



CS 540 Introduction to Artificial Intelligence

Midterm Review

University of Wisconsin-Madison
Fall 2025 Sections 1 & 2

Midterm Information

- **Time: October 23rd 7:30-9 PM**
- **Location (by section **):**
 - Section 001 (Tuesday/Thursday 1-2:15PM): 6210 Social Sciences Bldg
 - Section 002 (Tuesday/Thursday 4-5:15PM): B10 Ingraham Hall
 - Students with **McBurney exam accommodations** should have received an email with additional information.
 - Students who cannot take the exam on October 23rd 7:30-9 PM should contact their instructor if they have not done it yet.
- **Topics:** Everything covered in the slides and homework **until Neural Networks III (including Neural Networks III)**
- **Format:** MCQ
- **Cheat sheet:** a handwritten single piece of paper, front and back
- **Calculator:** fine if it doesn't have an Internet connection
- **Bring:** your WISC ID, pencil (No 2 or softer), your cheat sheet.
- **Practice questions:**
 - Canvas -> Files -> Practice Questions
 - Canvas->Quizzes->Practice Questions for Midterm

Bayesian Inference

- Terminology:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

- H is the hypothesis
- E is the evidence



Bayesian Inference


- Terminology:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \longleftarrow \text{Prior}$$

- Prior: estimate of the probability **without** evidence

Bayesian Inference

- Terminology:



A black arrow points from the word "Likelihood" to the term $P(E|H)$ in the numerator of the equation.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Likelihood

- Likelihood: probability of evidence **given a hypothesis**

Bayesian Inference

- Terminology:

$$\underset{\substack{\uparrow \\ \text{Posterior}}}{P(H|E)} = \frac{P(E|H)P(H)}{P(E)}$$

- Posterior: probability of hypothesis **given evidence**.

Naïve Bayes

- Conditional Probability & Bayes:

$$P(H|E_1, E_2, \dots, E_n) = \frac{P(E_1, \dots, E_n|H)P(H)}{P(E_1, E_2, \dots, E_n)}$$

- If we further make the **conditional independence assumption (a.k.a. Naïve Bayes)**

$$P(H|E_1, E_2, \dots, E_n) = \frac{P(E_1|H)P(E_2|H) \cdots P(E_n|H)P(H)}{P(E_1, E_2, \dots, E_n)}$$

Break & Quiz

Q 3.1: 50% of emails are spam. Software has been applied to filter spam. A certain brand of software claims that it can detect 99% of spam emails, and the probability for a false positive (a non-spam email detected as spam) is 5%. Now if an email is detected as spam, then what is the probability that it is in fact a nonspam email?

- A. $5/104$
- B. $95/100$
- C. $1/100$
- D. $1/2$

Break & Quiz

Q 3.1: 50% of emails are spam. Software has been applied to filter spam. A certain brand of software claims that it can detect 99% of spam emails, and the probability for a false positive (a non-spam email detected as spam) is 5%. Now if an email is detected as spam, then what is the probability that it is in fact a nonspam email?

- A. **5/104**
- B. 95/100
- C. 1/100
- D. 1/2

S : Spam

NS: Not Spam

DS: Detected as Spam

$P(S) = 50\%$ spam email

$P(NS) = 50\%$ not spam email

$P(DS|NS) = 5\%$ false positive, detected as spam but not spam

$P(DS|S) = 99\%$ detected as spam and it is spam

Applying Bayes Rule

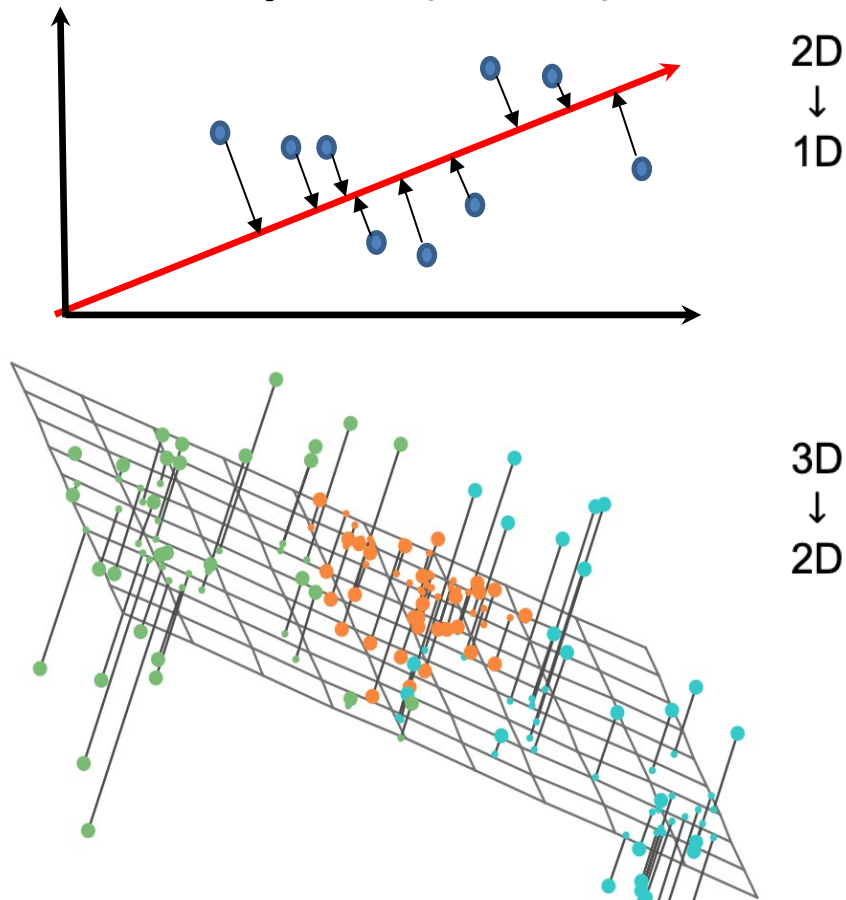
$$P(NS|DS) = (P(DS|NS) * P(NS)) / P(DS) = (P(DS|NS) * P(NS)) / (P(DS|NS) * P(NS) + P(DS|S) * P(S)) = 5/104$$



PCA

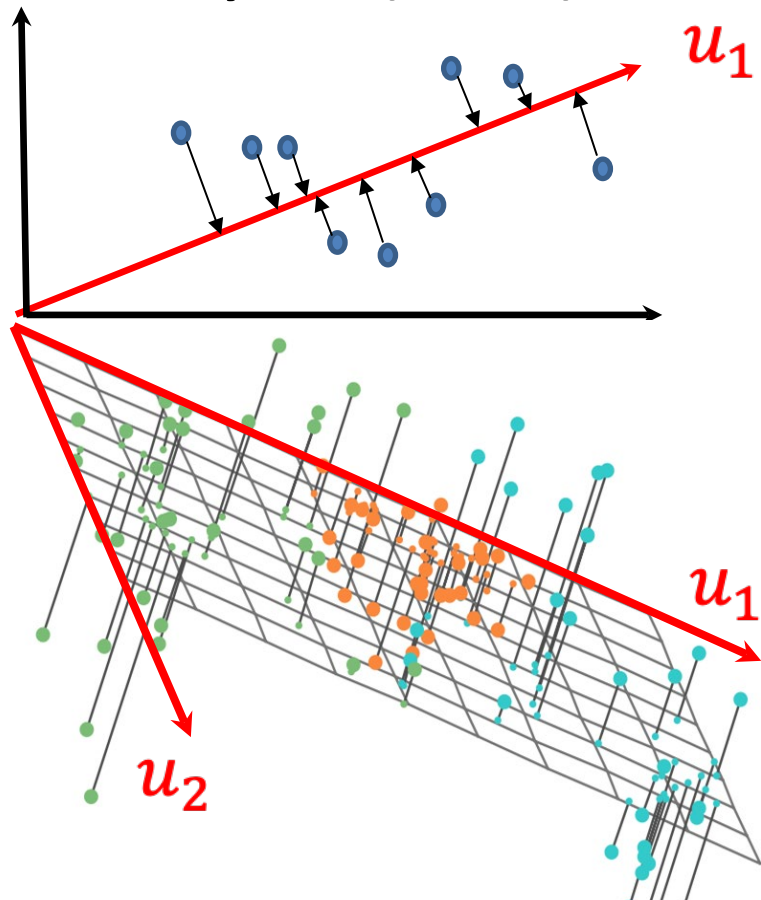
Principal Components Analysis (PCA)

- A type of dimensionality reduction approach
- For when data is **approximately lower dimensional**



Principal Components Analysis (PCA)

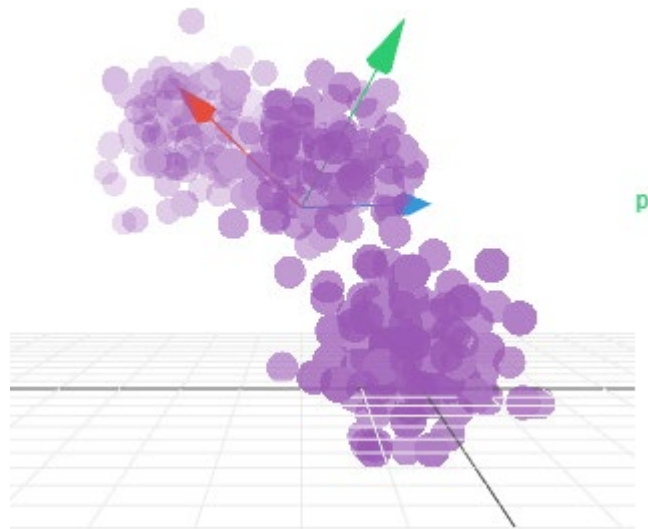
- Find axes $u_1, u_2, \dots, u_m \in \mathbb{R}^d$ of a subspace
 - Will project to this subspace
- Want to preserve data
 - Minimize projection error
- These vectors are the **principal components**



PCA Procedure

Inputs: data $x_1, x_2, \dots, x_n \in \mathbb{R}^d$

— Center data so that $\frac{1}{n} \sum_{i=1}^n x_i = 0$



Victor Powell

PCA Procedure

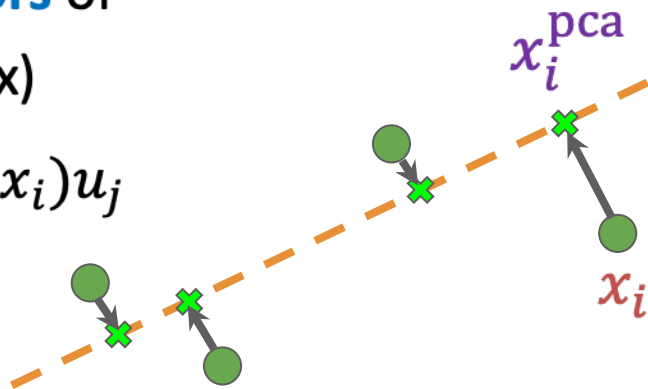
Output:

principal components $u_1, \dots, u_m \in \mathbb{R}^d$

- Orthogonal
- Can show: they are top- m **eigenvectors** of

$$S = \frac{1}{n-1} \sum_{i=1}^n x_i x_i^\top \text{ (covariance matrix)}$$

- Each x_i projected to $x_i^{\text{pca}} = \sum_{j=1}^m (u_j^\top x_i) u_j$





Logic

Propositional Logic Basics

Several rules in place

- Precedence: \neg , \wedge , \vee , \Rightarrow , \Leftrightarrow
- Use parentheses when needed
- Sentences: **well-formed** or not well-formed:
 - $P \Rightarrow Q \Rightarrow S$ **not well-formed (not associative!)**

Evaluating a Sentence

Example:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Note:

- If P is false, $P \Rightarrow Q$ is true regardless of Q (“5 is even implies 6 is odd” is True!)
- Causality not needed: “5 is odd implies the Sun is a star” is True!)

Evaluating a Sentence: Truth Table

- **Ex:**

P	Q	R	$\neg P$	$Q \wedge R$	$\neg P \vee Q \wedge R$	$\neg P \vee Q \wedge R \Rightarrow Q$
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	1	0	1	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	0	1
1	1	0	0	0	0	1
1	1	1	0	1	1	1

- **Satisfiable**

- There exists some interpretation where the sentence is true.

Break & Quiz

Q 1.1: Suppose P is false, Q is true, and R is true. Does this assignment satisfy

(i) $\neg(\neg p \rightarrow \neg q) \wedge r$

(ii) $(\neg p \vee \neg q) \rightarrow (p \vee \neg r)$

- A. Both
- B. Neither
- C. Just (i)
- D. Just (ii)

Break & Quiz

Q 1.1: Suppose P is false, Q is true, and R is true. Does this assignment satisfy

(i) $\neg(\neg p \rightarrow \neg q) \wedge r$

(ii) $(\neg p \vee \neg q) \rightarrow (p \vee \neg r)$

- A. Both
- B. Neither
- **C. Just (i)**
- D. Just (ii)

Break & Quiz

Q 1.1: Suppose P is false, Q is true, and R is true. Does this assignment satisfy

(i) $\neg(\neg p \rightarrow \neg q) \wedge r$

(ii) $(\neg p \vee \neg q) \rightarrow (p \vee \neg r)$

Plug interpretation into each sentence.

- A. Both
- B. Neither
- **C. Just (i)**
- D. Just (ii)

For (i): $(\neg p \rightarrow \neg q)$ will be false so $\neg(\neg p \rightarrow \neg q)$ will be true and r is true by assignment.

For (ii): $(\neg p \vee \neg q)$ is true and $(p \vee \neg r)$ is false which makes the implication false.



