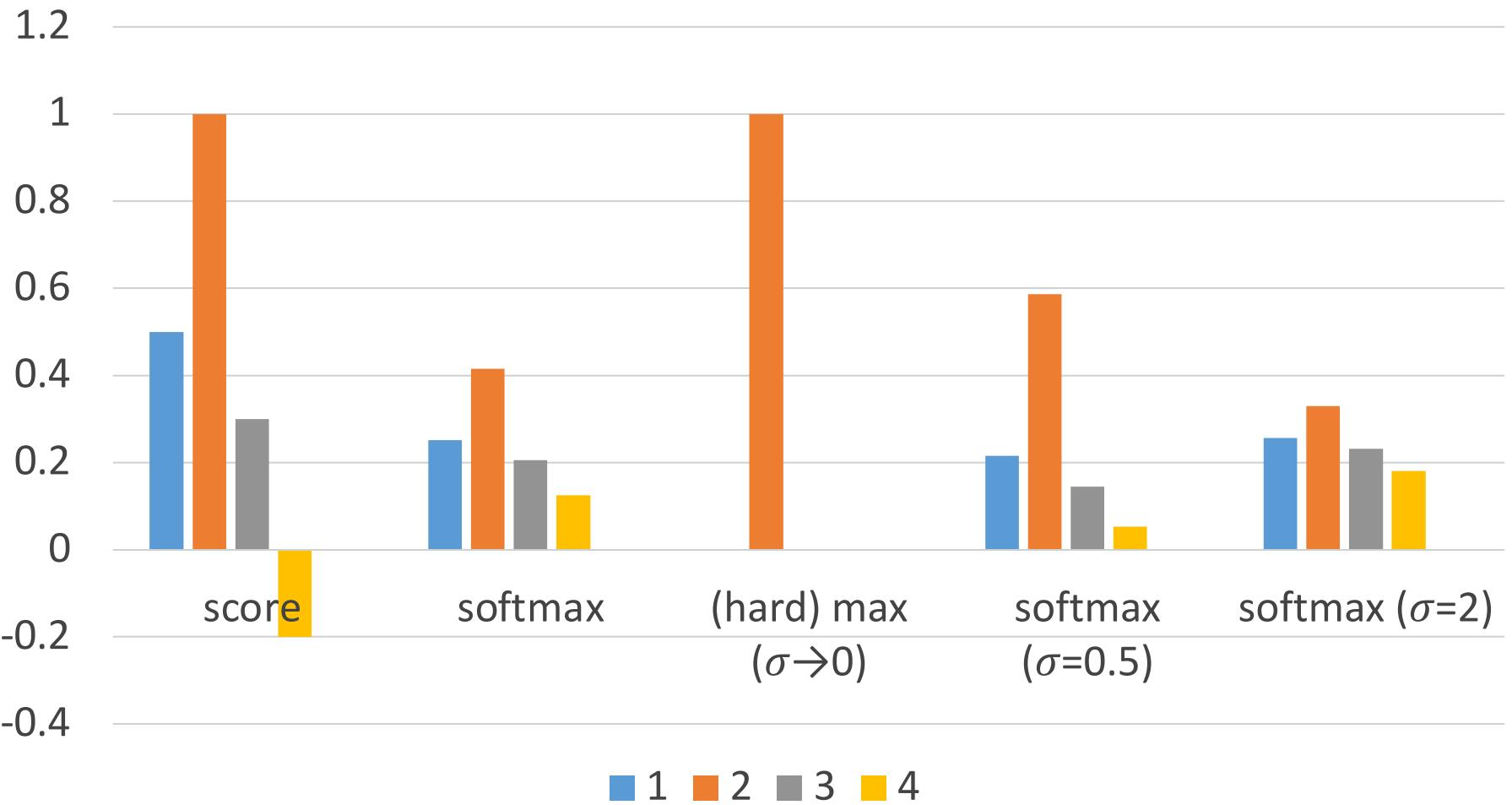
$$P(y) = \frac{\exp(s(y))}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(s(y'))}$$

- ❖ We can control the peakedness of the distribution

$$P(y|\sigma) = \frac{\exp(s(y)/\sigma)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(s(y'/\sigma))}$$

Example

$$S(1) = .5; \quad s(2)=1; \quad s(3)=0.3; \quad s(4)=-0.2$$



Log linear model

$$P(y|x, w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)}$$

$$\begin{aligned} \log P(y|x, w) &= \log(\exp(w_y^T x)) - \log(\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)) \\ &= w_y^T x - \log(\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)) \end{aligned}$$

Maximum log-likelihood estimation

- ❖ Training can be done by maximum log-likelihood estimation i.e. $\max_w \log P(D|w)$

$$D = \{(x_i, y_i)\}$$

$$P(D|w) = \prod_i \frac{\exp(w_{y_i}^T x_i)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x_i)}$$

$$\log P(D|w) = \sum_i [w_{y_i}^T x_i - \log \sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x_i)]$$

Comparisons

- ❖ Log-linear model (multi-class)

$$\min_w \sum_i [\log \sum_{k \in \{1, 2, \dots, K\}} \exp(w_k^T x_i) - w_{y_i}^T x_i]$$

- ❖ Log-linear mode (logistic regression)

$$\min_w \sum_i \log(1 + e^{-y_i(w^T x_i)})$$

Reduction v.s. single classifier

- ❖ Reduction
 - ❖ Future-proof: binary classification improved so does muti-class
 - ❖ Easy to implement
- ❖ Single classifier
 - ❖ Global optimization: directly minimize the empirical loss; easier for joint prediction

Lecture 12: Kernel Method

Fall 2021

Kai-Wei Chang

CS @ UCLA

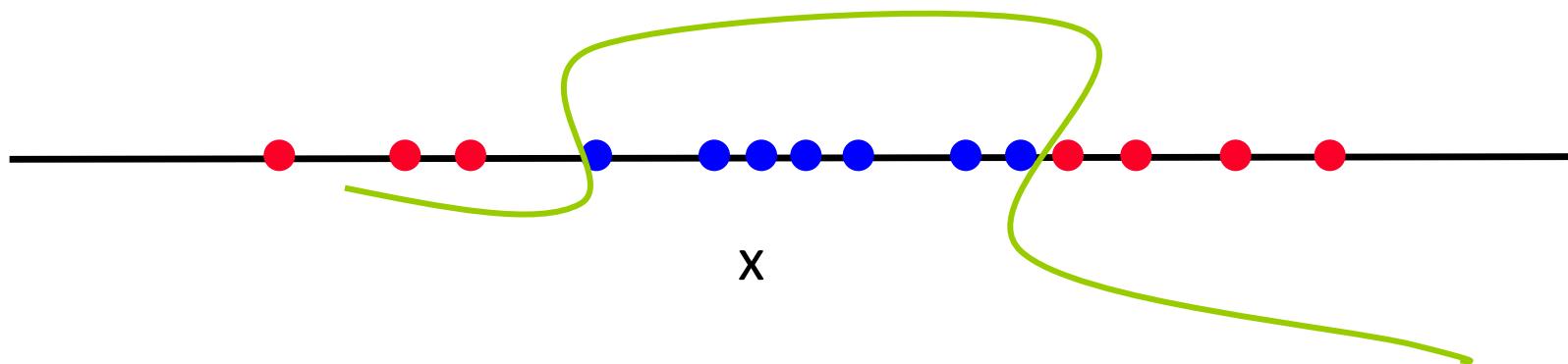
kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Kernel and Kernel methods

Functions Can be Made Linear

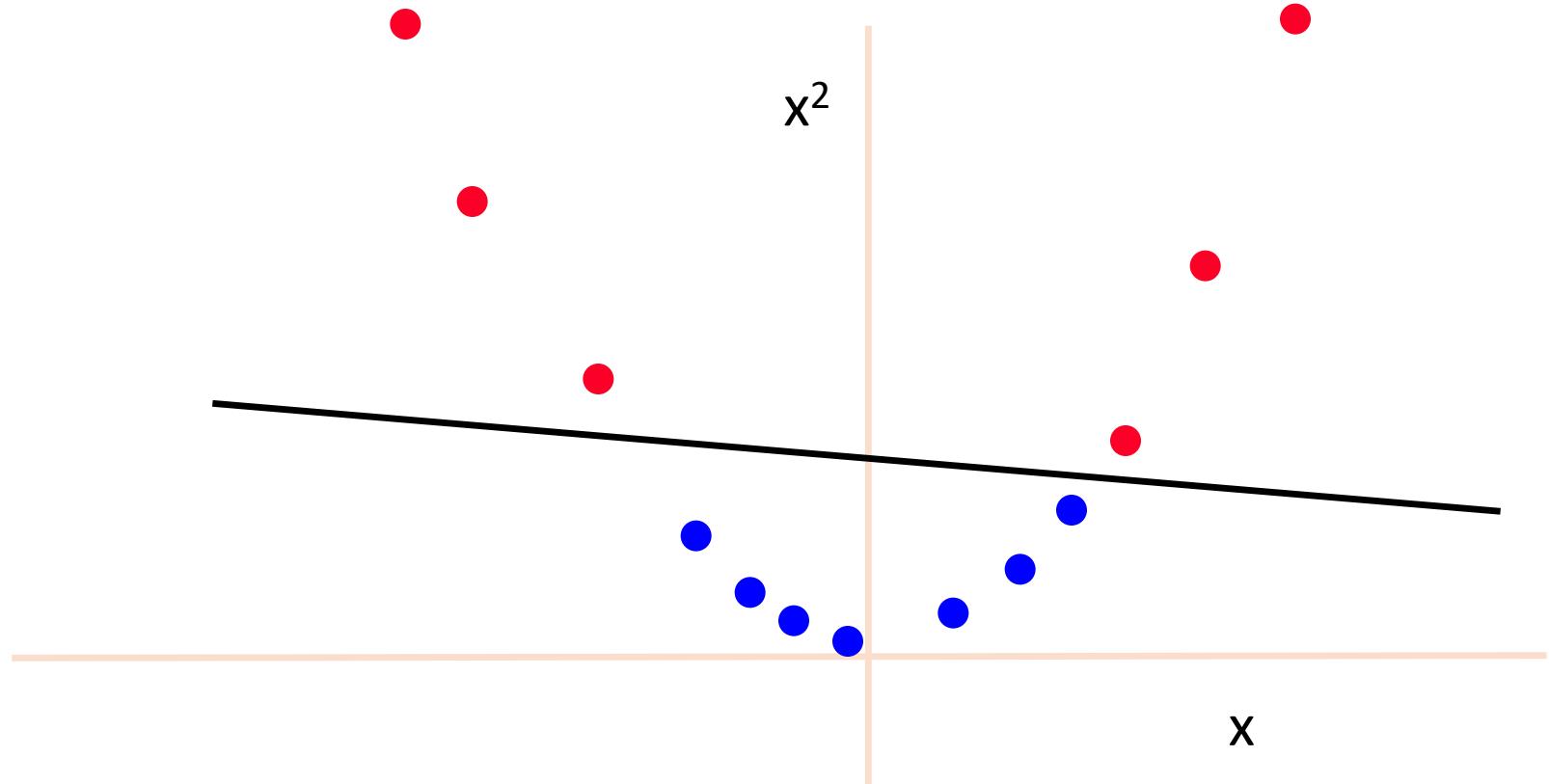
- ❖ Data are not linearly separable in one dimension
- ❖ Not separable if you insist on using a specific class of functions



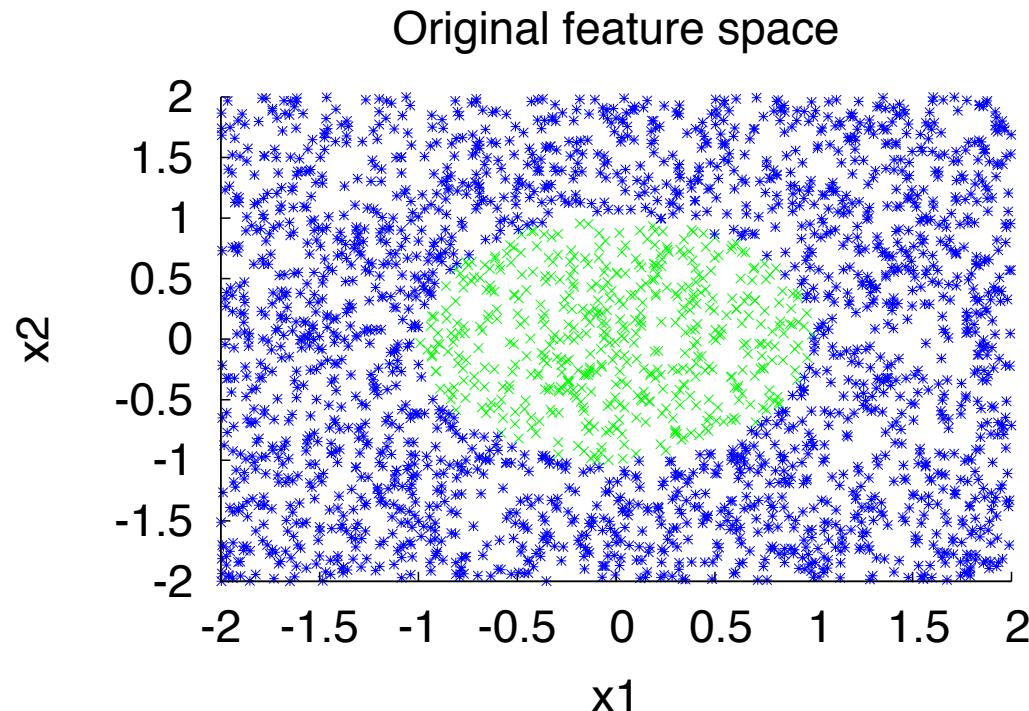
Can we do some mapping to make it linear spreadable?

Blown Up Feature Space

- ❖ Data are separable in $\langle x, x^2 \rangle$ space

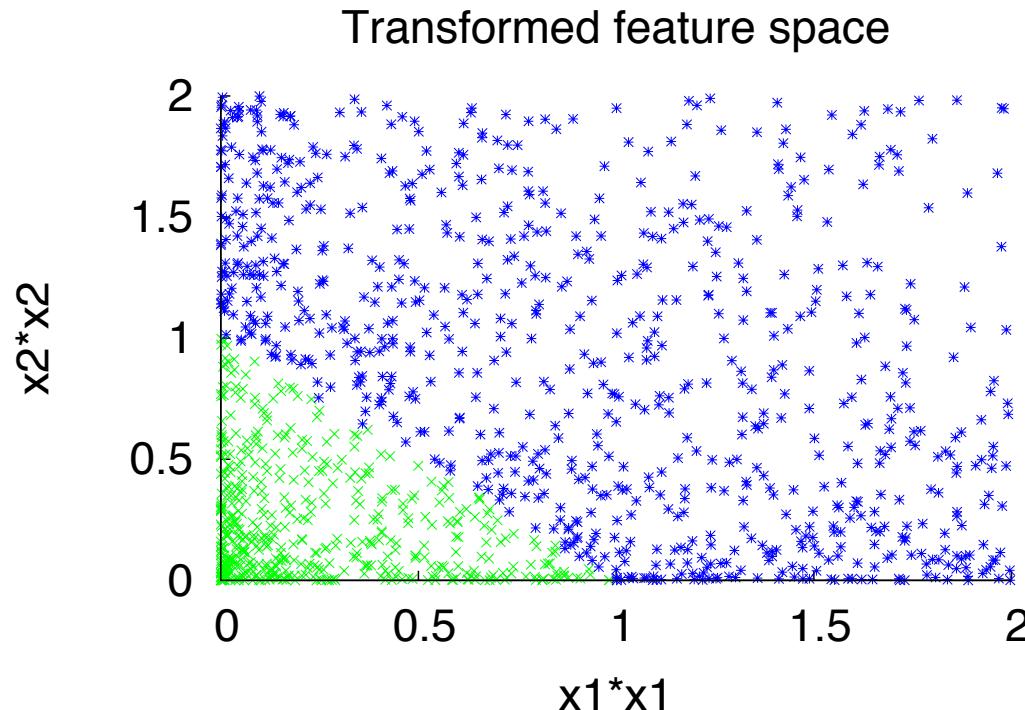


2D example



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Making data linearly separable



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Transform data: $\mathbf{x} = (x_1, x_2) \Rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2)$
 $f(\phi(\mathbf{x})) = 1 \text{ iff } \phi(\mathbf{x})_1 + \phi(\mathbf{x})_2 \leq 1$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^{2n}$

2. For (x, y) in \mathcal{D} :

3. if $y w^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq 0$

Assume $y \in \{1, -1\}$

4.

$w \leftarrow w + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$

5.

What if our mapping function is more complex?

E.g., mapping data to infinite # dimensions

Ans: it's okay if we can compute $\phi(x_i)^T \phi(x_j)$

Dual Representation

if $y(\mathbf{w}^\top \mathbf{x}) \leq 0$
 $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$

- ❖ Let \mathbf{w} be an initial weight vector for perceptron. Let $(x_1, +), (x_2, +), (x_3, -), (x_4, -)$ be examples and assume mistakes are made on x_1, x_2 and x_4 .
- ❖ What is the resulting weight vector?

$$\mathbf{w} = \mathbf{w} + x_1 + x_2 - x_4$$

- ❖ In general, the weight vector w can be written as a linear combination of examples:

$$w = \sum_{1..m} \alpha_i y_i x_i$$

- ❖ Where α_i is the **number of mistakes** made on x_i .
- ❖ What will be the prediction on x

Predicting with linear classifiers

- ❖ Prediction = $\text{sgn}(\mathbf{w}^T \mathbf{x})$ and $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$
- ❖ That is, we just showed that

$$\mathbf{w}^T \mathbf{x} = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}$$

- ❖ We only need to compute dot products between training examples and the new example \mathbf{x}
- ❖ This is true even if we map examples to a high dimensional space

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

Many learning algorithm require to compute inner products

- ❖ Perceptron:

$$y(\mathbf{w}^\top \mathbf{x}) \leq 0$$

- ❖ K-NN:

$$\text{similarity}(\mathbf{x}, \mathbf{x}^{\text{neighbor}}) = \mathbf{x}^T \mathbf{x}^{\text{neighbor}}$$

Is there a smarter way to compute the inner product?

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction with this *high dimensional lifting map* is

$$\operatorname{sgn}(\mathbf{w}^T \phi(\mathbf{x})) = \operatorname{sgn} \left(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction with this *high dimensional lifting map* is

$$\hat{y} = \mathbf{w}^T \phi(\mathbf{x}) = \left(\sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \right) \phi(\mathbf{x})$$

If we can compute the value of K without explicitly writing the blown up representation, then we will have a computational advantage.

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Example

❖ Assume the mapping:

$$\phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_D \\ x_1^2 \\ x_1x_2 \\ \vdots \\ x_1x_D \\ x_2x_1 \\ x_2^2 \\ \vdots \\ x_2x_D \\ \vdots \\ x_Dx_1 \\ \vdots \\ x_D^2 \end{pmatrix}$$

Computing $\mathbf{w}^T \phi(\mathbf{x})$ is $O(D^2)$.

Inner product can be computed efficiently

- ❖ However,

$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$$

- ❖ In fact, it can be computed in

$$O(D)$$

$$\phi(\mathbf{x}) =$$

$$\begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_D \\ x_1^2 \\ x_1x_2 \\ \vdots \\ x_1x_D \\ x_2x_1 \\ x_2^2 \\ \vdots \\ x_2x_D \\ \vdots \\ x_Dx_1 \\ \vdots \\ x_D^2 \end{pmatrix}$$

Example: polynomial kernel

Let us examine more closely the inner products $\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$ for a pair of data points \mathbf{x}_m and \mathbf{x}_n .

Polynomial-based nonlinear basis functions consider the following $\phi(\mathbf{x})$:

$$\phi : \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

This gives rise to an inner product in a special form,

$$\begin{aligned}\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n) &= x_{m1}^2 x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2 x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\mathbf{x}_m^T \mathbf{x}_n)^2\end{aligned}$$

Namely, the inner product can be computed by a function $(\mathbf{x}_m^T \mathbf{x}_n)^2$ defined in terms of the original features, *without computing $\phi(\cdot)$* .

The Kernel Trick

Suppose we wish to compute

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$

Here ϕ maps \mathbf{x} and \mathbf{z} to a high dimensional space

The Kernel Trick: Save time/space by computing the value of $K(\mathbf{x}, \mathbf{z})$ by performing operations in the original space (without a feature transformation!)

Kernel functions

- ❖ A kernel function $k(\cdot, \cdot)$ satisfies the following properties
- ❖ For any x_m, x_n

$$k(x_m, x_n) = k(x_n, x_m) \text{ and } k(x_m, x_n) = \phi(x_m)^T \phi(x_n)$$

for some function $\phi(\cdot)$

Example: $k(x_m, x_n) = (x_m^T x_n)^2$ is a kernel function:

$$\begin{aligned}\phi(x_m)^T \phi(x_n) &= x_{m1}^2 x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2 x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\mathbf{x}_m^T \mathbf{x}_n)^2\end{aligned}$$

Exercise

- ❖ Let $x \in R^2$, show $(4 + 9x_i^T x_j)^2$ is a valid kernel.

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}_{i=1}^m$

1. Initialize $w \leftarrow 0 \in R^{2n}$

2. For (x, y) in \mathcal{D} :

3. if $y w^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq 0$

Assume $y \in \{1, -1\}$

4. $w \leftarrow w + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$

5.

6. Return w

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T \begin{bmatrix} x \\ x^2 \end{bmatrix})$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}_{i=1}^m$

1. Initialize $w \leftarrow \mathbf{0}$
2. For (x, y) in \mathcal{D} :
3. if $y w^T \phi(x) \leq 0$
4. $w \leftarrow w + y \phi(x)$
- 5.
6. Return w

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T \phi(x))$

$$w^T \phi(x) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)$$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}_{i=1}^m$

1. Initialize $\alpha \leftarrow \mathbf{0} \in R^m$ w $\leftarrow \mathbf{0}$
2. For (x_i, y_i) in \mathcal{D} :
 if $y_i \sum_j \alpha_j y_j K(x_i, x_j) \leq 0$ y w^T φ(x) ≤ 0
3. $\alpha_i \leftarrow \alpha_i + 1$ w $\leftarrow w + y\phi(x)$
- 4.
5. Return w
- 6.

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T \phi(x)) = \sum_i \alpha_i y_i K(x_i, x)$

$$w^T \phi(x) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)$$

Lecture 13: Kernel Methods

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Multi-class Logistic Regression

One-against-All learning algorithm

- ❖ Learning: Given a dataset $D = \{(x_i, y_i)\}$
 $x_i \in R^n, y_i \in \{1, 2, 3, \dots, K\}$
- ❖ Decompose into K binary classification tasks
 - ❖ Learn K models: $w_1, w_2, w_3, \dots, w_K$
 - ❖ For class k, construct a binary classification task as:
 - ❖ Positive examples: Elements of D with label k
 - ❖ Negative examples: All other elements of D
 - ❖ The binary classification can be solved by any algorithm we have seen

Recall: (binary) logistic regression

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \log(1 + e^{-y_i(\mathbf{w}^T \mathbf{x}_i)})$$

Assume labels are generated using the following probability distribution:

$$P(y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$
$$P(y = -1 | \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

(multi-class) log-linear model

- ❖ Assumption:

$$P(y|x, w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)}$$

Partition function

- ❖ This is a valid probability assumption. Why?

soft-max function

Softmax

- ❖ Softmax: let $s(y)$ be the score for output y
here $s(y) = w^T \phi(x, y)$ (or $w_y^T x$) but it can be computed by other function.

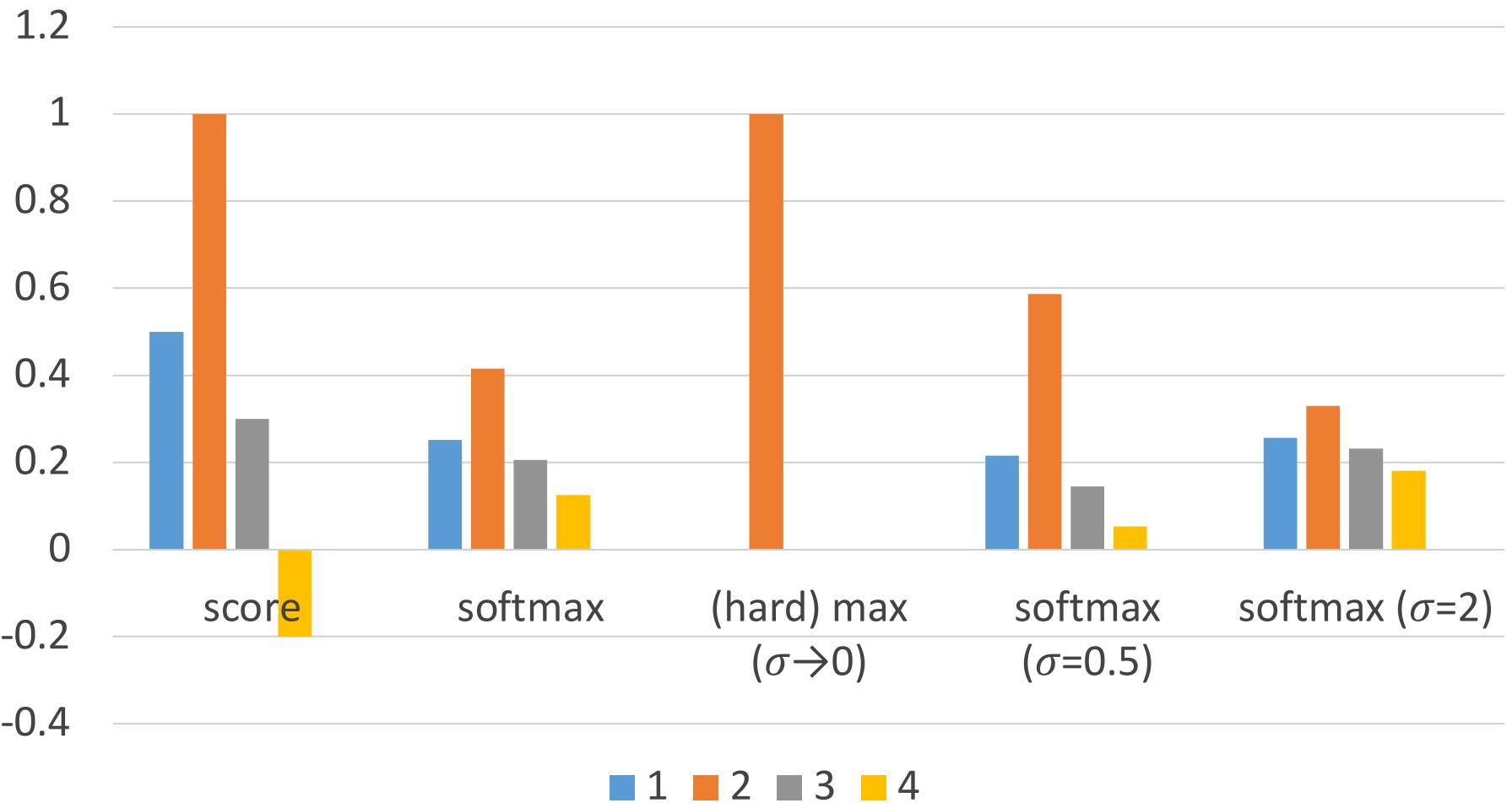
$$P(y) = \frac{\exp(s(y))}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(s(y'))}$$

- ❖ We can control the peakedness of the distribution

$$P(y|\sigma) = \frac{\exp(s(y)/\sigma)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(s(y'/\sigma))}$$

Example

$$S(1) = .5; \quad s(2)=1; \quad s(3)=0.3; \quad s(4)=-0.2$$



Log linear model

$$P(y|x, w) = \frac{\exp(w_y^T x)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)}$$

$$\begin{aligned} \log P(y|x, w) &= \log(\exp(w_y^T x)) - \log(\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)) \\ &= w_y^T x - \log(\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x)) \end{aligned}$$

Maximum log-likelihood estimation

- ❖ Training can be done by maximum log-likelihood estimation i.e. $\max_w \log P(D|w)$

$$D = \{(x_i, y_i)\}$$

$$P(D|w) = \prod_i \frac{\exp(w_{y_i}^T x_i)}{\sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x_i)}$$

$$\log P(D|w) = \sum_i [w_{y_i}^T x_i - \log \sum_{y' \in \{1, 2, \dots, K\}} \exp(w_{y'}^T x_i)]$$

Comparisons

- ❖ Log-linear model (multi-class)

$$\min_w \sum_i [\log \sum_{k \in \{1, 2, \dots, K\}} \exp(w_k^T x_i) - w_{y_i}^T x_i]$$

- ❖ Log-linear mode (logistic regression)

$$\min_w \sum_i \log(1 + e^{-y_i(w^T x_i)})$$

Reduction v.s. single classifier

- ❖ Reduction
 - ❖ Future-proof: binary classification improved so does multi-class
 - ❖ Easy to implement
- ❖ Single classifier
 - ❖ Global optimization: directly minimize the empirical loss; easier for joint prediction

Lecture 13: Kernel Method

Fall 2021

Kai-Wei Chang

CS @ UCLA

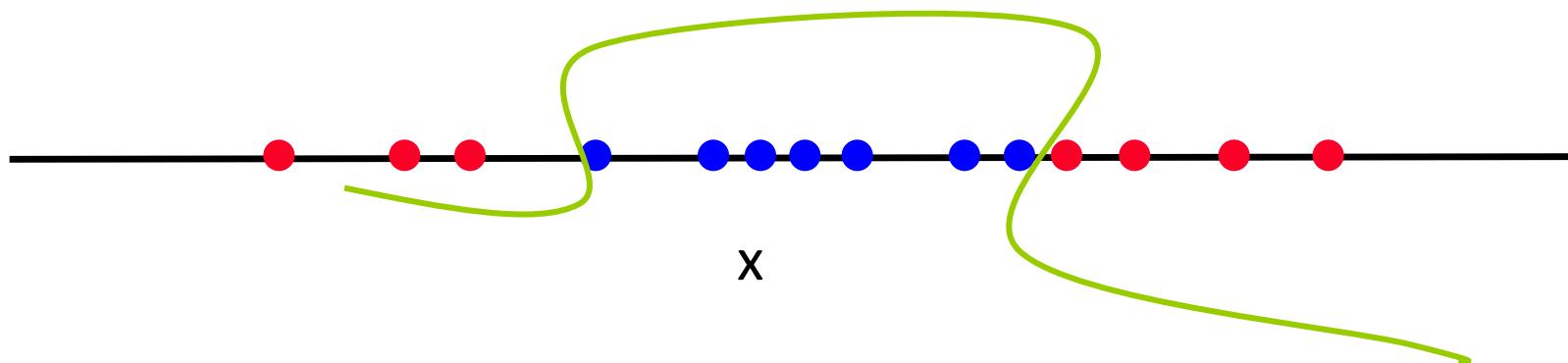
kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Kernel and Kernel methods

Functions Can be Made Linear

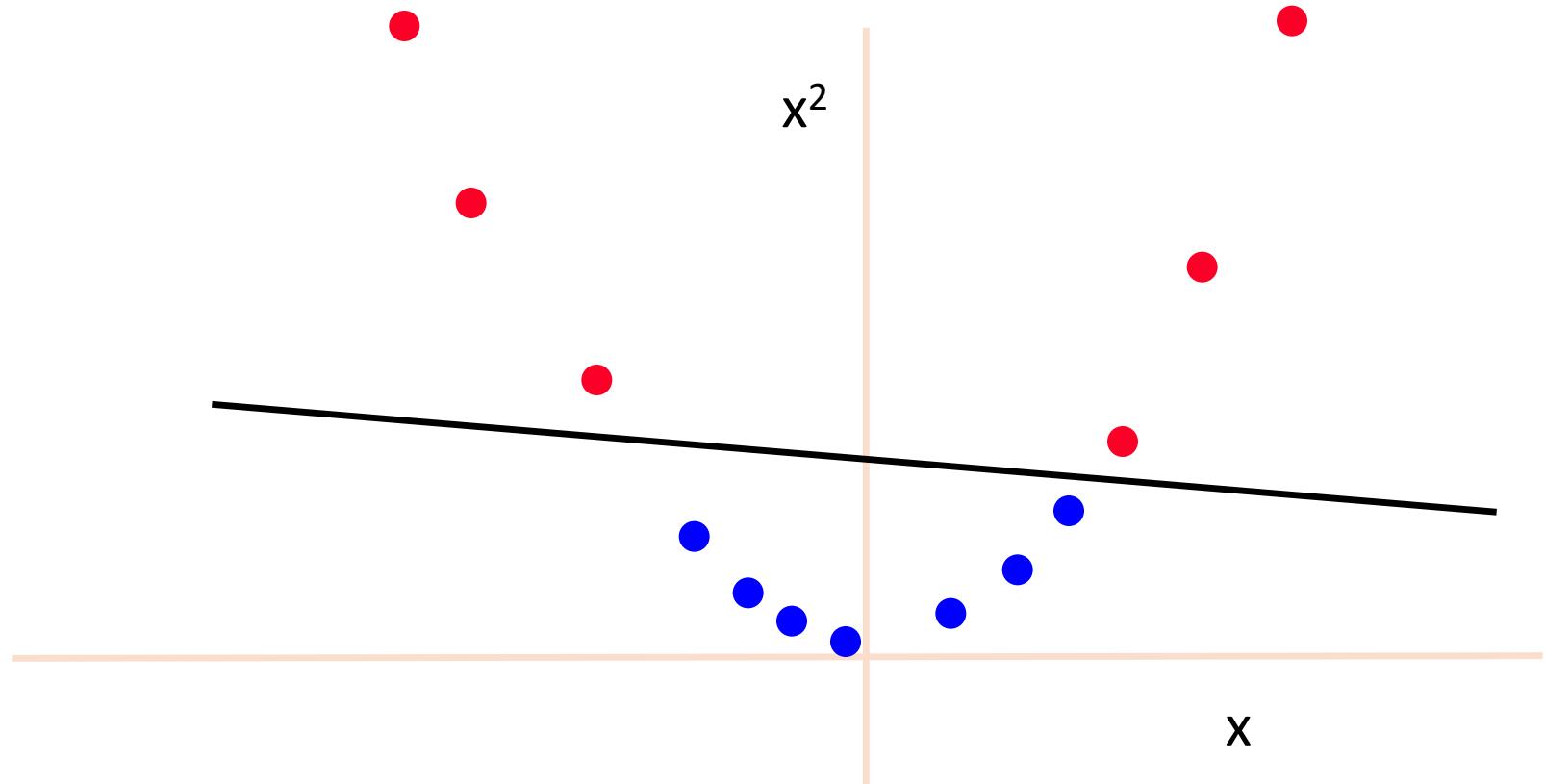
- ❖ Data are not linearly separable in one dimension
- ❖ Not separable if you insist on using a specific class of functions



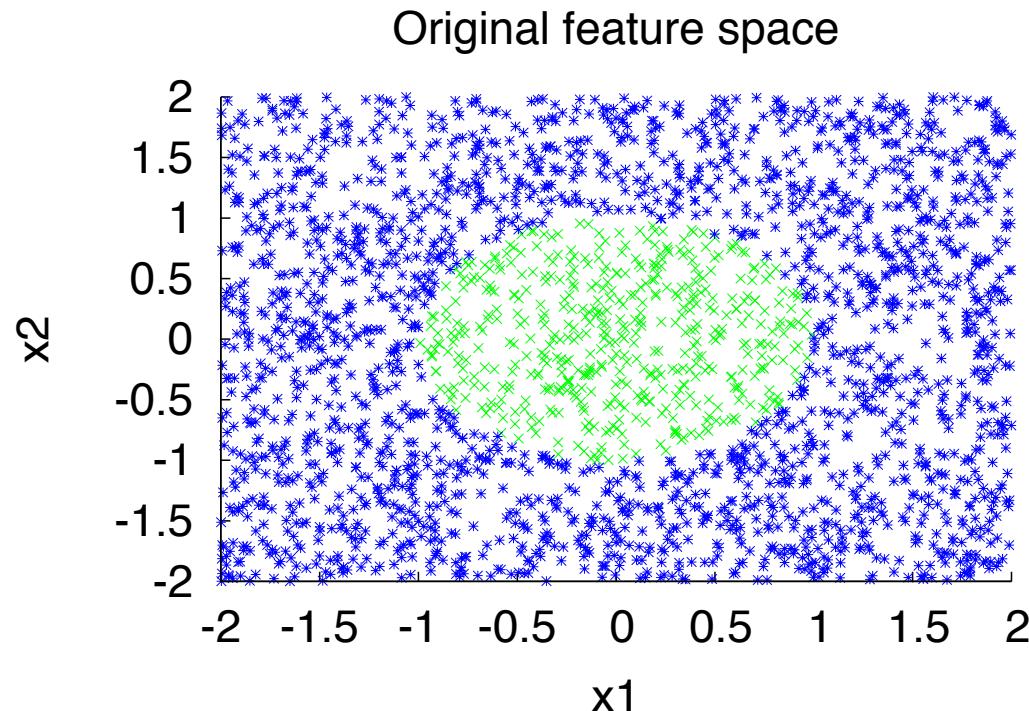
Can we do some mapping to make it linear spreadable?

Blown Up Feature Space

- ❖ Data are separable in $\langle x, x^2 \rangle$ space

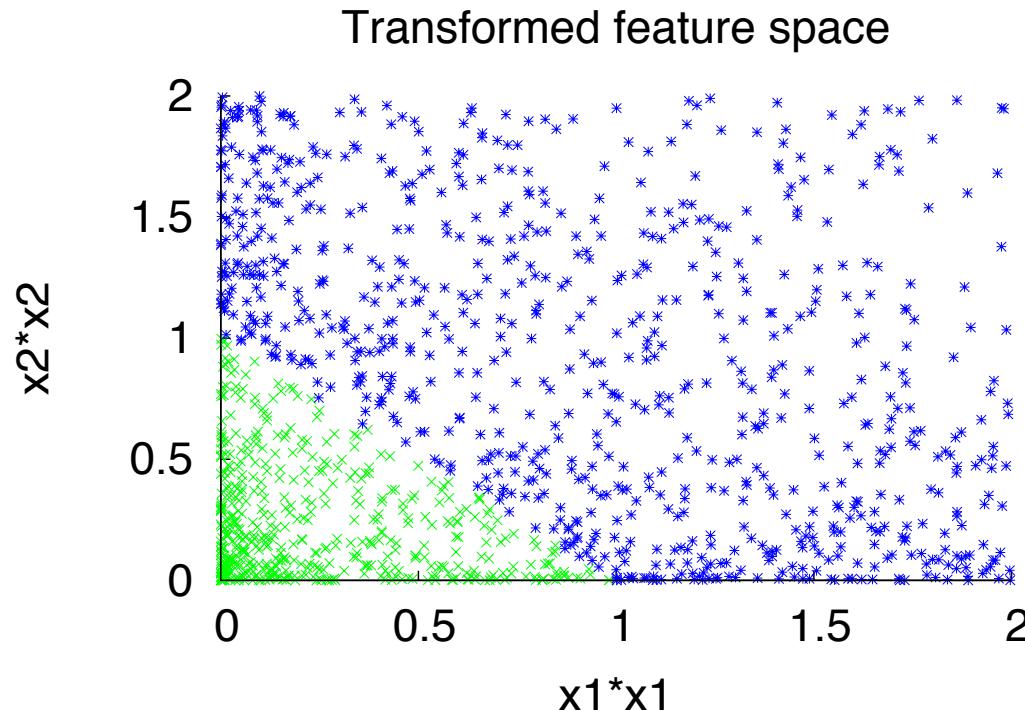


2D example



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Making data linearly separable



$$f(\mathbf{x}) = 1 \text{ iff } x_1^2 + x_2^2 \leq 1$$

Transform data: $\mathbf{x} = (x_1, x_2) \Rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2)$

$$f(\phi(\mathbf{x})) = 1 \text{ iff } \phi(\mathbf{x})_1 + \phi(\mathbf{x})_2 \leq 1$$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow 0 \in \mathbb{R}^{2n}$

2. For (x, y) in \mathcal{D} :

3. if $y w^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq 0$

Assume $y \in \{1, -1\}$

4.

$w \leftarrow w + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$

5.

What if our mapping function is more complex?

E.g., mapping data to infinite # dimensions

Ans: it's okay if we can compute $\phi(x_i)^T \phi(x_j)$

Dual Representation

if $y(\mathbf{w}^\top \mathbf{x}) \leq 0$
 $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$

- ❖ Let \mathbf{w} be an initial weight vector for perceptron. Let $(x_1, +), (x_2, +), (x_3, -), (x_4, -)$ be examples and assume mistakes are made on x_1, x_2 and x_4 .
- ❖ What is the resulting weight vector?

$$\mathbf{w} = \mathbf{w} + x_1 + x_2 - x_4$$

- ❖ In general, the weight vector w can be written as a linear combination of examples:

$$w = \sum_{1..m} \alpha_i y_i x_i$$

- ❖ Where α_i is the **number of mistakes** made on x_i .
- ❖ What will be the prediction on x

Predicting with linear classifiers

- ❖ Prediction = $\text{sgn}(\mathbf{w}^T \mathbf{x})$ and $\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$
- ❖ That is, we just showed that

$$\mathbf{w}^T \mathbf{x} = \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}$$

- ❖ We only need to compute dot products between training examples and the new example \mathbf{x}
- ❖ This is true even if we map examples to a high dimensional space

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

Many learning algorithm require to compute inner products

- ❖ Perceptron:

$$y(\mathbf{w}^\top \mathbf{x}) \leq 0$$

- ❖ K-NN:

$$\text{similarity}(\mathbf{x}, \mathbf{x}^{\text{neighbor}}) = \mathbf{x}^T \mathbf{x}^{\text{neighbor}}$$

Is there a smarter way to compute the inner product?

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction with this *high dimensional lifting map* is

$$\text{sgn}(\mathbf{w}^T \phi(\mathbf{x})) = \text{sgn} \left(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right)$$

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Dot products in high dimensional spaces

Let us define a dot product in the high dimensional space

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \phi(\mathbf{z})$$

So prediction with this *high dimensional lifting map* is

$$\hat{y} = \text{sign}\left(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right)$$

If we can compute the value of K without explicitly writing the blown up representation, then we will have a computational advantage.

because $\mathbf{w}^T \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$

Example

❖ Assume the mapping:

Computing $\mathbf{w}^T \phi(\mathbf{x})$ is $O(D^2)$.

$$\phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_D \\ x_1^2 \\ x_1x_2 \\ \vdots \\ x_1x_D \\ x_2x_1 \\ x_2^2 \\ \vdots \\ x_2x_D \\ \vdots \\ x_Dx_1 \\ \vdots \\ x_D^2 \end{pmatrix}$$

Inner product can be computed efficiently

- ❖ However,
$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$$
- ❖ In fact, it can be computed in
$$O(D)$$

Example: polynomial kernel

Let us examine more closely the inner products $\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$ for a pair of data points \mathbf{x}_m and \mathbf{x}_n .

Polynomial-based nonlinear basis functions consider the following $\phi(\mathbf{x})$:

$$\phi : \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

This gives rise to an inner product in a special form,

$$\begin{aligned}\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n) &= x_{m1}^2 x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2 x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\mathbf{x}_m^T \mathbf{x}_n)^2\end{aligned}$$

Namely, the inner product can be computed by a function $(\mathbf{x}_m^T \mathbf{x}_n)^2$ defined in terms of the original features, *without computing $\phi(\cdot)$* .

The Kernel Trick

Suppose we wish to compute

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$

Here ϕ maps \mathbf{x} and \mathbf{z} to a high dimensional space

The Kernel Trick: Save time/space by computing the value of $K(\mathbf{x}, \mathbf{z})$ by performing operations in the original space (without a feature transformation!)

Kernel functions

- ❖ A kernel function $k(\cdot, \cdot)$ satisfies the following properties
- ❖ For any x_m, x_n

$$k(x_m, x_n) = k(x_n, x_m) \text{ and } k(x_m, x_n) = \phi(x_m)^T \phi(x_n)$$

for some function $\phi(\cdot)$

Example: $k(x_m, x_n) = (x_m^T x_n)^2$ is a kernel function:

$$\begin{aligned}\phi(x_m)^T \phi(x_n) &= x_{m1}^2 x_{n1}^2 + 2x_{m1}x_{m2}x_{n1}x_{n2} + x_{m2}^2 x_{n2}^2 \\ &= (x_{m1}x_{n1} + x_{m2}x_{n2})^2 = (\mathbf{x}_m^T \mathbf{x}_n)^2\end{aligned}$$

Exercise

- ❖ Let $x \in R^2$, show $(4 + 9x_i^T x_j)^2$ is a valid kernel.

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}_{i=1}^m$

1. Initialize $w \leftarrow 0 \in R^{2n}$

2. For (x, y) in \mathcal{D} :

3. if $y w^T \begin{bmatrix} x \\ x^2 \end{bmatrix} \leq 0$

Assume $y \in \{1, -1\}$

4. $w \leftarrow w + y \begin{bmatrix} x \\ x^2 \end{bmatrix}$

5.

6. Return w

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T \begin{bmatrix} x \\ x^2 \end{bmatrix})$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}_{i=1}^m$

1. Initialize $w \leftarrow \mathbf{0}$
2. For (x, y) in \mathcal{D} :
3. if $y w^T \phi(x) \leq 0$
4. $w \leftarrow w + y \phi(x)$
- 5.
6. Return w

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T \phi(x))$

$$w^T \phi(x) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)$$

The Kernel Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}_{i=1}^m$

1. Initialize $\alpha \leftarrow \mathbf{0} \in R^m$ $w \leftarrow \mathbf{0}$
2. For (x_i, y_i) in \mathcal{D} : $y w^T \phi(x) \leq 0$
3. if $y_i \sum_j \alpha_j y_j K(x_i, x_j) \leq 0$ $y w^T \phi(x) \leq 0$
4. $\alpha_i \leftarrow \alpha_i + 1$ $w \leftarrow w + y \phi(x)$
5. Return w
- 6.

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^T \phi(x)) = \sum_i \alpha_i y_i K(x_i, x)$

$$w^T \phi(x) = \sum_i \alpha_i y_i \phi(x_i)^T \phi(x) = \sum_i \alpha_i y_i K(x_i, x)$$

Support Vector Machine

Recap: The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (\mathbf{x}, y) in \mathcal{D} :
3. if $y(\mathbf{w}^\top \mathbf{x}) \leq 0$
4. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- 5.
6. Return \mathbf{w}

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Recap: The Marginal Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (\mathbf{x}, y) in \mathcal{D} :
3. if $y(\mathbf{w}^\top \mathbf{x}) \leq \gamma$
4. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- 5.
6. Return \mathbf{w}

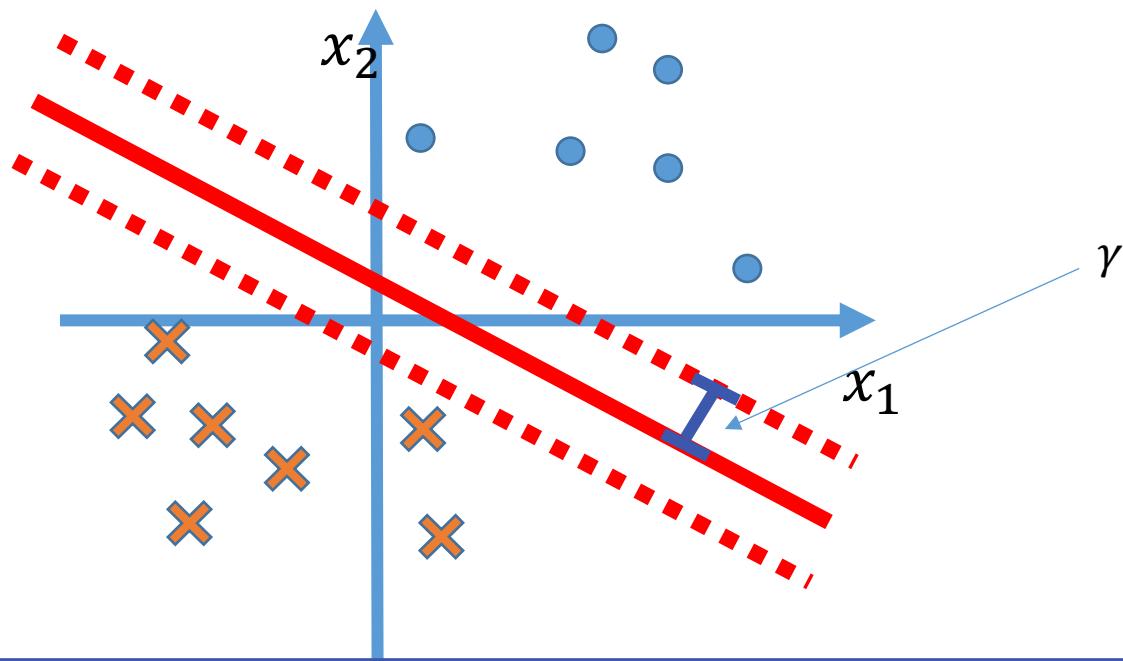
Assume $y \in \{1, -1\}$

$\gamma \geq 0$ is a hyper-parameter

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Marginal Perceptron



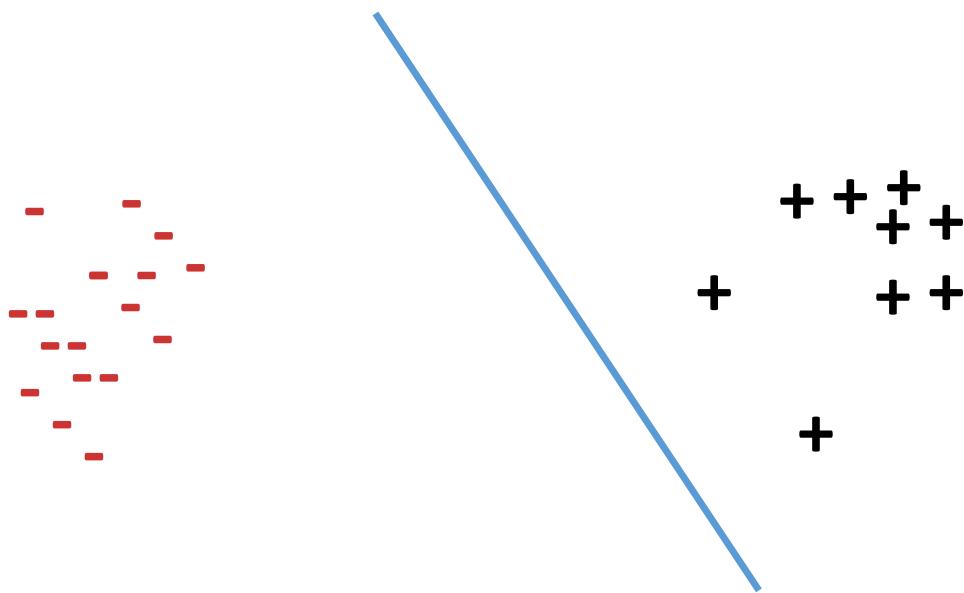
Is there a way to find out the best γ ! " # \$ % ! # & ! () *

This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

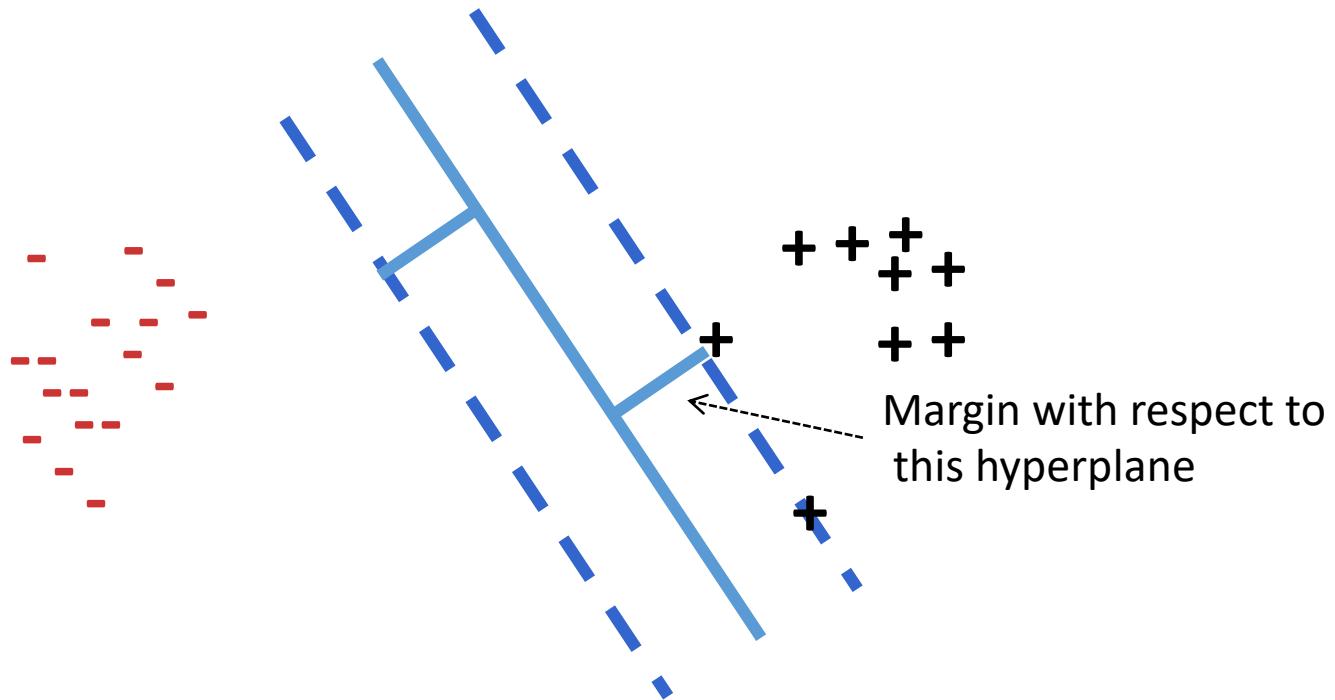
Recall: Margin

The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

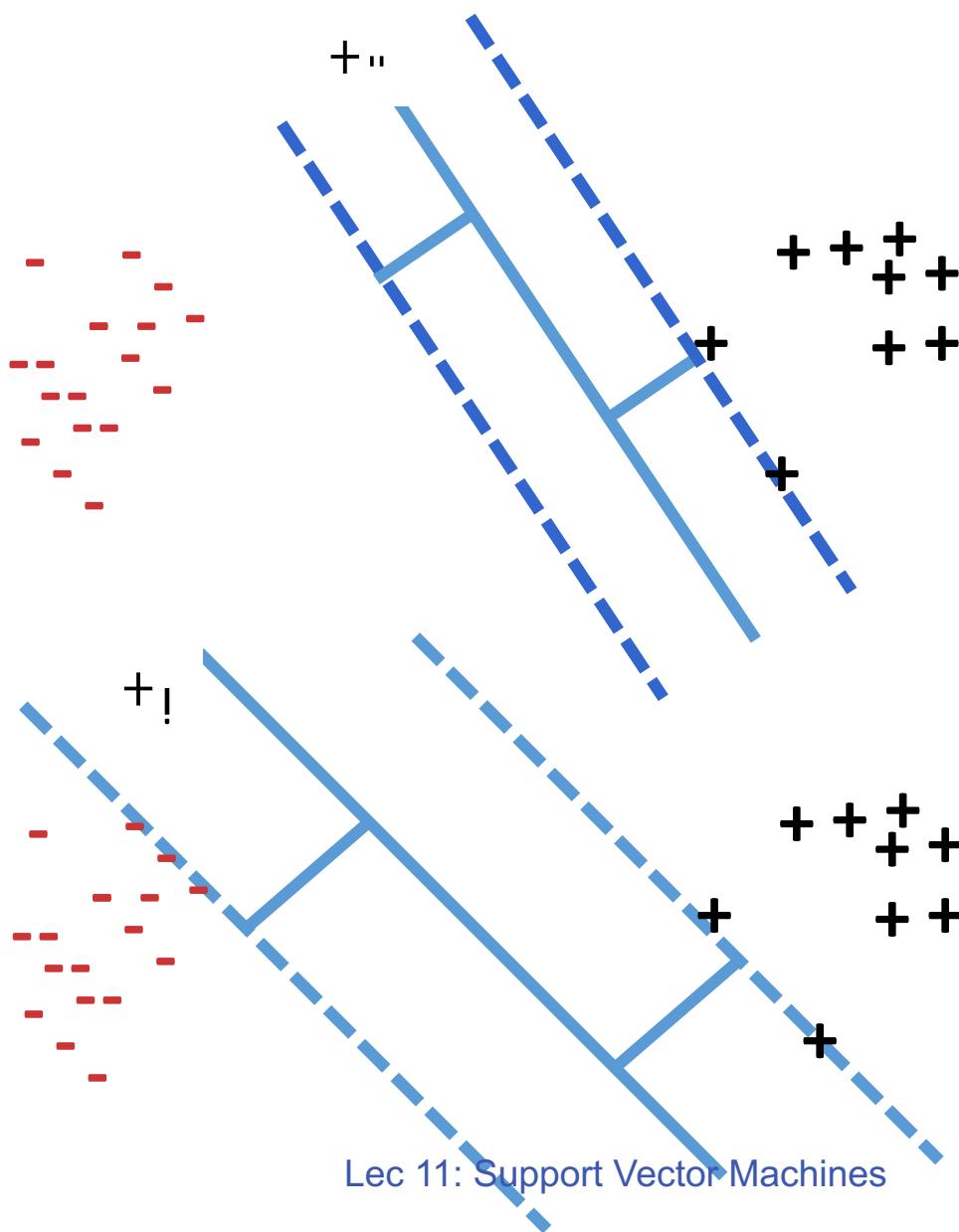


Recall: Margin

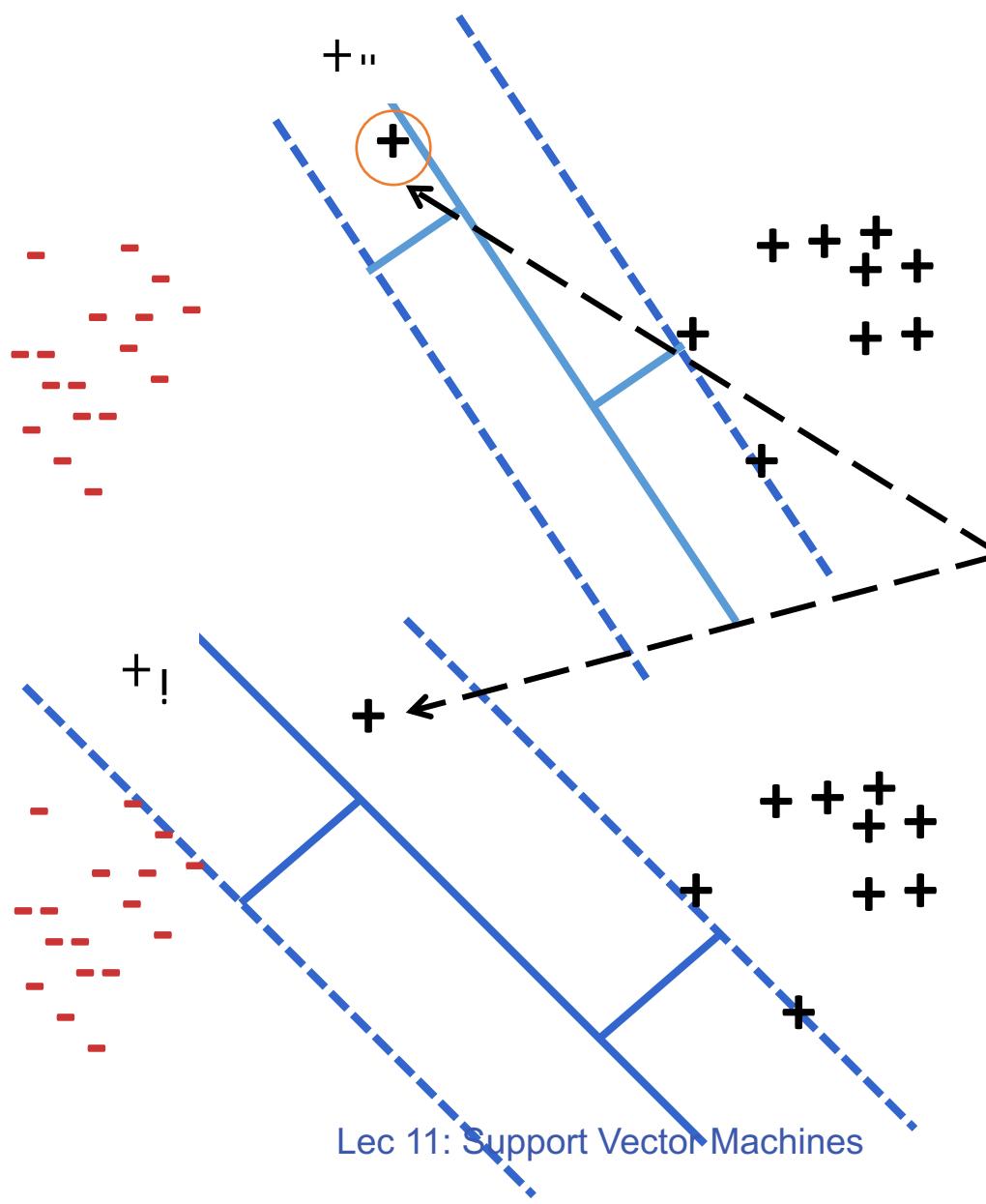
The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Which line is a better choice? Why?



Which line is a better choice? Why?



