

DS-GA 1011: Fundamentals of Natural Language Processing

Lecture 1: Introduction & Text Classification

Greg Durrett and Shauli Ravfogel



NYU

Today

- ▶ Introduction to this course
- ▶ Brief History of NLP
- ▶ Text classification

Logistics

- ▶ Lecture: Wednesday 10:15am - 12:15pm, 36 E 8th St (Cantor Film Ctr) Room 102
- ▶ Section: Thursday 10:15am–11:05am · Cantor, Room 101
- ▶ Course website (including **syllabus**): <https://gregdurrett.github.io/courses/fa2026/index.html>
- ▶ Discord, Gradescope: link in the BrightSpace.

Course Staff

- ▶ Greg Durrett
- ▶ Office hours: TBD
- ▶ Shauli Ravfogel
- ▶ Office hours: Wed 2:00-3:00, room 611 @ 60 5th - **Please mail in advance!**



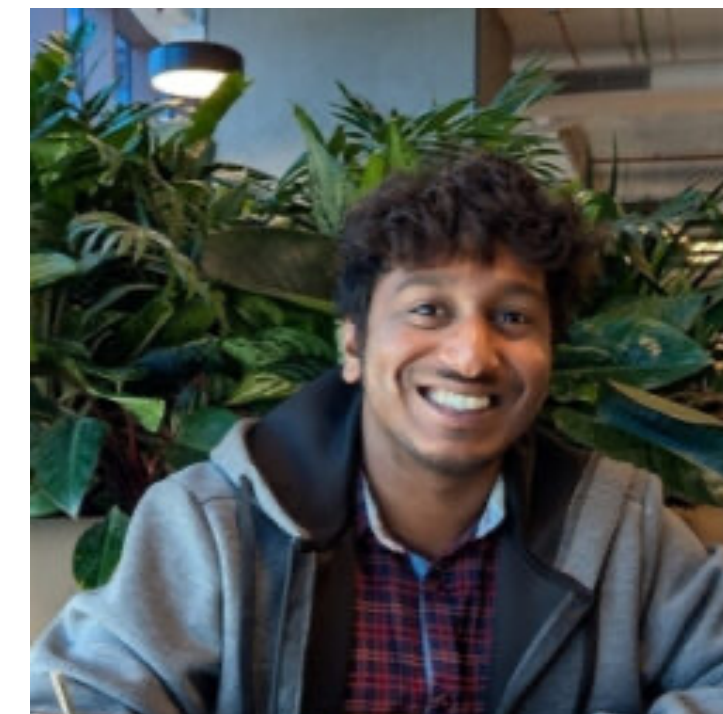
Shashwat Singh
ss20428@nyu.edu
Office hour: Tuesday 2:00 - 3:00 pm



Yihuai Hong
yihuaihong@nyu.edu
Office hour: Thursday, 2–3pm



Wenxuan Ding
yc7320@nyu.edu
Office hour: Monday 2:30 -3:30 pm



Varun Yerram
vy2142@nyu.edu
Office hour: Thursday 4:00 pm - 5:00 pm

What is this course

- ▶ Introductory-level class for modern NLP methods
- ▶ Learn the technical tools with natural language processing
- ▶ Equip you to further explore language technology
- ▶ Understand the capabilities and limitations of technology

What this course is not about

- ▶ General machine learning or deep learning course
 - ▶ We focus on modeling language phenomena
 - ▶ Learn how to formalize and frame NLP tasks as learning problems
 - ▶ We will cover recent deep learning techniques
- ▶ Project-based capstone class
- ▶ Advanced specialized seminar class

Prerequisites

- ▶ Probability, statistics, linear algebra (DS-GA.1002, MATH-UA.140, MATH-UA.235)
- ▶ Algorithms and data structure (CSCI-UA.102)
- ▶ Basic knowledge in machine learning (DS-GA.1003, CSCI-UA.0473).
We will not spend a significant amount of time on machine learning basics so some prior exposure to the supervised learning framework (e.g., loss functions, SGD) is expected.

Format and Accessibility

- ▶ Lectures will include times for discussion, in-class exercises, and questions.
- ▶ Required equipment: some way to do homework through..
 - ▶ Laptop / an access to server machine to do programming assignments.
 - ▶ GPU not required for earlier assignments, helpful for final projects and the last assignment.
 - ▶ We will offer GPU credits on NYU's server machines. More on this later!

What will be covered in the section

- ▶ Supplementary materials to lectures
- ▶ Hands-on exercises (Jupyter notebooks) on the concepts covered in the lecture
- ▶ Tutorials for using server machines
- ▶ Additional topics not covered in the lecture

Coursework: 1/4

- ▶ Homework: 20% of final grade (4 assignments)
- ▶ HW1-4: Homework with programming parts (generally released two weeks before it is due).
- ▶ Each homework will count equally to the final grade.

Coursework: 2/4

- ▶ Homework Quizzes: 30% of final grade (3 quizzes)
 - ▶ Conducted during the first 15 minutes of the lecture.
 - ▶ Three quizzes total: Sept 16, Oct 7, Nov 17.
 - ▶ Closed-book quiz covering materials covered in the lectures.
 - ▶ The quizzes will consist of questions (True/False, multiple choice, and short answer) that evaluate concepts covered during lectures in the weeks preceding the quiz. Prior to the first quiz, the course staff will post sample quiz questions to help with your preparation.

Coursework: 3/4

- ▶ Midterm Exam: 30% of final grade, in-class
 - ▶ Date: October 28
 - ▶ Covers lectures, labs & home assignments
 - ▶ Students will be allowed one standard letter (8.5" x 11") page of notes during exams

Coursework: 4/4

- ▶ Final Project (20% of grade)
 - ▶ Each project can be completed by a team of 2 to 3 students.
 - ▶ Open-ended projects applying techniques learned in the class.
 - ▶ A project proposal will be submitted **by Nov 4** and feedback will be provided by the course staff

Late Day Policy

- ▶ You have 5 slip days (24 hour extensions) throughout the semester to use for **assignments**. You cannot use it for project-related deadlines.
- ▶ For each day late an assignment is turned in not covered by a slip day or negotiated extension, 15% of the received grade for that assignment will be deducted

Course Materials

- ▶ No need to buy textbooks!
- ▶ Readings will be posted in the course website schedule.
- ▶ Often chapters from textbooks (freely available as PDFs online), or recent research papers.

Speech and Language Processing (3rd ed. draft)

[Dan Jurafsky](#) and [James H. Martin](#)

<https://web.stanford.edu/~jurafsky/slp3/>

Academic Honesty

- ▶ You may work in groups, but your final writeup and code **must be your (and your teammate's) own**
- ▶ Do **NOT** share code with others!
- ▶ If you are unsure, ask!
- ▶ For team projects, work TOGETHER with your partner, instead of simply dividing
- ▶ You can use AI assistants (including for coding assistants). However, you should document your usages briefly when you submit your work, and you are expected to understand every line of code or answer.

What will be covered in the lectures

Week 1	Introduction + Text classification	HW1: Feedforward Classifier	Quiz 1	
Week 2	Word embeddings			
Week 3	Sequence Modeling	HW2: Transformers	Quiz 2	
Week 4	LLMs			
Week 5	Generation		Quiz 3	
Week 6	Pre-training	HW3: LLM Prompting		Proposal
Week 7	Post-training	HW4: LLM Fine-tuning		
Week 8	Scaling			
Week 9	Evaluation & Benchmarking	Report		
Week 10	Benchmarking and Evaluation			
Week 11	Interpretability			
Week 12	Agents			
Week 13	TBD			

Tips for the course

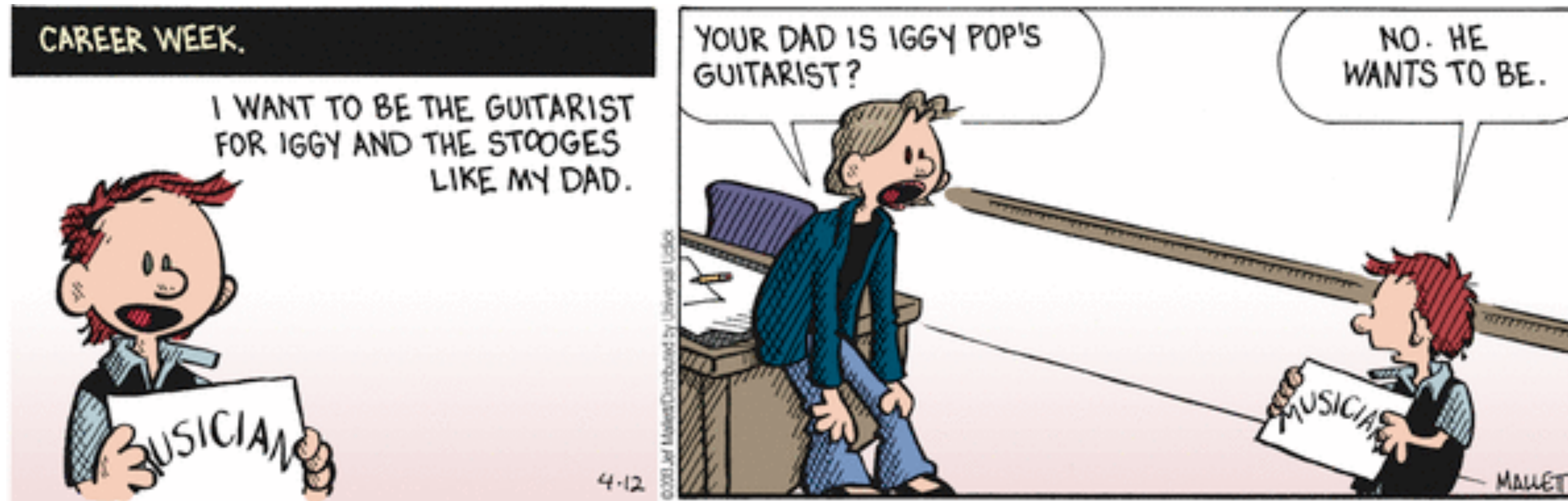
- ▶ Do NOT take this course with overloaded schedule
- ▶ Do NOT skip prerequisites
- ▶ Don't procrastinate on project / assignments

Today

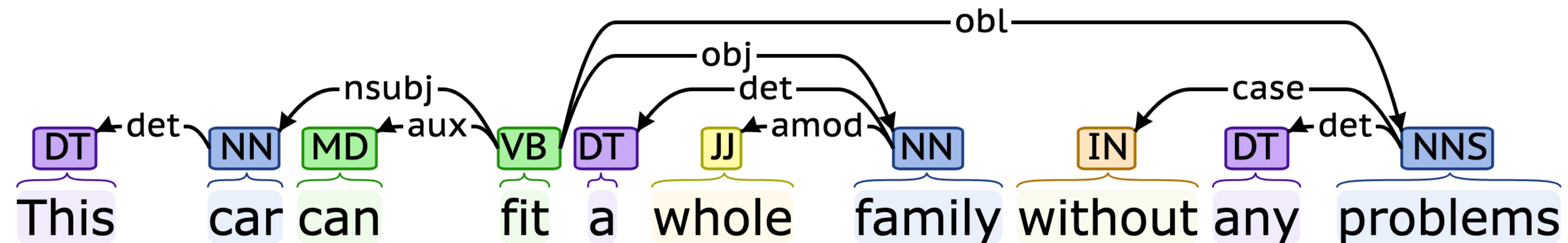
- ▶ Introduction to this course
- ▶ Brief introduction to NLP
 - ▶ Challenges of NLP
 - ▶ History of NLP
- ▶ Text classification

Why is language hard?

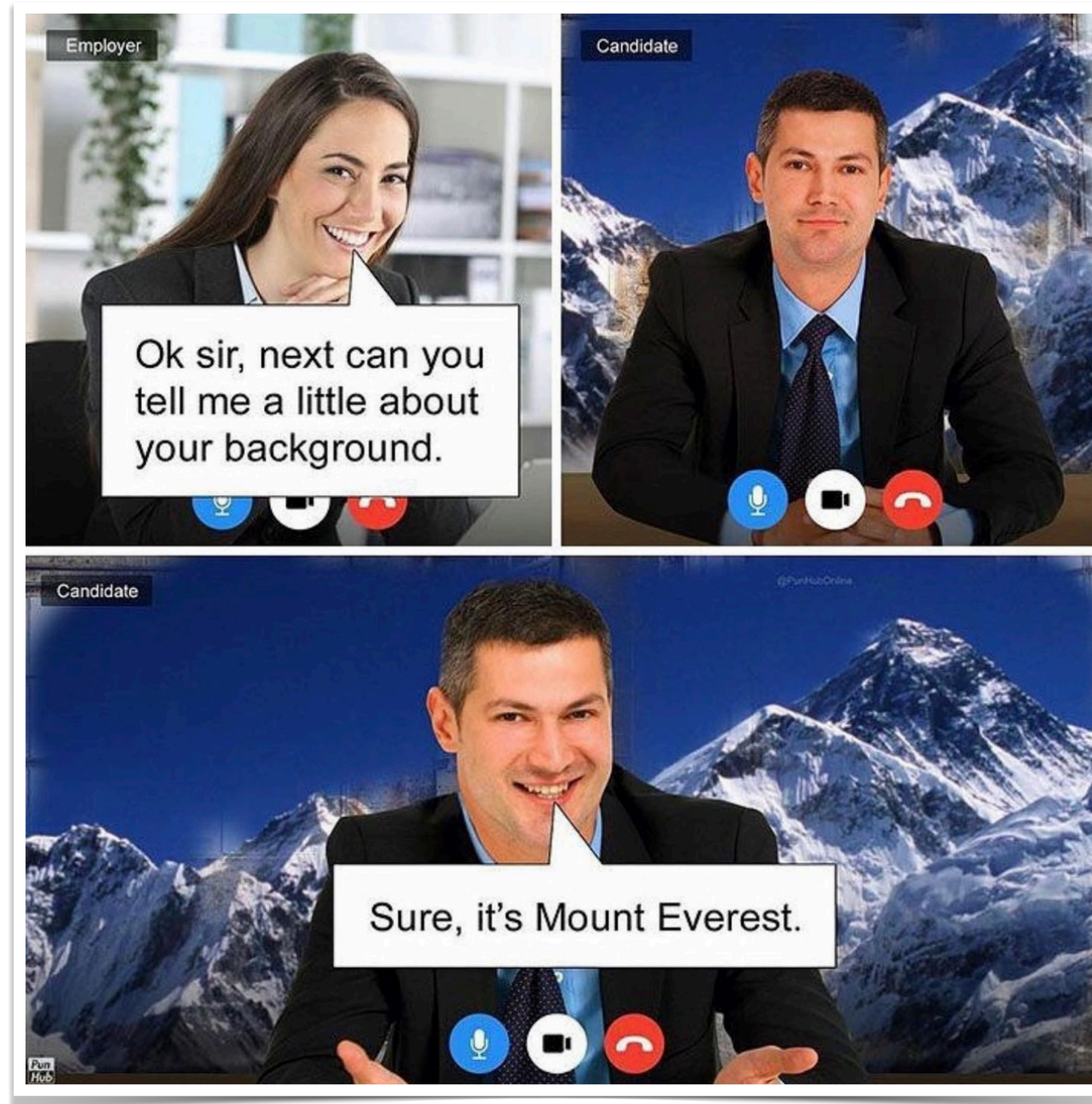
Language is Ambiguous!



Types of Ambiguity: Syntactic



Types of Ambiguity: Semantic



Language Ambiguity

At last, a computer that understands you like your mother.

- ▶ Direct meanings:
 - ▶ It understands you like your mother (does) [presumably well]
 - ▶ It understands (that) you like your mother
- ▶ “*mother*” could mean:
 - ▶ a woman who has given birth to a child
 - ▶ a stringy slimy substance consisting of yeast cells and bacteria; is added to cider or wine to produce vinegar
- ▶ Context matters, e.g. what if previous sentence was:
 - ▶ Wow, Amazon predicted that you would need to order a big batch of new vinegar brewing ingredients.

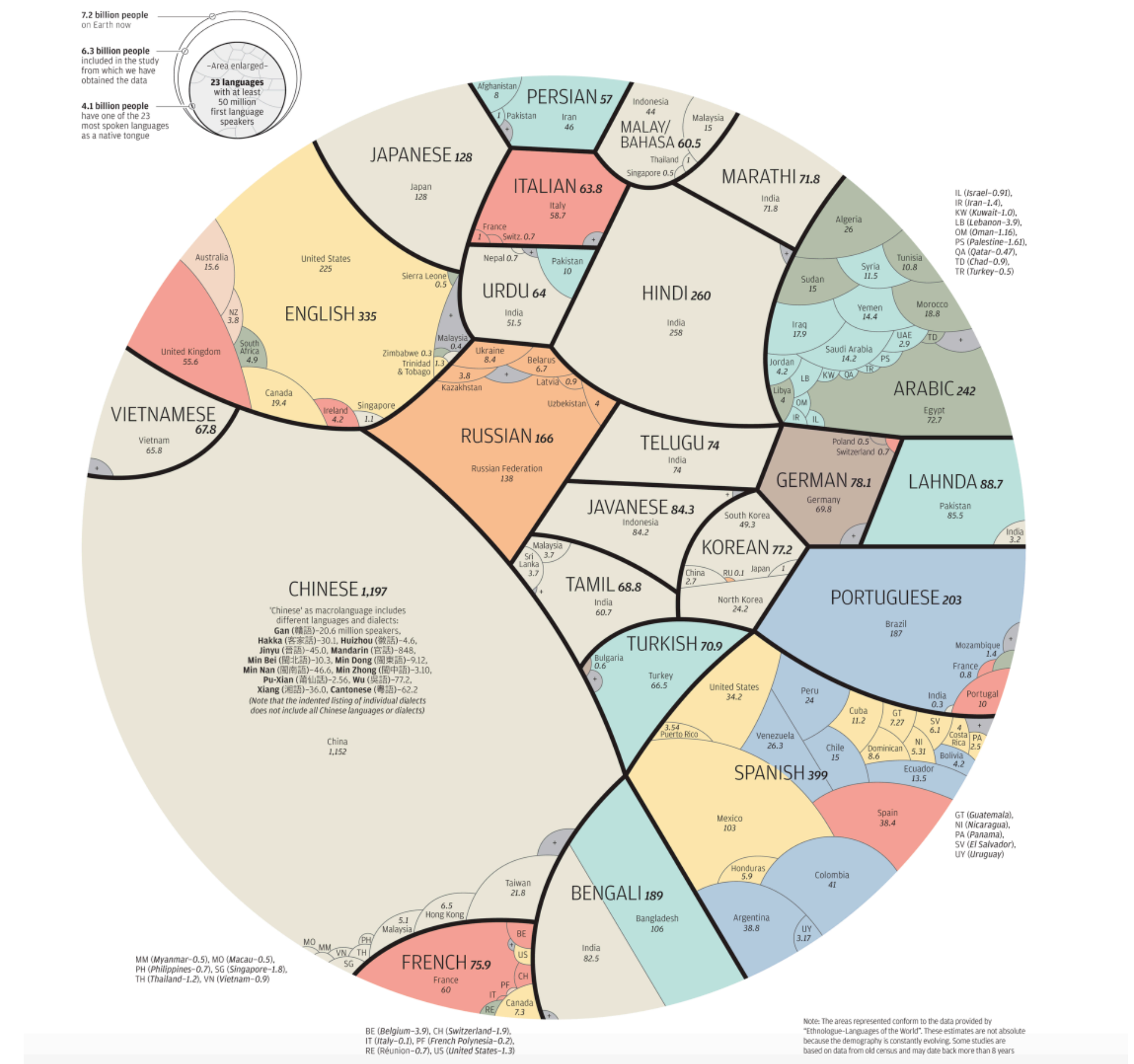
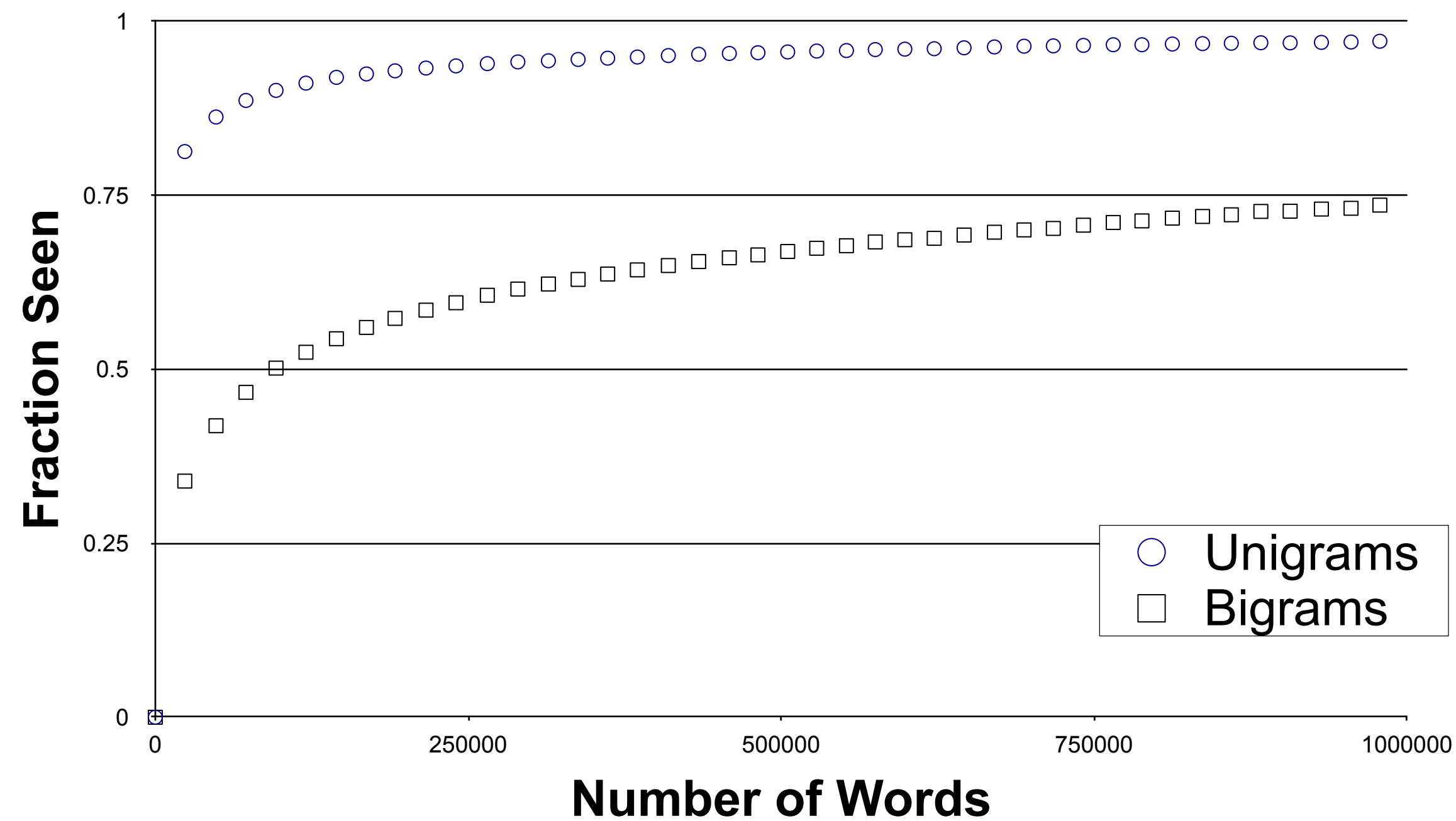
Situated Understanding of Language