NLP

Language Models

Basic idea: use probabilistic models to **assign a probability to a sentence W**

$$P(W) = P(w_1, w_2, \dots, w_n) \text{ or } P(w_{\text{next}} | w_1, w_2 \dots)$$

Goes back to Shannon

- Information theory: letters

Zero-order approximation	XFOML RXKHRJFFJUJ ALPWXFWJXYJ FFJEYVJCQSGHYD QPAAMKBZAACIBZLKJQD
First-order approximation	OCRO HLO RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL
Second-order approximation	ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE
Third-order approximation	IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE
First-order word approximation	REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE

Training: Make Assumptions

- Markov assumption with shorter history:

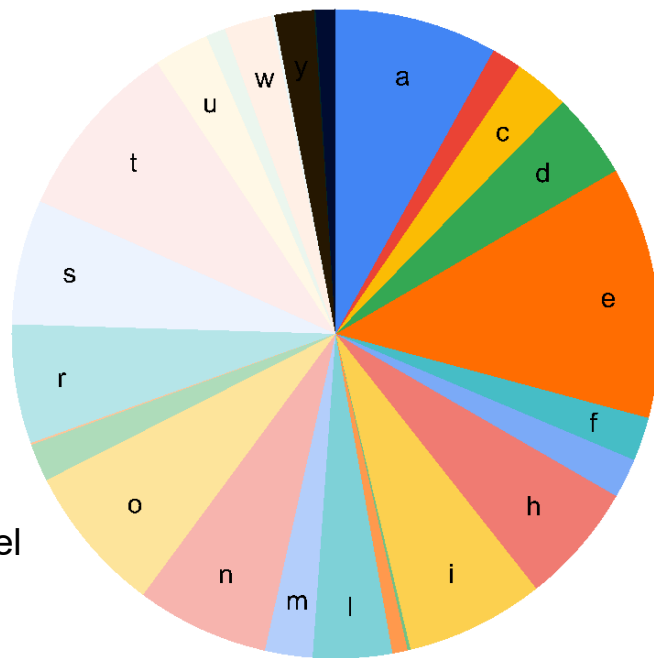
$$P(w_i | w_{i-1} w_{i-2} \dots w_1) = P(w_i | w_{i-1} w_{i-2} \dots w_{i-k})$$

- Present doesn't depend on whole past
 - Just recent past, i.e., *context*.
 - What's ***k=0?***

k=0: Unigram Model

- Full independence assumption:
 - (Present doesn't depend on the past)

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2) \dots P(w_n)$$



The English letter frequency wheel

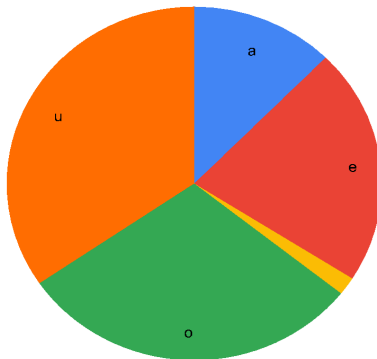
k=1: Bigram Model

- Markov Assumption:
 - (Present depends on immediate past)

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_n|w_{n-1})$$



$p(.|q)$: the “after q” wheel



$p(.|j)$: the “after j” wheel

texaco, rose, one, in, this, issue,
is, pursuing, growth, in, a, boiler,
house, said, mr., gurria, mexico, 's,
motion, control, proposal, without,
permission, from, five, hundred,
fifty, five, yen outside, new, car,
parking, lot, of, the, agreement,
reached this, would, be, a, record,
november

k=n-1: n-gram Model

Can do trigrams, 4-grams, and so on

- More expressive as n goes up
- Harder to estimate

Training: just count? I.e, for bigram:

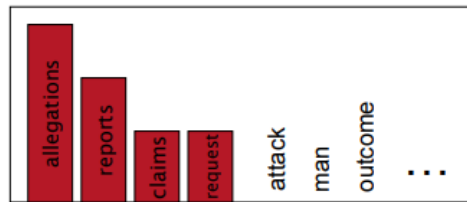
$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

n-gram Training

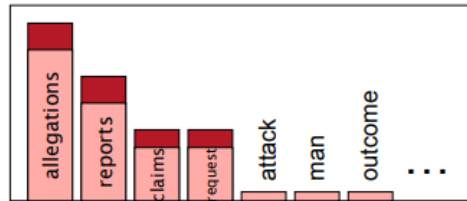
Issues:

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

- 1. Multiply tiny numbers?
 - **Solution:** use logs; add instead of multiply *P(w|denied the)*
- 2. n-grams with zero probability?
 - **Solution:** (Laplace) smoothing



$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i) + 1}{\text{count}(w_{i-1}) + V}$$



Break & Quiz

Q 1. Suppose we have the following training corpus in an NLP setup.

“I took my dog out for a walk but my dog ran away”

- (i) What is the $P(\text{dog})$ using a unigram model without Laplace smoothing?
- (ii) What is the $P(\text{my dog})$ using a unigram model without Laplace smoothing?
- (iii) What is the $P(\text{my dog})$ using a bigram model without Laplace smoothing?
- (iv) What is the $P(\text{dog} \mid \text{my})$ for a bigram model without Laplace smoothing

Break & Quiz

Q 1. Suppose we have the following training corpus in an NLP setup.

“I took my dog out for a walk but my dog ran away”

- (i) What is the $P(\text{dog})$ using a unigram model without Laplace smoothing?
- (ii) What is the $P(\text{my dog})$ using a unigram model without Laplace smoothing?
- (iii) What is the $P(\text{my dog})$ using a bigram model without Laplace smoothing?
- (iv) What is the $P(\text{dog} \mid \text{my})$ for a bigram model without Laplace smoothing?

- (i) $P(\text{dog}) = 2 / 13$ (note there are 13 words in the sentence; two are 'dog').
- (ii) $P(\text{my dog}) = P(\text{my}) * P(\text{dog}) = (2/13) * (2/13) = 4/169$
- (iii) $P(\text{my dog}) = 2 / 12 = 1 / 6$
- (iv) $P(\text{dog} \mid \text{my}) = \text{count}(\text{my,dog}) / \text{count}(\text{my}) = 2/2 = 1$



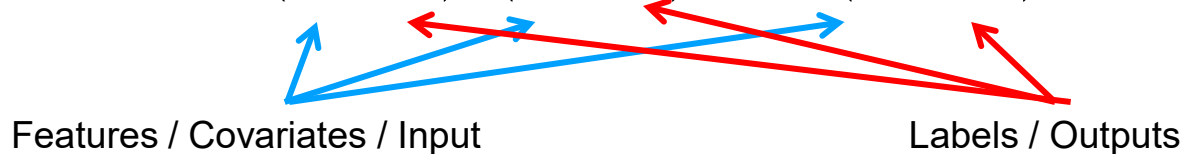
Machine Learning

Supervised/Unsupervised Learning

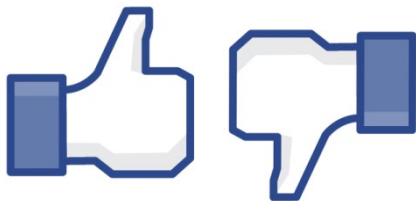
Supervised learning:

Make predictions, classify data, perform regression

Dataset: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$



Goal: find function $f : X \rightarrow Y$ to predict label on **new** data



indoor



outdoor

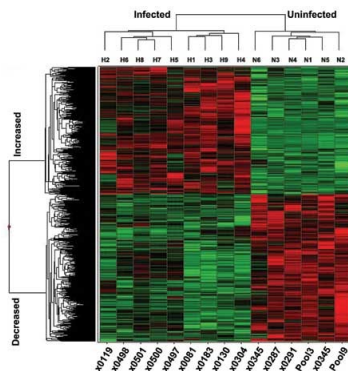
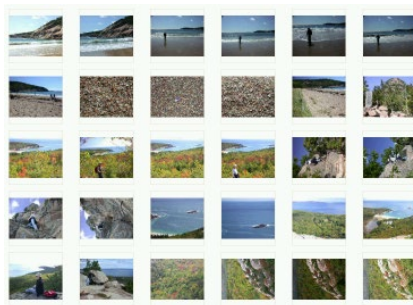
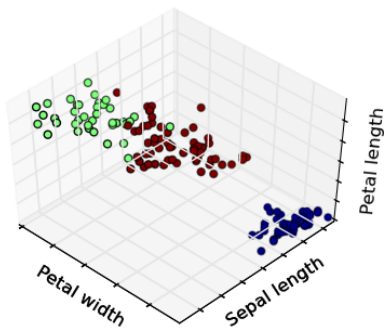
Supervised/Unsupervised Learning

Unsupervised learning:

No labels; generally, won't be making predictions

Dataset: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$

Goal: find patterns & structures that help better understand data.

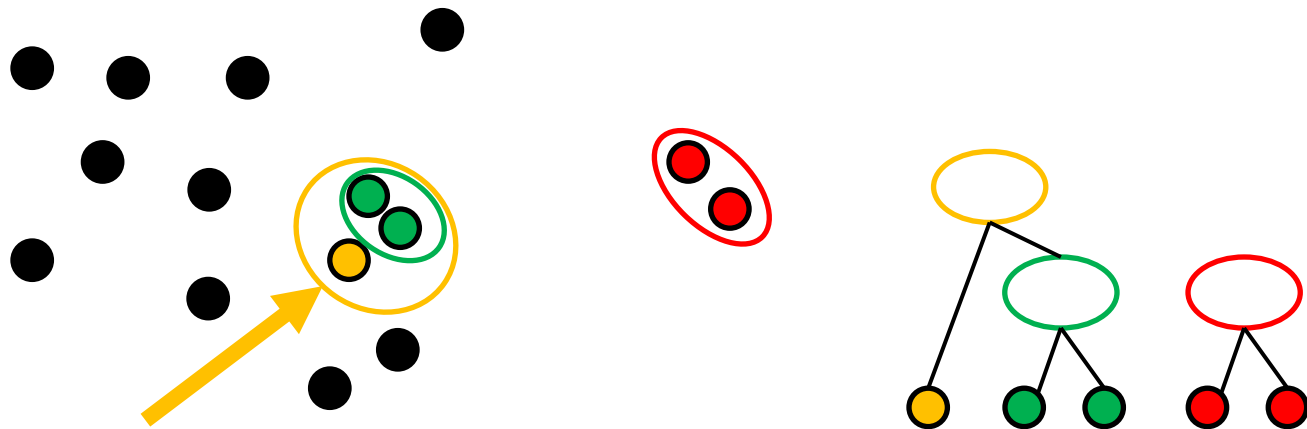




Unsupervised Learning

Agglomerative Clustering Example

Repeat: Get pair of clusters that are closest and merge



Merging Criteria

Merge: use closest clusters. Define closest?

Single-linkage

$$d(A, B) = \min_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

Complete-linkage

$$d(A, B) = \max_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

Average-linkage

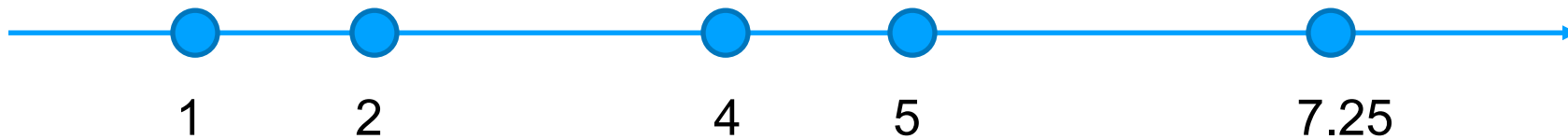
$$d(A, B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

Single-linkage Example

We'll merge using single-linkage

1-dimensional vectors.

Initial: all points are clusters

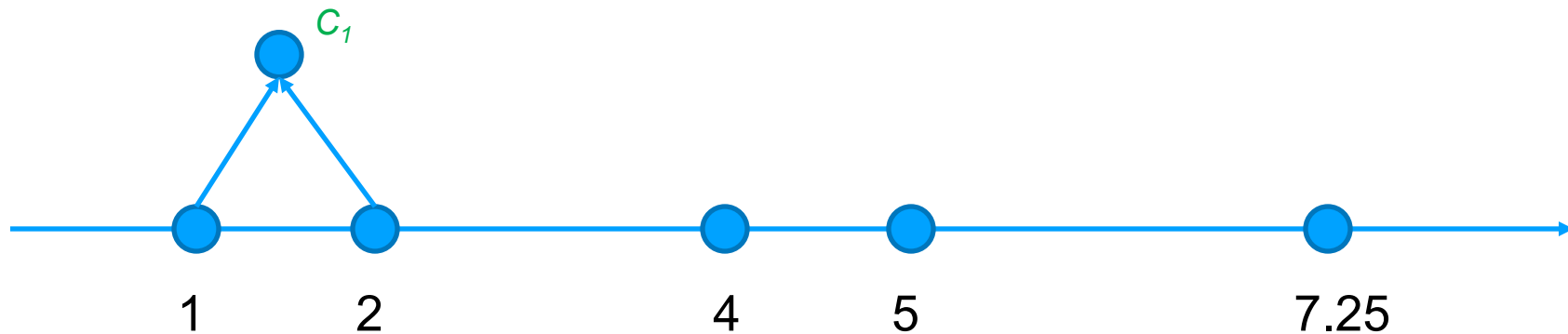


Single-linkage Example

We'll merge using single-linkage

$$d(C_1, \{4\}) = d(2, 4) = 2$$

$$d(\{4\}, \{5\}) = d(4, 5) = 1$$

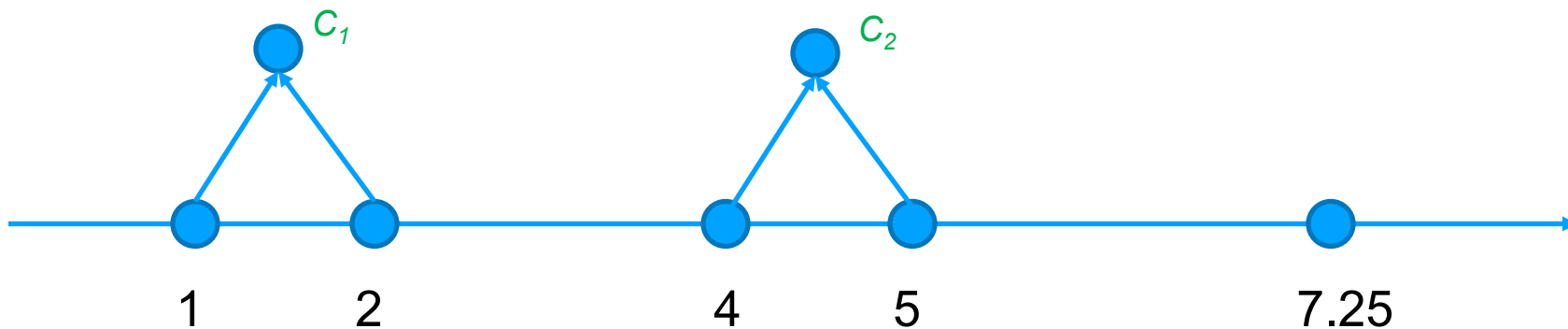


Single-linkage Example

Continue...