A new example, not from the training set might be misclassified if the margin is smaller

Advanced topic: (not included in the exam) Data dependent VC dimension

Theorem (Vapnik):

- ❖ Let H be the set of linear classifiers that separate the training set by a margin at least γ
- ❖ Then

$$VC(H) \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1$$

- ❖ R is the radius of the smallest sphere containing the data

Advanced topic: (not in the exam) Data dependent VC dimension

Theorem (Vapnik):

- ❖ Let H be the set of linear classifiers that separate the training set by a margin at least γ
- ❖ Then

$$VC(H) \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1$$

- ❖ R is the radius of the smallest sphere containing the data

Larger margin \rightarrow Lower VC dimension

Lower VC dimension \rightarrow Better generalization bound

Learning strategy

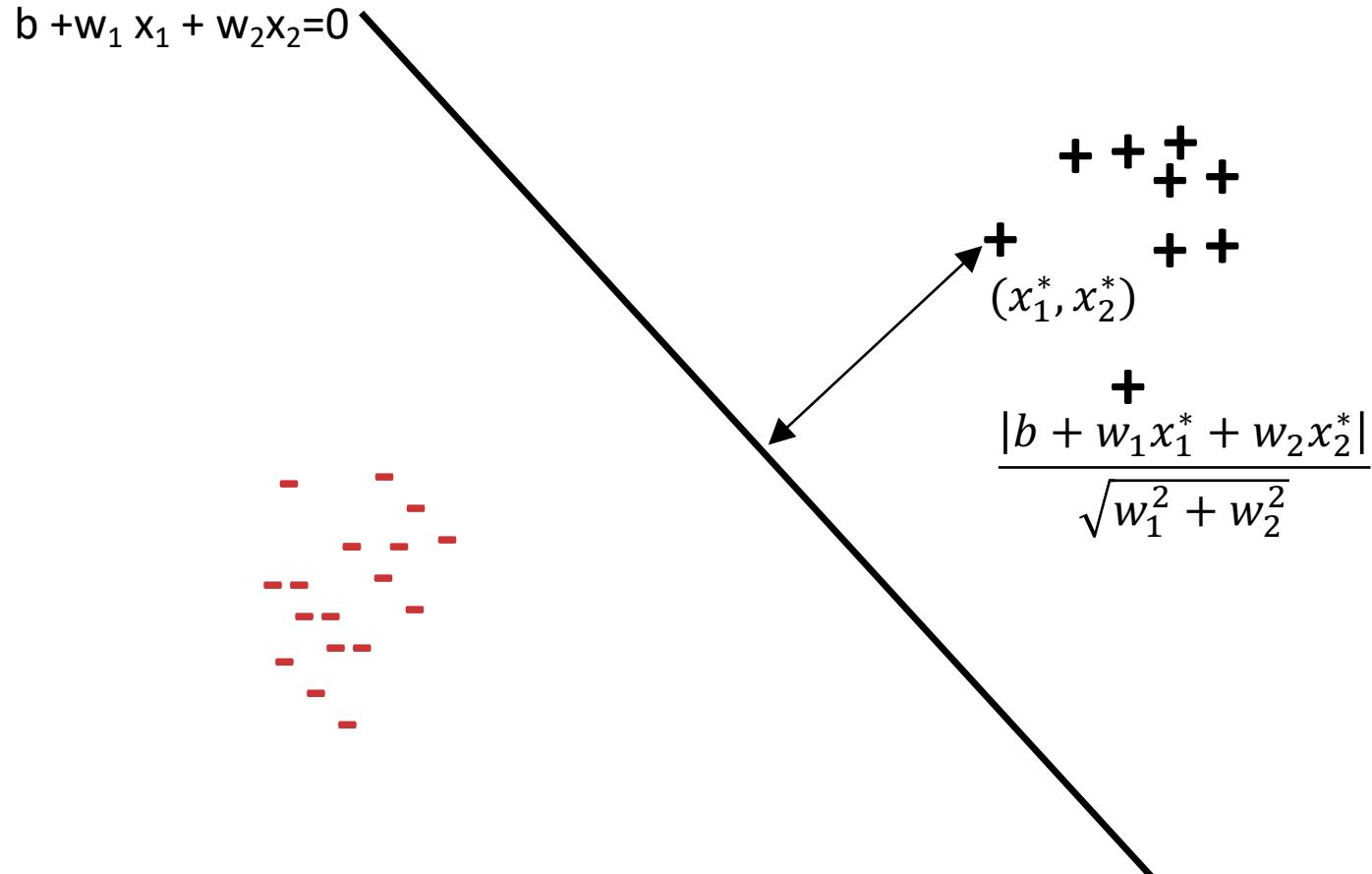
Find the linear separator that maximizes the margin

This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

Recall: The geometry of a linear classifier

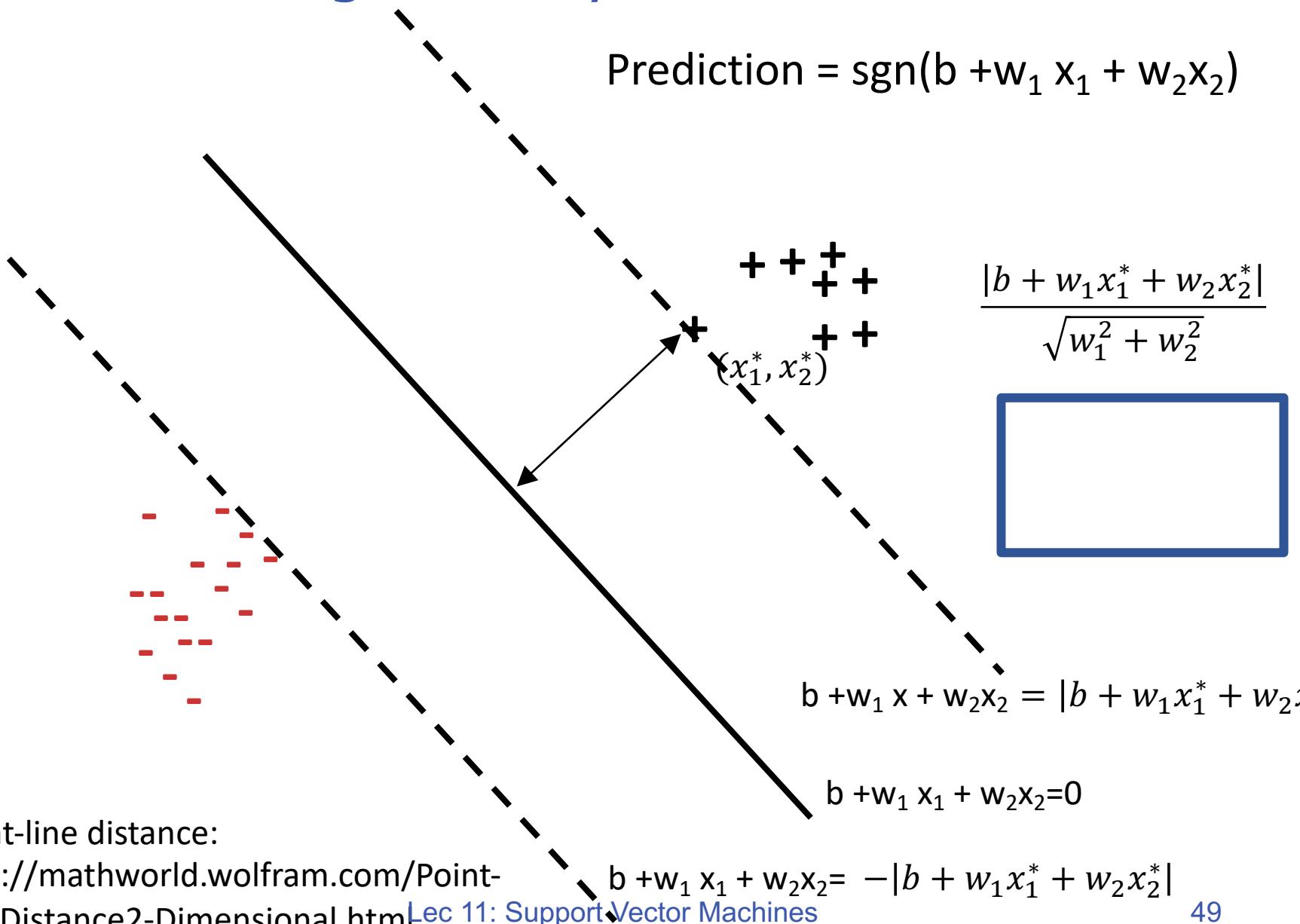
$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html> Lec 11: Support Vector Machines

Recall: The geometry of a linear classifier



Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

- ❖ We want $\max_{\mathbf{w}} \gamma$

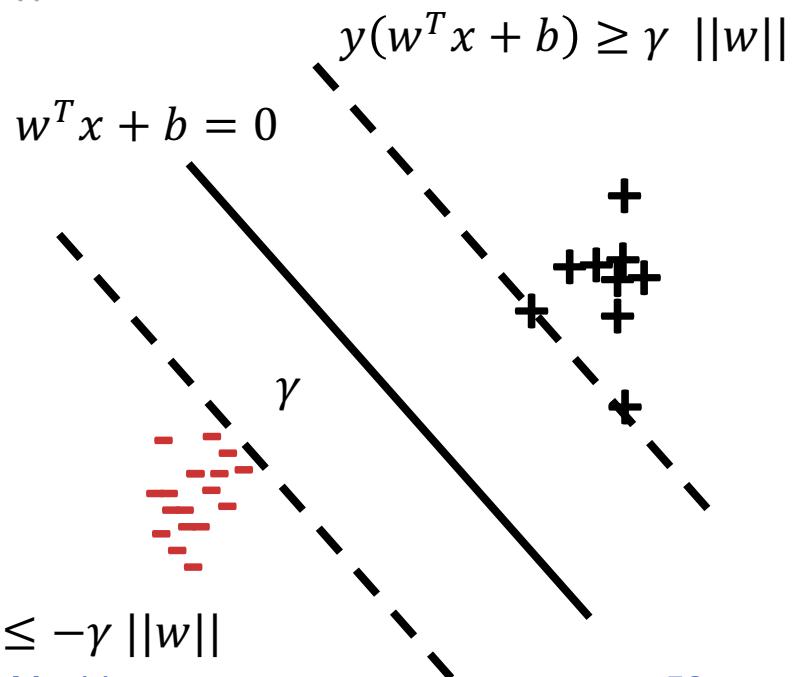
Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

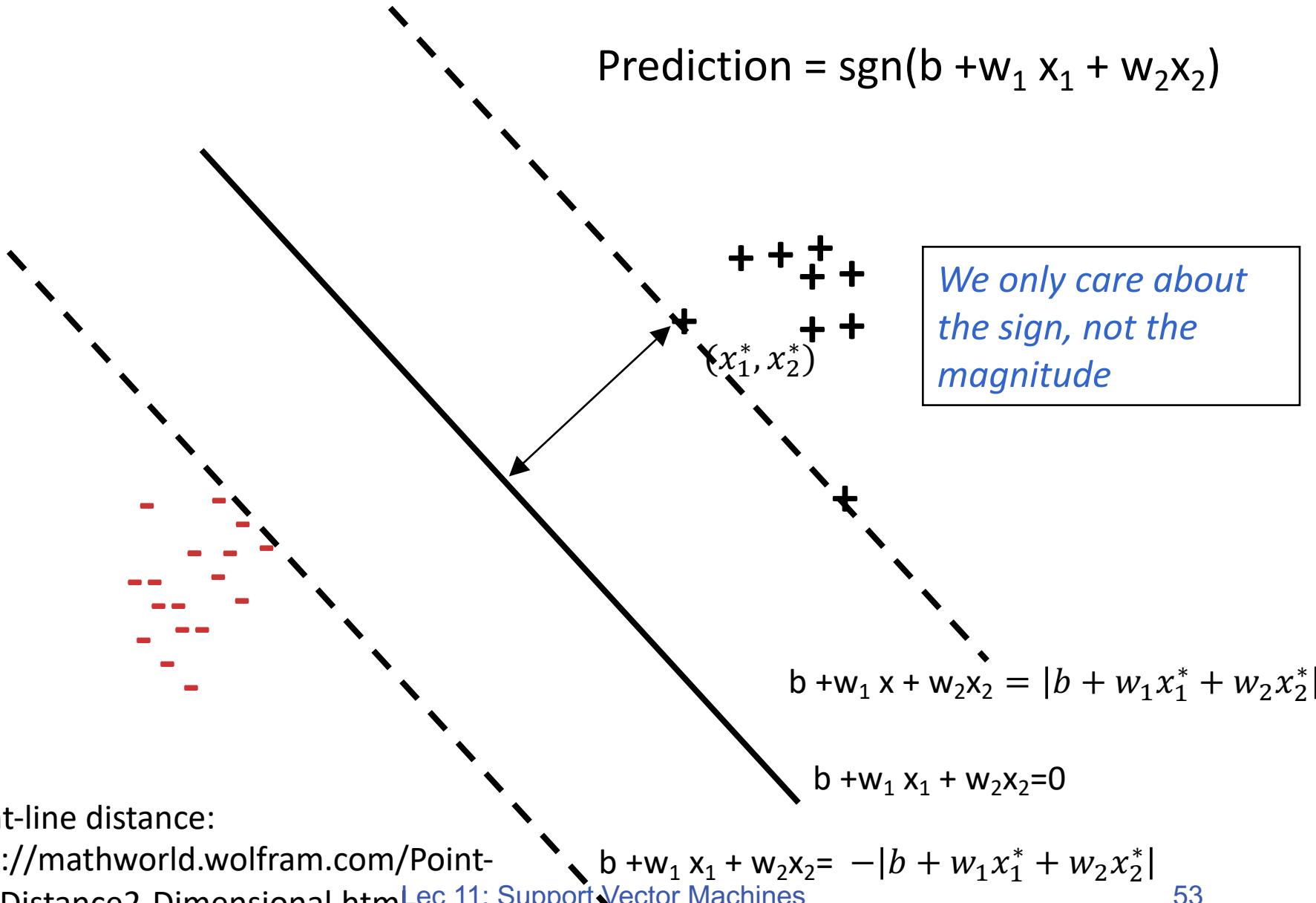
$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

$$s.t. \forall i, y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq \gamma \|\mathbf{w}\|$$

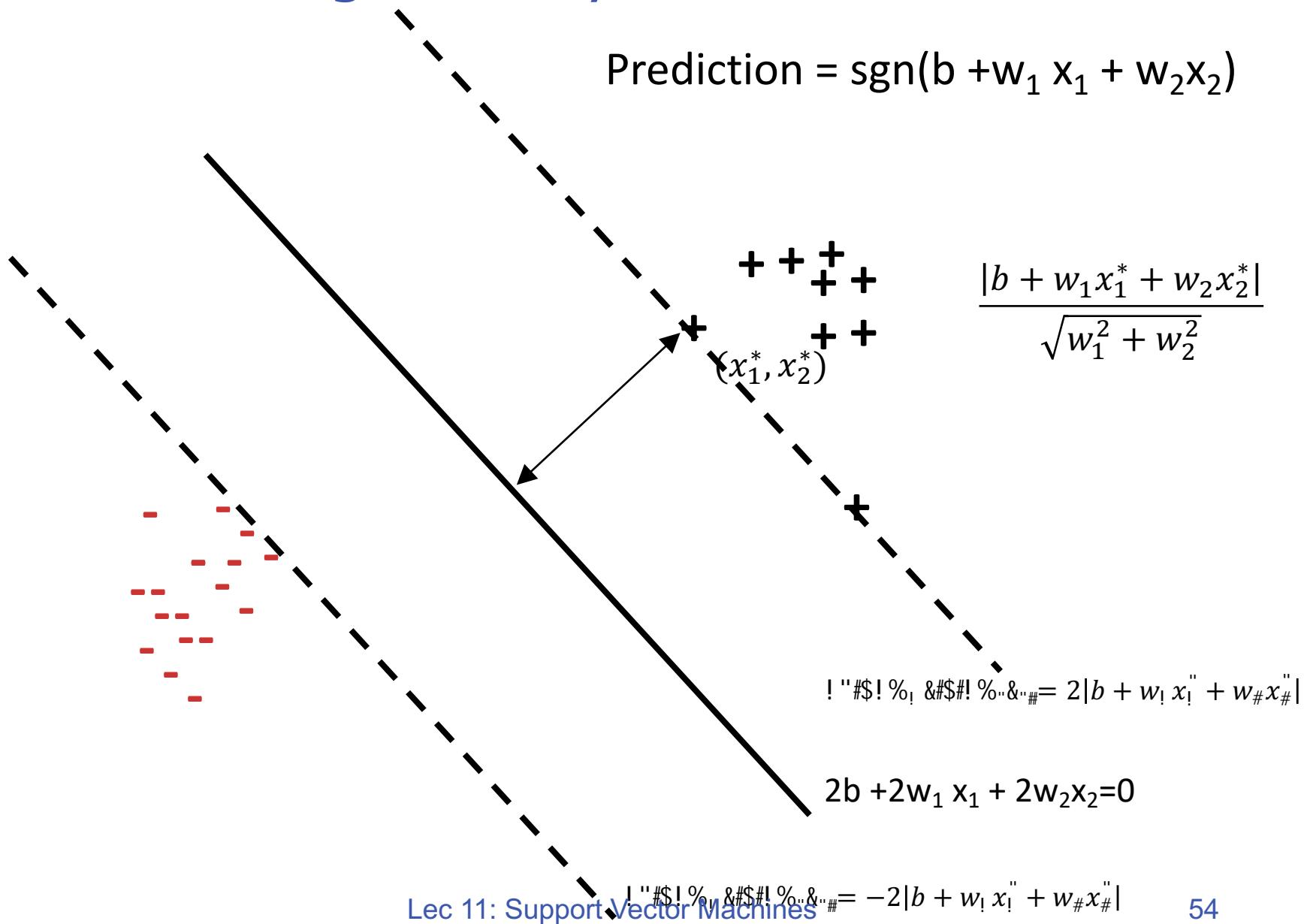
$$\max_{\gamma} \gamma$$



Recall: The geometry of a linear classifier

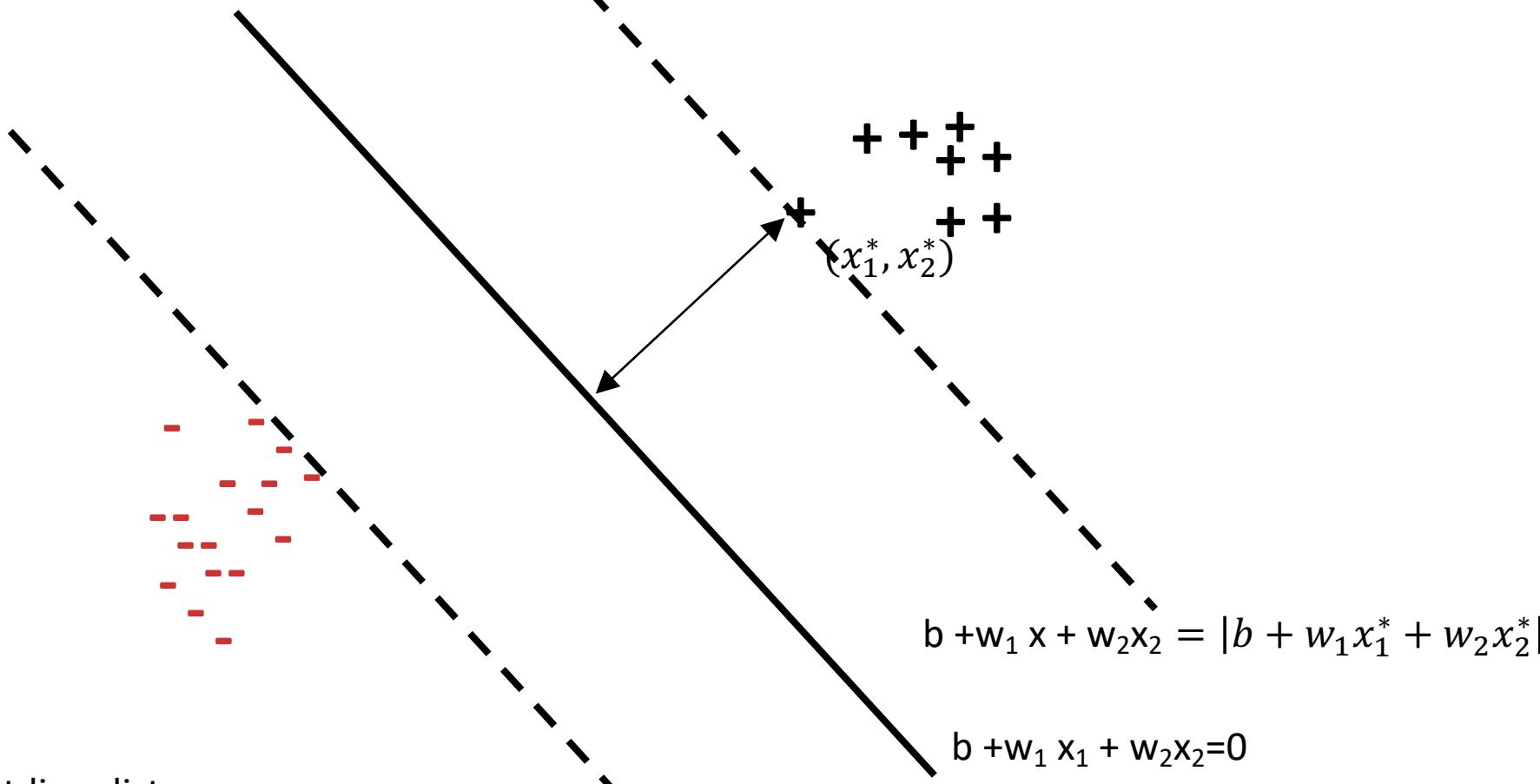


Recall: The geometry of a linear classifier



Recall: The geometry of a linear classifier

We have the freedom to scale up/down \mathbf{w} and b so that we can make $b + w_1x_1^* + w_2x_2^*=1$.



Point-line distance:

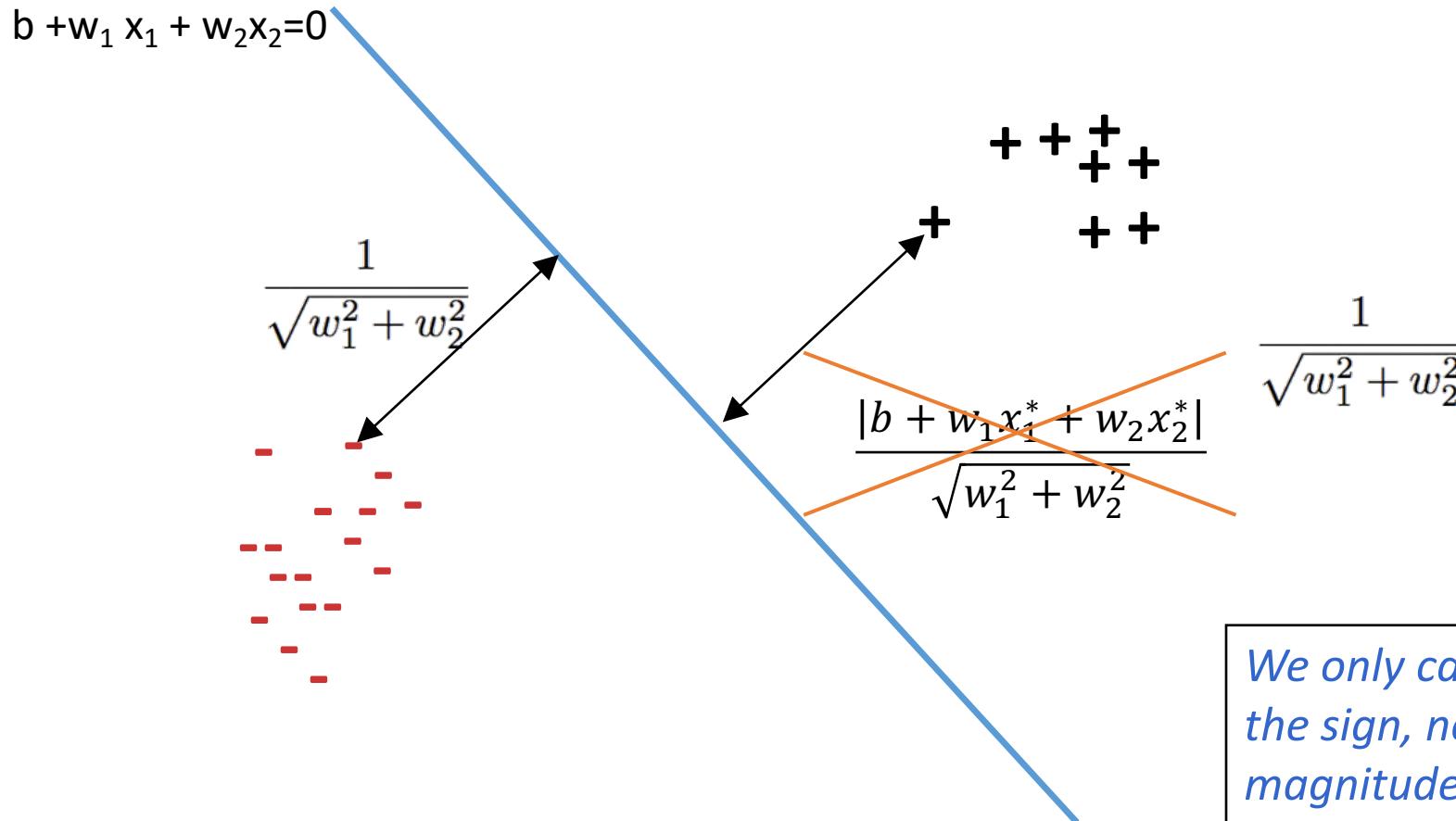
<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Lec 11: Support Vector Machines

$$b + w_1 x_1 + w_2 x_2 = -|b + w_1 x_1^* + w_2 x_2^*|$$

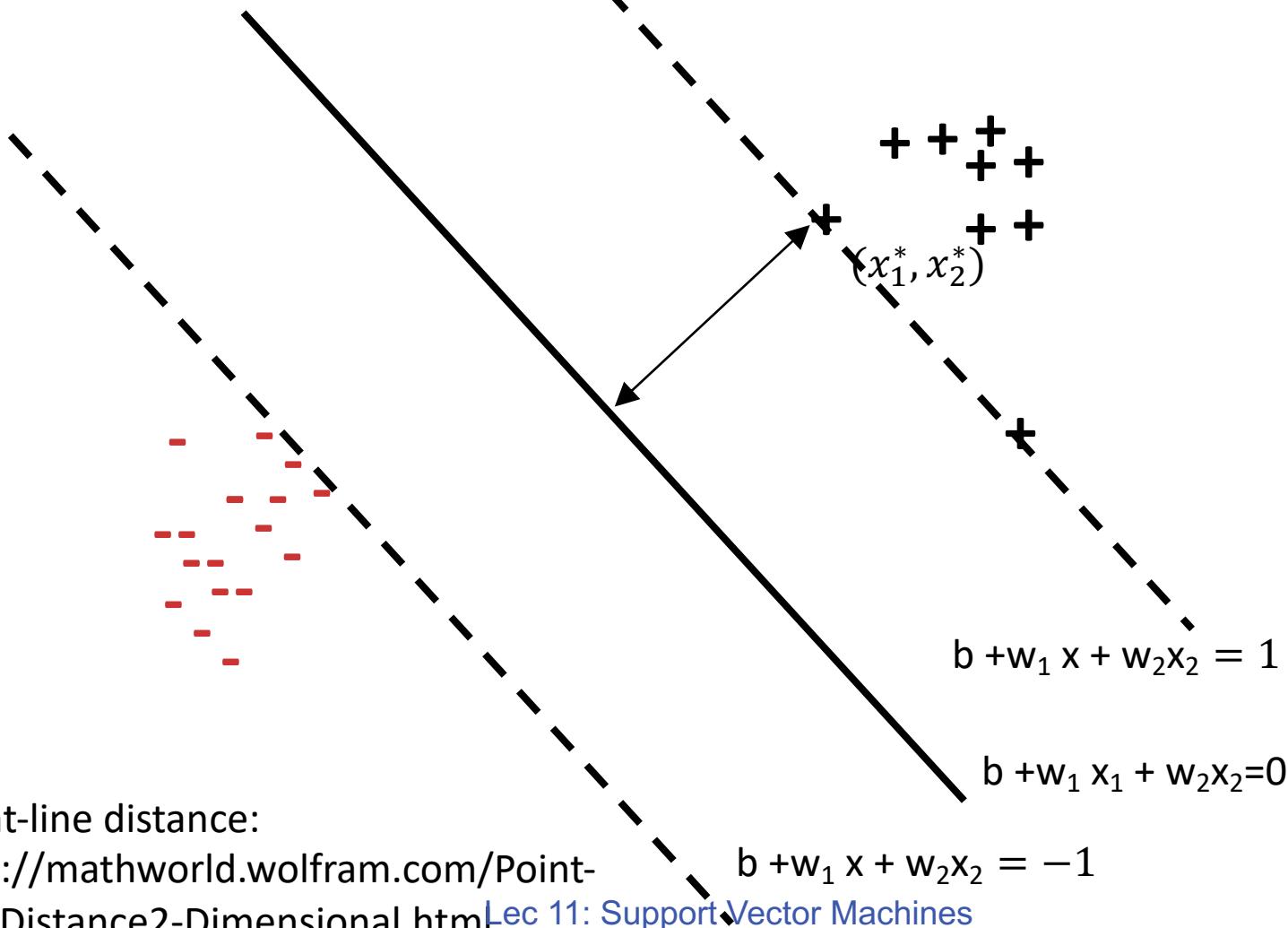
Recall: The geometry of a linear classifier

$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$



Recall: The geometry of a linear classifier

We have the freedom to scale up/down \mathbf{w} and b so that we can make $b + w_1x_1^* + w_2x_2^* = 1$.



Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

- ❖ We want $\max_{\mathbf{w}} \gamma$
- ❖ We only care about the sign of \mathbf{w} and b in the end and not the magnitude
 - ❖ Set the absolute score (functional margin) of the closest point to be 1 and allow \mathbf{w} to adjust itself

$\max_{\mathbf{w}} \gamma$ is equivalent to $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$ in this setting

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Max-margin classifiers

❖ Learning problem:

Mimimizing gives us $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Max-margin classifiers

- ❖ Learning problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Mimimizing gives us $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$

This condition is true for every example, specifically, for the example closest to the separator

- ❖ This is called the “hard” Support Vector Machine

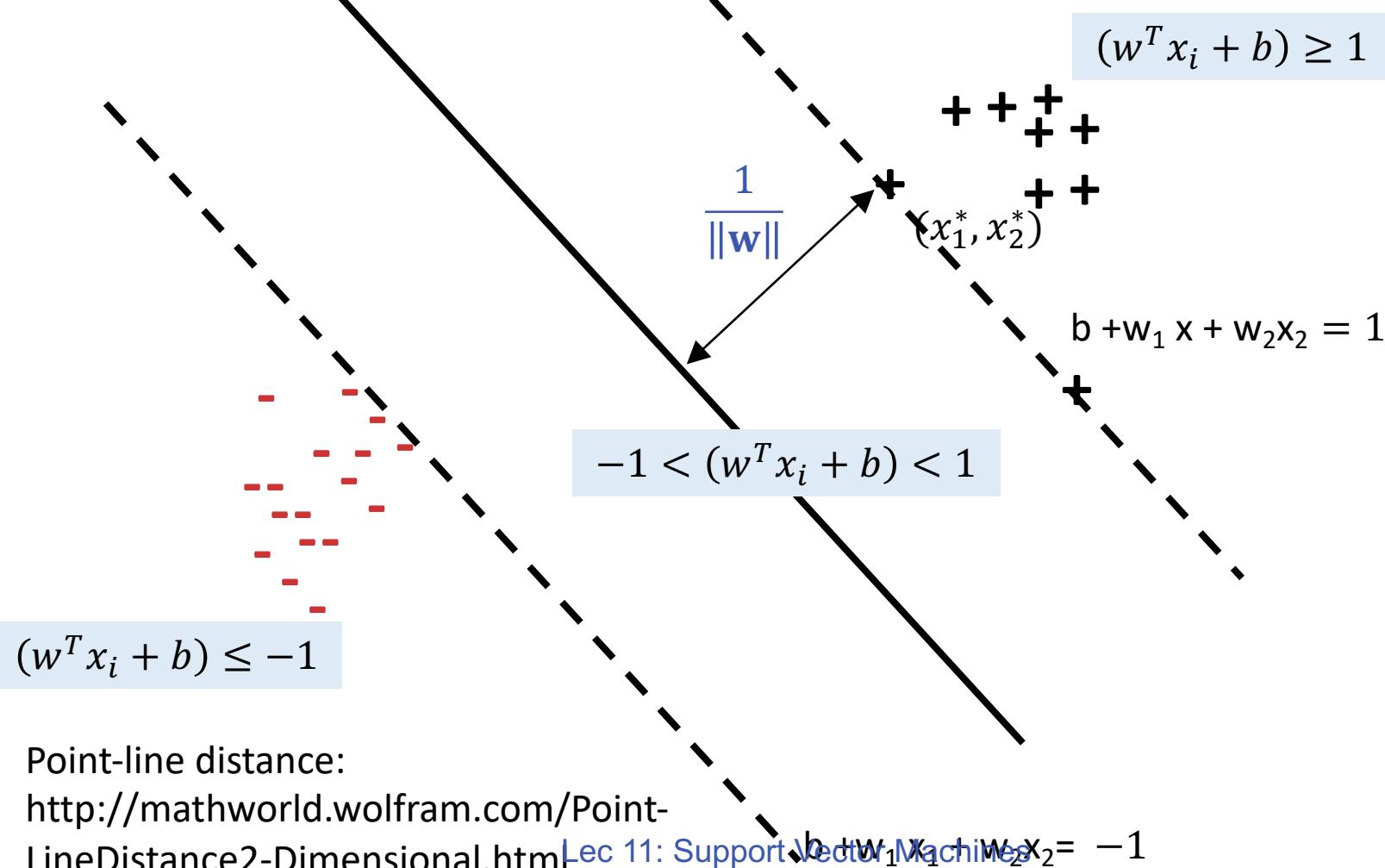
We will look at how to solve this optimization problem later

Hard SVM

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

$$b + w_1 x_1 + w_2 x_2 = 0$$



Point-line distance:

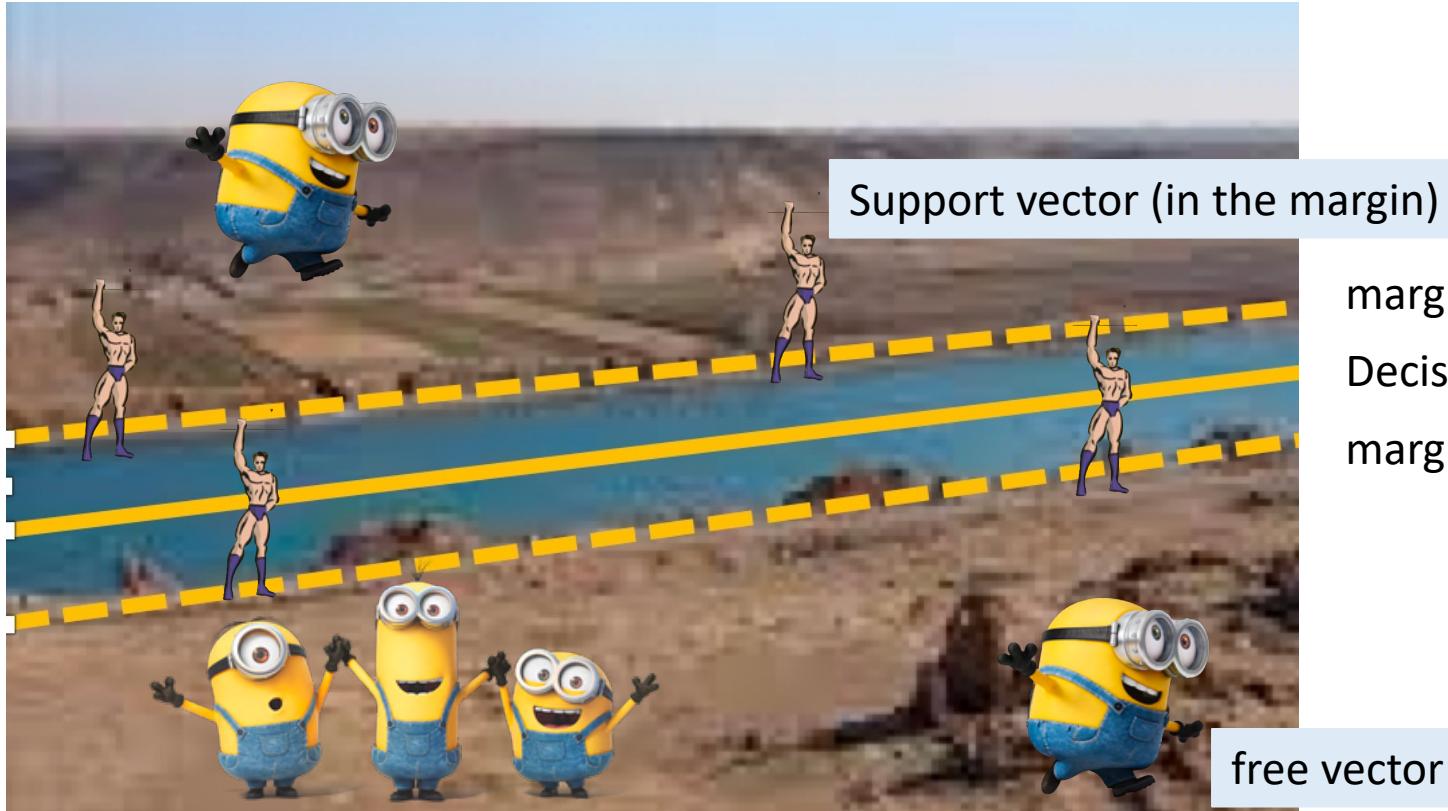
<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Lec 11: Support Vector Machines

$b + w_1 x_1 + w_2 x_2 = -1$

Hard support vector machines?

No training error can be made. All support vectors are on the boundary



What if the data is not separable?

Hard SVM

$$\min_{w,b} \frac{1}{2} w^T w$$

Maximize margin

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Every example has an functional margin of at least 1

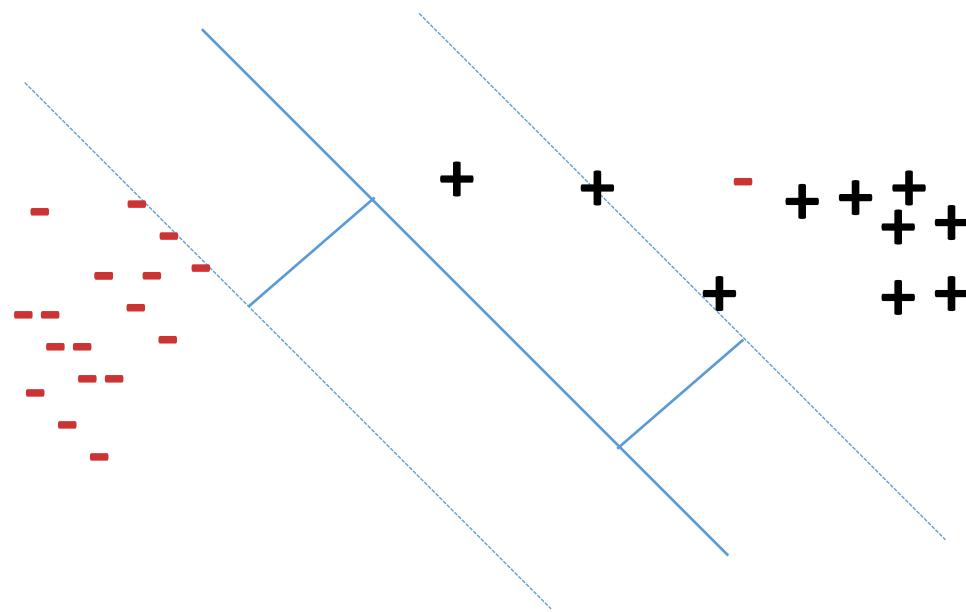
- ❖ This is a constrained optimization problem
- ❖ If the data is not separable, there is no **w** that will classify the data
- ❖ Infeasible problem, no solution!



If you made an mistake in your midterm, got 0 point!

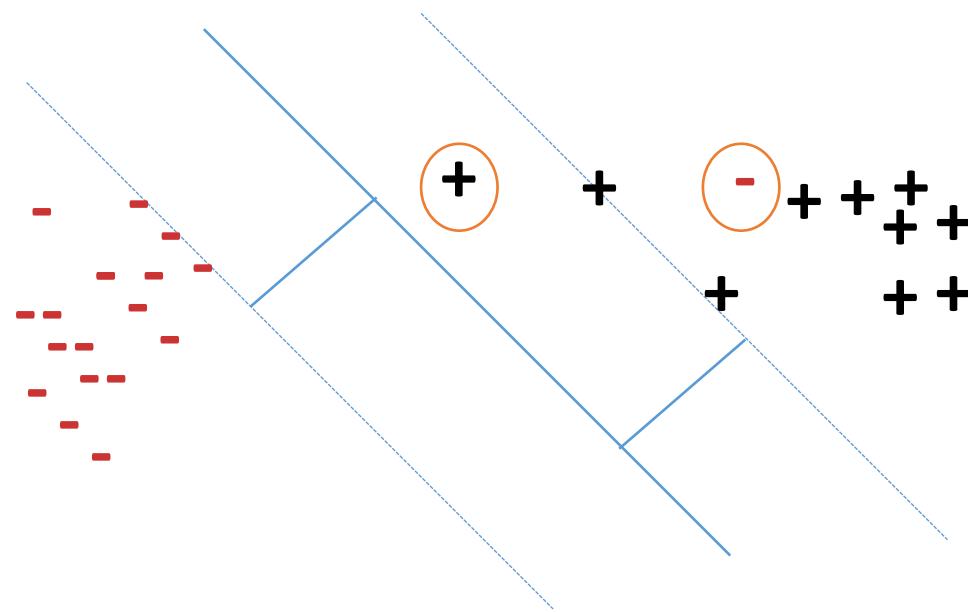
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin” or “make mistake”



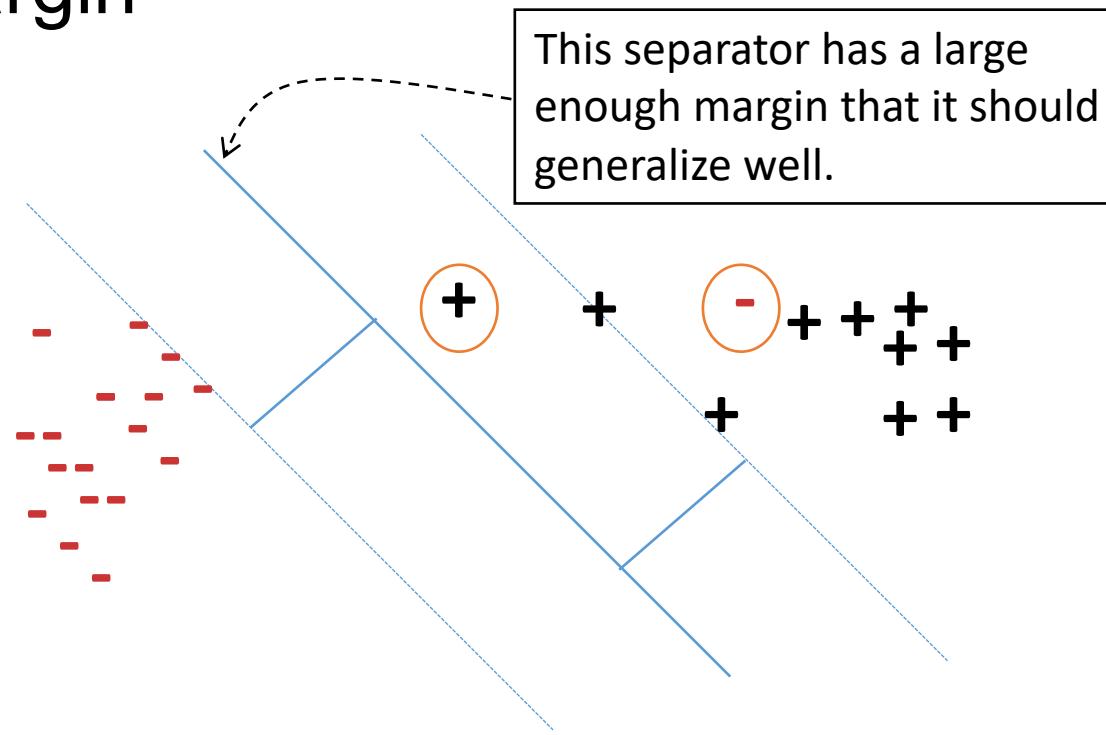
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”



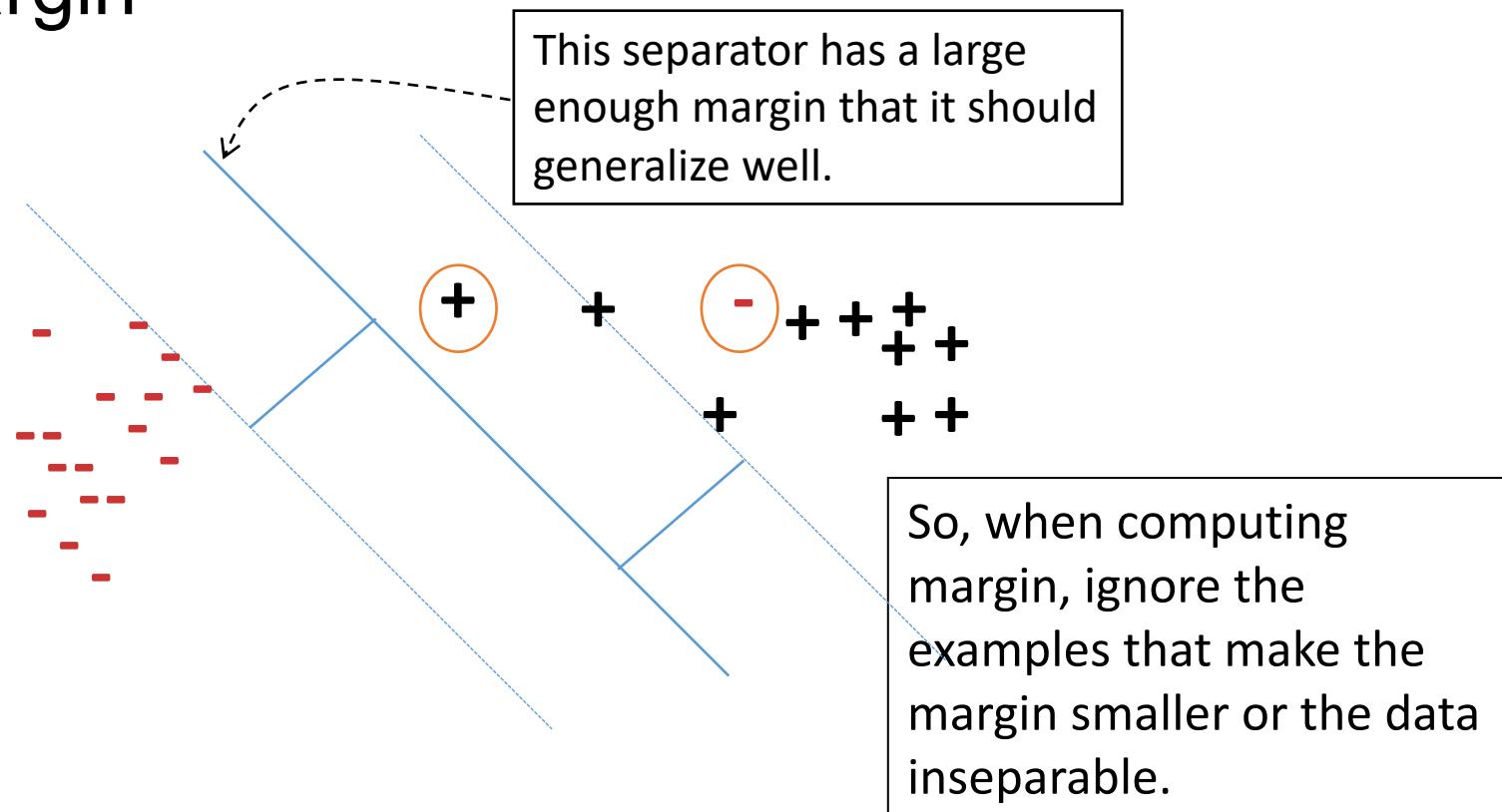
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”



Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”



Soft SVM

❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w$$

Maximize margin

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Every example has an
functional margin of at least 1

Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ Introduce one *slack variable* ξ_i per example

- ❖ And require $y_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ Introduce one *slack variable* ξ_i per example

- ❖ And require $y_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

Intuition: The slack variable allows examples to “break” into the margin

If the slack value is zero, then the example is either on or outside the margin

Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ New optimization problem for learning

$$\min_{w,b, \xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

C is the hyper-parameter

Lecture 14: Support Vector Machines Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Recap: The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (\mathbf{x}, y) in \mathcal{D} :
3. if $y(\mathbf{w}^\top \mathbf{x}) \leq 0$
4. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- 5.
6. Return \mathbf{w}

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Recap: The Marginal Perceptron Algorithm

Given a training set $\mathcal{D} = \{(\mathbf{x}, y)\}$

1. Initialize $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (\mathbf{x}, y) in \mathcal{D} :
3. if $y(\mathbf{w}^\top \mathbf{x}) \leq \gamma$
4. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
- 5.
6. Return \mathbf{w}

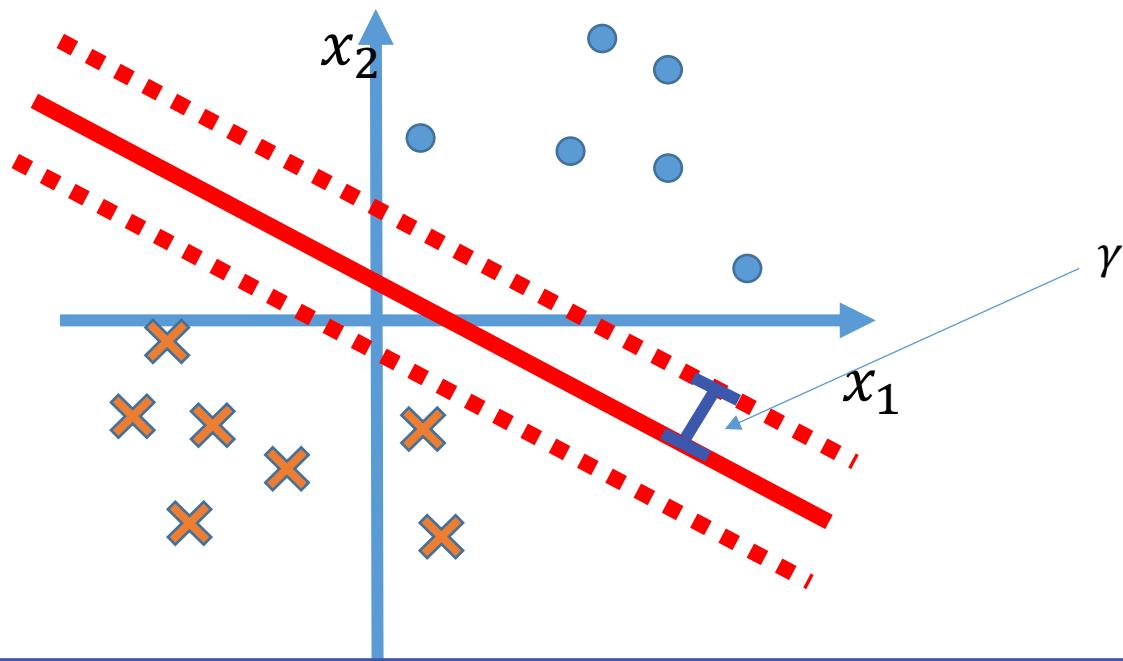
Assume $y \in \{1, -1\}$

$\gamma \geq 0$ is a hyper-parameter

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(\mathbf{w}^\top \mathbf{x}^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Marginal Perceptron



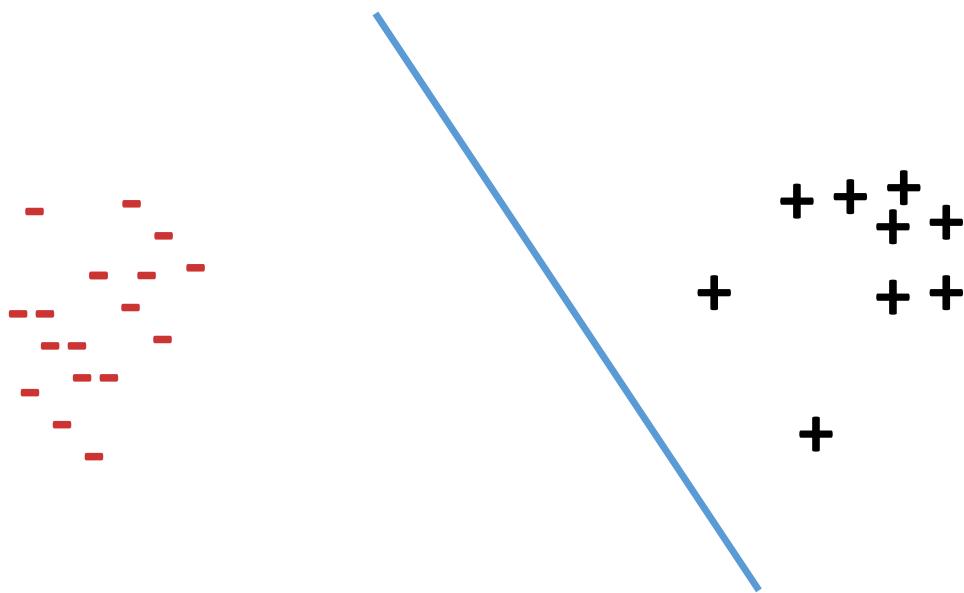
Is there a way to find out the best ! ! " # \$ % ! # & ! () *

This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

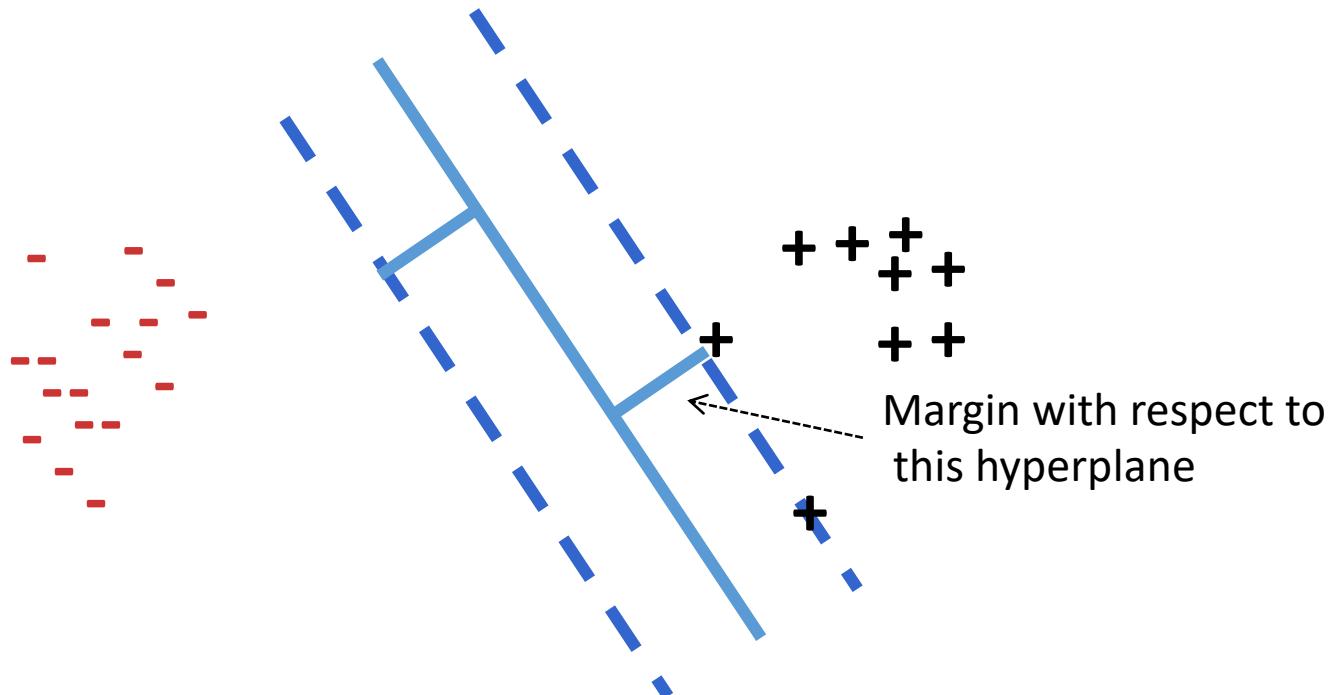
Recall: Margin

The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.

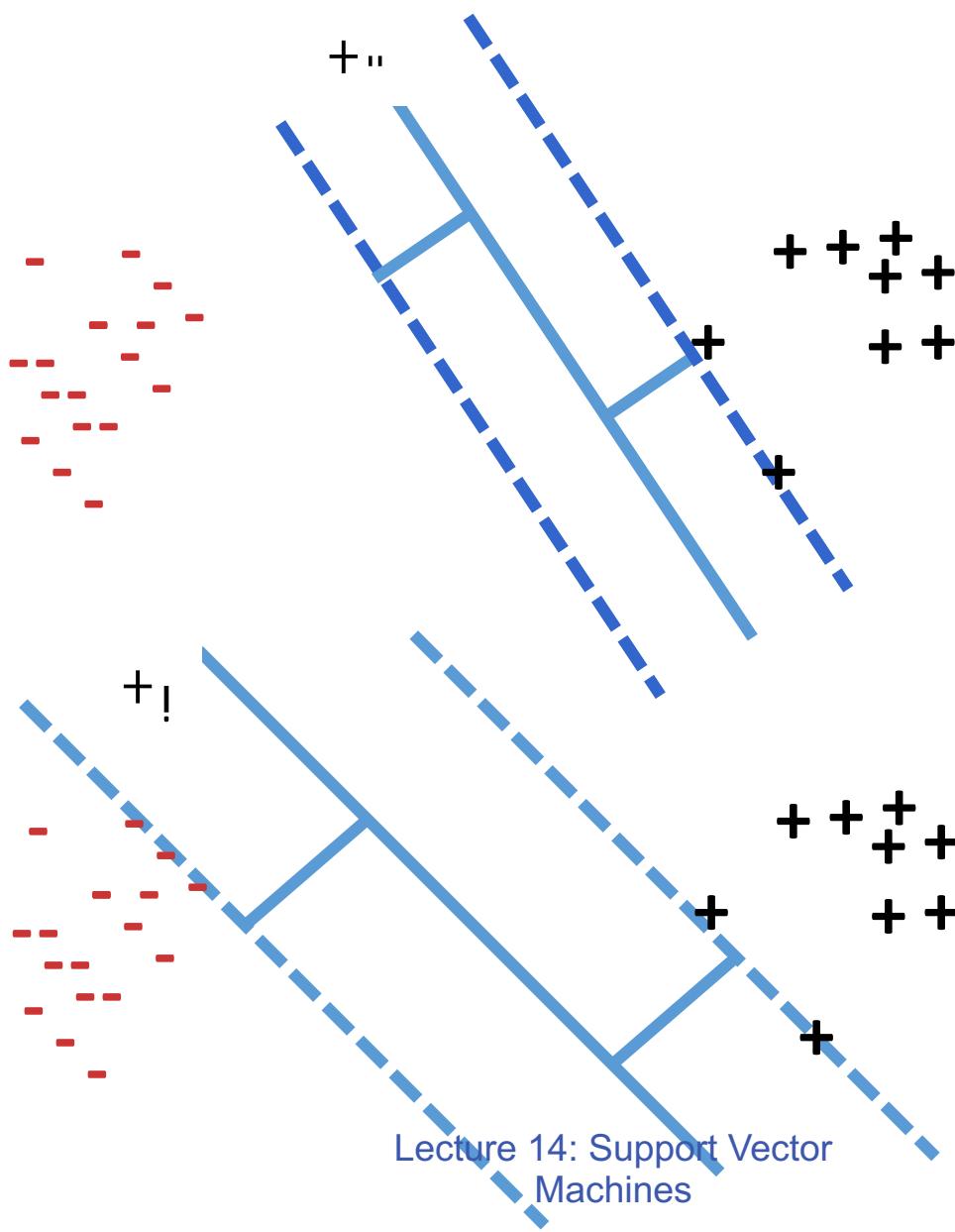


Recall: Margin

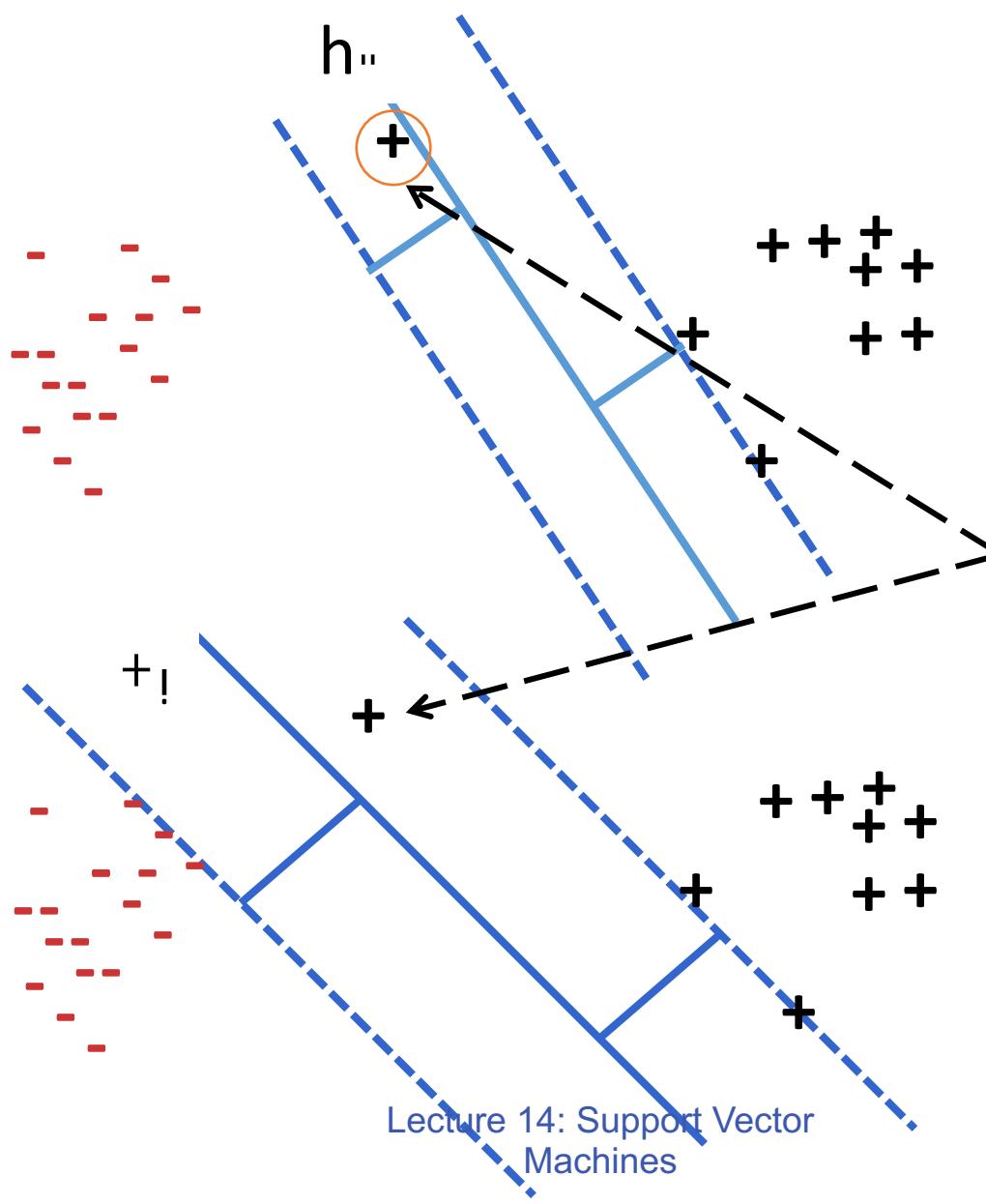
The **margin** of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Which line is a better choice? Why?



Which line is a better choice? Why?