What the British say	What others understand	What the British mean
I hear what you say	He accepts my point of view	I disagree and do not want to discuss it further
With the greatest respect...	He is listening to me	I think you are an idiot
That's not bad	That's poor	That's good
That is a very brave proposal	He thinks I have courage	You are insane
Quite good	Quite good	A bit disappointing
I would suggest...	Think about the idea, but do what you like	Do it or be prepared to justify yourself
Oh, incidentally / by the way...	That is not very important	The primary purpose of our discussion is...
I was a bit dissapointed that	It doesn't really matter	I am annoyed that
Very interesting	They are impressed	That is clearly nonsense.
I'll bear it in mind	They will probably do it	I've forgotten it already
I'm sure it's my fault	Why do you they think it was their fault?	It's your fault
You must come for dinner	I will get an invitation soon	It's not an invitation, I'm just being polite

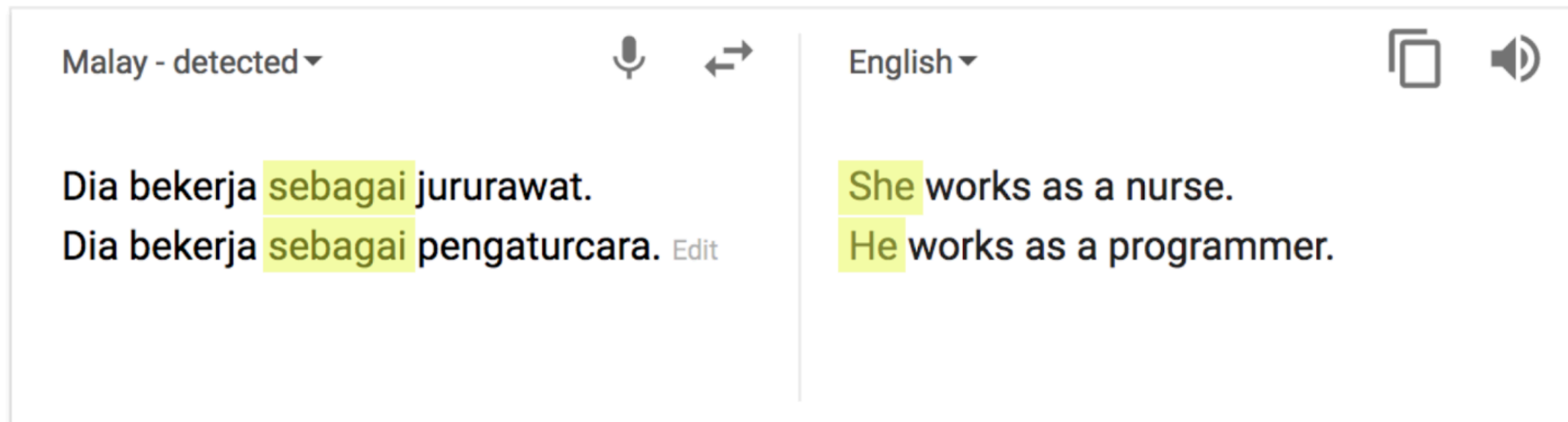
Sparsity of the data

- ▶ New word constantly comes up!
- ▶ Even worse for low resource languages



Challenges in Machine Translation

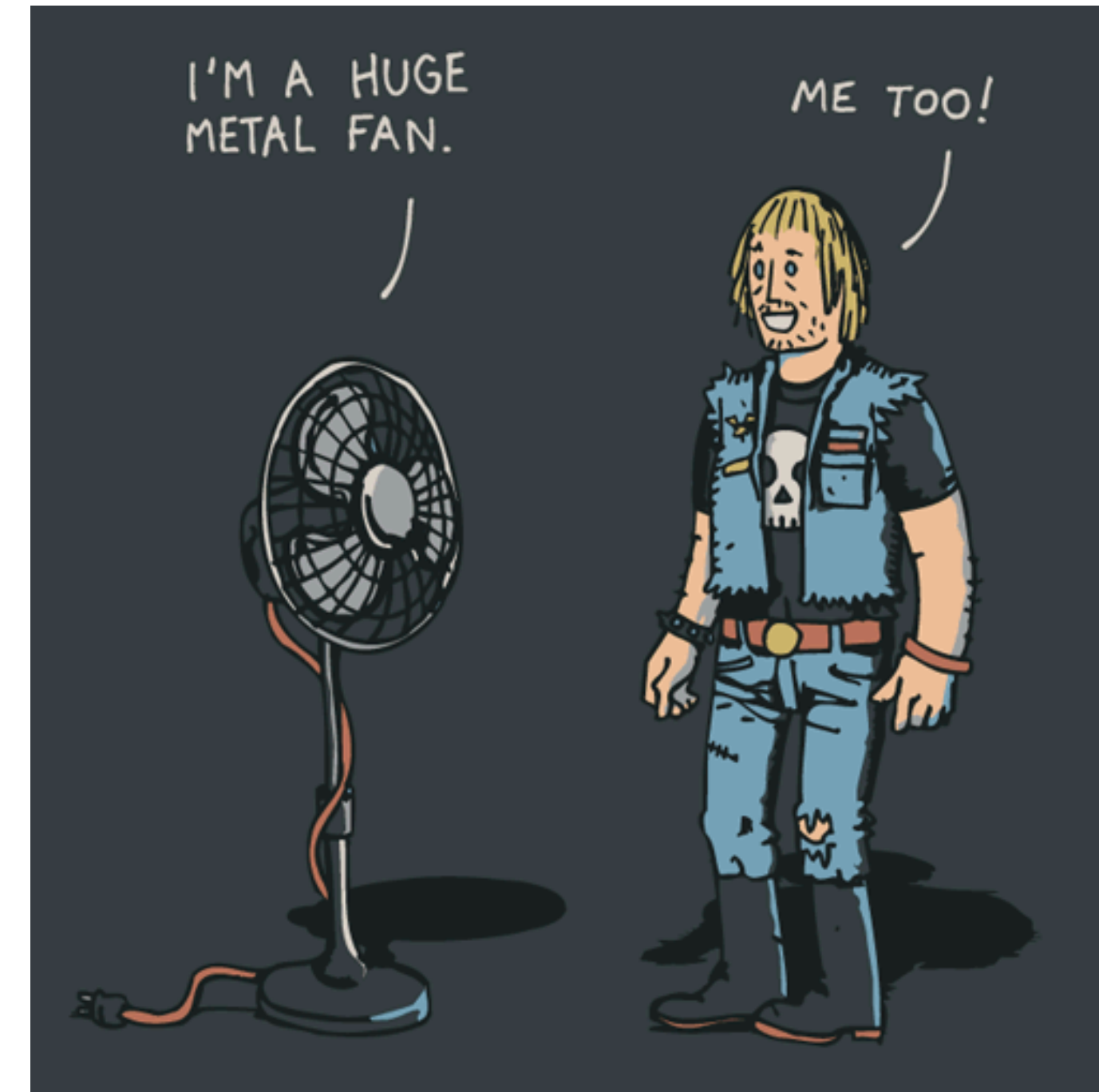
- Ambiguity can lead to biases



sebagai is a pronoun that does not specify a gender "he" or "she"

Pronouns **she** and **he** were chosen by the NMT system based on neighboring words.

- Lacking common sense



"I'm a huge metal fan!"

"Je suis un énorme ventilateur en métal" ("I'm a large ventilator made of metal.")

Today

- ▶ Introduction to this course
- ▶ Brief introduction to NLP
 - ▶ Challenges of NLP
 - ▶ History of NLP
- ▶ Text classification

Brief History of NLP

- ▶ 1940-50s: introducing probability
- ▶ 1950-80s: expert hand-written rules
- ▶ 1990s: statistical model coming back
- ▶ 2010s: large-scale models

Analyzing the dependent probabilities of letters and words appearing in combination with each other: statistical modeling of English

First attempt: Statistical Modeling of Language

- ▶ The Shannon Game:
- ▶ How well can we predict the next word?

When I eat pizza, I wipe off the _____

Many children are allergic to _____

I saw a _____

grease 0.5
sauce 0.4
dust 0.05
....
mice 0.0001
....
the 1e-100

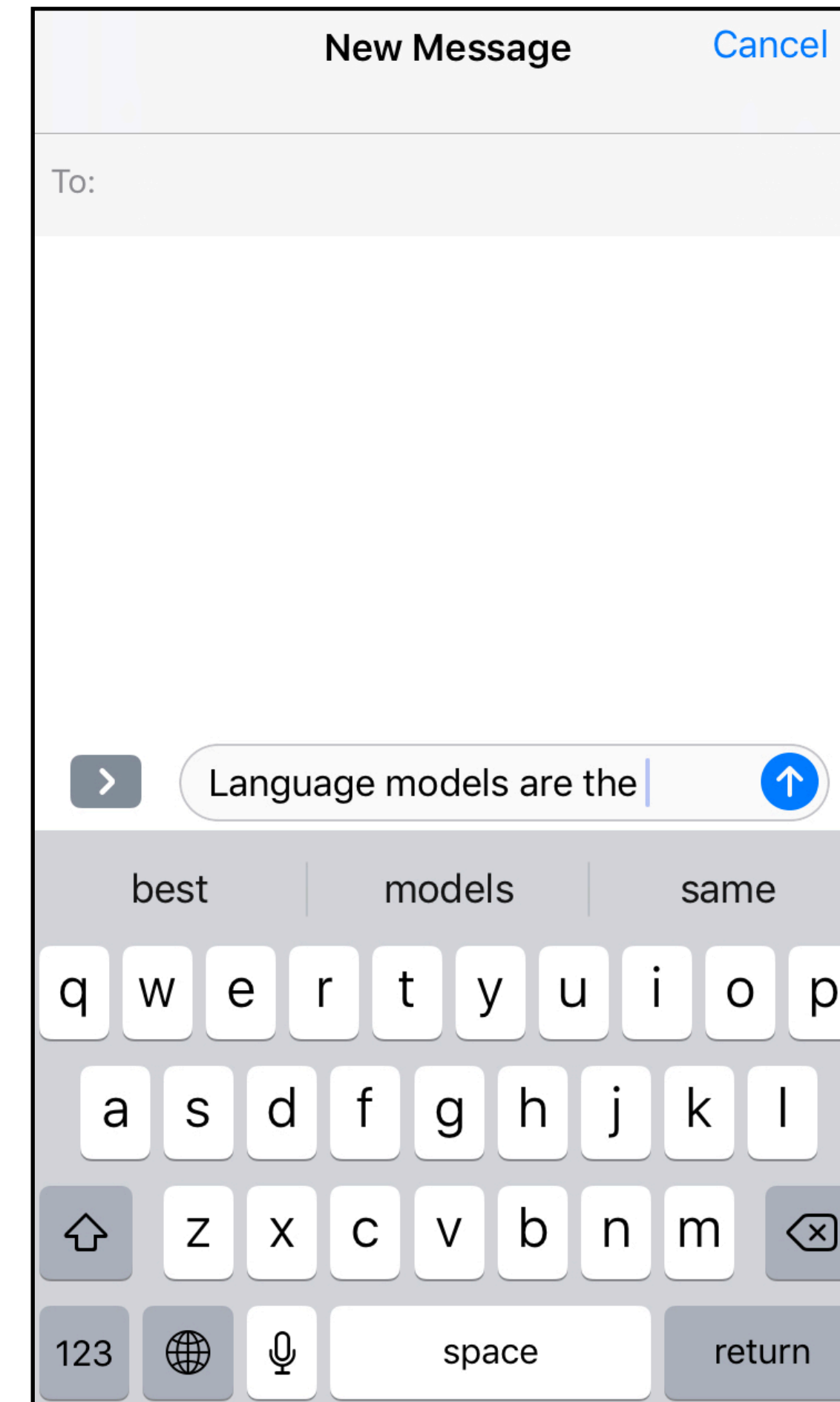


Claude Shannon

- ▶ How good are we doing?
- ▶ Compute per word log likelihood (total n words):

$$l = \frac{1}{n} \sum_{i=1}^n \log P(x_i | x_1, x_2 \dots x_{i-1})$$

Predicting next word is useful!



$P(\text{high school } \mathbf{principal}) > P(\text{high school } \mathbf{principle})$

Brief History of NLP

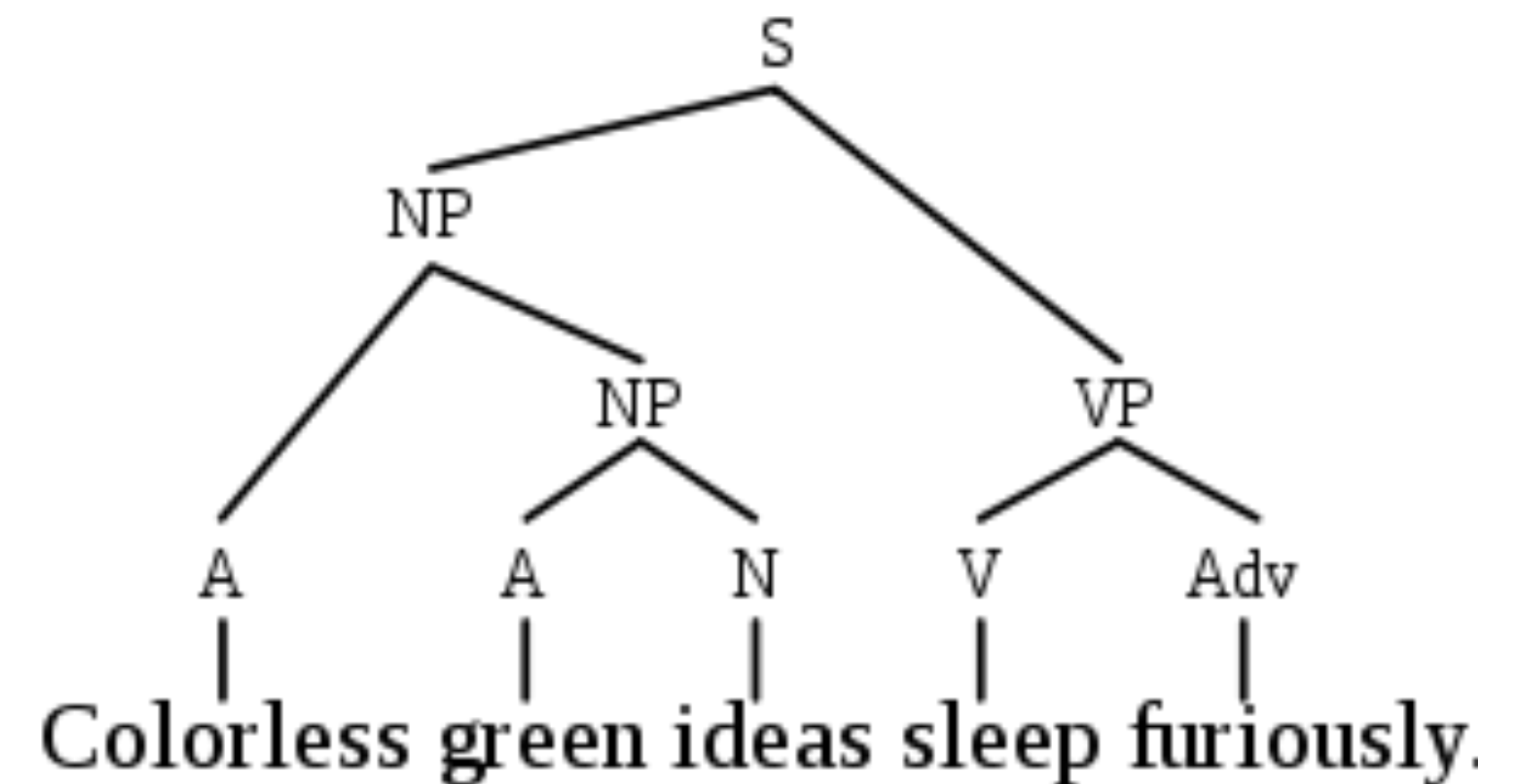
- ▶ 1940-50s: introducing probability
- ▶ **1950-80s: expert hand-written rules**
- ▶ 1990s: statistical model coming back
- ▶ 2010s: large-scale models

Second Attempt: Writing Rules

- (1) Colorless green ideas sleep furiously.
- (2) Furiously sleep ideas green colorless.

It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) had ever occurred in an English discourse. Hence, in any statistical model for grammaticality, these sentences will be ruled out on identical grounds as equally "remote" from English. Yet (1), though nonsensical, is grammatical, while (2) is not."

(Chomsky 1957)



Brief History of NLP

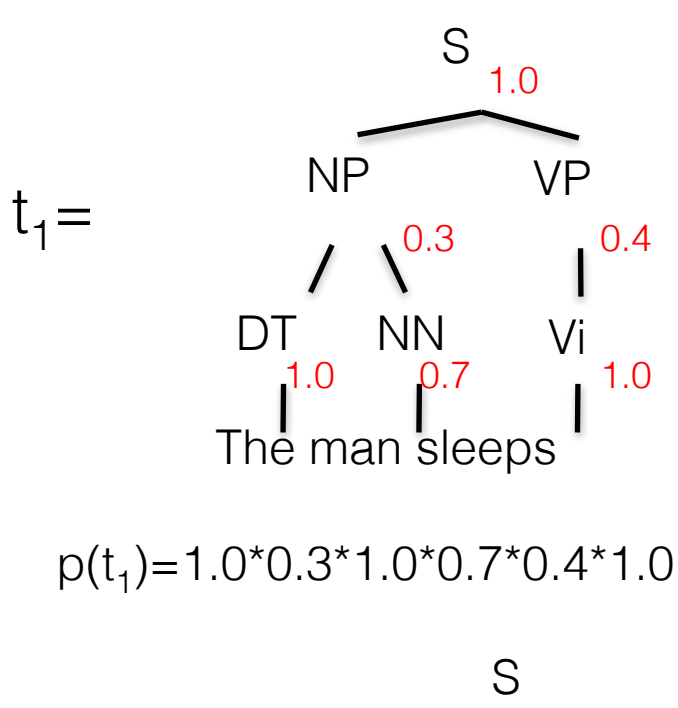
- ▶ 1940-50s: introducing probability
- ▶ 1950-80s: expert hand-written rules
- ▶ **1990s: statistical model coming back**
- ▶ 2010s: large-scale models

Annotating Data: Penn Treebank (1993)

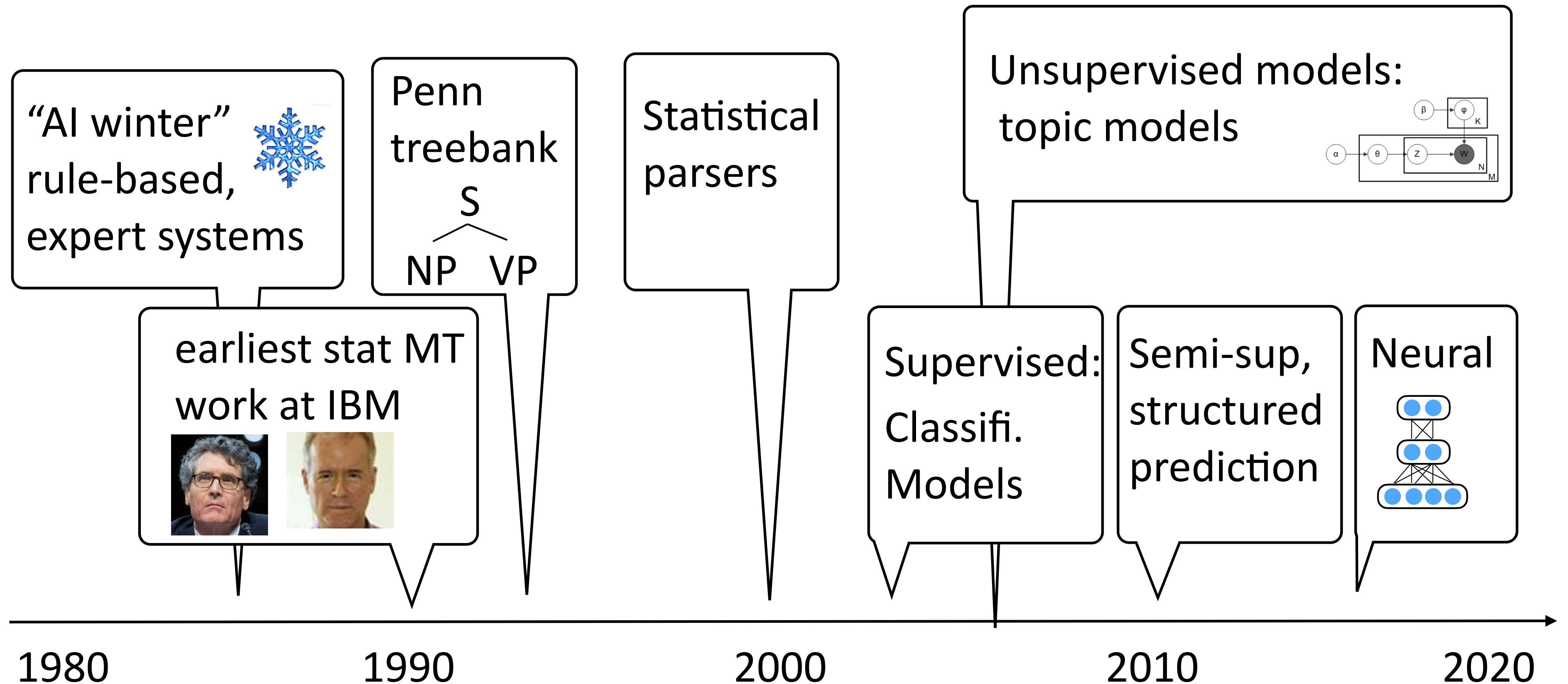
((S
 (NP-SBJ (DT The) (NN move))
 (VP (VBD followed)
 (NP
 (NP (DT a) (NN round))
 (PP (IN of)
 (NP
 (NP (JJ similar) (NNS increases))
 (PP (IN by)
 (NP (JJ other) (NNS lenders)))
 (PP (IN against)
 (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
 (, ,)
 (S-ADV
 (NP-SBJ (-NONE- *))
 (VP (VBG reflecting)
 (NP
 (NP (DT a) (VBG continuing) (NN decline))
 (PP-LOC (IN in)
 (NP (DT that) (NN market))))))
 (. .)))

