CS 540 Introduction to Artificial Intelligence
Midterm Review

University of Wisconsin-Madison

Spring 2025

Announcements

- **Homeworks**:
  - HW6 online, deadline on Monday **March. 17th at 11:59 PM**
- Class roadmap:

Midterm Review
Machine Learning: Deep Learning

$\left.\begin{matrix} \\ \\ \end{matrix}\right\}$ Machine Learning

- Midterm Evaluation

Midterm Information

- **Time:** March 13th 7:30-9 PM
- **Location:**
  - Section 001 : Ingraham Hall B10
  - Section 002 : Psychology 105
  - **Section 003: split in two locations according to the last name:**
    - **Chamberlin Hall 2103 (last name starting with A-L)**
    - **Sterling Hall 1310 (last name starting with M-Z)**
- McBurney students and students requesting alternate: reach out to your instructor if you have not received any email!
- Format: multiple choice
- WISC ID
- Cheat sheet: single piece of paper, front and back
- Calculator: fine if it doesn’t have an Internet connection
- Detailed topic list + practice on Piazza and Canvas

Bayesian **Inference**

- Fancy name for what we just did. Terminology:

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

- *H* is the hypothesis
- *E* is the evidence

Bayesian **Inference**

- Terminology:

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

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

Bayesian Inference

- Terminology:

Likelihood

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

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

Bayesian Inference

- Terminology:

$$\underbrace{P(H|E)}_{\uparrow\ \textbf{Posterior}} = \frac{P(E|H)P(H)}{P(E)}$$

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

Naïve Bayes

- Conditional Probability & Bayes:

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

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

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

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

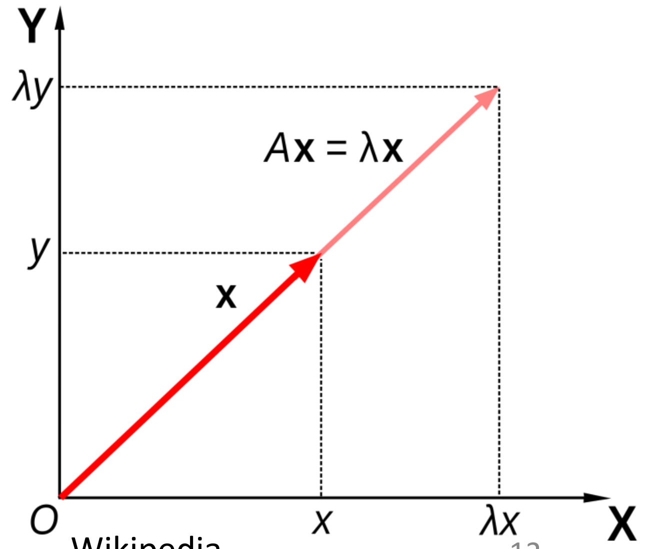
P(S) = 50 % spam email
P(NS) = 50% not spam email
P(DS|NS) = 5% false positive, detected as spam but not spam
P(DS|S) = 99% detected as spam and it is spam

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

Eigenvalues & Eigenvectors

- For a square matrix *A*, solutions to $Av = \lambda v$
  - *v* (nonzero) is a vector: **eigenvector**
  - λ is a scalar: **eigenvalue**

  - Intuition: A is a linear transformation;
  - Can stretch/rotate vectors;
  - E-vectors: only stretched (by e-vals)



Wikipedia

Break & Quiz

Q 2.2: What are the eigenvalues of $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

A. -1, 2, 4
B. 0.5, 0.2, 1.0
C. 0, 2, 5
D. 2, 5, 1

Break & Quiz

Q 2.2: What are the eigenvalues of $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

A. -1, 2, 4
B. 0.5, 0.2, 1.0
C. 0, 2, 5
D. 2, 5, 1

Solution #1: You may recall from a linear algebra course that the eigenvalues of a diagonal matrix (in which only diagonal entries are non-zero) are just the entries along the diagonal. Hence D is the correct answer.

Break & Quiz

Q 2.2: What are the eigenvalues of $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Solution #2: Use the definition of eigenvectors and values: $Av = \lambda v$

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} = \begin{bmatrix} 2v_1 + 0v_2 + 0v_3 \\ 0v_1 + 5v_2 + 0v_3 \\ 0v_1 + 0v_2 + 1v_3 \end{bmatrix} = \begin{bmatrix} 2v_1 \\ 5v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \lambda v_1 \\ \lambda v_2 \\ \lambda v_3 \end{bmatrix}$$

A. -1, 2, 4
B. 0.5, 0.2, 1.0
C. 0, 2, 5
D. 2, 5, 1

Since A is a 3x3 matrix, A has 3 eigenvalues and so there are 3 combinations of values for λ and v that will satisfy the above equation. The simple form of the equations suggests starting by checking each of the standard basis vectors* as v and then solving for λ. Doing so gives D as the correct answer.