A new example, not from the training set might be misclassified if the margin is smaller

Advanced topic: (not included in the exam) Data dependent VC dimension

Theorem (Vapnik):

- ❖ Let H be the set of linear classifiers that separate the training set by a margin at least γ
- ❖ Then

$$VC(H) \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1$$

- ❖ R is the radius of the smallest sphere containing the data

Advanced topic: (not in the exam) Data dependent VC dimension

Theorem (Vapnik):

- ❖ Let H be the set of linear classifiers that separate the training set by a margin at least γ
- ❖ Then

$$VC(H) \leq \min\left(\frac{R^2}{\gamma^2}, d\right) + 1$$

- ❖ R is the radius of the smallest sphere containing the data

Larger margin \rightarrow Lower VC dimension

Lower VC dimension \rightarrow Better generalization bound

Learning strategy

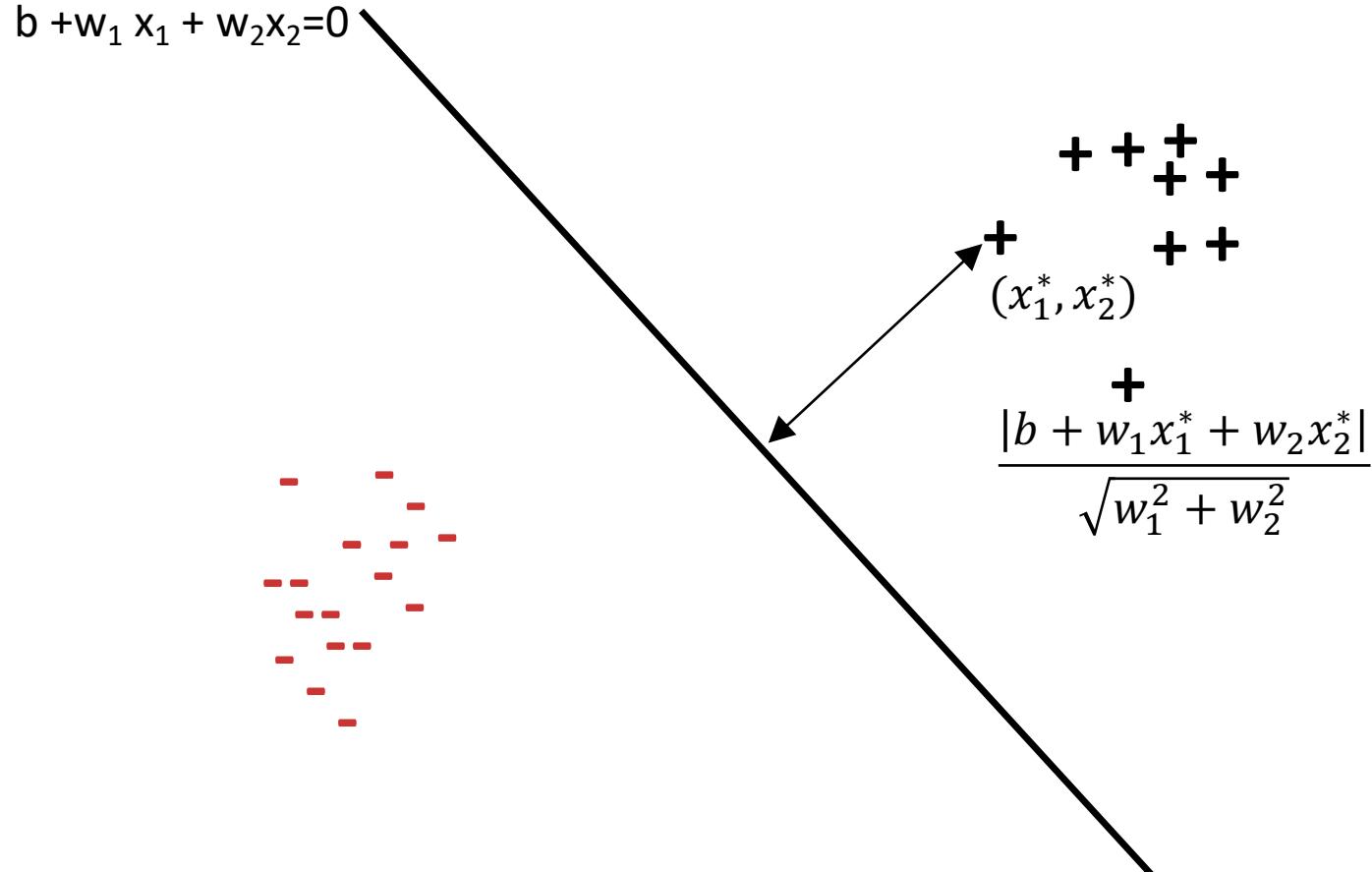
Find the linear separator that maximizes the margin

This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

Recall: The geometry of a linear classifier

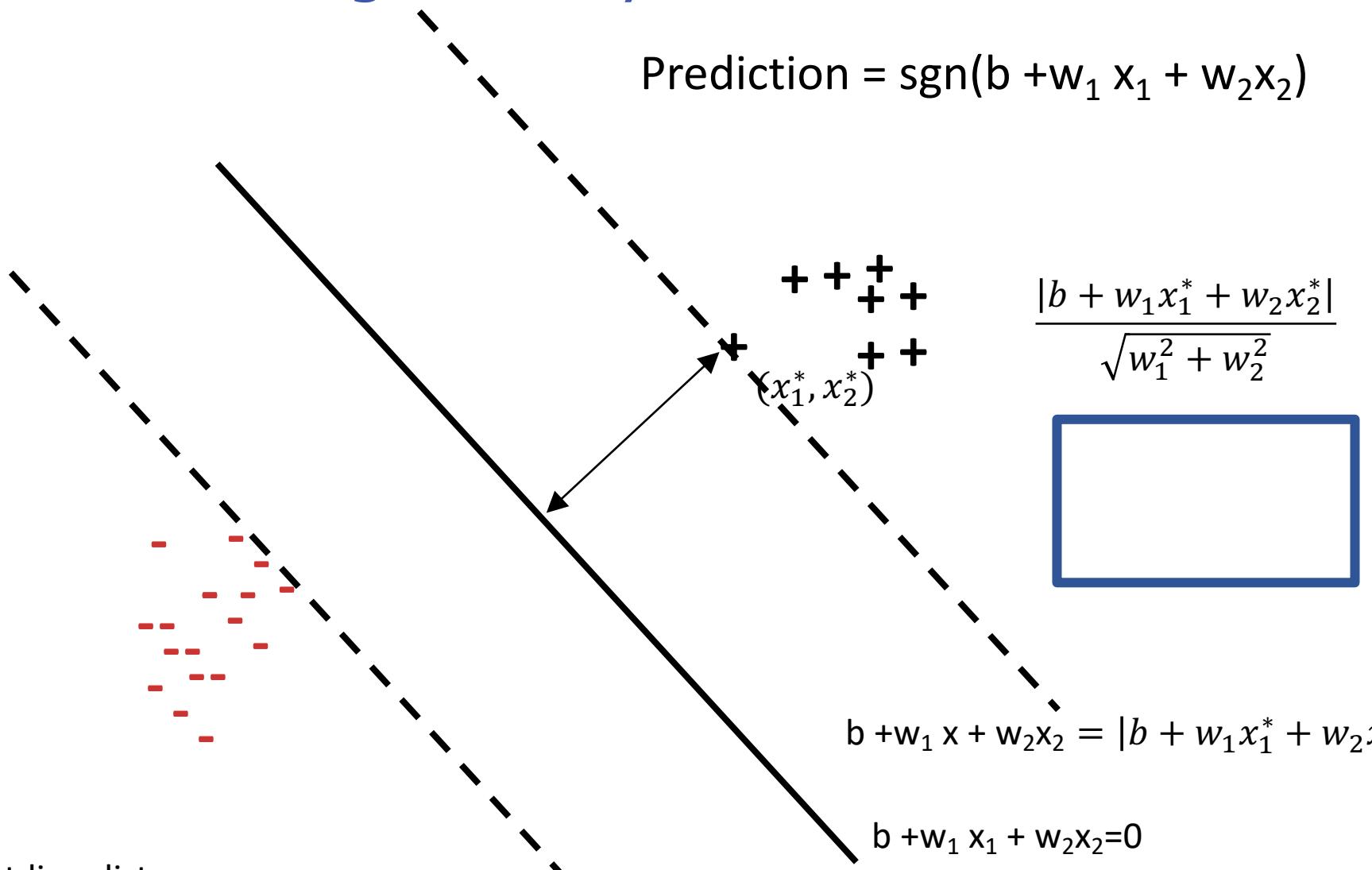
$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Recall: The geometry of a linear classifier



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

- ❖ We want $\max_{\mathbf{w}} \gamma$

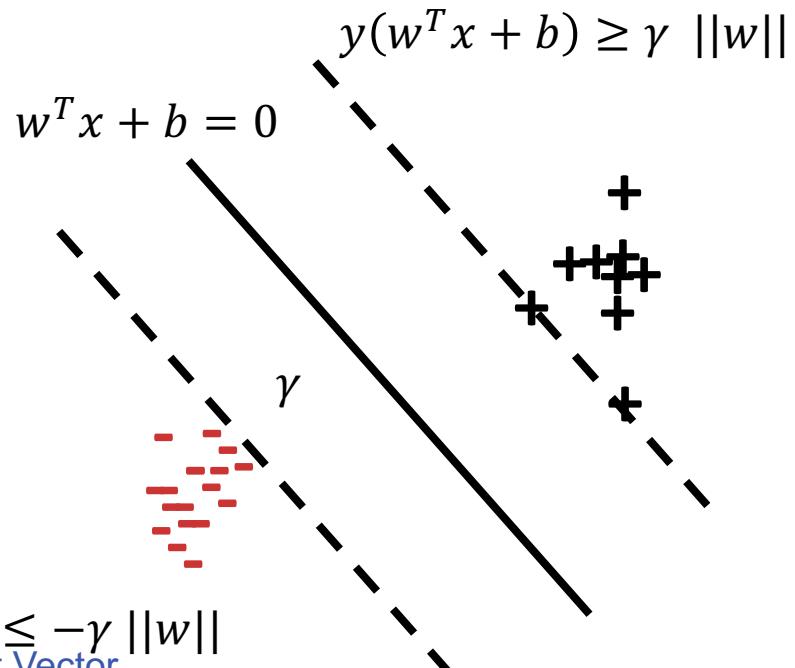
Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

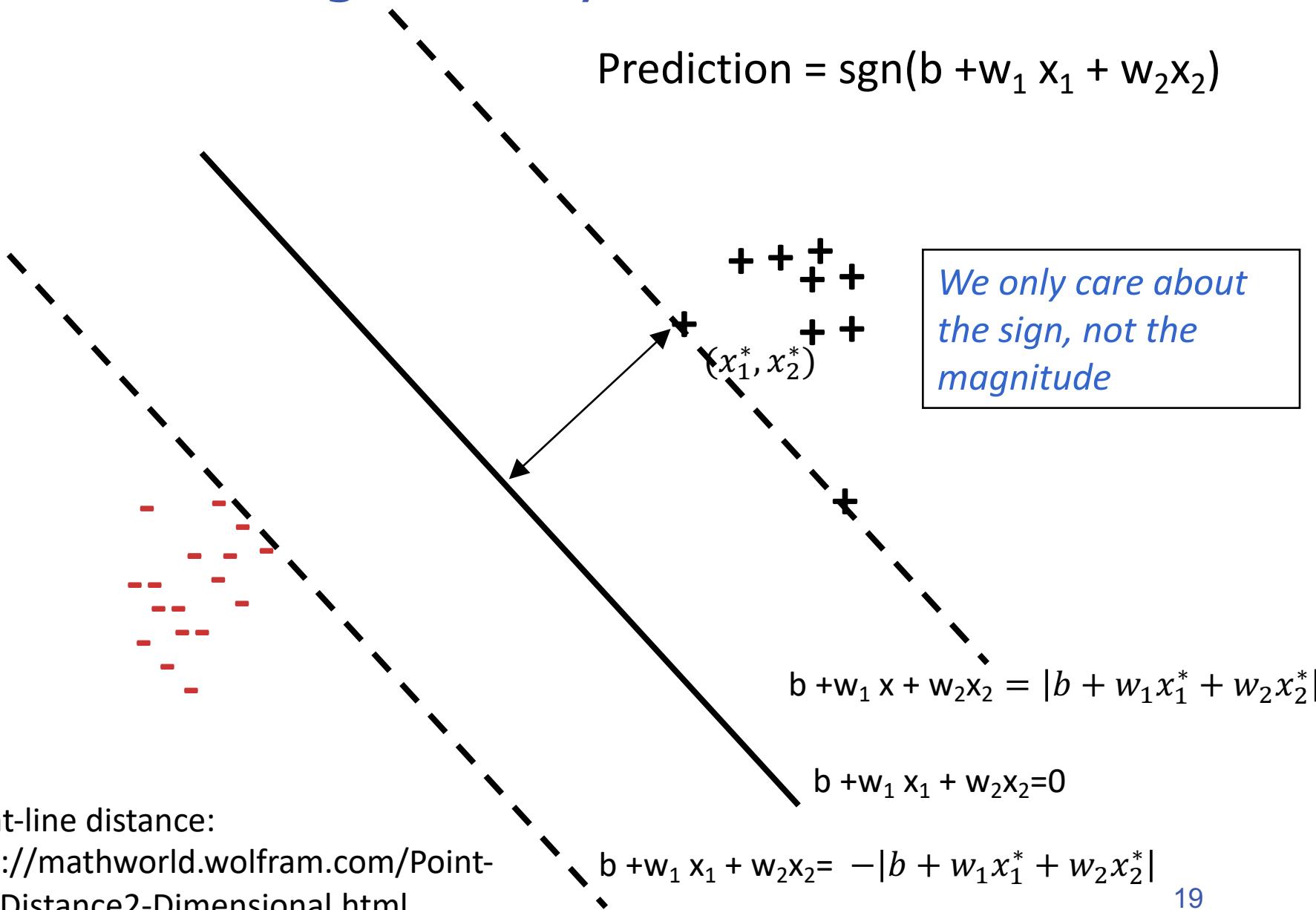
$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

$$s.t. \forall i, y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq \gamma \|\mathbf{w}\|$$

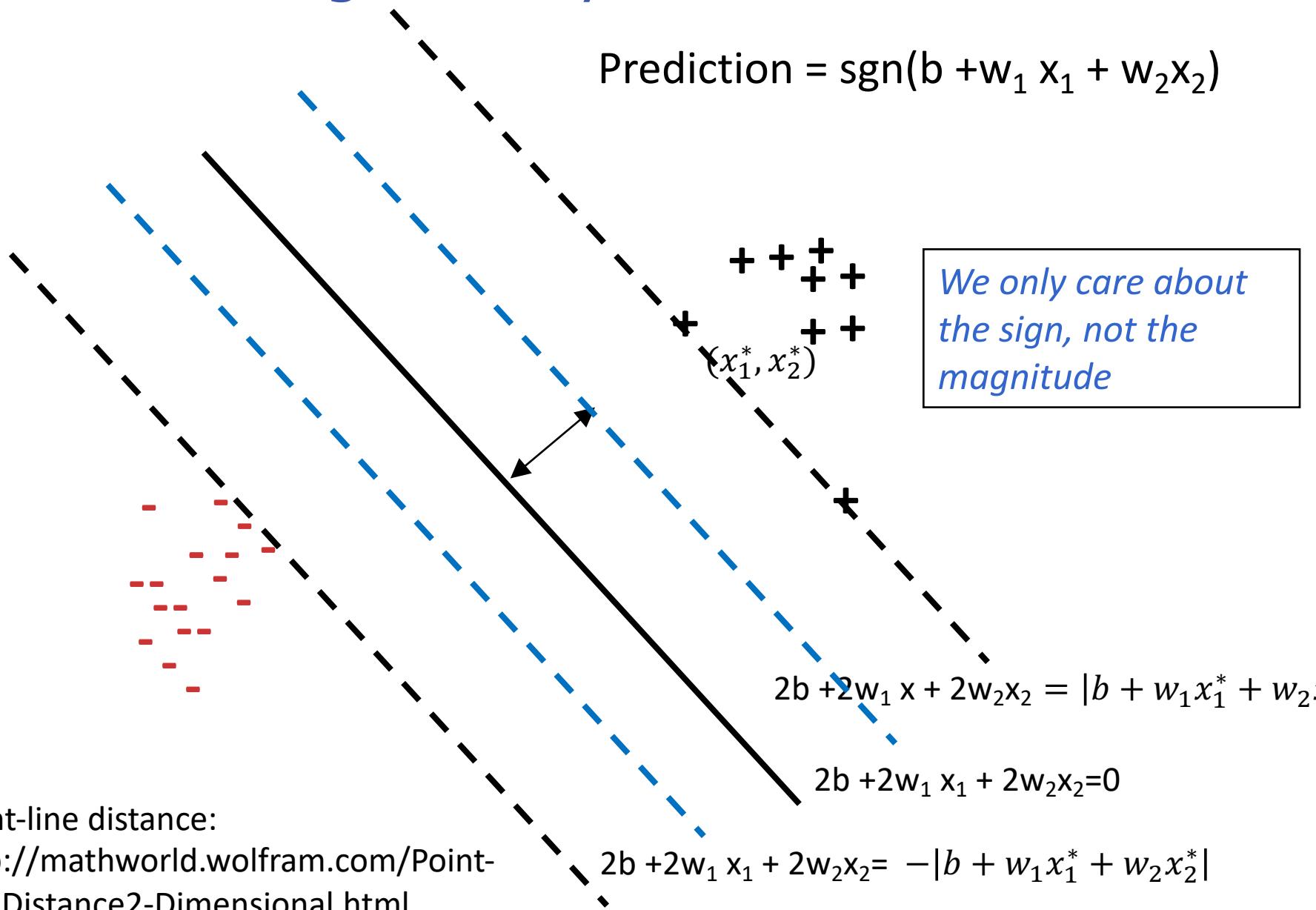
$$\max_{\gamma} \gamma$$



Recall: The geometry of a linear classifier

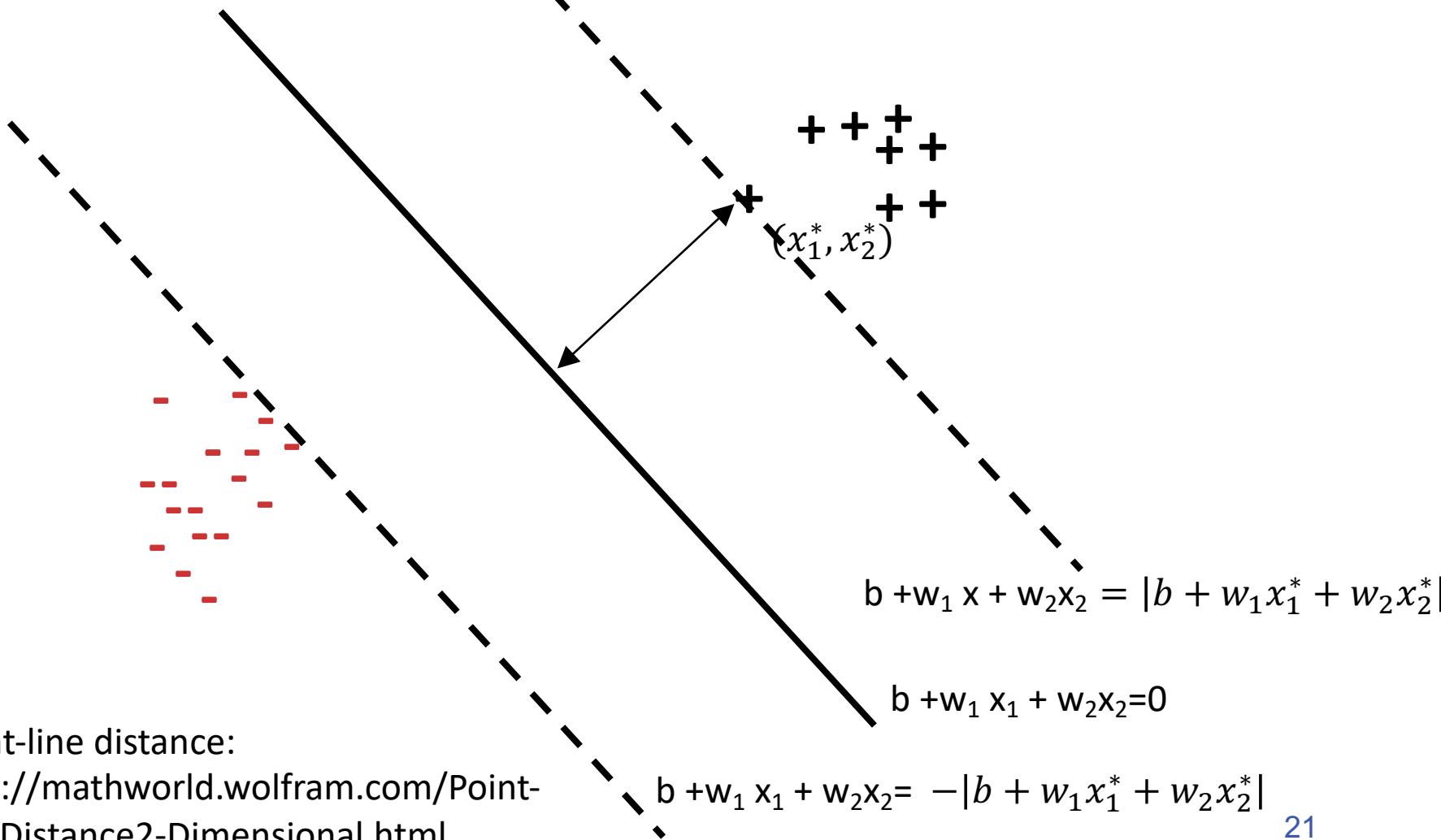


Recall: The geometry of a linear classifier



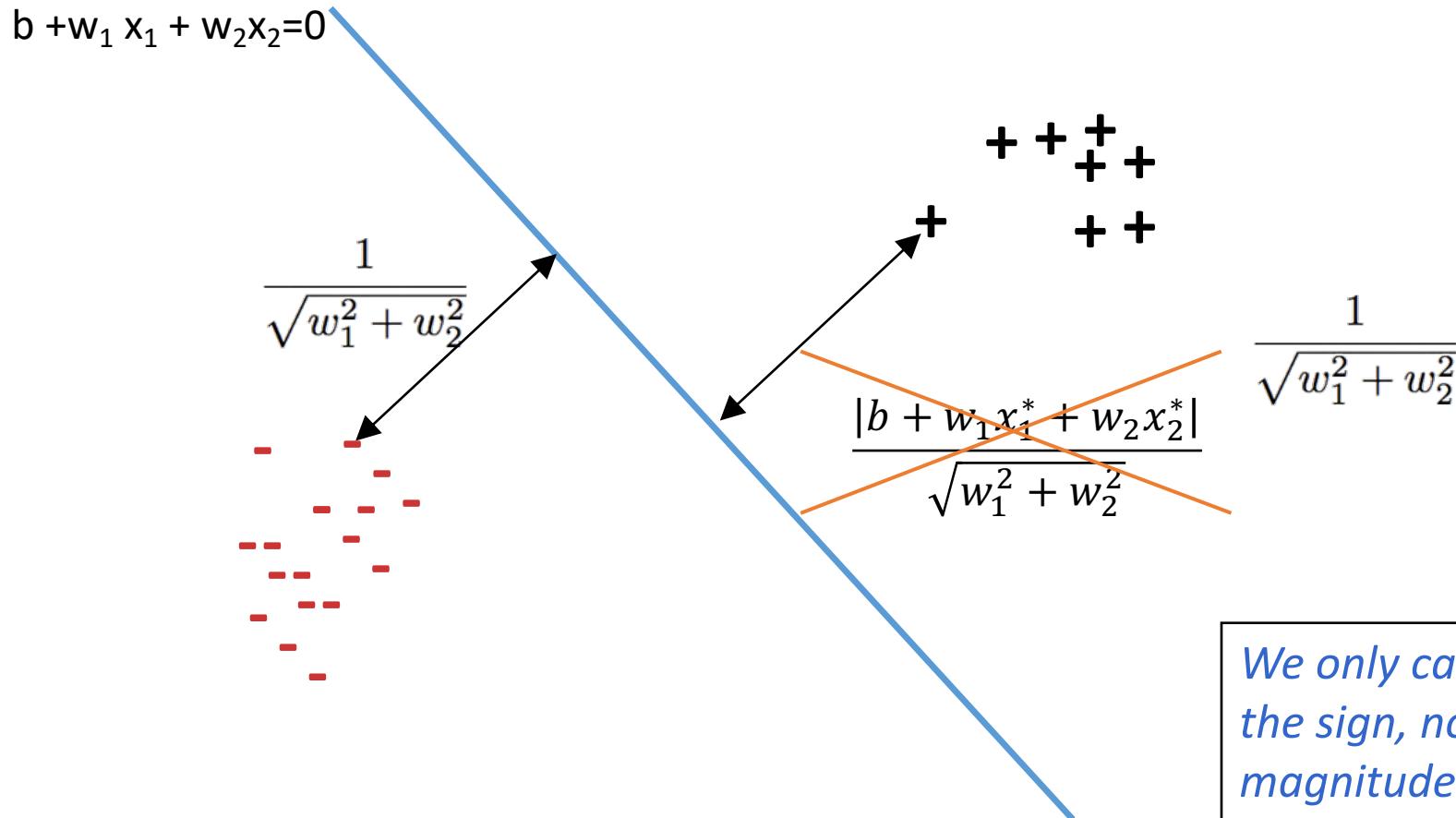
Recall: The geometry of a linear classifier

We have the freedom to scale up/down w and b so that we can make $\|w\|_2^2 = \|b\|_2^2 = 1$.



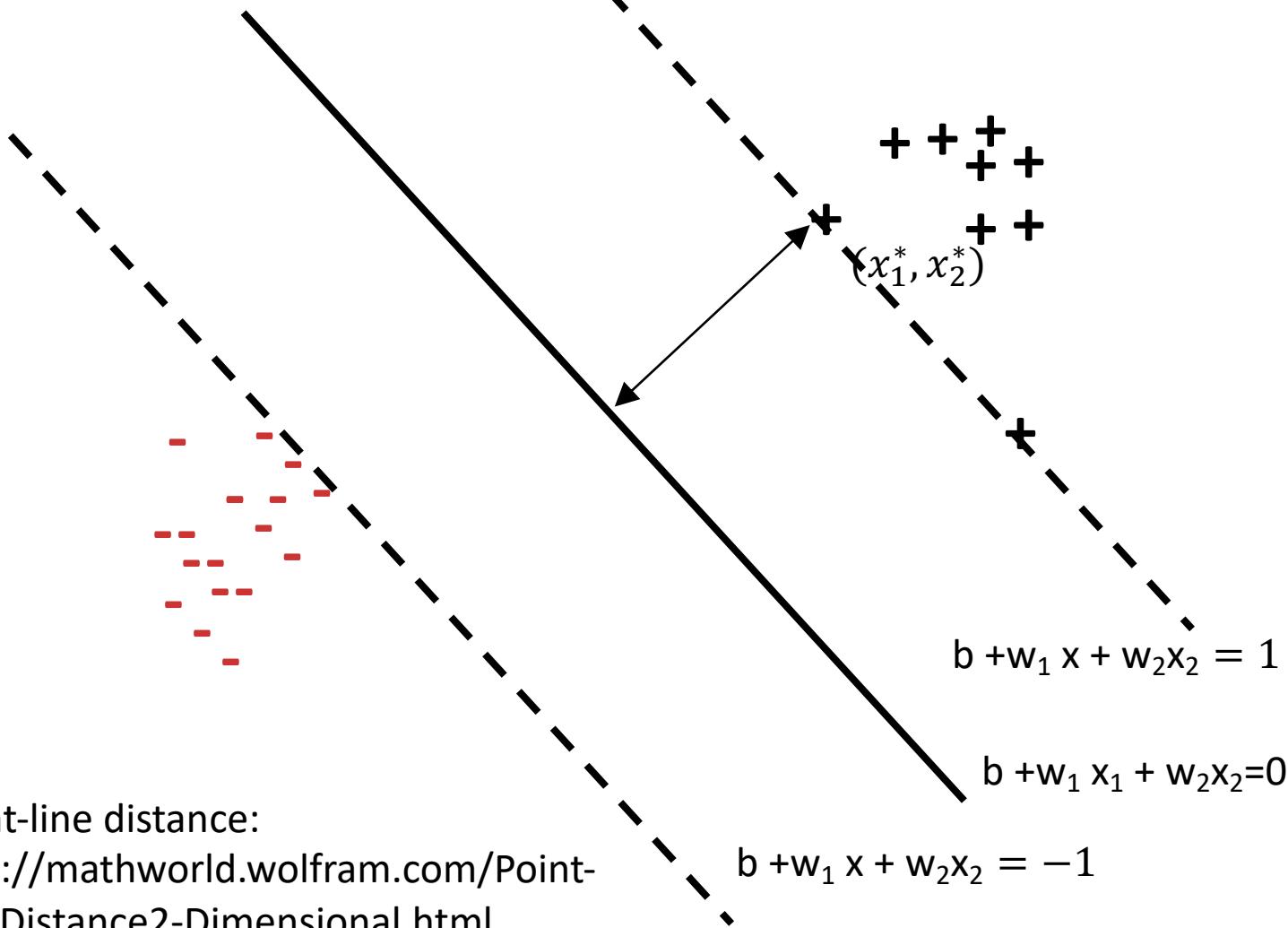
Recall: The geometry of a linear classifier

$$\text{Prediction} = \text{sgn}(b + w_1 x_1 + w_2 x_2)$$



Recall: The geometry of a linear classifier

We have the freedom to scale up/down w and b so that we can make $\|w\|_2^2 / \|w\|_1 = 1$.



Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

Maximizing margin

- ❖ Margin = distance of the closest point from the hyperplane

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

- ❖ We want $\max_{\mathbf{w}} \gamma$
- ❖ We only care about the sign of \mathbf{w} and b in the end and not the magnitude
 - ❖ Set the absolute score (functional margin) of the closest point to be 1 and allow \mathbf{w} to adjust itself

$\max_{\mathbf{w}} \gamma$ is equivalent to $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$ in this setting

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Max-margin classifiers

❖ Learning problem:

Mimimizing gives us $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i (\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

Max-margin classifiers

- ❖ Learning problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Mimimizing gives us $\max_{\mathbf{w}} \frac{1}{\|\mathbf{w}\|}$

This condition is true for every example, specifically, for the example closest to the separator

- ❖ This is called the “hard” Support Vector Machine

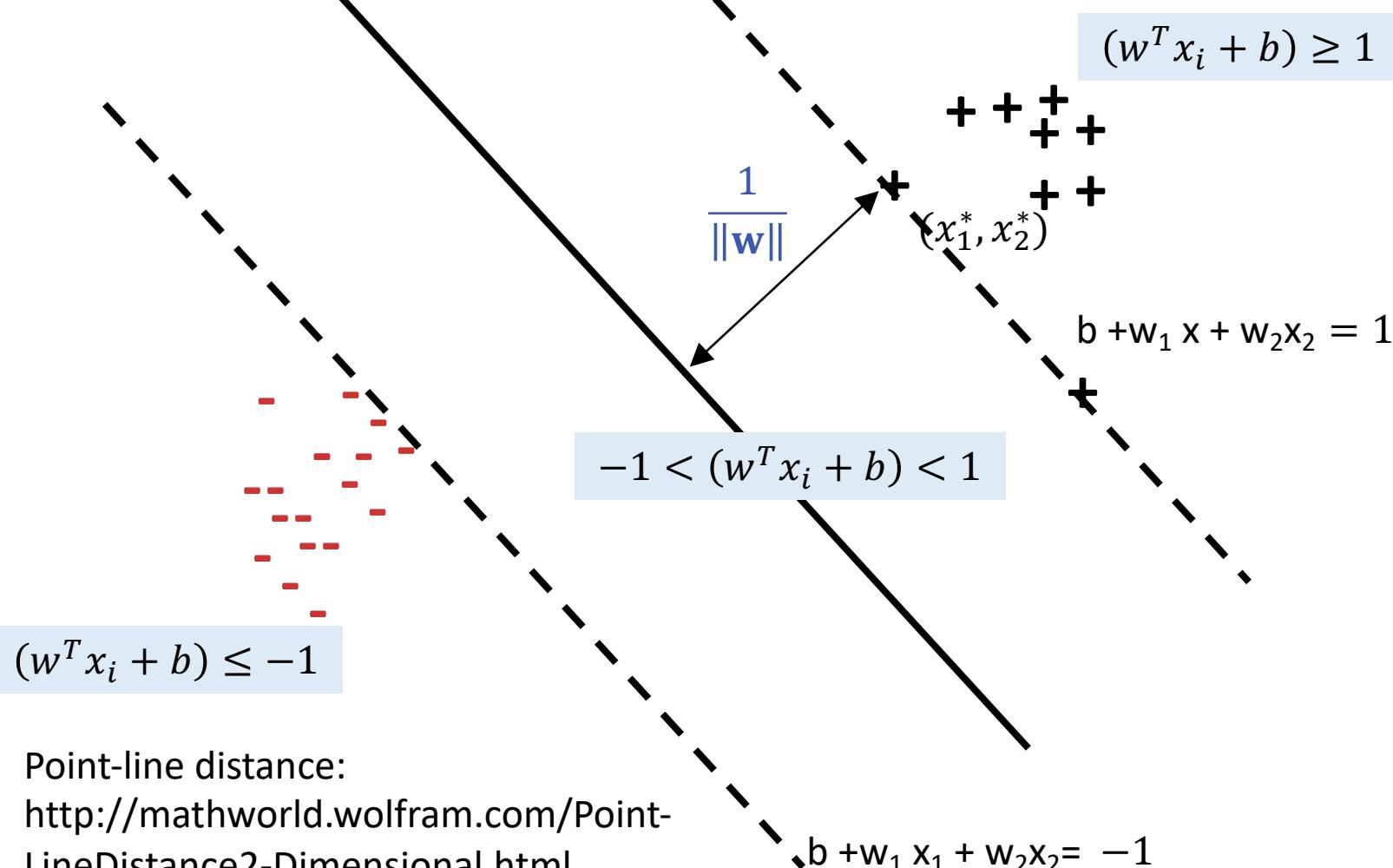
We will look at how to solve this optimization problem later

Hard SVM

$$\min_{w,b} \frac{1}{2} w^T w$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

$$b + w_1 x_1 + w_2 x_2 = 0$$



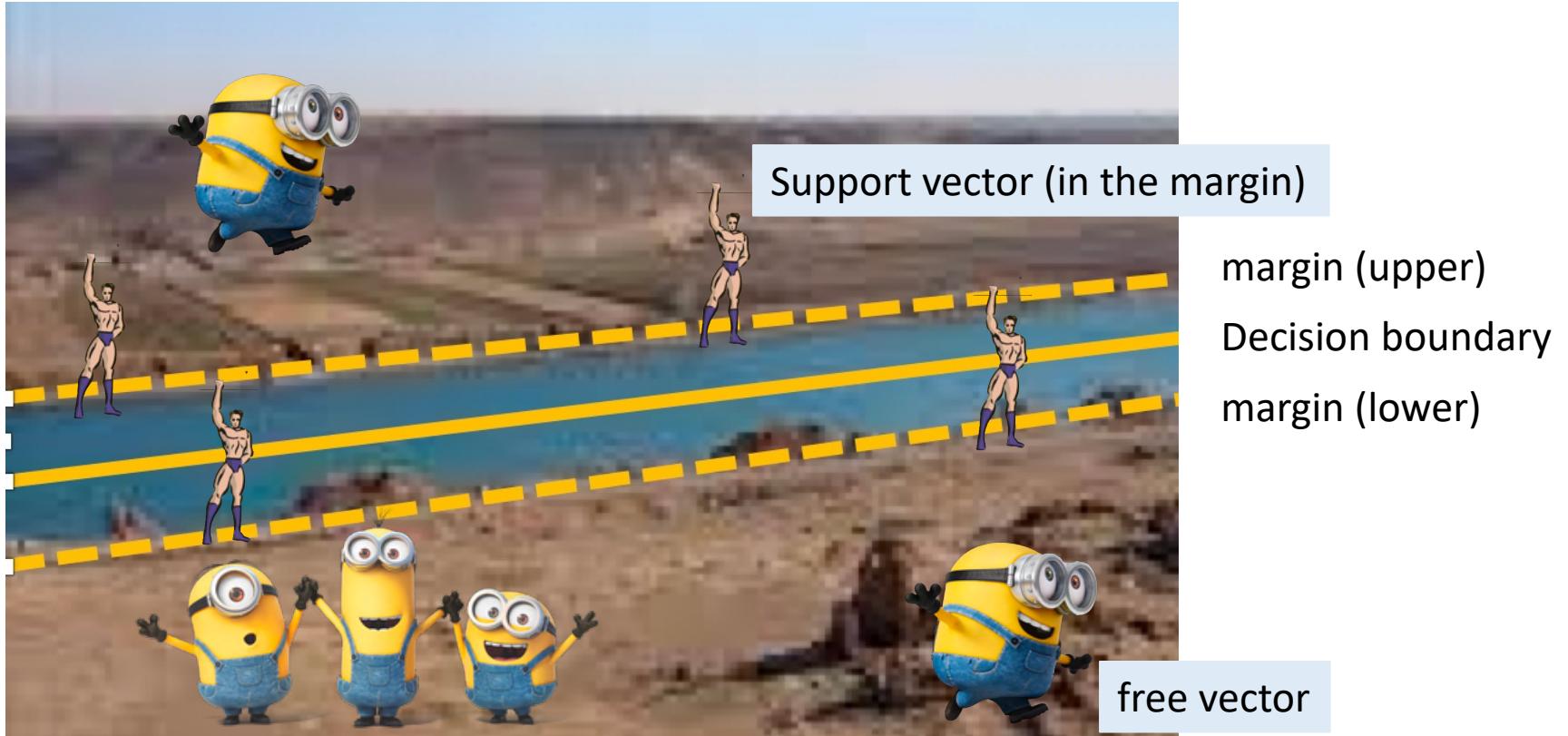
Point-line distance:

<http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

$$b + w_1 x_1 + w_2 x_2 = -1$$

Hard support vector machines?

No training error can be made. All support vectors are on the boundary



What if the data is not separable?

Hard SVM

$$\min_{w,b} \frac{1}{2} w^T w$$

Maximize margin

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Every example has an functional margin of at least 1

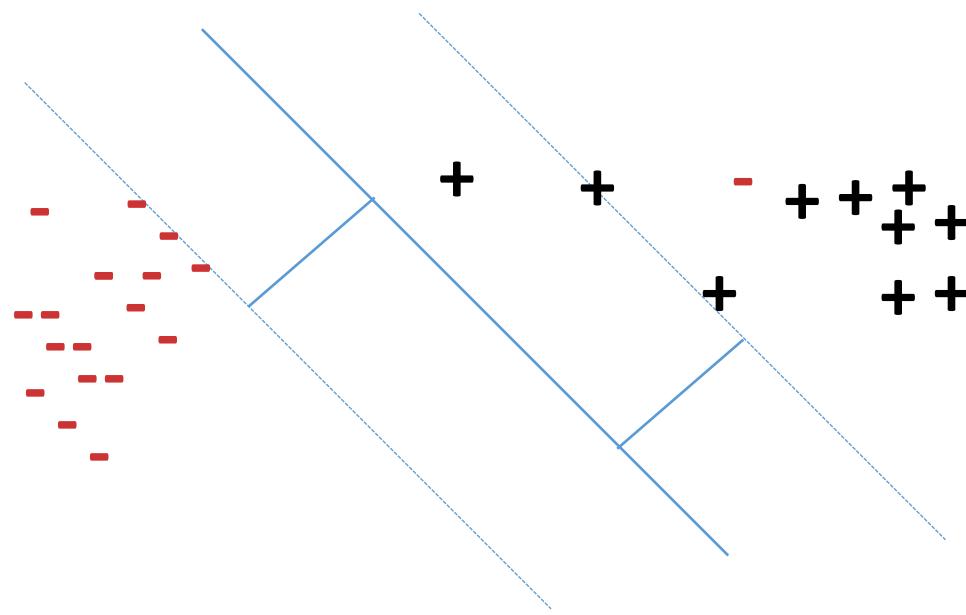
- ❖ This is a constrained optimization problem
- ❖ If the data is not separable, there is no **w** that will classify the data
- ❖ Infeasible problem, no solution!



If you made an mistake in your midterm, got 0 point!

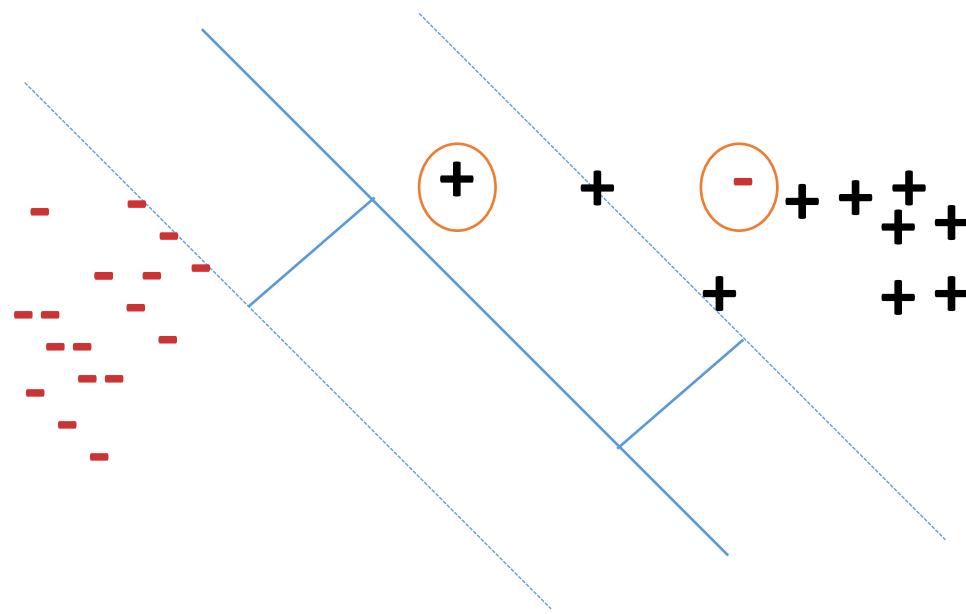
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin” or “make mistake”



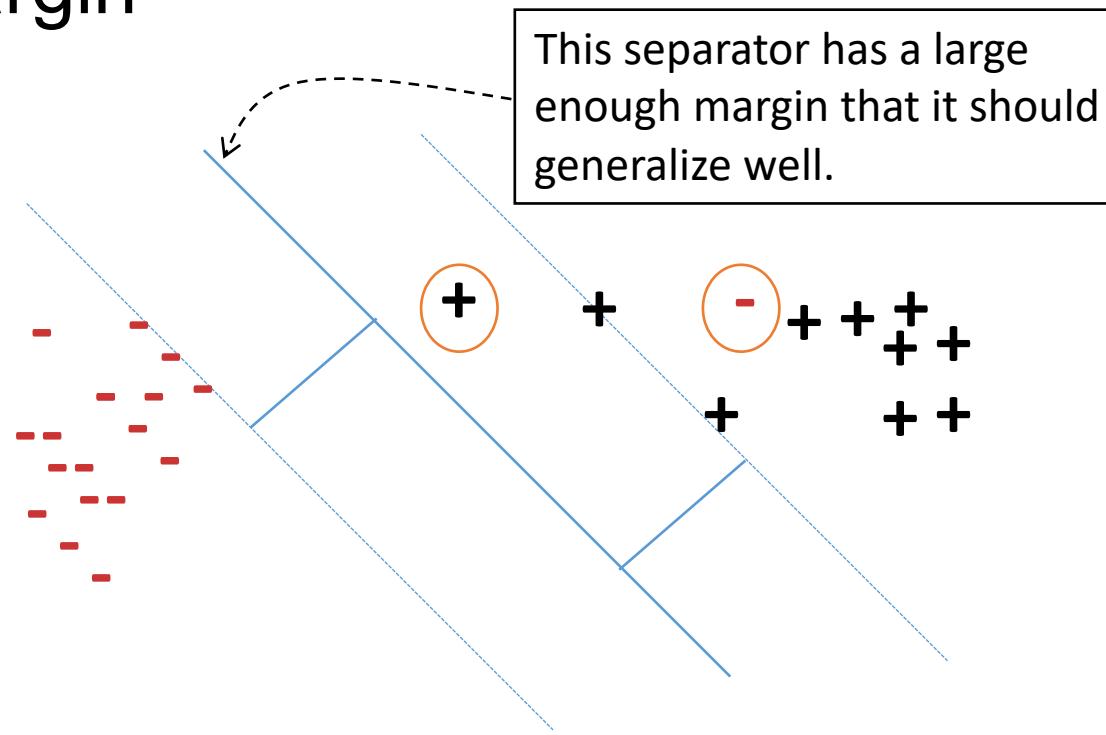
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”



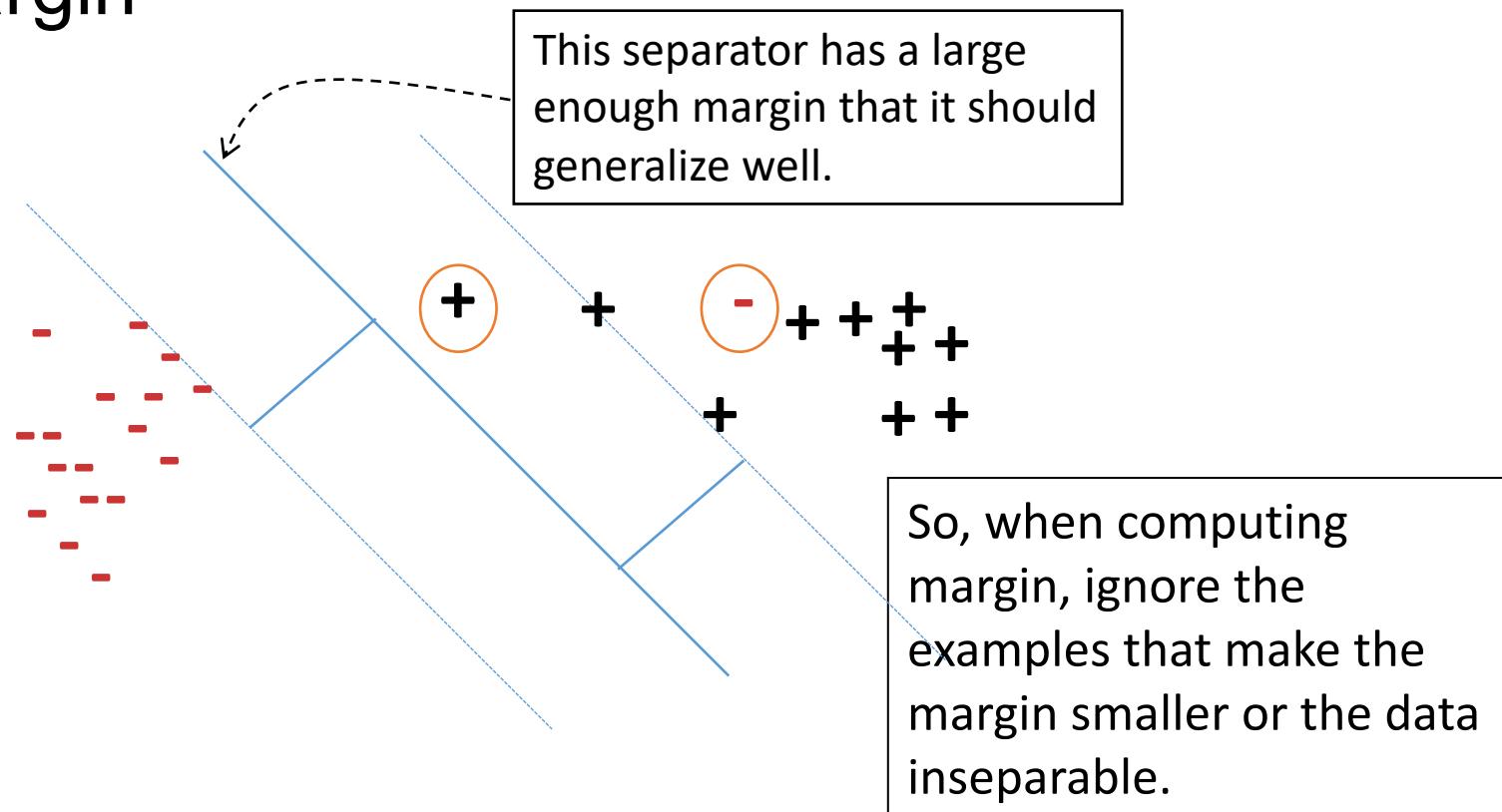
Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”



Dealing with non-separable data

Key idea: Allow some examples to “break into the margin”



Soft SVM

❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w$$

Maximize margin

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1$$

Every example has an
functional margin of at least 1

Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ Introduce one *slack variable* ξ_i per example

- ❖ And require $y_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ Introduce one *slack variable* ξ_i per example

- ❖ And require $y_i(w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

Intuition: The slack variable allows examples to “break” into the margin

If the slack value is zero, then the example is either on or outside the margin

Soft SVM

- ❖ Hard SVM:

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{Maximize margin}$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 \quad \text{Every example has an functional margin of at least 1}$$

- ❖ New optimization problem for learning

$$\min_{w,b, \xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$s.t. \quad \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

C is the hyper-parameter
Lecture 14: Support Vector
Machines

Soft SVM

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

Maximize margin

Tradeoff between the two terms

Minimize total slack (i.e. allow as few examples as possible to violate the margin)

$$\begin{aligned} s.t. \quad & \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

Equivalently, we can eliminate the slack variables to rewrite this:

Soft SVM

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

Maximize margin
Tradeoff between the two terms
Minimize total slack (i.e. allow as few examples as possible to violate the margin)

$$s.t. \quad \forall i, \quad y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

Equivalently, we can eliminate the slack variables to rewrite this:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$\begin{aligned} s.t. \quad & \forall i, \quad y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

Maximizing margin and minimizing loss

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

Maximize margin

Penalty for the prediction

Maximizing margin and minimizing loss

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

Maximize margin Penalty for the prediction

We can consider three cases

- ❖ Example is **correctly** classified and is outside the margin:
penalty = 0
- ❖ Example is **incorrectly** classified:
penalty = $1 - y_i(w^T x_i + b)$
- ❖ Example is **correctly** classified but **within the margin**:
penalty = $1 - y_i(w^T x_i + b)$

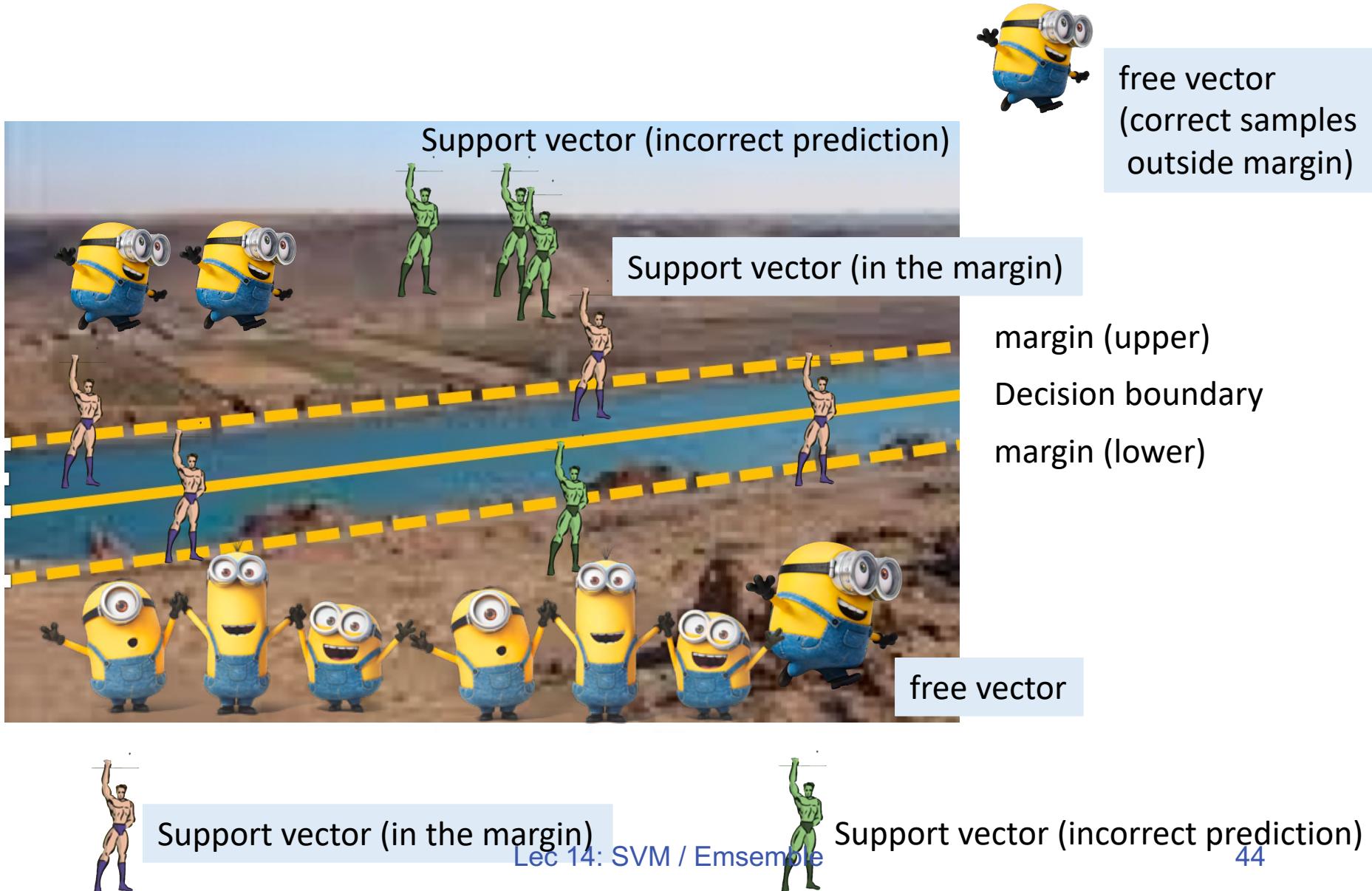
This is the **hinge loss** function

$$L_{\text{Hinge}}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y \mathbf{w}^T \mathbf{x})$$

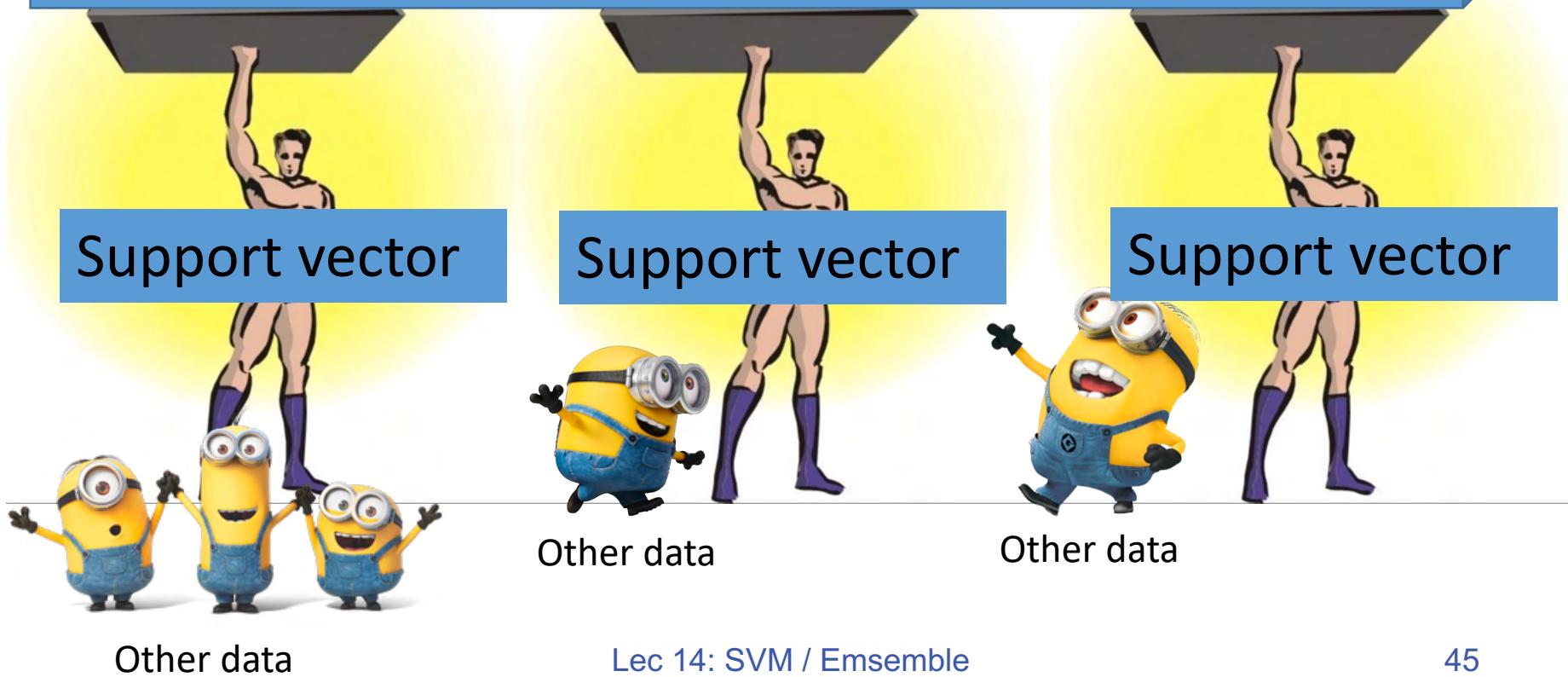
Why it called support vector machines?



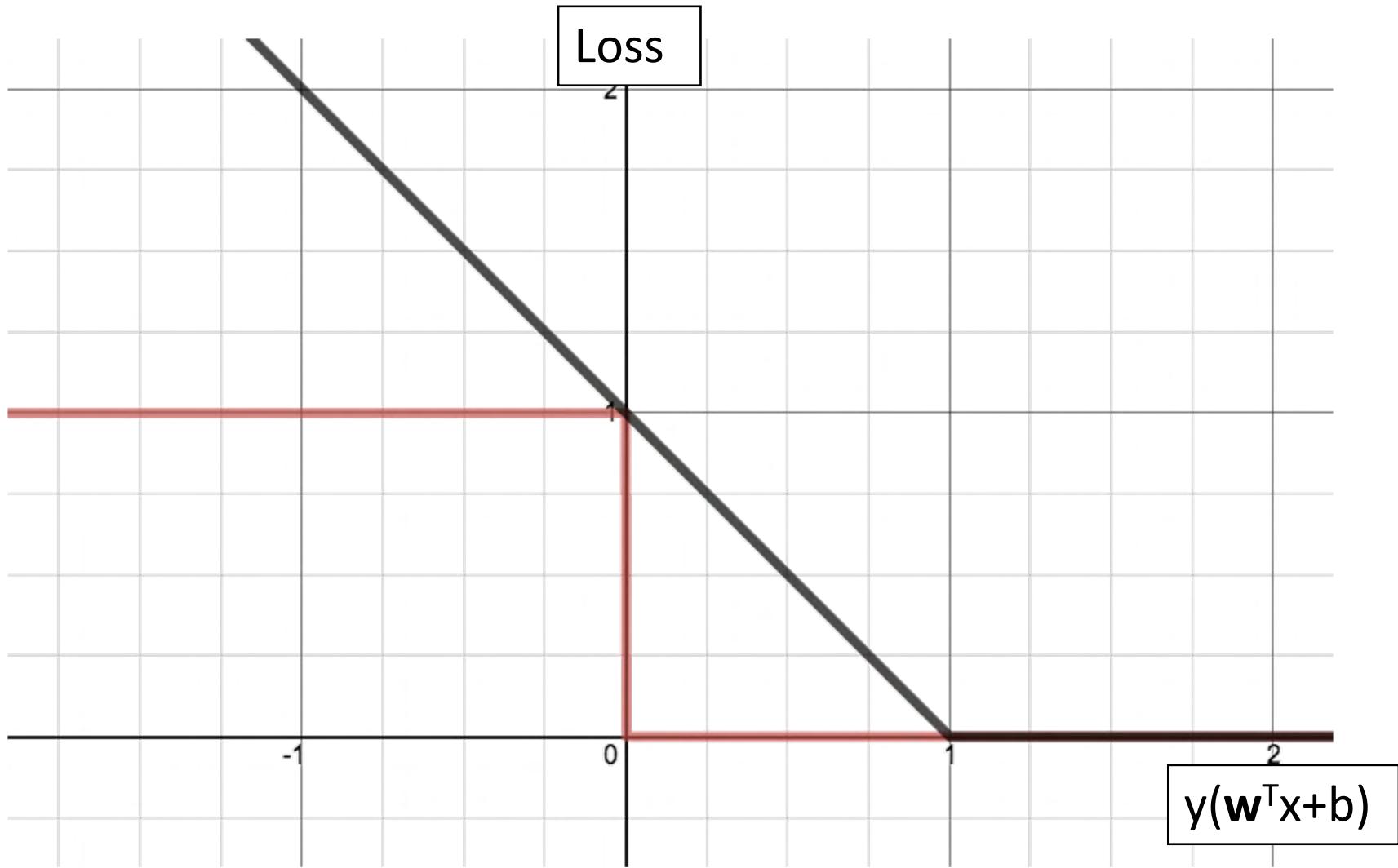
Why it called support vector machines?



Decision Boundary

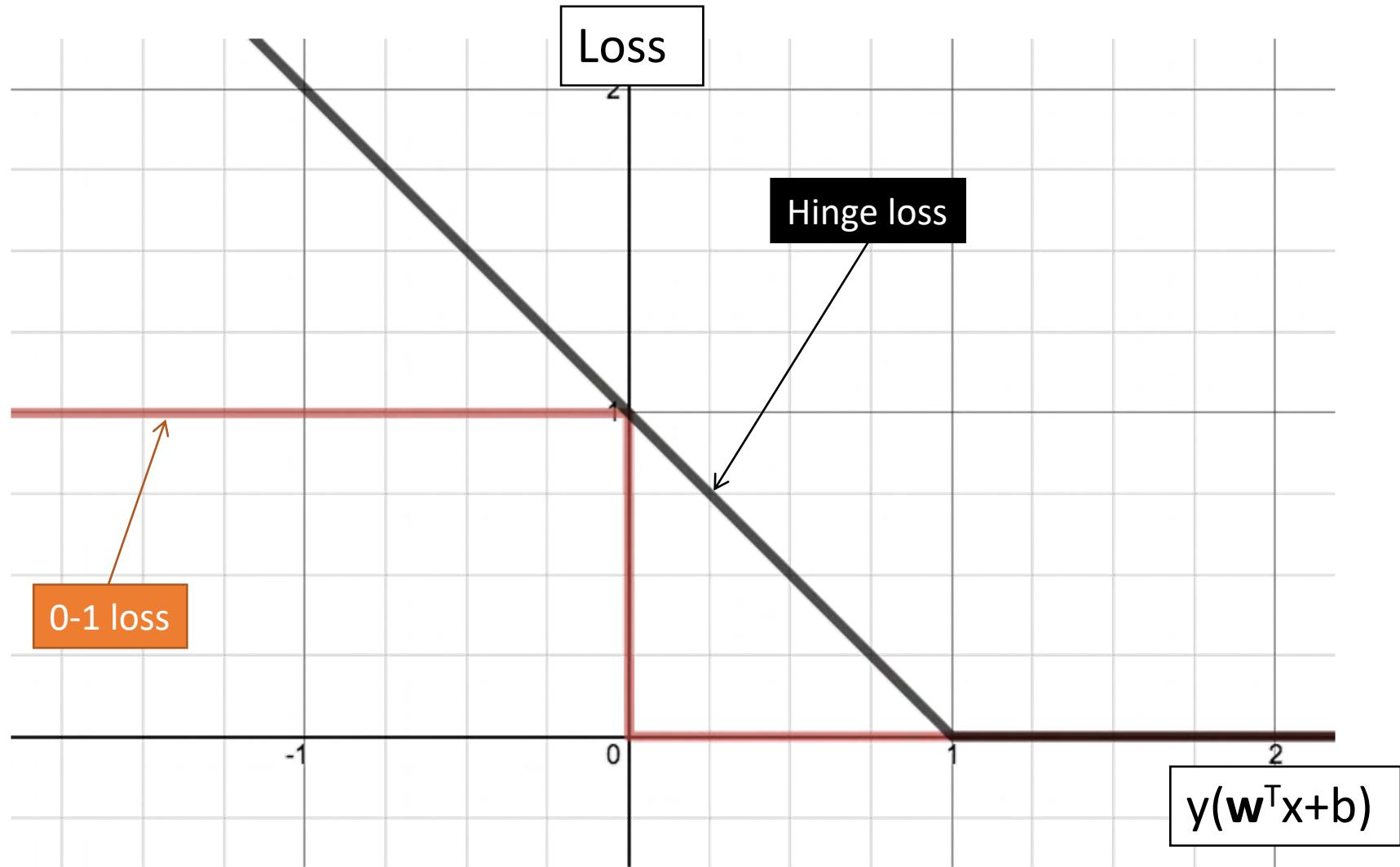


The Hinge Loss



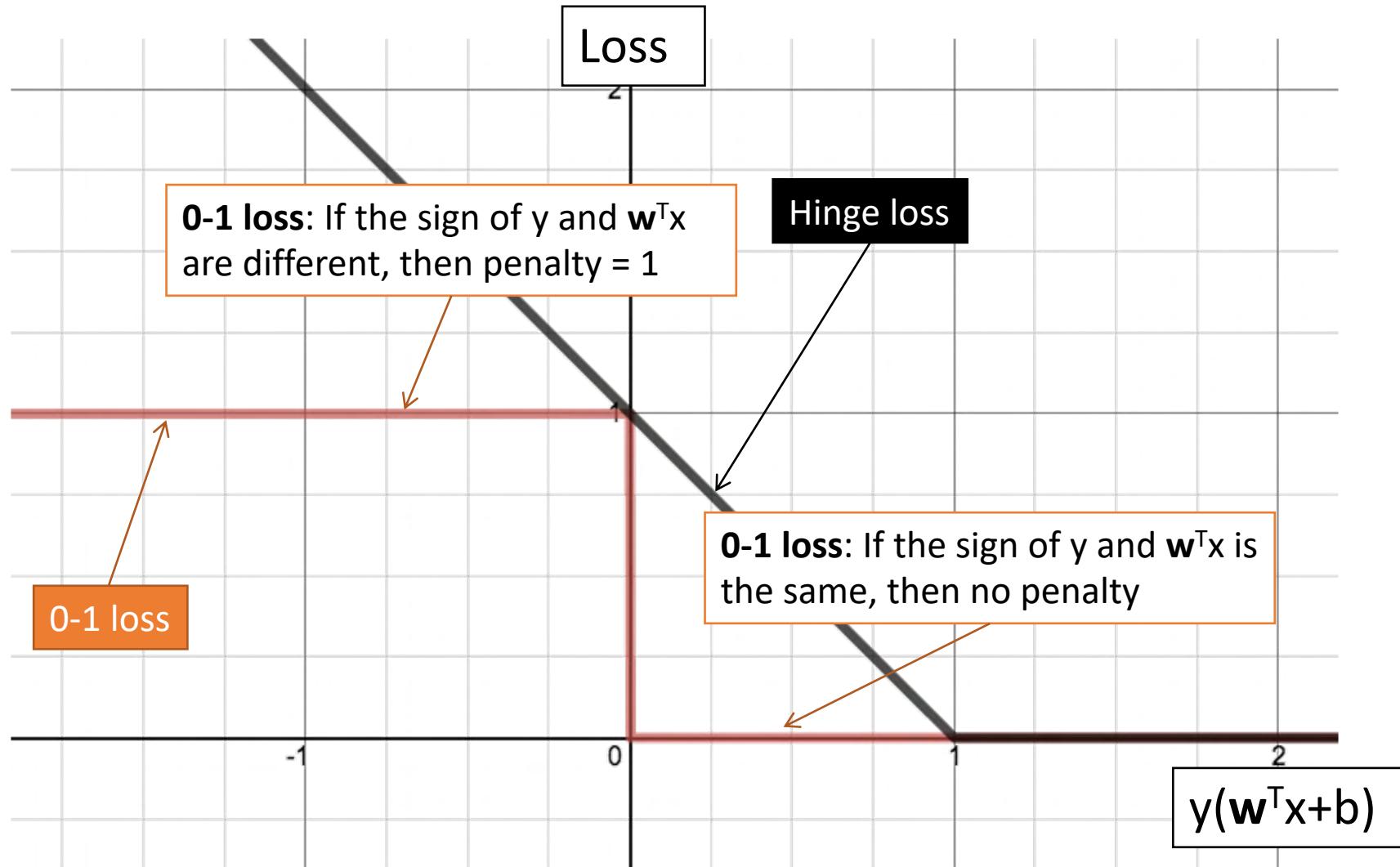
$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

The Hinge Loss



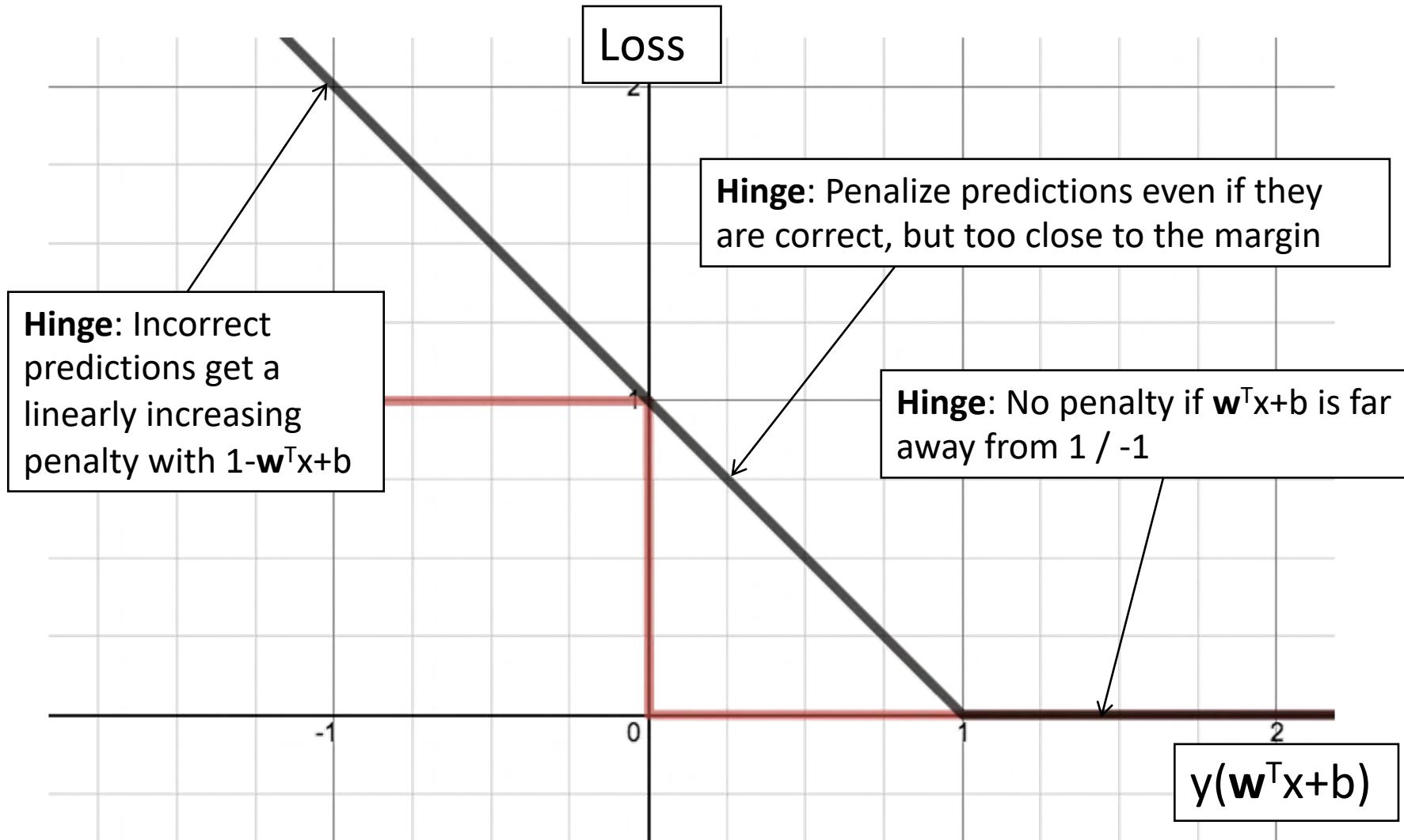
$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

The Hinge Loss



$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

The Hinge Loss



General learning principle

Risk minimization

Define the notion of “loss” over the training data as a function of a hypothesis

Learning = find the hypothesis that has lowest loss on the training data

General learning principle

Regularized risk minimization

Define a regularization function
that penalizes over-complex
hypothesis.