- ▶ 50,000 annotated sentences!
- ▶ Usual set-up:
 - ▶ 40,000 training
 - ▶ 2,400 test

S	⇒	NP	VP	1.0
VP	⇒	Vi		0.4
VP	⇒	Vt	NP	0.4
VP	⇒	VP	PP	0.2
NP	⇒	DT	NN	0.3
NP	⇒	NP	PP	0.7
PP	⇒	IN	NP	1.0



A brief history of statistical NLP



Related Fields

- ▶ Computational Linguistics:
 - ▶ Use computational tools to study **language**
- ▶ Cognitive Science
 - ▶ Figuring out how the human brain works
 - ▶ Includes parts that do language
- ▶ Speech
 - ▶ Mapping audio ***signals*** to text
 - ▶ Traditionally separate from NLP, converging?
 - ▶ Traditionally two components: acoustic models and language models

Today

- ▶ Introduction to this course
- ▶ Brief History of NLP
- ▶ Text classification

Classification

- ▶ Introduction: Classification Problem
- ▶ Learning a classifier
 - ▶ Log-linear classifier (maximum entropy models)
 - ▶ Perceptron
 - ▶ Feedforward Neural Network

Classification

- ▶ Input: some text x (e.g., a sentence, document), usually a sequence of words of length $|x|$

$$x = w_1, w_2, \dots, w_{|x|}$$

- ▶ Output: a label y (from a finite, pre-defined label set C)

$$y \in c_1, c_2, \dots, c_N$$

- ▶ Classifier:

$$f : x \rightarrow y$$

Classification is useful!

Task	X	Y	
Sentiment Analysis	product review	{positive, negative}	Binary classification
Topic Classification	documents	{politics, sports, arts, ...}	
Author Identification	Text (could be tweet, books)	{Shakespear, Obama, ...}	Multiclass classification
Medical Diagnosis	doctors' note	{flu, cold, allergy, COVID,..}	
... many others			

Image Classification



→ Dog



→ Car

- ▶ Thousands of classes (ImageNet)

Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.



Lance Edward Armstrong is an American former professional road cyclist



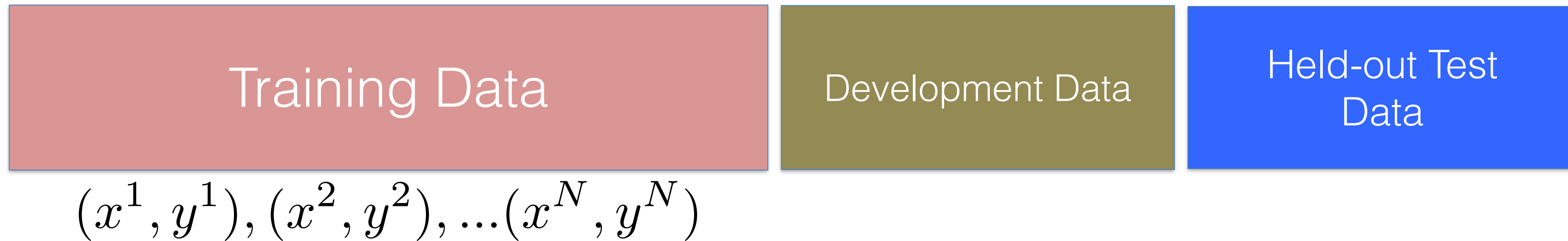
Armstrong County is a county in Pennsylvania...

?

?

- ▶ 6,000,000 classes (all articles in Wikipedia)

Supervised Learning



- ▶ We learn a model from annotated ***data***
- ▶ Limitations?
 - ▶ Annotations are often expensive, noisy
 - ▶ Data shifts (changing label sets, input texts)

Evaluation

Model Prediction	True Label		
		1	0
	1	True positive	False positive
	0	False negative	True negative

► Accuracy

$$\frac{\# \text{ true positive} + \# \text{ true negative}}{\# \text{ all examples}}$$

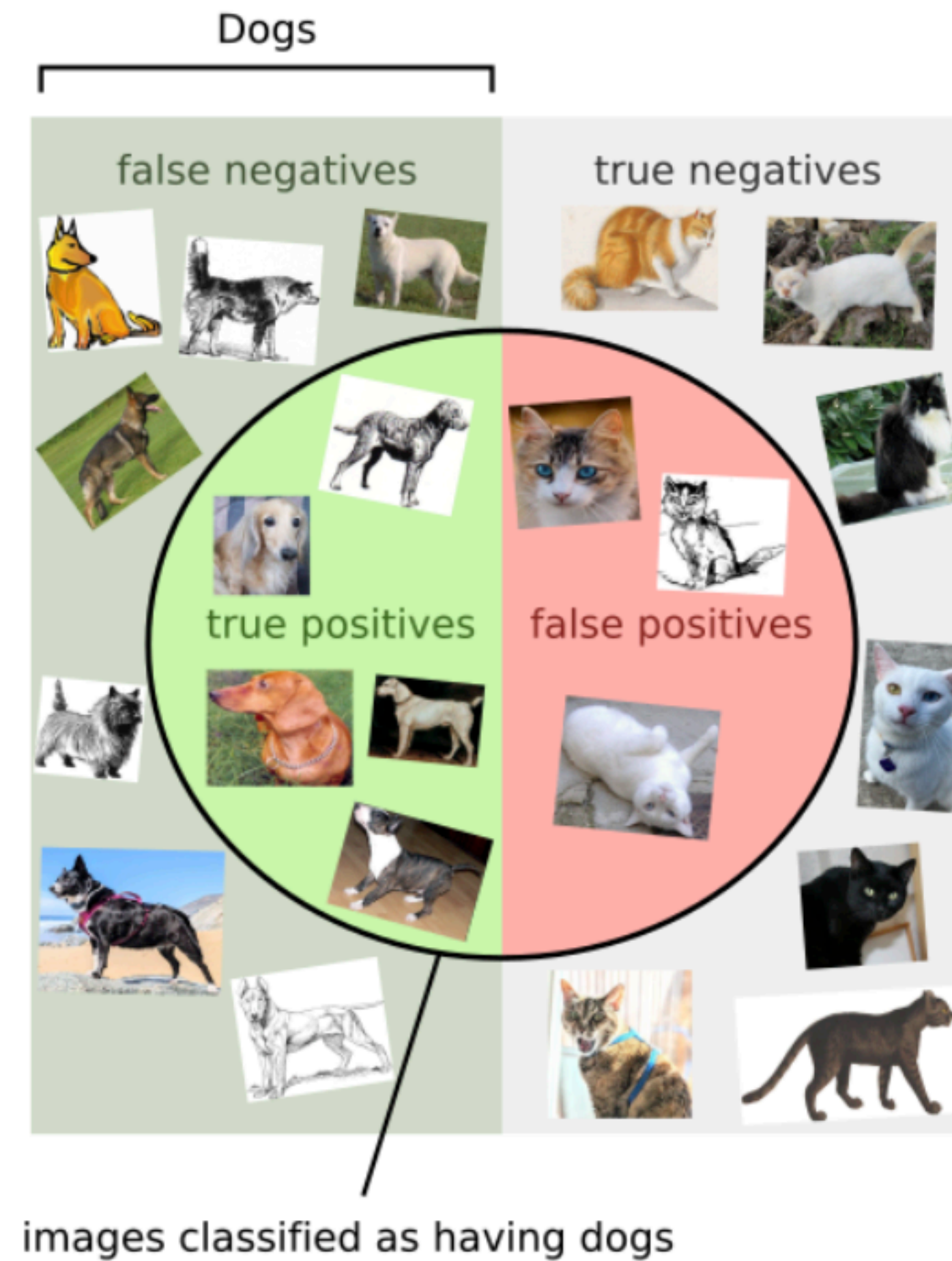
► Alternative Measures

► Precision(p): $\frac{\# \text{ true positive}}{\# \text{ true positive} + \# \text{ false positive}}$

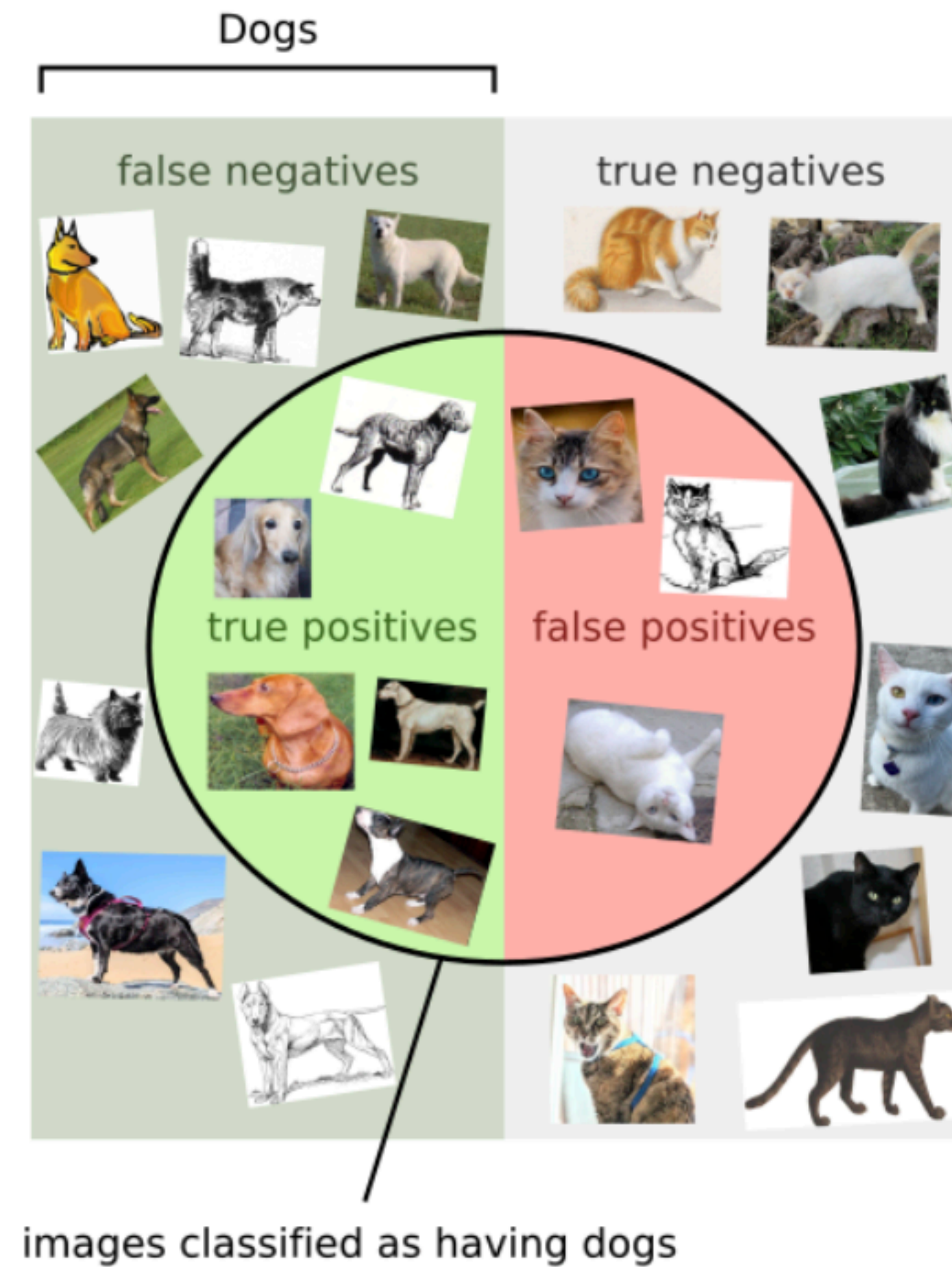
► Recall(r): $\frac{\# \text{ true positive}}{\# \text{ true positive} + \# \text{ false negative}}$

► F-measure: $\frac{2pr}{p + r}$

Evaluation



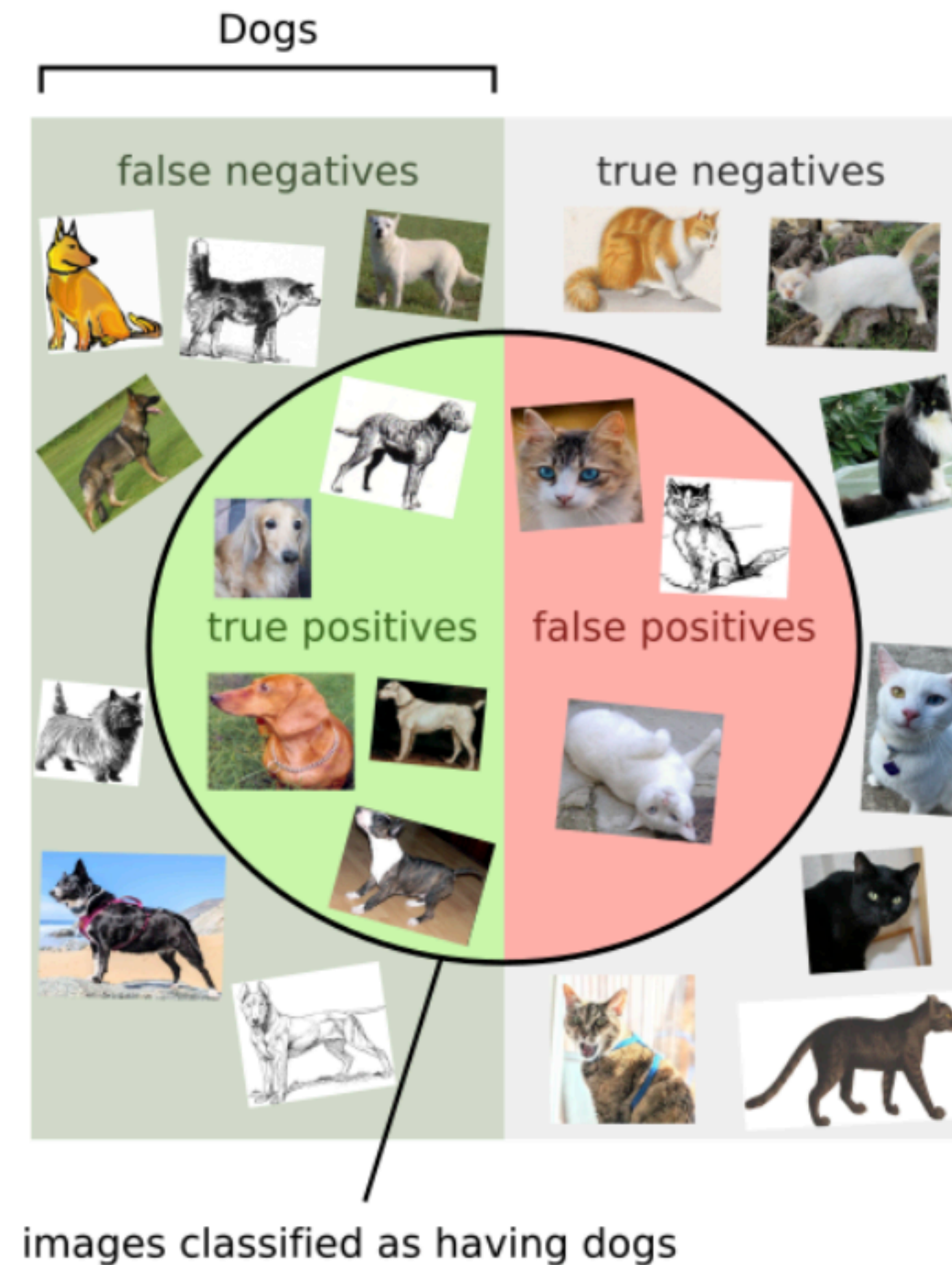
Evaluation



$$\text{Precision} = \frac{5 \text{ true pos.}}{8 \text{ total pos.}}$$

$$\text{Recall} = \frac{5 \text{ true pos.}}{12 \text{ total dogs}}$$

Evaluation



$$\text{Precision} = \frac{5 \text{ true pos.}}{8 \text{ total pos.}}$$

$$\text{Recall} = \frac{5 \text{ true pos.}}{12 \text{ total dogs}}$$

$$\text{Accuracy} = \frac{5 \text{ true pos.} + 7 \text{ true neg.}}{22 \text{ total images}}$$

Thinking about classification task

- ▶ cancer screening test
 - ▶ company hiring
 - ▶ movie recommendation
-
- ▶ where would accuracy not informative?
 - ▶ where would precision matter more than recall?
 - ▶ where would recall matter more than precision?

Overview

- ▶ Classification Problem
- ▶ Learning a classifier
 - ▶ **Logistic Regression**
- ▶ Perceptron
- ▶ Feedforward Neural Network

Logistic Regression Model

- ▶ **Building Blocks:**

- ▶ Represent input in a feature representation
- ▶ Classification function to compute y^* using $P(y | x)$
- ▶ Loss function (for learning)
- ▶ Optimization algorithm

- ▶ **Training:**

Learn the parameters of the model (w) to minimize loss function

- ▶ **Test:**

Apply parameters to predict class given a new input x

Features

- ▶ Mapping an input to a fixed dimensional vector

$$\phi(x) \in R^d$$

Sparse Feature Vector

- ▶ Traditional approaches build one-hot feature vector for each word.
- ▶ Simplest version: “Unigram” (one hot) vector

truck = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

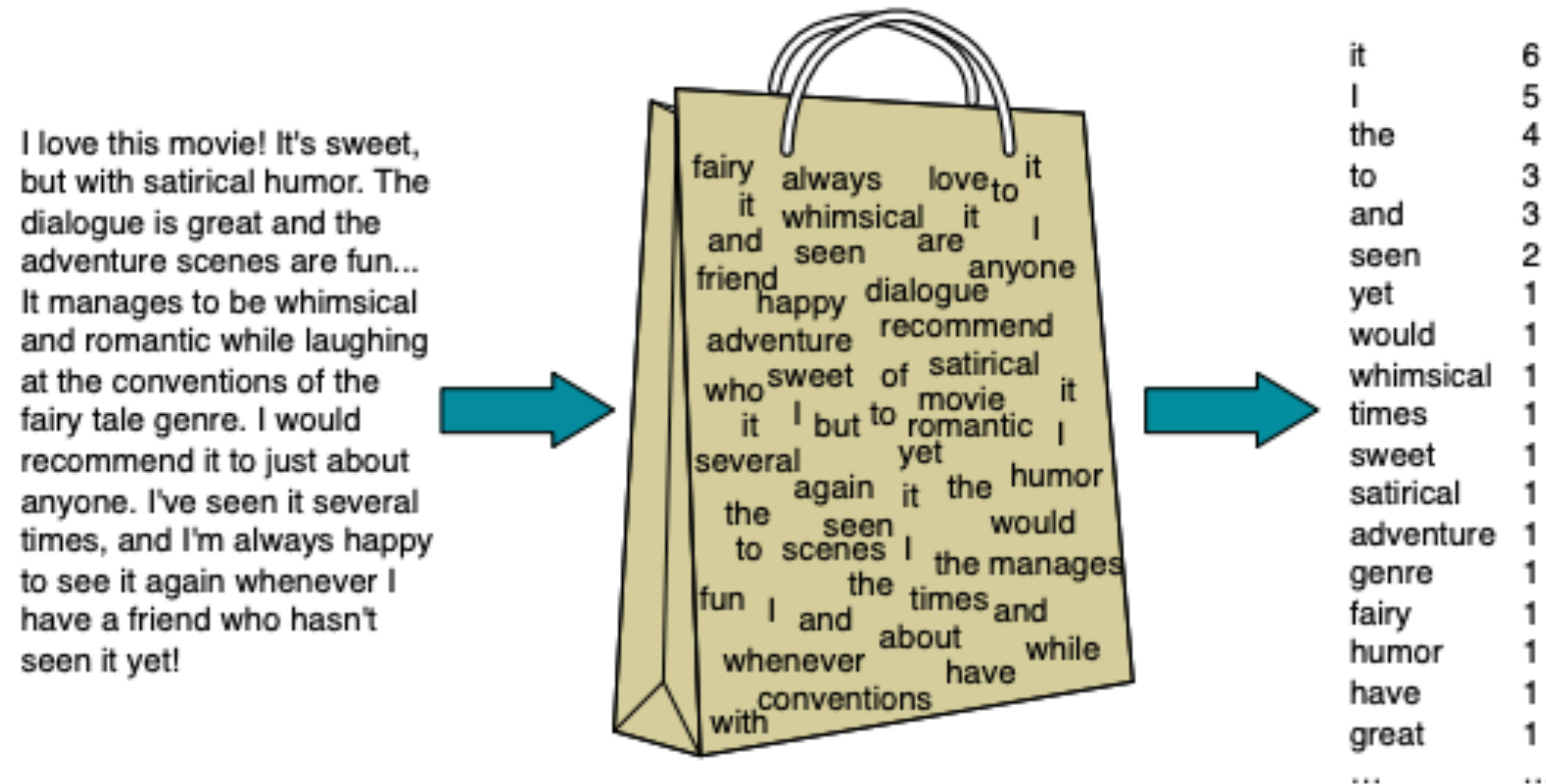
taco = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

- ▶ The dimension of the vector?
- ▶ No notion of similarity / relationship between words..