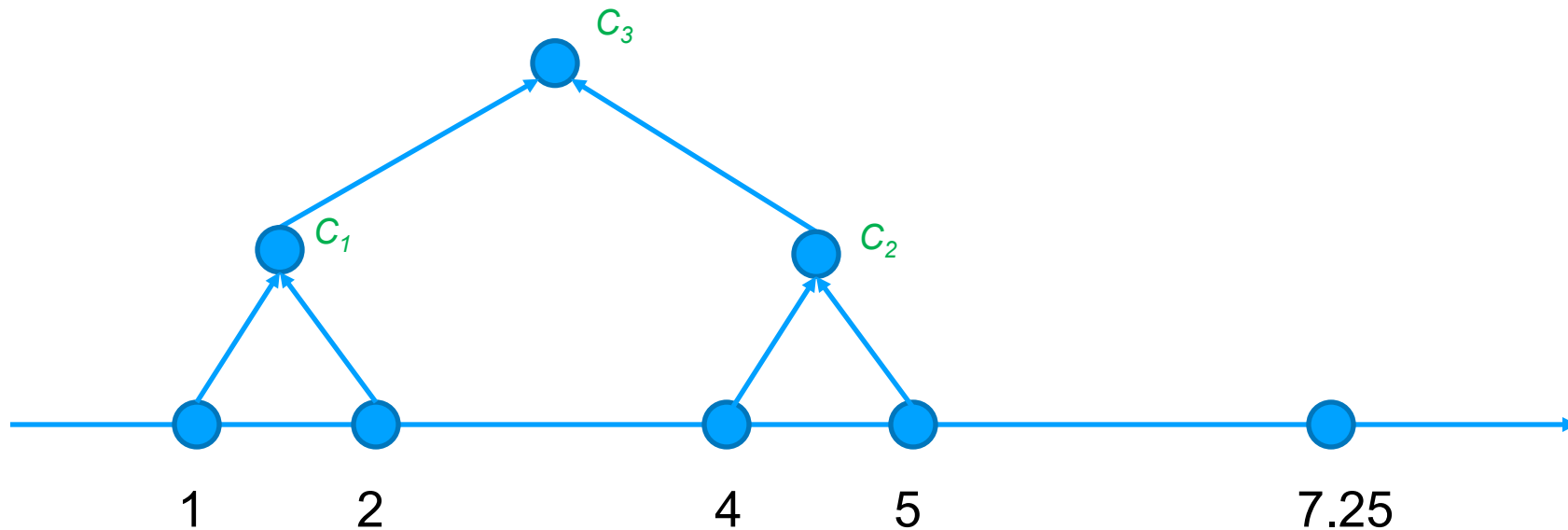
$$d(C_1, C_2) = d(2, 4) = 2$$

$$d(C_2, \{7.25\}) = d(5, 7.25) = 2.25$$

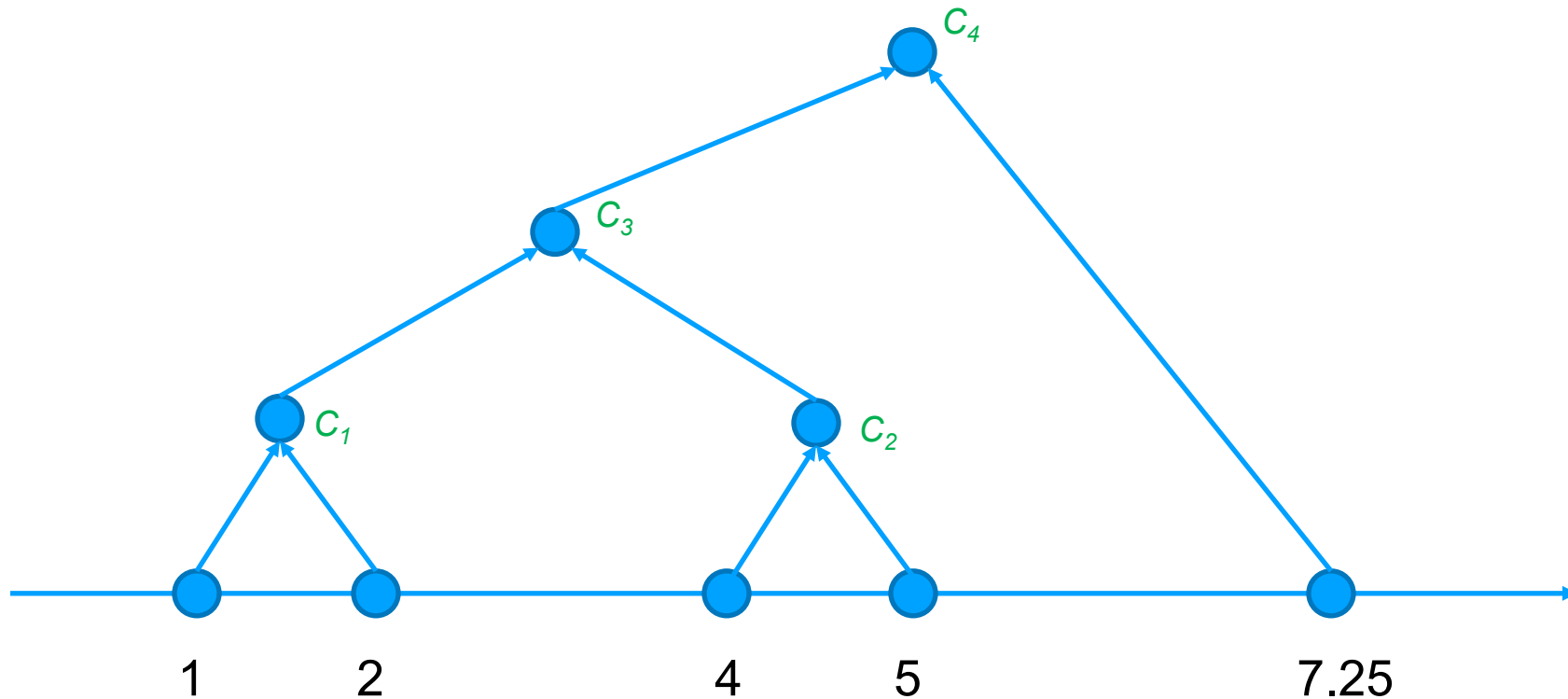


Single-linkage Example

Continue...



Single-linkage Example

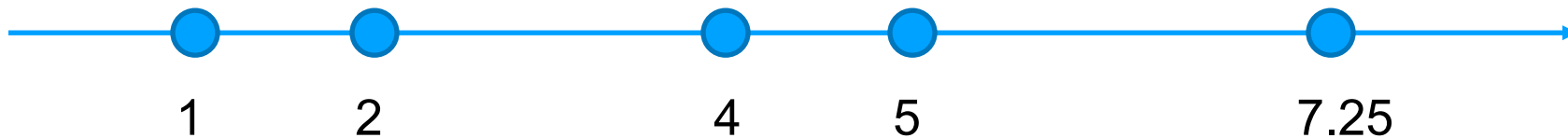


Complete-linkage Example

We'll merge using complete-linkage

1-dimensional vectors.

Initial: all points are clusters

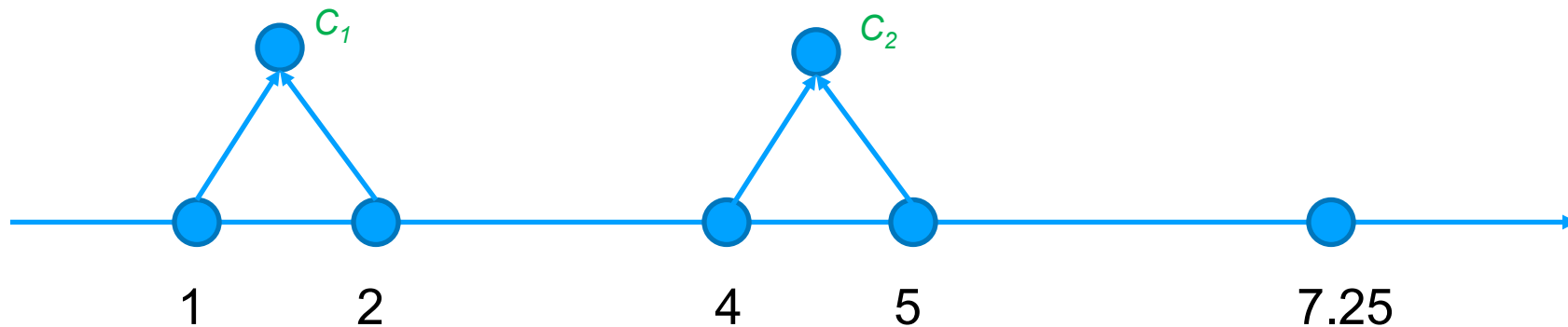


Complete-linkage Example

Beginning is the same...

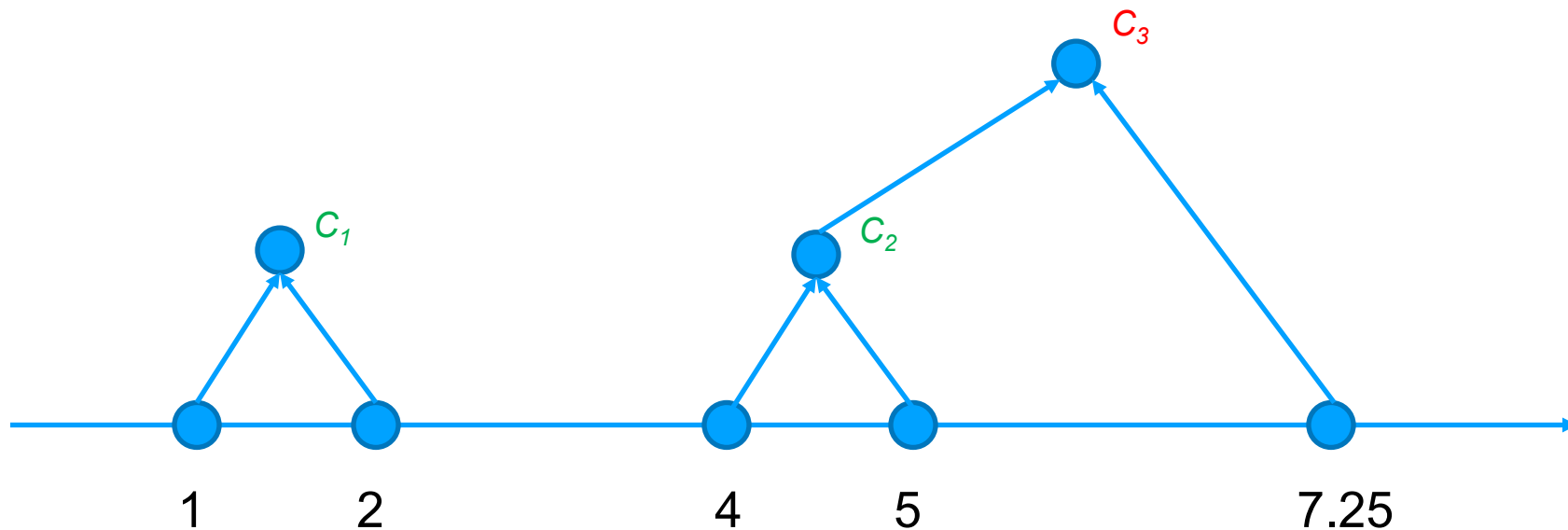
$$d(C_1, C_2) = d(1, 5) = 4$$

$$d(C_2, \{7.25\}) = d(4, 7.25) = 3.25$$

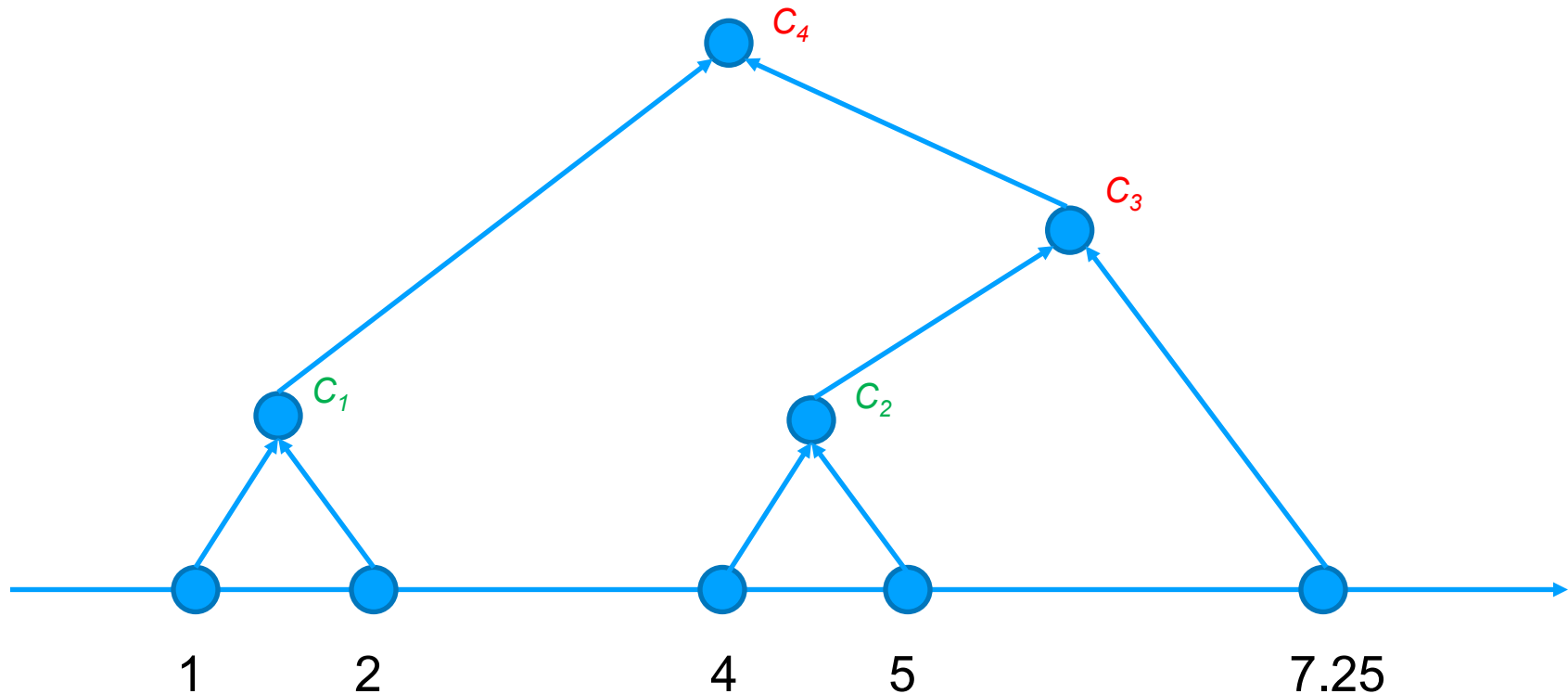


Complete-linkage Example

Now we diverge:



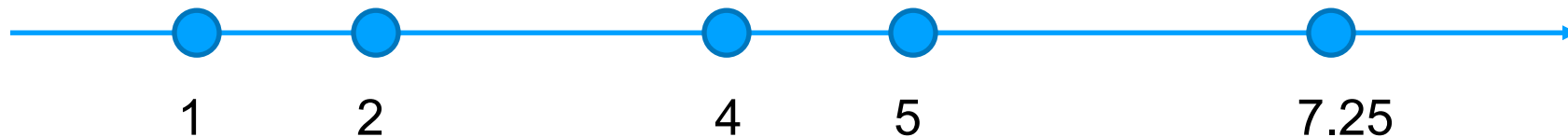
Complete-linkage Example



Break & Quiz

Q 1.1: Let's do hierarchical clustering for two clusters with average linkage on the dataset below. What are the clusters?

- A. $\{1\}, \{2, 4, 5, 7.25\}$
- B. $\{1, 2\}, \{4, 5, 7.25\}$
- C. $\{1, 2, 4\}, \{5, 7.25\}$
- D. $\{1, 2, 4, 5\}, \{7.25\}$



Break & Quiz

Q 1.1: Let's do hierarchical clustering for two clusters with average linkage on the dataset below. What are the clusters?