Define the notion of “loss”
over the training data as a
function of a hypothesis

Capacity control gives better
generalization

Learning =
find the hypothesis that has lowest
[Regularizer + loss on the training data]

SVM objective function

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:

- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

SVM objective function

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

Regularization term:

- Maximize the margin
- Imposes a preference over the hypothesis space and pushes for better generalization
- Can be replaced with other regularization terms which impose other preferences

Empirical Loss:

- Hinge loss
- Penalizes weight vectors that make mistakes
- Can be replaced with other loss functions which impose other preferences

A **hyper-parameter** that controls the tradeoff between a large margin and a small hinge-loss

This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

Solving the SVM optimization problem

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$$

This function is **convex** in **w**

Outline: Training SVM by optimization

1. Stochastic gradient descent
2. Sub-derivatives of the hinge loss
3. Stochastic sub-gradient descent for SVM
4. Comparison to perceptron

Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch 1 ... T :
3. For (x, y) in \mathcal{D} :
4. Update $w \leftarrow w - \eta \nabla_w f(x, y)$
5. Return w

$$\min \sum_{(x,y) \in D} f(x, y)$$

We will see more example later in this lecture

Hinge loss is not differentiable!

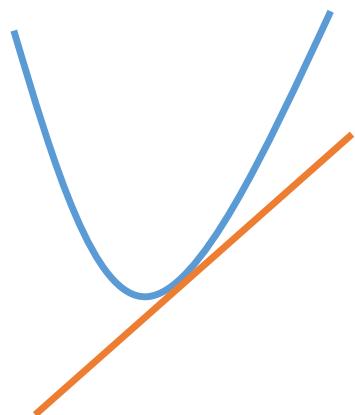
What is the derivative of the hinge loss with respect to w?

$$\frac{1}{2} w^T w + C \max(0, 1 - y_i(w^T x_i + b))$$

Detour: Sub-gradients

Generalization of gradients to non-differentiable functions

(Remember that every tangent lies below the function for convex functions)



Informally, a sub-tangent at a point is any line lies below the function at the point.

A sub-gradient is the slope of that line

Advanced topic [not in exam] Sub-gradients

Formally, g is a subgradient to f at x if

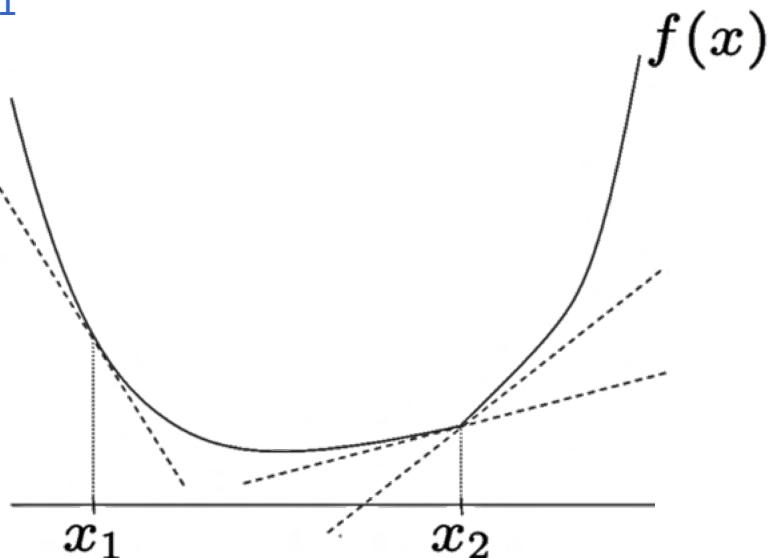
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

f is differentiable at x_1

Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

g_1 is a gradient at x_1



Sub-gradients

Formally, g is a subgradient to f at x if

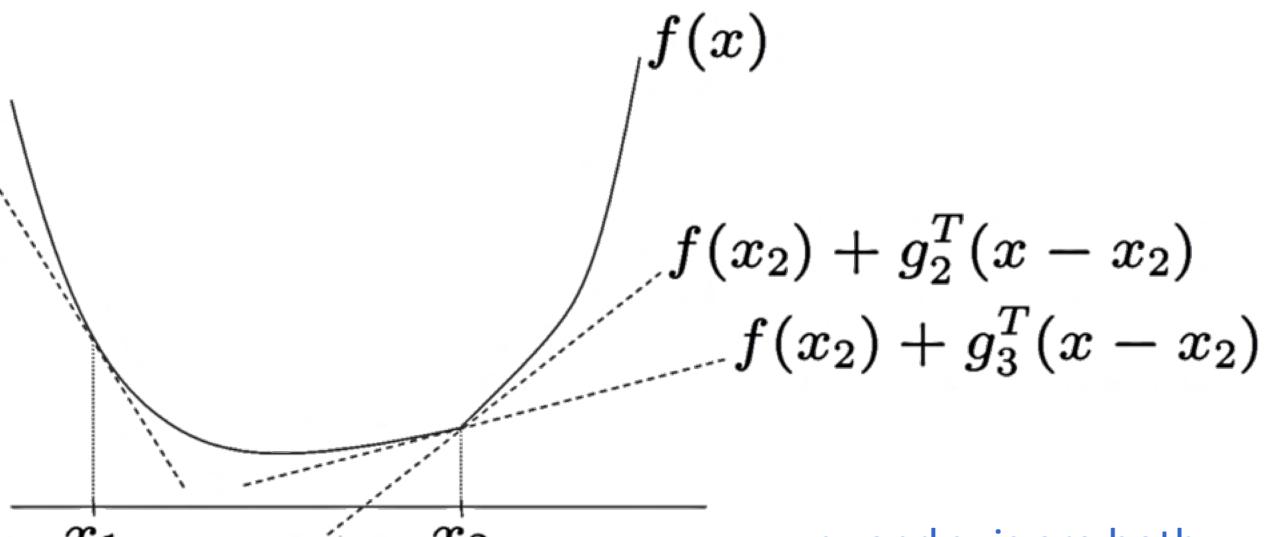
$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$

f is differentiable at x_1

Tangent at this point

$$f(x_1) + g_1^T(x - x_1)$$

g_1 is a gradient at x_1



g_2 and g_3 are both
subgradients at x_2

Sub-gradient of the SVM objective

$$J^t(w) = \frac{1}{2} w^T w + C \max(0, 1 - y_i(w^T x_i + b))$$

General strategy: First solve the max and compute the gradient for each case

$$\nabla J^t = \begin{cases} \mathbf{w} & \text{if } \max(0, 1 - y_i(w^T x_i + b)) = 0 \\ \mathbf{w} - Cy_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

Outline: Training SVM by optimization

1. Stochastic gradient descent
2. Sub-derivatives of the hinge loss
3. Stochastic sub-gradient descent for SVM
4. Comparison to perceptron

Stochastic gradient Descent

Given a training set $\mathcal{D} = \{(x, y)\}$

Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$

For epoch 1 ... T :

 For (x, y) in \mathcal{D} :

 if $y w^T x \geq 1$

$w \leftarrow w - \eta w$

 else

$w \leftarrow w - \eta(w - C y x)$

Return w

$$\nabla J^t = \begin{cases} \mathbf{w} & \text{if } \max(0, 1 - y_i(w^T x_i + b)) = 0 \\ \mathbf{w} - C y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

Recap: The Perceptron Algorithm

Given a training set $\mathcal{D} = \{(x, y)\}$

1. Initialize $w \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For (x, y) in \mathcal{D} :
3. if $y(w^\top x + b) \leq 0$
4. $w \leftarrow w + yx$
- 5.
6. Return w

SVM:

If $y(w^\top x + b) < 1$
 $w \leftarrow w - \eta(w - Cyx)$
else: $w \leftarrow w - \eta w$

Prediction: $y^{\text{test}} \leftarrow \text{sgn}(w^\top x^{\text{test}})$

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Perceptron vs. SVM

- ❖ Perceptron: Stochastic sub-gradient descent for a different loss
 - ❖ No regularization though

$$L_{\text{Perceptron}}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T \mathbf{x})$$

- ❖ SVM optimizes the hinge loss
 - ❖ With regularization

$$L_{\text{Hinge}}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

This lecture: Support vector machines

- ❖ Training by maximizing margin
- ❖ The SVM objective
- ❖ Solving the SVM optimization problem
- ❖ Support vectors, duals and kernels

SVM: Primal and dual

The SVM objective

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \sum_i \xi_i$$

$$\begin{aligned} s.t. \quad & \forall i, \quad y_i (w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

This is called the *primal form* of the objective

This can be converted to its *dual form*, which will let us prove a very useful property

Support vector machines

Let \mathbf{w} be the minimizer of the SVM problem for some dataset with m examples: $\{(\mathbf{x}_i, y_i)\}$

Then, for $i = 1 \dots m$, there exist $\alpha_i \geq 0$ such that the optimum \mathbf{w} can be written as

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

Support vector machines

Let \mathbf{w} be the minimizer of the SVM problem for some dataset with m examples: $\{(\mathbf{x}_i, y_i)\}$

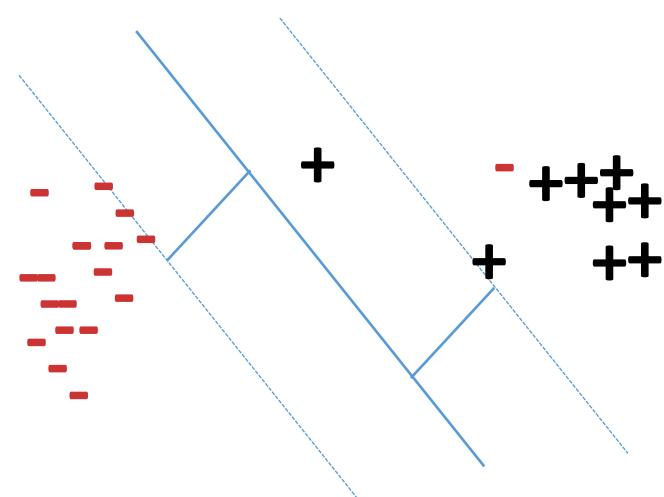
Then, for $i = 1 \dots m$, there exist $\alpha_i \geq 0$ such that the optimum \mathbf{w} can be written as

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$\alpha_i = 0 \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

All points outside the margin



Support vector machines

Let \mathbf{w} be the minimizer of the SVM problem for some dataset with m examples: $\{(\mathbf{x}_i, y_i)\}$

Then, for $i = 1 \dots m$, there exist $\alpha_i \geq 0$ such that the optimum \mathbf{w} can be written as

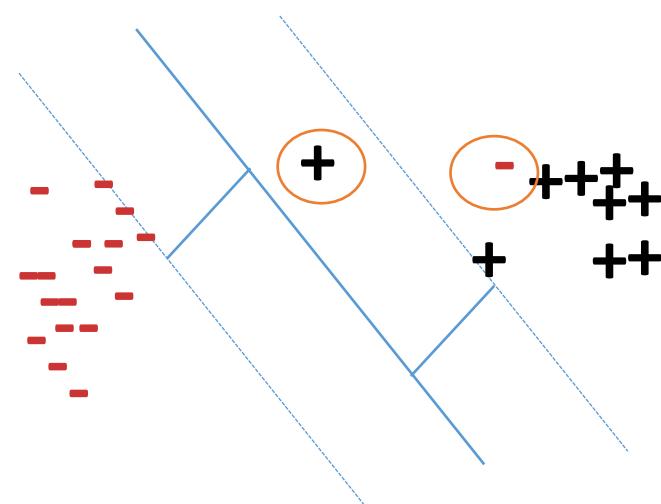
$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$\alpha_i = 0 \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

$$\alpha_i = C \quad \Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \leq 1$$

All points on the wrong side of the margin



Support vector machines

Let \mathbf{w} be the minimizer of the SVM problem for some dataset with m examples: $\{(\mathbf{x}_i, y_i)\}$

Then, for $i = 1 \dots m$, there exist $\alpha_i \geq 0$ such that the optimum \mathbf{w} can be written as

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

Furthermore,

$$\alpha_i = 0$$

$$\Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$$

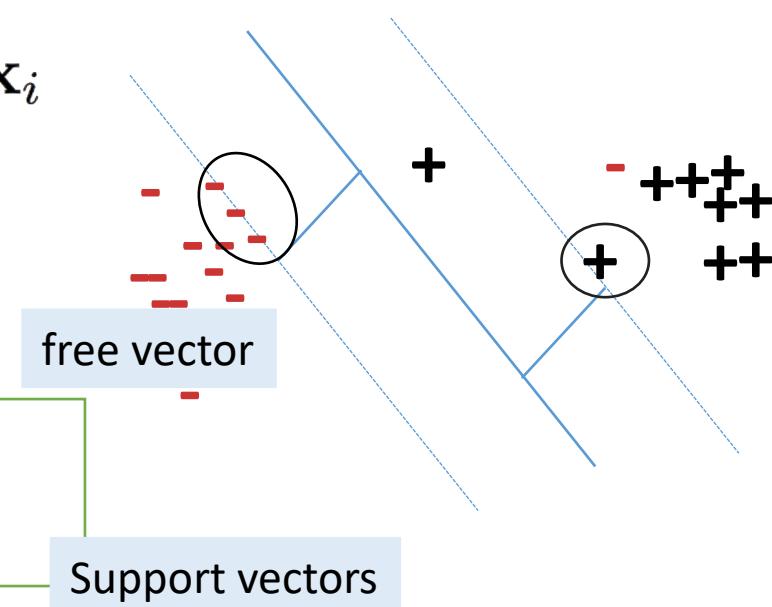
$$\alpha_i = C$$

$$\Rightarrow y_i \mathbf{w}^T \mathbf{x}_i \leq 1$$

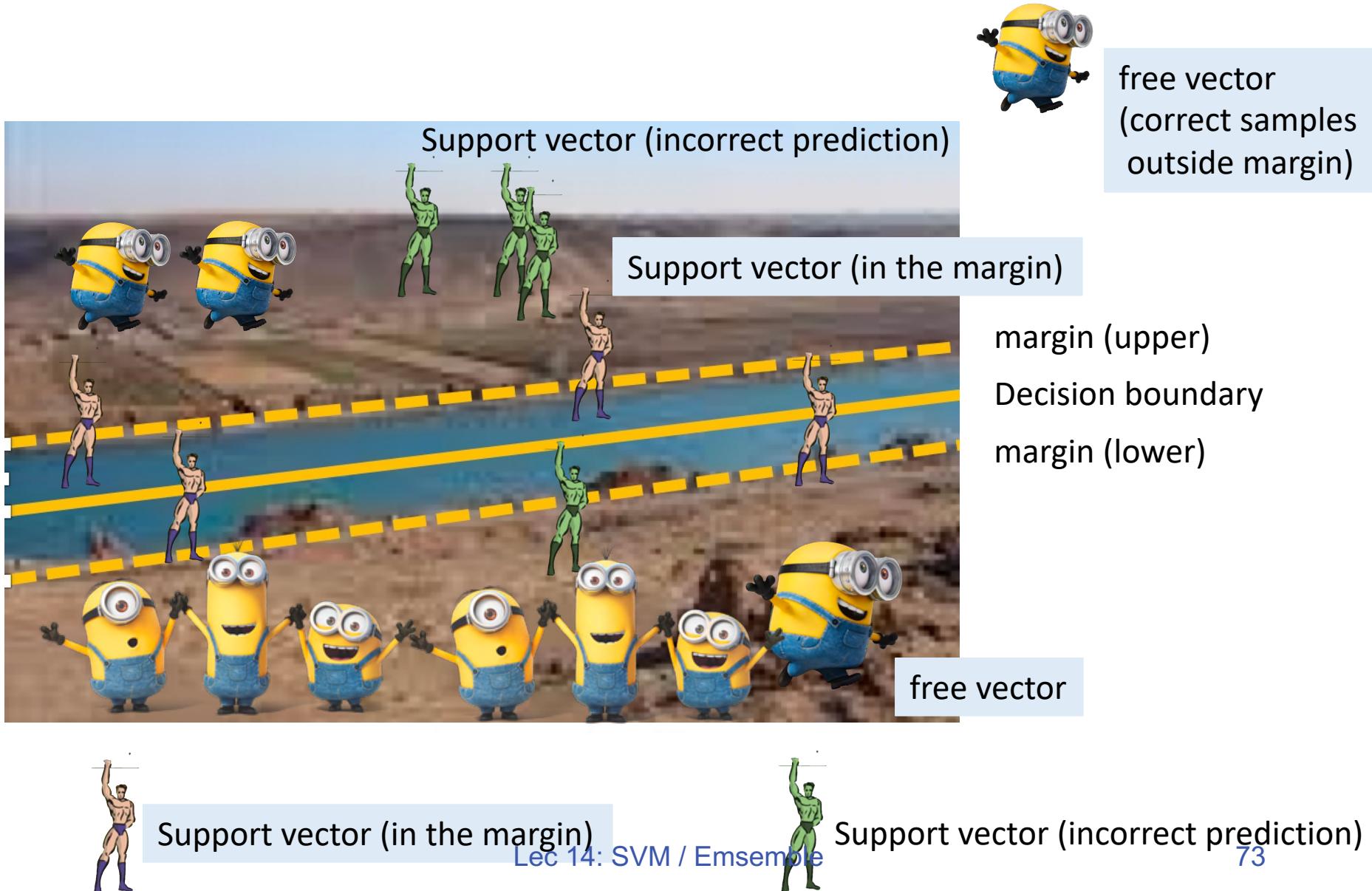
$$0 \leq \alpha_i \leq C$$

$$\Rightarrow y_i \mathbf{w}^T \mathbf{x}_i = 1$$

All points on the margin



Why it called support vector machines?



Support vectors

The weight vector is completely defined by training examples whose α_i s are not zero

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$$

These examples are called the *support vectors*

Lecture 15: Debugging ML Models

Fall 2021

Kai-Wei Chang

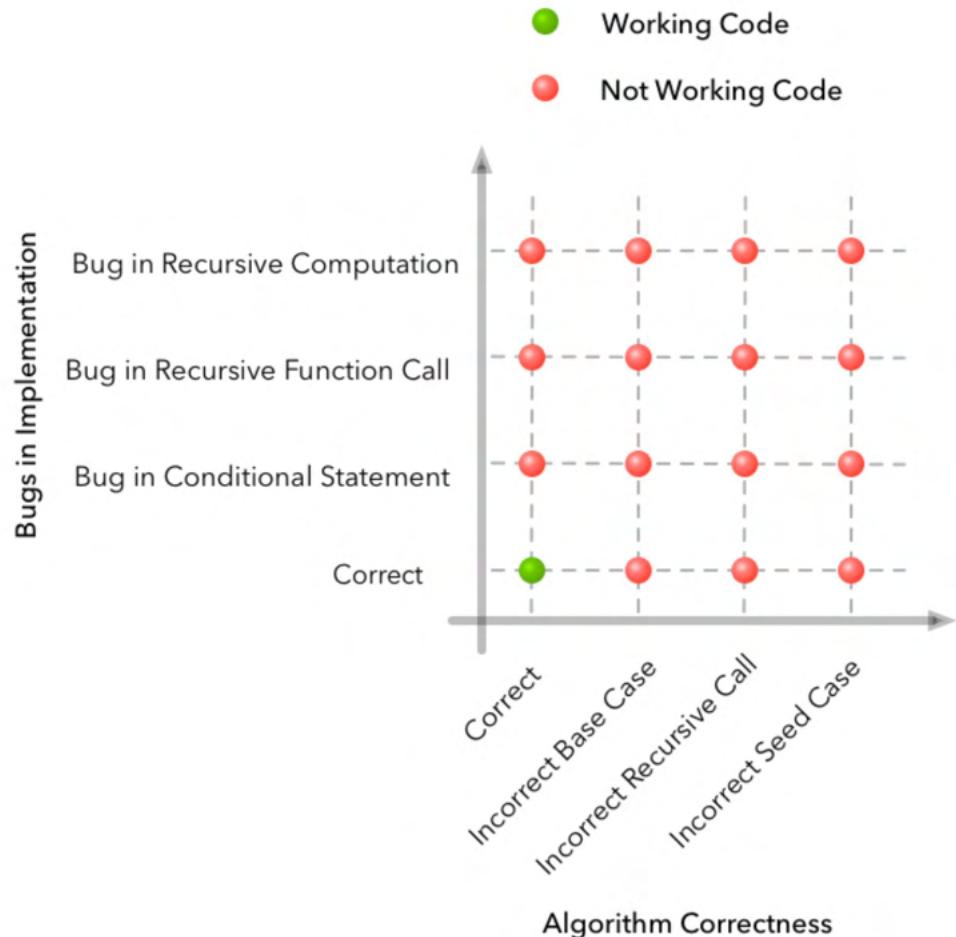
CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Challenges: Debugging

Debugging a program

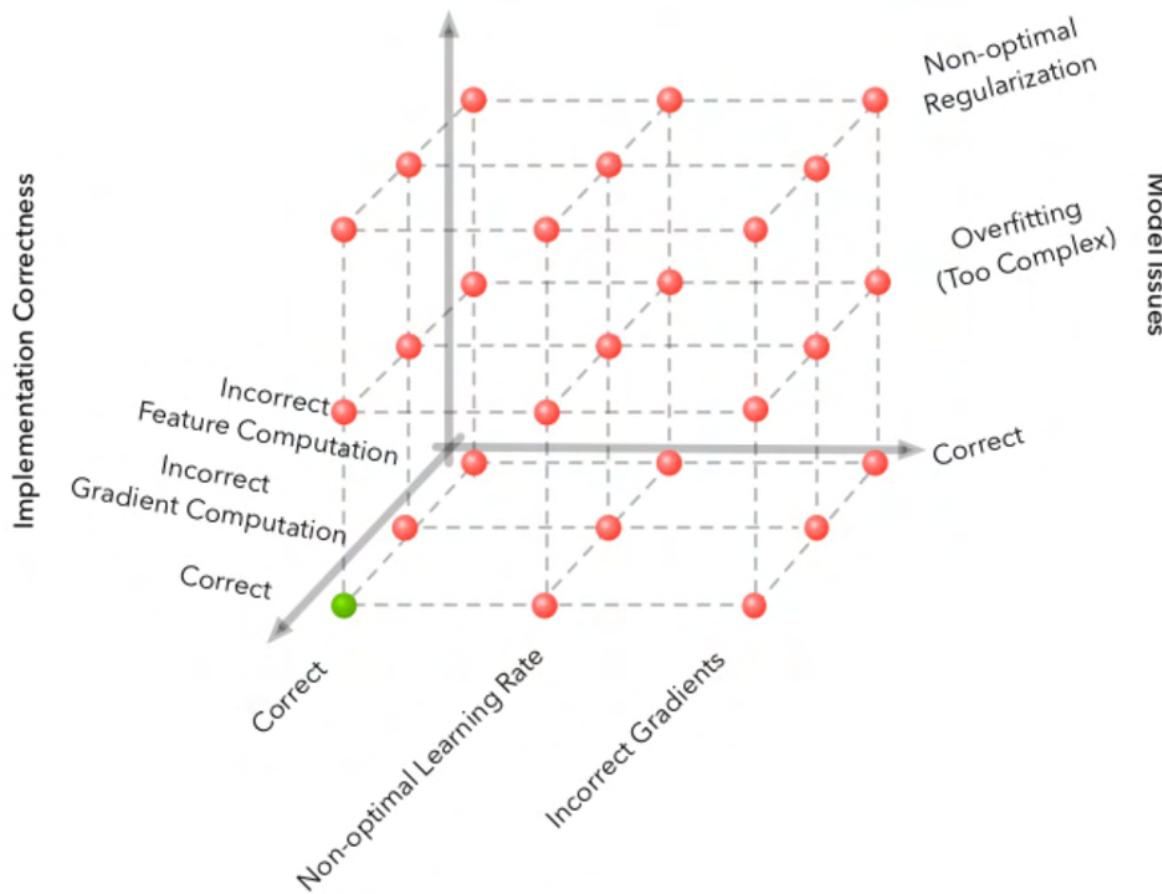


Credit: S. Zayd Enam

<http://ai.stanford.edu/~zayd/why-is-machine-learning-hard.html>

Challenges: Debugging

Debugging a ML model



Credit: S. Zayd Enam

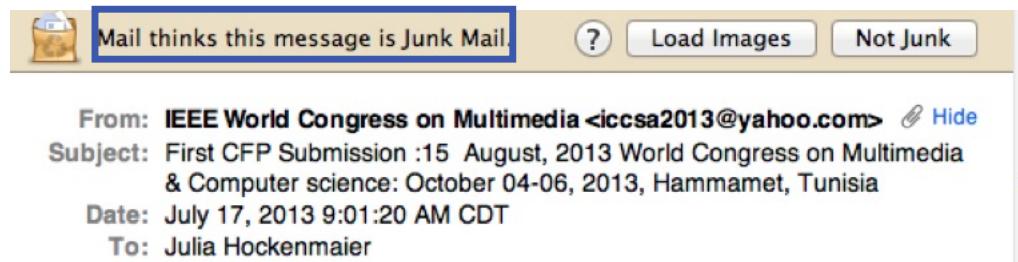
Challenges: Debugging

Debugging a ML model

Data issues



Motivative Example: Spam Detection



- ❖ A binary classification task:
Assign one of two labels (i.e. Spam/Ham) to the input (here, an email message)
- ❖ You collect 2,000 examples and train a logistic regression model
- ❖ The model performs terribly when deploying on the real data

Outline

- ❖ Diagnostics of your learning algorithm
- ❖ Error analysis
- ❖ Model explanation

*What could possibly
go wrong?*

What can go wrong?

- ❖ Data set is too small
- ❖ Data is noisy / Data is imbalanced
- ❖ Bad choice of model
- ❖ Bad choice of hyper-parameters
- ❖ The model has not converged
 - ❖ The optimization method is incorrectly implemented
- ❖ Evaluation metrics is not right (e.g., accuracy is not a good measurement)
- ❖ Test data has a different distribution
- ❖ ...

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

How to make the diagnosis
systematic?

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

First, diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)
2. Under-fitting (high bias)
3. Your learning does not converge
4. Are you measuring the right thing?

Bias and variance

Every learning algorithm requires assumptions about the hypothesis space.

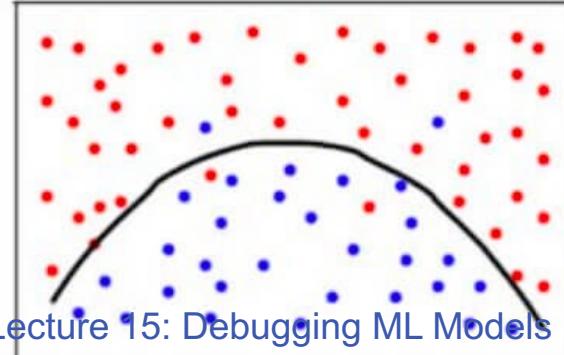
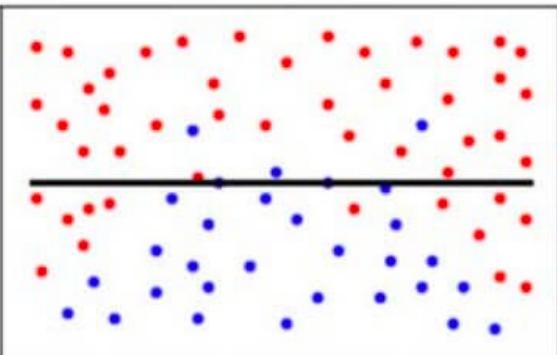
Eg: “My hypothesis space is

- ❖ ...linear”
- ❖ ...decision trees with 5 nodes”
- ❖ ...deep neural network with 12 layers”

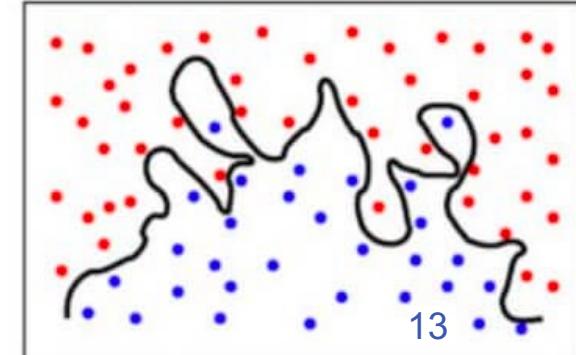
Underfitting



Overfitting



Lecture 15: Debugging ML Models



Detecting over or under fitting

Over-fitting: The training accuracy is much higher than the test accuracy

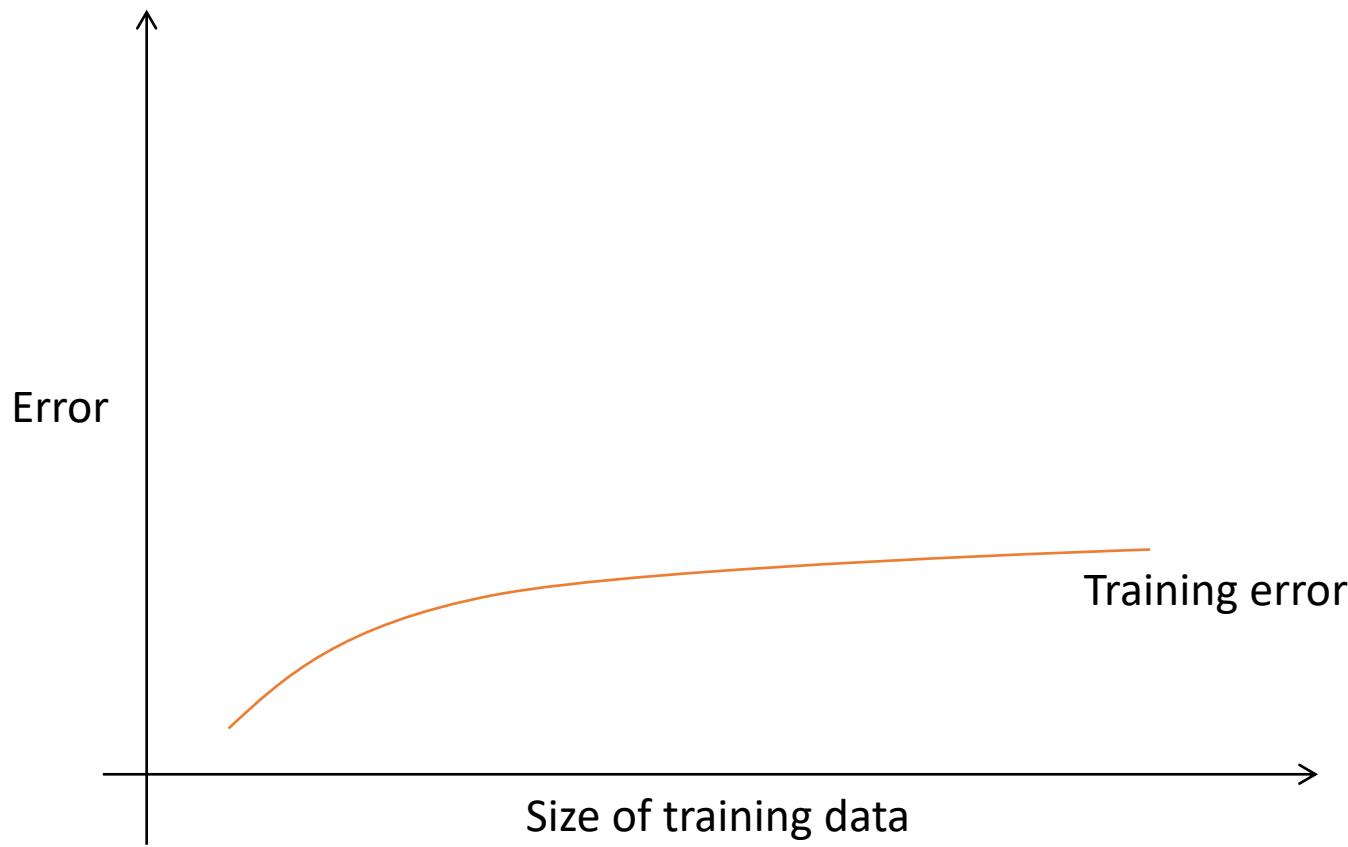
- ❖ The model explains the training set very well, but poor generalization

Under-fitting: Both accuracies are unacceptably low

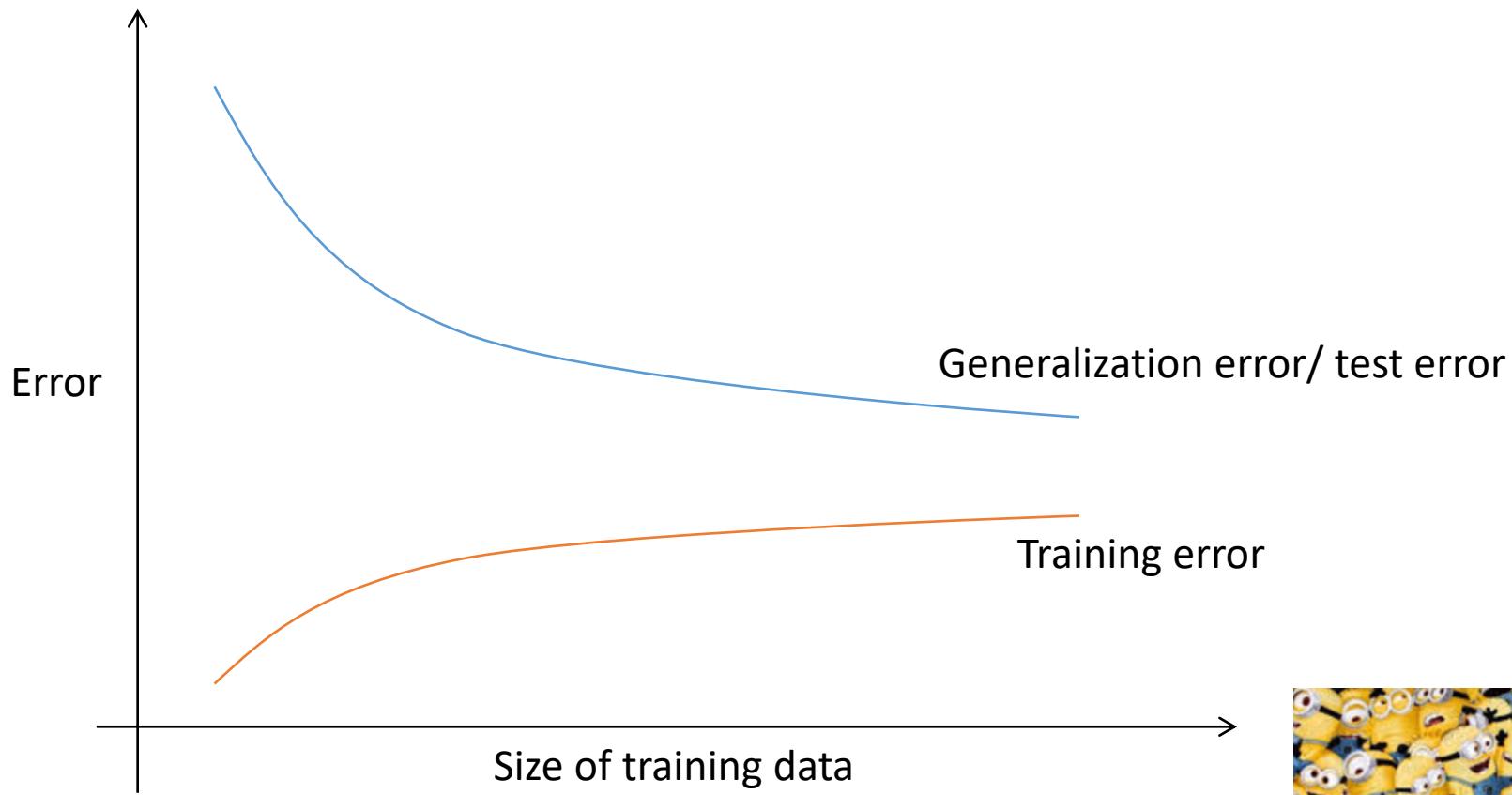
- ❖ The model can not represent the concept well enough

What accuracy is considered as high/low?

Detecting high variance using learning curves



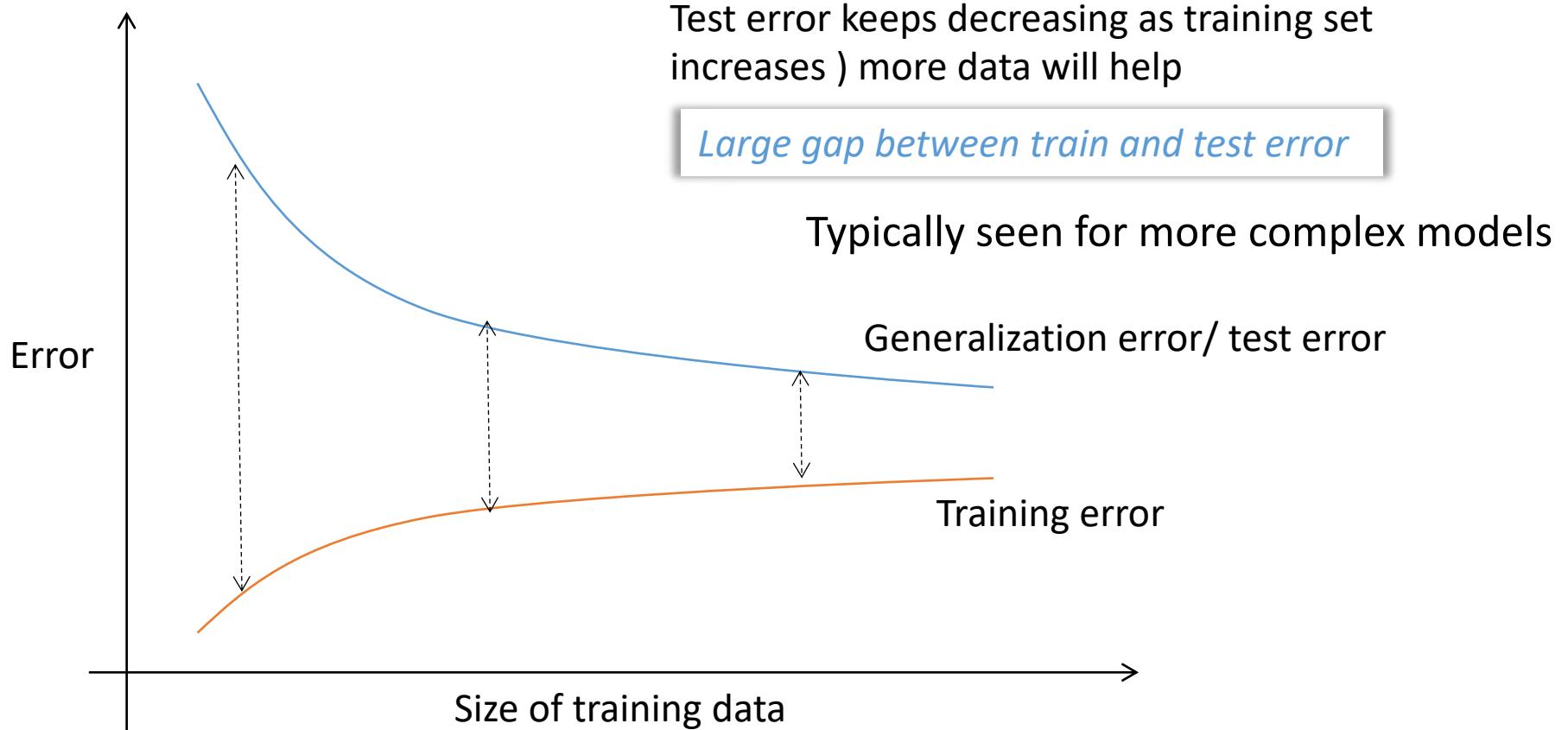
Detecting high variance using learning curves



Do we need more data?

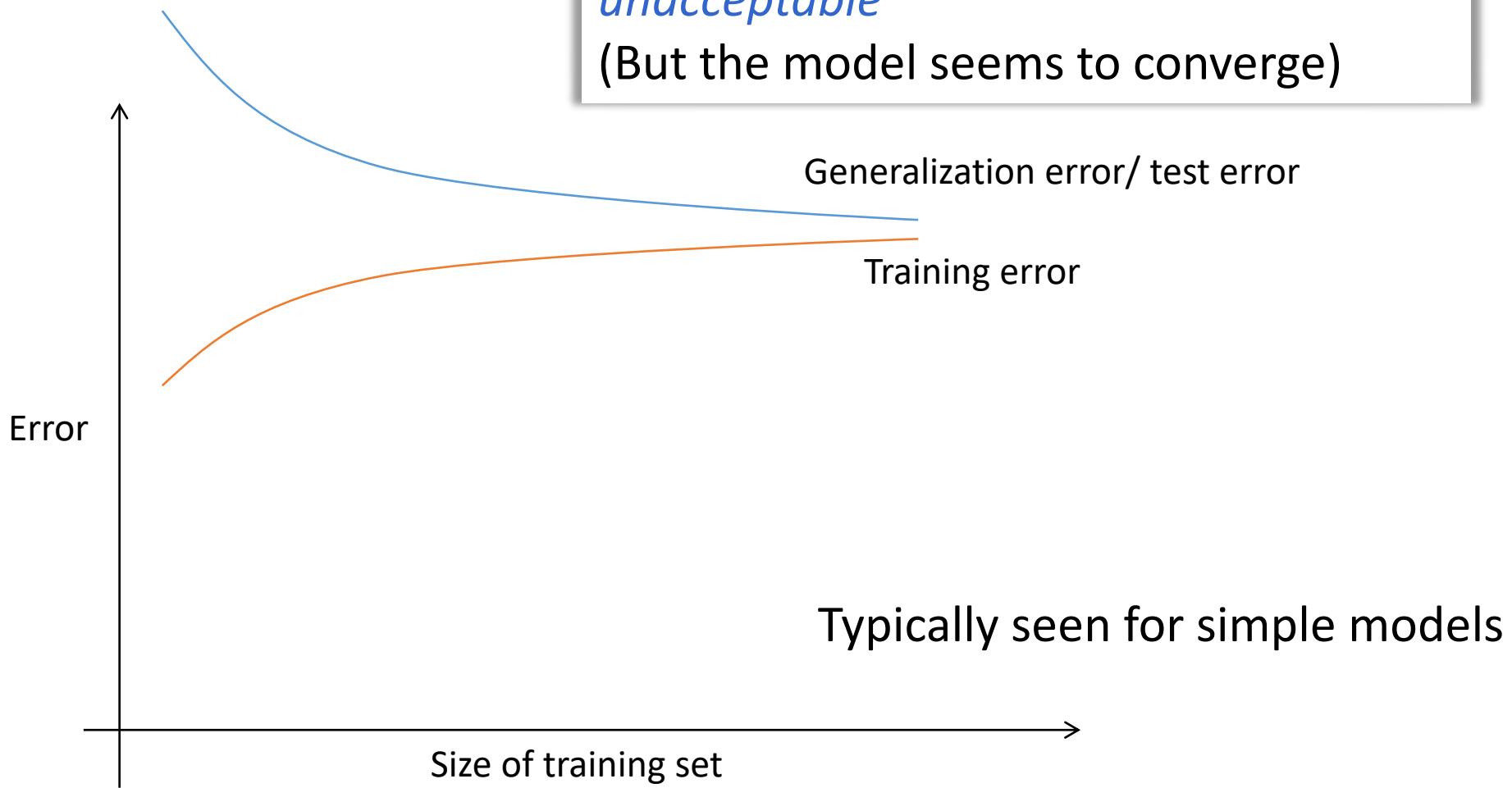


Detecting high variance using learning curves



Detecting high bias using learning curves

Both train and test error are unacceptable
(But the model seems to converge)



Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Different ways to improve your model

More training data

Helps with over-fitting

Features

1. Use more features Helps with under-fitting
2. Use fewer features Helps with over-fitting
3. Use other features Helps with over-fitting or under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm Could help with over-fitting / and under-fitting
3. Use a different classifier
4. Play with regularization

Hyper-parameters

- ❖ Decision trees of a fixed depth
 - ❖ Increasing depth decreases bias, increases variance
- ❖ SVMs
 - ❖ Stronger regularization increases bias, decreases variance
- ❖ K nearest neighbors
 - ❖ Increasing k generally increases bias, reduces variance
- ❖ Neural Network
 - ❖ Early stopping, dropout

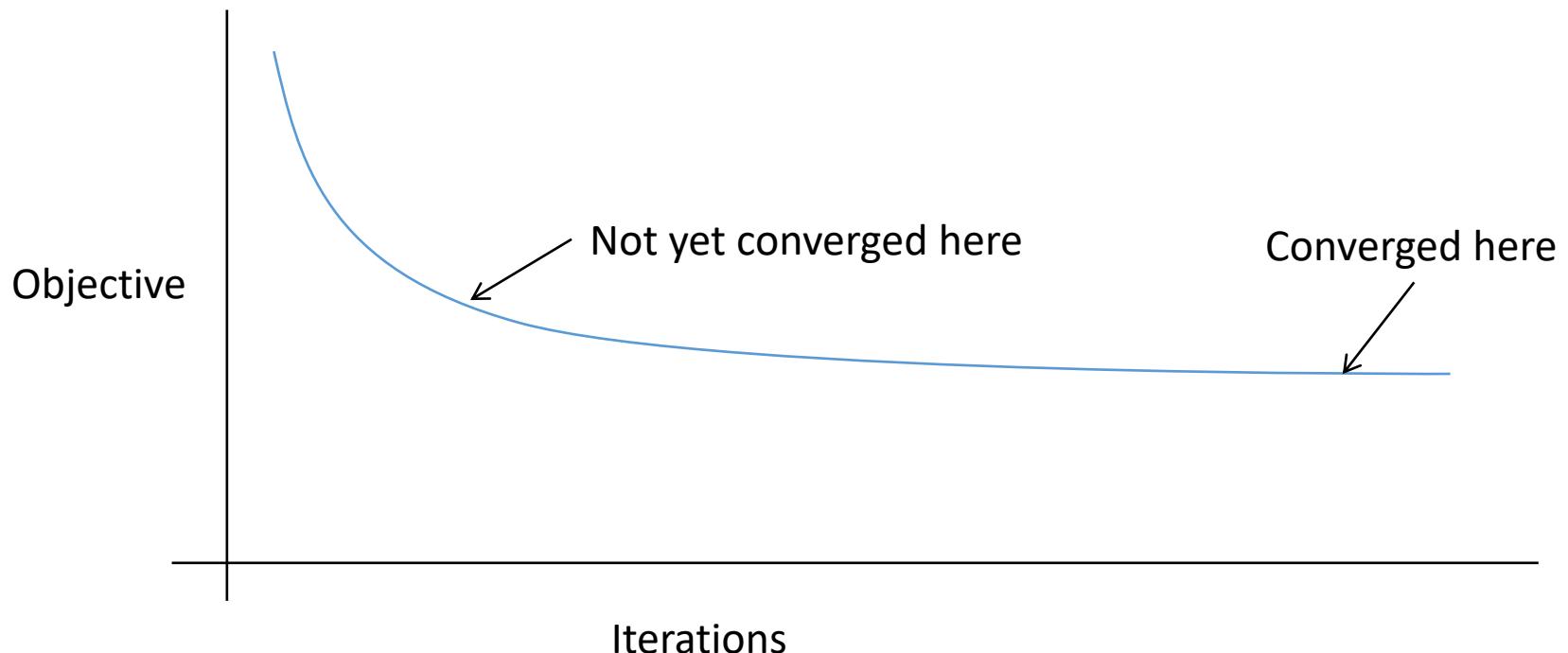
Diagnostics

Some possible problems:

- ✓ Over-fitting (high variance)
 - ✓ Under-fitting (high bias)
3. Your learning does not converge
 4. Are you measuring the right thing?

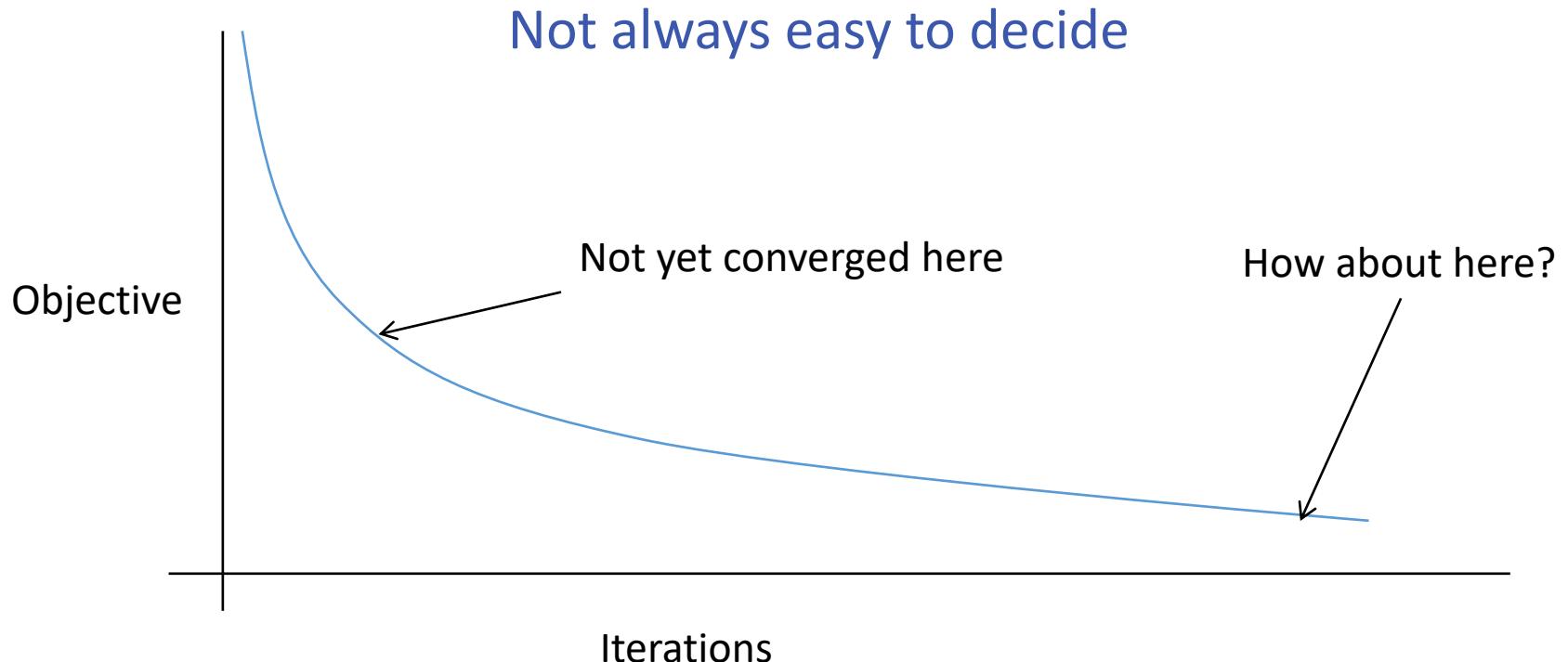
Does your learning algorithm converge?

If learning is framed as an optimization problem, track the objective function value



Does your learning algorithm converge?

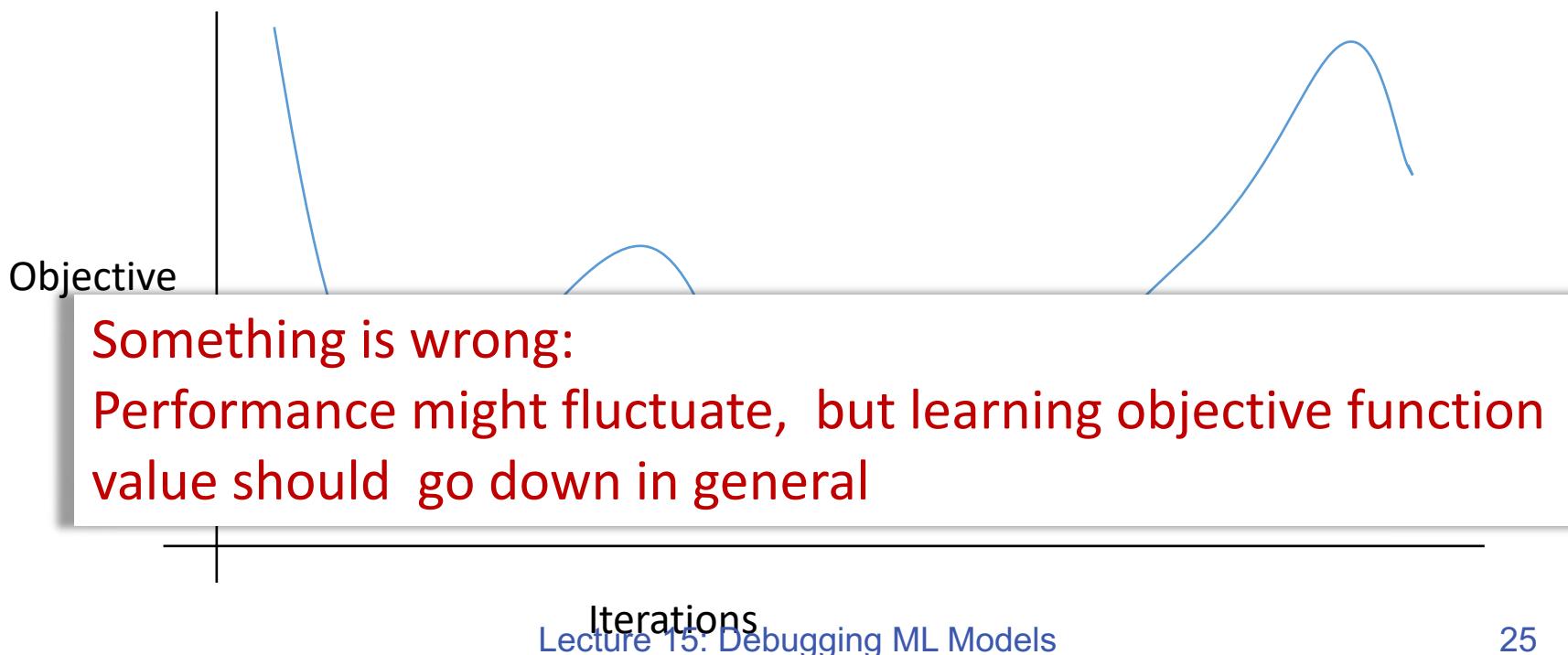
If learning is framed as an optimization problem, track the objective function value



Does your learning algorithm converge?

If learning is framed as an optimization problem, track the objective function value

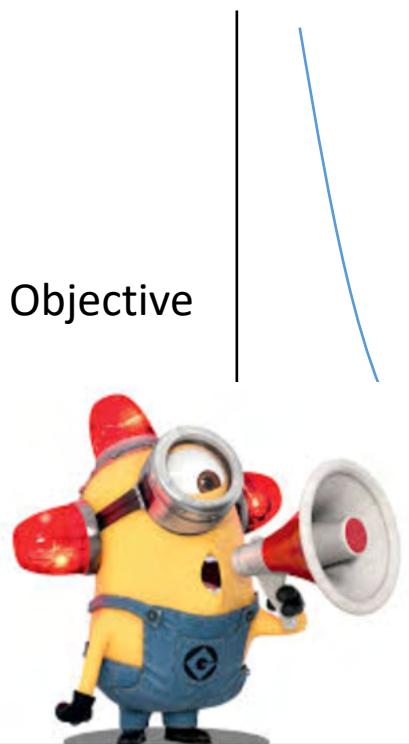
Check gradient!



Does your learning algorithm converge?

If learning is framed as an optimization problem, track the objective

Check gradient & related hyper-parameters!



Modern ML libraries with automatic differentiation make implementation easy. However, **gradient of some functions may not be able to estimate**.

The optimization process in these libraries may not converge (or converge very slowly)

Something is wrong

Diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

- ✓ Over-fitting (high variance)
- ✓ Under-fitting (high bias)
- ✓ Your learning does not converge
- ❖ Are you measuring the right thing?

What to measure

- ❖ Accuracy of prediction is the most common measurement
- ❖ If your data set is unbalanced, accuracy may be misleading
 - ❖ 1000 positive examples, 1 negative example
 - ❖ A classifier that always predicts positive will get 99.9% accuracy. Has it really learned anything?
- ❖ Unbalanced labels → measure label specific precision, recall and F-measure

Outline

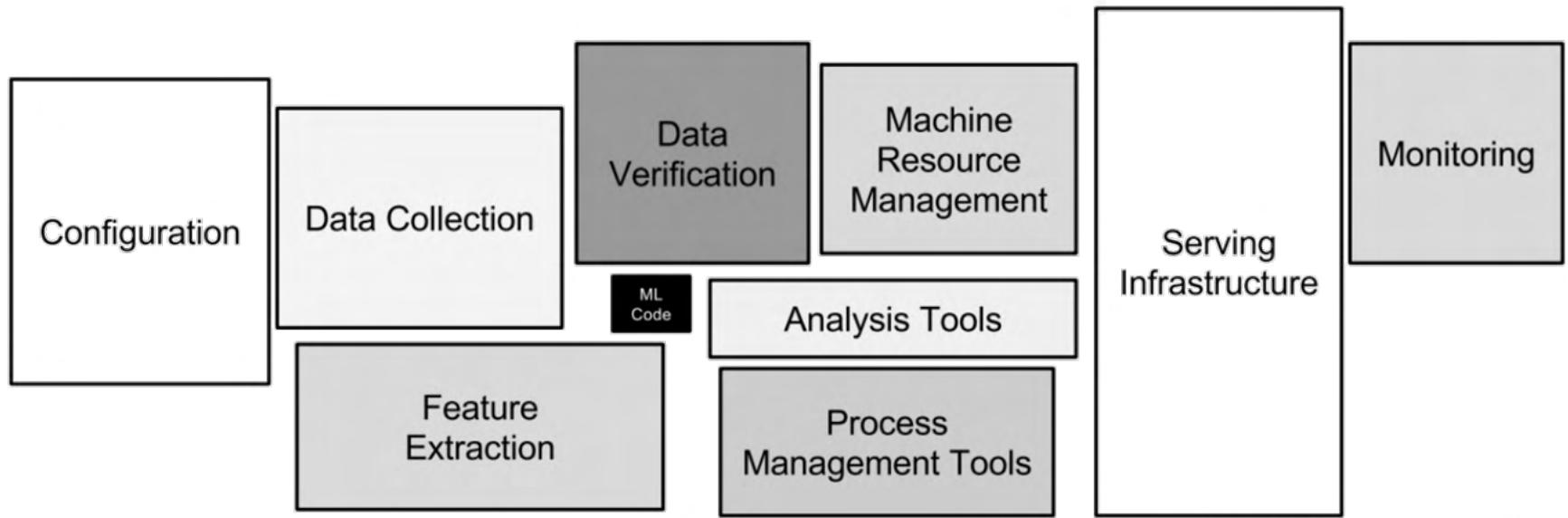
- ❖ Diagnostics of your learning algorithm
- ❖ Error analysis
- ❖ Injecting machine learning into *Your Favorite Task*

Machine Learning in this class



ML
code

Machine Learning in context



Error Analysis

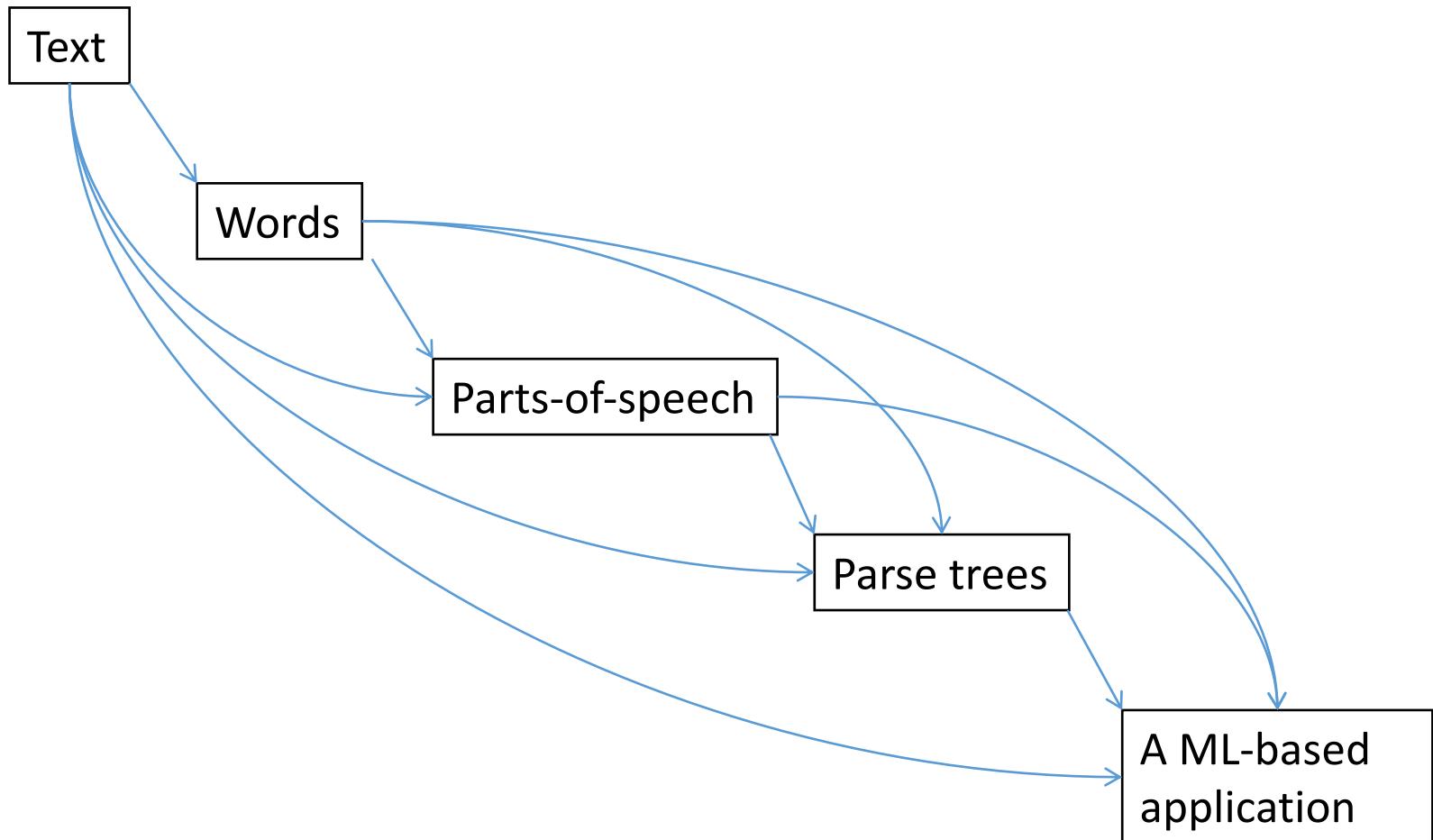
Generally machine learning plays a small role in a larger application

- ❖ Pre-processing
- ❖ Feature extraction (possibly by other ML based methods)
- ❖ Data transformations

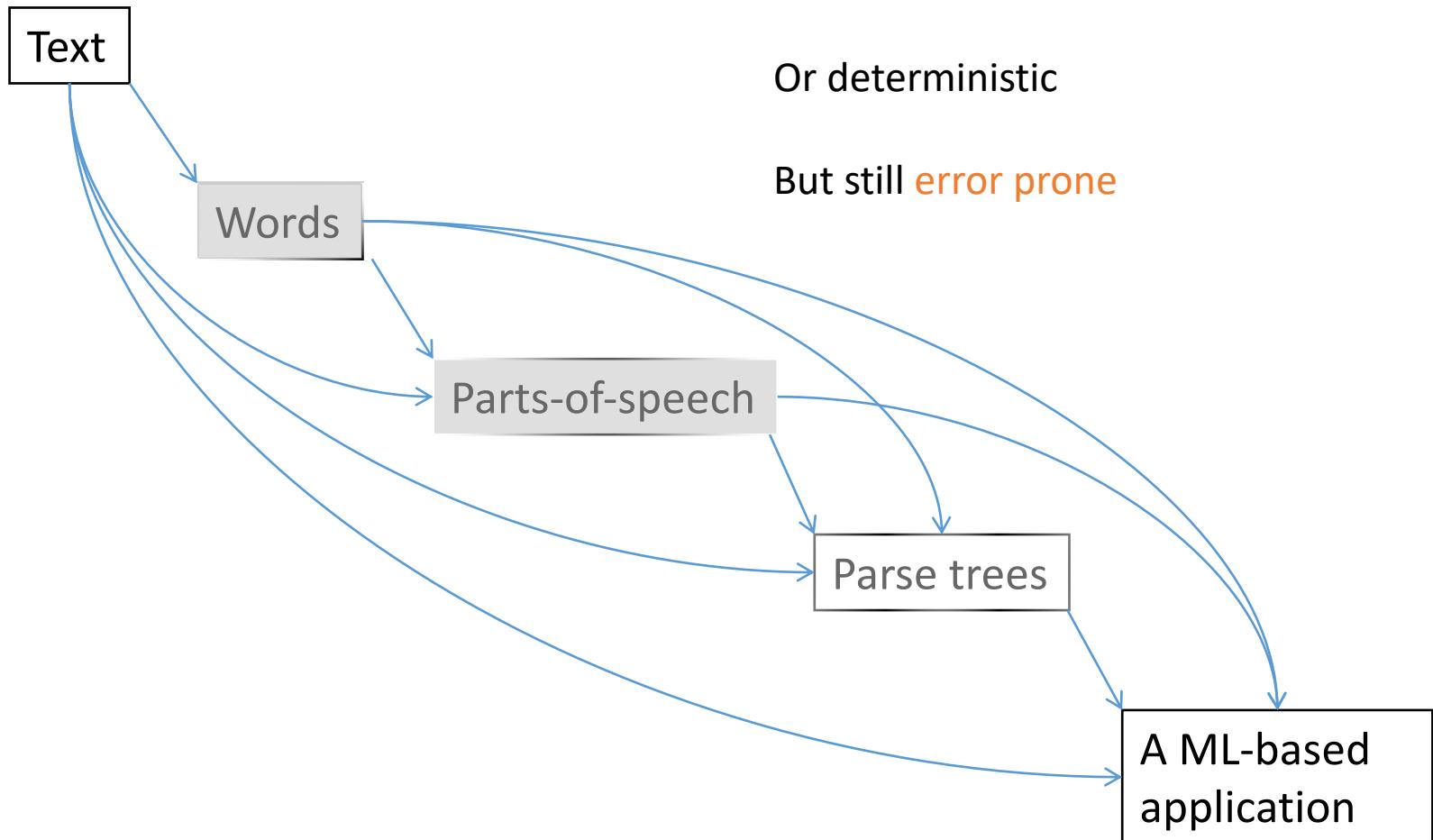
How much do each of these contribute to the error?