Car = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]

Bus = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Bag of Words



truck = [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

taco = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

“taco truck” = [0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Application: Sentiment Analysis

this movie was great! would watch again

+

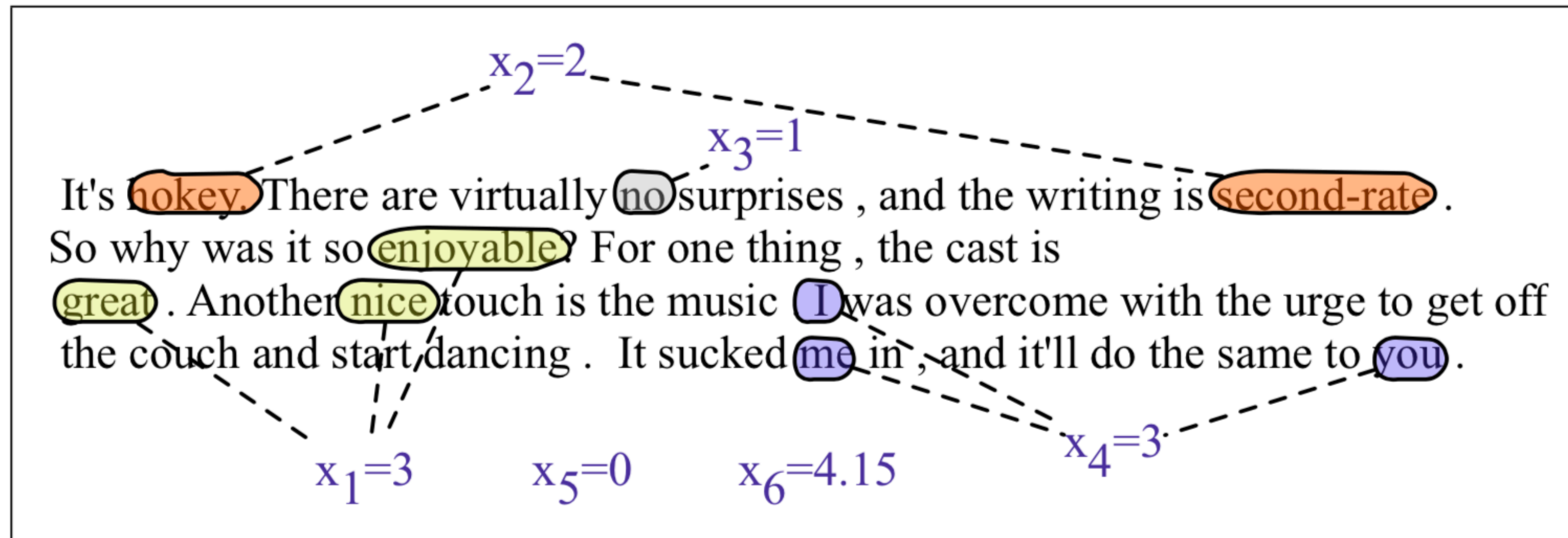
the movie was gross and overwrought, but I liked it

+

this movie was not really very enjoyable

—

Application: Sentiment Analysis



Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no" } \in \text{ doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!" } \in \text{ doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

Classification Function

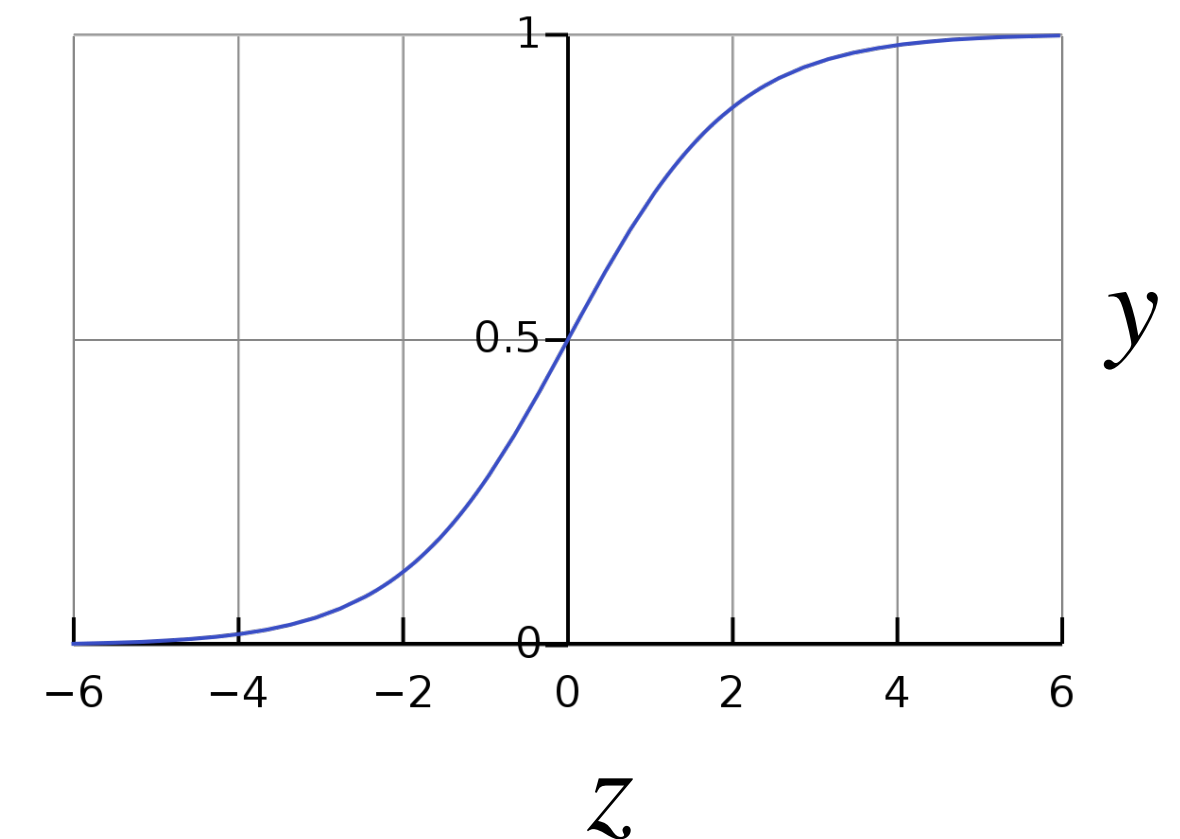
- ▶ Given an input feature vector: $\phi(x) \in R^d$
- ▶ A weight vector: $w \in R^d$
- ▶ Output: $p(y = 1 | x, w)$ Given input x and weight vector w, what is p(y=1)?
- ▶ Learning a classification function: $f : R^d \rightarrow [0,1]$

- ▶ Sigmoid:

$$z = w \cdot \phi(x)$$

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

$$p(y = 1 | x, w) = \sigma(w \cdot \phi(x))$$



Binary Classification

► Probabilities:
$$P(y = 1 | x, w) = \frac{1}{1 + e^{-(w \cdot \phi(x))}}$$

► Decision Boundary:

$$y^* = \operatorname{argmax}_{y \in \{0,1\}} p(y | x, w)$$

Classification Function

- ▶ **binary** classifier (sigmoid):

Feature vector: $\phi(x)$

$$P(y = 1 \mid x, w) = \frac{1}{1 + e^{-(w \cdot \phi(x))}}$$

$$y^* = \operatorname{argmax}_{y \in \{0,1\}} p(y \mid x, w)$$

- ▶ **multi class** classifier (softmax):

$\phi(x, y)$ ▶ Softmax =
“exponentiate and
normalize”

$$p(y \mid x, w) = \frac{e^{(w \cdot \phi(x, y))}}{\sum_{y'} e^{(w \cdot \phi(x, y'))}}$$

$$y^* = \operatorname{argmax}_{y \in C} w \cdot \phi(x, y)$$

Logistic Regression Model

- ▶ **Building Blocks:**

- ▶ Represent input in a feature representation
- ▶ Classification function to compute y^* using $P(y | x)$
- ▶ Loss function (for learning)
- ▶ Optimization algorithm

- ▶ **Training:**

Learn the parameters of the model (w) to minimize loss function

- ▶ **Test:**

Apply parameters to predict class given a new input x

Logistic Regression: Classification function

- ▶ You compute the score for each label, and predict the label with the highest score

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

$$y^* = \operatorname{argmax}_{y \in \{0,1\}} p(y | x, w)$$

How do we learn the weight vector w ?

What is the best weights?

- ▶ Goal: Choose the weights that will give highest ***test*** set accuracy / F1
 - ▶ But we don't have an access to the test set
- ▶ Maybe choose the weights that gives highest ***training*** set accuracy / F1?
 - ▶ Optimization is not easy
 - ▶ May not generalize to test set

Learning Objective

- ▶ Model: we use the scores to compute probabilities

$$p(y = 1 | x, w) = \frac{1}{(1 + e^{(-w \cdot \phi(x))})}$$

- ▶ Learning: maximize the **log conditional likelihood** of training data

$$\log \prod_{i=1}^N p(y^i | x^i, w) = \sum_{i=1}^N \log(p(y^i | x^i, w))$$

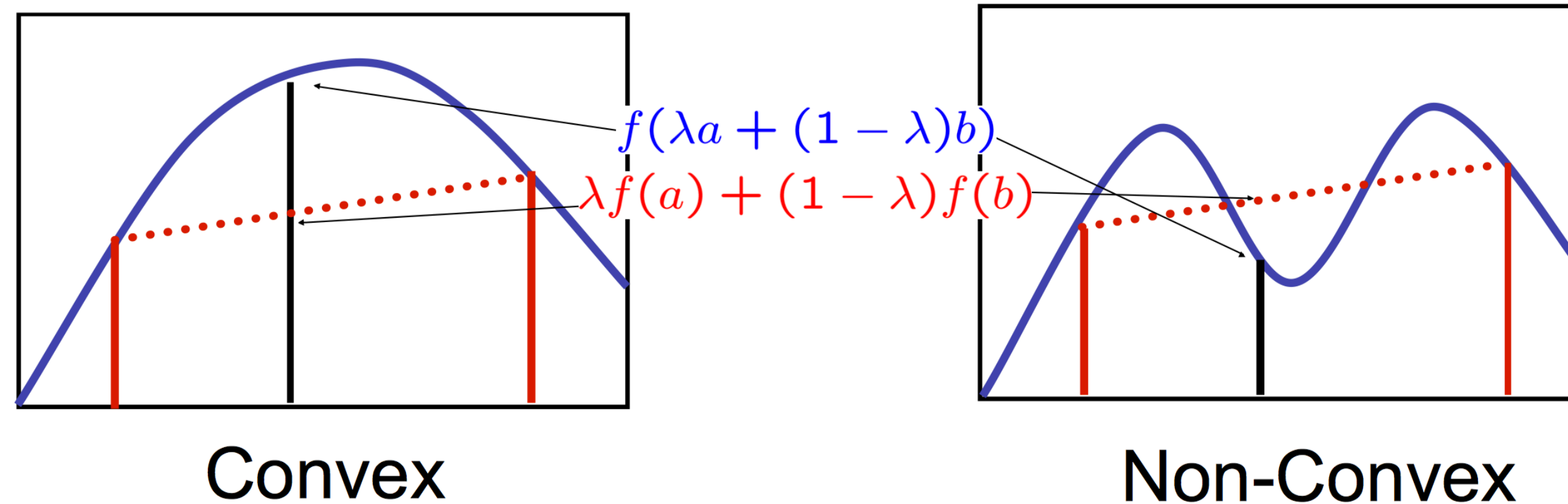
- ▶ We define loss as *minimizing negative* log likelihood of data

$$L(w) = - \sum_{i=1}^N \log(p(y^i | x^i, w))$$

Negative Log Likelihood Loss

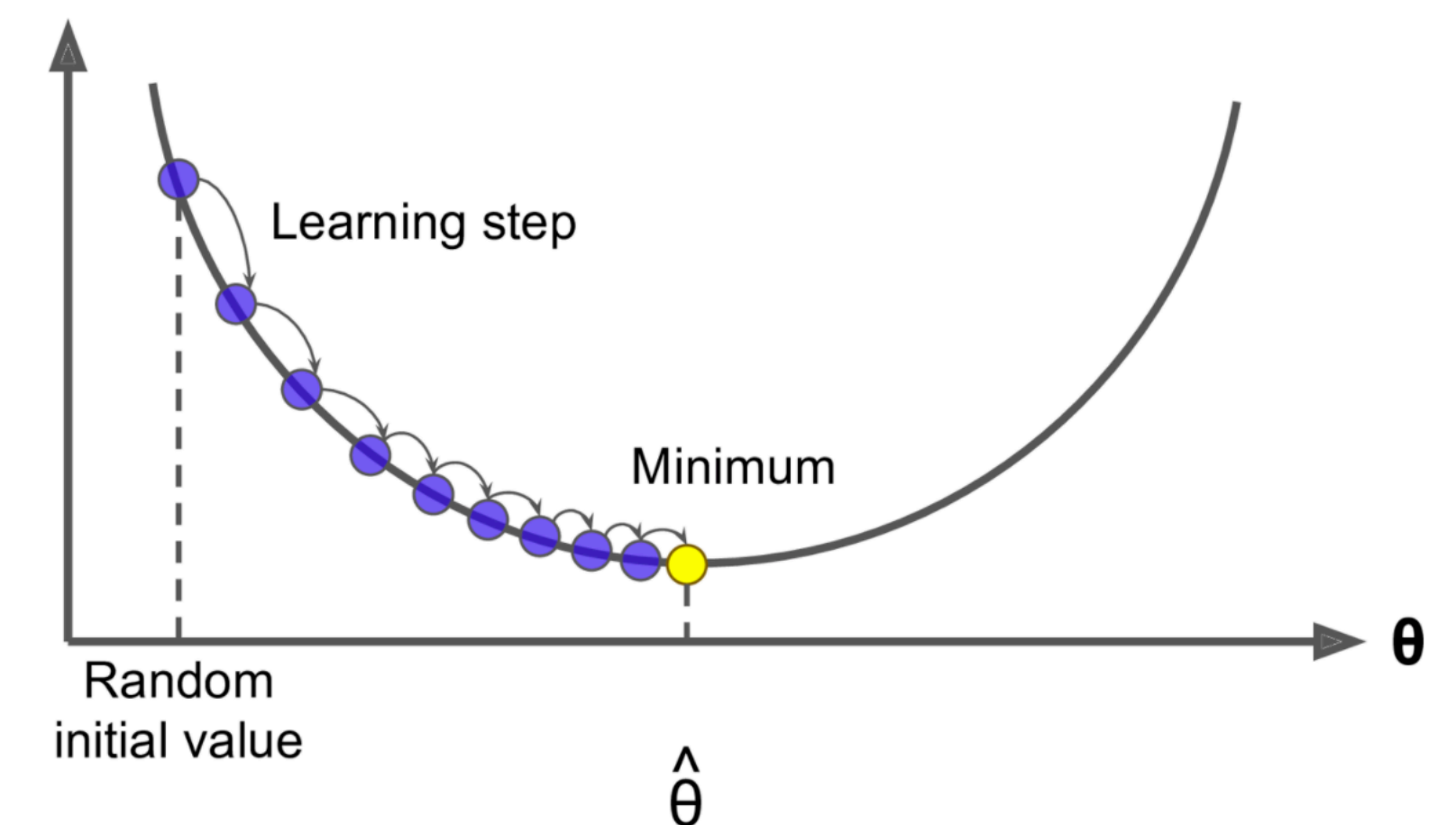
- ▶ Unfortunately, this does not have a closed form solution
- ▶ Differentiable
- ▶ Convex (local optima = global optima)

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

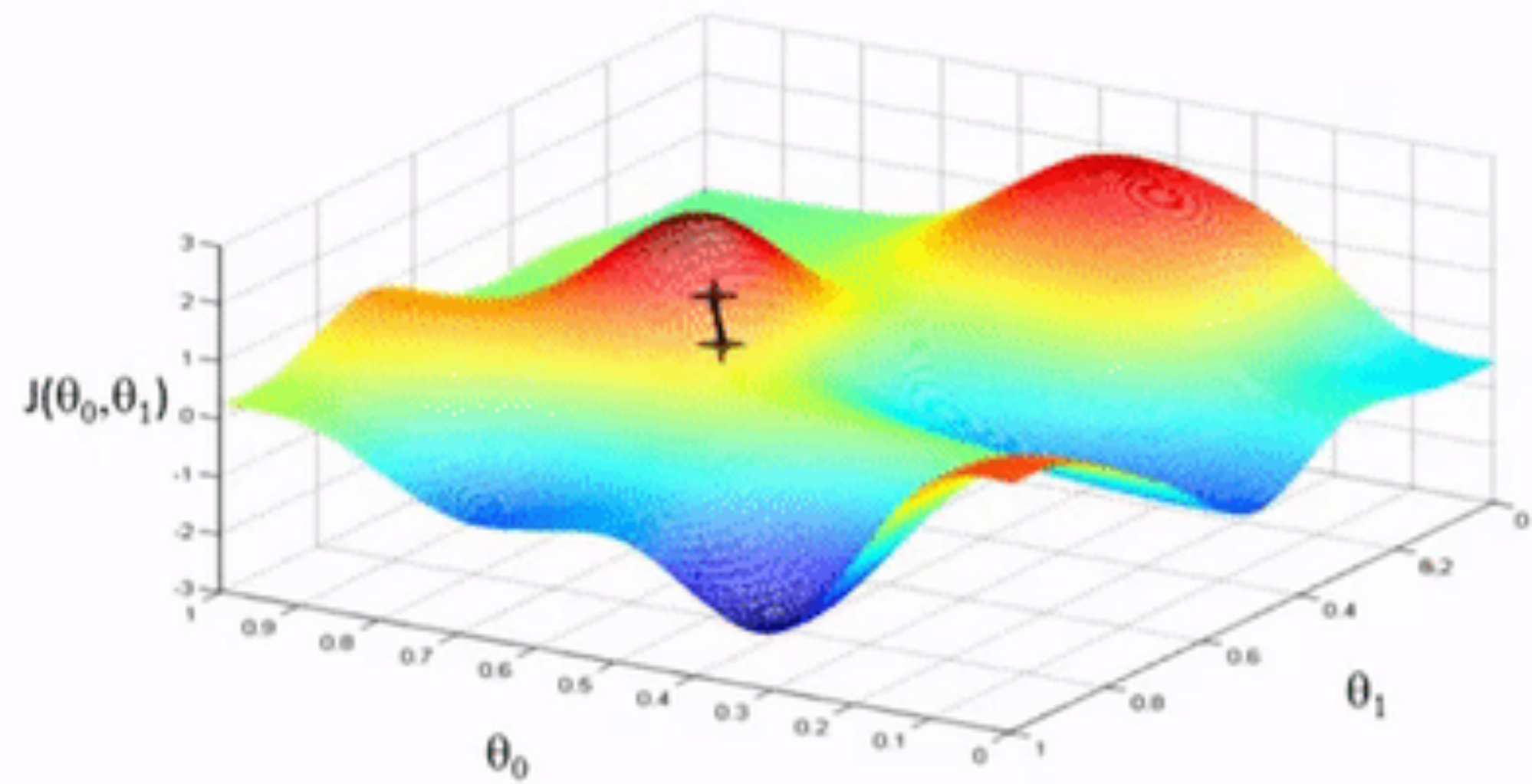


Optimization

- ▶ Basic Idea:
 - ▶ Compute the derivative, move following the gradient incrementally
 - ▶ At local optimum, derivative will be zero
- ▶ Stochastic Gradient Descent
 - ▶ Compute the loss and minimize after each training example