- A. $\{1\}, \{2, 4, 5, 7.25\}$
- B. $\{1, 2\}, \{4, 5, 7.25\}$**
- C. $\{1, 2, 4\}, \{5, 7.25\}$
- D. $\{1, 2, 4, 5\}, \{7.25\}$

Iteration 1: merge 1 and 2

Iteration 2: merge 4 and 5

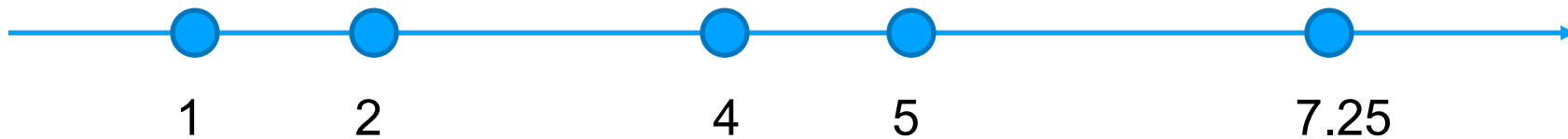
Iteration 3: Now we have clusters $\{1, 2\}, \{4, 5\}, \{7.25\}$.

$\text{distance}(\{1, 2\}, \{4, 5\}) = 3$

$\text{distance}(\{4, 5\}, \{7.25\}) = 2.75$

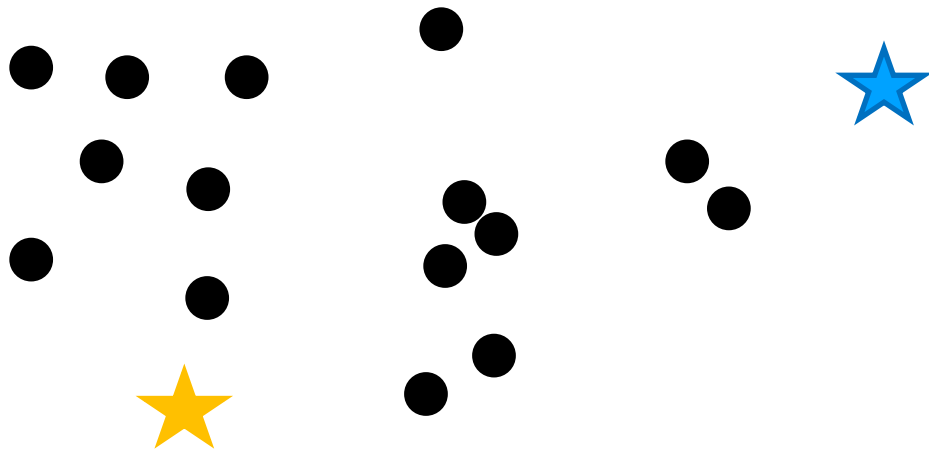
$\text{distance}(\{1, 2\}, \{7.25\})$ is clearly larger than the above two.

So average linkage will merge $\{4, 5\}$ and $\{7.25\}$



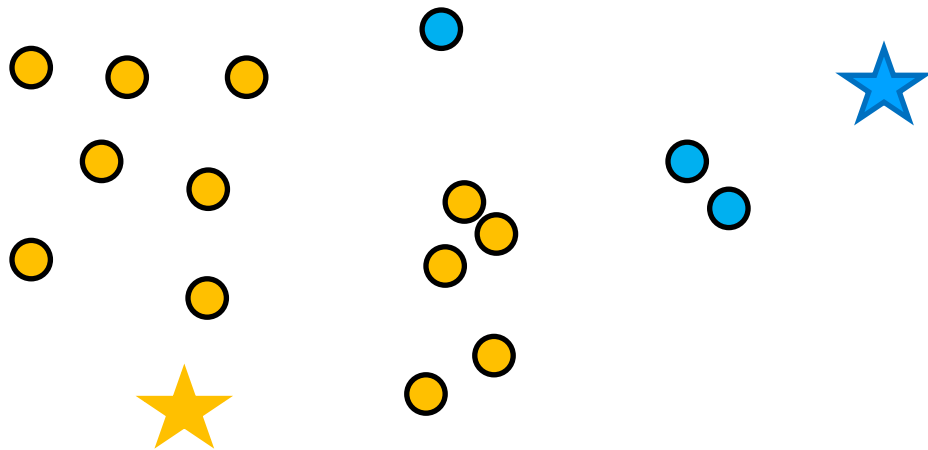
K-Means Clustering

Steps: 1. Randomly pick k cluster centers



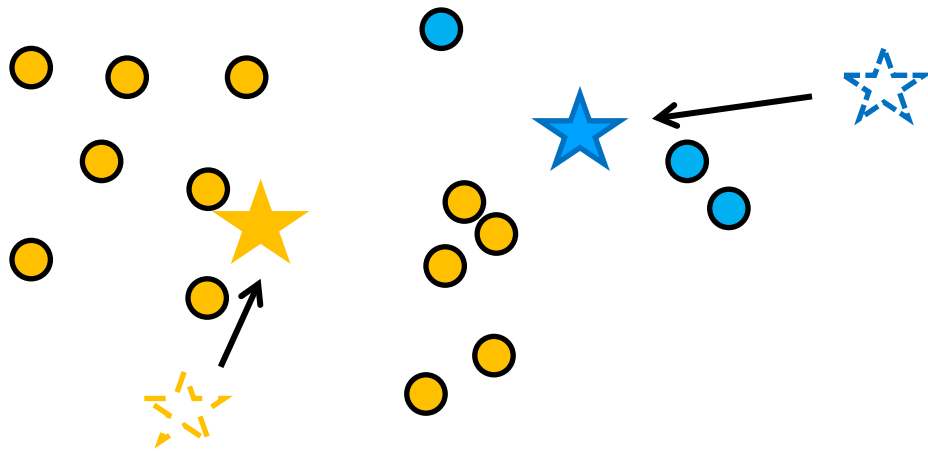
K-Means Clustering

2. Find closest center for each point



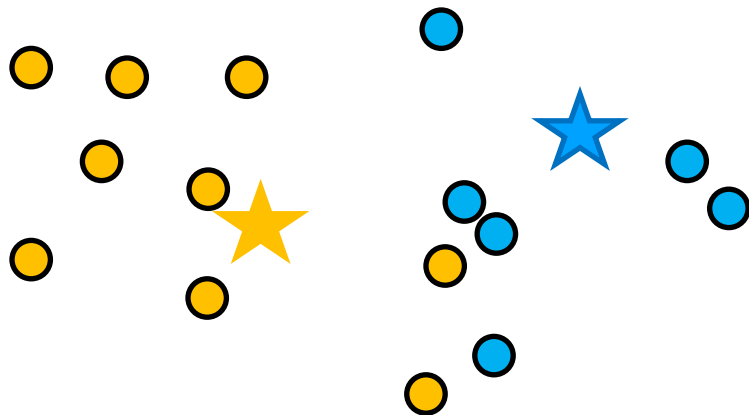
K-Means Clustering

3. Update cluster centers by computing centroids



K-Means Clustering

Repeat Steps 2 & 3 until convergence



K-means algorithm

Input: x_1, x_2, \dots, x_n, k

Step 1: select k cluster centers c_1, c_2, \dots, c_k

Step 2: for each point x_i , assign it to the closest center in Euclidean distance:

$$y(x_i) = \operatorname{argmin}_j ||x_i - c_j||$$

Step 3: update all cluster centers as the centroids:

$$c_j = \frac{\sum_{x:y(x)=j} x}{\sum_{x:y(x)=j} 1}$$

Repeat Step 2 and 3 until cluster centers no longer change

Questions on k-means

- What is k-means trying to optimize?
- Will k-means stop (converge)?
- Will it find a global or local optimum?
- How to pick starting cluster centers?
- How many clusters should we use?

Break & Quiz

Q 2.1: You have seven 2-dimensional points. You run 3-means on it, with initial clusters

$$C_1 = \{(2, 2), (4, 4), (6, 6)\}, C_2 = \{(0, 4), (4, 0)\}, C_3 = \{(5, 5), (9, 9)\}$$

Cluster centroids are updated to?

- A. $C_1: (4,4)$, $C_2: (2,2)$, $C_3: (7,7)$
- B. $C_1: (6,6)$, $C_2: (4,4)$, $C_3: (9,9)$
- C. $C_1: (2,2)$, $C_2: (0,0)$, $C_3: (5,5)$
- D. $C_1: (2,6)$, $C_2: (0,4)$, $C_3: (5,9)$

Break & Quiz

Q 2.1: You have seven 2-dimensional points. You run 3-means on it, with initial clusters

$$C_1 = \{(2, 2), (4, 4), (6, 6)\}, C_2 = \{(0, 4), (4, 0)\}, C_3 = \{(5, 5), (9, 9)\}$$

Cluster centroids are updated to?

- **A. $C_1: (4,4)$, $C_2: (2,2)$, $C_3: (7,7)$**
- B. $C_1: (6,6)$, $C_2: (4,4)$, $C_3: (9,9)$
- C. $C_1: (2,2)$, $C_2: (0,0)$, $C_3: (5,5)$
- D. $C_1: (2,6)$, $C_2: (0,4)$, $C_3: (5,9)$

The average of points in C1 is (4,4).

The average of points in C2 is (2,2).

The average of points in C3 is (7,7).

Break & Quiz

Q 2.2: We are running 3-means again. We have 3 centers, C_1 (0,1), C_2 (2,1), C_3 (-1,2). Which cluster assignment is possible for the points (1,1) and (-1,1), respectively? Ties are broken arbitrarily:

(i) C_1, C_1 (ii) C_2, C_3 (iii) C_1, C_3

- A. Only (i)
- B. Only (ii) and (iii)
- C. Only (i) and (iii)
- D. All of them

Break & Quiz

Q 2.2: We are running 3-means again. We have 3 centers, C_1 (0,1), C_2 (2,1), C_3 (-1,2). Which cluster assignment is possible for the points (1,1) and (-1,1), respectively? Ties are broken arbitrarily:

(i) C_1, C_1 (ii) C_2, C_3 (iii) C_1, C_3

- A. Only (i)
- B. Only (ii) and (iii)
- C. Only (i) and (iii)
- **D. All of them**

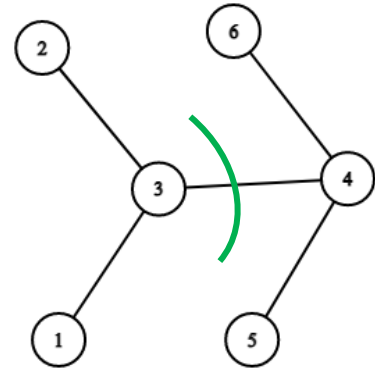
For the point (1,1): square-Euclidean-distance to C_1 is 1, to C_2 is 1, to C_3 is 5
So it can be assigned to C_1 or C_2

For the point (-1,1): square-Euclidean-distance to C_1 is 1, to C_2 is 9, to C_3 is 1
So it can be assigned to C_1 or C_3

Graph-Based Clustering

Want: partition V into V_1 and V_2

- Implies a graph “cut”
- One idea: minimize the **weight** of the cut



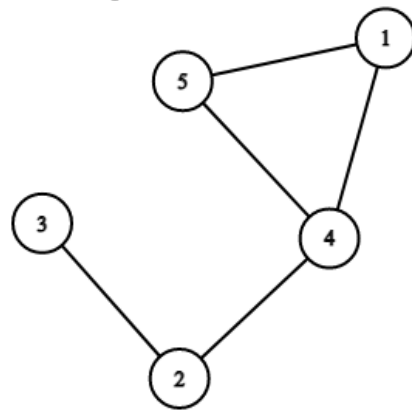
$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

$$\text{cut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k W(A_i, \bar{A}_i).$$

Partition-Based Clustering

How do we compute these?

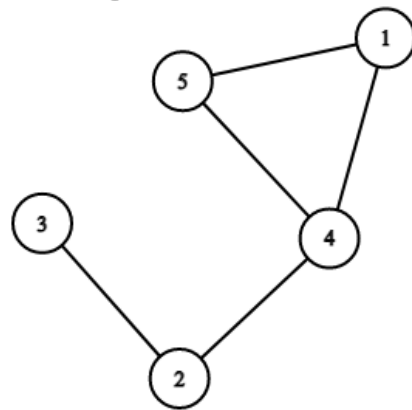
- Hard problem \rightarrow heuristics
 - Greedy algorithm
 - “Spectral” approaches
- Spectral clustering approach:
 - **Adjacency** matrix



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Partition-Based Clustering

- Spectral clustering approach:
 - **Adjacency** matrix
 - **Degree** matrix

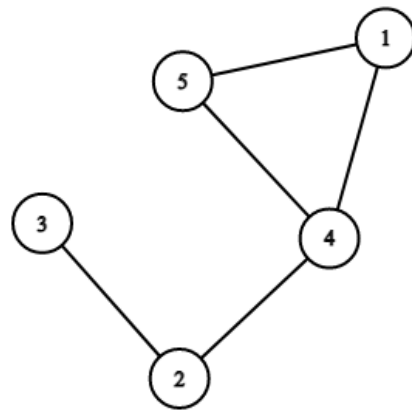


$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Spectral Clustering

- Spectral clustering approach:
 - 1. Compute **Laplacian** $L = D - A$
(Important tool in graph theory)



$$L = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}}_{\text{Degree Matrix}} - \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Adjacency Matrix}} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}}_{\text{Laplacian}}$$

Spectral Clustering

- Spectral clustering approach:

- 1. Compute **Laplacian** $\mathbf{L} = \mathbf{D} - \mathbf{A}$

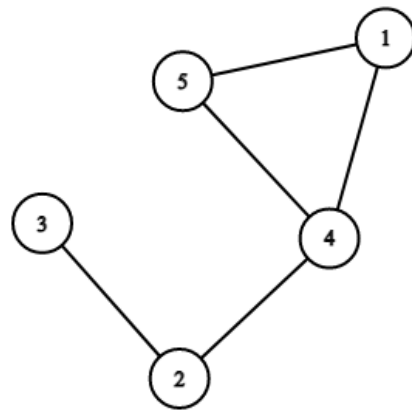
- 1a (optional): compute normalized Laplacian:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}, \text{ or } \mathbf{L} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A}$$

- 2. Compute k **smallest** eigenvectors of \mathbf{L}

- 3. Set U to be the $n \times k$ matrix with u_1, \dots, u_k as columns. Take the n rows formed as points

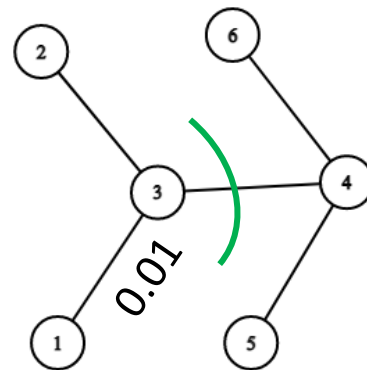
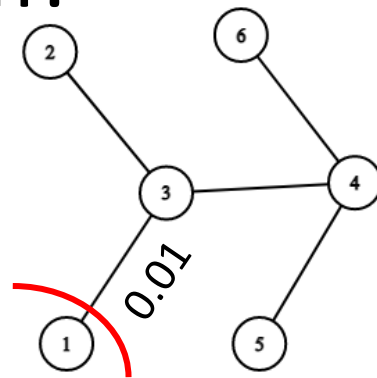
- 4. Run k-means on the representations



Why normalized Laplacian?