Error analysis tries to explain why a system is not performing perfectly

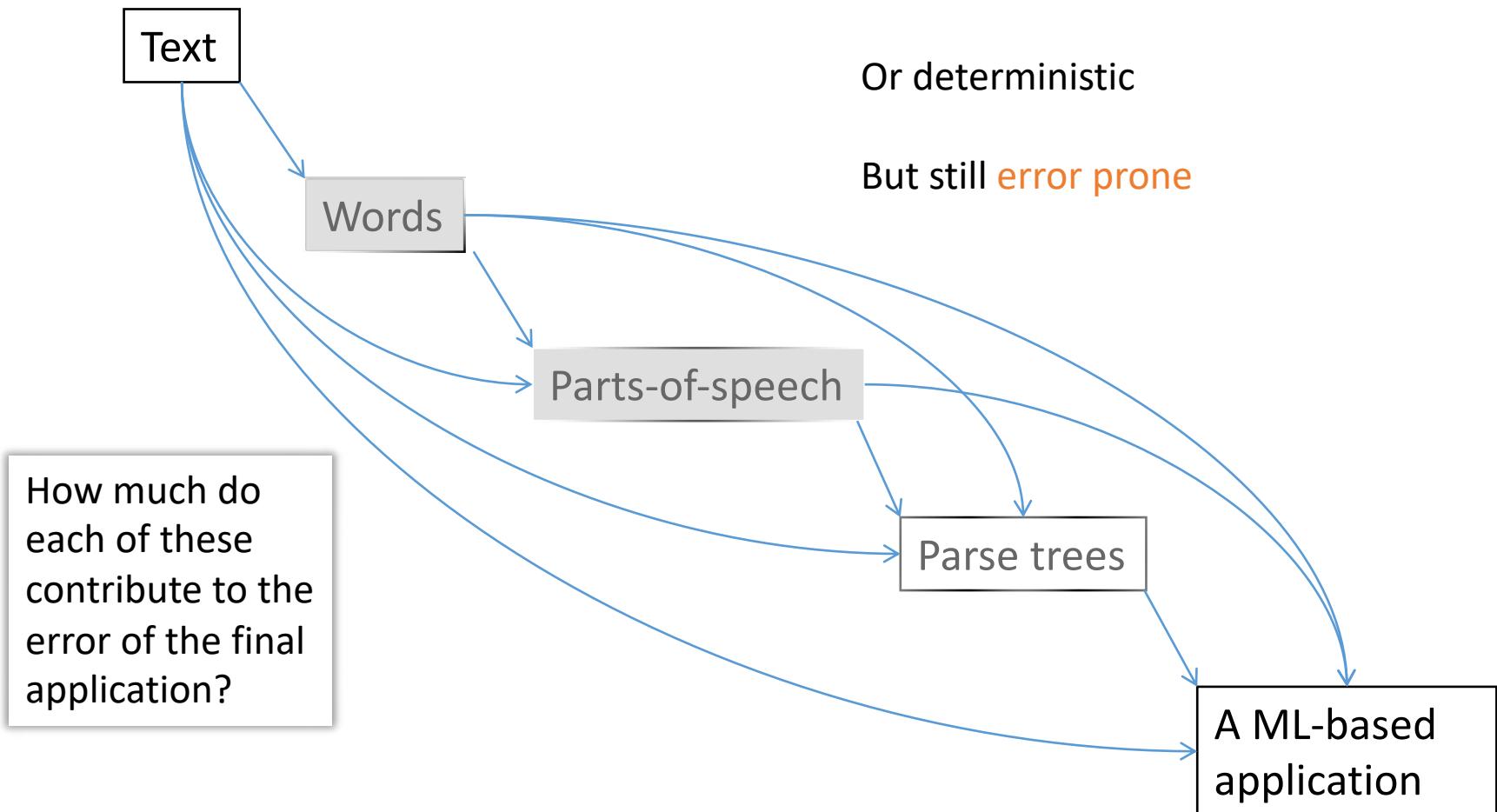
Example: A typical text processing pipeline



Example: A typical text processing pipeline



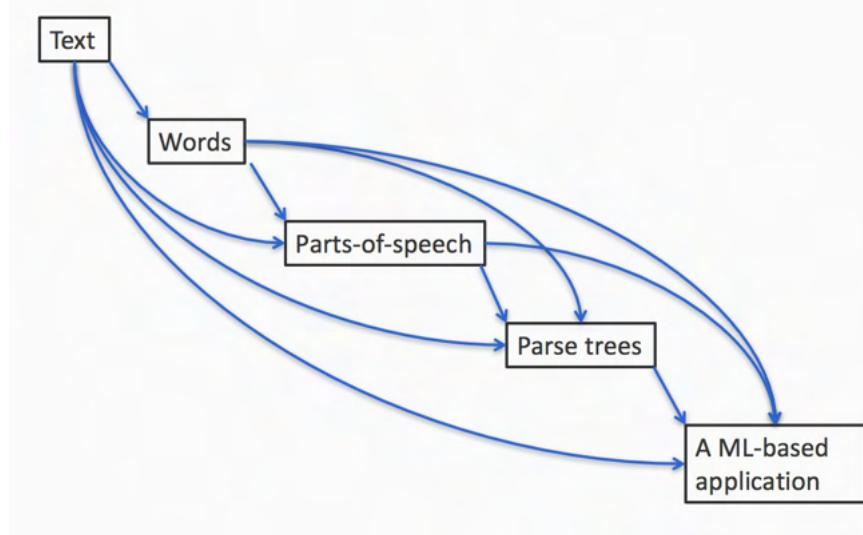
Example: A typical text processing pipeline



Tracking errors in a complex system

Plug in the ground truth for the intermediate components and see how much the accuracy of the final system changes

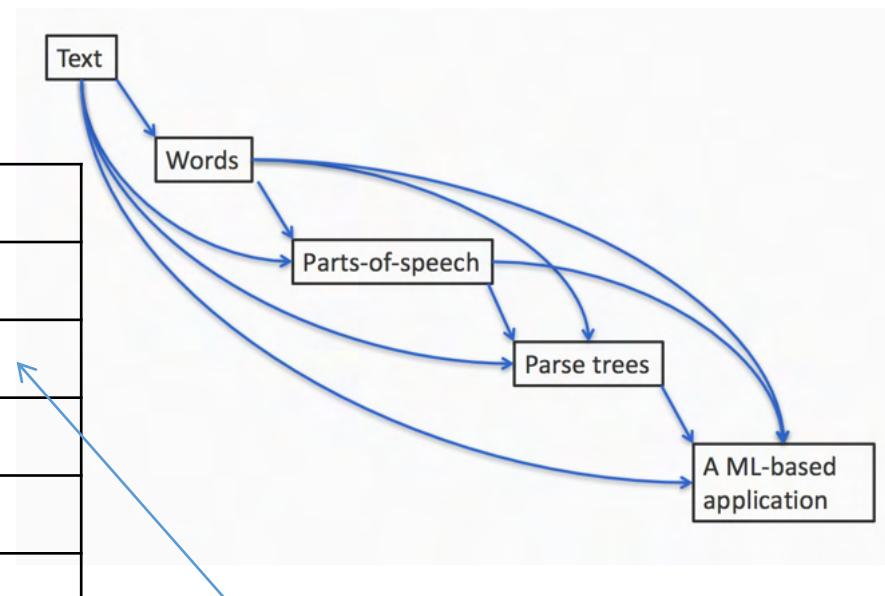
System	Accuracy
End-to-end predicted	55%
With ground truth words	60%
+ ground truth parts-of-speech	84 %
+ ground truth parse trees	89 %
+ ground truth final output	100 %



Tracking errors in a complex system

Plug in the ground truth for the intermediate components and see how much the accuracy of the final system changes

System	Accuracy
End-to-end predicted	55%
With ground truth words	60%
+ ground truth parts-of-speech	84 %
+ ground truth parse trees	89 %
+ ground truth final output	100 %



Error in the
part-of-speech
component hurts
the most

Ablative study

Explaining difference between the performance between a strong model and a much weaker one (a baseline)

Usually seen with features

Evaluate simpler systems that progressively use fewer and fewer features to see which features give the highest boost

Outline

- ❖ Diagnostics of your learning algorithm
- ❖ Error analysis
- ❖ Model explanation

Explain the model

❖ Spam prediction

	CONTENT	CLASS
267	PSY is a good guy	-1
173	For Christmas Song visit my channel! ;)	1

Example from <https://christophm.github.io/interpretable-ml-book/lime.html#lime-for-text>

Interpret Linear model

label_prob	feature	feature_weight
0.1701170	good	-3.563420
0.1701170	a	0.000000
0.1701170	is	-0.347321
0.9939024	channel!	6.180747
0.9939024	For	0.000000
0.9939024	;)	0.000000

Others

- ❖ Check data pre-processing
- ❖ Avoid data collection biases
- ❖ No leakage of label information
- ❖ Unit Testing
- ❖ Always push your code to Github and store the random seed and committee version number for reproducibility

Lecture 16: Clustering & Bayesian Learning

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

- ❖ Last hw – Due on 12/1
- ❖ Last Quiz – Due on 11/22

Clustering

Lec 16: Clustering & Bayesian
Learning

Hogwarts (Harry Potter)

- ❖ Sorting Hat – cluster kids into four groups based on four underlying prototypes



Godric
Gryffindor



Helga
Hufflepuff



Rowena
Ravenclaw



Salazar
Slytherin

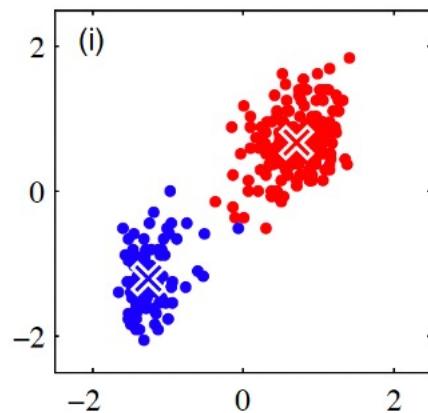
Goal of Clustering

- ❖ Given a collection of data points, the goal is to find structure in the data:
organize that data into sensible groups.

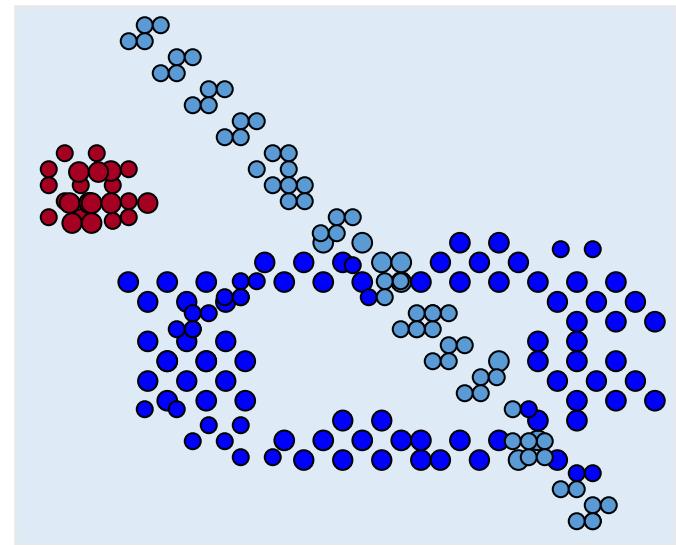
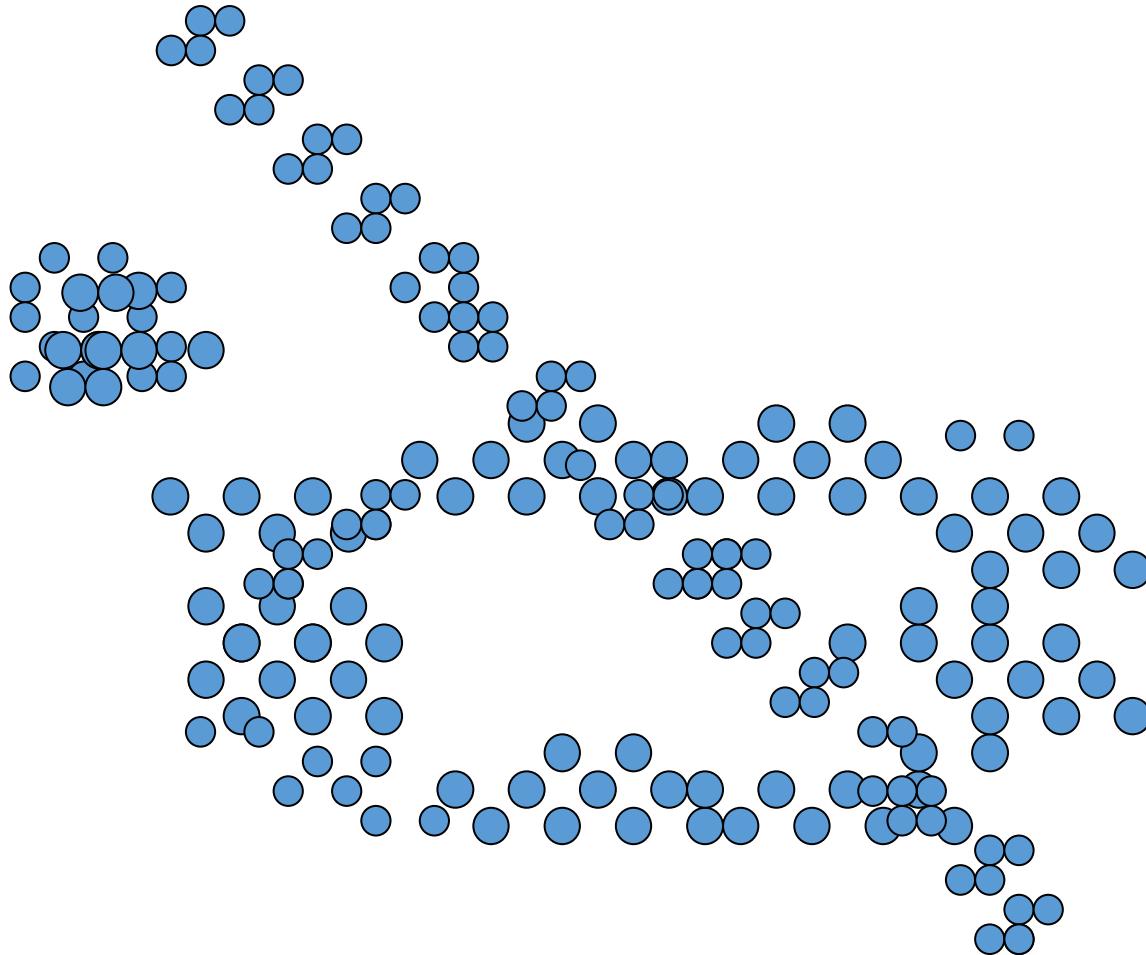
- ❖ Applications
 - ❖ Topics in news articles
 - ❖ Identify communities within social networks

How to define clusters

- ❖ A set of entities which are “alike”
- ❖ May be described as connected regions of a multi-dimensional space
- ❖ “We recognize a cluster when we see it”

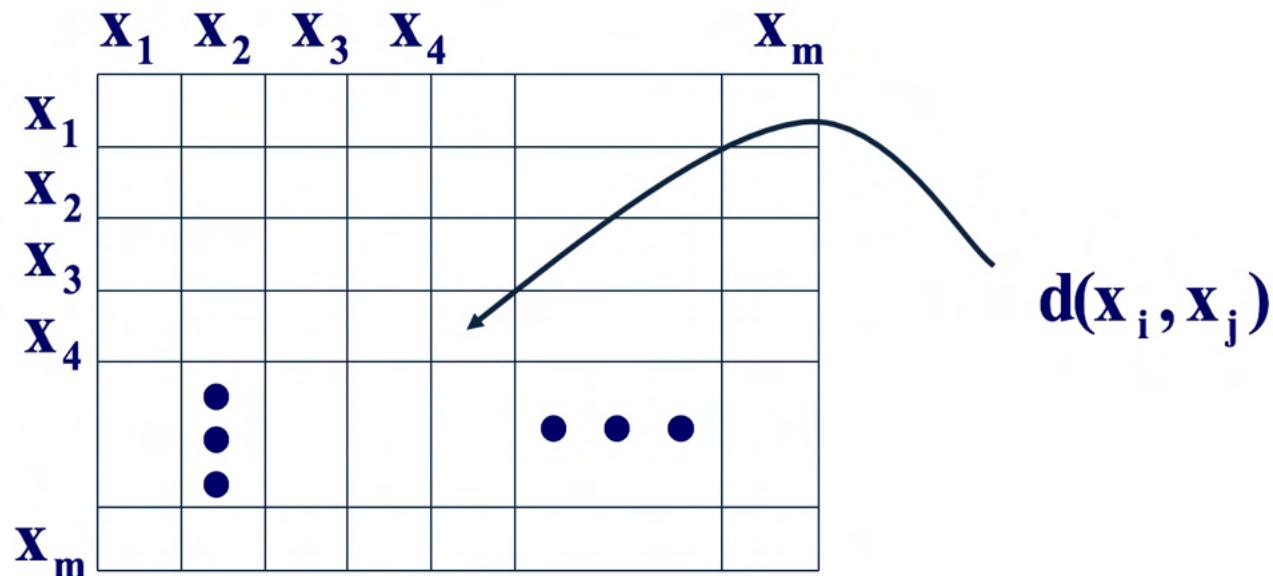


How many clusters are there?



Pairwise distance

- ❖ The pairwise distances are given
- ❖ We assume that the input to the problem is a matrix of distances between all pairs



Clustering

- ❖ An optimization problem:
 - ❖ Given a set of points and a pairwise distance, devise an algorithm f that splits the data so that it optimizes some conditions.

Setup Given $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ and K , we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes (or centroids) of clusters
- $A(\mathbf{x}_n) \in \{1, 2, \dots, K\}$: the cluster membership, i.e., the cluster ID assigned to \mathbf{x}_n

K-Means

Lec 16: Clustering & Bayesian
Learning

K-Means Intuition



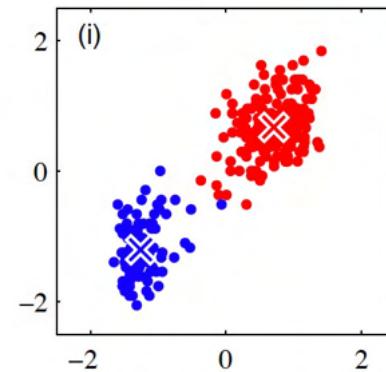
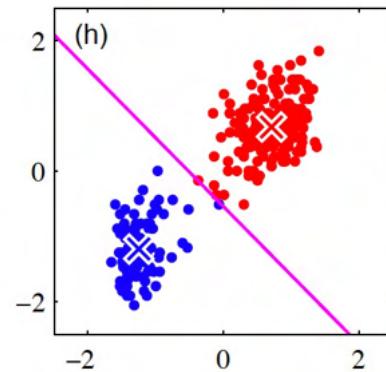
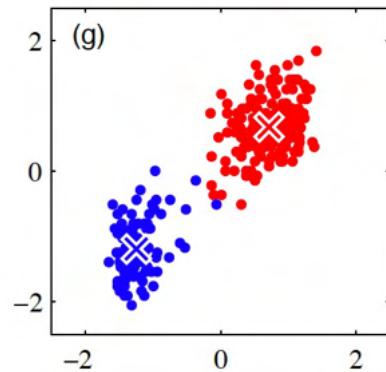
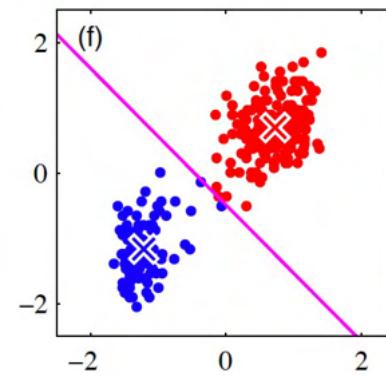
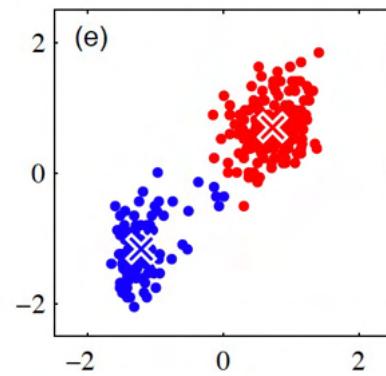
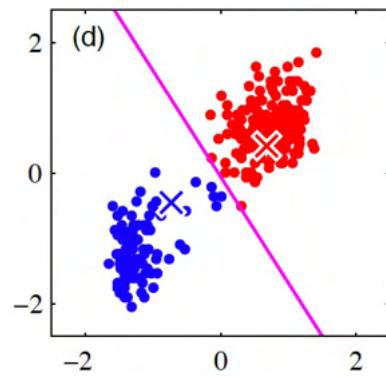
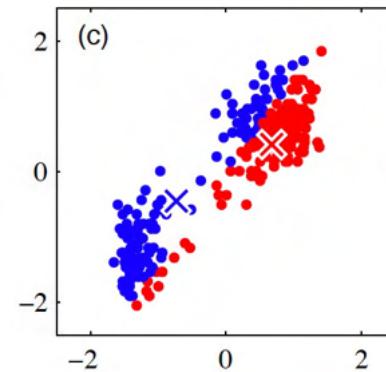
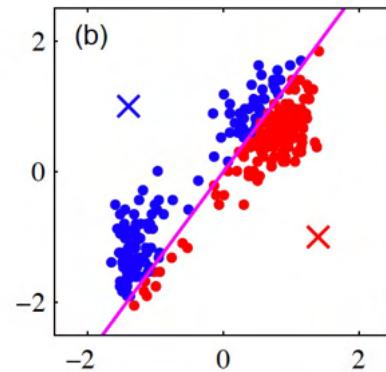
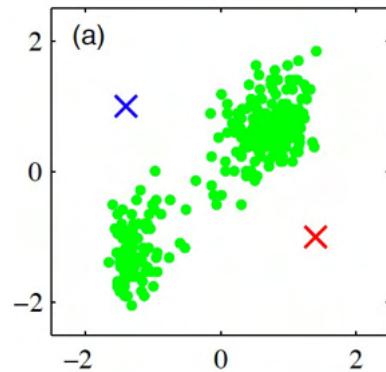
- ❖ Sorting Hat – cluster kids into four groups based on four underlying prototypes
- ❖ The prototype of each house is *the average of all kids of the house*
- ❖ Algorithm:
Alternatively, updating the prototype & the cluster assignment



<http://shabal.in/visuals/kmeans/6.html>

Lec 16: Clustering & Bayesian
Learning

Intuition of K-Means



K-means clustering

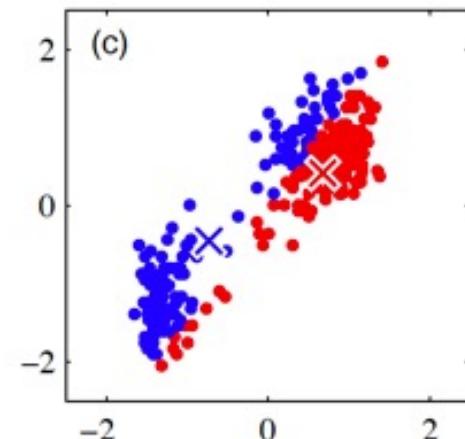
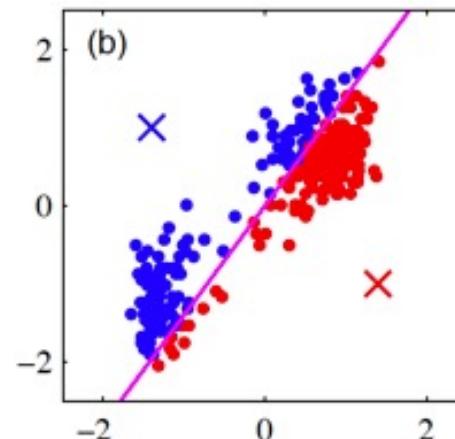
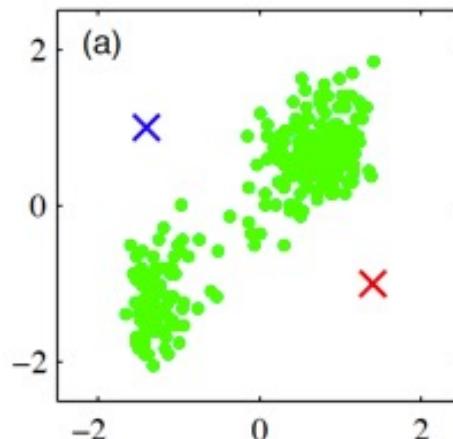
Sum of distances of all the points to their cluster center

- ❖ Distortion measure
(loss function for clustering)

$$J(\{r_{nk}\}, \{\mu_k\}) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if } A(\mathbf{x}_n) = k$$



K-means objective

$$\operatorname{argmin}_{\{r_{nk}\}, \{\mu_k\}} J(\{r_{nk}\}, \{\mu_k\}) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if } A(\mathbf{x}_n) = k$$

- ❖ It is a non-convex objective function
- ❖ Minimizing the above objective is NP-hard.

K-means algorithm a.k.a Lloyd's algorithm

- ❖ A greedy algorithm for minimizing K-means objective
 - alternative update $\{r_{nk}\}, \{\mu_k\}$
- ❖ Step 0: randomly assign the cluster centers $\{\mu_k\}$
- ❖ Step 1: Minimize J over $\{r_{nk}\}$ -- reassign cluster member
- ❖ Step 2: Minimize J over $\{\mu_k\}$ -- update the cluster centers
- ❖ Loop until it converges

$$J(\{r_{nk}\}, \{\mu_k\}) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|_2^2$$

K-means algorithm a.k.a Lloyd's algorithm

- ❖ Step 0: randomly assign the cluster centers $\{\mu_k\}$
- ❖ Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

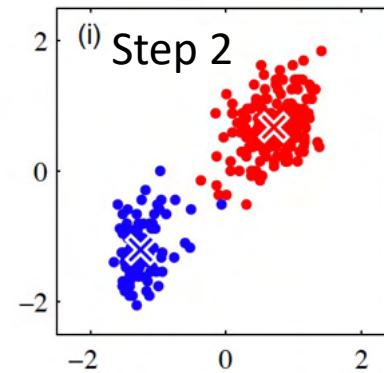
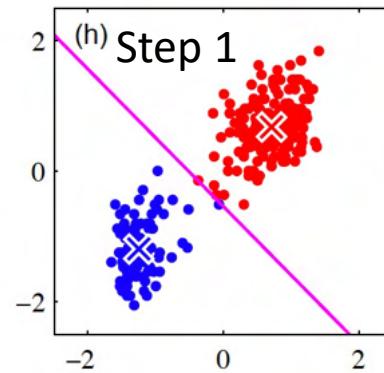
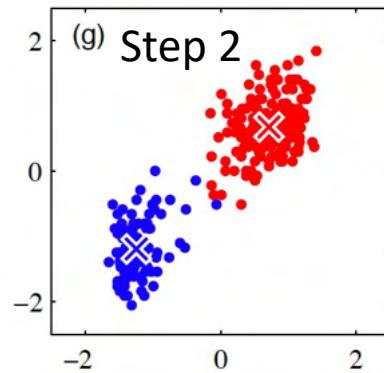
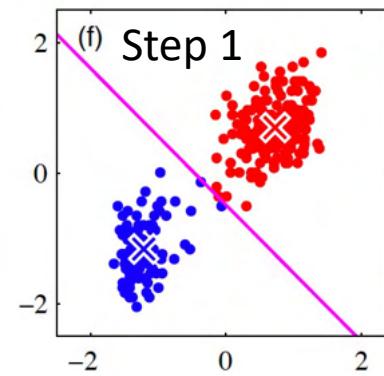
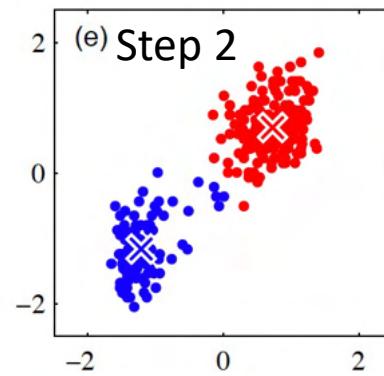
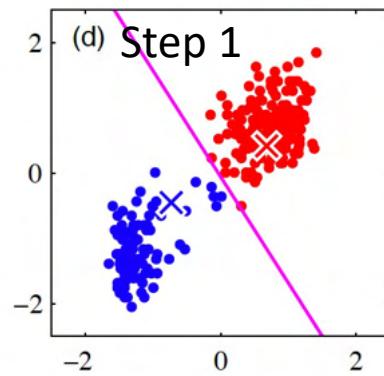
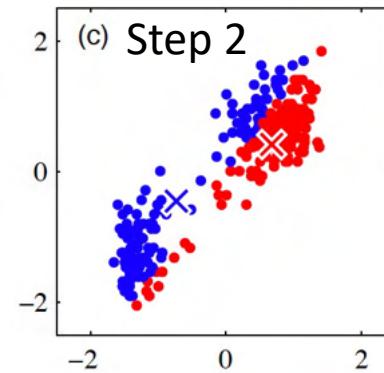
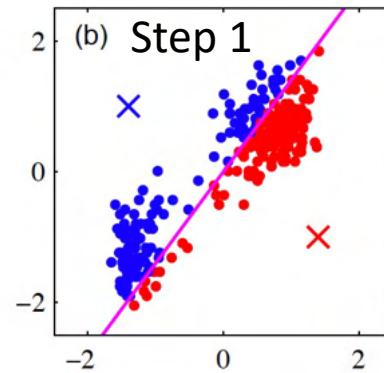
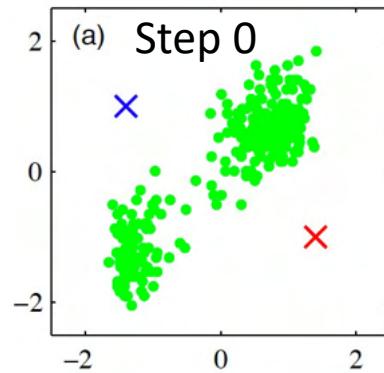
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- ❖ Step 2: Minimize J over $\{\mu_k\}$ -- update the cluster centers

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

- ❖ Loop until it converges

Example



Remarks

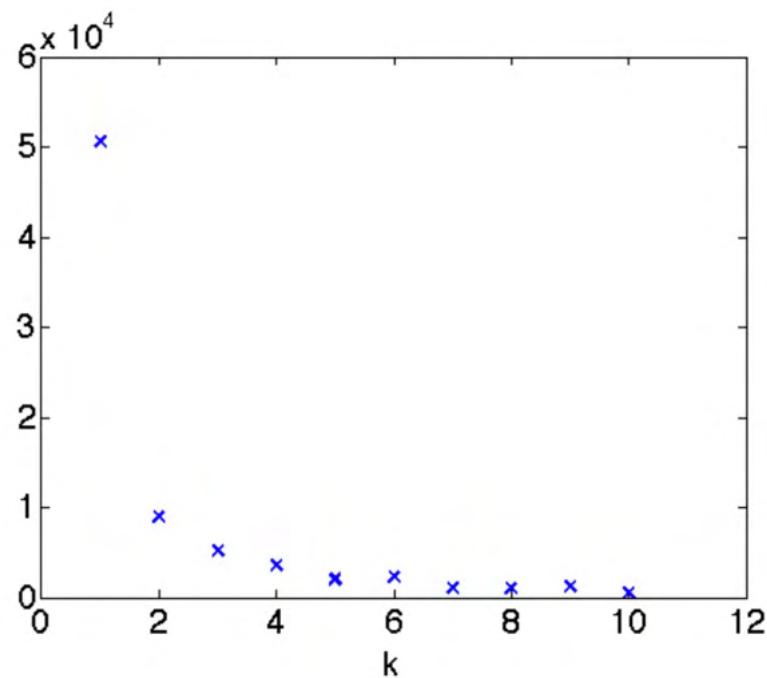
- ❖ Prototype μ_k is the mean of data points assigned to the cluster k , hence 'K-means'
- ❖ μ_k may not in the training set
- ❖ Need to pre-define k
 - ❖ There are some other approaches for the case k is unknown – not cover in class
- ❖ The procedure reduces J in both Step 1 and Step 2 and thus makes improvements or stay the same on each iteration

Properties of the K-means algorithm

- ❖ Does the K-means algorithm converge
 - ❖ Yes
- ❖ How long does it take to converge?
 - ❖ In the worst case, exponential in the number of data points
 - ❖ In practice, usually quick
- ❖ How good is its solution?
 - ❖ Local minimum (depends on the initialization)

Choosing K

- ❖ Increasing K will always decrease the optimal value of the K-means objective
 - ❖ It doesn't mean a better clustering
 - ❖ Analogous to overfitting in supervised learning.



K-means can be sensitive to the outlier

- ❖ One data point can make the center shift



K-Medoids

Lec 16: Clustering & Bayesian
Learning

K-medoids

- ❖ K-means is sensitive to outliers.
- ❖ In some applications we want the prototypes to be one of the points.
- ❖ Leads to K-medoids.

Intuition@ Hogwarts



- ❖ Sorting Hat – cluster students into four groups based on four underlying prototypes
- ❖ The prototype of each house is the most represented student of the house
 - ❖ Alternatively, updating the prototype & the student assignment

K-medoids algorithm

- ❖ Step 0: randomly selecting K points as the cluster centers $\{\mu_k\}$
- ❖ Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- ❖ Step 2: Update the cluster centers— the porotype for a cluster is the data that is closest to all other data points in the cluster

$$k* = \arg \min_{m:r_{mk}=1} \sum_n r_{nk} \|x_n - x_m\|_2^2$$

$$\mu_k = x_{k*}$$

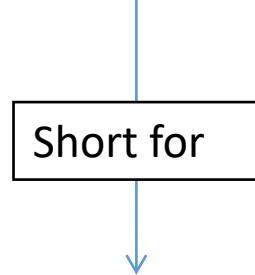
- ❖ Loop until it converges

Bayesian Learning

Lec 16: Clustering & Bayesian
Learning

Recap: Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$



$$\forall x, y \quad P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)}$$

Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Prior probability: What is our belief in Y before we see X?

Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Likelihood: What is the likelihood of observing X given a specific Y?

Prior probability: What is our belief in Y before we see X?

Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Posterior probability: What is the probability of Y given that X is observed?

Likelihood: What is the likelihood of observing X given a specific Y?

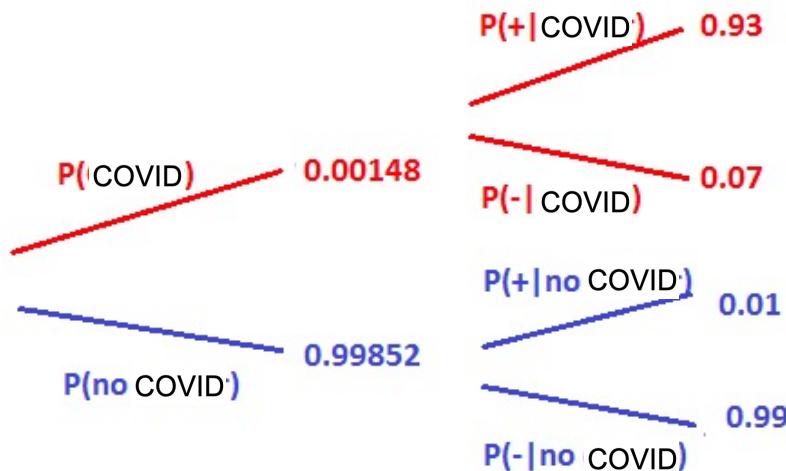
Prior probability: What is our belief in Y before we see X?

Recap: Bayes Theorem Example

- ❖ How likely the patient got COVID if the test is positive?

$$P(\text{COVID} | +) = \frac{P(\text{COVID and } +)}{P(\text{COVID and } +) + P(\text{no COVID and } +)} = 0.12$$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$



Bayes Theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Posterior probability: What is the probability of Y given that X is observed?

Likelihood: What is the likelihood of observing X given a specific Y?

Prior probability: What is our belief in Y before we see X?

$$\begin{aligned} \forall x, y \quad P(Y = y|X = x) &= \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)} \\ &= \frac{P(X = x|Y = y)P(Y = y)}{\sum_{y'} P(X = x|Y = y')P(Y = y')} \end{aligned}$$

Probabilistic Learning

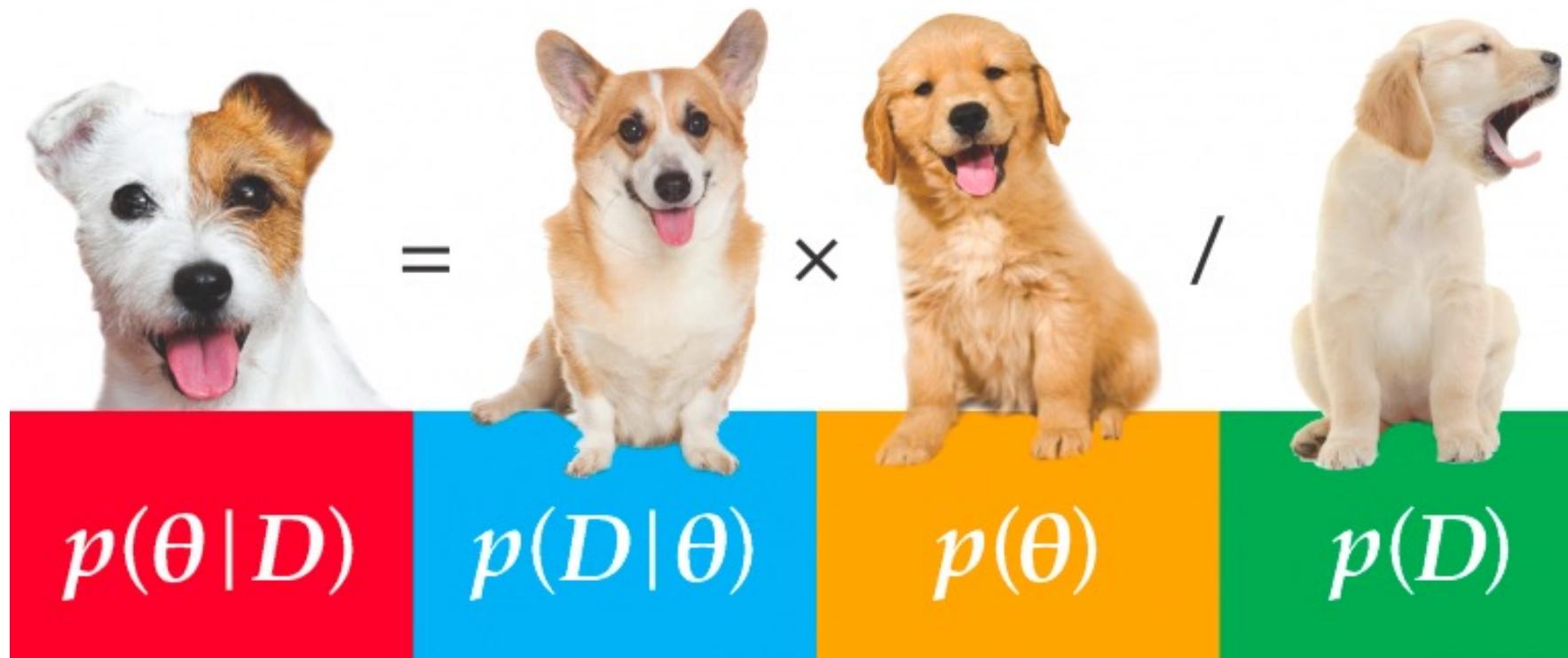
Two different notions of probabilistic learning

- ❖ **Bayesian Learning:** Use of a probabilistic criterion in selecting a hypothesis ($P(\Theta|D)$)
 - ❖ The hypothesis can be deterministic, a Boolean function
 - ❖ The criterion for selecting the hypothesis is probabilistic
- ❖ **Learning probabilistic concepts** ($P(Y|X)$)
 - ❖ The learned concept is a function $c:X \rightarrow [0,1]$
 - ❖ $c(x)$ may be interpreted as the probability that the label 1 is assigned to x

Today's lecture

- ❖ Bayesian Learning
- ❖ Maximum a posteriori and maximum likelihood estimation
- ❖ Naïve Bayes

Probabilistic models and Bayesian Learning



Bayesian Learning: The basics

- ❖ Goal: To find the **best** hypothesis from some space H of hypotheses, using the observed data D
- ❖ Define **best** = most probable hypothesis in H
- ❖ We assume a probability distribution over the class H

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h

What does *best* mean?

Bayesian learning uses $P(h | D)$, the conditional probability of a hypothesis given the data, to define *best*.

Bayesian Learning

Given a dataset D, we want to find
the best hypothesis h
What does *best* mean?

$$P(h|D)$$

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h
What does *best* mean?

$$P(h|D)$$

Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h
What does *best* mean?

$$P(h|D)$$

Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

! "#%&%(): Both h and D are events.

- D: The event that we observed *this* particular dataset
- h: The event that the hypothesis h is the true hypothesis

So we can apply the Bayes rule here.

Bayesian Learning

Given a dataset D, we want to find
the best hypothesis h
What does *best* mean?

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

! "#%&%(): Both h and D are events.

- D: The event that we observed *this* particular dataset
- h: The event that the hypothesis h is the true hypothesis

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h
What does *best* mean?

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

Prior probability of h: Background knowledge. What do we expect the hypothesis to be even before we see any data? For example, in the absence of any information, maybe the uniform distribution.

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h
What does *best* mean?

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

Likelihood: What is the probability that this data point (an example or an entire dataset) is observed, given that the hypothesis is h?

Prior probability of h: Background knowledge. What do we expect the hypothesis to be even before we see any data? For example, in the absence of any information, maybe the uniform distribution.

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h
What does *best* mean?

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

Likelihood: What is the probability that this data point (an example or an entire dataset) is observed, given that the hypothesis is h?

Prior probability of h: Background knowledge. What do we expect the hypothesis to be even before we see any data? For example, in the absence of any information, maybe the uniform distribution.

What is the probability that the data D is observed (independent of any knowledge about the hypothesis)?

Today's lecture

- ❖ Bayesian Learning
- ❖ Maximum a posteriori and maximum likelihood estimation
- ❖ Naïve Bayes

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(h|D)$$

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

Posterior / Likelihood \propto Prior

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

If we assume that the prior is uniform i.e. $P(h_i) = P(h_j)$, for all h_i, h_j

- ❖ Simplify this to get the Maximum Likelihood hypothesis

$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

Often computationally easier to maximize *log likelihood*

Maximum Likelihood estimation

Maximum Likelihood estimation (MLE)

$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

What we need in order to define learning:

1. A hypothesis space H
2. A model that says how data D is generated given h

Example: Bernoulli trials

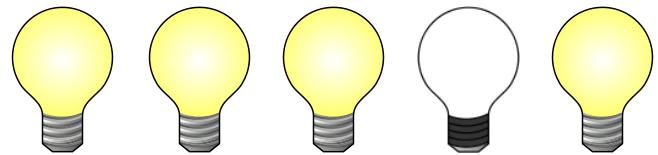
The CEO of a startup hires you for your first consulting job

- ❖ *CEO:* My company makes light bulbs. I need to know what is the probability they are faulty.
- ❖ *You:* Sure. I can help you out. Are they all identical?
- ❖ *CEO:* Yes!
- ❖ *You:* Excellent. I know how to help. We need to experiment...

Faulty lightbulbs

The experiment:

Try out 100 lightbulbs
80 work, 20 don't



You: The probability is $P(\text{failure}) = 0.2$

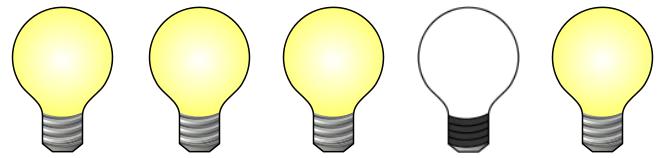
CEO: But how do you know?

You: Because...



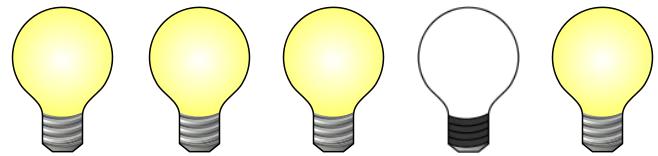
Bernoulli trials

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$
- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed



Bernoulli trials

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



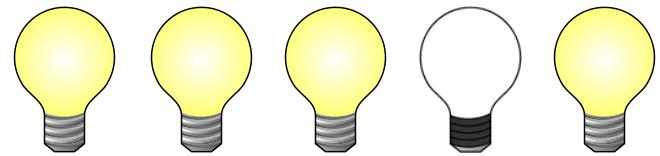
- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

Bernoulli trials

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

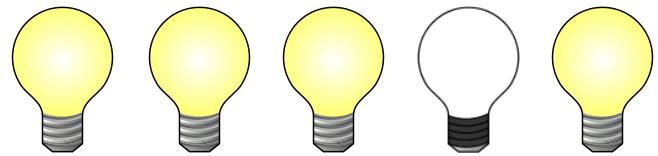
- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

- ❖ The most likely value of p for this observation is?

Bernoulli trials

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

- ❖ The most likely value of p for this observation is?

$$\underset{p}{\operatorname{argmax}} P(D|p) = \underset{p}{\operatorname{argmax}} \binom{100}{80} p^{80} (1-p)^{20}$$

The “learning” algorithm

Say you have a Work and b Not-Work

$$\begin{aligned} p_{best} &= \operatorname{argmax}_p P(D|h) \\ &= \operatorname{argmax}_p \log P(D|h) \\ &= \operatorname{argmax}_p \log \left(\binom{a+b}{a} p^a (1-p)^b \right) \\ &= \operatorname{argmax}_p a \log p + b \log(1-p) \end{aligned}$$

Calculus 101: Set the derivative to zero

$$P_{best} = a/(a + b)$$

The “learning” algorithm

Say you have a Work and b Not-Work

$$\begin{aligned} p_{best} &= \underset{p}{\operatorname{argmax}} P(D|h) \\ &= \underset{p}{\operatorname{argmax}} \log P(D | h) \quad \xrightarrow{\text{Log likelihood}} \\ &= \underset{p}{\operatorname{argmax}} \log \left(\binom{a+b}{a} p^a (1-p)^b \right) \\ &= \underset{p}{\operatorname{argmax}} a \log p + b \log(1 - p) \end{aligned}$$

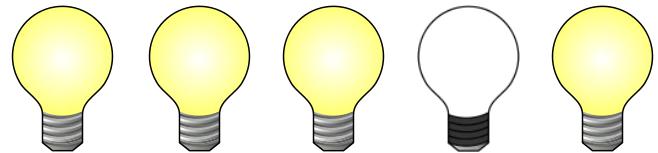
Calculus 101: Set the derivative to zero

$$P_{best} = a/(a + b)$$

Faulty lightbulbs

The experiment:

Try out 100 lightbulbs
80 work, 20 don't



You: The probability is $P(\text{failure}) = 0.2$

CEO: But how do you know?

You: Because...

CEO: Okay, but you only test 100 lightbulbs, can you calibrate your results based on our prior tests?

MAP estimation

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

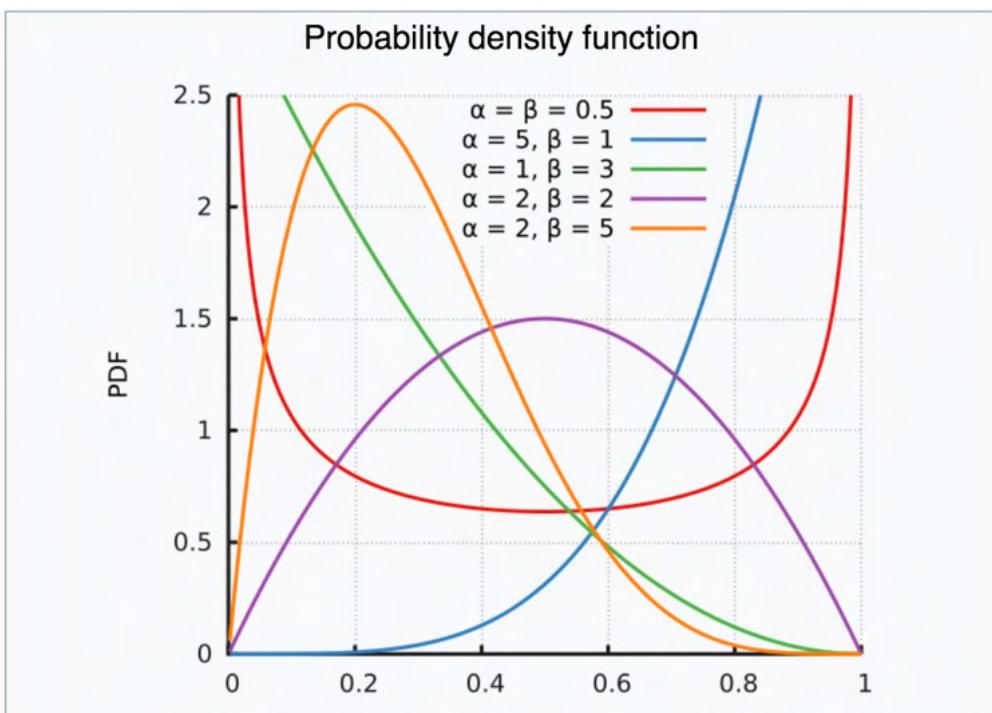
If we assume that the prior is uniform i.e. $P(h_i) = P(h_j)$, for all h_i, h_j

- ❖ Simplify this to get the Maximum Likelihood hypothesis

$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

Often computationally easier to maximize log likelihood
Leo Breiman & Bayesian
Learning

Beta distribution



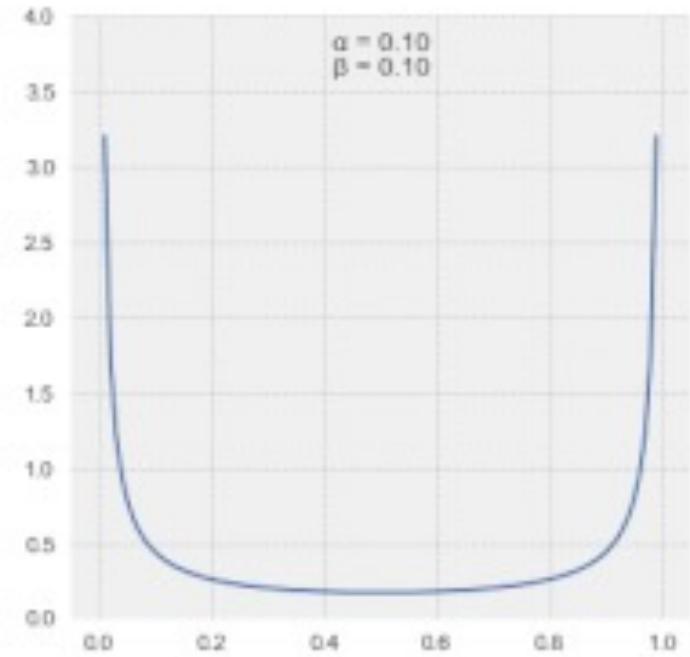
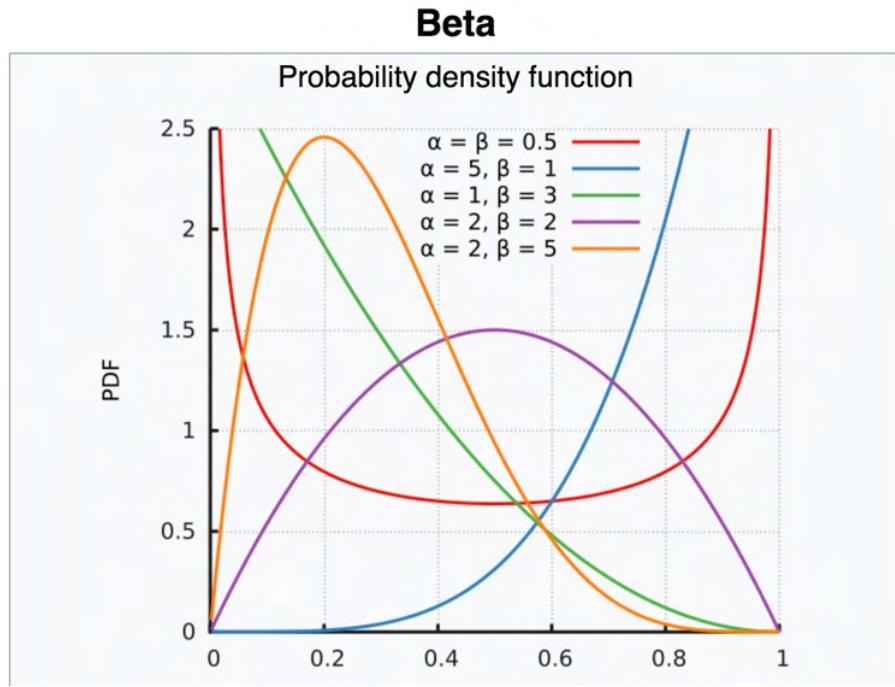
$$\begin{aligned} f(x; \alpha, \beta) &= \text{constant} \cdot x^{\alpha-1} (1-x)^{\beta-1} \\ &= \frac{x^{\alpha-1} (1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1} (1-u)^{\beta-1} du} \\ &= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1} \\ &= \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1} \end{aligned}$$

$$\alpha > 0, \beta > 0$$

https://en.wikipedia.org/wiki/Beta_distribution

Prior distribution

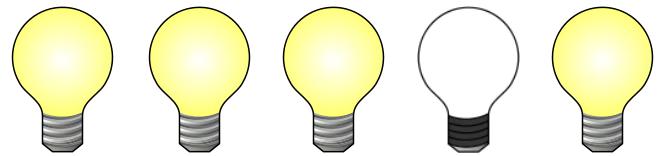
- ❖ The boss has a prior belief of the distribution of faulty lightbulb



MAP for Bernoulli trials

p is the parameter for hypothesis

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

MAP estimation

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

$$P(D|p) = \binom{a+b}{a} p^a (1-p)^b$$

$$P(p) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1-p)^{\beta-1}$$

$$\begin{aligned} p_{best} &= \operatorname{argmax}_p P(D|h) P(h) \\ &= \operatorname{argmax}_p \log P(D | h) + \log P(h) \\ &= \operatorname{argmax}_p \log \left(\frac{\binom{a+b}{a}}{B(\alpha, \beta)} p^a (1-p)^b p^{\alpha-1} (1-p)^{\beta-1} \right) \\ &= \operatorname{argmax}_p (a + \alpha - 1) \log p + (b + \beta - 1) \log(1 - p) \end{aligned}$$

MAP v.s. MLE

❖ MLE:

$$\operatorname{argmax}_p a \log p + b \log(1 - p)$$

$$\Rightarrow p_{best} = \frac{a}{a + b}$$

❖ MAP (w/ Beta distribution as prior)

$$\operatorname{argmax}_p (a + \alpha - 1) \log p + (b + \beta - 1) \log(1 - p)$$

$$\Rightarrow p_{best} = \frac{a + \alpha - 1}{a + b + \alpha + \beta - 2}$$

MAP v.s. MLE

❖ MAP

$$\operatorname{argmax}_p (a + \alpha - 1) \log p + (b + \beta - 1) \log(1 - p)$$

$$\Rightarrow p_{best} = \frac{a + \alpha - 1}{a + b + \alpha + \beta - 2}$$

❖ Let $\alpha = 100, \beta = 10$

❖ $a = 10, b = 20 \Rightarrow p_{best} \approx 0.79$

❖ $a = 1000, b = 2000 \Rightarrow p_{best} \approx 0.36$

❖ $a = 100,000, b = 200,000 \Rightarrow p_{best} \approx 0.33$

MAP for logistic regression

Let's get back to the MLE for logistic regression

- ❖ Training data
 - ❖ $S = \{(x_i, y_i)\}$, m examples
- ❖ What we want
 - ❖ Find a w such that $P(S | w)$ is maximized
 - ❖ We know that our examples are drawn independently and are identically distributed (i.i.d)
 - ❖ How do we proceed?

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

The usual trick: Convert products to sums by taking log

Recall that this works only because log is an increasing function and the maximizer will not change

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

But (by definition) we know that

$$P(y|\mathbf{w}, \mathbf{x}) = \sigma(y_i \mathbf{w}^T \mathbf{x}_i) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

$$P(y|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$P(y|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

Maximum likelihood estimation

$\underset{\mathbf{w}}{\operatorname{argmax}}$

$$\underset{\mathbf{w}}{\operatorname{argmax}} P(S|\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

The goal: Maximum likelihood training of a discriminative probabilistic classifier under the logistic model for the posterior distribution.

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

Equivalent to: Training a linear classifier by minimizing the *logistic loss*.

Maximum a posteriori estimation

We could also add a prior on the weights

Suppose each weight in the weight vector is drawn independently from the normal distribution with zero mean and standard deviation σ

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Let us work through this procedure again to see what changes

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$
$$\left| \max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w}) \right|$$

Equivalent to solving

$$\downarrow$$
$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Let us work through this procedure again to see what changes

What is the goal of MAP estimation? (In maximum likelihood, we maximized the likelihood of the data)

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

What is the goal of MAP estimation? (In maximum likelihood, we maximized the likelihood of the data)

To maximize the posterior probability of the model given the data (i.e. to find the most probable model, given the data)

$$P(\mathbf{w}|S) \propto P(S|\mathbf{w})P(\mathbf{w})$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|S) = \operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

We have already expanded out the first term.

$$\sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

Expand the log prior

$$\sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \sum_{j=1}^d \frac{-w_j^2}{\sigma^2} + \text{constants}$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \sum_{j=1}^d \frac{-w_j^2}{\sigma^2} + \text{constants}$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) - \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w}$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) - \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w}$$

Maximizing a negative function is the same as minimizing the function
Lec 16: Clustering & Bayesian Learning

Learning a logistic regression classifier

Learning a logistic regression classifier is equivalent to solving

$$\min_{\mathbf{w}} \sum_i^m \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w}$$

Lecture: Fairness in ML

Dr. Sunipa Dev
Postdoc and Computing Innovation Fellow at UCLA NLP

AI Uses in Daily Life

- Chatbots and virtual assistants; Siri, Alexa
- Automatic mail sorting
- Face recognition phone and computer locks
- Resume sorting and job recommendations
- Loan applications and fraud detection

Any more examples?

Are these applications fair to all?

COMPAS

	WHITE	AFRICAN AMERICAN
Labeled Higher Risk, But Didn't Re-Offend	23.5%	44.9%
Labeled Lower Risk, Yet Did Re-Offend	47.7%	28.0%

Overall, Northpointe's assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes. (Source: ProPublica analysis of data from Broward County, Fla.)

Definition: Bias in ML

- Disparate performance/accuracy for persons with different protected attributes
- Potential for causing harm
- Distinct from model error —> disparity is learnt by model

XING Ranking

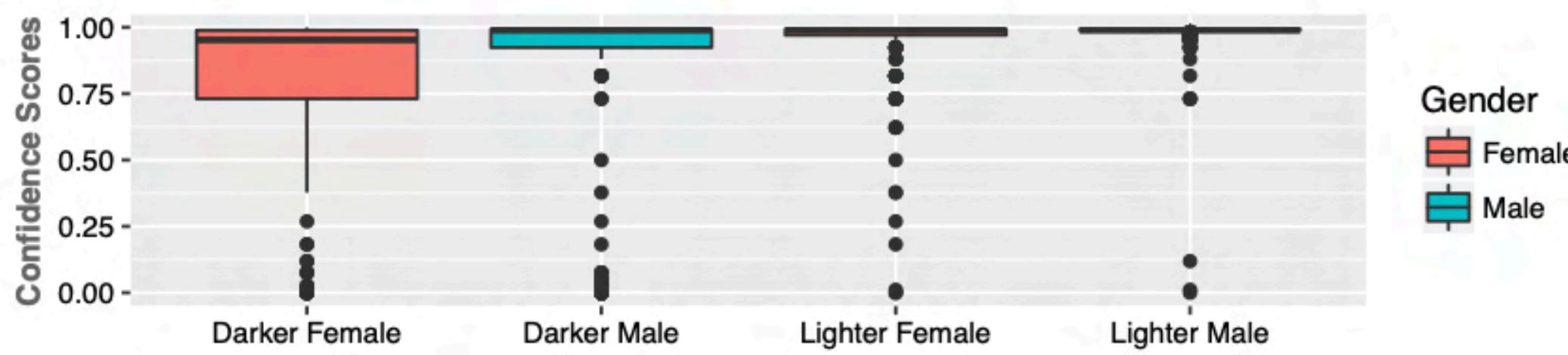
Search query	Work experience	Education experience	Profile views	Candidate	Xing ranking
Brand Strategist	146	57	12992	male	1
Brand Strategist	327	0	4715	female	2
Brand Strategist	502	74	6978	male	3
Brand Strategist	444	56	1504	female	4
Brand Strategist	139	25	63	male	5
Brand Strategist	110	65	3479	female	6
Brand Strategist	12	73	846	male	7
Brand Strategist	99	41	3019	male	8
Brand Strategist	42	51	1359	female	9
Brand Strategist	220	102	17186	female	10

Bias in Facial Recognition

Classifier	Metric	All	F	M	Darker	Lighter	DF	DM	LF	LM
MSFT	PPV(%)	93.7	89.3	97.4	87.1	99.3	79.2	94.0	98.3	100
	Error Rate(%)	6.3	10.7	2.6	12.9	0.7	20.8	6.0	1.7	0.0
	TPR (%)	93.7	96.5	91.7	87.1	99.3	92.1	83.7	100	98.7
	FPR (%)	6.3	8.3	3.5	12.9	0.7	16.3	7.9	1.3	0.0
Face++	PPV(%)	90.0	78.7	99.3	83.5	95.3	65.5	99.3	94.0	99.2
	Error Rate(%)	10.0	21.3	0.7	16.5	4.7	34.5	0.7	6.0	0.8
	TPR (%)	90.0	98.9	85.1	83.5	95.3	98.8	76.6	98.9	92.9
	FPR (%)	10.0	14.9	1.1	16.5	4.7	23.4	1.2	7.1	1.1
IBM	PPV(%)	87.9	79.7	94.4	77.6	96.8	65.3	88.0	92.9	99.7
	Error Rate(%)	12.1	20.3	5.6	22.4	3.2	34.7	12.0	7.1	0.3
	TPR (%)	87.9	92.1	85.2	77.6	96.8	82.3	74.8	99.6	94.8
	FPR (%)	12.1	14.8	7.9	22.4	3.2	25.2	17.7	5.20	0.4

Fig4: The bias in commercial face recognition services([Buolamwini and Gebru, 2018](#)). DF, DM, LF, LM stand for: darker skin female, darker skin male, lighter skin female and lighter skin male. PPV, TPR, FPR stand for predictive positive value, true positive rate and false positive rate.

Bias in Facial Recognition



Gender classification confidence scores from IBM ([IBM](#)). Scores are near 1 for lighter male and female subjects while they range from $\sim 0.75 - 1$ for darker females.