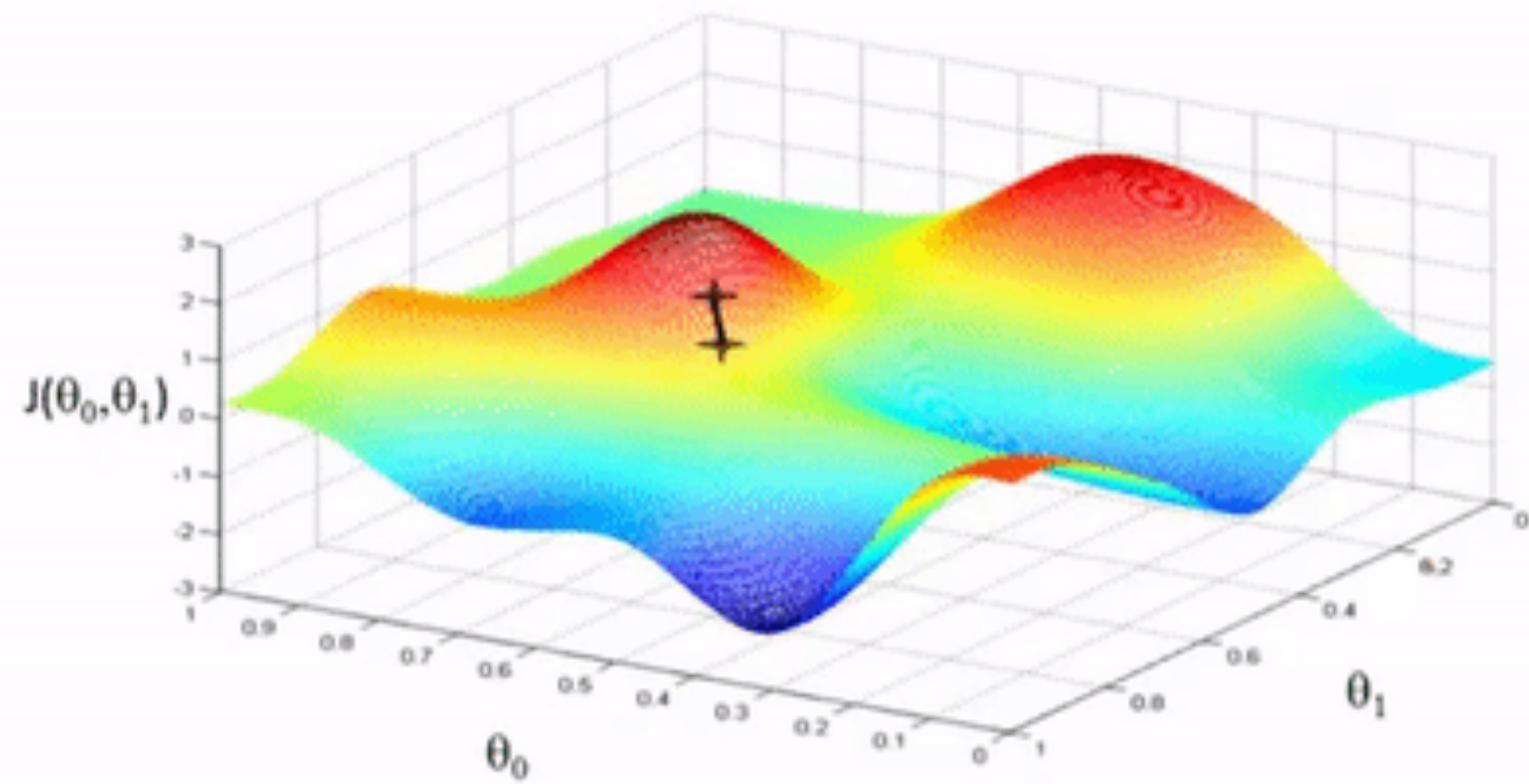


Optimization

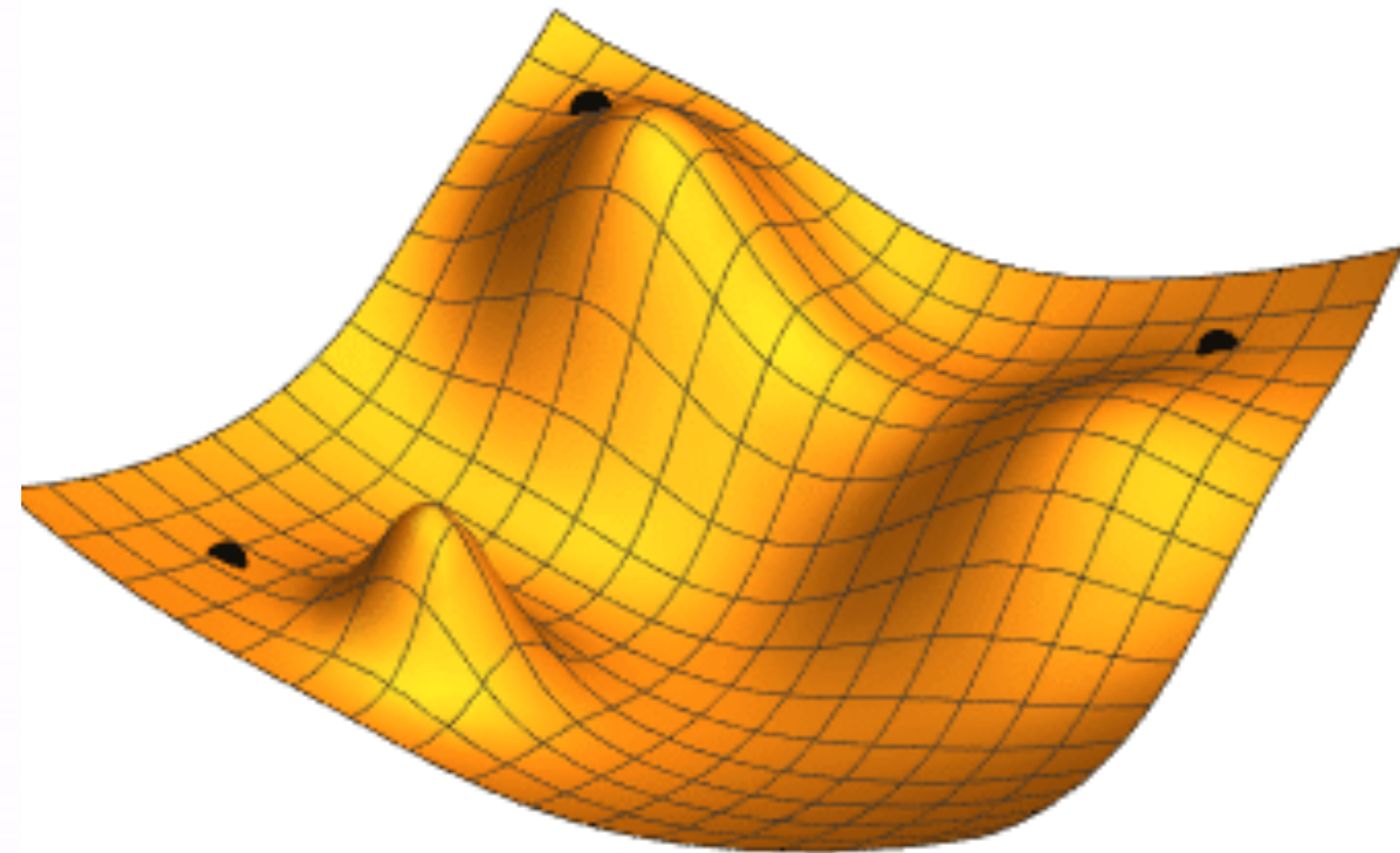


Andrew Ng

Optimization

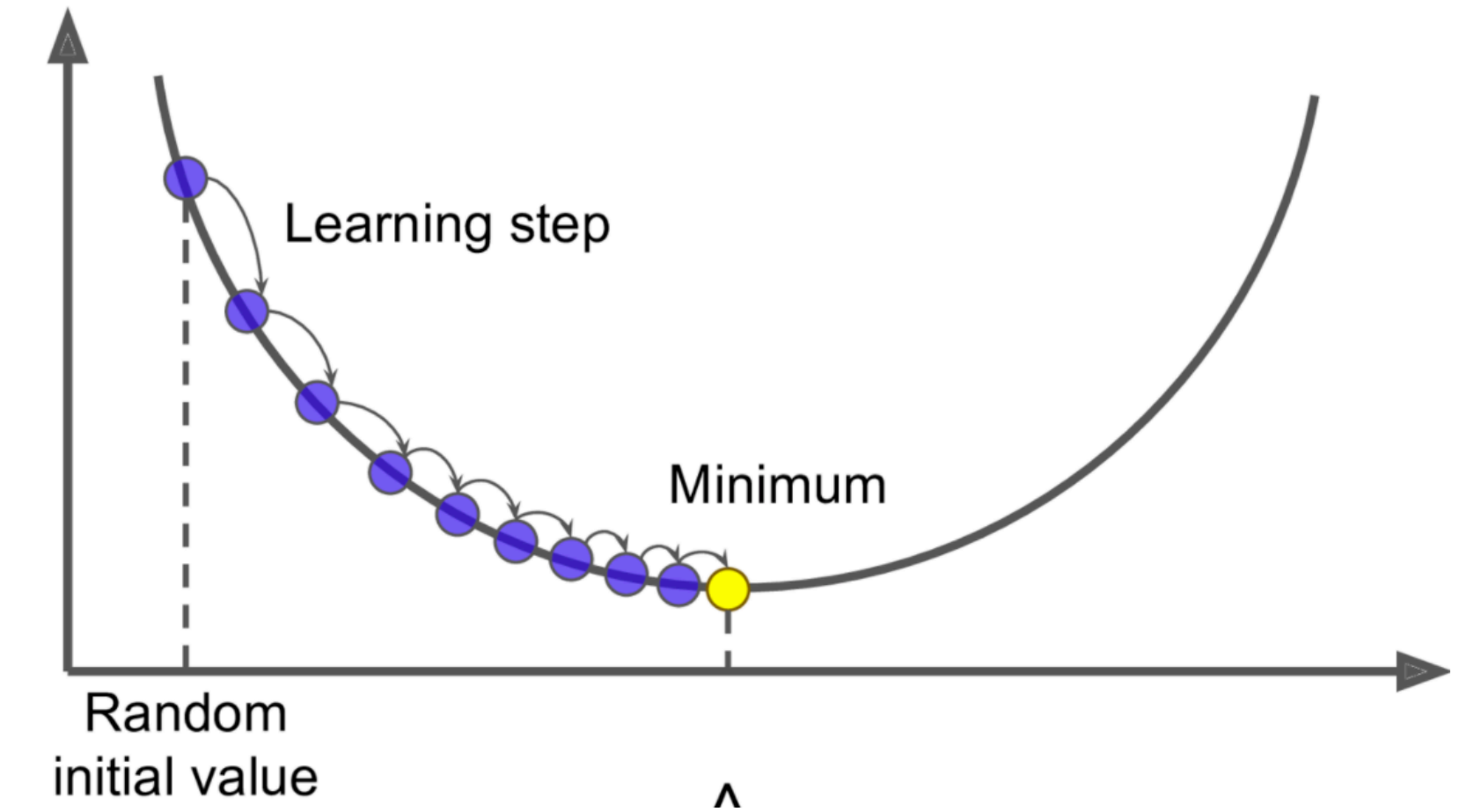


Andrew Ng

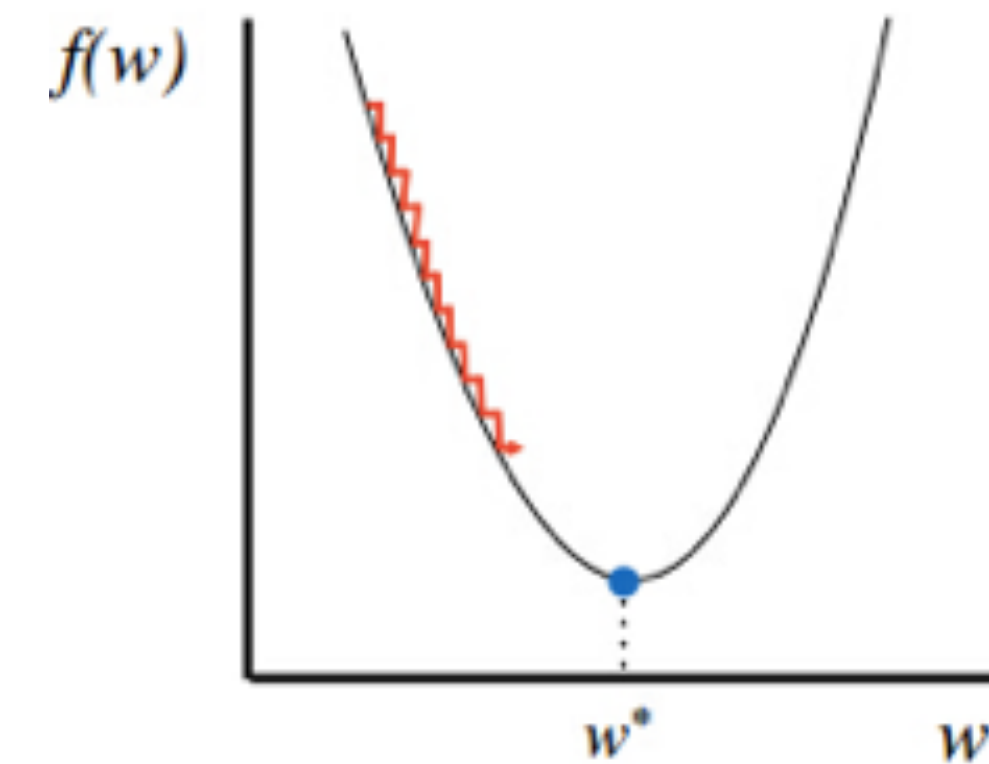


Learning Rate

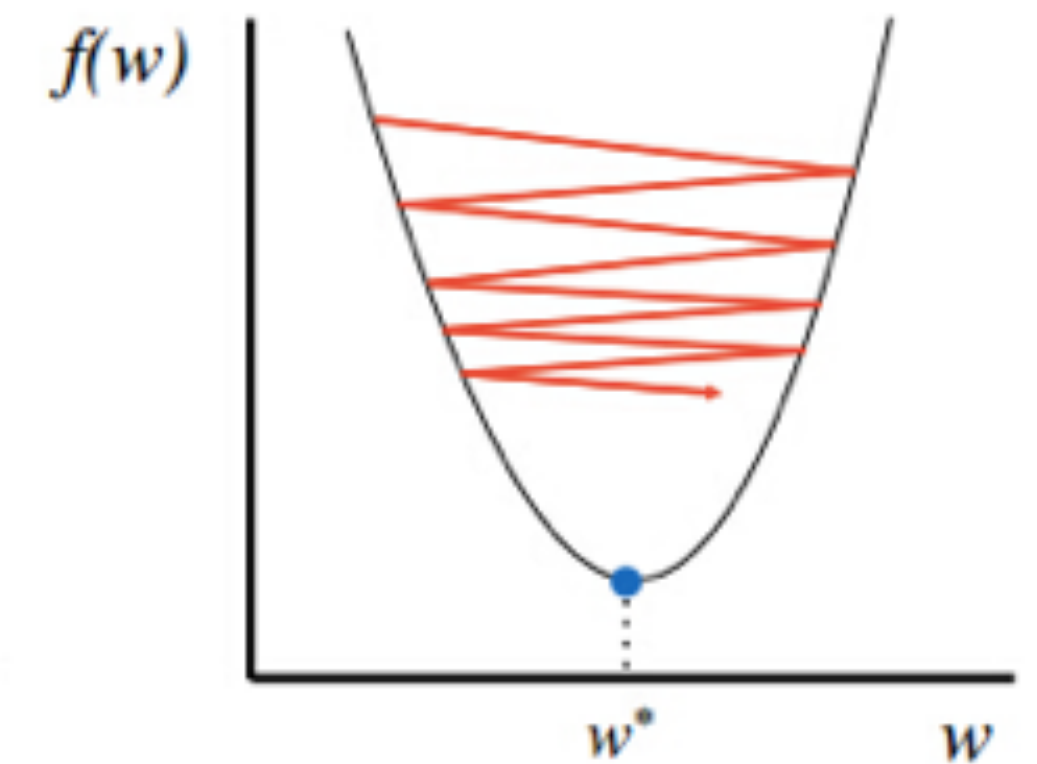
Updates: $w \leftarrow w - \eta \frac{d}{dw} L(w)$



Higher/faster learning rate (η):
larger updates to parameters



Too small: converge
very slowly



Too big: overshoot and
even diverge

Logistic Regression Model

- ▶ **Building Blocks:**

- ▶ Represent input in a feature representation
- ▶ Classification function to compute y^* using $P(y | x)$
- ▶ Loss function (for learning)
- ▶ Optimization algorithm

- ▶ **Training:**

Learn the parameters of the model (w) to minimize loss function

- ▶ **Test:**

Apply parameters to predict class given a new input x

Let's compute gradient! (Binary)

$$L(w) = -\log \prod_{i=1}^N p(y^i | x^i, w)$$

$$\hat{y} = \sigma(w \cdot \phi(x^{(i)}))$$

$$= -\sum_{i=1}^n [y^{(i)} \log \hat{y} + (1 - y^{(i)}) \log(1 - \hat{y})]$$

Calculus!

$$= -\sum_{i=1}^n [y^{(i)} \log \sigma(w \cdot \phi(x^{(i)})) + (1 - y^{(i)}) \log(1 - \sigma(w \cdot \phi(x^{(i)})))]$$

$$\frac{dL(w)}{dw_j} = -\sum_{i=1}^n [y^{(i)} \phi(x^i)_j - \sigma(w \cdot \phi(x^{(i)})) \phi(x^i)_j]$$

Total count of feature j in positive candidate
Expected count of feature j in predicted positive candidate

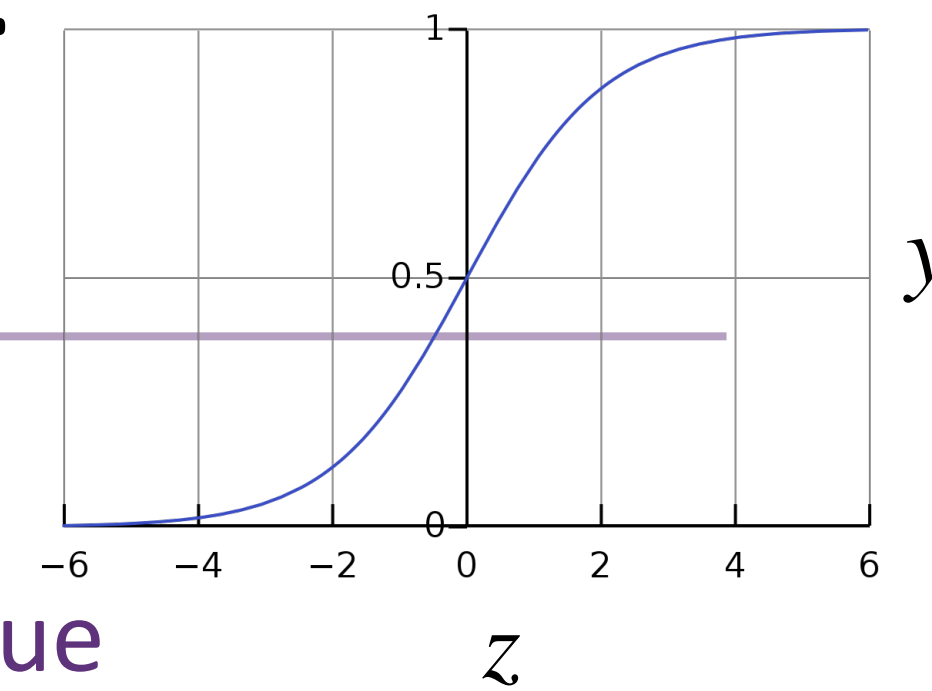
$$\frac{dL(w)}{dw_j} = -\sum_{i=1}^n [y^{(i)} - \hat{y}] \phi(x^i)_j$$

correct candidate
expected candidate
Input feature value

Exercise

► Sigmoid:

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



$$\frac{dL(w)}{dw_j} = - \sum_{i=1}^n [y^{(i)} - \hat{y}] \phi(x^i)_j$$

Annotations for the equation:

- $y^{(i)}$: correct candidate (blue box and arrow)
- $\hat{y}$: expected candidate (red box and arrow)
- $\phi(x^i)_j$: Input feature value (purple box and arrow)

- Suppose we have a single training example $[1, 1, 0]$ with true label = 1.
- Initialize our weights to all zeroes: $w = [0, 0, 0]$
- (1) What are the class probabilities $P(y|x)$ for each class given by logistic regression with these weights?
- (2) What would the LR update be with these class probabilities? That is, taking one step of stochastic gradient descent on this example with step size = 1, what weight vector do we get?
- (3) Suppose we train on this single example repeatedly. Qualitatively (i.e., you don't need to compute it, just describe what happens), how do the logistic regression weights change after encountering this example on subsequent iterations?

Implementation

- ▶ Supposing k active features on an instance, gradient is only nonzero on k dimensions
- ▶ $k < 100$, total num features = 1M+ on many problems
- ▶ Be smart about applying updates!
- ▶ In PyTorch: applying sparse gradients only works for certain optimizers and sparse updates are very slow (assumes dense feature vectors)

Classification Function

- ▶ **binary** classifier (sigmoid):

Feature vector: $\phi(x)$

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

$$y^* = \operatorname{argmax}_{y \in \{0,1\}} p(y | x, w)$$

$$\frac{dL(w)}{dw_j} = - \sum_{i=1}^n [y^{(i)} \phi(x^i)_j - \sigma(w \cdot \phi(x^{(i)})) \phi(x^i)_j]$$

- ▶ **multi class** classifier (softmax):

$\phi(x, y)$

$$p(y | x, w) = \frac{e^{(w \cdot \phi(x, y))}}{\sum_{y'} e^{(w \cdot \phi(x, y'))}}$$

$$y^* = \operatorname{argmax}_{y \in C} w \cdot \phi(x, y)$$

$$\frac{dL(w)}{dw_j} = - \sum_{i=1}^N (\phi(x^i, y^i)_j - \sum_{y'} P(y' | x^i, w) \phi(x^i, y')_j)$$

Expected count of feature j in predicted candidate

Total count of feature j in correct candidate

Overfitting?

- ▶ Regularization:
 - ▶ Imposing an additional penalty on the **form** of the solution
 - ▶ Add a L2 regularization term to the likelihood, to suppress very large weights. We want an **economic** solution.