Want: partition V into V_1 and V_2

- Implies a graph “cut”
- One idea: minimize the **weight** of the cut
 - Downside: might just cut off one node
 - Need: “**balanced**” cut



Why Normalized Laplacian?

Want: partition V into V_1 and V_2

- Just minimizing weight is not always a good idea.
- We want **balance**!

$$\text{Ncut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{\text{vol}(A_i)}$$

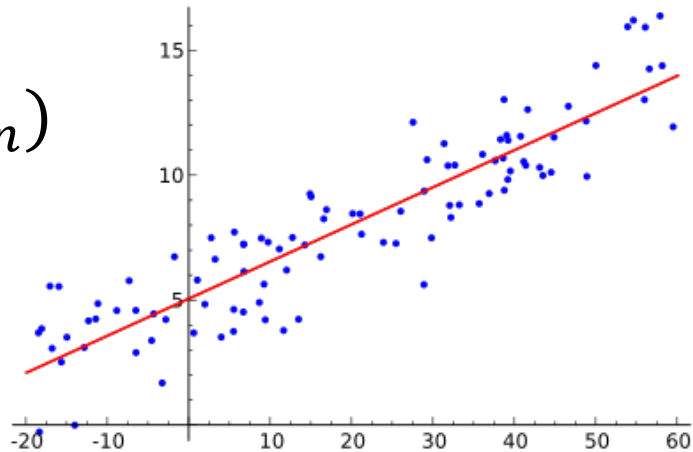
$$\text{vol}(A) = \sum_{i \in A} \text{degree}(i)$$



Linear Regression

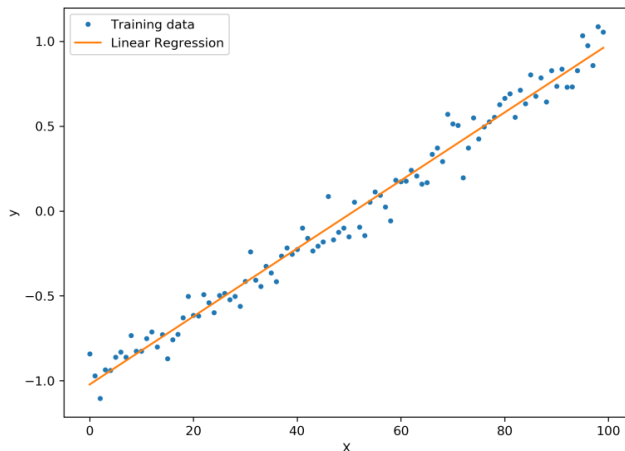
Linear Regression

- Simplest form of regression
- Find a line that fits the data
- Example:
 - Training data $(x_1, y_1), \dots, (x_n, y_n)$
 - Find $a, b \in \mathbb{R}$ such that
$$\forall i, y_i \approx ax_i + b$$



Linear Regression

- **Inputs:** $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$
 - \mathbf{x} 's are vectors, y 's are scalars.
 - “**Linear**”: predict a linear combination of \mathbf{x} components + intercept



C. Hansen

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \theta_0 + x^T \theta$$

- **Want:** parameters θ

Linear Regression Setup

Problem Setup

- Goal: figure out how to minimize square loss
- Let's organize it. Train set

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$$

Notational Trick

- When x is a scalar: $f_{(a,b)}(x) = ax + b$

- When x is a vector:

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d = \theta_0 + x^T \theta$$

- Give x a “dummy dimension” to simplify notation

$$\begin{array}{ll} \text{Old} & \text{New} \\ x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} & x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad f(x) = [1 \quad x_1 \quad x_2 \quad \cdots \quad x_d] \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ x_d \end{bmatrix} = \langle x, \theta \rangle = x^T \theta \end{array}$$

Linear Regression Setup

Problem Setup

- Train set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$
- Take train features and make it a $n \times (d+1)$ matrix, and y a vector:


$$X = \begin{bmatrix} x_1^T \\ \dots \\ x_n^T \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}$$

- Then, the empirical risk is $\frac{1}{n} \|X\theta - y\|^2$

Finding The Estimated Parameters

Have our loss: $\frac{1}{n} \|X\theta - y\|^2$

- Could optimize it with SGD, etc...
- But the minimum also has a closed-form solution (vector calculus):

Hat: indicates an estimate 

$$\hat{\theta} = (X^T X)^{-1} X^T y$$


Not always
invertible...

**“Normal
Equations”**

Break & Quiz

Q 2.3: You have a dataset for regression given by $(x_1, y_1) = ([-1, 0, 1], 2)$ and $(x_2, y_2) = ([2, 3, 1], 4)$.

We have the weights $\beta_0 = 0, \beta_1 = 2, \beta_2 = 1, \beta_3 = 1$. What is the mean squared error (MSE) on the training set?

- A. 9
- B. $13/2$
- C. $25/2$
- D. 25

Break & Quiz

Q 2.3: You have a dataset for regression given by $(x_1, y_1) = ([-1, 0, 1], 2)$ and $(x_2, y_2) = ([2, 3, 1], 4)$.

We have the weights $\beta_0 = 0, \beta_1 = 2, \beta_2 = 1, \beta_3 = 1$. What is the mean squared error (MSE) on the training set?

- A. 9
- B. $13/2$
- C. **$25/2$**
- D. 25

Break & Quiz

Q 2.3: You have a dataset for regression given by $(x_1, y_1) = ([-1, 0, 1], 2)$ and $(x_2, y_2) = ([2, 3, 1], 4)$.

We have the weights $\beta_0 = 0, \beta_1 = 2, \beta_2 = 1, \beta_3 = 1$. What is the mean squared error (MSE) on the training set?

- A. 9
- B. $13/2$
- C. $25/2$
- D. 25

Compute the predicted label for each data point, then compute the squared error for each data point, then take the mean error of the two points:

$$\hat{y}_1 = -1 * \beta_1 + 0 * \beta_2 + 1 * \beta_3 = -1$$
$$\ell(\hat{y}_1, y_1) = (-1 - 2)^2 = 9$$

$$\hat{y}_2 = 2 * \beta_1 + 3 * \beta_2 + 1 * \beta_3 = 8$$
$$\ell(\hat{y}_2, y_2) = (8 - 4)^2 = 16$$

$$\text{MSE} = (16 + 9) / 2 = 25 / 2$$



K-Nearest Neighbors

K-nearest neighbors for classification

- Input: **Training data** $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$
Distance function $d(\mathbf{x}_i, \mathbf{x}_j)$; **number of neighbors** k ; **test data** \mathbf{x}^*
 1. Find the k training instances $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}$ closest to \mathbf{x}^* under $d(\mathbf{x}_i, \mathbf{x}_j)$
 2. Output y^* , the majority class of y_{i_1}, \dots, y_{i_k} . Break ties randomly.

k-Nearest Neighbors: Distances

Discrete features: Hamming distance

$$d_H(x^{(i)}, x^{(j)}) = \sum_{a=1}^d 1\{x_a^{(i)} \neq x_a^{(j)}\}$$

Continuous features:

Euclidean distance:

$$d(x^{(i)}, x^{(j)}) = \left(\sum_{a=1}^d (x_a^{(i)} - x_a^{(j)})^2 \right)^{\frac{1}{2}}$$

L1 (Manhattan) dist.:

$$d(x^{(i)}, x^{(j)}) = \sum_{a=1}^d |x_a^{(i)} - x_a^{(j)}|$$

k-Nearest Neighbors: Regression

Training/learning: given

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

Prediction: for x , find k most similar training points

Return

$$\hat{y} = \frac{1}{k} \sum_{i=1}^k y^{(i)}$$

- I.e., among the k points, output mean label.

More on **distance functions**...

- Be careful with **scale**
- Same feature but different units may change relative distance (fixing other features)
- Sometimes OK to normalize each feature dimension

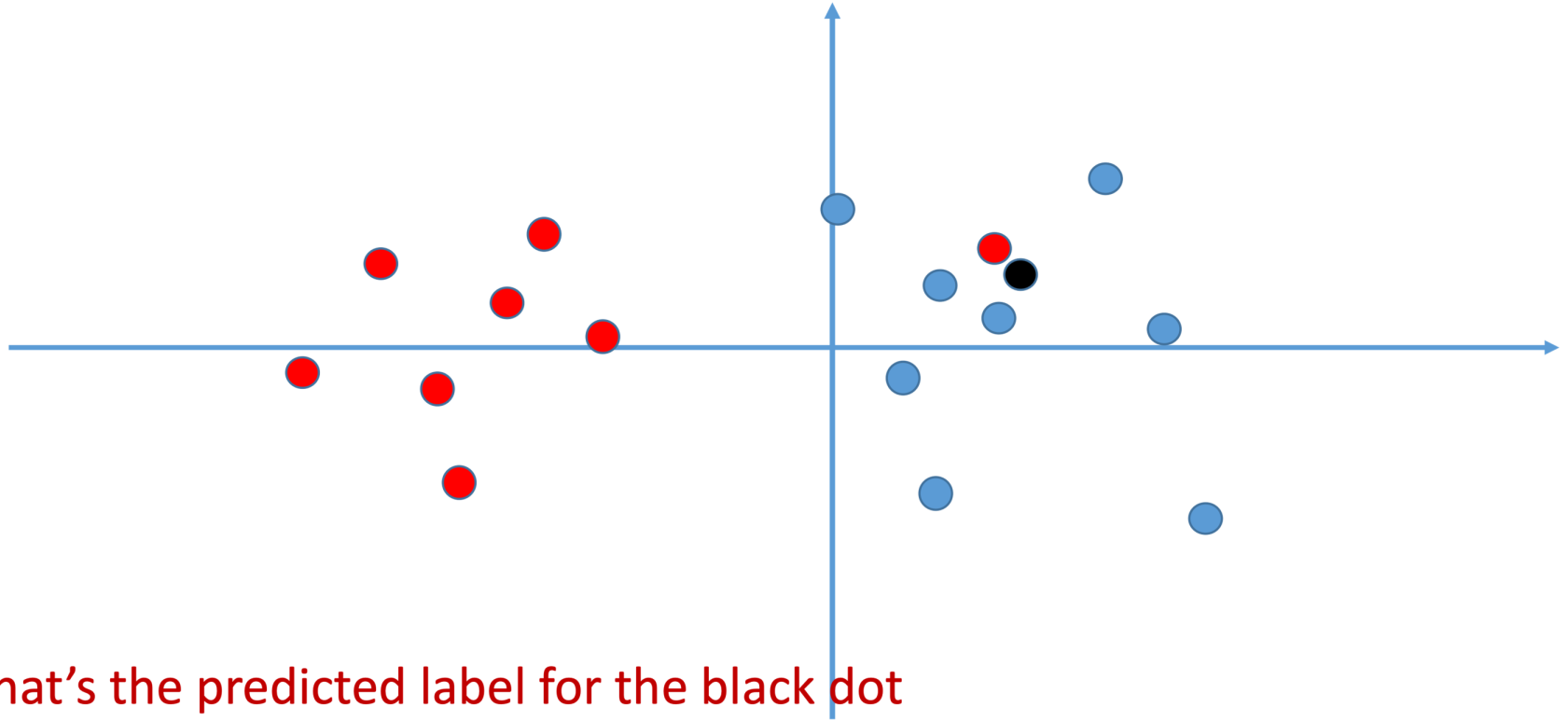
$$x'_{id} = \frac{x_{id} - \mu_d}{\sigma_d}, \forall i = 1 \dots n, \forall d$$

Training set mean for dimension d

Training set standard deviation for dimension d

- Other times not OK: e.g. dimension contains small random noise

Effect of k



What's the predicted label for the black dot
using 1 neighbor? 3 neighbors?

How to pick k, **the number of neighbors**

- Split data into training and **tuning sets**
- Classify tuning set with different k
- Pick k that produces least tuning-set error

(Shuffle whole dataset first)





Neural Networks

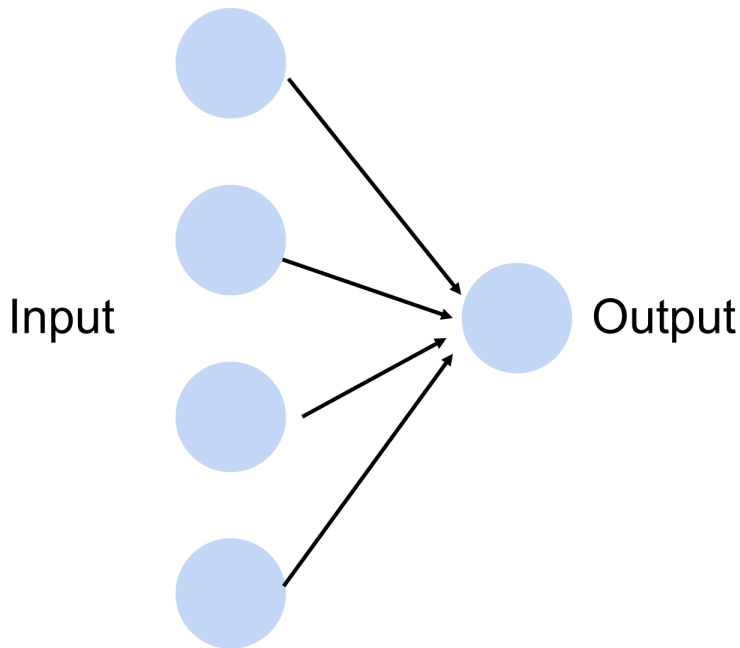
Review: Perceptron

- Given input \mathbf{x} , weight \mathbf{w} and bias b , perceptron outputs:

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$
 Activation function

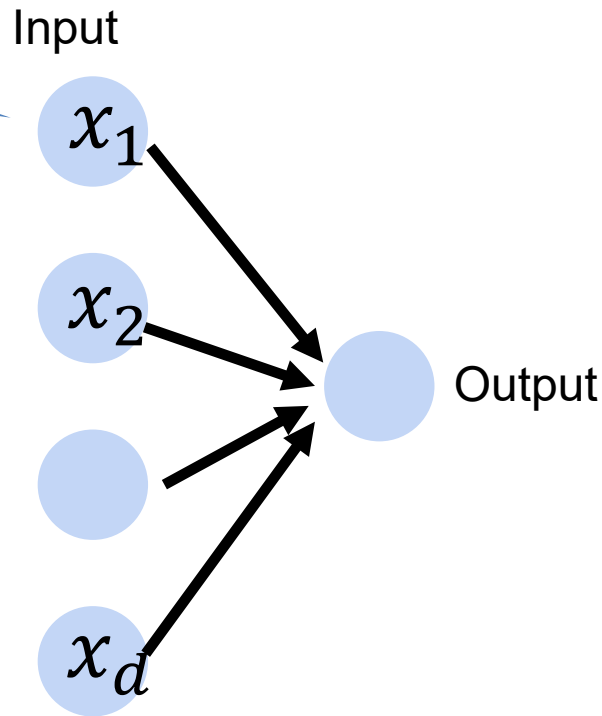
Cats vs. dogs?