Dataset Gender Bias



Male

33%

66%



Female

Model Bias After Training

16%



Male

84%



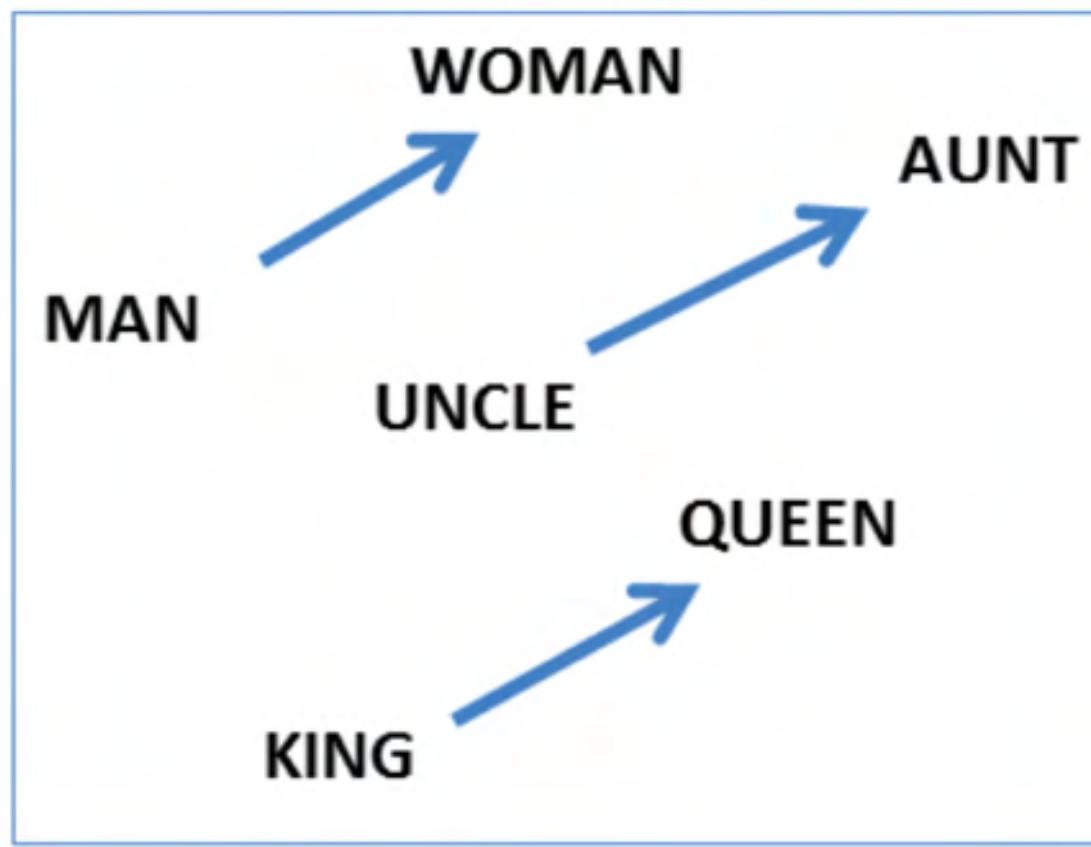
Female

Word Embeddings can be Dreadfully Sexist

[nips16, reported by **NPR**, **MIT tech review**]



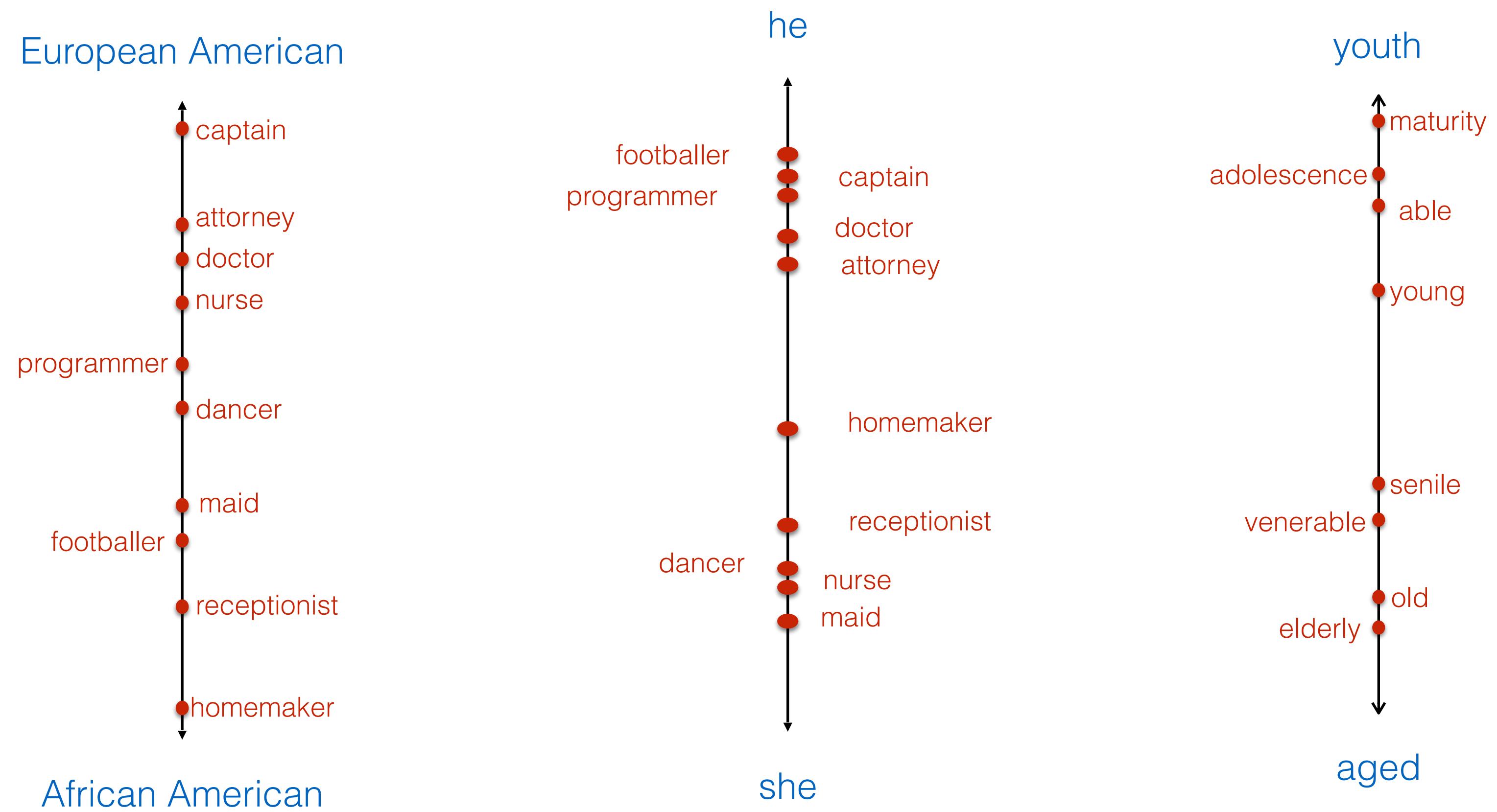
$$v_{man} - v_{woman} + v_{uncle} \sim v_{aunt}$$



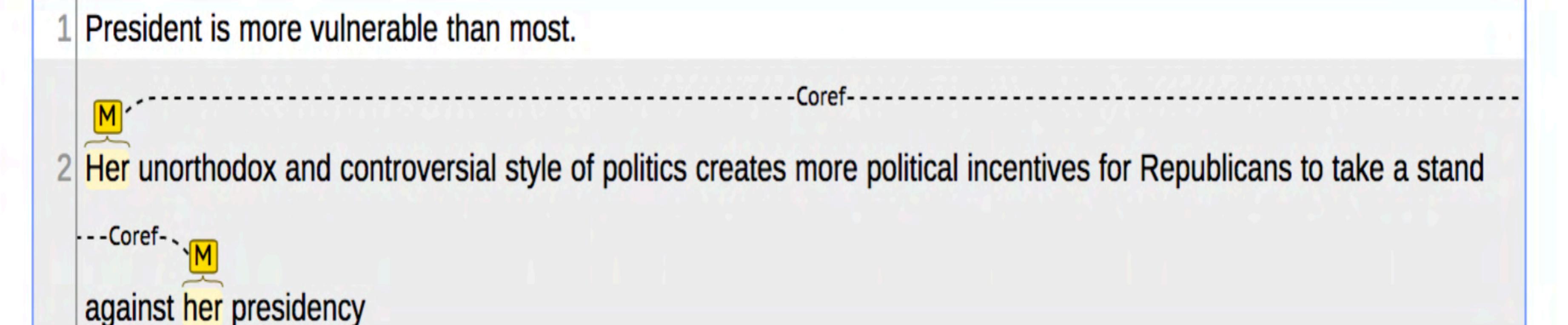
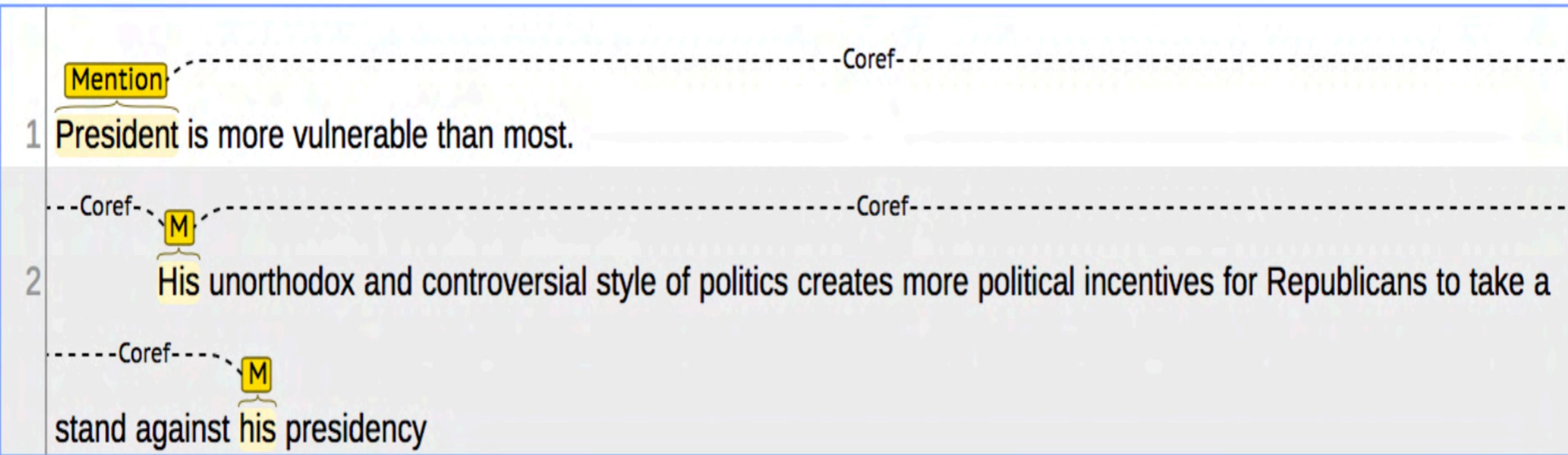
he: __	she: __
uncle	aunt
lion	
surgeon	
architect	
beer	
professor	

We use Google w2v embedding trained from the news

Bias in Word Representation



Gender Bias in Coref [NAACL 2018]



Concurrent work (Rudinger et al., 2018) @NAACL18 also studied gender bias in Coref.
Natural Language Processing

Gender-Inclusivity in NLP

1 Allen and 0 Amy went to the market and 0 Amy asked 1 Allen if 1 he wants anything .

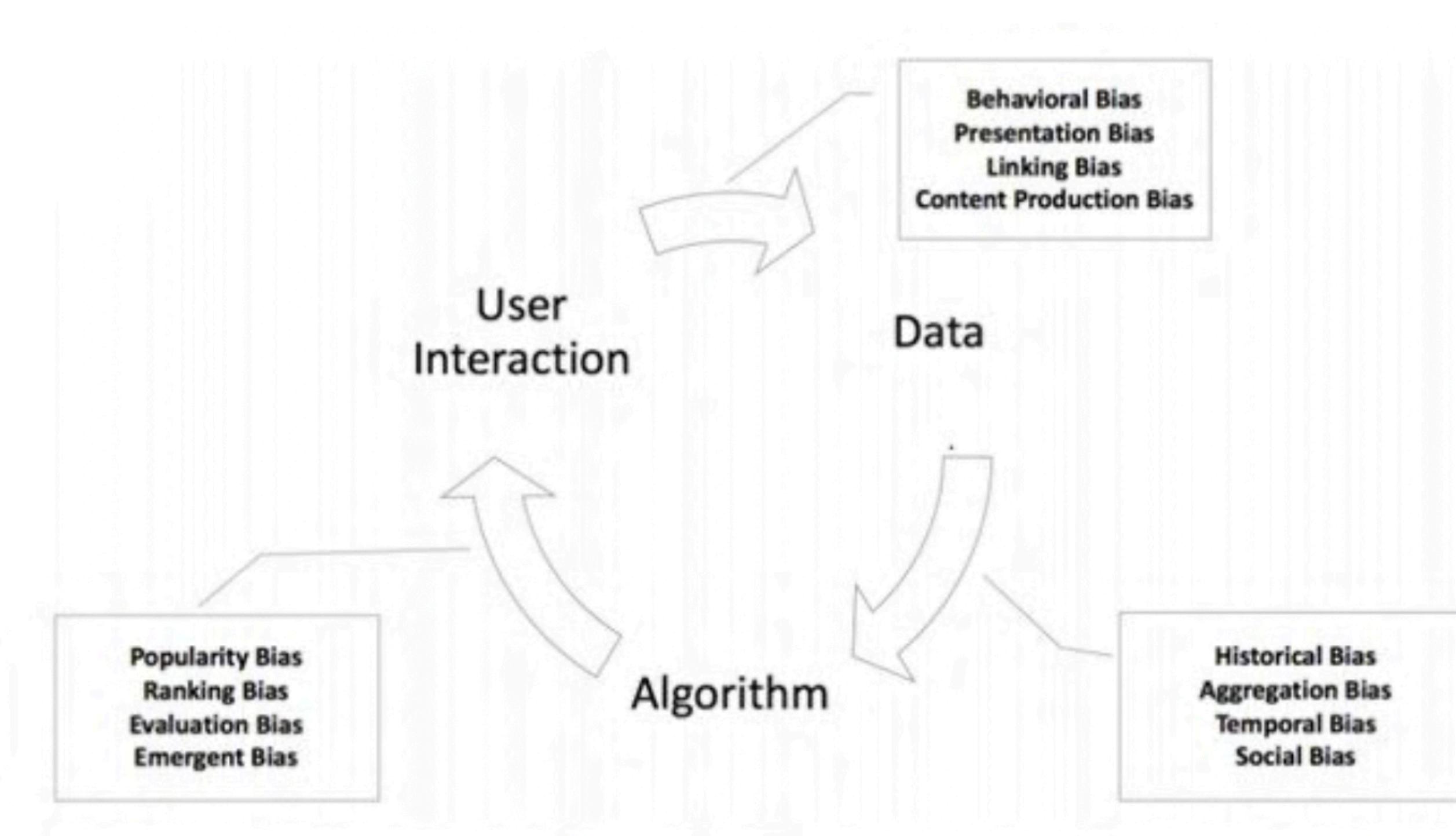
1 Allen and 0 Amy went to the market and 0 Amy asked 1 Allen if they want anything .

AI Uses in Daily Life

- Chatbots and virtual assistants; Siri, Alexa
- Automatic mail sorting
- Face recognition phone and computer locks
- Resume sorting and job recommendations
- Loan applications and fraud detection

Why do these biases appear?

Why do these biases appear?



Why do these biases appear?

Skewed Samples

Wikipedia text ~4.5 billion tokens

- he: 15 million
- she: 4.8 million
- they: 4.9 million
- ze: 7.4 thousand
- xe: 4.5 thousand

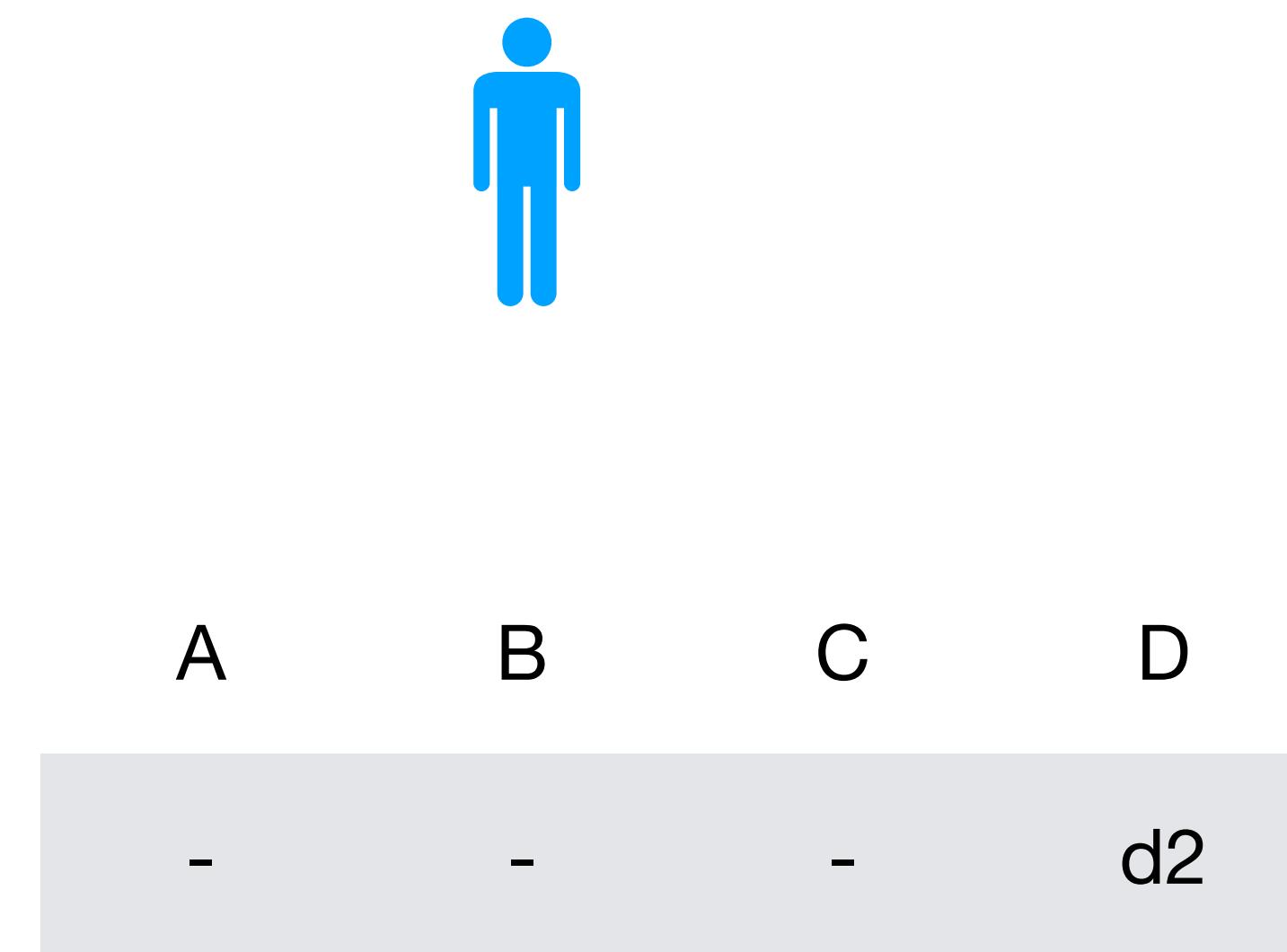
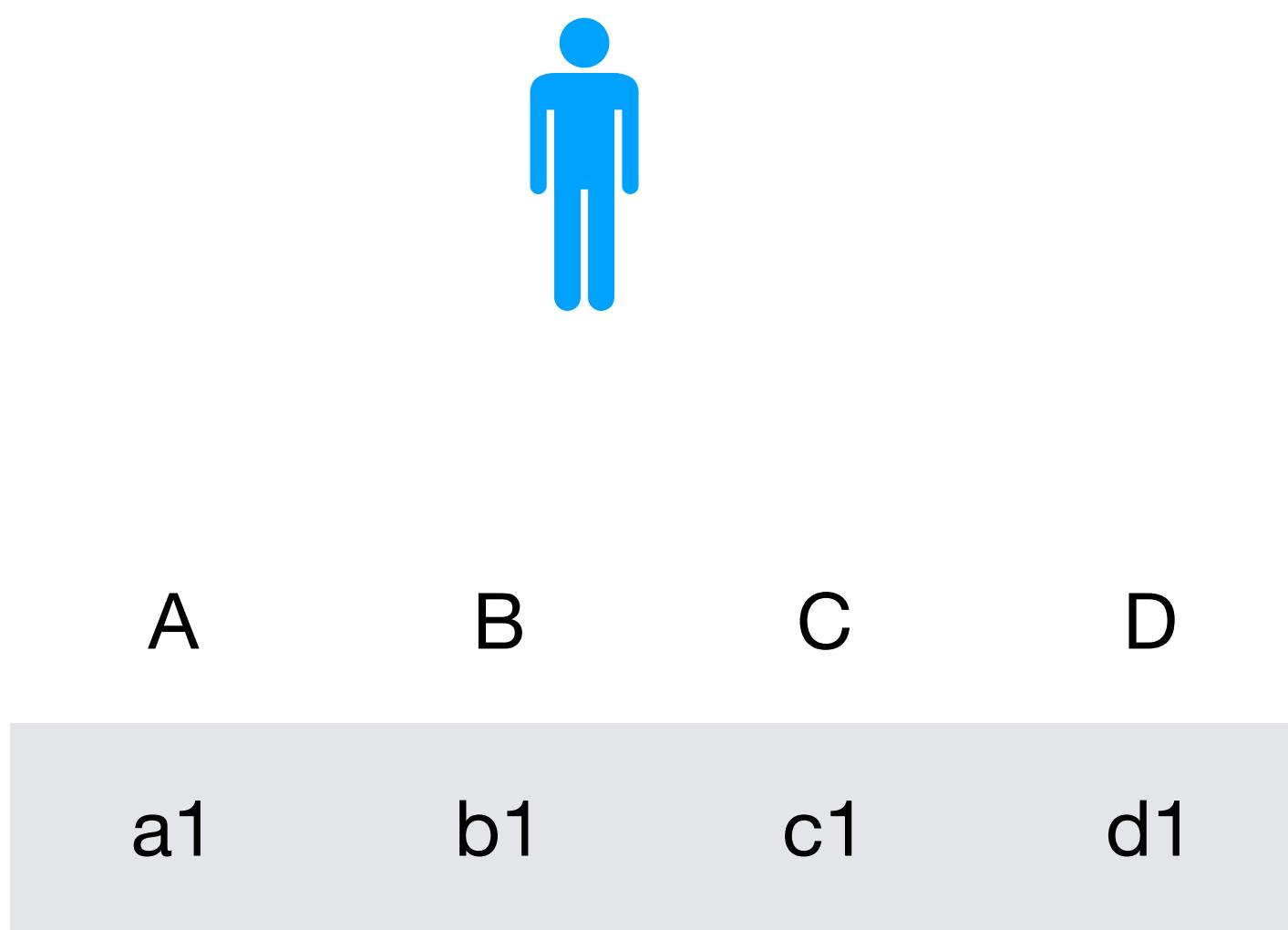
Why do these biases appear?

Tainted Examples

- Negative examples
- e.g.: label taken as hiring decisions by manager as compared to capability of applicant

Why do these biases appear?

Limited Features

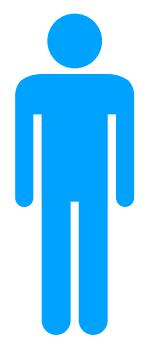


Why do these biases appear?

Limited Features



A	B	C	D
a1	b1	c1	d1

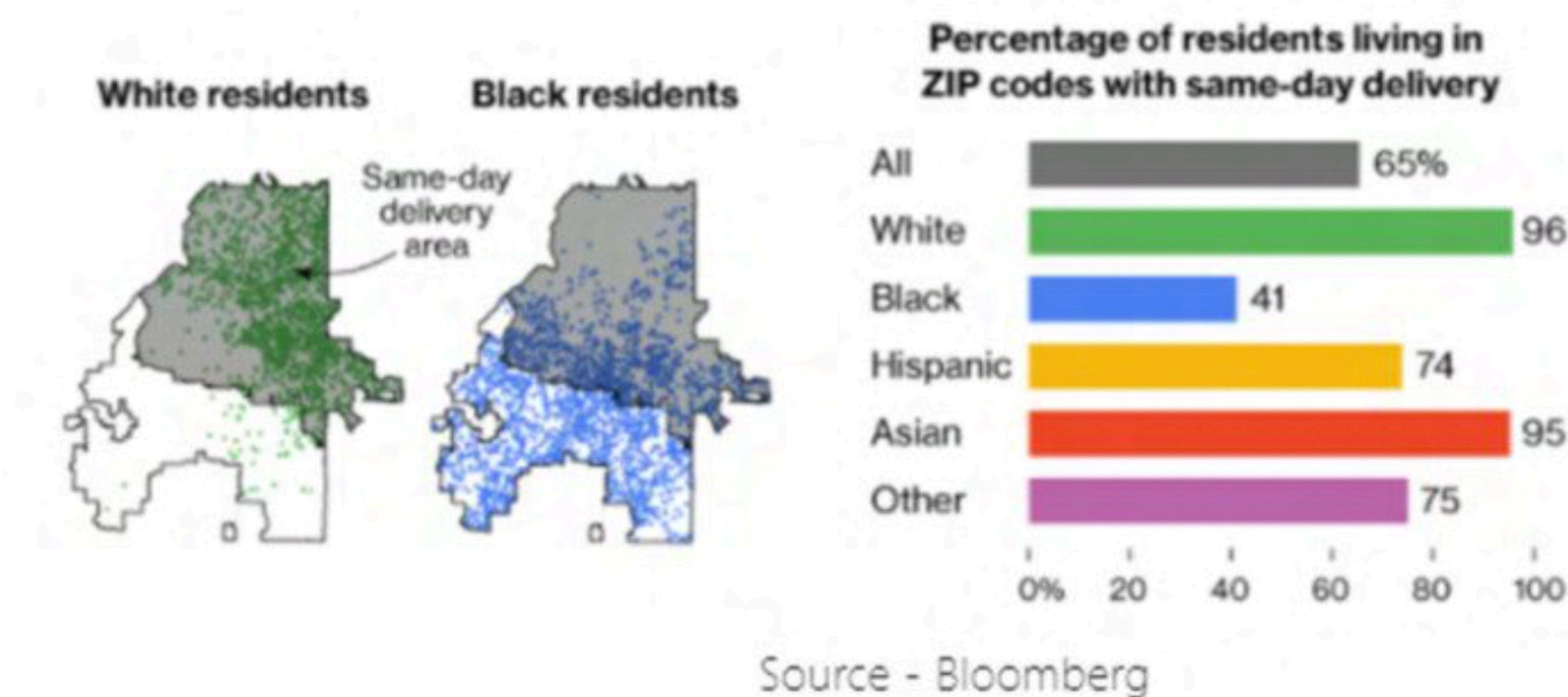


Minority/marginalized community
E.g. POC or women in medical datasets

A	B	C	D
-	-	-	d2

Why do these biases appear?

Proxies



COMPAS

	WHITE	AFRICAN AMERICAN
Labeled Higher Risk, But Didn't Re-Offend	23.5%	44.9%
Labeled Lower Risk, Yet Did Re-Offend	47.7%	28.0%

Overall, Northpointe's assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes. (Source: ProPublica analysis of data from Broward County, Fla.)

Harms

Representational Harm

- Stereotypes
- Negative generalizations
- Misrepresentation of distribution of different social groups

Allocational Harm

- Unfair distribution of resources
- E.g. Resume sorting for jobs, credit and crime profiling



Exercise

Task: Pain level detection



A	B	C	D
a1	b1	c1	d1



Minority/marginalized community
E.g. POC or women in medical datasets

A	B	C	D
-	-	-	d2



Predicts lower than correct

Definitions of Fairness

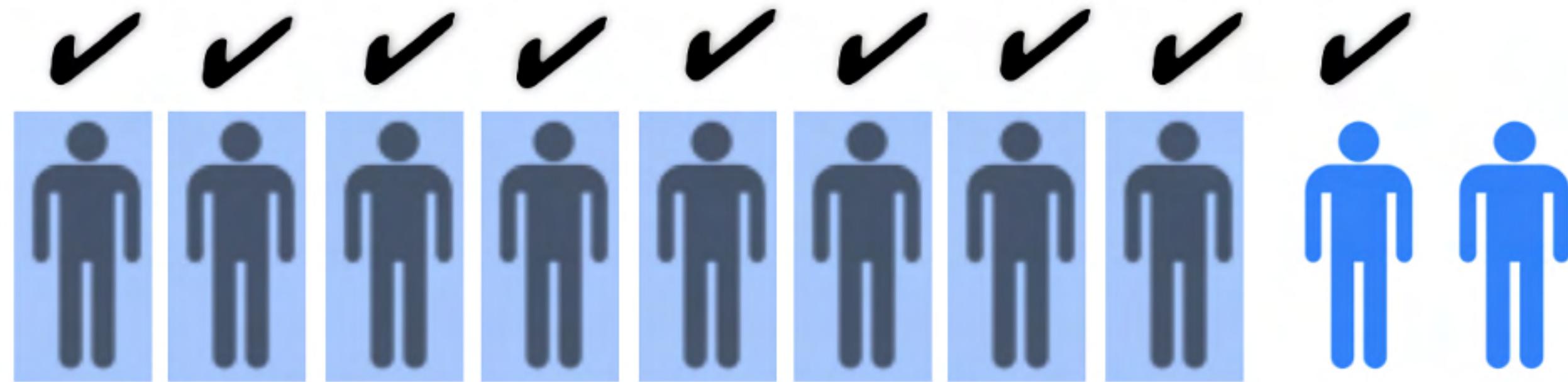
Individual Fairness

Search query	Work experience	Education experience	Profile views	Candidate	Xing ranking
Brand Strategist	146	57	12992	male	1
Brand Strategist	327	0	4715	female	2
Brand Strategist	502	74	6978	male	3
Brand Strategist	444	56	1504	female	4
Brand Strategist	139	25	63	male	5
Brand Strategist	110	65	3479	female	6
Brand Strategist	12	73	846	male	7
Brand Strategist	99	41	3019	male	8
Brand Strategist	42	51	1359	female	9
Brand Strategist	220	102	17186	female	10

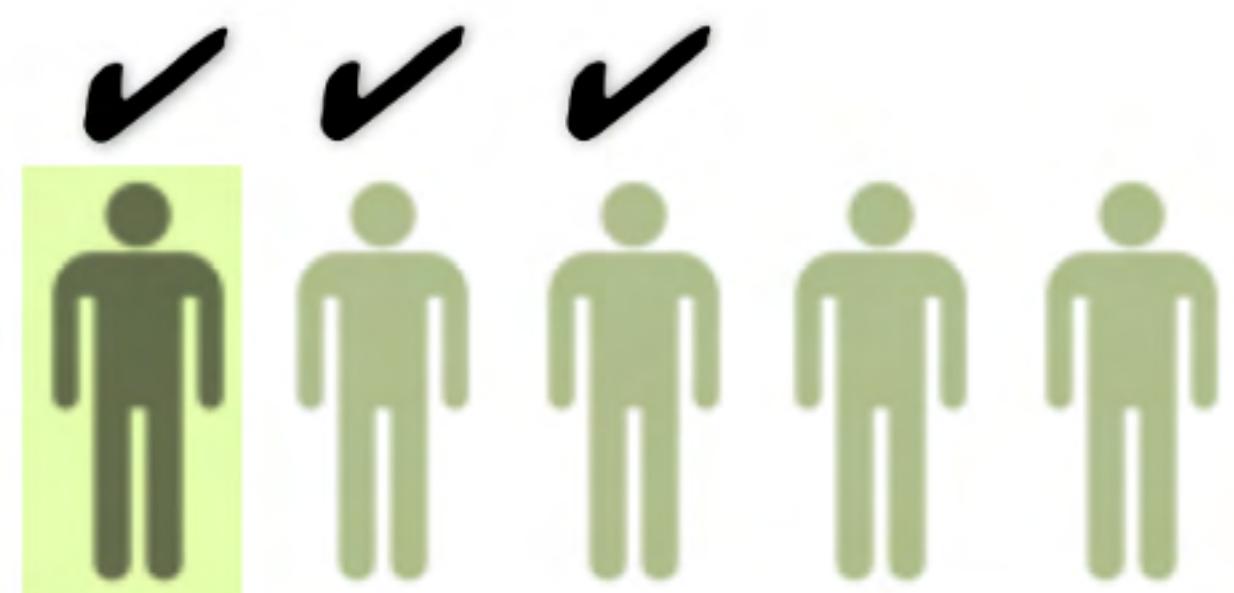
Similar individuals receiving similar outcomes or treatment.

Definitions of Fairness

Group Fairness



$$A = 1$$

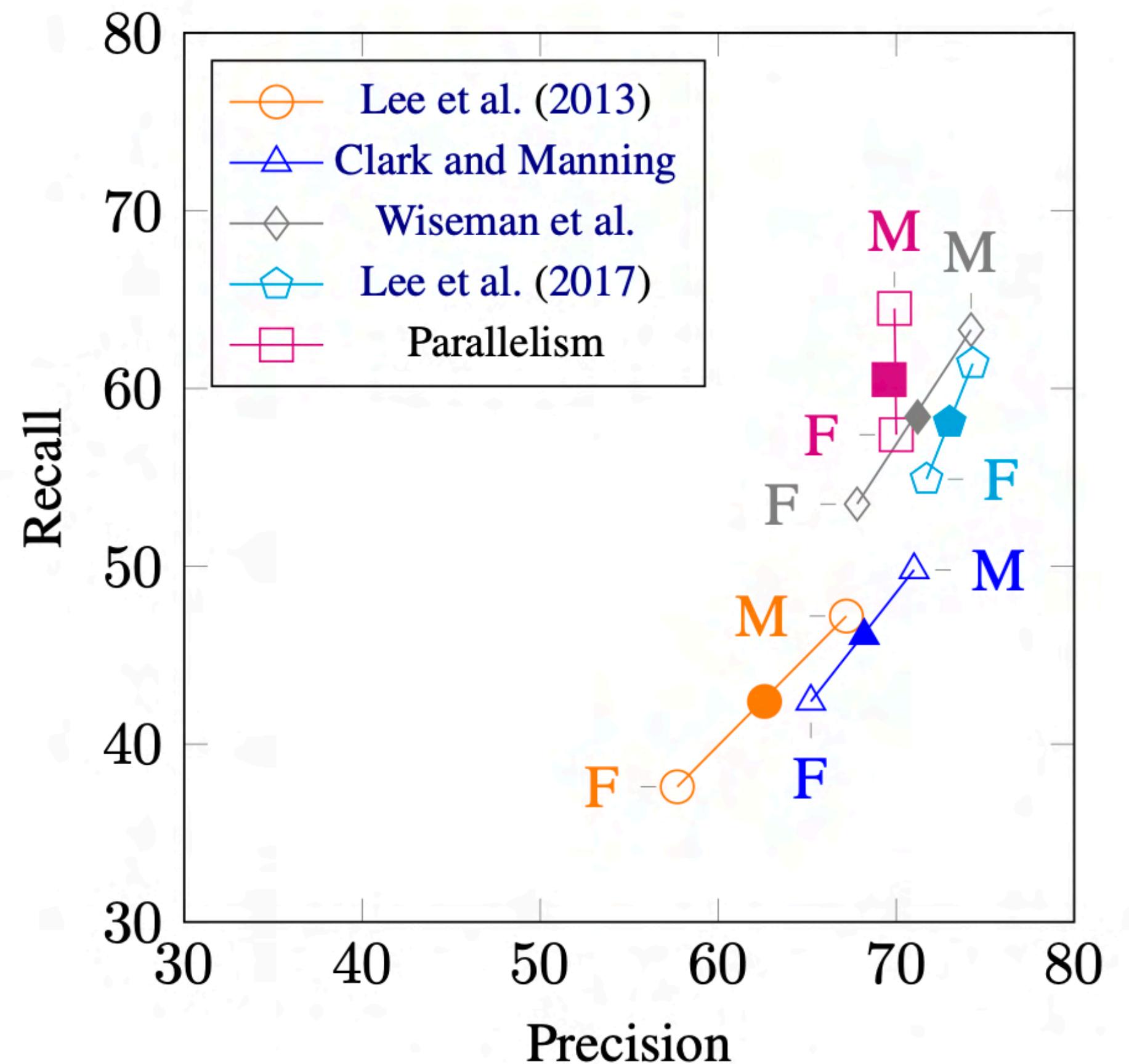


$$A = 0$$

Groups defined by protected attributes receiving similar outcomes or treatment.

Definitions of Fairness

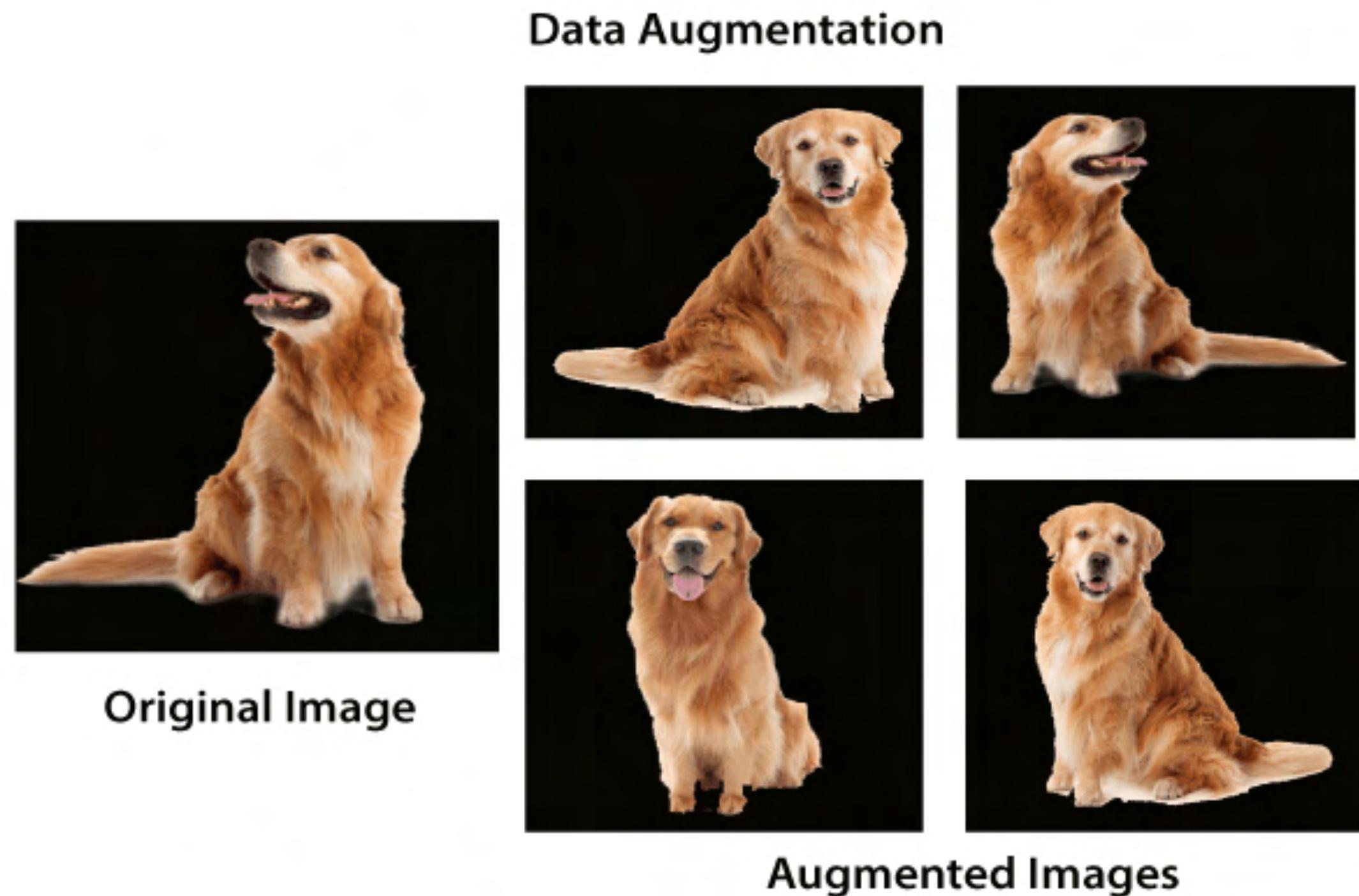
Group Fairness



Precision-Recall on the GAP development data set—Overall (solid markers), Masculine, Feminine—for off-the-shelf resolvers and Parallelism.

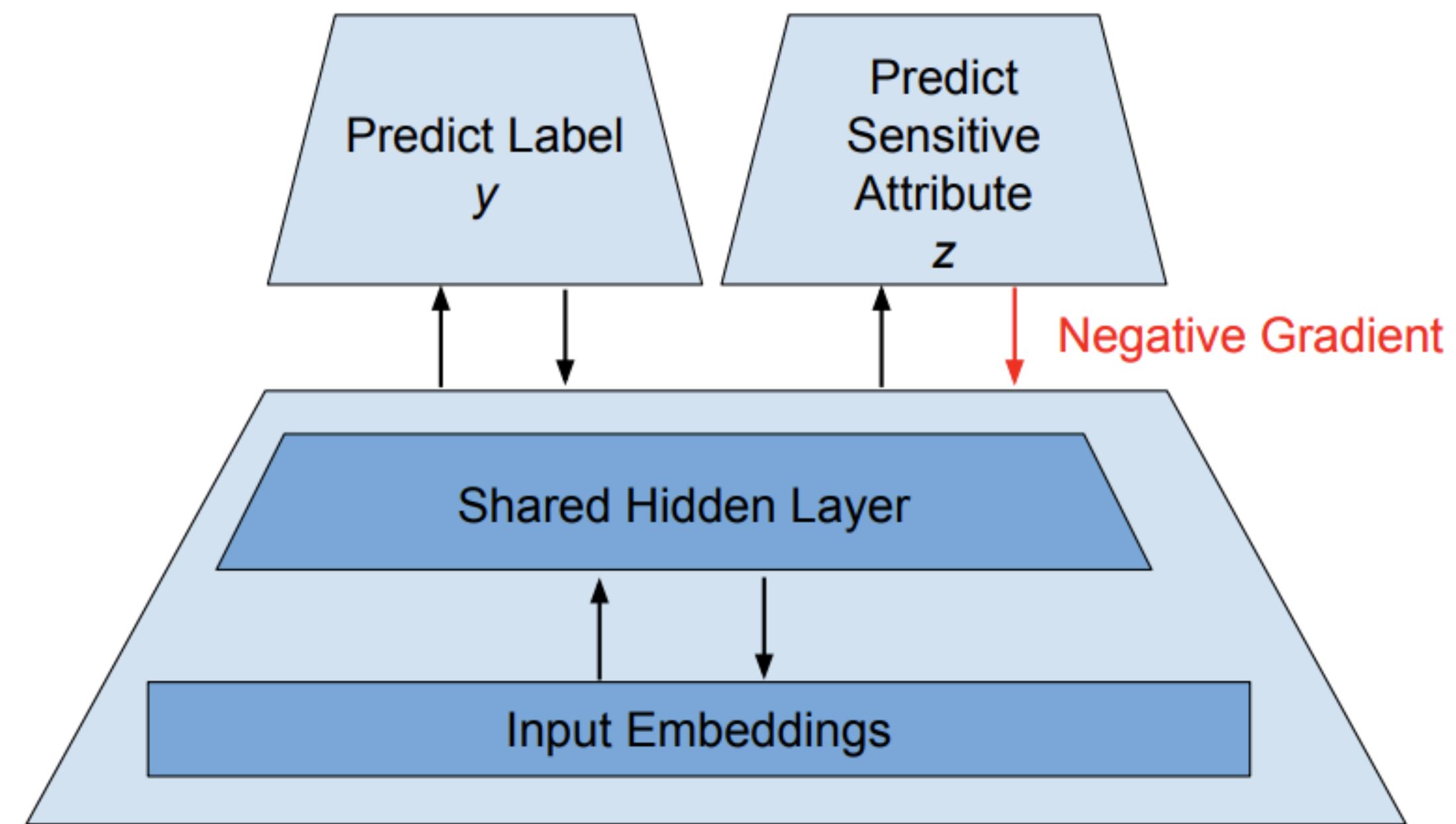
Fairness in the Pipeline

Pre-processing



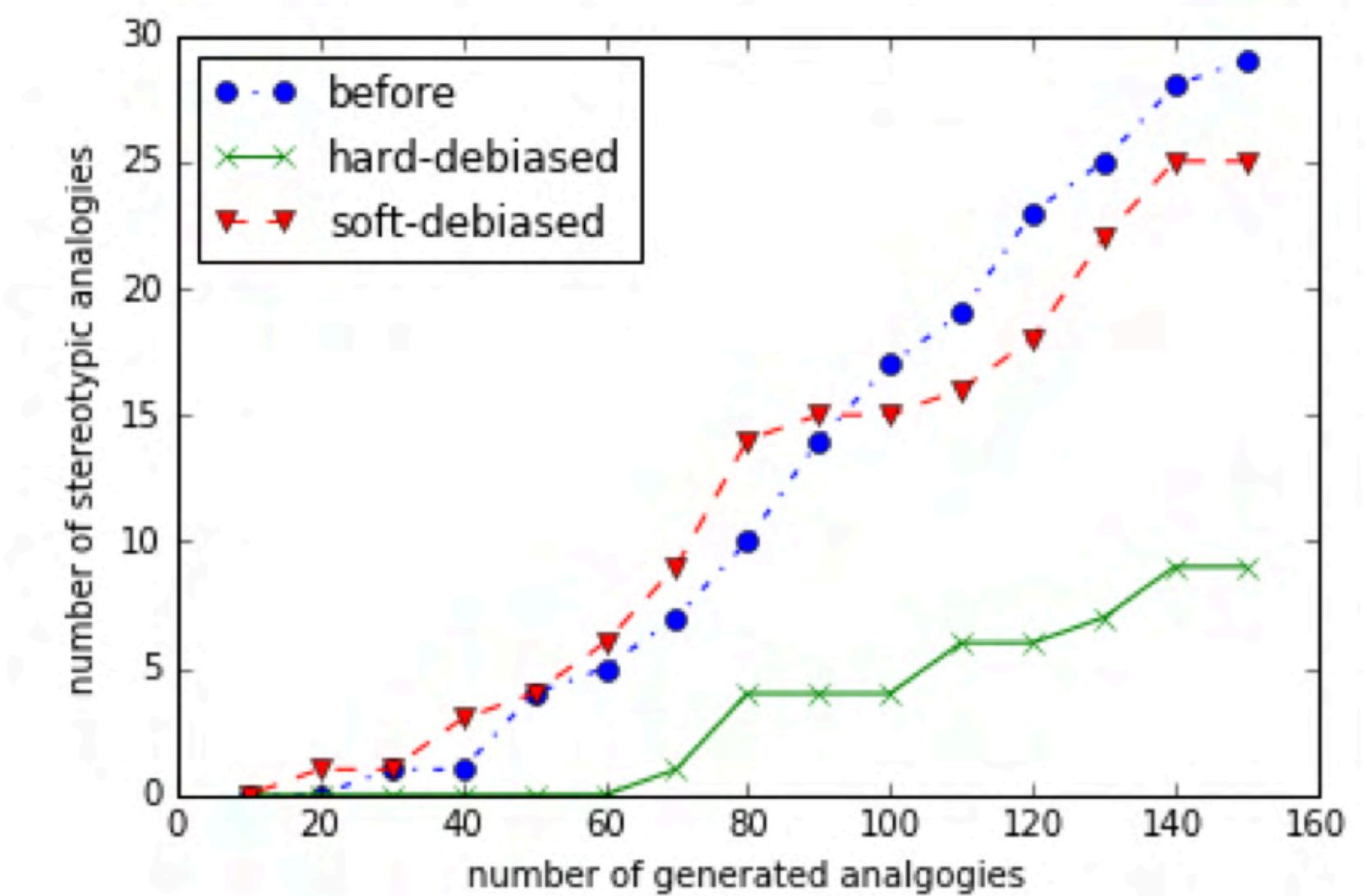
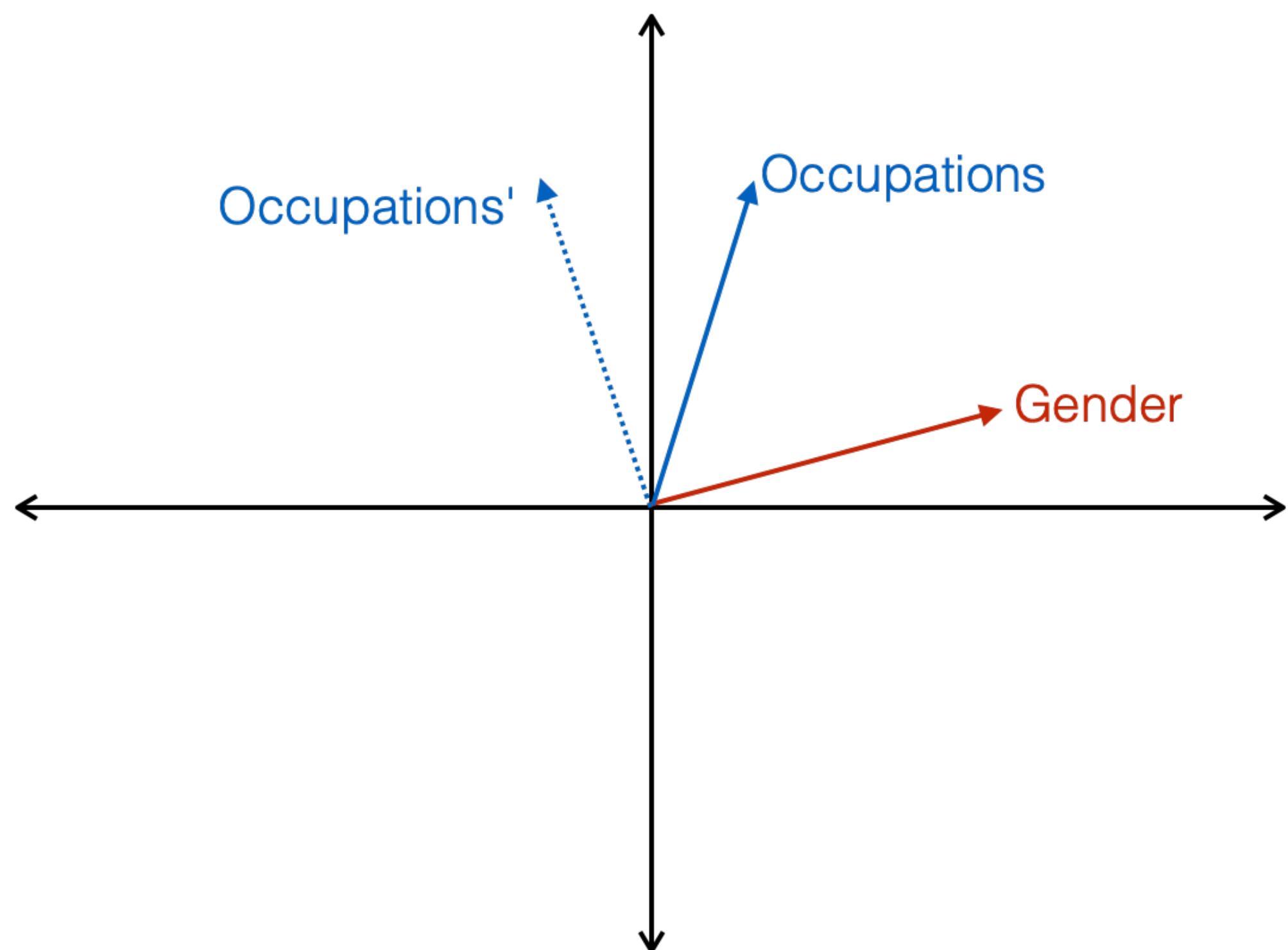
Fairness in the Pipeline

Model changes



Fairness in the Pipeline

Post-processing

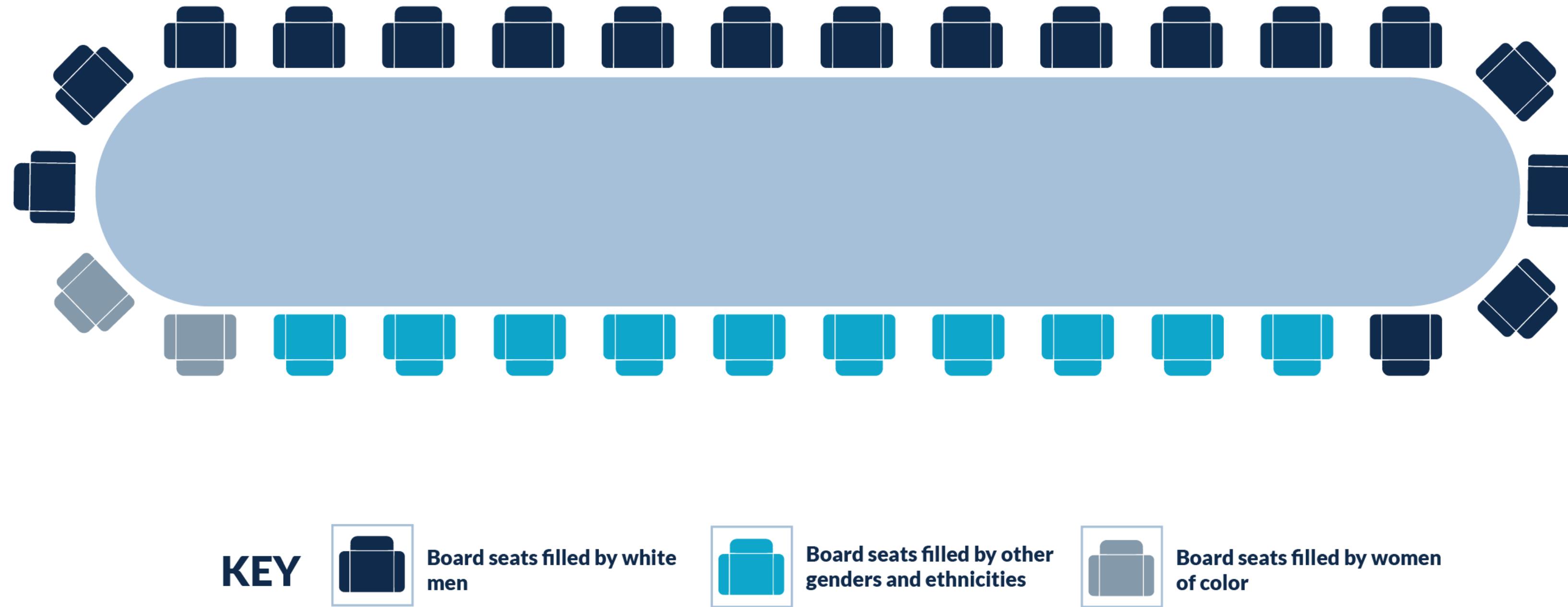


Biases are Complex!

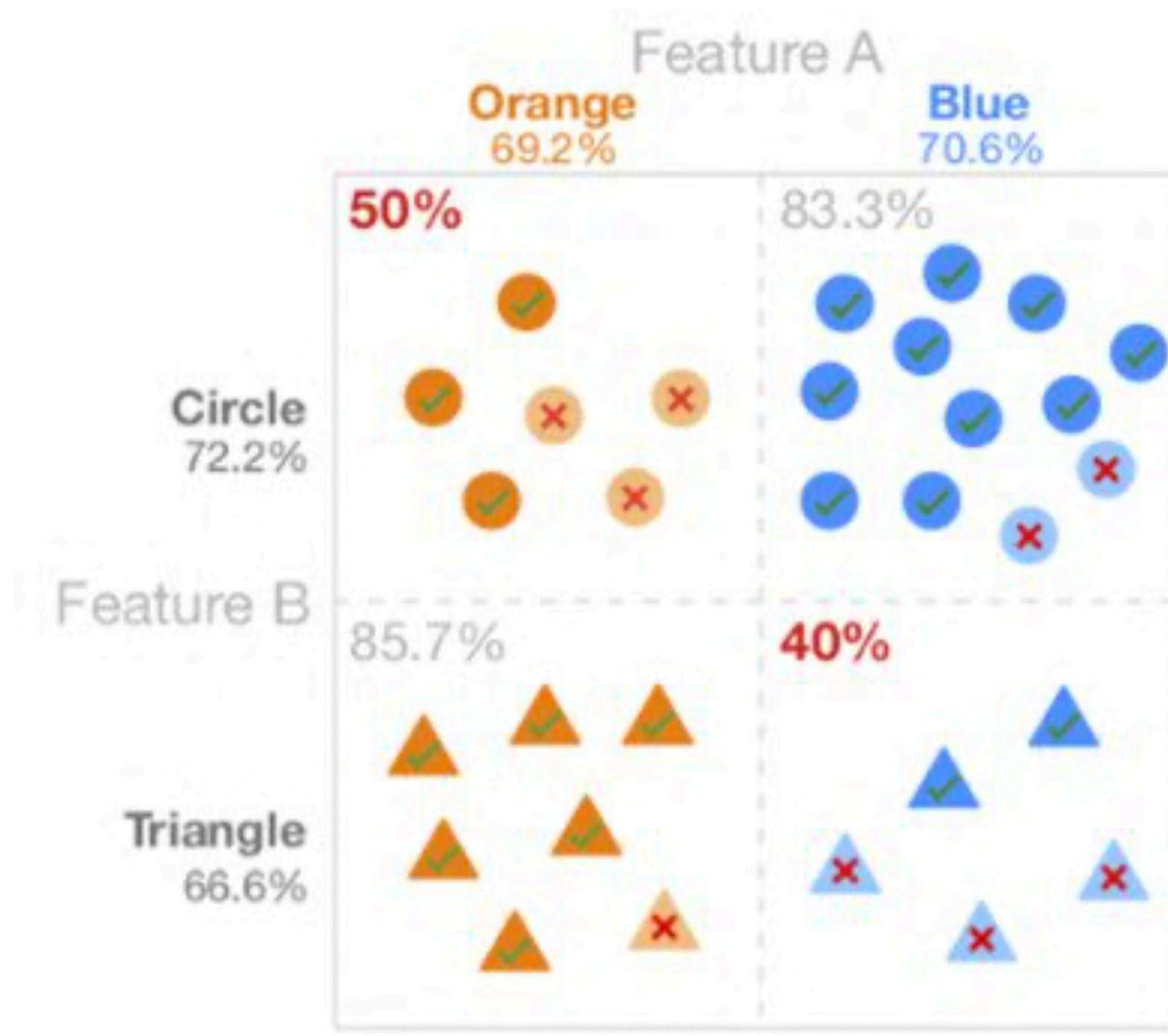
- Intersectional: more than a compounding effect; not linearly separable
- Non-Discrete Categories: fluid, mutable, non-bucketable demographics
- Erasure: bias caused by cyclic exclusion
- Biases vary across cultures!

Intersectional Bias

As of 2018, less than 6% of all Fortune 100 board seats are filled by women of color



Intersectional Bias



Demographic Fluidity



Lecture 17: Bayesian Learning Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Bayesian Learning

Lec 17: Bayesian Learning

Bayes Theorem



Posterior probability: What is the probability of Y given that X is observed?

Likelihood: What is the likelihood of observing X given a specific Y?

Prior probability: What is our belief in Y before we see X?

Bayes Theorem

(X, Y) can be (Data, Model), or (Input, Output)



Posterior probability: What is the probability of Y given that X is observed?

Likelihood: What is the likelihood of observing X given a specific Y?

Prior probability: What is our belief in Y before we see X?

$$\forall x, y \quad P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)}$$
$$= \frac{P(X = x|Y = y)P(Y = y)}{\sum_{y'} P(X = x|Y = y')P(Y = y')}$$

Probabilistic Learning

Two different notions of probabilistic learning

- ❖ **Bayesian Learning:** Use of a probabilistic criterion in selecting a hypothesis ($P(\Theta|D)$)
 - ❖ The hypothesis can be deterministic, a Boolean function
 - ❖ The criterion for selecting the hypothesis is probabilistic
- ❖ **Learning probabilistic concepts** ($P(Y|X)$)
 - ❖ The learned concept is a function $c:X \rightarrow [0,1]$
 - ❖ $c(x)$ may be interpreted as the probability that the label 1 is assigned to x

Bayesian Learning

Given a dataset D, we want to find the best hypothesis h
What does $P(h|D)$ mean?



Posterior probability: What is the probability that h is the hypothesis, given that the data D is observed?

Likelihood: What is the probability that this data point (an example or an entire dataset) is observed, given that the hypothesis is h?

Prior probability of h: Background knowledge. What do we expect the hypothesis to be even before we see any data? For example, in the absence of any information, maybe the uniform distribution.

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(h|D)$$

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$\begin{aligned} h_{MAP} &= \arg \max_{h \in H} P(h|D) \\ &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

Choosing a hypothesis

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

If we assume that the prior is uniform i.e. $P(h_i) = P(h_j)$, for all h_i, h_j

- ❖ Simplify this to get the Maximum Likelihood hypothesis

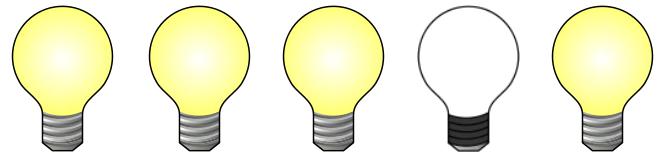
$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

Often computationally easier to maximize *log likelihood*

Bernoulli trials

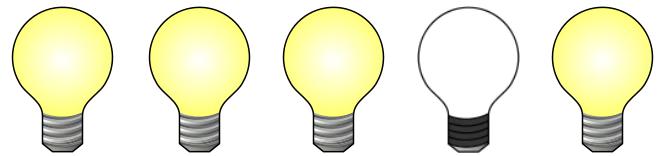
- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$

- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed



Bernoulli trials

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



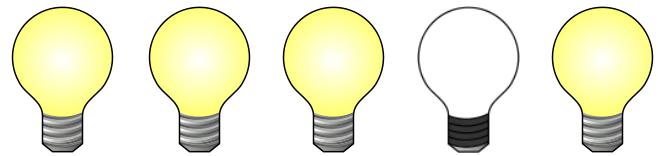
- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

Bernoulli trials

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

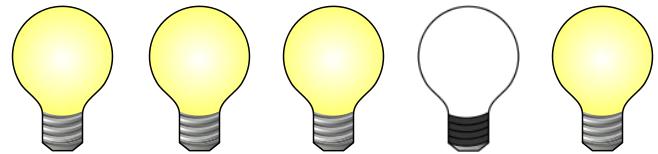
- ❖ The most likely value of p for this observation is?

$$\underset{p}{\operatorname{argmax}} P(D|p) = \underset{p}{\operatorname{argmax}} \binom{100}{80} p^{80} (1-p)^{20}$$

Faulty lightbulbs

The experiment:

Try out 100 lightbulbs
80 work, 20 don't



You: The probability is $P(\text{failure}) = 0.2$

CEO: But how do you know?

You: Because...

CEO: Okay, but you only test 100 lightbulbs, can you calibrate your results based on our prior tests?