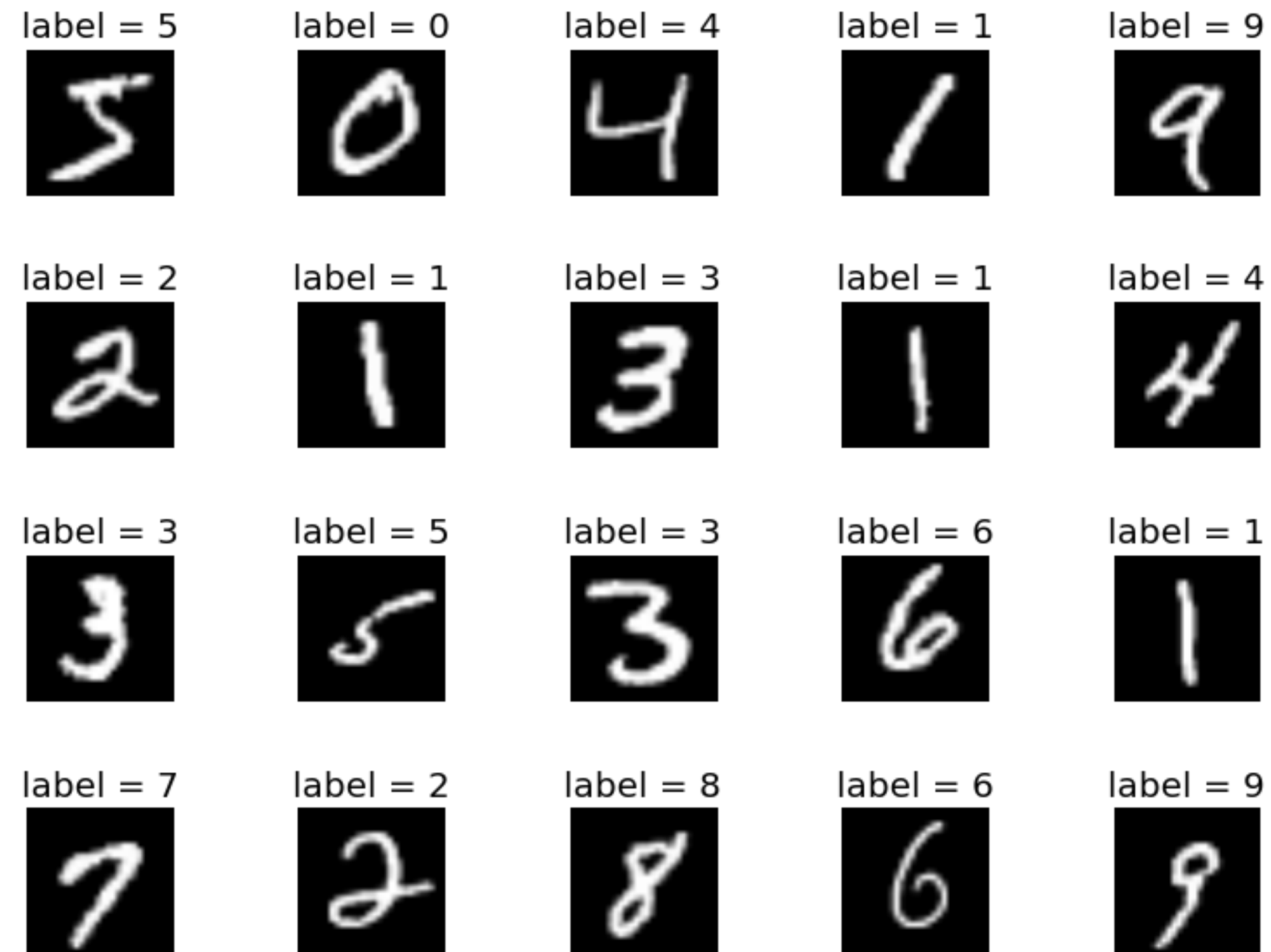
$$L(w) = -\log \prod_{i=1}^N p(y^i | x^i, w) + \frac{\lambda}{2} ||w||^2$$

$$\frac{dL(w)}{dw_j} = - \sum_{i=1}^n [y^{(i)} - \sigma(w \cdot \phi(x^{(i)}))] \phi(x^i)_j + \lambda w_j$$

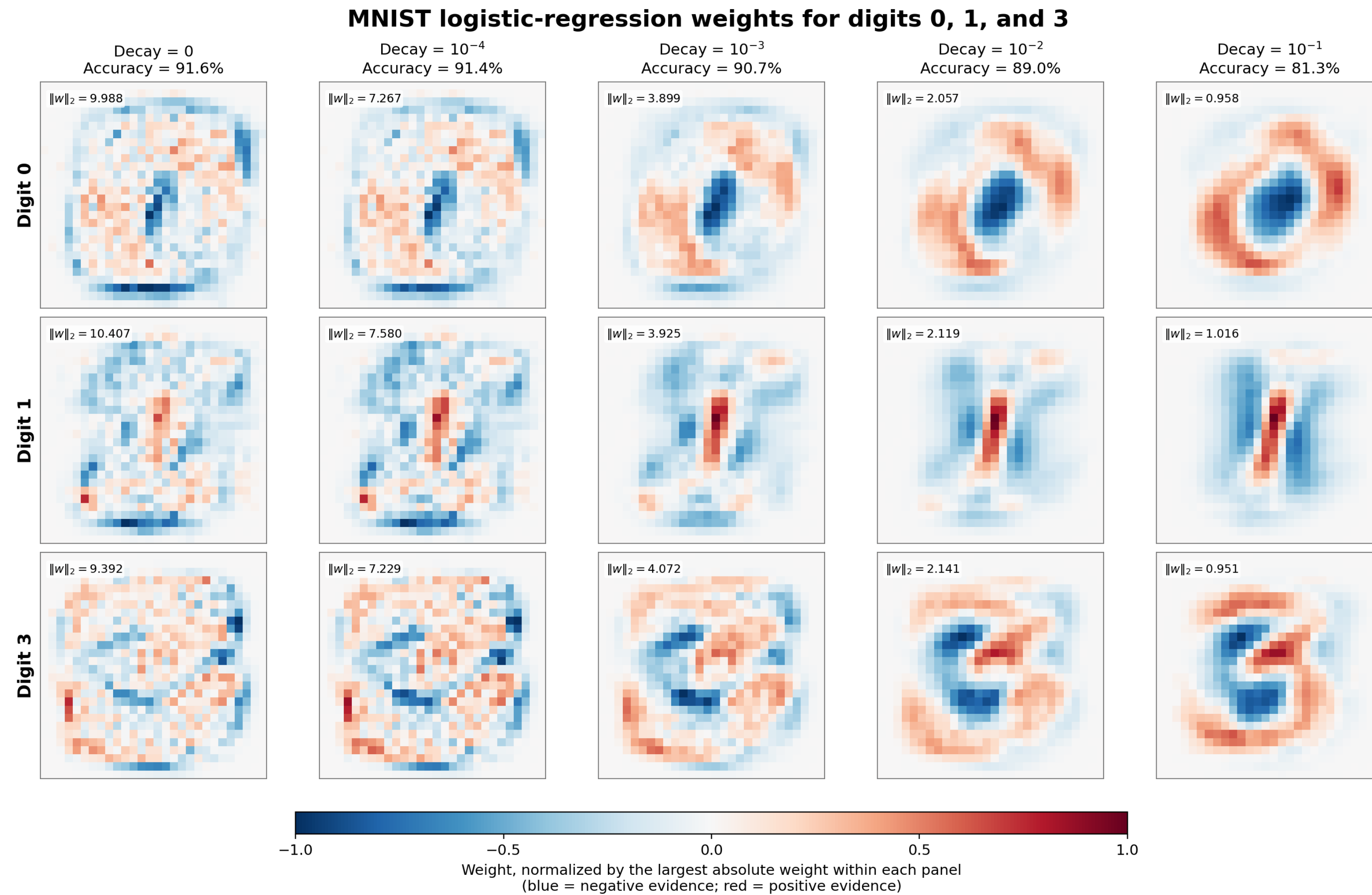


Big weights are bad

Overfitting?



Overfitting?



Types of Supervised Classifiers

- ▶ Probabilistic: can develop rich features $y^* = \operatorname{argmax}_y p(y | x)$
- ▶ Q: Do we have to estimate a probability distribution at all?
- ▶ Linear predictor: $y^* = \operatorname{sign}(w \cdot \phi(x))$
- ▶ Perceptron algorithm
 - Error driven, simple, additive updates

Perceptron

- ▶ Start with zero weight vector.
- ▶ Visit training examples one by one.
- ▶ Decision rule: $y^* = \text{sign}(w \cdot \phi(x^i))$
 - ▶ If correct ($y^* = y^i$): do nothing!
 - ▶ If incorrect: if label is positive, $w \leftarrow w + \phi(x^i)$
negative, $w \leftarrow w - \phi(x^i)$

Multi-class Perceptron

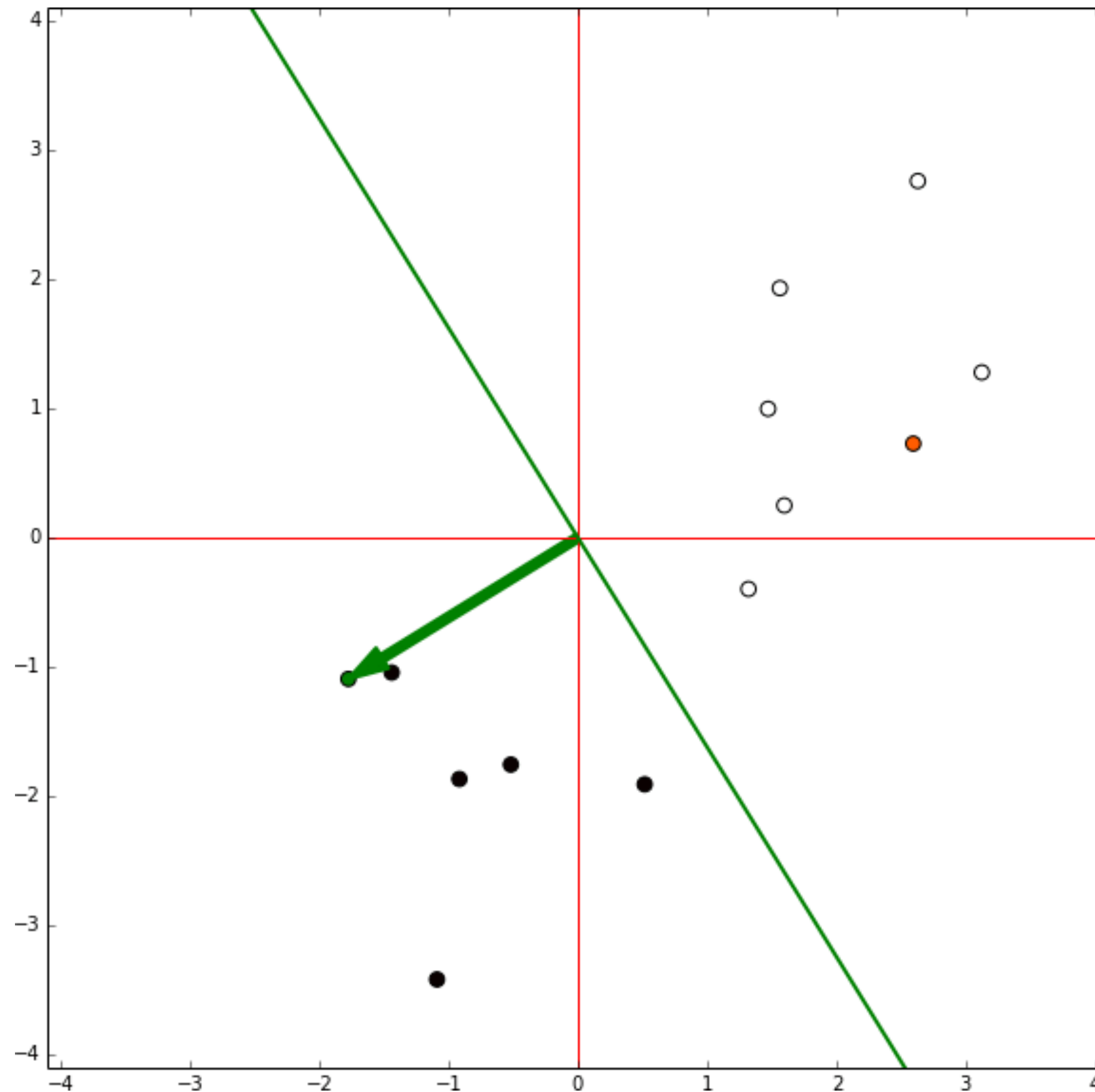
- ▶ Start with zero weights
- ▶ Visit training instances $(x^{(i)}, y^{(i)})$ one by one
- ▶ Make a prediction

$$y^* = \operatorname{argmax}_{y \in C} w \cdot \phi(x^i, y)$$

- ▶ If correct ($y^* == y^{(i)}$): no change, go to next example!
- ▶ If wrong: adjust weights

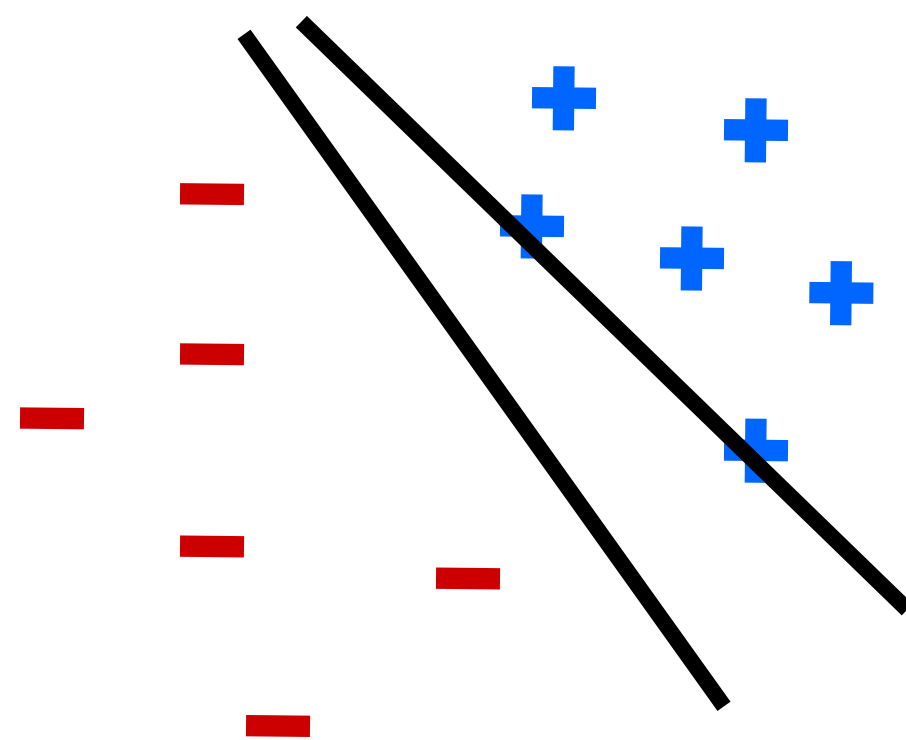
$$w = w + \phi(x^i, y^i) - \phi(x^i, y^*)$$

Multi-class Perceptron

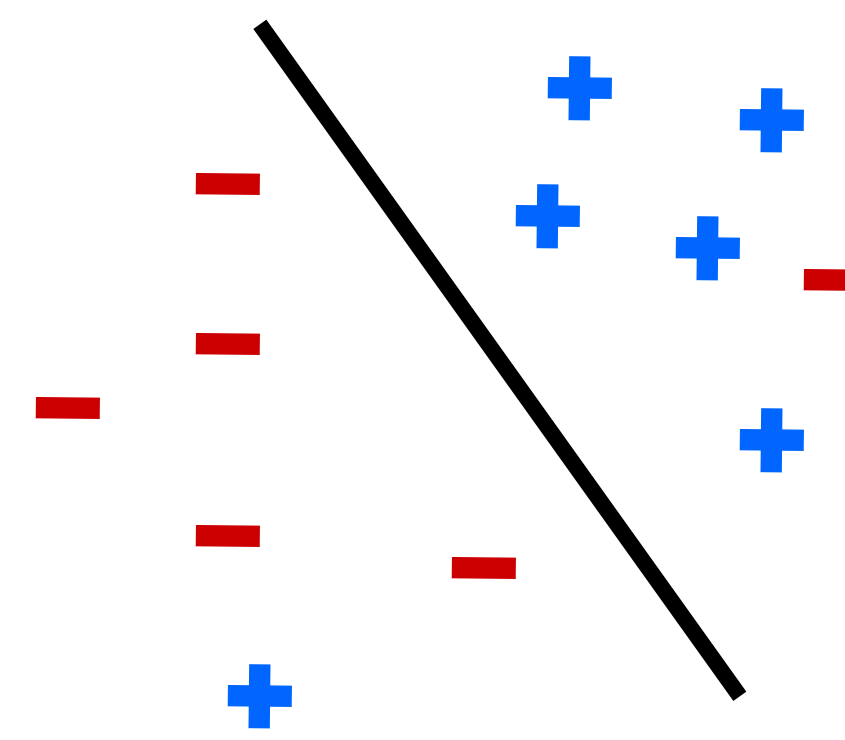


Perceptron Learning

- ▶ No counting or computing probabilities on training set
- ▶ **Separability**: some parameters get the training set perfectly correct
- ▶ **Convergence**: if the training is separable, perceptron will eventually converge
- ▶ **Mistake Bound**: the maximum number of mistakes (binary case) related to the margin or degree of separability



Separable



Non-Separable

Perceptron vs. Logistic Regression

Prediction

Perceptron

$$y^* = \text{sign}(w \cdot \phi(x))$$

Logistic Regression

$$P(y = 1 | x, w) = \hat{y} = \sigma(w \cdot \phi(x))$$

$$y^* = \operatorname{argmax}_{y \in \{0,1\}} p(y|x, w)$$

Learning

▶ If correct: do nothing!

▶ If incorrect:

▶ if label is positive, $w \leftarrow w + \phi(x)$

if negative, $w \leftarrow w - \phi(x)$

▶ Always: $w \leftarrow w - \eta \frac{d}{dw} L(w)$

$$\frac{dL(w)}{dw_j} = - \sum_{i=1}^n [y - \hat{y}] \phi(x)_j$$

if positive, $w \leftarrow w + \phi(x)(1 - \hat{y})$

if negative, $w \leftarrow w - \phi(x)\hat{y}$

Overview

- ▶ Classification Problem
- ▶ Learning a classifier
 - ▶ **Logistic Regression**
- ▶ Perceptron
- ▶ Feedforward Neural Network

What makes neural network different
from classifiers we learned so far?

Why Neural Network?

- ▶ Linear classification: $\operatorname{argmax}_{y \in \{0,1\}} w \cdot \phi(x)$
- ▶ Want to learn intermediate **conjunctive** features of the input

*the movie was **not** all that **good***

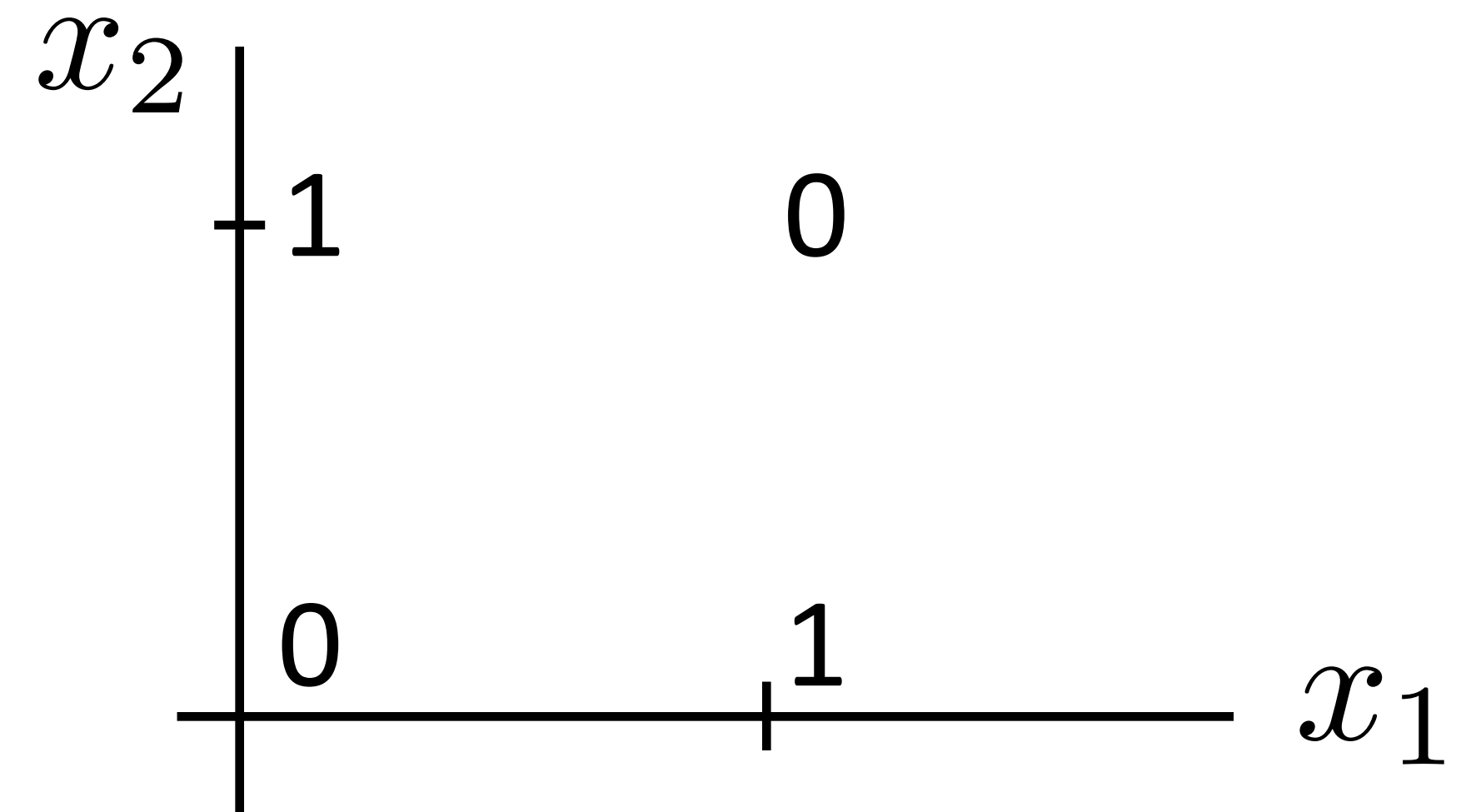
$I[\text{contains } not \ \& \ \text{contains } good]$

- ▶ How do we learn this if our feature vector is just the unigram indicators?