The perceptron has one layer of weights

Each input node receives one scalar feature (e.g., one pixel)

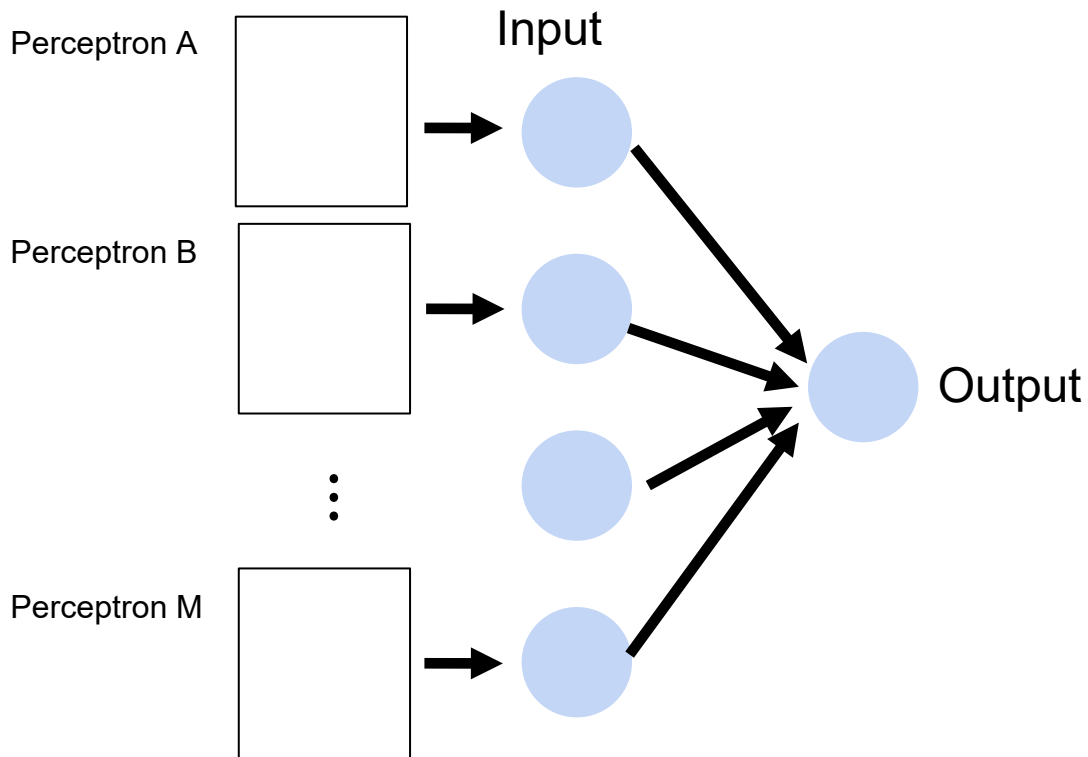
Cats vs. dogs?



Beyond one layer

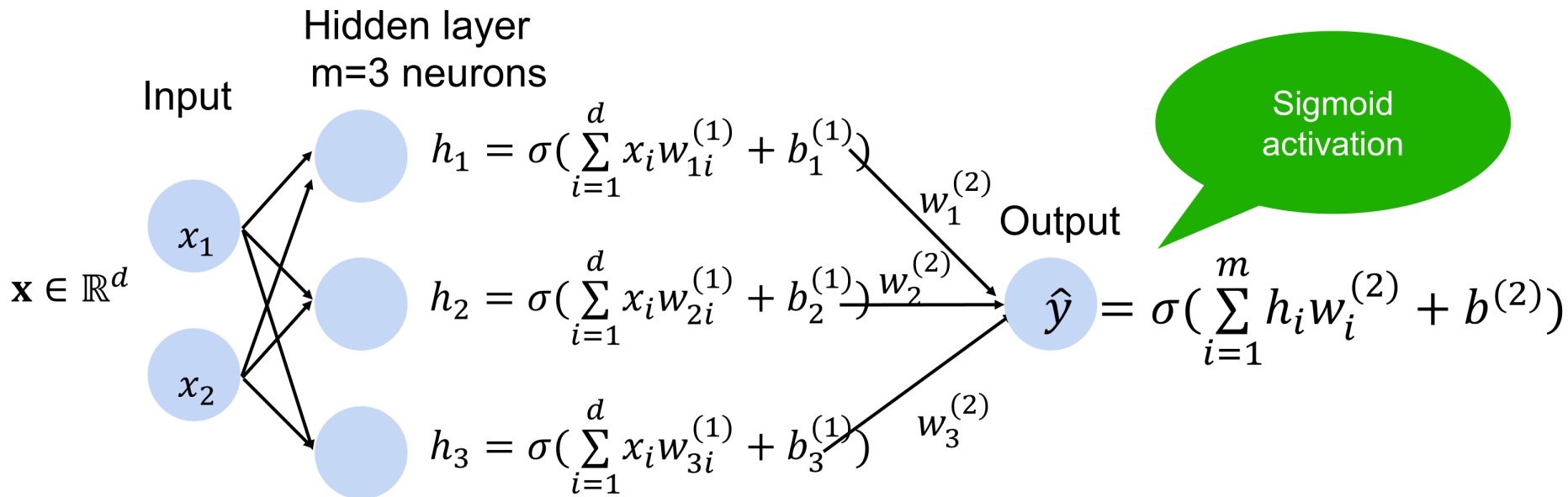
Big Idea: take our inputs from other perceptrons!

Cats vs. dogs?



Multi-layer perceptron: Example

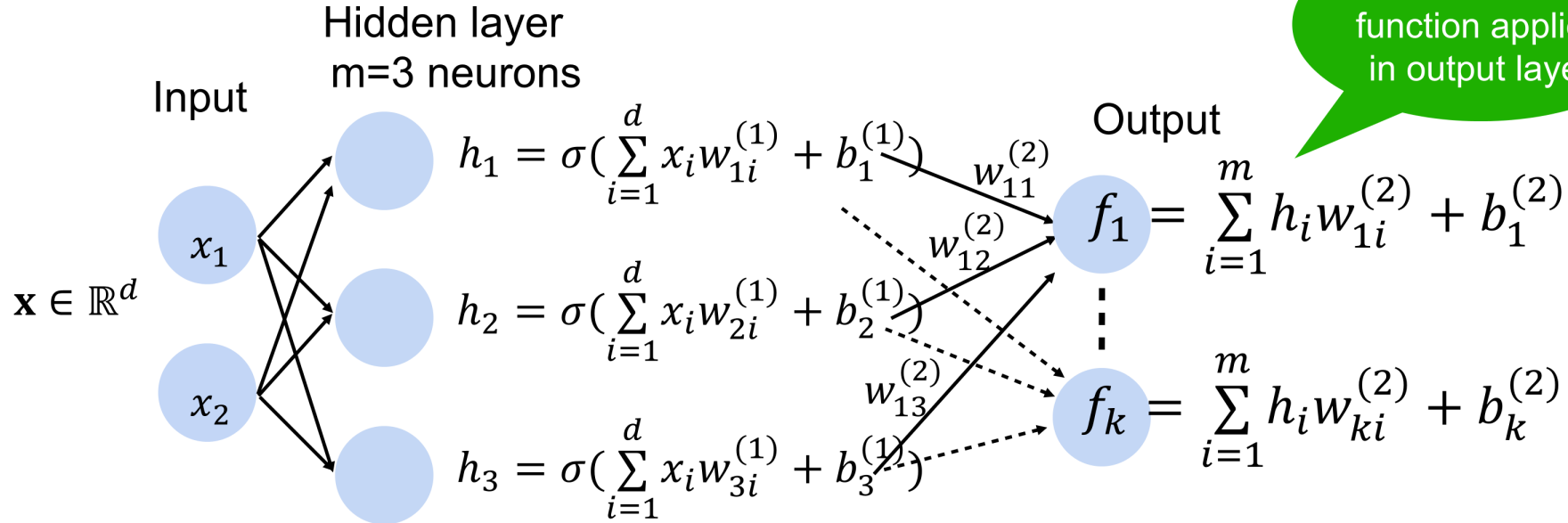
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



Neural network for K-way classification

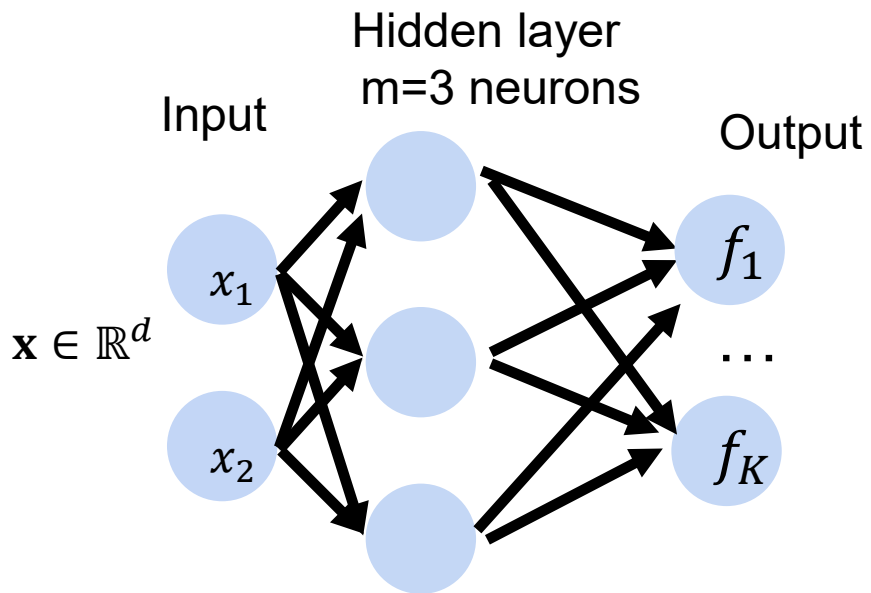
- K outputs in the final layer

Multi-class classification (e.g., ImageNet with K=1000)



Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)



$$\begin{aligned} p(y|\mathbf{x}) &= \textit{softmax}(f) \\ &= \frac{\exp(f_y(x))}{\sum_{k=1}^K \exp(f_k(x))} \end{aligned}$$

More complicated neural networks: multiple hidden layers

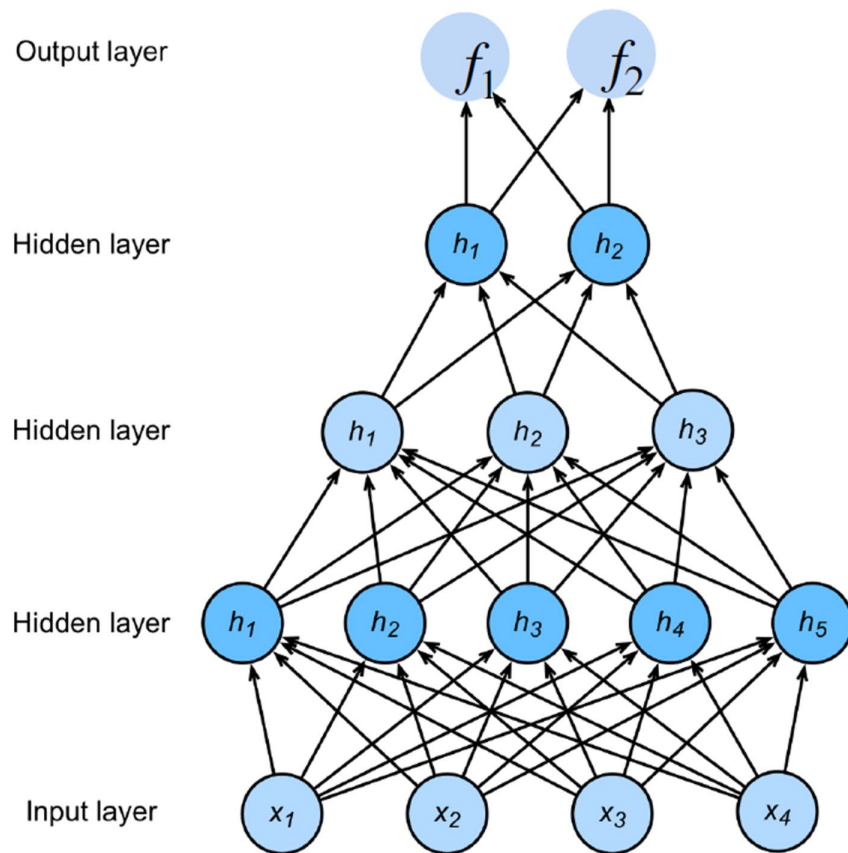
$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}^{(2)}\mathbf{h}_1 + \mathbf{b}^{(2)})$$

$$\mathbf{h}_3 = \sigma(\mathbf{W}^{(3)}\mathbf{h}_2 + \mathbf{b}^{(3)})$$

$$\mathbf{f} = \mathbf{W}^{(4)}\mathbf{h}_3 + \mathbf{b}^{(4)}$$

$$\mathbf{p} = \text{softmax}(\mathbf{f})$$



Quiz Break

Suppose you are given a 3-layer multilayer perceptron (2 hidden layers h_1 and h_2 and 1 output layer). All activation functions are sigmoids, and the output layer uses a softmax function. Suppose h_1 has 1024 units and h_2 has 512 units. Given a dataset with 2 input features and 3 unique class labels, how many learnable parameters does the perceptron have in total?