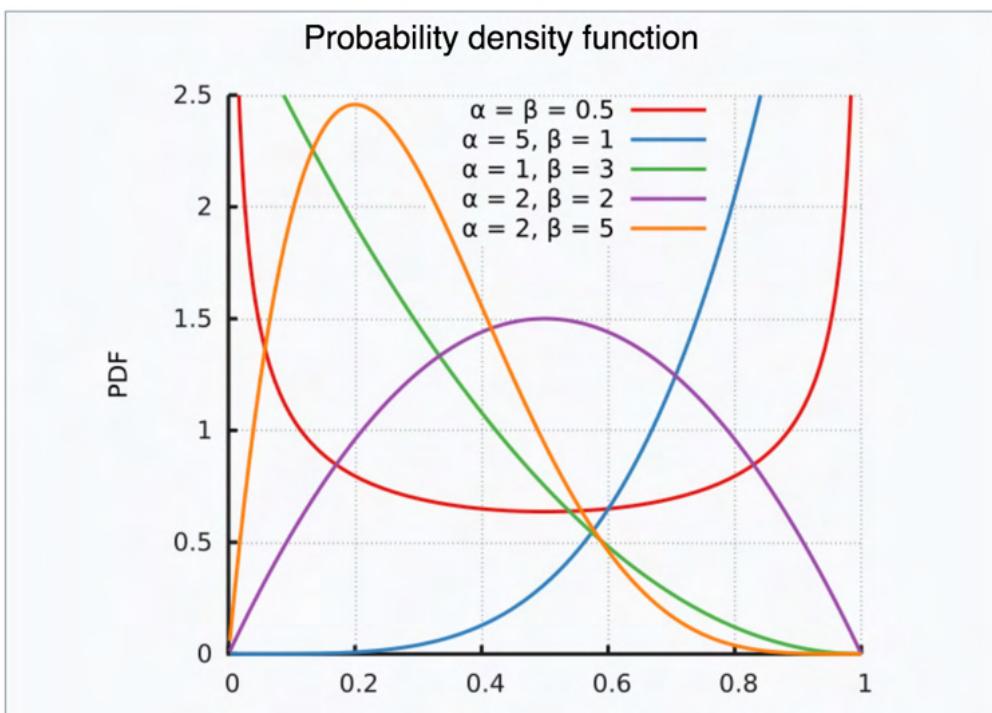
MAP estimation

Given some data, find the most probable hypothesis

- ❖ The Maximum a Posteriori hypothesis h_{MAP}

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

Beta distribution

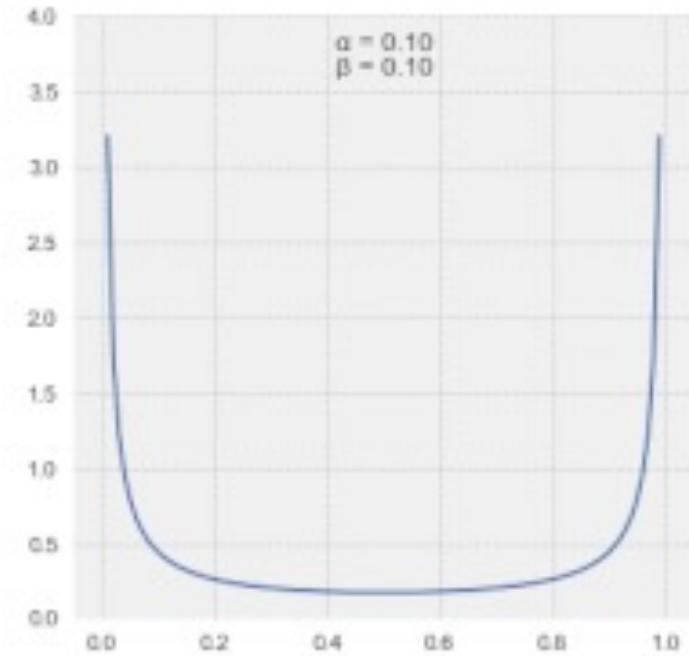
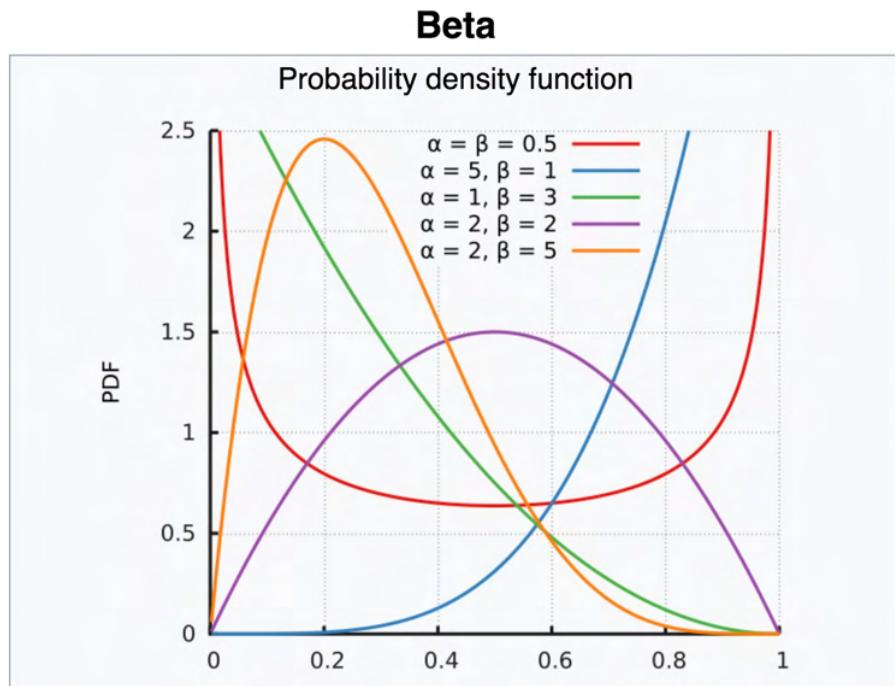


$$\begin{aligned} f(x; \alpha, \beta) &= \text{constant} \cdot x^{\alpha-1} (1-x)^{\beta-1} \\ &= \frac{x^{\alpha-1} (1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1} (1-u)^{\beta-1} du} \\ &= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1} \\ &= \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1} \end{aligned}$$
$$\alpha > 0, \beta > 0$$

https://en.wikipedia.org/wiki/Beta_distribution

Prior distribution

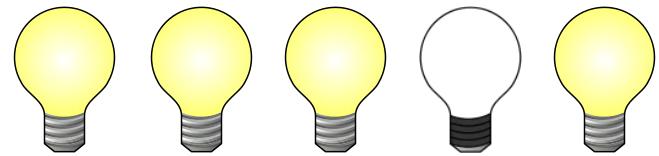
- ❖ The boss has a prior belief of the distribution of faulty lightbulb



MAP for Bernoulli trials

p is the parameter for hypothesis

- ❖ $P(\text{success}) = p, P(\text{failure}) = 1 - p$



- ❖ Each trial is i.i.d
 - ❖ Independent and identically distributed

- ❖ You have seen $D = \{80 \text{ work, } 20 \text{ don't}\}$

$$P(D|p) = \binom{100}{80} p^{80} (1-p)^{20}$$

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

MAP estimation

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

$$P(D|p) = \binom{a+b}{a} p^a (1-p)^b$$

$$P(p) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1-p)^{\beta-1}$$

$$\begin{aligned} p_{best} &= \operatorname{argmax}_p P(D|h) P(h) \\ &= \operatorname{argmax}_p \log P(D | h) + \log P(h) \\ &= \operatorname{argmax}_p \log \left(\frac{\binom{a+b}{a}}{B(\alpha, \beta)} p^a (1-p)^b p^{\alpha-1} (1-p)^{\beta-1} \right) \\ &= \operatorname{argmax}_p (a + \alpha - 1) \log p + (b + \beta - 1) \log(1 - p) \end{aligned}$$

MAP v.s. MLE

- ❖ MLE:

$$\operatorname{argmax}_p a \log p + b \log(1 - p)$$

$$\Rightarrow p_{best} = \frac{a}{a + b}$$

- ❖ MAP (w/ Beta distribution as prior)

$$\operatorname{argmax}_p (a + \alpha - 1) \log p + (b + \beta - 1) \log(1 - p)$$

$$\Rightarrow p_{best} = \frac{a + \alpha - 1}{a + b + \alpha + \beta - 2}$$

MAP v.s. MLE

❖ MAP

$$\operatorname{argmax}_p (a + \alpha - 1) \log p + (b + \beta - 1) \log(1 - p)$$

$$\Rightarrow p_{best} = \frac{a + \alpha - 1}{a + b + \alpha + \beta - 2}$$

❖ Let $\alpha = 100, \beta = 10$

❖ $a = 10, b = 20 \Rightarrow p_{best} \approx 0.79$

❖ $a = 1000, b = 2000 \Rightarrow p_{best} \approx 0.36$

❖ $a = 100,000, b = 200,000 \Rightarrow p_{best} \approx 0.33$

MAP for logistic regression

Let's get back to the MLE for logistic regression

- ❖ Training data
 - ❖ $S = \{(x_i, y_i)\}$, m examples
- ❖ What we want
 - ❖ Find a w such that $P(S | w)$ is maximized
 - ❖ We know that our examples are drawn independently and are identically distributed (i.i.d)
 - ❖ How do we proceed?

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

The usual trick: Convert products to sums by taking log

Recall that this works only because log is an increasing function and the maximizer will not change

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

But (by definition) we know that

$$P(y|\mathbf{w}, \mathbf{x}) = \sigma(y_i \mathbf{w}^T \mathbf{x}_i) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

$$P(y|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

Maximum likelihood estimation

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w}) = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$P(y|\mathbf{w}, \mathbf{x}) = \frac{1}{1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)}$$

Maximum likelihood estimation

$\underset{\mathbf{w}}{\operatorname{argmax}}$

$$\underset{\mathbf{w}}{\operatorname{argmax}} P(S|\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

The goal: Maximum likelihood training of a discriminative probabilistic classifier under the logistic model for the posterior distribution.

$$\max_{\mathbf{w}} \sum_i^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

Equivalent to: Training a linear classifier by minimizing the *logistic loss*.

Maximum a posteriori estimation

We could also add a prior on the weights

Suppose each weight in the weight vector is drawn independently from the normal distribution with zero mean and standard deviation σ

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_i) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_i^2}{\sigma^2}\right)$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Let us work through this procedure again to see what changes

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Let us work through this procedure again to see what changes

What is the goal of MAP estimation? (In maximum likelihood, we maximized the likelihood of the data)

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

What is the goal of MAP estimation? (In maximum likelihood, we maximized the likelihood of the data)

To maximize the posterior probability of the model given the data (i.e. to find the most probable model, given the data)

$$P(\mathbf{w}|S) \propto P(S|\mathbf{w})P(\mathbf{w})$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|S) = \operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

We have already expanded out the first term.

$$\sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

Expand the log prior

$$\sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \sum_{j=1}^d \frac{-w_j^2}{\sigma^2} + \text{constants}$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \sum_{j=1}^d \frac{-w_j^2}{\sigma^2} + \text{constants}$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) - \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w}$$

MAP estimation for logistic regression

Maximum likelihood estimation

$$\arg \max_{\mathbf{w}} P(S|\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^m P(y_i|\mathbf{x}_i, \mathbf{w})$$

$$\max_{\mathbf{w}} \sum_{i=1}^m \log P(y_i|\mathbf{x}_i, \mathbf{w})$$

Equivalent to solving

$$\max_{\mathbf{w}} \sum_{i=1}^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$$

$$p(\mathbf{w}) = \prod_{j=1}^d p(w_j) = \prod_{j=1}^d \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-w_j^2}{\sigma^2}\right)$$

Learning by solving

$$\operatorname{argmax}_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w})$$

Take log to simplify

$$\max_{\mathbf{w}} \log P(S|\mathbf{w}) + \log P(\mathbf{w})$$

$$\max_{\mathbf{w}} \sum_i^m -\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) - \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w}$$

Maximizing a negative function is the same as minimizing the function
Lec 17: Bayesian Learning

Learning a logistic regression classifier

Learning a logistic regression classifier is equivalent to solving

$$\min_{\mathbf{w}} \sum_i^m \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \frac{1}{\sigma^2} \mathbf{w}^T \mathbf{w}$$

Empirical Loss Minimization

Batch Learning Setting

- ❖ Learning as loss minimization
- 1. Collect training data $\widehat{D} = \{(x, y)\}$
- 2. Pick a hypothesis class
 - ❖ E.g., linear classifiers, deep neural networks
- 3. Choose a **loss function**
 - ❖ negative log-likelihood
 - ❖ We can impose a preference (i.e., prior) over hypotheses, e.g., simpler is better
- 4. Minimize the **empirical loss**
 - ❖ SGD, coordinate descent, Newton methods, LBFGS

Batch learning setup

- ❖ Data drawn from a fixed, unknown distribution \mathcal{D}
- ❖ A hidden oracle classifier f^* , $y = f(x)$
- ❖ We wish to find a hypothesis $h \in H$ that mimics f^*
- ❖ We define a loss function $L(h(x), f^*(x))$ that penalizes mistakes
- ❖ What is the ideal f ?

$$\arg \min_{h \in H} E_{x \sim D} [L(h(x), f(x))] \\ \text{expected loss}$$

Batch learning setup

- ❖ Data drawn from a fixed, unknown distribution \mathcal{D}
- ❖ A hidden oracle classifier f , $y = f(x)$
- ❖ We wish to find a hypothesis $h \in H$ that mimics f^*
- ❖ We define a loss function $L(h(x), f(x))$ that penalizes mistakes
- ❖ What is the ideal h ?

Let's define

$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y \neq y' \\ 0 & \text{if } y = y' \end{cases}$$

$$\min_{h \in H} E_{x \sim D} [L_{0-1}(h(x), f(x))] = \min_{f \in H} E_{x \sim D} [\# \text{mistakes}]$$

This is difficult to optimize!

How can we learn f from \widehat{D}

- ❖ We don't know D , we only see samples in
$$\widehat{D} = \{(x, y)\}$$
- ❖ \widehat{D} is training data (i.i.d. sample from D)
- ❖ Instead of $\min_{h \in H} E_{x \sim D} [L_{0-1}(h(x), f(x))]$
- ❖ we minimize the **empirical loss**

$$\min_{h \in H} \frac{1}{|\widehat{D}|} \sum_{(x, y) \in \widehat{D}} [L(h(x), f(x))]$$

Empirical loss minimization

Learning = minimize *empirical loss* on the training set

$$\min_{h \in H} \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

Is there a problem here?

Empirical loss minimization

Learning = minimize *empirical loss* on the training set

$$\min_{h \in H} \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

Is there a problem here?

Overfitting!

We need something that biases the learner towards simpler hypotheses

- ❖ Achieved using a **regularizer**, which penalizes complex hypotheses

$$\min_{h \in H} R(h) + \frac{1}{m} \sum_i [L(h(x_i), f(x_i))]$$

Regularized loss minimization

❖ Learning: $\min_{h \in H} R(h) + \frac{1}{m} \sum_i [L(h(x_i), f(x_i))]$

❖ With linear classifiers:

$$H: \{h(x): \text{sgn}(w^T x \geq 0)\}, w \in R^d$$

$$\min_w \frac{1}{2} w^T w + C \sum_i L(y_i, \mathbf{x}_i, \mathbf{w})$$

- ❖ What is a **loss function**?
- ❖ Loss functions should penalize mistakes
 - ❖ We are minimizing average loss over the training data
- ❖ What is the ideal loss function for classification?

The 0-1 loss

Penalize classification mistakes between true label y and prediction y'

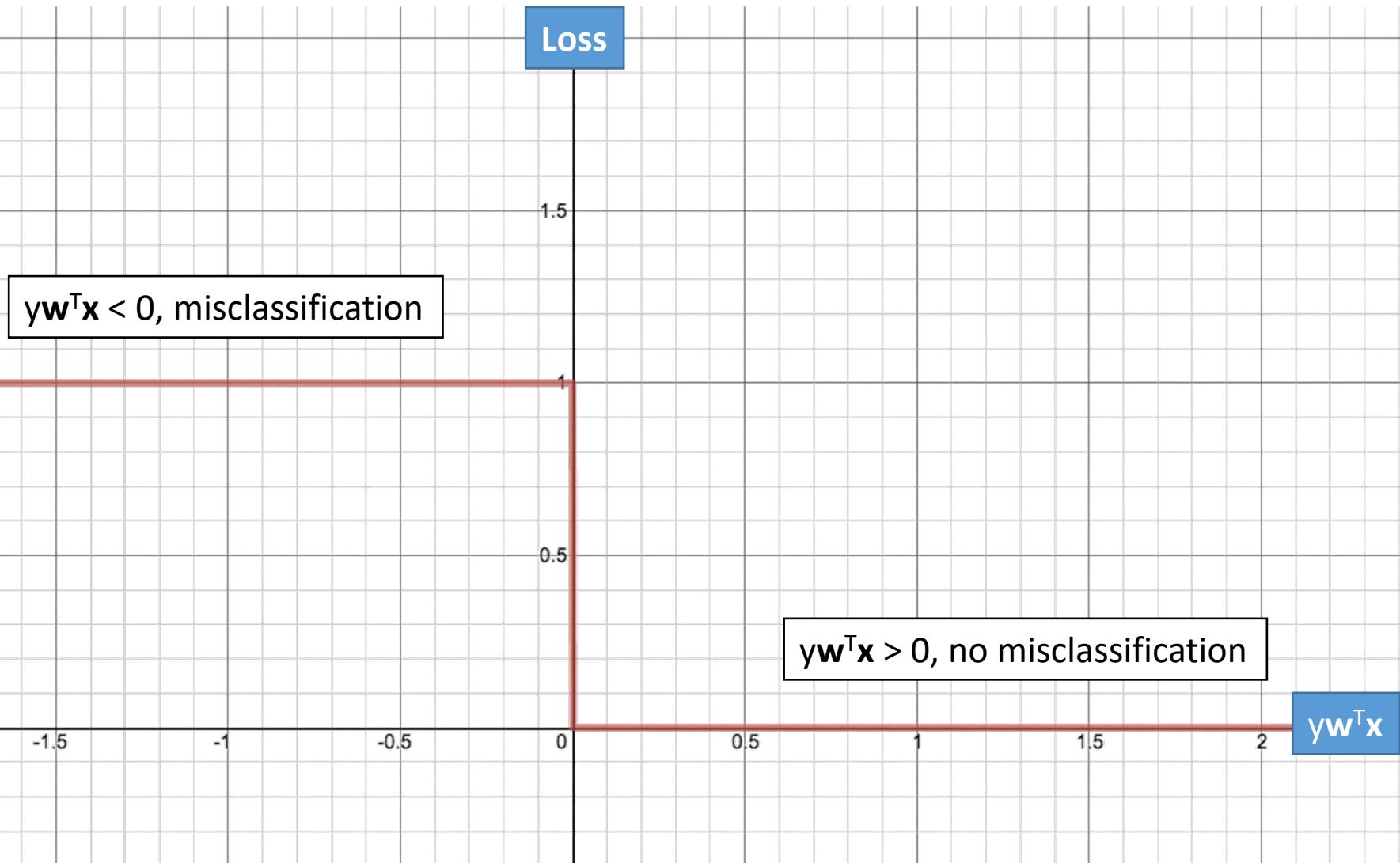
$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y \neq y', \\ 0 & \text{if } y = y'. \end{cases}$$

- ❖ For linear classifiers, the prediction is $\text{sgn}(\mathbf{w}^T \mathbf{x})$
 - ❖ Mistake if $y \mathbf{w}^T \mathbf{x} \leq 0$

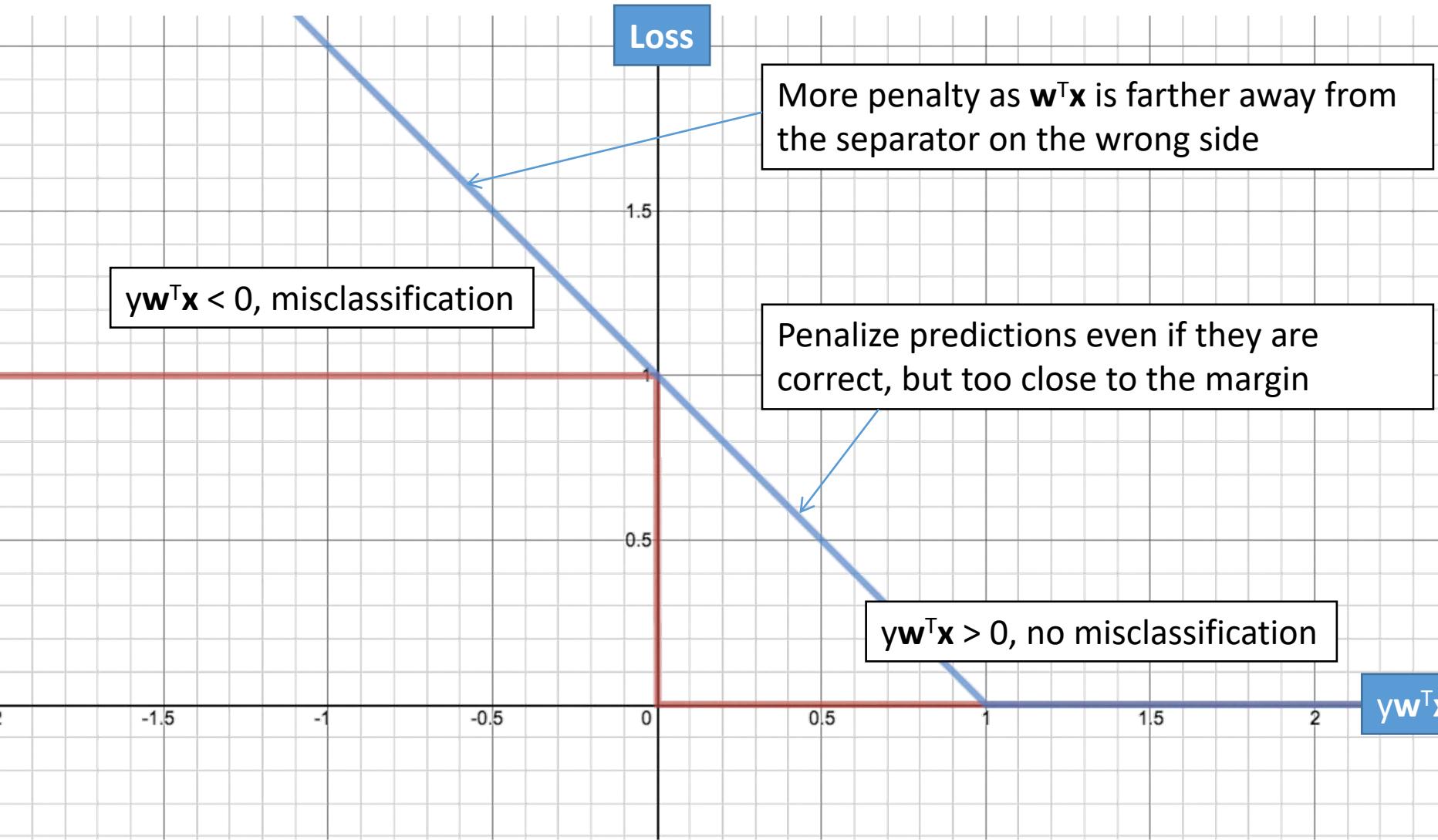
$$L_{0-1}(y, \mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } y \mathbf{w}^T \mathbf{x} \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Minimizing 0-1 loss is intractable. Need surrogates

The 0-1 loss



Compare to the hinge loss



$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo

Many loss functions exist

- ❖ Perceptron loss

$$L_{\text{Perceptron}}(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\mathbf{w}^T \mathbf{x})$$

- ❖ Logistic loss (logistic regression)

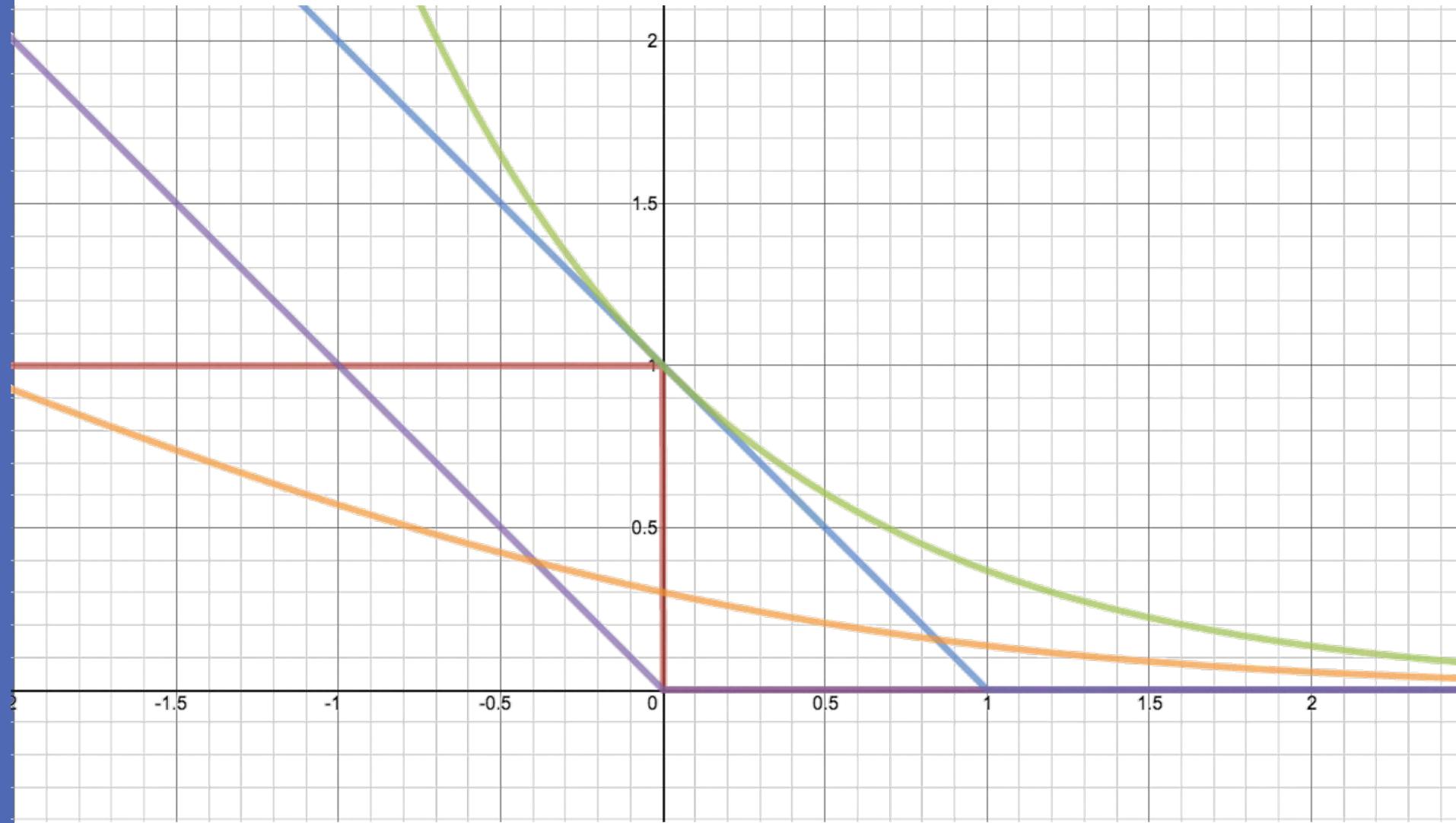
$$L_{\text{Logistic}}(y, \mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\mathbf{w}^T \mathbf{x}})$$

- ❖ Hinge loss (SVM)

$$L_{\text{Hinge}}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T \mathbf{x})$$

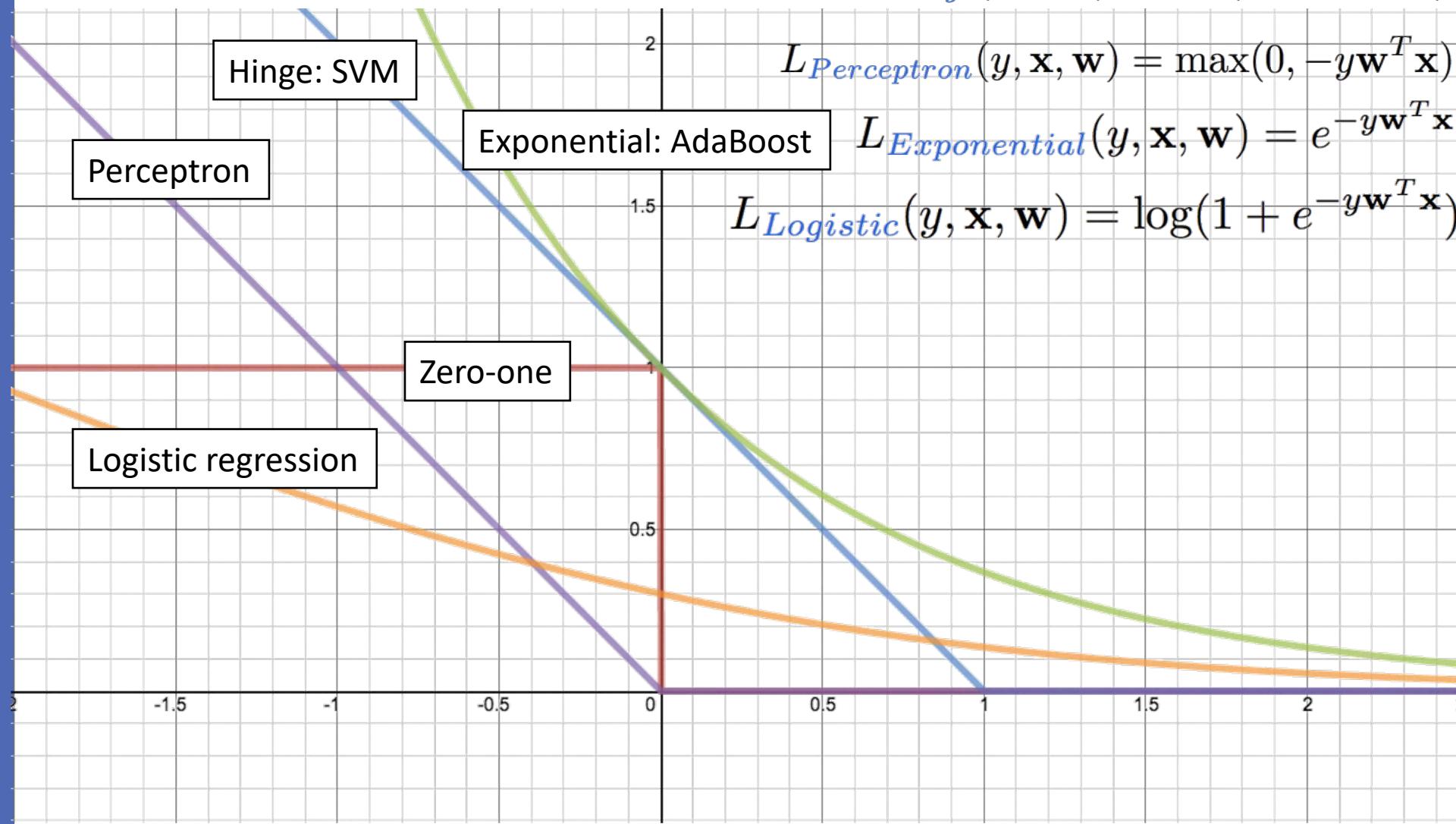
$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo



$$\min_{h \in H} \text{regularizer}(h) + C \frac{1}{m} \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

The loss function zoo



Many choices of $R(w)$

- ❖ Minimizing the empirical loss with linear function

$$\min_{w \in R^d} R(\mathbf{w}) + \frac{1}{|\widehat{D}|} \sum_{(x,y) \in \widehat{D}} [L(\mathbf{x}, \mathbf{w}, y)]$$

- ❖ Prefer simpler model: (how?)

- ❖ Sparse:

$R(\mathbf{w}) = \#\text{non-zero elements in } w$ (L0 regularizer)

$R(\mathbf{w}) = \sum_i |w_i|$ (L1 regularizer)

- ❖ Gaussian prior (large margin w/ hinge loss):

$R(\mathbf{w}) = \sum_i w_i^2 = \mathbf{w}^T \mathbf{w}$ (L2 regularizer)

Learning via Loss Minimization: Summary

- ❖ Learning via Loss Minimization
 - ❖ Write down a loss function
 - ❖ Minimize empirical loss
- ❖ Regularize to avoid overfitting
 - ❖ Neural networks use other strategies such as dropout
- ❖ Widely applicable, different loss functions and regularizes

Naïve Bayes

Lec 17: Bayesian Learning

Bayes Theorem



Posterior probability: What is the probability of Y given that X is observed?

Likelihood: What is the likelihood of observing X given a specific Y?

Prior probability: What is our belief in Y before we see X?

Probabilistic Learning

Two different notions of probabilistic learning

- ❖ **Bayesian Learning:** Use of a probabilistic criterion in selecting a hypothesis ($P(\Theta|D)$)
 - ❖ The hypothesis can be deterministic, a Boolean function
 - ❖ The criterion for selecting the hypothesis is probabilistic
- ❖ **Learning probabilistic concepts** ($P(Y|X)$)
 - ❖ The learned concept is a function $c:X \rightarrow [0,1]$
 - ❖ $c(x)$ may be interpreted as the probability that the label 1 is assigned to x

MAP prediction

Let's use the Bayes rule for predicting y given an input \mathbf{x}

$$P(Y = y|X = \mathbf{x}) = \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

Posterior probability of label being y for this input \mathbf{x}

MAP prediction

Let's use the Bayes rule for predicting y given an input \mathbf{x}

$$P(Y = y|X = \mathbf{x}) = \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

Predict y for the input \mathbf{x} using

$$\arg \max_y \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

MAP prediction

Let's use the Bayes rule for predicting y given an input \mathbf{x}

$$P(Y = y|X = \mathbf{x}) = \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

Predict y for the input \mathbf{x} using

$$\arg \max_y P(X = \mathbf{x}|Y = y)P(Y = y)$$

MAP prediction

Don't confuse with *MAP learning*:

finds hypothesis by

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

Let's use the Bayes rule for predicting y given an input \mathbf{x}

$$P(Y = y|X = \mathbf{x}) = \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

Predict y for the input \mathbf{x} using

$$\arg \max_y P(X = \mathbf{x}|Y = y)P(Y = y)$$

MAP prediction

Predict y for the input \mathbf{x} using

$$\arg \max_y P(X = \mathbf{x}|Y = y) | P(Y = y)$$

Likelihood of observing this input \mathbf{x} when the label is y

Prior probability of the label being y

All we need are these two sets of probabilities

Example: Tennis

Input:

Temperature = Hot (H)

Wind = Weak (W)

Should I play tennis?

Temperature	Wind	$P(T, W \text{Tennis} = \text{Yes})$
Hot	Strong	0.15
Hot	Weak	0.4
Cold	Strong	0.1
Cold	Weak	0.35

Likelihood

Temperature	Wind	$P(T, W \text{Tennis} = \text{No})$
Hot	Strong	0.4
Hot	Weak	0.1
Cold	Strong	0.3
Cold	Weak	0.2

Example: Tennis

Prior	Play tennis	$P(\text{Play tennis})$
	Yes	0.3
	No	0.7

Without any other information,
what is the prior probability that I
should play tennis?

Example: Tennis

Prior	Play tennis	$P(\text{Play tennis})$
	Yes	0.3
	No	0.7

Without any other information, what is the prior probability that I should play tennis?

Temperature	Wind	$P(T, W \text{Tennis} = \text{Yes})$
Hot	Strong	0.15
Hot	Weak	0.4
Cold	Strong	0.1
Cold	Weak	0.35

On days that I do play tennis, what is the probability that the temperature is T and the wind is W?

Temperature	Wind	$P(T, W \text{Tennis} = \text{No})$
Hot	Strong	0.4
Hot	Weak	0.1
Cold	Strong	0.3
Cold	Weak	0.2

On days that I don't play tennis, what is the probability that the temperature is T and the wind is W?

Example: Tennis

Prior	Play tennis	$P(\text{Play tennis})$
	Yes	0.3
	No	0.7

Temperature	Wind	$P(T, W \text{Tennis} = \text{Yes})$
Hot	Strong	0.15
Hot	Weak	0.4
Cold	Strong	0.1
Cold	Weak	0.35

Temperature	Wind	$P(T, W \text{Tennis} = \text{No})$
Hot	Strong	0.4
Hot	Weak	0.1
Cold	Strong	0.3
Cold	Weak	0.2

Input:

Temperature = Hot (H)

Wind = Weak (W)

Should I play tennis?

Example: Tennis

Prior	Play tennis	$P(\text{Play tennis})$
	Yes	0.3
	No	0.7

Likelihood	Temperature	Wind	$P(T, W \text{Tennis} = \text{Yes})$
	Hot	Strong	0.15
	Hot	Weak	0.4
	Cold	Strong	0.1
	Cold	Weak	0.35

Likelihood	Temperature	Wind	$P(T, W \text{Tennis} = \text{No})$
	Hot	Strong	0.4
	Hot	Weak	0.1
	Cold	Strong	0.3
	Cold	Weak	0.2

Input:

Temperature = Hot (H)

Wind = Weak (W)

Should I play tennis?

$\text{argmax}_y P(H, W | \text{play?}) P(\text{play?})$

Example: Tennis

Prior	Play tennis	P(Play tennis)
	Yes	0.3
	No	0.7

Temperature	Wind	P(T, W Tennis = Yes)
Hot	Strong	0.15
Hot	Weak	0.4
Cold	Strong	0.1
Cold	Weak	0.35

Temperature	Wind	P(T, W Tennis = No)
Hot	Strong	0.4
Hot	Weak	0.1
Cold	Strong	0.3
Cold	Weak	0.2

Input:

Temperature = Hot (H)

Wind = Weak (W)

Should I play tennis?

$$\text{argmax}_y P(H, W | \text{play?}) P(\text{play?})$$

$$P(H, W | \text{Yes}) P(\text{Yes}) = 0.4 \times 0.3 \\ = 0.12$$

$$P(H, W | \text{No}) P(\text{No}) = 0.1 \times 0.7 \\ = 0.07$$

MAP prediction = Yes

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

Temperature: H(ot),
M(edium),
C(ool)

Humidity: H(igh),
N(ormal),
L(ow)

Wind: S(strong),
W(eak)

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

We need to learn
Temperature

1. The prior $P(\text{Play?})$
2. The likelihoods $P(X \mid \text{Play?})$

Humidity: N(ormal),
L(ow)

Wind: S(strong),
W(eak)

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Prior $P(\text{play?})$

- A single number (Why only one?)

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Prior $P(\text{play?})$

- A single number (Why only one?)

Likelihood $P(\mathbf{X} | \text{Play?})$

- There are 4 features
- For each value of Play? (+/-), we need a value for each possible assignment: $P(x_1, x_2, x_3, x_4 | \text{Play?})$

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-
	3	3	3	2	

Values for this feature

Prior $P(\text{play?})$

- A single number (Why only one?)

Likelihood $P(\mathbf{X} | \text{Play?})$

- There are 4 features
- For each value of Play? (+/-), we need a value for each possible assignment: $P(x_1, x_2, x_3, x_4 | \text{Play?})$

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-
	3	3	3	2	

Values for this feature

Prior $P(\text{play?})$

- A single number (Why only one?)

Likelihood $P(\mathbf{X} | \text{Play?})$

- There are 4 features
- For each value of Play? (+/-), we need a value for each possible assignment: $P(x_1, x_2, x_3, x_4 | \text{Play?})$
- $(3 \cdot 3 \cdot 3 \cdot 2 - 1)$ parameters in each case

a lot of parameters!!

One for each assignment

Need a lot of data to estimate these many numbers!

How hard is it to learn probabilistic models?

Prior $P(Y)$

- If there are k labels, then $k - 1$ parameters

Likelihood $P(\mathbf{X} \mid Y)$

- We need a value for each possible $P(x_1, x_2, \dots, x_d \mid y)$ for each y
- Need a lot of parameters!!

Need a lot of data to estimate these many numbers!

High model complexity

If there is very limited data, high variance in the parameters

How can we deal with this?

Answer: Make independence assumptions

Lecture 18: Naïve Bayes

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Bayes Theorem



Posterior probability: What is the probability of Y given that X is observed?

Likelihood: What is the likelihood of observing X given a specific Y?

Prior probability: What is our belief in Y before we see X?

MAP prediction

Let's use the Bayes rule for predicting y given an input \mathbf{x}

$$P(Y = y|X = \mathbf{x}) = \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

Predict y for the input \mathbf{x} using

$$\arg \max_y P(X = \mathbf{x}|Y = y)P(Y = y)$$

MAP prediction

Don't confuse with *MAP learning*:

finds hypothesis by

$$h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$$

Let's use the Bayes rule for predicting y given an input \mathbf{x}

$$P(Y = y|X = \mathbf{x}) = \frac{P(X = \mathbf{x}|Y = y)P(Y = y)}{P(X = \mathbf{x})}$$

Predict y for the input \mathbf{x} using

$$\arg \max_y P(X = \mathbf{x}|Y = y)P(Y = y)$$

MAP prediction

Predict y for the input \mathbf{x} using

$$\arg \max_y P(X = \mathbf{x}|Y = y) | P(Y = y)$$

Likelihood of observing this input \mathbf{x} when the label is y

Prior probability of the label being y

All we need are these two sets of probabilities

Example: Tennis

Prior	Play tennis	$P(\text{Play tennis})$
	Yes	0.3
	No	0.7

Without any other information, what is the prior probability that I should play tennis?

Temperature	Wind	$P(T, W \text{Tennis} = \text{Yes})$
Hot	Strong	0.15
Hot	Weak	0.4
Cold	Strong	0.1
Cold	Weak	0.35

On days that I do play tennis, what is the probability that the temperature is T and the wind is W?

Temperature	Wind	$P(T, W \text{Tennis} = \text{No})$
Hot	Strong	0.4
Hot	Weak	0.1
Cold	Strong	0.3
Cold	Weak	0.2

On days that I don't play tennis, what is the probability that the temperature is T and the wind is W?

Example: Tennis

Prior	Play tennis	$P(\text{Play tennis})$
	Yes	0.3
	No	0.7

Temperature	Wind	$P(T, W \mid \text{Tennis} = \text{Yes})$
Hot	Strong	0.15
Hot	Weak	0.4
Cold	Strong	0.1
Cold	Weak	0.35

Temperature	Wind	$P(T, W \mid \text{Tennis} = \text{No})$
Hot	Strong	0.4
Hot	Weak	0.1
Cold	Strong	0.3
Cold	Weak	0.2

Input:

Temperature = Hot (H)

Wind = Weak (W)

Should I play tennis?

Example: Tennis

Prior	Play tennis	P(Play tennis)
	Yes	0.3
	No	0.7

Likelihood	Temperature	Wind	P(T, W Tennis = Yes)
	Hot	Strong	0.15
	Hot	Weak	0.4
	Cold	Strong	0.1
	Cold	Weak	0.35

Likelihood	Temperature	Wind	P(T, W Tennis = No)
	Hot	Strong	0.4
	Hot	Weak	0.1
	Cold	Strong	0.3
	Cold	Weak	0.2

Input:

Temperature = Hot (H)

Wind = Weak (W)

Should I play tennis?

$$\text{argmax}_y P(H, W | \text{play?}) P(\text{play?})$$

$$P(H, W | \text{Yes}) P(\text{Yes}) = 0.4 \times 0.3 \\ = 0.12$$

$$P(H, W | \text{No}) P(\text{No}) = 0.1 \times 0.7 \\ = 0.07$$

MAP prediction = Yes

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

Temperature: H(ot),
M(edium),
C(ool)

Humidity: H(igh),
N(ormal),
L(ow)

Wind: S(strong),
W(eak)

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny),
O(vercast),
R(ainy)

We need to learn
Temperature

1. The prior $P(\text{Play?})$
2. The likelihoods $P(X \mid \text{Play?})$

Humidity: N(ormal),
L(ow)

Wind: S(strong),
W(eak)

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Prior $P(\text{play?})$

- A single number

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Prior $P(\text{play?})$

- A single number (Why only one?)

Likelihood $P(\mathbf{X} | \text{Play?})$

- There are 4 features
- For each value of Play? (+/-), we need a value for each possible assignment: $P(x_1, x_2, x_3, x_4 | \text{Play?})$

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-
	3	3	3	2	

Values for this feature

Prior $P(\text{play?})$

- A single number (Why only one?)

Likelihood $P(\mathbf{X} \mid \text{Play?})$

- There are 4 features
- For each value of Play? (+/-), we need a value for each possible assignment: $P(x_1, x_2, x_3, x_4 \mid \text{Play?})$

How hard is it to learn probabilistic models?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-
	3	3	3	2	

Values for this feature

Prior $P(\text{play?})$

- A single number (Why only one?)

Likelihood $P(\mathbf{X} | \text{Play?})$

- There are 4 features
- For each value of Play? (+/-), we need a value for each possible assignment: $P(x_1, x_2, x_3, x_4 | \text{Play?})$
- $(3 \cdot 3 \cdot 3 \cdot 2 - 1)$ parameters in each case

a lot of parameters!!

One for each assignment

Need a lot of data to estimate these many numbers!

How hard is it to learn probabilistic models?

Prior $P(Y)$

- If there are k labels, then $k - 1$ parameters

Likelihood $P(\mathbf{X} | Y)$

- We need a value for each possible $P(x_1, x_2, \dots, x_d | y)$ for each y
- Need a lot of parameters!!

Need a lot of data to estimate these many numbers!

High model complexity

If there is very limited data, high variance in the parameters

How can we deal with this?

Answer: Make independence assumptions

Recall: Conditional independence

Suppose X, Y and Z are random variables

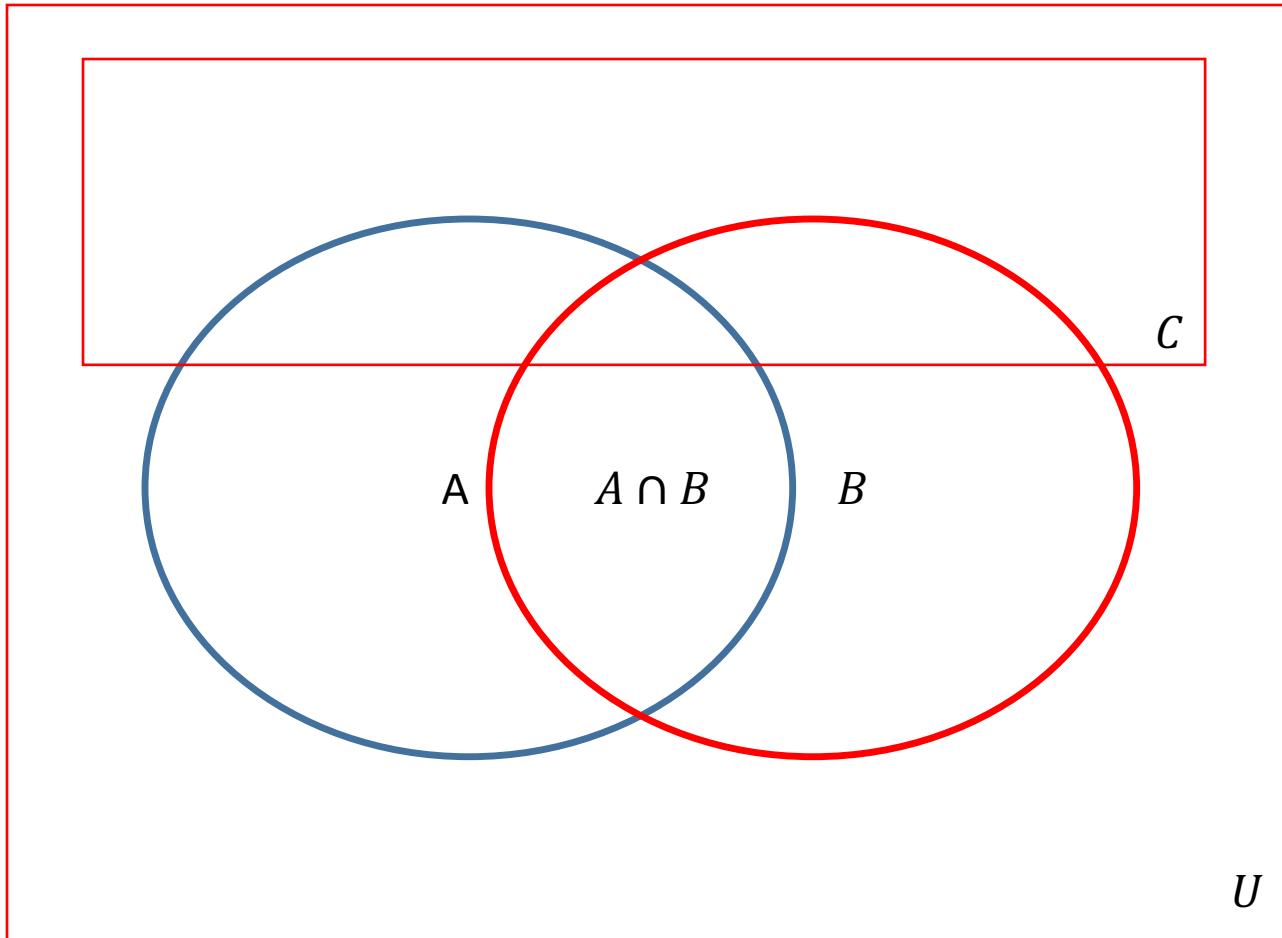
X is *conditionally independent* of Y given Z if the probability distribution of X is independent of the value of Y when Z is observed

$$P(X|Y, Z) = P(X|Z)$$

Or equivalently

$$P(X, Y|Z) = P(X|Z)P(Y|Z)$$

conditionally independent \neq independent



Modeling the features

$P(x_1, x_2, \dots, x_d | y)$ required a lot of parameters

Consider we have d Boolean features for k class, we need $k(2^d - 1)$ parameters

What if all the features were conditionally independent given the label?

Modeling the features

$P(x_1, x_2, \dots, x_d | y)$ required $k(2^d - 1)$ parameters

What if all the features were conditionally independent given the label?

The Naïve Bayes Assumption

That is,

$$P(x_1, x_2, \dots, x_d | y) = P(x_1 | y)P(x_2 | y) \cdots P(x_d | y)$$

Requires only d numbers for each label. kd parameters overall. Not bad!

The Naïve Bayes Classifier

Assumption: Features are conditionally independent given the label Y

To predict, we need two sets of probabilities

- ❖ Prior $P(y)$
- ❖ For each x_j , we have the likelihood $P(x_j | y)$

The Naïve Bayes Classifier

Assumption: Features are conditionally independent given the label Y

To predict, we need two sets of probabilities

- ❖ Prior $P(y)$
- ❖ For each x_j , we have the likelihood $P(x_j | y)$

Decision rule

$$h_{NB}(x) = \operatorname{argmax}_y P(y)P(x_1, x_2, \dots, x_d | y)$$

The Naïve Bayes Classifier

Assumption: Features are conditionally independent given the label Y

To predict, we need two sets of probabilities

- ❖ Prior $P(y)$
- ❖ For each x_j , we have the likelihood $P(x_j | y)$

Decision rule

$$\begin{aligned} h_{NB}(x) &= \operatorname{argmax}_y P(y)P(x_1, x_2, \dots, x_d | y) \\ &= \operatorname{argmax}_y P(y) \prod_j P(x_j | y) \end{aligned}$$

Decision boundaries of naïve Bayes

What is the decision boundary of the naïve Bayes classifier?

Consider the two class case. We predict the label to be + if

$$P(y = +) \prod_j P(x_j | y = +) > P(y = -) \prod_j P(x_j | y = -)$$

Decision boundaries of naïve Bayes

What is the decision boundary of the naïve Bayes classifier?

Consider the two class case. We predict the label to be + if

$$P(y = +) \prod_j P(x_j | y = +) > P(y = -) \prod_j P(x_j | y = -)$$

$$\frac{P(y = +) \prod_j P(x_j | y = +)}{P(y = -) \prod_j P(x_j | y = -)} > 1$$

Decision boundaries of naïve Bayes

What is the decision boundary of the naïve Bayes classifier?

Taking log and simplifying, we can show that the decision boundary of naïve Bayes is a linear function

$$\begin{aligned} \log \frac{P(y = -|\mathbf{x})}{P(y = +|\mathbf{x})} &= \log \frac{P(y = +) \prod_j P(x_j|y = +)}{P(y = -) \prod_j P(x_j|y = -)} \\ &= \log P(y = +) - \log P(y = -) + \sum_j (\log P(x_j|y = +) - \log P(x_j|y = -)) \end{aligned}$$

This is a linear function of the feature space!

Today's lecture

- ❖ The naïve Bayes Classifier
- ❖ Learning the naïve Bayes Classifier
- ❖ Generative model

Learning the naïve Bayes Classifier

- ❖ What is the hypothesis function h defined by?
 - ❖ A collection of probabilities

hypothesis?

Learning the naïve Bayes Classifier

- ❖ What is the hypothesis function h defined by?
 - ❖ A collection of probabilities
 - ❖ Prior for each label: $P(y)$
 - ❖ Likelihoods for feature x_j given a label: $P(x_j|y)$

hypothesis?

Learning the naïve Bayes Classifier

- ❖ What is the hypothesis function h defined by?
 - ❖ A collection of probabilities
 - ❖ Prior for each label: $P(y)$
 - ❖ Likelihoods for feature x_j given a label: $P(x_j | y)$

Suppose we have a data set $D = \{(x_i, y_i)\}$ with m examples

A note on convention for this section:

- Examples in the dataset are indexed by the subscript i (e.g. x_i)
- Features within an example are indexed by the subscript j
 - The j^{th} feature of the i^{th} example will be x_{ij}

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

Here h is defined by all the probabilities used to construct the naïve Bayes decision

Maximum likelihood estimation

Given a dataset $D = \{(\mathbf{x}_i, y_i)\}$ with m examples

$$h_{ML} = \arg \max_h \prod_{i=1}^m P((\mathbf{x}_i, y_i) | h)$$

$$h_{ML} = \arg \max_{h \in H} P(D|h)$$

Each example in the dataset is independent and identically distributed

So we can represent $P(D | h)$ as this product

Maximum likelihood estimation

Given a dataset $D = \{(\mathbf{x}_i, y_i)\}$ with m examples

$$h_{ML} = \arg \max_h \prod_{i=1}^m P((\mathbf{x}_i, y_i) | h)$$

Each example in the dataset is independent and identically distributed

So we can represent $P(D | h)$ as this product

Asks “What probability would this particular h assign to the pair (\mathbf{x}_i, y_i) ? ”

Maximum likelihood estimation

Given a dataset $D = \{(x_i, y_i)\}$ with m examples

$$\begin{aligned} h_{ML} &= \arg \max_h \prod_{i=1}^m P((\mathbf{x}_i, y_i) | h) \\ &= \arg \max_h \prod_{i=1}^m P(\mathbf{x}_i | y_i, h) P(y_i | h) \end{aligned}$$

Maximum likelihood estimation

Given a dataset $D = \{(\mathbf{x}_i, y_i)\}$ with m examples

$$\begin{aligned} h_{ML} &= \arg \max_h \prod_{i=1}^m P((\mathbf{x}_i, y_i) | h) \\ &= \arg \max_h \prod_{i=1}^m P(\mathbf{x}_i | y_i, h) P(y_i | h) \\ &= \arg \max_h \prod_{i=1}^m P(y_i | h) \prod_j P(x_{i,j} | y_i, h) \end{aligned}$$

\mathbf{x}_{ij} is the j^{th} feature of \mathbf{x}_i

The Naïve Bayes assumption

Maximum likelihood estimation

Given a dataset $D = \{(x_i, y_i)\}$ with m examples

$$\begin{aligned} h_{ML} &= \arg \max_h \prod_{i=1}^m P((\mathbf{x}_i, y_i) | h) \\ &= \arg \max_h \prod_{i=1}^m P(\mathbf{x}_i | y_i, h) P(y_i | h) \\ &= \arg \max_h \prod_{i=1}^m P(y_i | h) \prod_j P(x_{i,j} | y_i, h) \end{aligned}$$

How do we proceed?

Maximum likelihood estimation

Given a dataset $D = \{(x_i, y_i)\}$ with m examples

$$\begin{aligned} h_{ML} &= \arg \max_h \prod_{i=1}^m P((\mathbf{x}_i, y_i) | h) \\ &= \arg \max_h \prod_{i=1}^m P(\mathbf{x}_i | y_i, h) P(y_i | h) \\ &= \arg \max_h \prod_{i=1}^m P(y_i | h) \prod_j P(x_{i,j} | y_i, h) \\ &= \arg \max_h \sum_{i=1}^m \log P(y_i | h) + \sum_i \sum_j \log P(x_{i,j} | y_i, h) \end{aligned}$$

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_h \sum_{i=1}^m \log P(y_i|h) + \sum_i \sum_j \log P(x_{i,j}|y_i, h)$$

What next?

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_h \sum_{i=1}^m \log P(y_i|h) + \sum_i \sum_j \log P(x_{i,j}|y_i, h)$$

For simplicity, suppose there are two labels 1 and 0 and all features are binary

- **Prior:** $P(y = 1) = p$ and $P(y = 0) = 1 - p$

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_h \sum_{i=1}^m \log P(y_i|h) + \sum_i \sum_j \log P(x_{i,j}|y_i, h)$$

For simplicity, suppose there are two labels **1** and **0** and all features are binary

- **Prior:** $P(y = 1) = p$ and $P(y = 0) = 1 - p$
- **Likelihood** for each feature given a label
 - $P(x_j = 1 | y = 1) = a_j$ and $P(x_j = 0 | y = 1) = 1 - a_j$

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_h \sum_{i=1}^m \log P(y_i|h) + \sum_i \sum_j \log P(x_{i,j}|y_i, h)$$

For simplicity, suppose there are two labels **1** and **0** and all features are binary

- **Prior:** $P(y = 1) = p$ and $P(y = 0) = 1 - p$
- **Likelihood** for each feature given a label
 - $P(x_j = 1 | y = 1) = a_j$ and $P(x_j = 0 | y = 1) = 1 - a_j$
 - $P(x_j = 1 | y = 0) = b_j$ and $P(x_j = 0 | y = 0) = 1 - b_j$

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_h \sum_{i=1}^m \log P(y_i|h) + \sum_i \sum_j \log P(x_{i,j}|y_i, h)$$

- Prior: $P(y = 1) = p$ and $P(y = 0) = 1 - p$

$$P(y_i|h) = p^{[y_i=1]}(1-p)^{[y_i=0]}$$

$[z]$ is called the indicator function or the Iverson bracket

Its value is 1 if the argument z is true and zero otherwise

Learning the naïve Bayes Classifier

Maximum likelihood estimation

$$h_{ML} = \arg \max_h \sum_{i=1}^m \log P(y_i|h) + \sum_i \sum_j \log P(x_{i,j}|y_i, h)$$

Likelihood for each feature given a label

- $P(x_j = 1 | y = 1) = a_j$ and $P(x_j = 0 | y = 1) = 1 - a_j$
- $P(x_j = 1 | y = 0) = b_j$ and $P(x_j = 0 | y = 0) = 1 - b_j$

$$P(x_{ij}|y_i, h) = a_j^{[y_i=1, x_{ij}=1]} \times (1 - a_j)^{[y_i=1, x_{ij}=0]} \times b_j^{[y_i=0, x_{ij}=1]} \times (1 - b_j)^{[y_i=0, x_{ij}=0]}$$

Learning the naïve Bayes Classifier

Substituting and deriving the argmax, we get

$$p = \frac{\text{Count}(y_i = 1)}{\text{Count}(y_i = 1) + \text{Count}(y_i = 0)} \quad \longleftarrow P(y = 1) = p$$

Learning the naïve Bayes Classifier

Substituting and deriving the argmax, we get

$$p = \frac{\text{Count}(y_i = 1)}{\text{Count}(y_i = 1) + \text{Count}(y_i = 0)} \quad \longleftarrow P(y = 1) = p$$

$$a_j = \frac{\text{Count}(y_i = 1, x_{ij} = 1)}{\text{Count}(y_i = 1)} \quad \longleftarrow P(x_j = 1 \mid y = 1) = a_j$$

Learning the naïve Bayes Classifier

Substituting and deriving the argmax, we get

$$p = \frac{\text{Count}(y_i = 1)}{\text{Count}(y_i = 1) + \text{Count}(y_i = 0)} \quad \longleftarrow P(y = 1) = p$$

$$a_j = \frac{\text{Count}(y_i = 1, x_{ij} = 1)}{\text{Count}(y_i = 1)} \quad \longleftarrow P(x_j = 1 \mid y = 1) = a_j$$

$$b_j = \frac{\text{Count}(y_i = 0, x_{ij} = 1)}{\text{Count}(y_i = 0)} \quad \longleftarrow P(x_j = 1 \mid y = 0) = b_j$$

Let's learn a naïve Bayes classifier

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Let's learn a naïve Bayes classifier

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$P(\text{Play} = +) = 9/14$$

$$P(\text{Play} = -) = 5/14$$

Let's learn a naïve Bayes classifier

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$P(\text{Play} = +) = 9/14$$

$$P(\text{Play} = -) = 5/14$$

$$P(O = S \mid \text{Play} = +) = 2/9$$

Let's learn a naïve Bayes classifier

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$P(\text{Play} = +) = 9/14$$

$$P(\text{Play} = -) = 5/14$$

$$P(O = S \mid \text{Play} = +) = 2/9$$

$$P(O = R \mid \text{Play} = +) = 3/9$$

Let's learn a naïve Bayes classifier

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$P(\text{Play} = +) = 9/14$$

$$P(\text{Play} = -) = 5/14$$

$$P(O = S \mid \text{Play} = +) = 2/9$$

$$P(O = R \mid \text{Play} = +) = 3/9$$

$$P(O = O \mid \text{Play} = +) = 4/9$$

And so on, for other attributes and also for $\text{Play} = -$

Naïve Bayes: Learning and Prediction

- ❖ **Learning**
 - ❖ Count how often features occur with each label.
Normalize to get likelihoods
 - ❖ Priors from fraction of examples with each label
 - ❖ Generalizes to multiclass
- ❖ **Prediction**
 - ❖ Use learned probabilities to find highest scoring label

Important caveats with Naïve Bayes

1. In practice, features may not be conditionally independent given the label
 - ❖ Just because we assume that they are doesn't mean that that's how they behave in nature
 - ❖ We made a modeling assumption because it makes computation and learning easier

Important caveats with Naïve Bayes

2. Not enough training data to get good estimates of the probabilities from counts

The basic operation for learning likelihoods is counting how often a feature occurs with a label.

What if we never see a particular feature with a particular label?

Should we treat those counts as zero?

$$P(y) \prod_j P(x_j|y)$$

Important caveats with Naïve Bayes

2. Not enough training data to get good estimates of the probabilities from counts

The basic operation for learning likelihoods is counting how often a feature occurs with a label.

What if we never see a particular feature with a particular label?

That will make the probabilities zero

Should we treat those counts as zero?

Answer: [Smoothing](#)

- Add fake counts (very small numbers so that the counts are not zero)

Example: Classifying text

- ❖ Instance space: Text documents
- ❖ Labels: **Spam** or **NotSpam**
- ❖ Goal: To learn a function that can predict whether a new document is **Spam** or **NotSpam**

How would you build a Naïve Bayes classifier?

Let us brainstorm

- How to represent documents?
- How to estimate probabilities?
- How to classify?

Example: Classifying text

1. Represent documents by a vector of words
A sparse vector consisting of one feature per word
2. Learning from N labeled documents

1. Priors $P(\text{Spam}) = \frac{\text{Count}(\text{Spam})}{N}; P(\text{NotSpam}) = 1 - P(\text{Spam})$

2. For each word w in vocabulary :

$$P(w|\text{Spam}) = \frac{\text{Count}(w, \text{Spam}) + 1}{\text{Count}(\text{Spam}) + |\text{Vocabulary}|}$$

$$P(w|\text{NotSpam}) = \frac{\text{Count}(w, \text{NotSpam}) + 1}{\text{Count}(\text{NotSpam}) + |\text{Vocabulary}|}$$

Example: Classifying text

1. Represent documents by a vector of words
A sparse vector consisting of one feature per word
2. Learning from N labeled documents

1. Priors $P(\text{Spam}) = \frac{\text{Count}(\text{Spam})}{N}; P(\text{NotSpam}) = 1 - P(\text{Spam})$

2. For each word w in vocabulary :

$$P(w|\text{Spam}) = \frac{\text{Count}(w, \text{Spam}) + 1}{\text{Count}(\text{Spam}) + |\text{Vocabulary}|}$$

$$P(w|\text{NotSpam}) = \frac{\text{Count}(w, \text{NotSpam}) + 1}{\text{Count}(\text{NotSpam}) + |\text{Vocabulary}|}$$

How often does a word occur with a label?

Example: Classifying text

1. Represent documents by a vector of words
A sparse vector consisting of one feature per word
2. Learning from N labeled documents

1. Priors $P(\text{Spam}) = \frac{\text{Count}(\text{Spam})}{N}; P(\text{NotSpam}) = 1 - P(\text{Spam})$

2. For each word w in vocabulary :

$$P(w|\text{Spam}) = \frac{\text{Count}(w, \text{Spam}) + 1}{\text{Count}(\text{Spam}) + |\text{Vocabulary}|}$$

$$P(w|\text{NotSpam}) = \frac{\text{Count}(w, \text{NotSpam}) + 1}{\text{Count}(\text{NotSpam}) + |\text{Vocabulary}|}$$

Smoothing

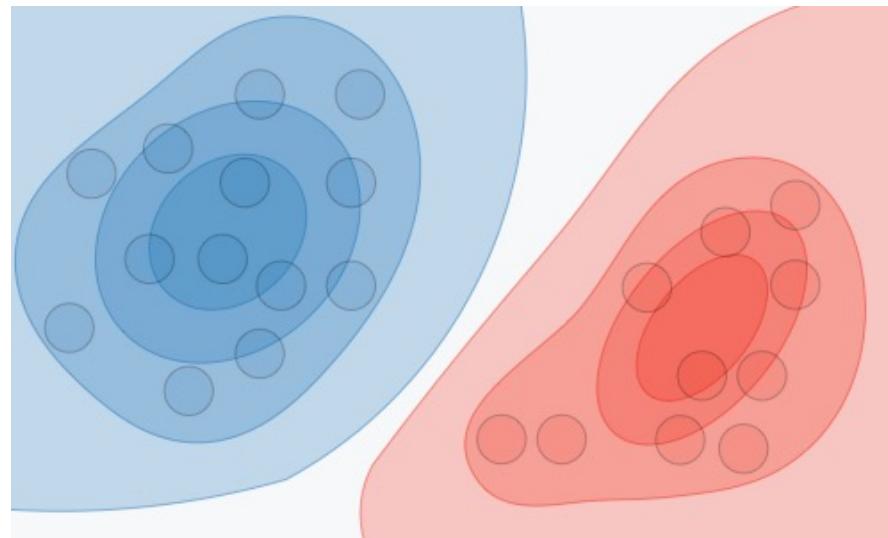
Generative v.s Discriminative Models

Generative (e.g., Naïve Bayes)

or $P(Y)$ & $P(X|Y)$

- ❖ Estimate $P(Y|X)$ through $P(X, Y)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

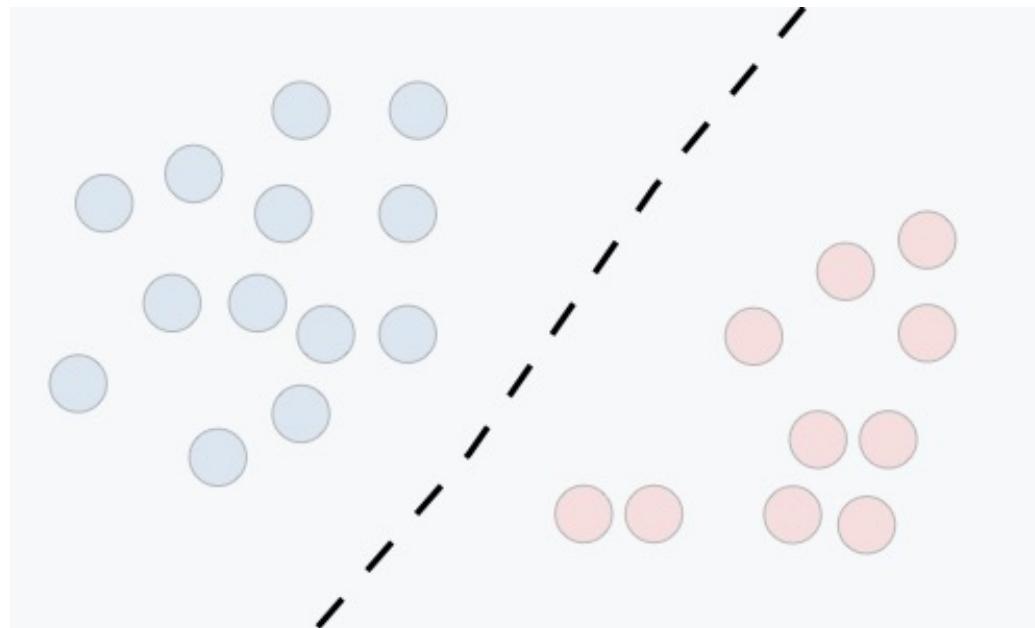


What is the input instances X look like if they belongs to the class Y

Discriminative (e.g., logistic regression, SVM)

or $P(Y)$ & $P(X|Y)$

- ❖ Estimate $P(Y|X)$



What features can be used to differentiate between different classes

Example



V.S.