$I[\text{contains } not], I[\text{contains } good]$

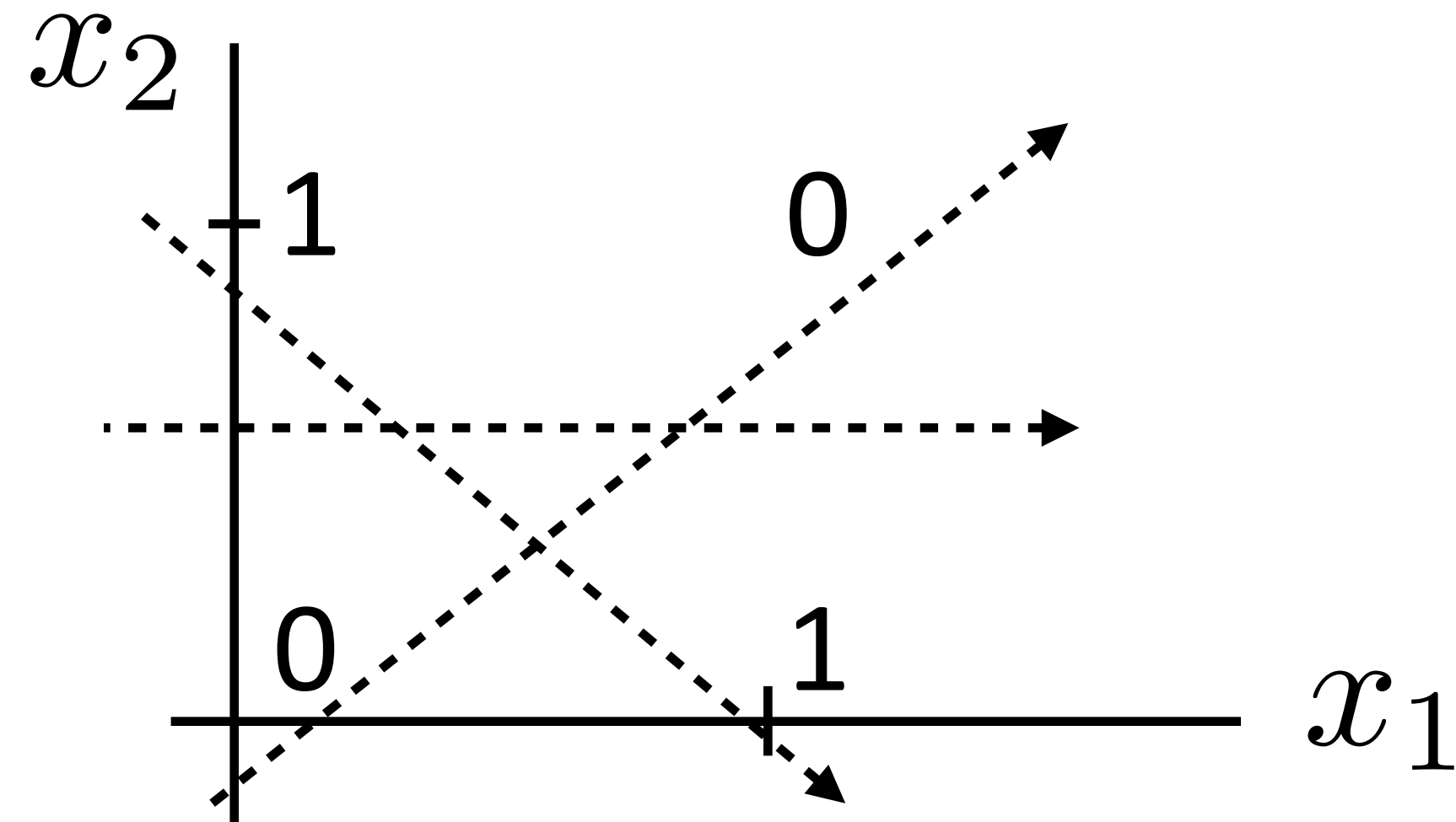
Neural Networks: XOR

- ▶ Let's see how we can use neural nets to learn a simple nonlinear function
- ▶ Inputs x_1, x_2
- ▶ Output y



x_1	x_2	$y = x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Neural Networks: XOR



$$y = a_1 x_1 + a_2 x_2$$

X

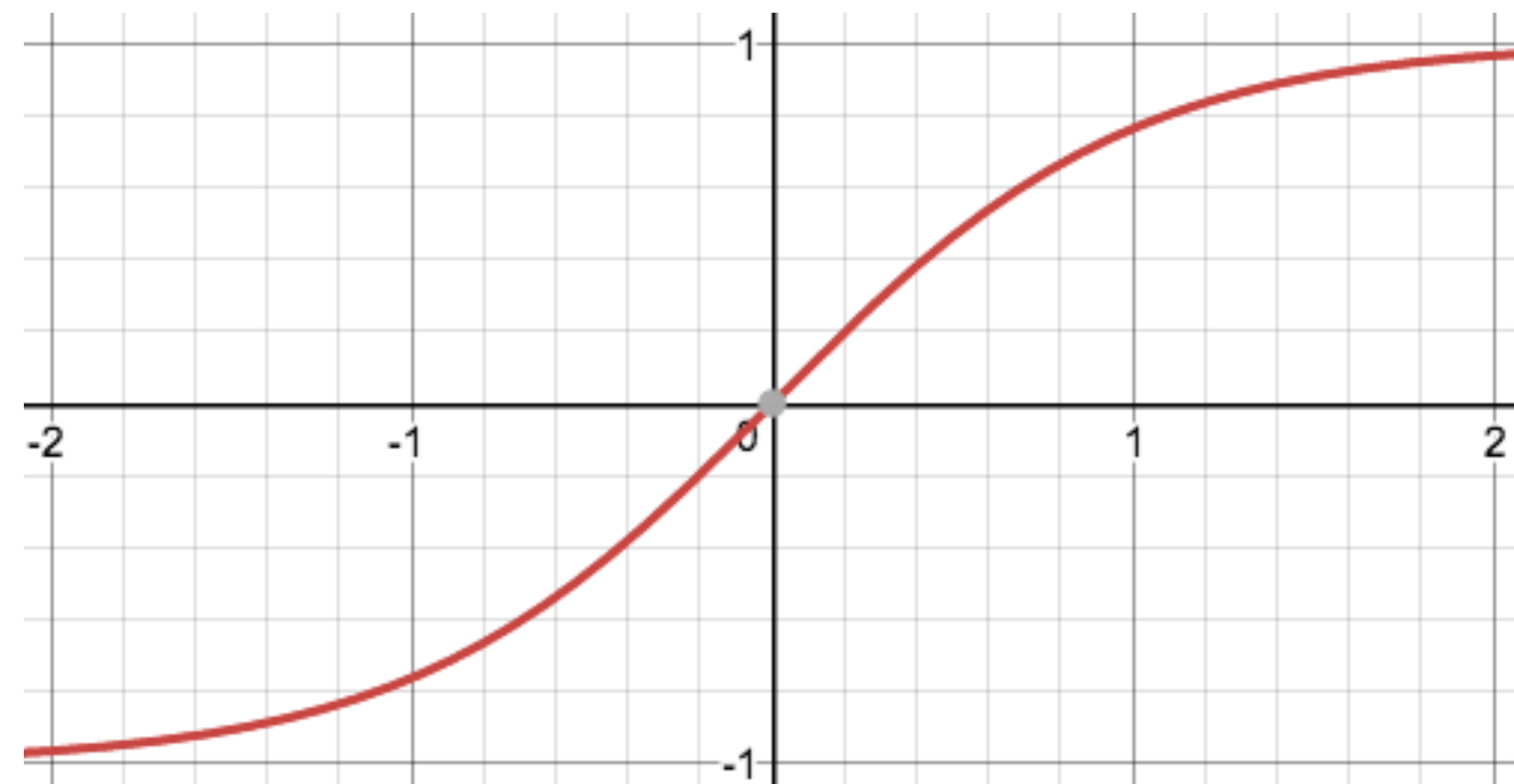
$$y = a_1 x_1 + a_2 x_2 + a_3 \tanh(x_1 + x_2)$$

“or”

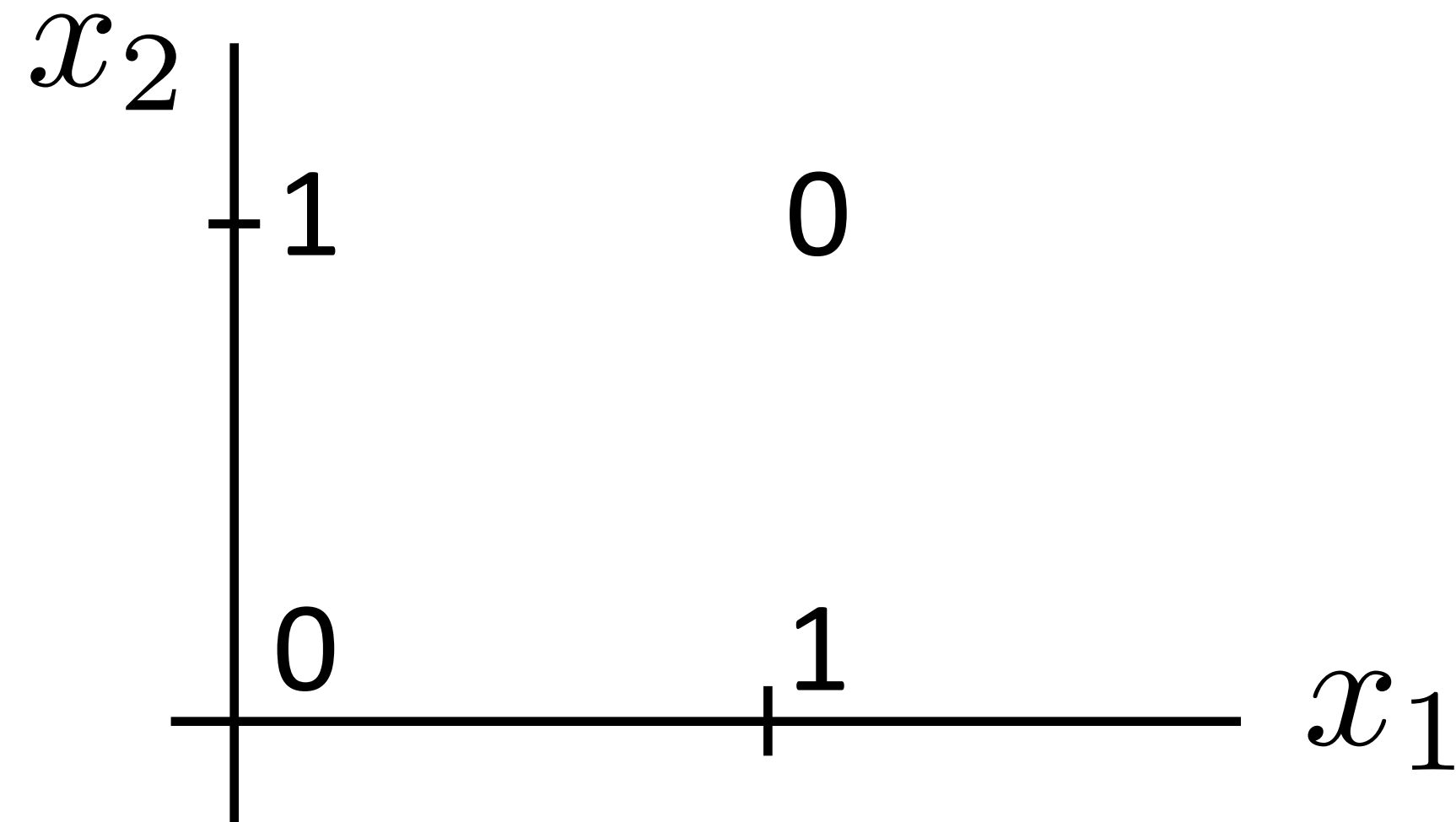
✓

(looks like action potential in neuron)

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0



Neural Networks: XOR



$$y = a_1 x_1 + a_2 x_2$$

✗

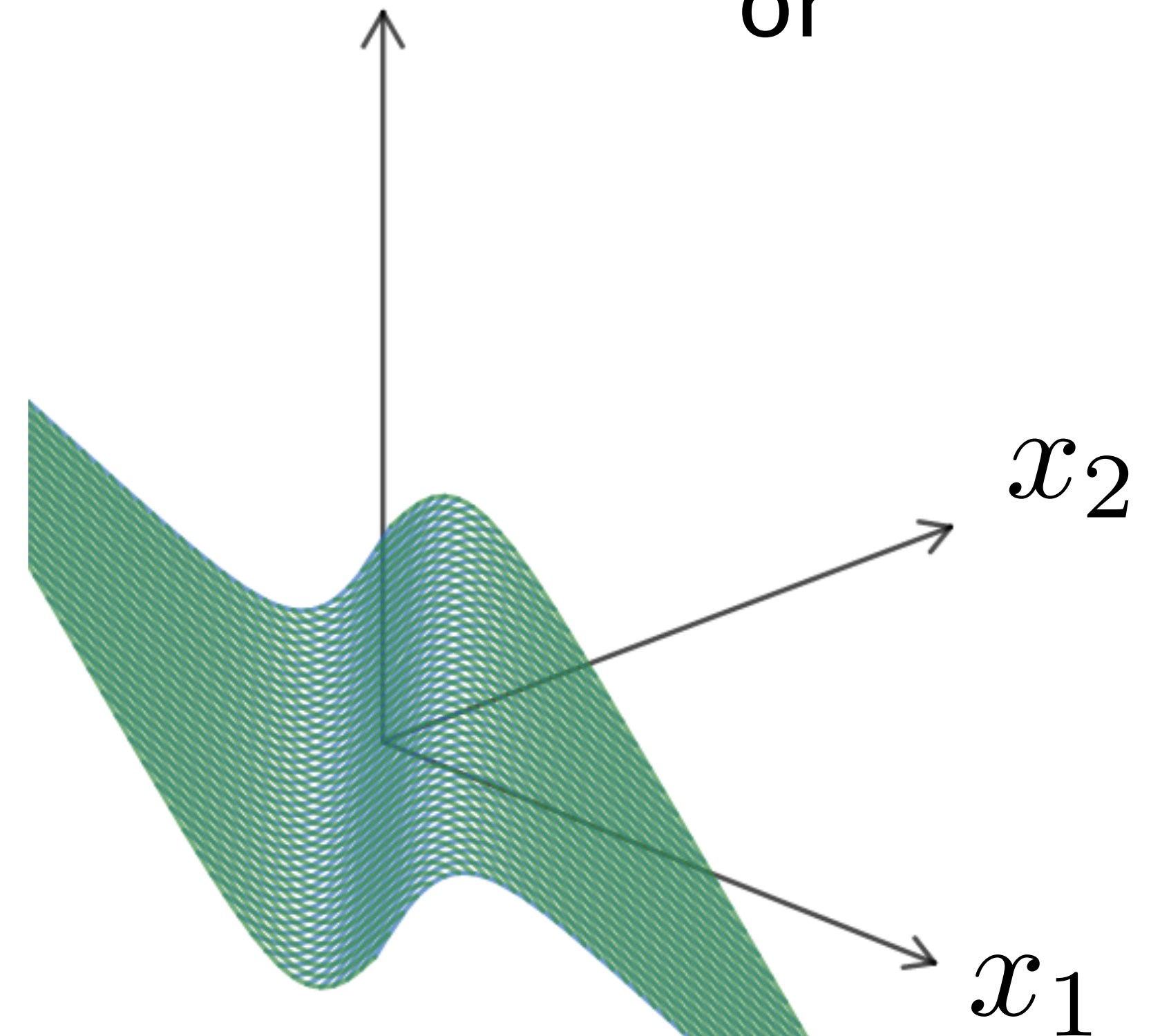
$$y = a_1 x_1 + a_2 x_2 + a_3 \tanh(x_1 + x_2)$$

✓

$$y = -x_1 - x_2 + 2 \tanh(x_1 + x_2)$$

“or”

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

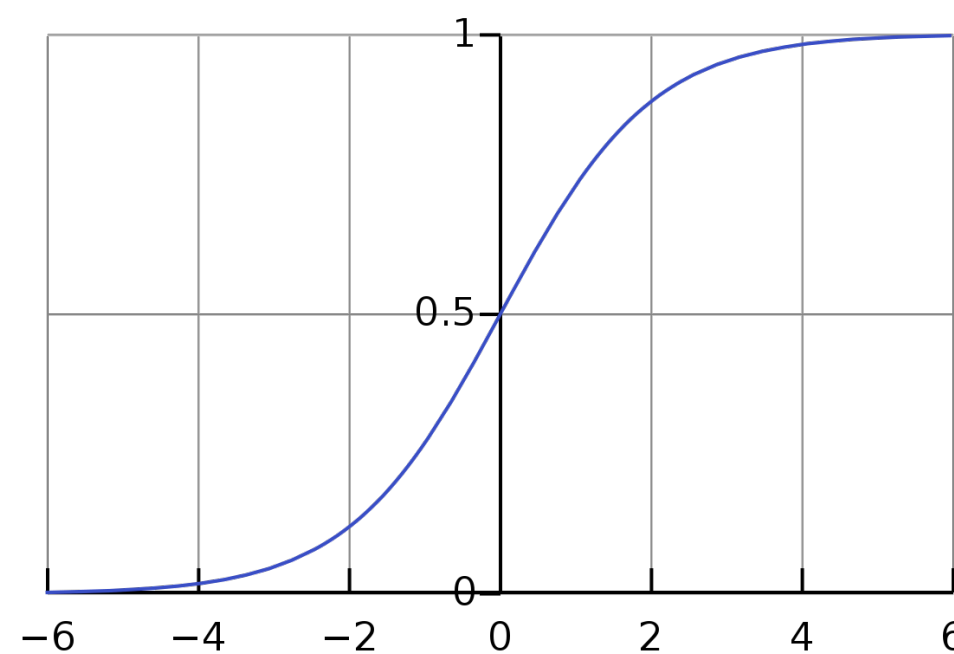


Non linear activation function

- Entry-wise function: $g : \mathbb{R} \rightarrow \mathbb{R}$

sigmoid

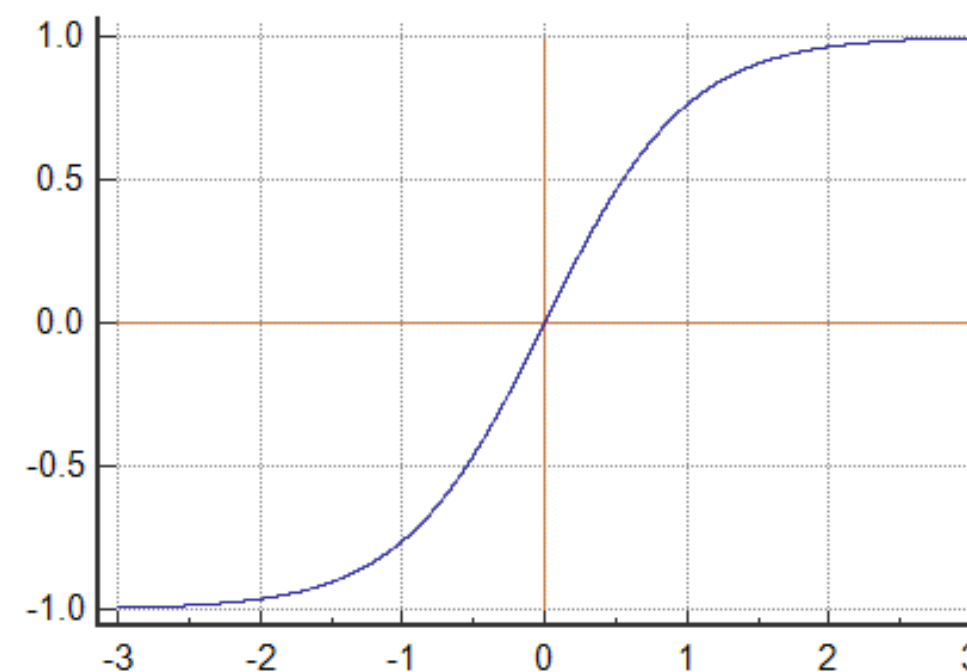
$$g(z) = \frac{1}{1 + e^{-z}}$$



$$g'(z) = f(z) \times (1 - f(z))$$

tanh

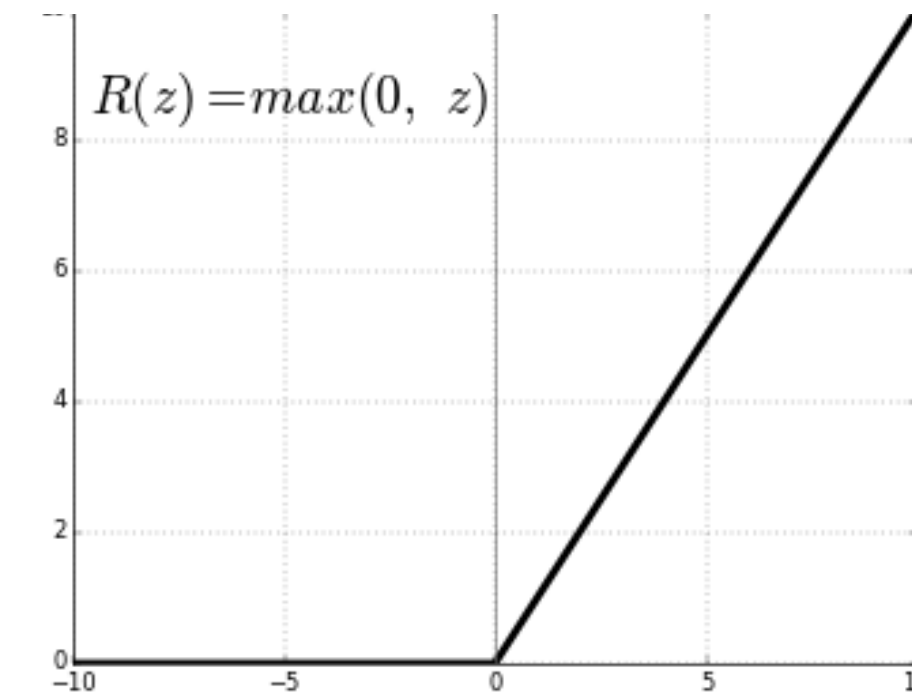
$$g(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$



$$g'(z) = 1 - f(z)^2$$

ReLU
(rectified linear unit)

$$g(z) = \max(0, z)$$

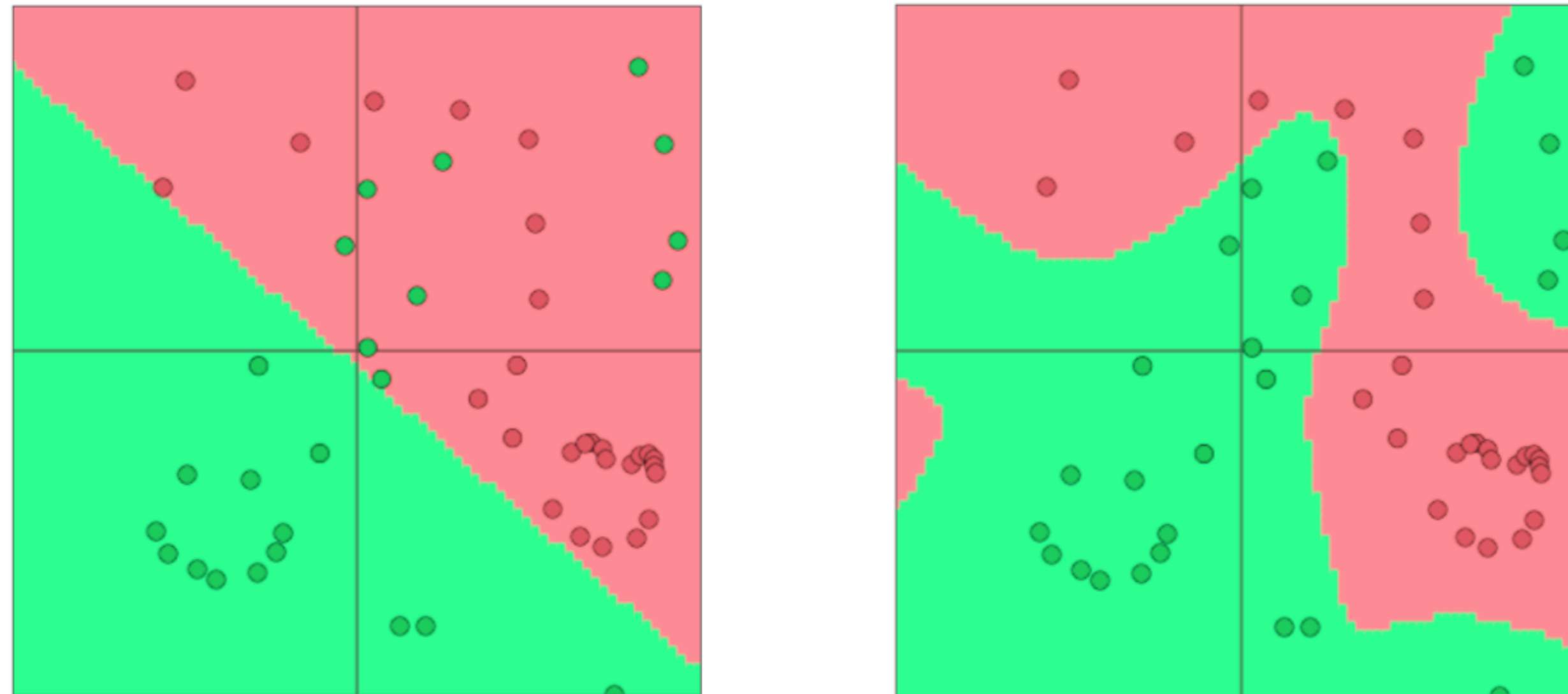


$$g'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases}$$

Advantages of ReLU?