Quiz Break

Suppose you are given a 3-layer multilayer perceptron (2 hidden layers h1 and h2 and 1 output layer). All activation functions are sigmoids, and the output layer uses a softmax function. Suppose h1 has 1024 units and h2 has 512 units. Given a dataset with 2 input features and 3 unique class labels, how many learnable parameters does the perceptron have in total?

$$1024 * 2 + 1024 + 512 * 1024 + 512 + 512 * 3 + 3 = 529411$$

How to train a neural network? Binary classification

Loss function: $\frac{1}{|D|} \sum_{(\mathbf{x}, y) \in D} \ell(\mathbf{x}, y)$

Per-sample loss:

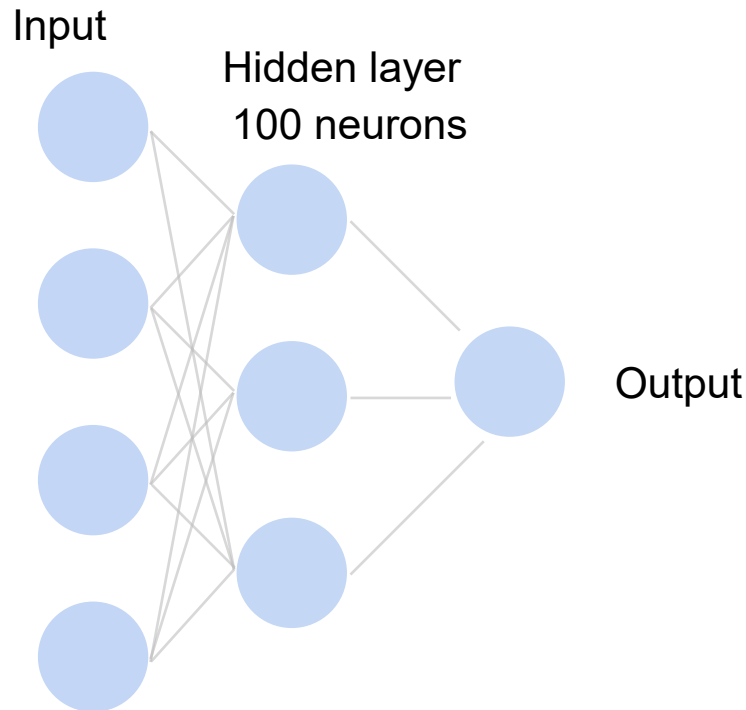
$$\ell(\mathbf{x}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$



Negative log likelihood

**Minimizing NLL is equivalent to Max
Likelihood Learning (MLE)**

Also known as **binary cross-entropy loss**

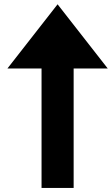


How to train a neural network? Multi-class classification

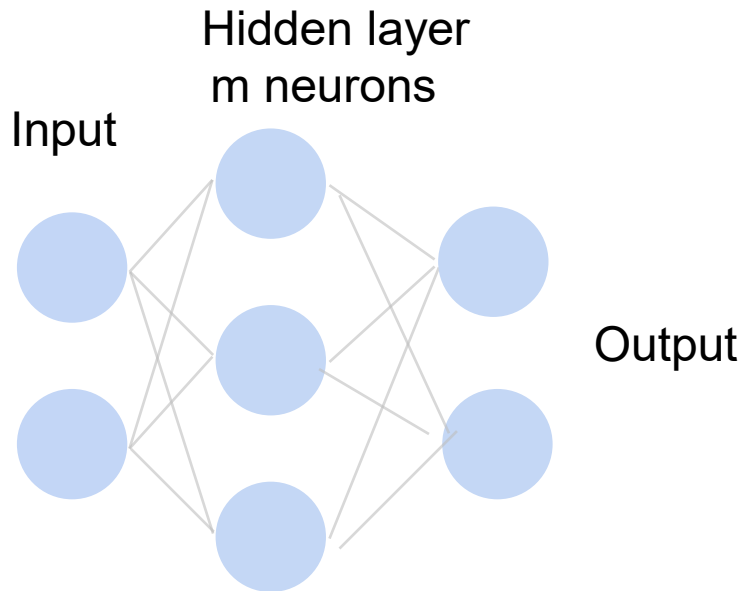
Loss function: $\frac{1}{|D|} \sum_i \ell(\mathbf{x}_i, y_i)$

Per-sample loss:

$$\ell(\mathbf{x}, y) = \sum_{j=1}^K -y_j \log p_j$$



Also known as **cross-entropy loss**
or **softmax loss**

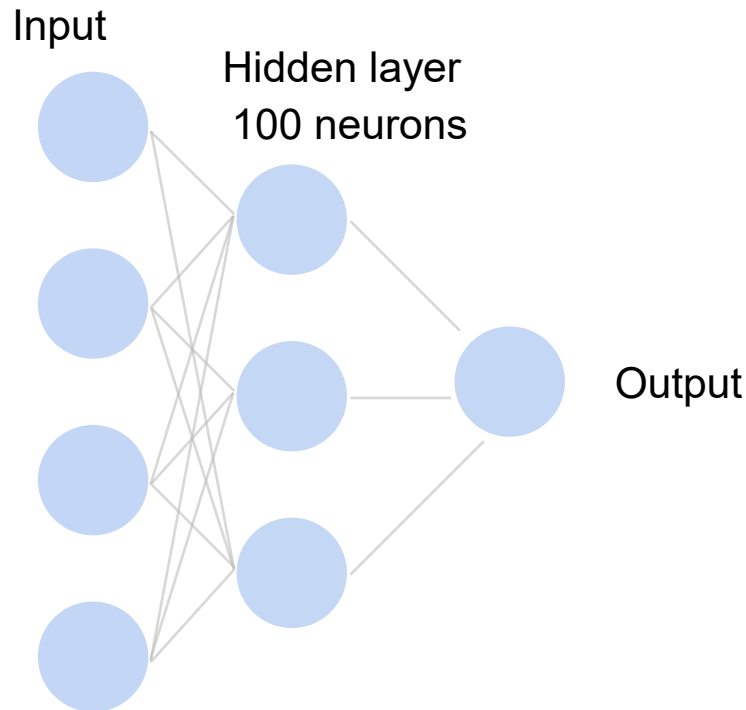


How to train a neural network? Multi-class classification

Update the weights W to minimize the loss function

$$L = \frac{1}{|D|} \sum_{(\mathbf{x}, y) \in D} \ell(\mathbf{x}, y)$$

Use gradient descent!



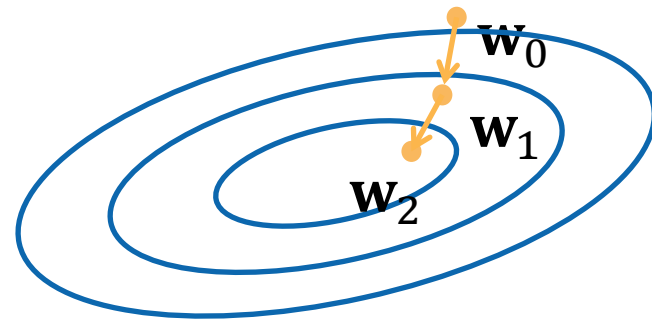
Gradient Descent

- Choose a learning rate $\eta > 0$
- Initialize the model parameters w_0
- For $t = 1, 2, \dots$
 - Update parameters:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial L}{\partial \mathbf{w}_{t-1}}$$

$$= \mathbf{w}_{t-1} - \eta \frac{1}{|D|} \sum_{(\mathbf{x}, y) \in D} \frac{\partial \ell(\mathbf{x}, y)}{\partial \mathbf{w}_{t-1}}$$

- Repeat until converges

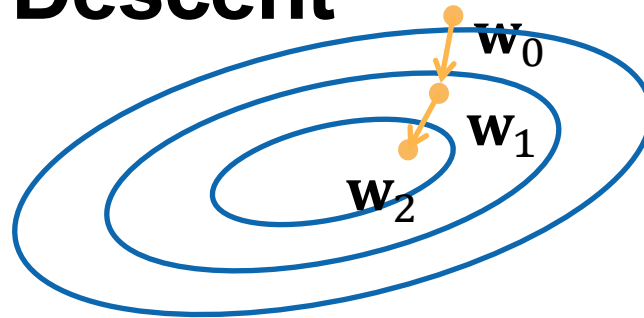


D can be very large. Expensive per iteration

The gradient w.r.t. all parameters is obtained by concatenating the partial derivatives w.r.t. each parameter

Minibatch Stochastic Gradient Descent

- Choose a learning rate $\eta > 0$
- Initialize the model parameters w_0
- For $t = 1, 2, \dots$



- **Randomly sample a subset (mini-batch) $B \subset D$**
- Update parameters:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{1}{|B|} \sum_{(\mathbf{x}, y) \in B} \frac{\partial \ell(\mathbf{x}, y)}{\partial \mathbf{w}_{t-1}}$$

- Repeat until converges

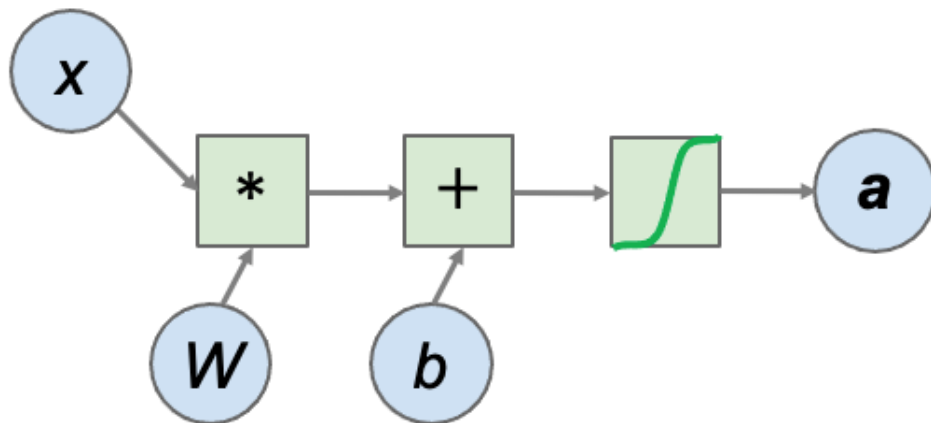


Computational Graphs

Neural networks as variables + operations

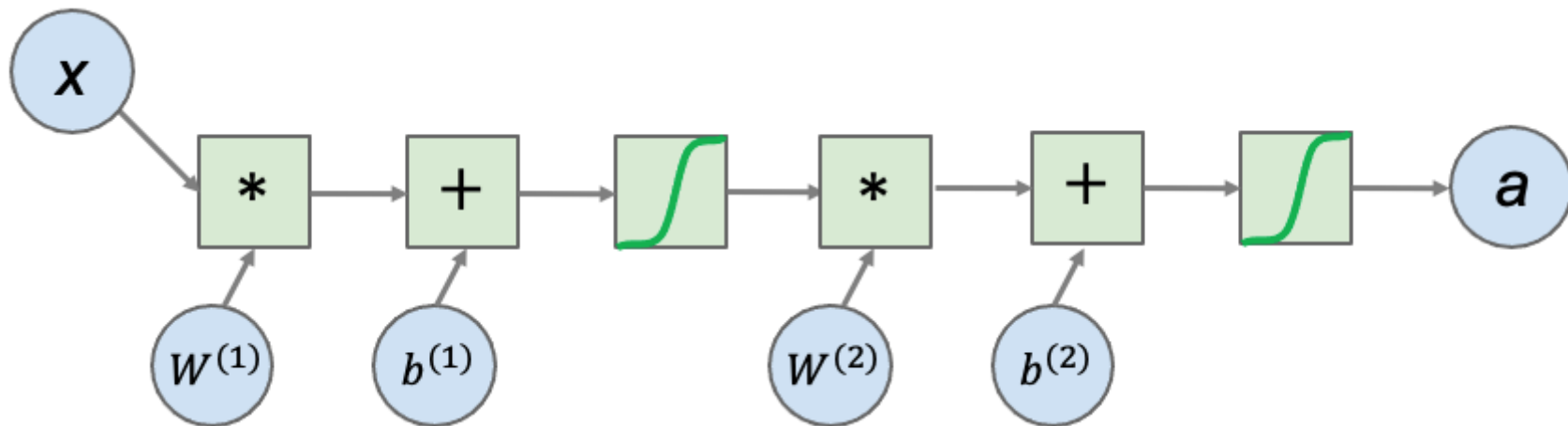
$$\mathbf{a} = \textit{sigmoid}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- Can describe with a **computational graph**
- Decompose functions into atomic operations
- Separate data (**variables**) and computing (**operations**)



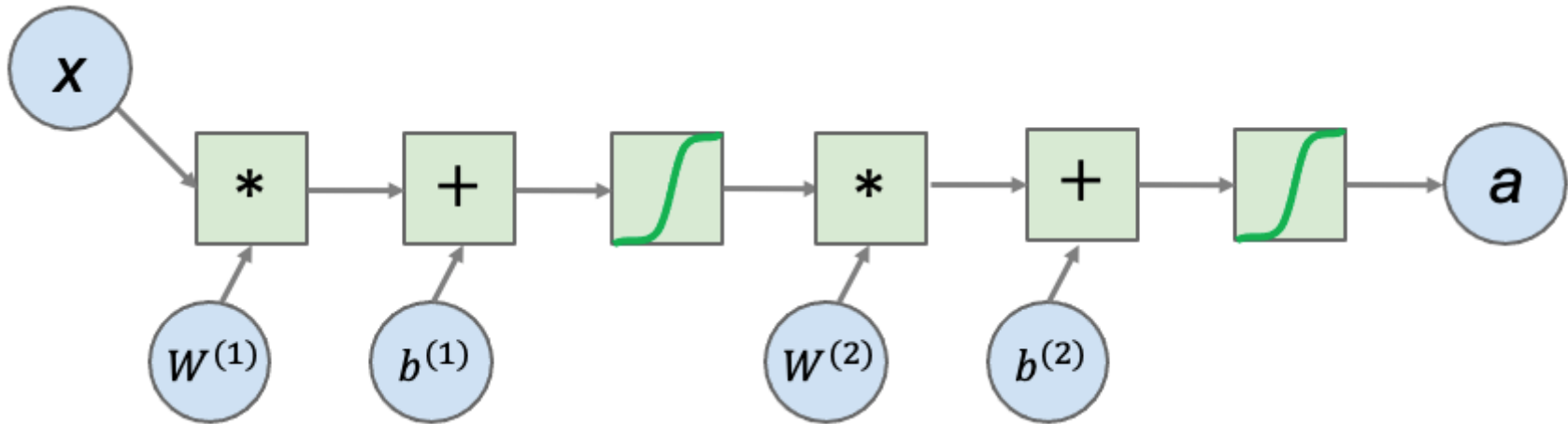
Neural networks as a computational graph

- A two-layer neural network



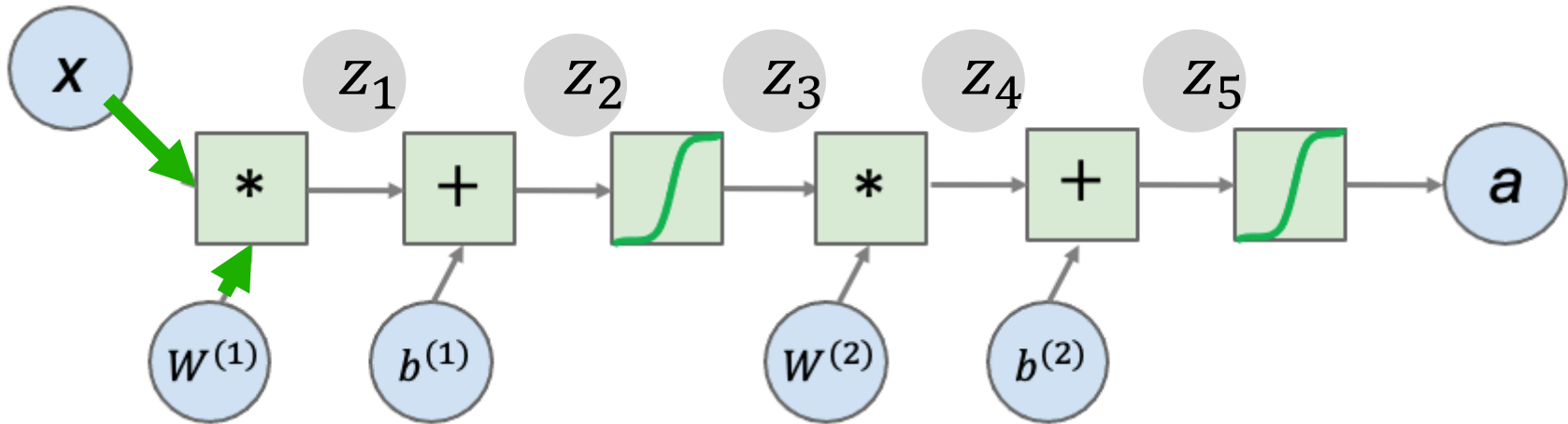
Neural networks as a computational graph