Lecture 19: EM & GMM

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Final Exam

- ❖ Official exam time 12/6 8:00am –11:00am
 - ❖ Submission accepted until 12/7 8:00am
 - ❖ Must complete in 4hr
(3 hr: exam time 1 hr: grace period)
 - ❖ No more extension/late submission
- ❖ Open-Book;
No internet search/ **No discussion**
- ❖ Please write your answers clearly and submit it to Gradescope
 - ❖ Type or make sure your writing is readable

Final Exam

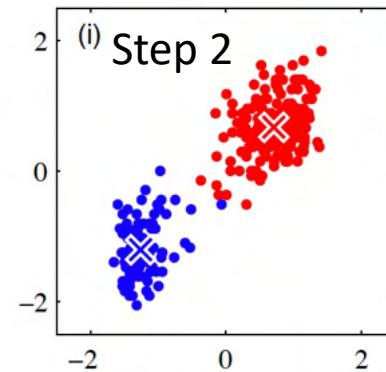
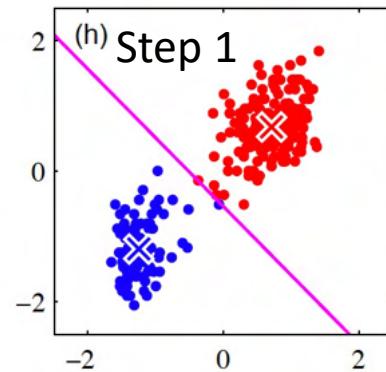
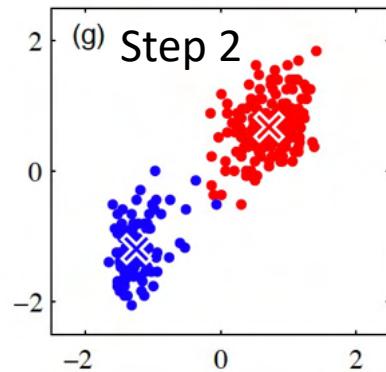
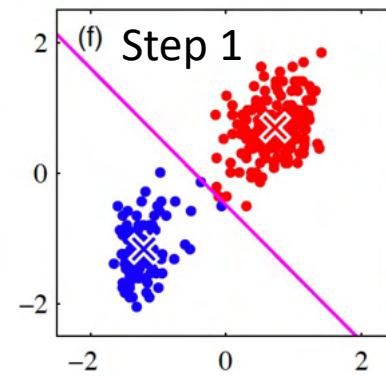
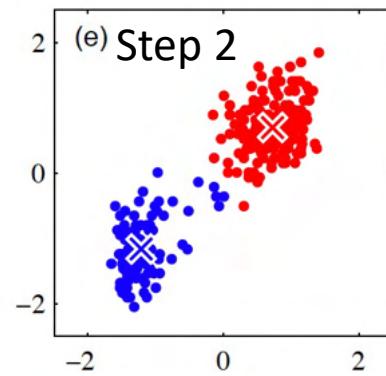
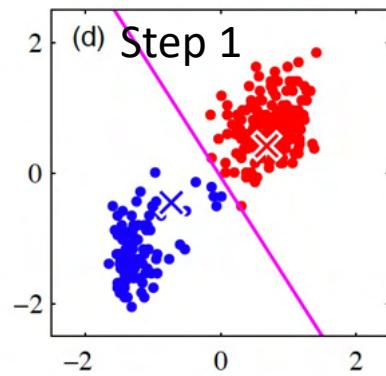
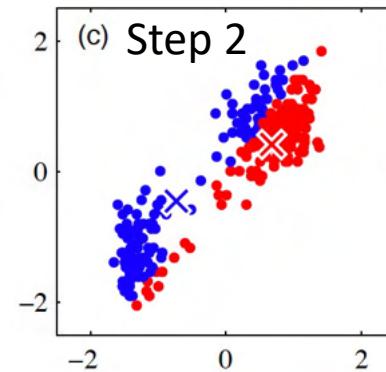
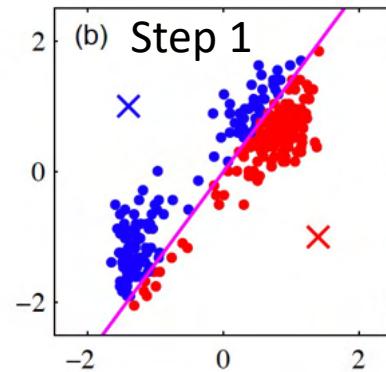
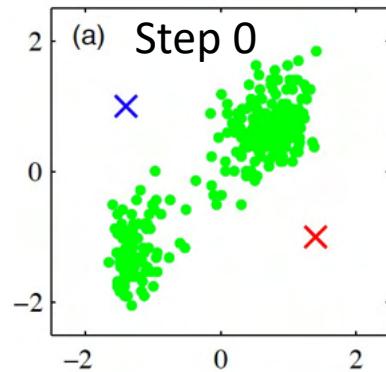
- ❖ Ask **private** questions at Piazza
 - ❖ Only visible to TAs and Instructor
 - ❖ Ask only clarification questions / typos in exam
 - ❖ We won't be able to provide any hint
 - ❖ Announcement will be posted in Piazza
 - ❖ We will actively monitor and answer questions at Piazza during the exam time
12/6 8:00am – 11:00am
- ❖ You may include additional note at the end
 - ❖ Write down your assumptions if the question is unclear
 - ❖ if you file regrading request later, we can only regrade based on what is written in the answer sheet

Grading

- ❖ We expect to complete the grading by 12/13
- ❖ We will allow 24 hours for submitting regrading request
- ❖ We will grade based on rubrics
- ❖ Final grade will be computed based on the formulation in Lecture 1.

Unsupervised Learning

K-Means



K-means algorithm a.k.a Lloyd's algorithm

- ❖ Step 0: randomly assign the cluster centers $\{\mu_k\}$
- ❖ Step 1: Minimize J over $\{r_{nk}\}$ -- Assign every point to the closest cluster center

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- ❖ Step 2: Minimize J over $\{\mu_k\}$ -- update the cluster centers

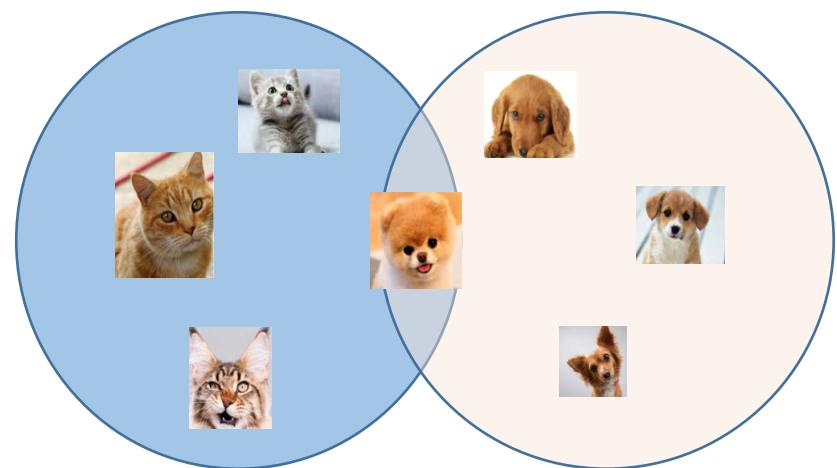
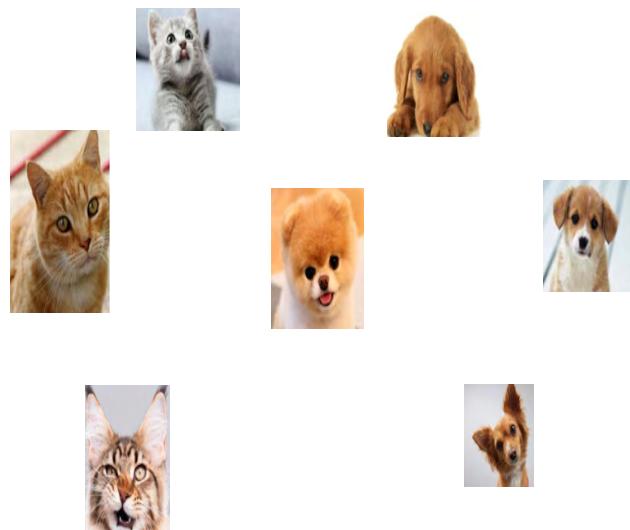
$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

- ❖ Loop until it converges

Unsupervised Learning – Expectation Maximization (EM)

How about unsupervised learning

- ❖ In unsupervised learning, we only observed input distribution $\tilde{P}(X)$



MLE in unsupervised learning

- ❖ We only have observation of $\tilde{P}(X)$
- ❖ In generative model, we model $P(X, Y | \Theta)$
- ❖ We know $P(X | \Theta) = \sum_Y P(X, Y | \Theta)$
- ❖ Therefore, MLE is

$$\operatorname{argmax}_{\Theta} P(X | \Theta) =$$

$$\operatorname{argmax}_{\Theta} \sum_Y P(X, Y | \Theta)$$

EM algorithm

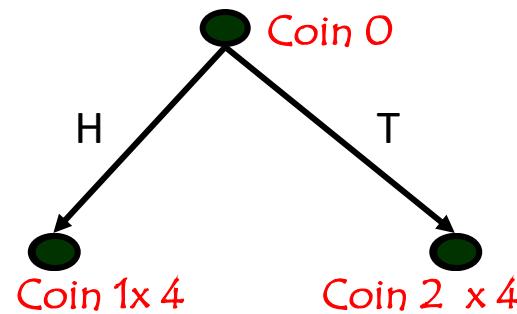
- ❖ EM algorithm essentially solves
 $\operatorname{argmax}_{\Theta} \log \sum_Y P(X, Y | \Theta)$ by iteratively
updating Θ (come back to this point later)
 - ❖ At iteration t, the model Θ^t
 - ❖ E-Step: Estimate $P(Y | X, \Theta^t)$
 - ❖ M-Step: Optimize
$$\max_{\Theta} \sum_Y [P(Y | X, \Theta^t) \log P(X, Y | \Theta)]$$
- ❖ In general, it converges to a local maximum

Three Coins Example

- ❖ We observe a series of coin tosses generated in the following way:
- ❖ A person has three coins.
 - ❖ Coin 0: probability of Head is α
 - ❖ Coin 1: probability of Head p
 - ❖ Coin 2: probability of Head q
- ❖ Consider the following coin-tossing scenarios:

Scenario I

- ❖ Toss coin 0.
If Head – toss coin 1; o/w – toss coin 2



Observing the sequence

HHHHT, THTHT, HHHHT, HHTTH

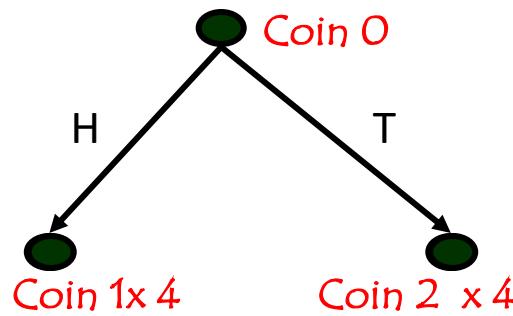
produced by Coin 0 , Coin1 and Coin2

Question: Estimate most likely values for α , p , q (the probability of H in each coin) and the probability to use each of the coins

Supervised Learning

Scenario II

- ❖ Toss coin 0.
If Head – toss coin 1; o/w – toss coin 2



Observing the sequence

HHHT, HTHT, HHHT, HTTH

produced by ~~Coin 0~~, Coin1 or Coin2

Question: Estimate most likely values for α , p , q (the probability of H in each coin) and the probability to use each of the coins

Intuition of EM algorithm

- ❖ Use an iterative approach for estimating the parameters:
 - ❖ Guess the probability that a given data point came from Coin 1 or 2; Generate fictional labels, weighted according to this probability.
 - ❖ Then, compute the most likely value of the parameters. [supervised learning]
 - ❖ Compute the likelihood of the data given this model.
 - ❖ Re-estimate the parameter setting: set them to maximize the likelihood of the data.

Step 1: initialization

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

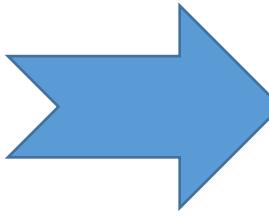
- ❖ Guess the probability that a given data point came from Coin 1 or 2

Sequence	coin 0=H	coin 0=T
HHHT	100%	0 %
HTHT	100%	0%
HHHT	100%	0%
HTTH	0%	100%

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- ❖ Now, compute the most likely value of the parameters. [Supervised Learning]

Sequence	coin 0=H	coin 0=T	
HHHT	100%	0 %	
HTHT	100%	0%	
HHHT	100%	0%	
HTTH	0%	100%	
			HHHHT
			HHTHT
			HHHHT
			THTTH

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- ❖ Now, compute the most likely value of the parameters.

Sequence

HHHHT

HHTHT

HHHHT

THTTH

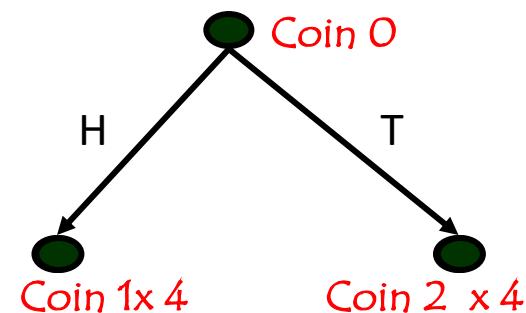
$$\alpha_1 = \frac{3}{3 + 1} = \frac{3}{4}$$

$$p_1 = \frac{8}{8 + 4} = \frac{2}{3}$$

$$q_1 = \frac{2}{2 + 2} = \frac{1}{2}$$

Step3: Compute Posterior

$$P(\text{coin 0} | \text{Sequence}; \alpha_1, p_1, q_1)$$

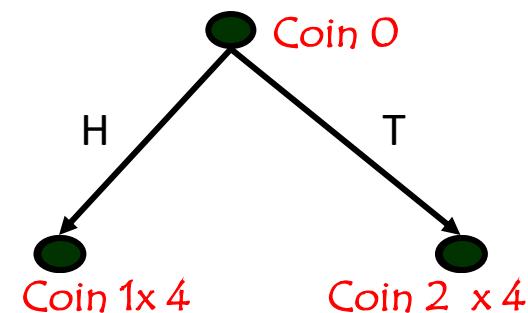


- ❖ Compute the likelihood of the data given this model

	$\alpha_1 p_1^{\#H} (1 - p_1)^{\#T}$	$(1 - \alpha_1) q_1^{\#H} (1 - q_1)^{\#T}$	$\alpha_1 = \frac{3}{4}$
Sequence	coin 0 = H	coin 0 = T	
HHHT	$\frac{3}{4} \left(\frac{2}{3}\right)^3 \frac{1}{3}$	$\frac{1}{4} \left(\frac{1}{2}\right)^3 \frac{1}{2}$	
HTHT	$\frac{3}{4} \left(\frac{2}{3}\right)^2 \left(\frac{1}{3}\right)^2$	$\frac{1}{4} \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2$	$p_1 = \frac{2}{3}$
HHHT	$\frac{3}{4} \left(\frac{2}{3}\right)^3 \frac{1}{3}$	$\frac{1}{4} \left(\frac{1}{2}\right)^3 \frac{1}{2}$	
HTTH	$\frac{3}{4} \left(\frac{2}{3}\right)^2 \left(\frac{1}{3}\right)^2$	$\frac{1}{4} \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2$	$q_1 = \frac{1}{2}$

Step3: Compute Posterior

$$P(\text{coin } 0 = H \mid Seq) = \frac{P(\text{coin}0=H, Seq)}{P(\text{coin}0=H, Seq) + P(\text{coin}0=T, Seq)}$$



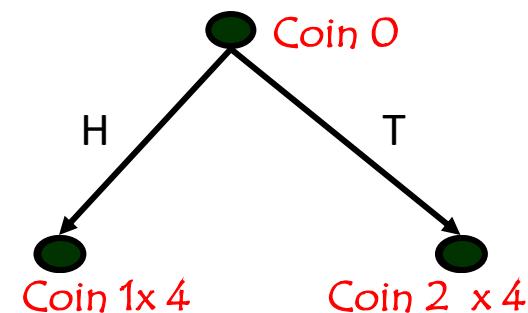
❖ Compute $P(\text{coin}0, \text{Seq})$

Sequence	$\alpha_1 p_1^{\#H} (1 - p_1)^{\#T}$	$(1 - \alpha_1) q_1^{\#H} (1 - q_1)^{\#T}$	$\alpha_1 = \frac{3}{4}$
HHHT	0.074	0.0156	
HTHT	0.037	0.0156	$p_1 = \frac{2}{3}$
HHHT	0.074	0.0156	
HTTH	0.037	0.0156	$q_1 = \frac{1}{2}$

$$\frac{0.074}{0.074 + 0.0156} = 82.6\%$$

Step3: Compute Posterior

$$P(\text{coin } 0 = H \mid Seq) = \frac{P(\text{coin } 0 = H, Seq)}{P(\text{coin } 0 = H, Seq) + P(\text{coin } 0 = T, Seq)}$$



❖ Compute $P(\text{coin } 0 = H \mid Seq)$

	coin 0=H	coin 0=T
HHHT	82.6%	17.4%
HTHT	70.3%	29.7%
HHHT	82.6%	17.4%
HTTH	70.3%	29.7%

$$\alpha_1 = \frac{3}{4}$$

$$p_1 = \frac{2}{3}$$

$$q_1 = \frac{1}{2}$$

$$\frac{0.074}{0.074 + 0.0156} = 82.6\%$$

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- ❖ Now, compute the most likely value of the parameters. [recall the scenario I]

	coin 0=H	coin 0=T	Soft label assignment/ weighted instance
HHHT	82.6%	17.4%	0.826 HHHHT
HTHT	70.3%	29.7%	0.174 THHHT
HHHT	82.6%	17.4%	0.703 HHTHT
HTTH	70.3%	29.7%	0.297 THTHT
			0.826 HHHHT
			0.174 THHHT
			0.703 HHTTH
			0.297 THTTH

Step 2: Maximum Conditional Likelihood

Coin 0: probability of Head is α
Coin 1: probability of Head p
Coin 2: probability of Head q

- ❖ Now, compute the most likely value of the parameters. [recall the scenario I]

0.826 HHHT

0.174 THHHT

0.703 HHTHT

0.297 THTHT

0.826 HHHT

0.174 THHHT

0.703 HHTTH

0.297 THTTH

$$\alpha_2 = \frac{0.826 \times 2 + 0.703 \times 2}{4} = 0.765$$

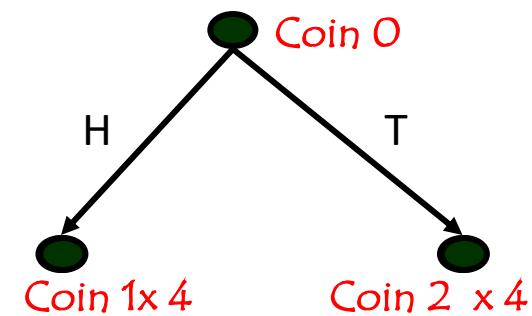
$$p_2 = \frac{0.826 \times 6 + 0.703 \times 4}{0.826 \times 8 + 0.703 \times 8} = 0.635$$

$$q_2 = \frac{0.174 \times 6 + 0.297 \times 4}{0.174 \times 8 + 0.297 \times 8} = 0.592$$

Step3: Compute Posterior

$$P(\text{coin } 0 = H \mid \text{Seq}) = \frac{P(\text{coin}0=H, \text{Seq})}{P(\text{coin}0=H, \text{Seq}) + P(\text{coin}0=T, \text{Seq})}$$

❖ Compute $P(\text{coin}0 \mid \text{Seq})$



$$\alpha_2 = 0.765$$

$$p_2 = 0.635$$

$$q_2 = 0.592$$

Sequence

HHHT

HTHT

HHHT

HTTH

coin 0=H

coin 0=T

Intuition of EM algorithm

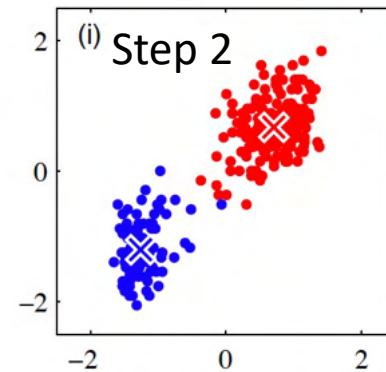
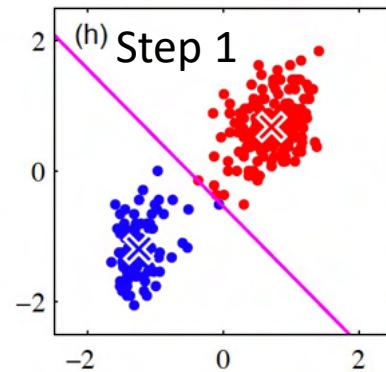
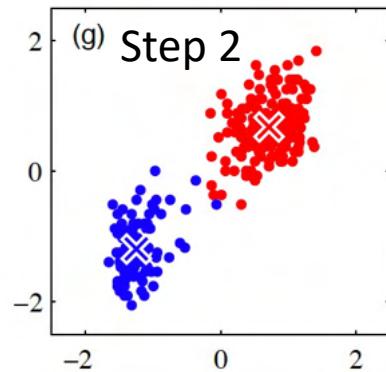
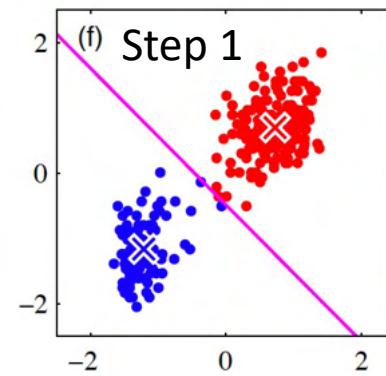
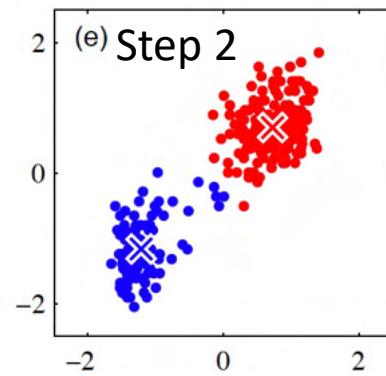
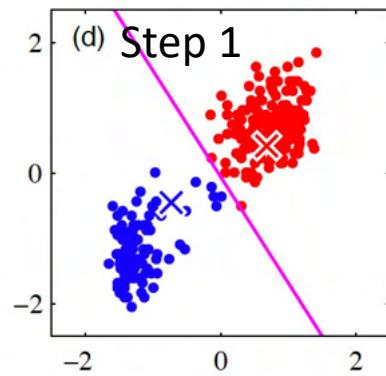
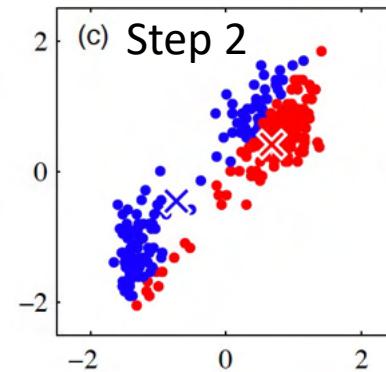
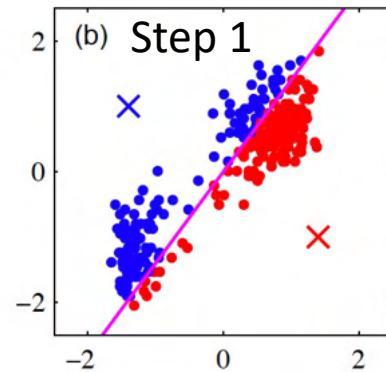
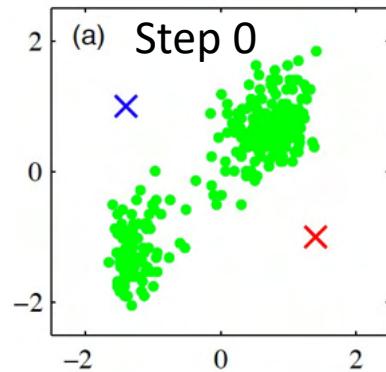
- ❖ Use an iterative approach for estimating the parameters:
 - ❖ Guess the probability that a given data point came from Coin 1 or 2; Generate fictional labels, weighted according to this probability.
 - ❖ Then, compute the most likely value of the parameters. [supervised learning]
 - ❖ Compute the likelihood of the data given this model.
 - ❖ Re-estimate the parameter setting: set them to maximize the likelihood of the data.

EM algorithm

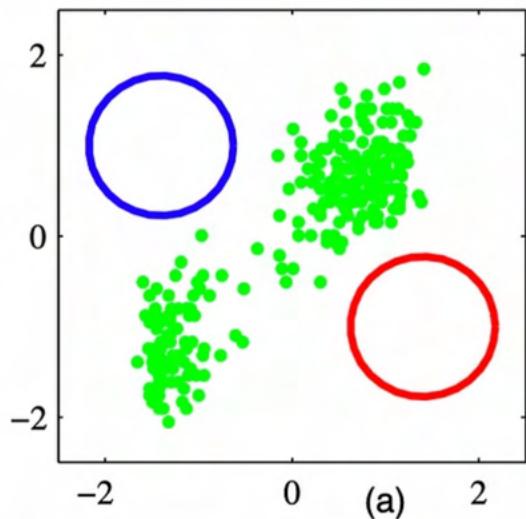
- ❖ EM algorithm essentially solves
 $\operatorname{argmax}_{\Theta} \log \sum_Y P(X, Y | \Theta)$ by iteratively
updating Θ (come back to this point later)
 - ❖ At iteration t, the model Θ^t
 - ❖ E-Step: Estimate $P(Y | X, \Theta^t)$
 - ❖ M-Step: Optimize
$$\max_{\Theta} \sum_Y [P(Y | X, \Theta^t) \log P(X, Y | \Theta)]$$
- ❖ In general, it converges to a local maximum

Gaussian Mixture Models

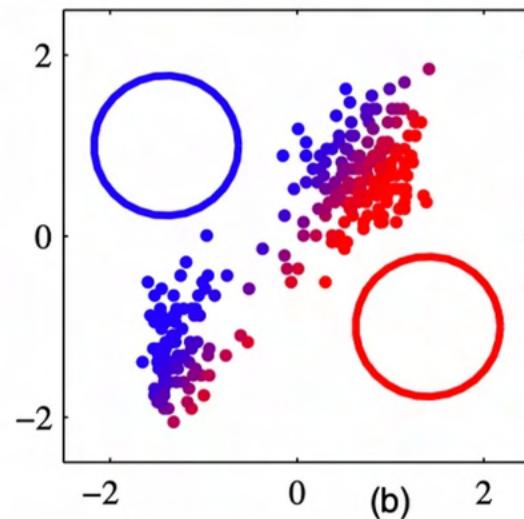
K-Means



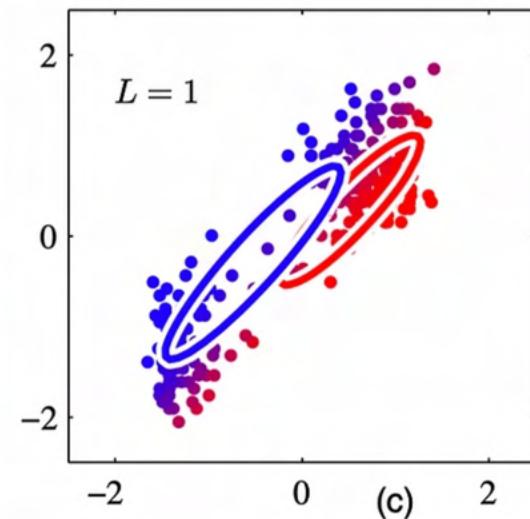
GMM



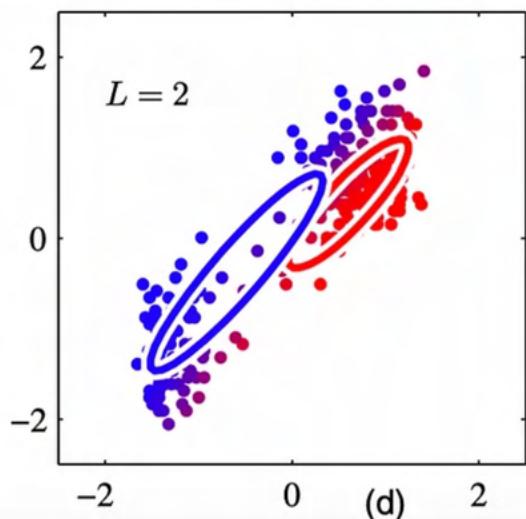
(a)



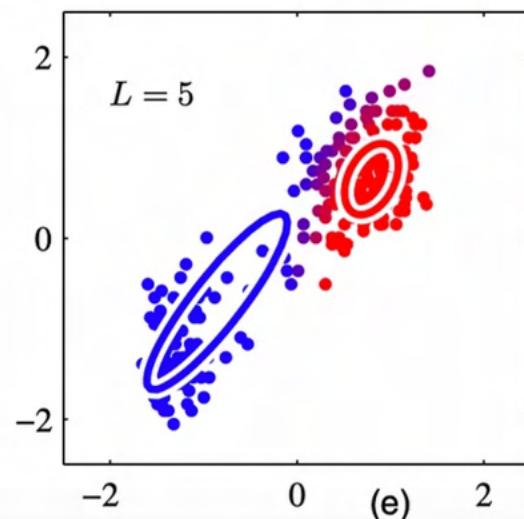
(b)



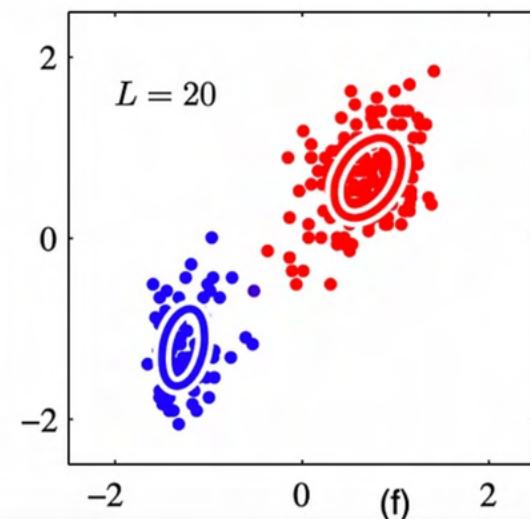
(c)



(d)



(e)

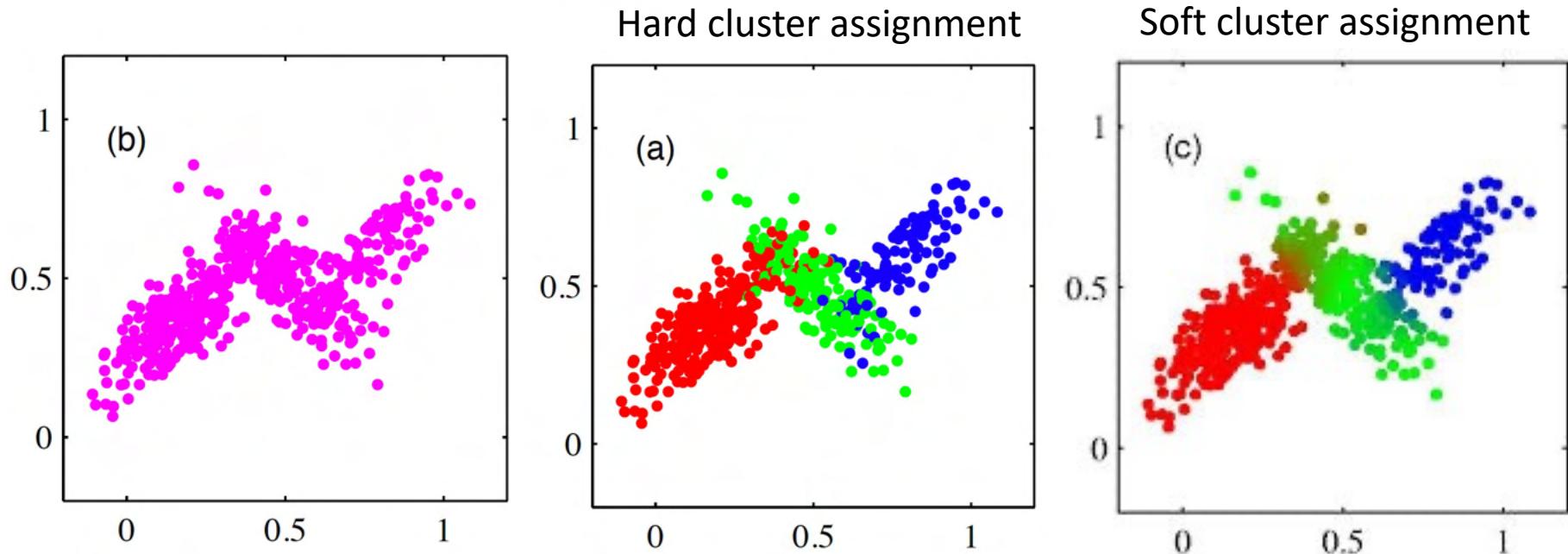


(f)

Gaussian mixture models

- ❖ Assume the probability density function for x as

$$p(x) = \sum_{k=1}^K \omega_k N(x|\mu_k, \Sigma_k)$$



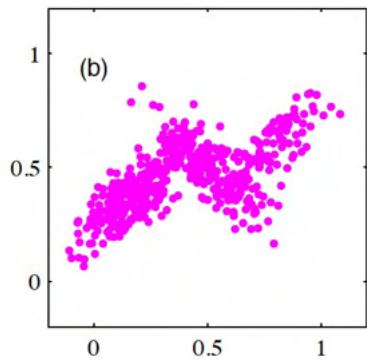
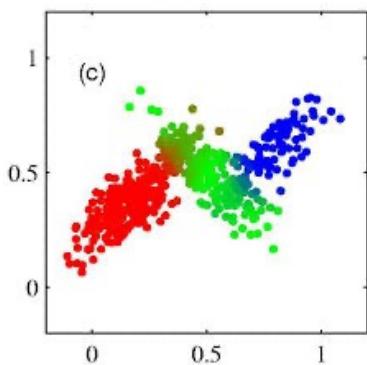
Example

The conditional distribution between \mathbf{x} and z (representing color) are

$$p(\mathbf{x}|z = red) = N(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$

$$p(\mathbf{x}|z = blue) = N(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

$$p(\mathbf{x}|z = green) = N(\mathbf{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is thus

$$p(\mathbf{x}) = p(red)N(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(blue)N(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p(green)N(\mathbf{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

ω_k

Training the Gaussian Mixture Model

Iterative procedure (similar to k-means)

- ❖ Let θ represent all parameters $\{\omega_k, \mu_k, \Sigma_k\}$

Step 0: initialize θ with some values (random or otherwise)

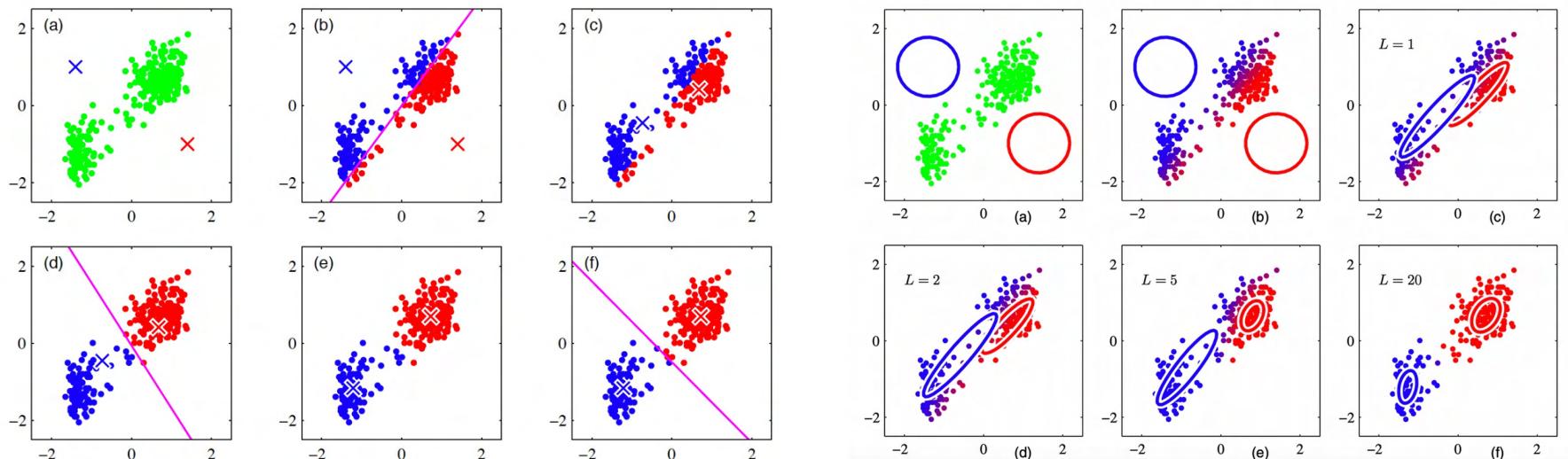
Step 1: compute γ_{nk} using the current θ

Step 2: update θ using the just computed γ_{nk}

Step 3: go back to Step 1

Similar to K-means

- ❖ Alternatively:
 - 1) assign points to clusters
 - 2) update cluster centers/variances...



Estimate γ_{nk}

e.g., probability x_n belongs to the red cluster,
given the input and model

- ❖ $\gamma_{nk} = P(z_n = k|x_n)$
the assignment of z_n to cluster k
- ❖ posterior probability

$$p(z_n = k|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|z_n = k)p(z_n = k)}{p(\mathbf{x}_n)} = \frac{p(\mathbf{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^K p(\mathbf{x}_n|z_n = k')p(z_n = k')}$$

$N(x|\mu_k, \Sigma_k)$ ω_k

Parameter estimation for GMMs

- ❖ If cluster assignments are observed $\{z_n\}$ are given
 - ❖ We know the cluster of each point
 - ❖ Let $\gamma_{nk} \in [0, 1]$ is a soft assignment of instance n to cluster k ,
- ❖ Then the maximum likelihood estimation is

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k) (\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

- For ω_k : count the number of data points whose z_n is k and divide by the total number of data points (note that $\sum_k \sum_n \gamma_{nk} = N$)
- For $\boldsymbol{\mu}_k$: get all the data points whose z_n is k , compute their mean
- For $\boldsymbol{\Sigma}_k$: get all the data points whose z_n is k , compute their covariance matrix

Thank you

- ❖ Please log in MyUCLA for course survey



Lecture 20: Generative v.s. Discriminative

Fall 2021

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikumar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Final Exam

- ❖ Official exam time 12/6 8:00am –11:00am
 - ❖ Submission accepted until 12/7 8:00am
 - ❖ Must complete in 4hr
(3 hr: exam time 1 hr: grace period)
 - ❖ No more extension/late submission
- ❖ Open-Book;
No internet search/ **No discussion**
- ❖ Please write your answers clearly and submit it to Gradescope
 - ❖ Type or make sure your writing is readable

Final Exam

- ❖ Ask **private** questions at Piazza
 - ❖ Only visible to TAs and Instructor
 - ❖ Ask only clarification questions / typos in exam
 - ❖ We won't be able to provide any hint
 - ❖ Announcement will be posted in Piazza
 - ❖ We will actively monitor and answer questions at Piazza during the exam time
12/6 8:00am – 11:00am
- ❖ You may include additional note at the end
 - ❖ Write down your assumptions if the question is unclear
 - ❖ if you file regrading request later, we can only regrade based on what is written in the answer sheet

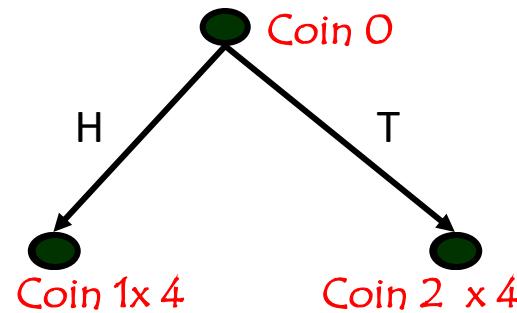
Grading

- ❖ We expect to complete the grading by 12/13
- ❖ We will allow 24 hours for submitting regrading request
- ❖ We will grade based on rubrics
- ❖ Final grade will be computed based on the formulation in Lecture 1.

Unsupervised Learning – Expectation Maximization (EM)

Scenario I

- ❖ Toss coin 0.
If Head – toss coin 1; o/w – toss coin 2



Observing the sequence

HHHHT, THTHT, HHHHT, HHTTH

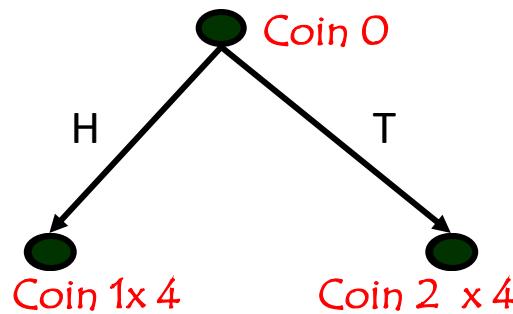
produced by Coin 0 , Coin1 and Coin2

Question: Estimate most likely values for α , p , q (the probability of H in each coin) and the probability to use each of the coins

Supervised Learning

Scenario II

- ❖ Toss coin 0.
If Head – toss coin 1; o/w – toss coin 2



Observing the sequence

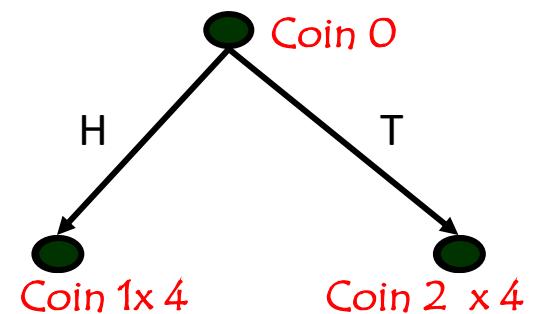
HHHT, HTHT, HHHT, HTTH

produced by ~~Coin 0~~, Coin1 and/or Coin2

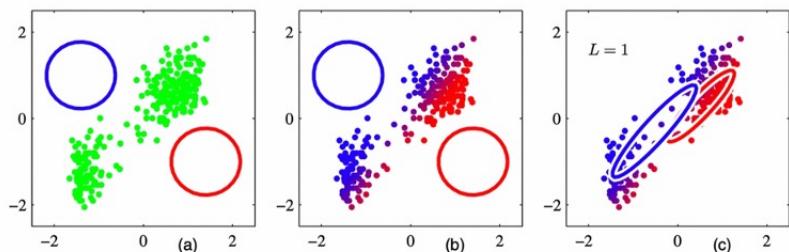
Question: Estimate most likely values for α , p , q (the probability of H in each coin) and the probability to use each of the coins

EM

- ❖ Observing the sequence
HHHT, HTHT, HHHT, HTTH
- ❖ Initialize with a random guess of outcome of Coin 0
HHHHT, THTHT, THHHT, HHTTH
- ❖ Alternative the following two steps; in iter t :
 - ❖ M-step: Estimate $\alpha^{t+1}, p^{t+1}, q^{t+1}$ based on sequence and $P(\text{coin 0} | \text{sequence}; \alpha^t, p^t, q^t)$
 - ❖ E-step: Estimate $P(\text{coin 0} | \text{sequence}; \alpha^{t+1}, p^{t+1}, q^{t+1})$



GMM is a special case



- ❖ Observing data point $\{x_n\}$
- ❖ Initialize with a random guess of clusters r_{nk}
- ❖ Alternative the following two steps; in iter t :
 - ❖ M-step: Estimate w, μ, Σ based on $\{x_n\}$ and $\{r_{n_k}\}$
 - ❖ For Gaussian
$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \mu_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} x_n$$
$$\Sigma_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$
 - ❖ E-step: Estimate $\{r_{n_k}\}$ based on w, μ, Σ
$$\gamma_{nk} = P(z_n = k | x_n) = \frac{w_k N(x_n | \mu_k, \Sigma_k)}{\sum_{k'} w_{k'} N(x_n | \mu_{k'}, \Sigma_{k'})}$$

Generative v.s Discriminative Models

Example



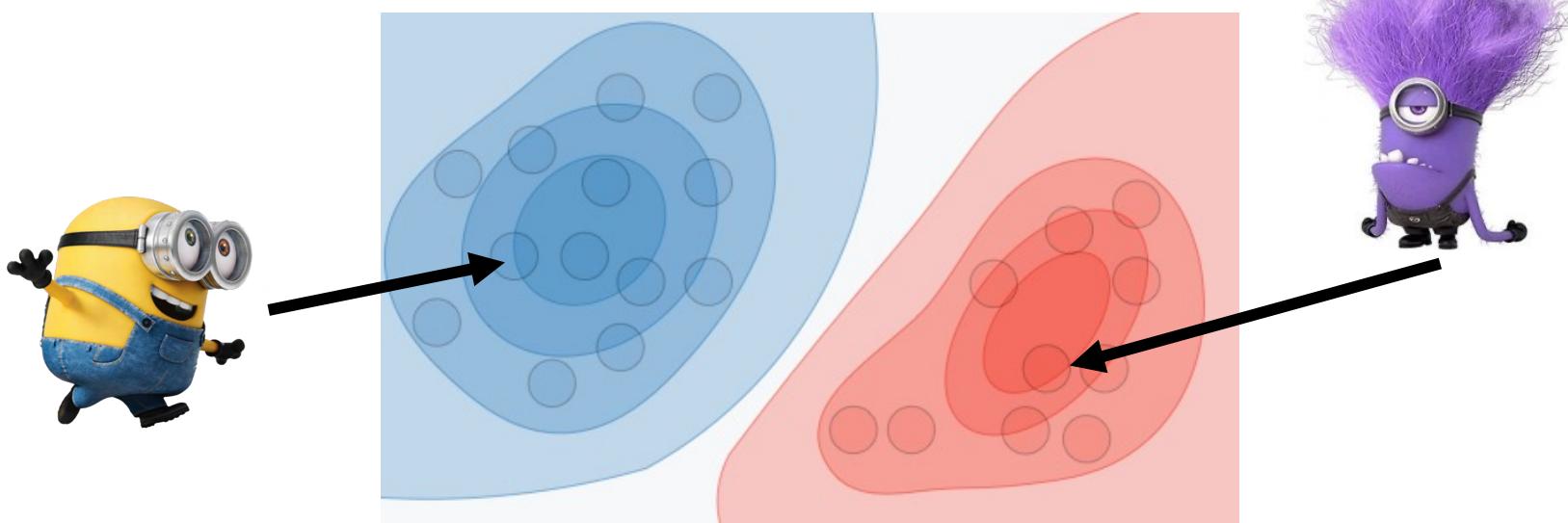
V.S.



Generative (e.g., Naïve Bayes)

- ❖ Estimate $P(Y|X)$ through $P(X, Y)$
- ❖ MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i, x_i)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

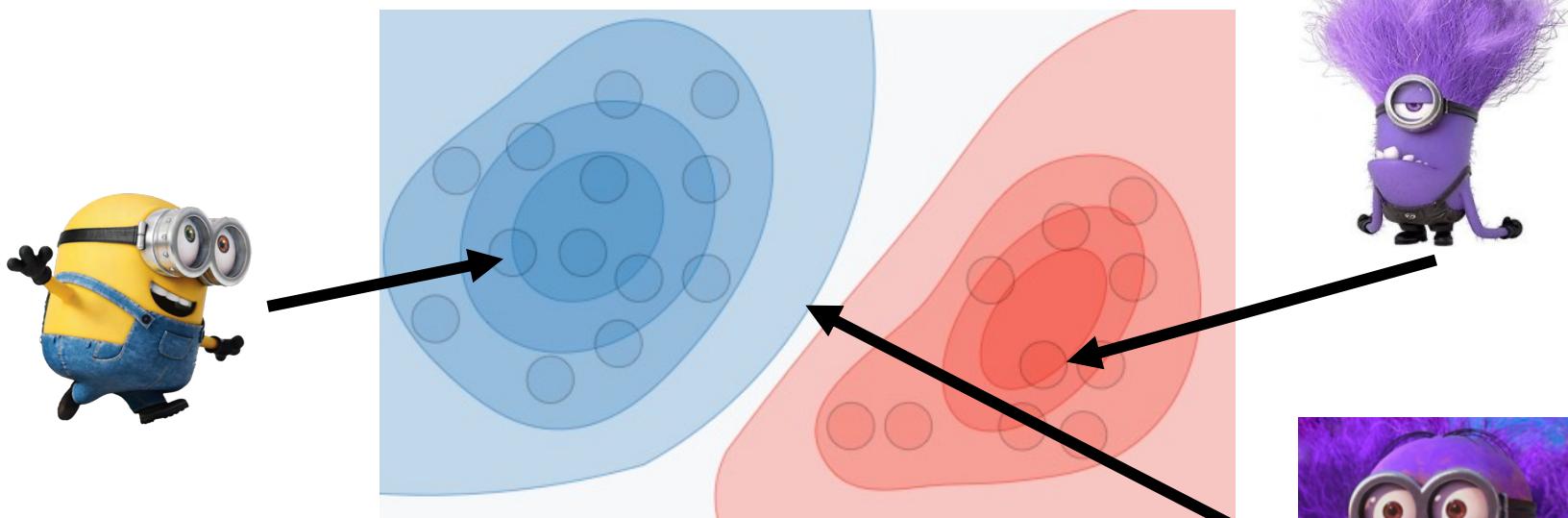


What is the input instances X look like if they belongs to the class Y

Generative (e.g., Naïve Bayes)

- ❖ Estimate $P(Y|X)$ through $P(X, Y)$
- ❖ MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i, x_i)$

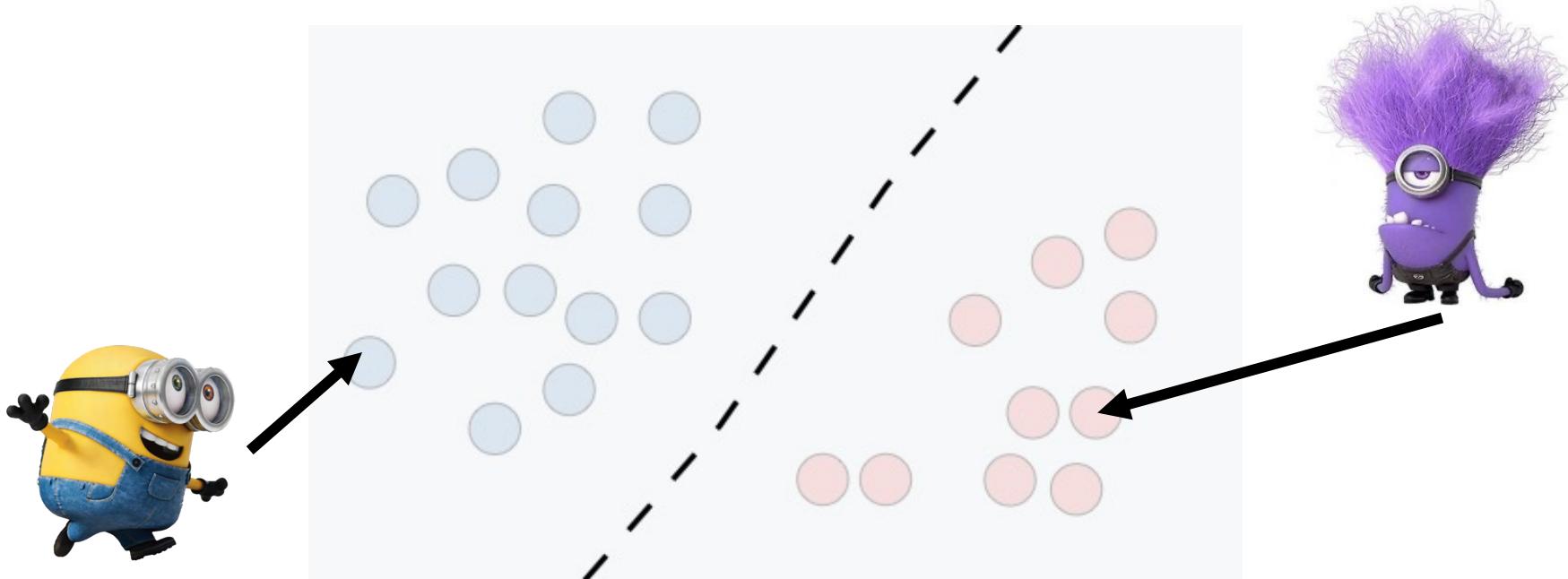
$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$



What is the input instances X look like if they belongs to the

Discriminative (e.g., logistic regression, SVM)

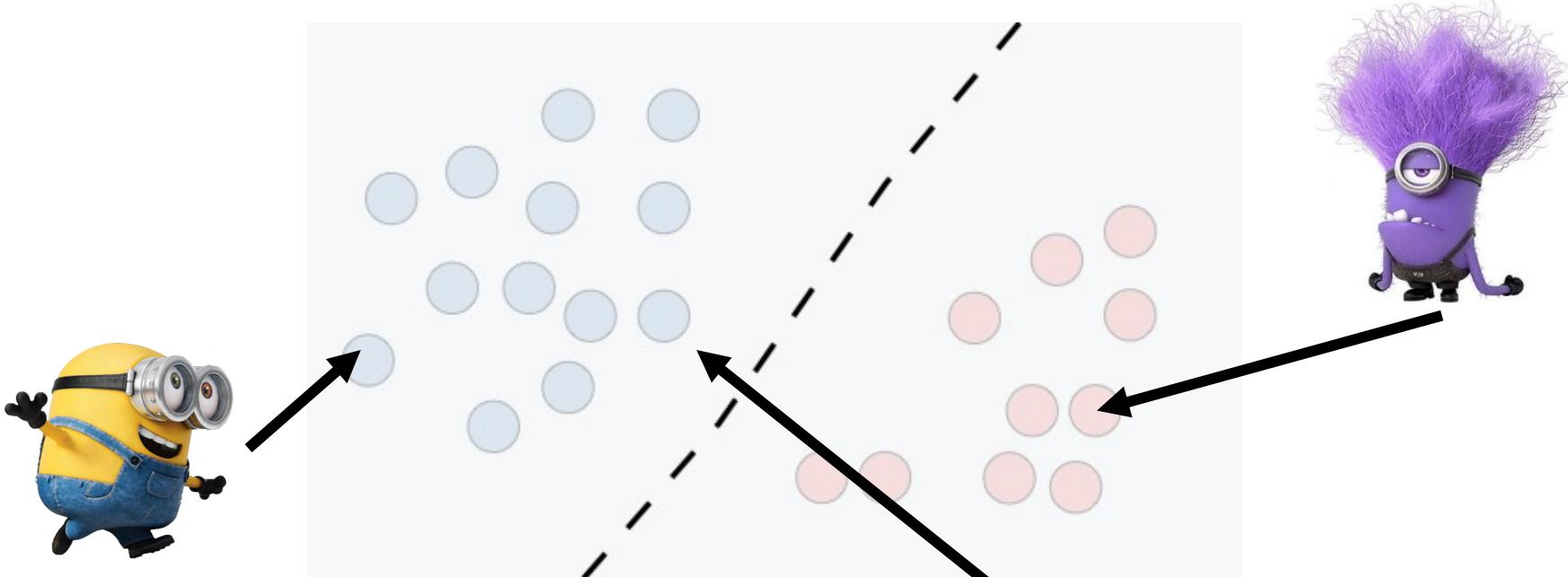
- ❖ Estimate $P(Y|X)$
- ❖ MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i|x_i)$



What features can be used to differentiate between different classes

Discriminative (e.g., logistic regression, SVM)

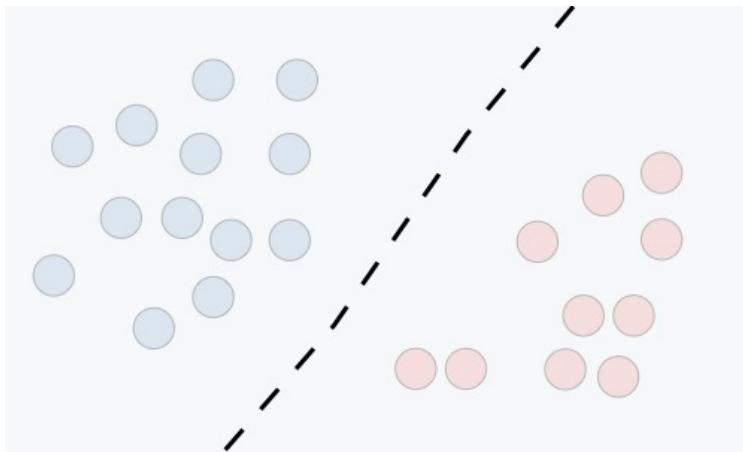
- ❖ Estimate $P(Y|X)$
- ❖ MLE: $\max_{\theta} \sum_{i=1..n} \log P(y_i|x_i)$



What features can be used to differentiate between different classes?



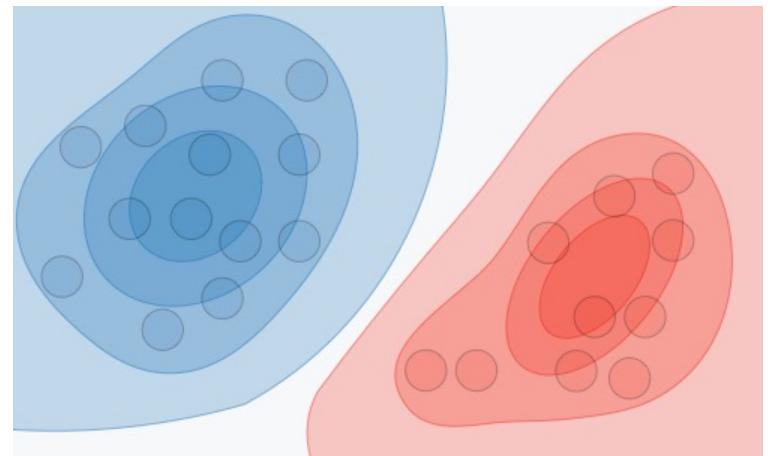
Discriminative vs Generative Models



Estimate $P(Y|X)$

$$\text{MLE: } \max_{\theta} \sum_{i=1..n} \log P(y_i|x_i)$$

Discriminative



Estimate $P(Y|X)$ through $P(X, Y)$

$$\text{MLE: } \max_{\theta} \sum_{i=1..n} \log P(y_i, x_i)$$

Generative

What features can be used to differentiate between different classes

A retrospective look at the course

Learning = generalization

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .”

Tom Mitchell (1999)

We saw different “models”

what kind of a function should a learner learn

- ❖ K-NN
- ❖ Linear classifiers
- ❖ Decision trees
- ❖ Non-linear classifiers, feature transformations, neural networks

Different learning protocols

- ❖ **Supervised learning**
 - ❖ A *teacher* supplies a collection of examples with labels
 - ❖ The *learner* has to learn to label new examples using this data
- ❖ **Unsupervised learning**
 - ❖ No *teacher*, *learner* has only unlabeled examples

The theory of machine learning

Mathematically defining learning

- ❖ Entropy (information theorem)
- ❖ Probably Approximately Correct (PAC) Learning
- ❖ MLE, MAP
- ❖ SGD (optimization)

Some general recipes in ML

- ❖ Empirical Loss Minimization
 - ❖ Define loss and regularizer -> SGD
- ❖ Deep Learning
 - ❖ Define model architecture -> SGD
- ❖ Probabilistic models
 - ❖ Define model $P(Y|X)$ or $P(X,Y)$
-> MLE, MAP -> SGD, or closed form

Practical advices

Bias and variance

Every learning algorithm requires assumptions about the hypothesis space.

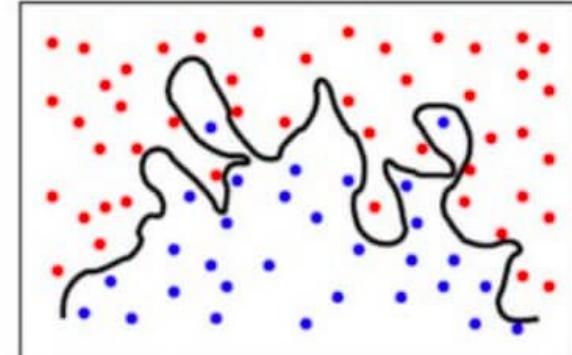
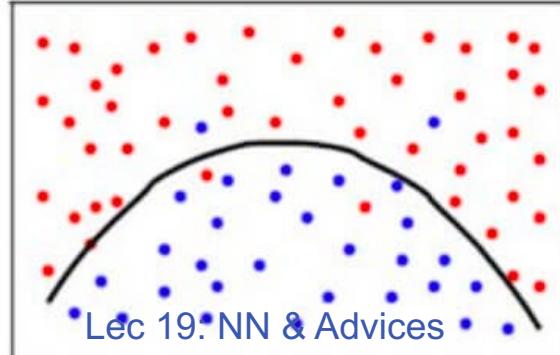
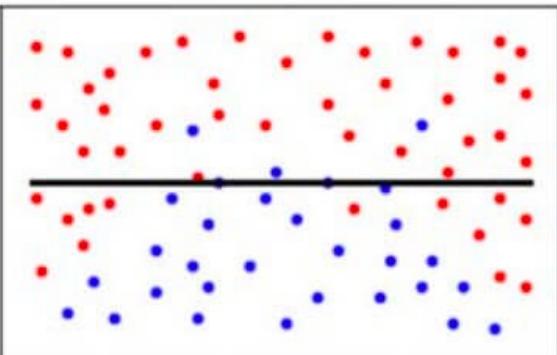
Eg: “My hypothesis space is

- ❖ ...linear”
- ❖ ...decision trees with 5 nodes”
- ❖ ...deep neural network with 12 layers”

Underfitting



Overfitting



Managing bias and variance

- ❖ Decision trees of a fixed depth
 - ❖ Increasing depth decreases bias, increases variance
- ❖ SVMs
 - ❖ Stronger regularization (i.e., smaller C in our formulation) increases bias, decreases variance
- ❖ K nearest neighbors
 - ❖ Increasing k generally increases bias, reduces variance
- ❖ Other approaches: Drop out, Ensemble, etc...

Tune your parameters!!

- ❖ Always tune parameters when comparing different settings / algorithms
- ❖ Never tune your parameters on the test set



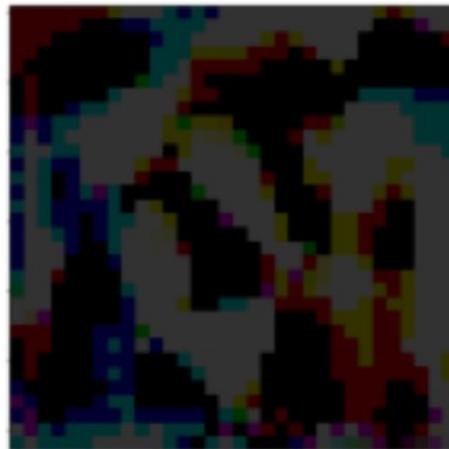
ML is more than Curve Fitting

- ❖ Real world is adversarial



93%, 20 Km/h Sign

+ ϵx



$sign(\nabla * J(\theta, x, y))$

=



90%, 80 Km/h Sign



ML is more than Curve Fitting

- ❖ Build system works well for everyone

The screenshot shows a digital interface for uploading a passport photo. At the top, there is a green button labeled "Select photo" with a white arrow icon, followed by a question mark icon. Below this, a red box highlights an error message: "The photo you want to upload does not meet our criteria because: Subject eyes are closed". A red arrow points from this message down to a blue box containing the text "Subject eyes are closed". To the right of the error message, it says "Please refer to the technical requirements. You have 9 attempts left." and "Check the photo [requirements](#)". Below that, it says "Read more about [common photo problems and](#)".

**Subject eyes
are closed**

If you wish to [contact us](#) about the photo, you must provide us with the reference number given above.

Please print this information for your records.

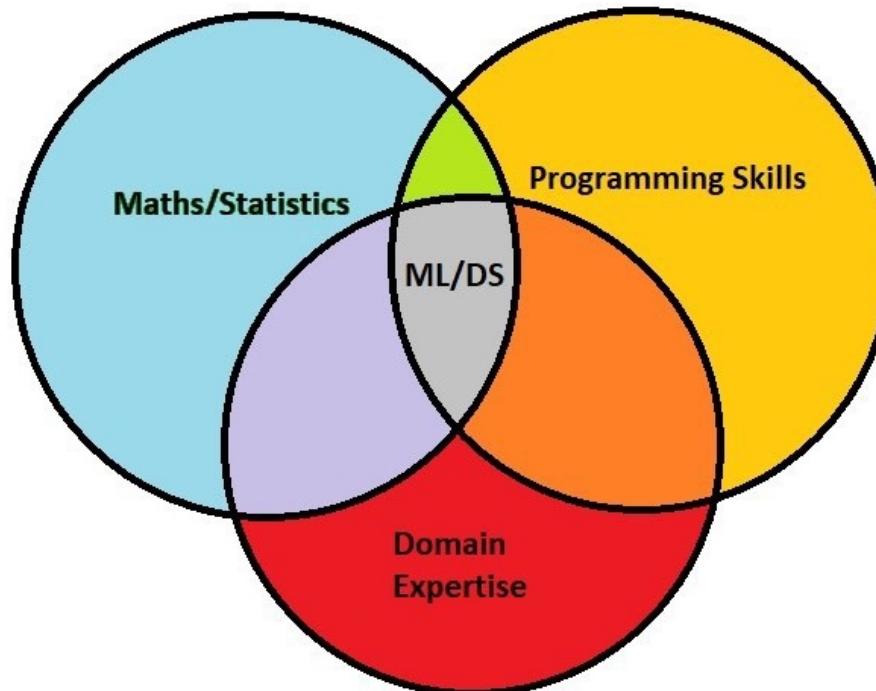
Il need to
TCHA security
81

A portrait photograph of Richard Lee, a young man with dark hair, wearing a black turtleneck sweater and a patterned purple jacket.

A screenshot of New Zealand man Richard Lee's passport photo rejection notice, supplied to Reuters December 7, 2016. Richard Lee/Handout via REUTERS

ML is more than Curve Fitting

- ❖ Domain knowledge is important!



Thank you

- ❖ Please log in MyUCLA for course survey