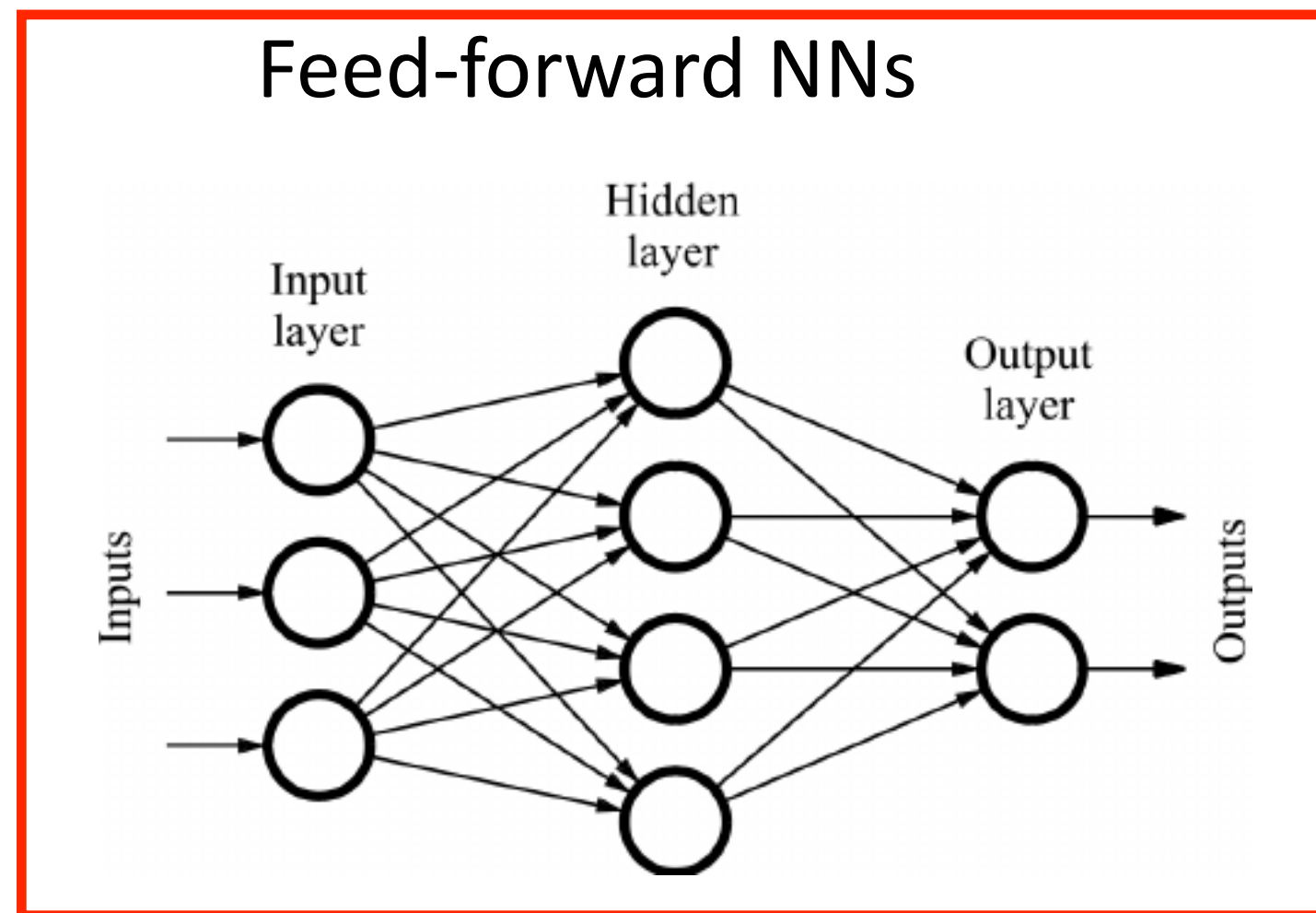
Non-linearity & Deep network

- ▶ Neural network can learn much more complex functions and nonlinear decision boundaries

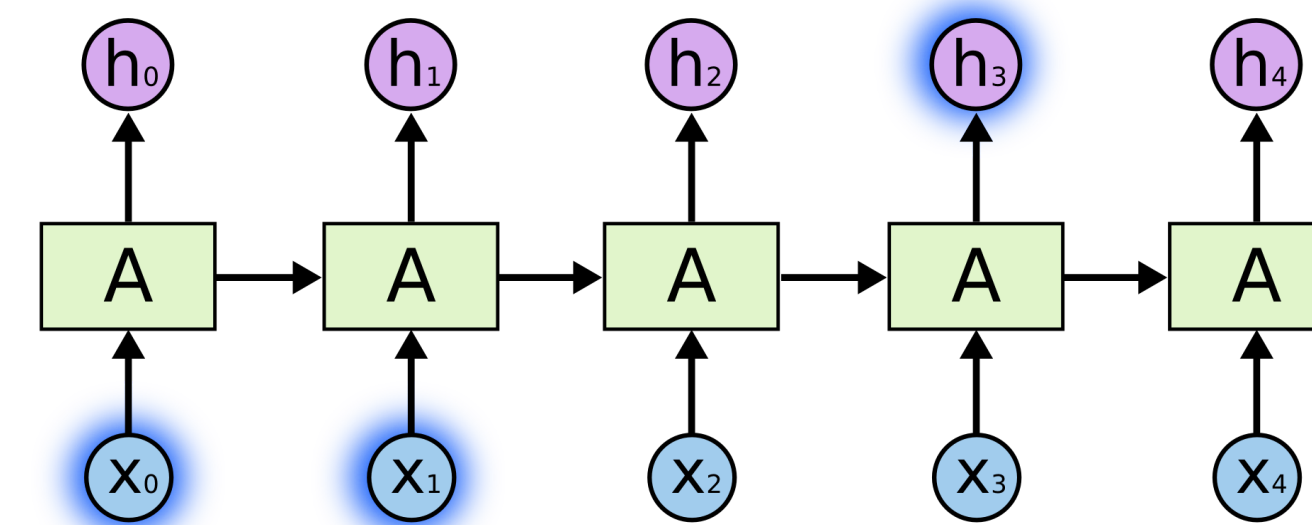


Neural Networks in NLP

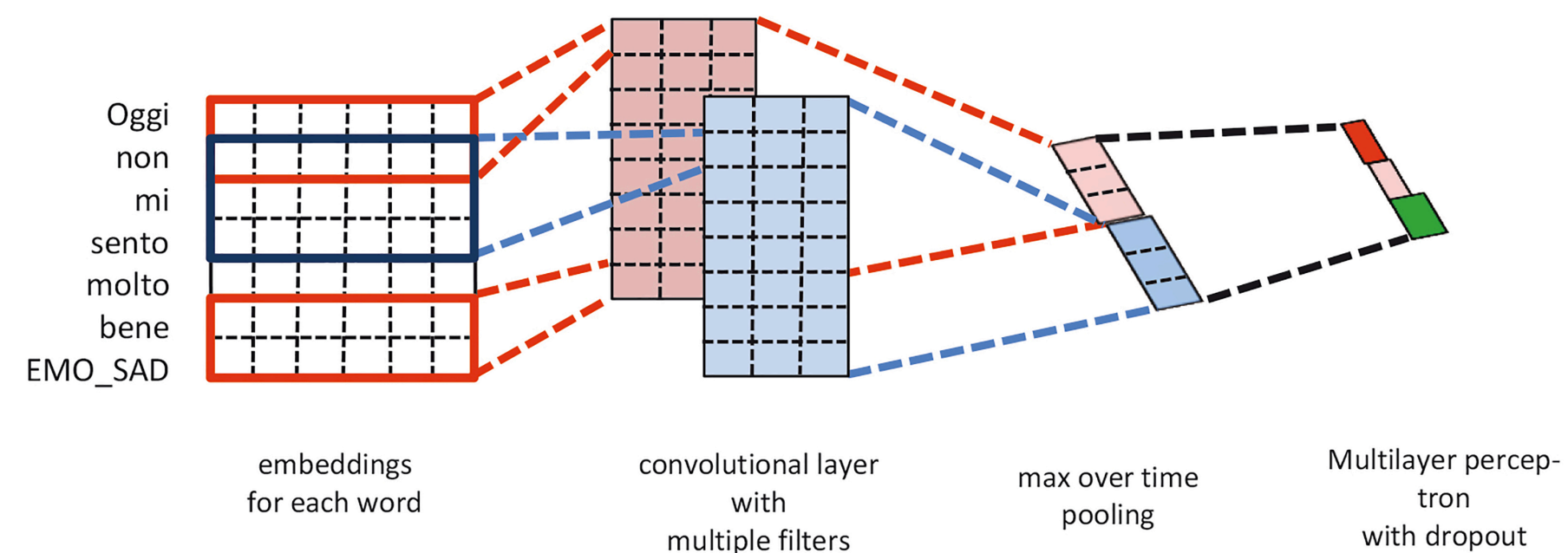
Feed-forward NNs



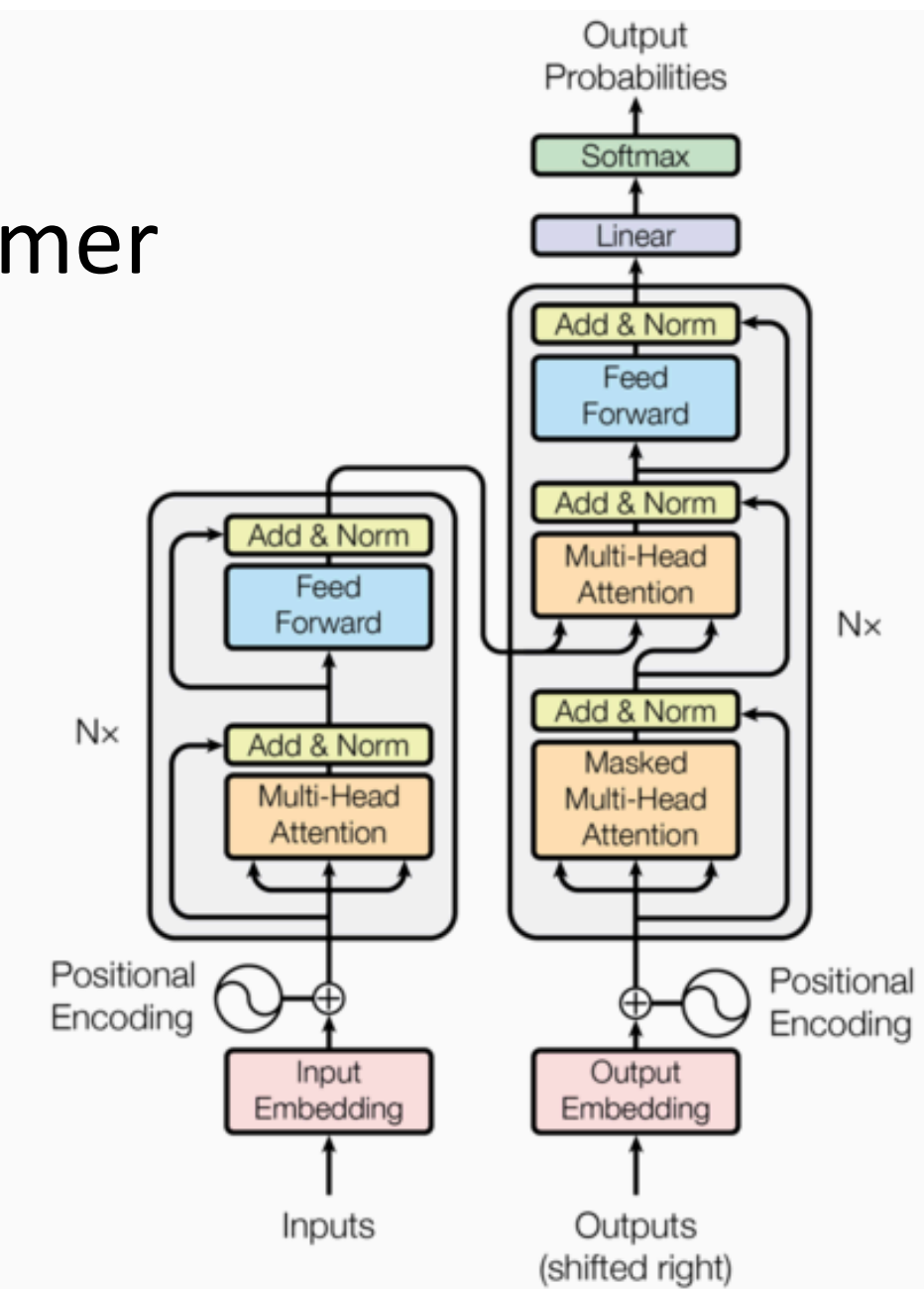
Recurrent NNs



Convolutional NNs



Transformer



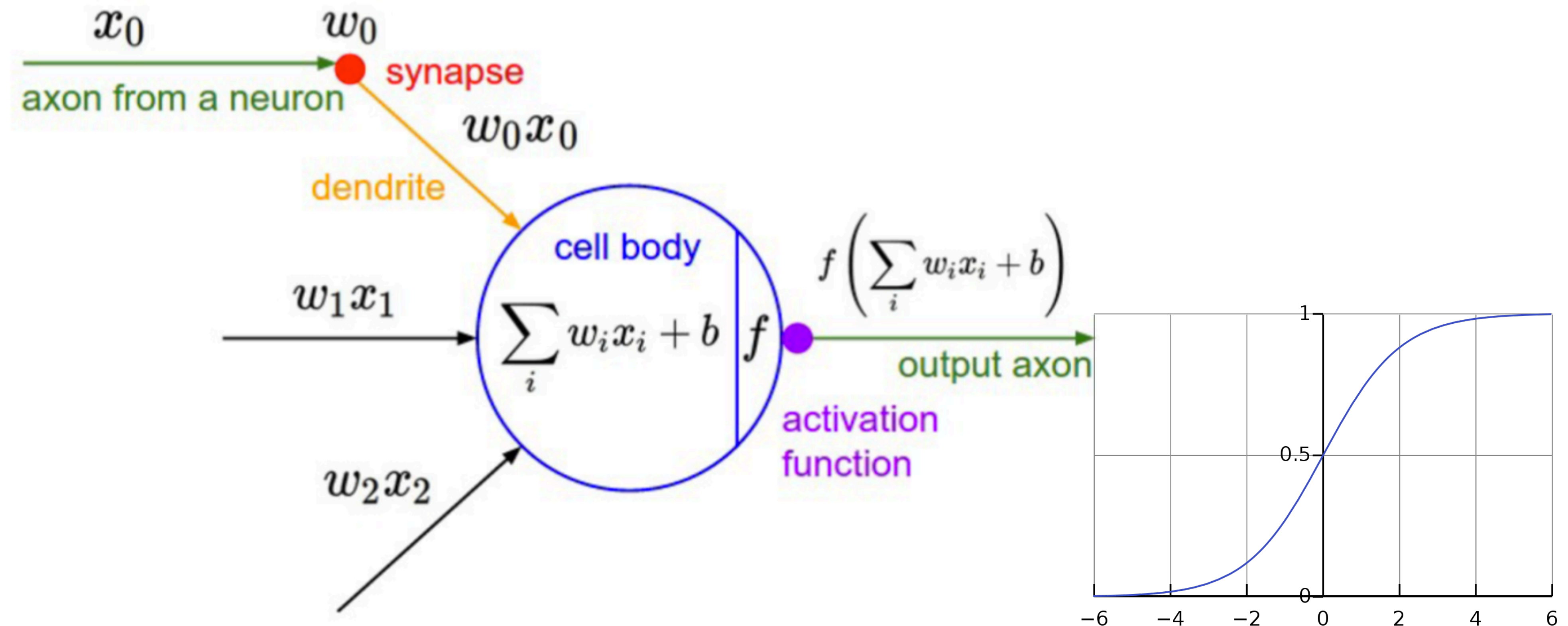
Always coupled with word embeddings...

Credits: Princeton NLP course

Feedforward Networks

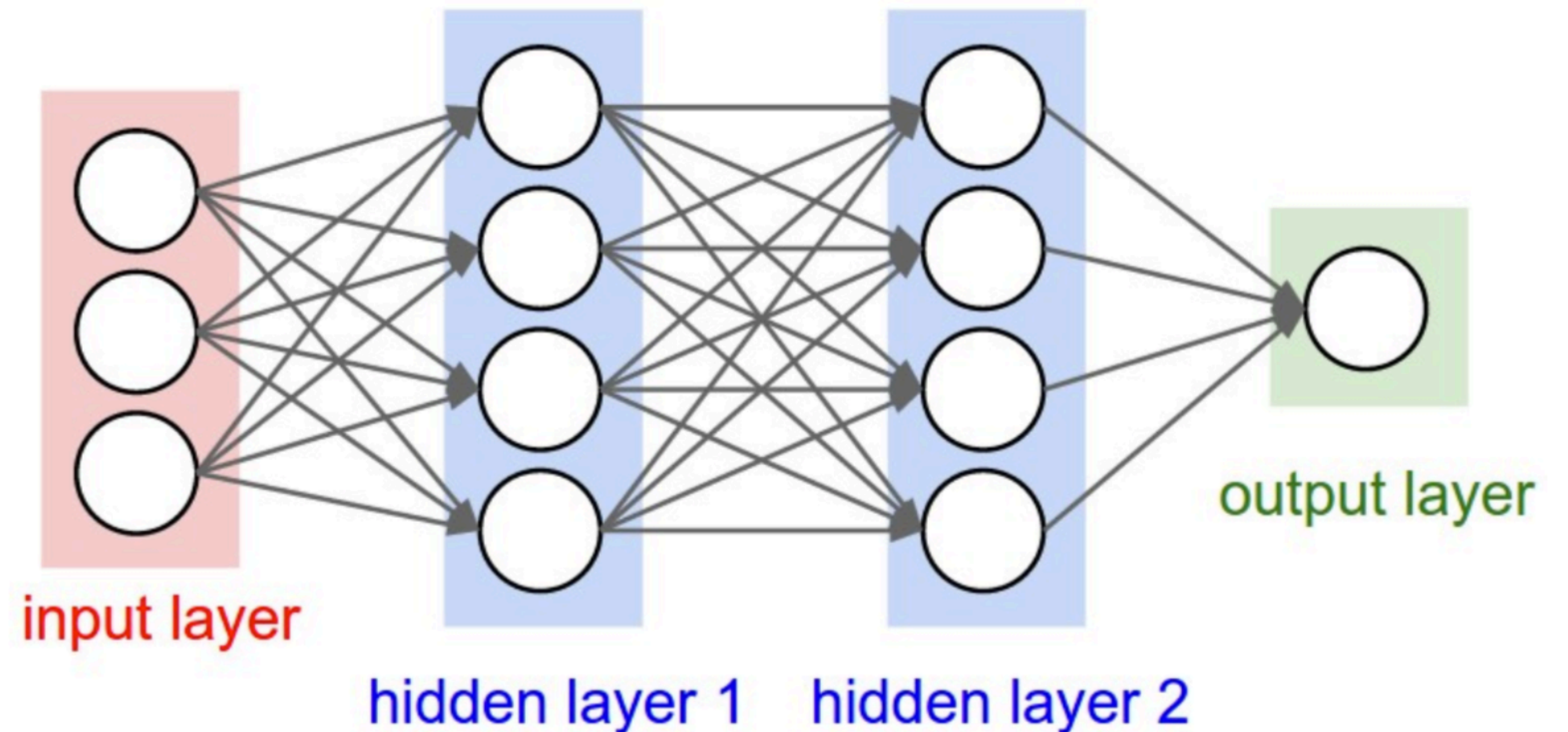
Sigmoid Neuron in Neural Network

- ▶ A single neuron is a computational unit
- ▶ The neuron multiplies each input by its weight, sums them, applied a **nonlinear function** to the result, and passes it to its output.



A neural network

- ▶ If we feed inputs through multiple logistic regression functions, then we can construct a output vector...
- ▶ which we can feed into another logistic regression function as an input.



Acknowledgements

- ▶ We thank the followings for the materials.
 - ▶ Eunsol Choi Natural Language Processing class.
 - ▶ Greg Durrett CS 371-n lecture notes
 - ▶ Yoav Artzi LM class