- A two-layer neural network
- Forward propagation vs. backward propagation



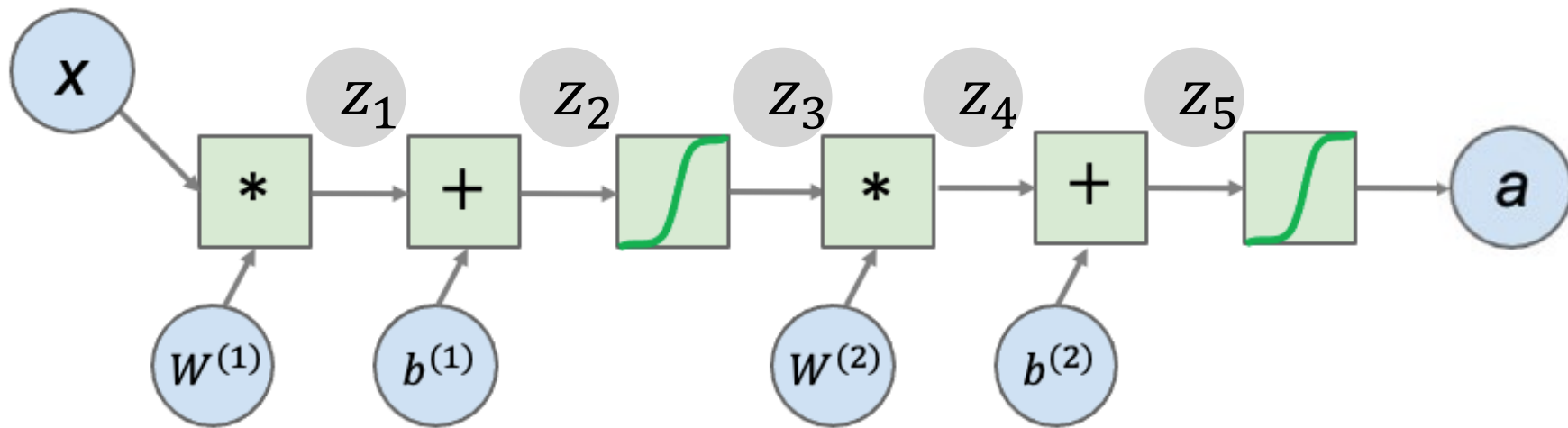
Neural networks: forward propagation

- A two-layer neural network
- Intermediate variables Z



Neural networks: backward propagation

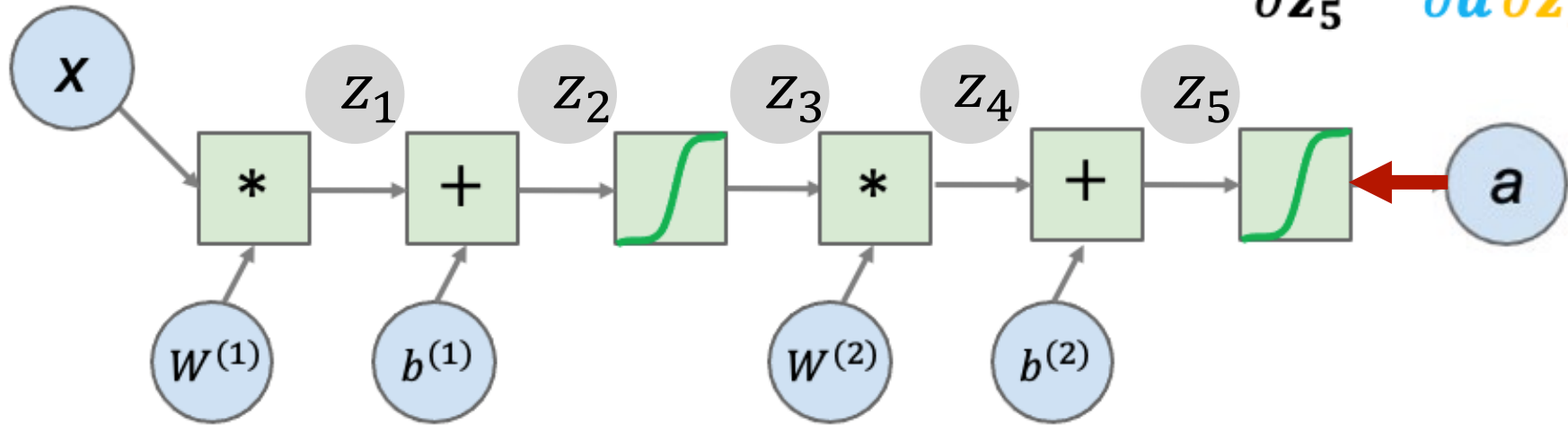
- A two-layer neural network
- Assuming forward propagation is done
- Minimize a **loss function** L



Neural networks: backward propagation

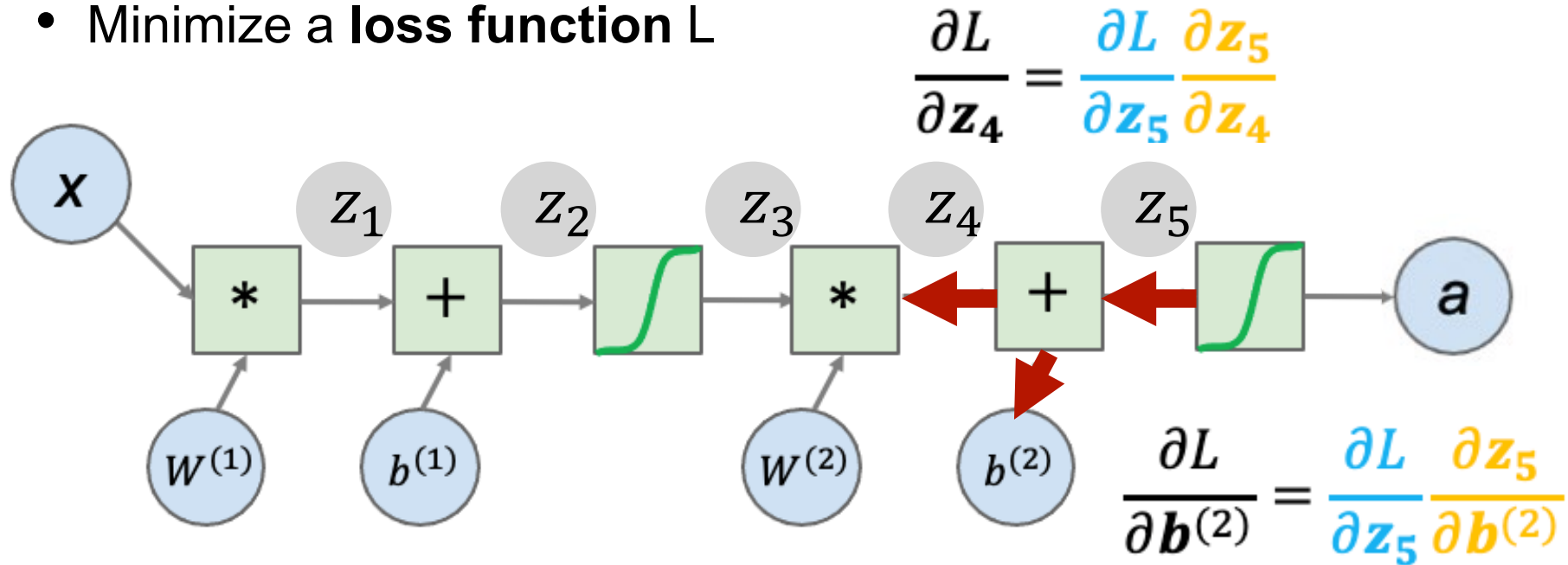
- A two-layer neural network
- Assuming forward propagation is done
- Minimize a **loss function** L

$$\frac{\partial L}{\partial \mathbf{z}_5} = \frac{\partial L}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{z}_5}$$



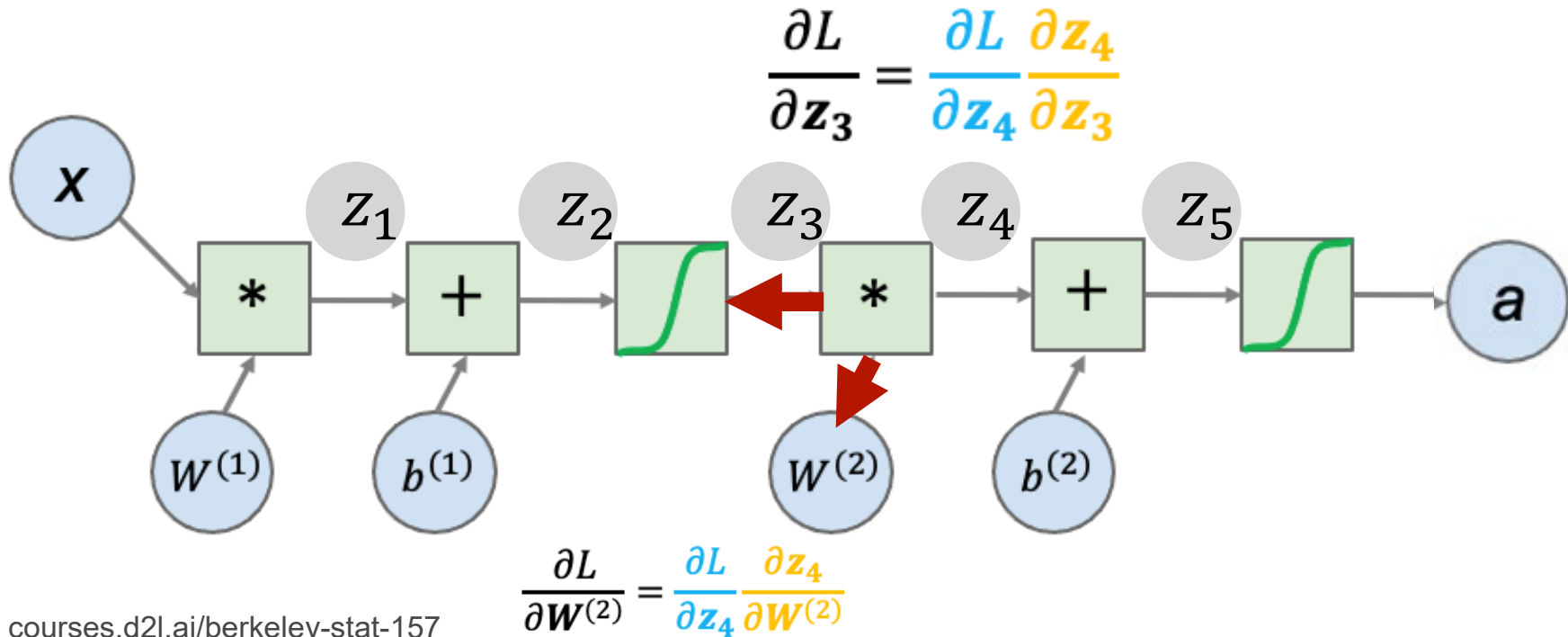
Neural networks: backward propagation

- A two-layer neural network
- Assuming forward propagation is done
- Minimize a **loss function** L



Neural networks: backward propagation

- A two-layer neural network
- Assuming forward propagation is done





Numerical Stability

Gradients for Neural Networks

Compute the gradient of the loss ℓ w.r.t. \mathbf{W}_t

$$\frac{\partial \ell}{\partial \mathbf{W}^t} = \frac{\partial \ell}{\partial \mathbf{h}^d} \frac{\partial \mathbf{h}^d}{\partial \mathbf{h}^{d-1}} \cdots \frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \frac{\partial \mathbf{h}^t}{\partial \mathbf{W}^t}$$

Multiplication of *many* matrices



Wikipedia

Two Issues for Deep Neural Networks

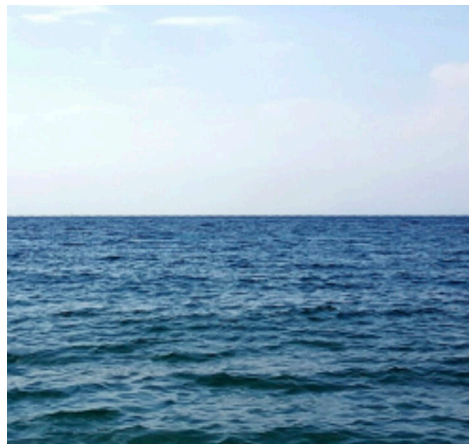
$$\prod_{i=t}^{d-1} \frac{\partial \mathbf{h}^{i+1}}{\partial \mathbf{h}^i}$$

Gradient Exploding



$$1.5^{100} \approx 4 \times 10^{17}$$

Gradient Vanishing



$$0.8^{100} \approx 2 \times 10^{-10}$$

Issues with Gradient Exploding

Value out of range: infinity value (NaN)

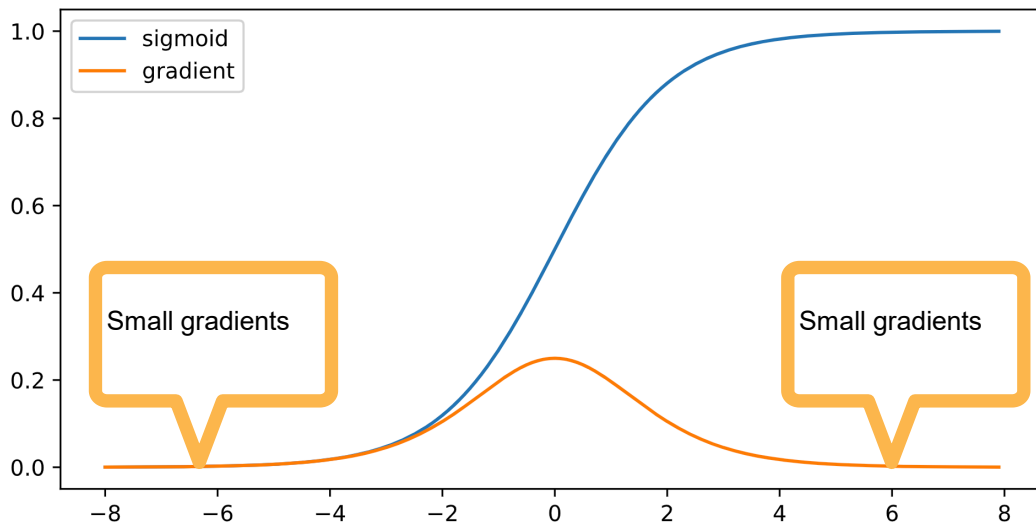
Sensitive to learning rate (LR)

- Not small enough LR \rightarrow larger gradients
- Too small LR \rightarrow No progress
- May need to change LR dramatically during training

Gradient Vanishing

Use sigmoid as the activation function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \sigma'(x) = \sigma(x)(1 - \sigma(x))$$



Issues with Gradient Vanishing

Gradients with value 0

No progress in training

- No matter how to choose learning rate

Severe with bottom layers (those near the input)

- Only top layers (near output) are well trained
- No benefit to make networks deeper



Thanks!