*Each standard basis vector $e_i \in \mathbb{R}^n$ is the vector in which all components are zero except component i is 1.

Dimensionality Reduction

Reduce dimensions

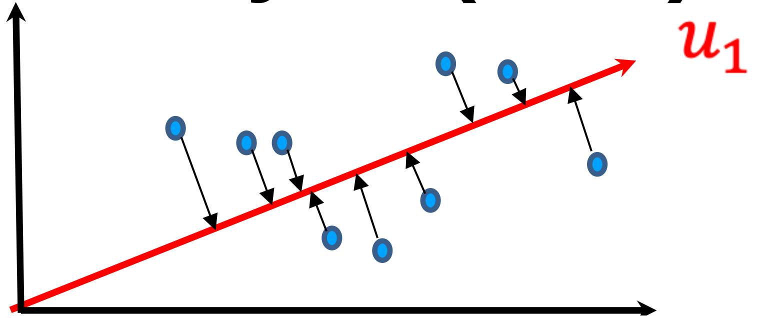
- Why?
  - Lots of features redundant
  - Storage & computation costs

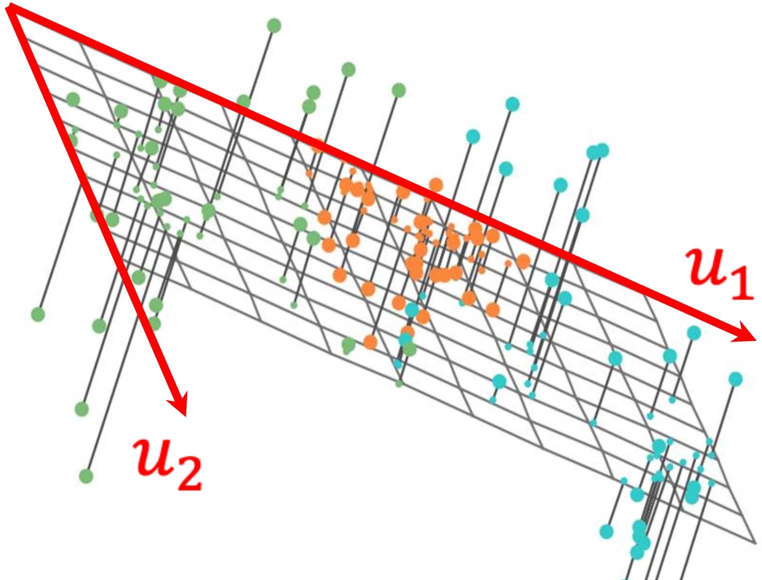
CreativeBlog

- Goal: take $x \in \mathbb{R}^d \rightarrow x \in \mathbb{R}^r$ for $r << d$
  - But, minimize information loss

Principal Components Analysis (PCA)

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

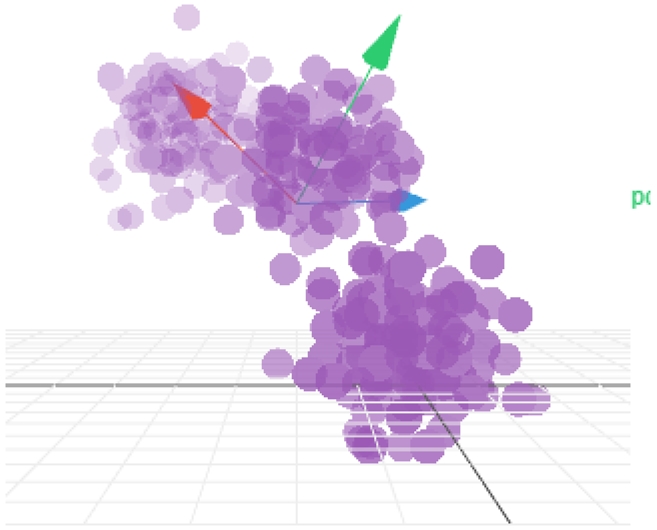




PCA Procedure

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

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



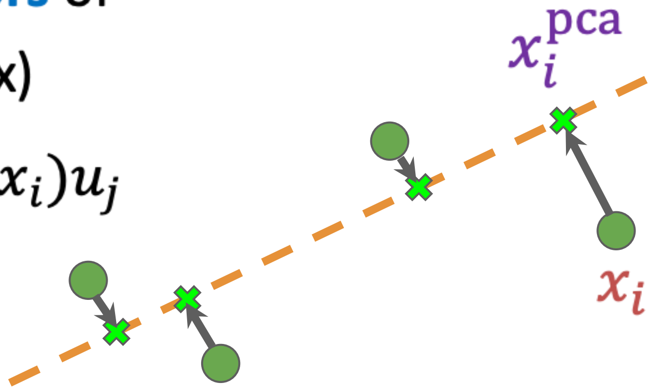
Victor Powell

PCA Procedure

Output:

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

- Orthogonal
- Can show: they are top-m **eigenvectors** of
  $S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top$ (covariance matrix)
- Each x_i projected to $x_i^{\mathrm{pca}} = \sum_{j=1}^{m}(u_j^\top x_i)u_j$



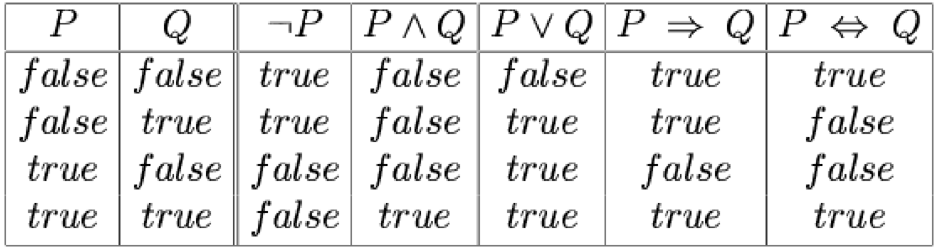
Basic Logic

Arguments, premises, conclusions

- Argument: a set of sentences (premises) + a sentence (a conclusion)
- **Validity:** argument is valid iff it’s necessary that if all premises are true, the conclusion is true
- **Soundness:** argument is sound iff valid & premises true
- **Entailment:** when valid arg., premises entail conclusion

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!)

Break & Quiz

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

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

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

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

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

Plug interpretation into each sentence.

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.

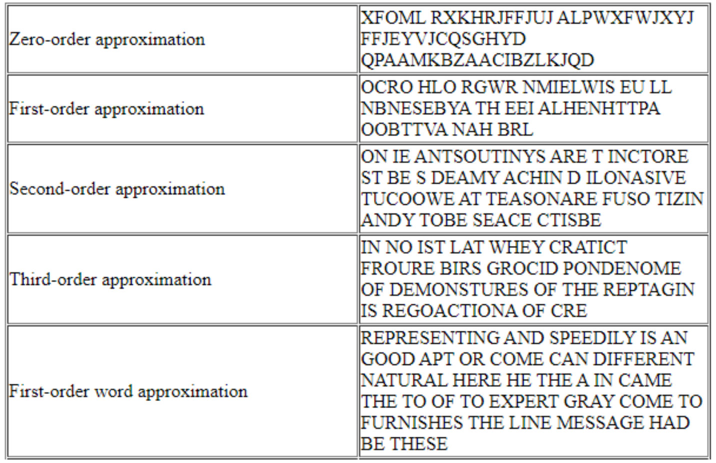
Language Models

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

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

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:

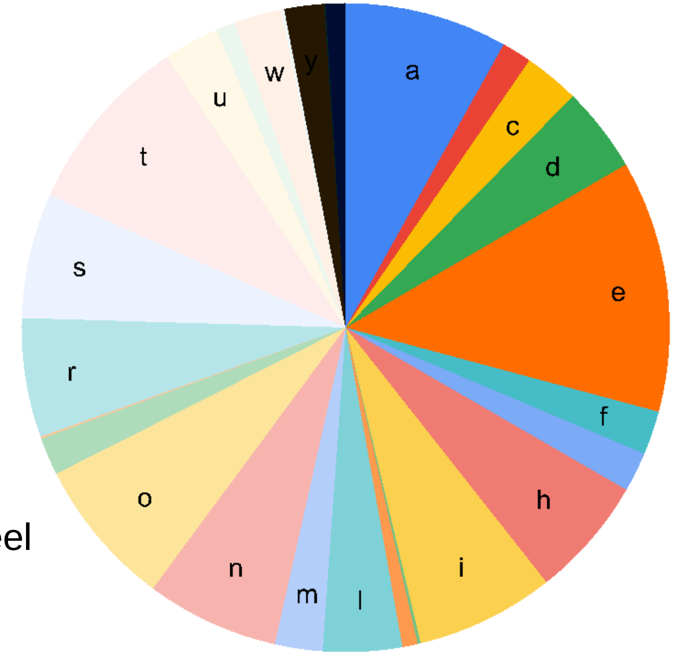
$$P(w_i|w_{i-1}w_{i-2}\dots w_1) = P(w_i|w_{i-1}w_{i-2}\dots w_{i-k})$$

- 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)

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

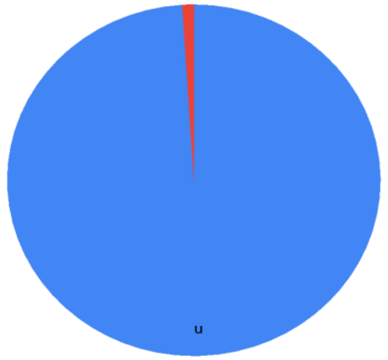


The English letter frequency wheel

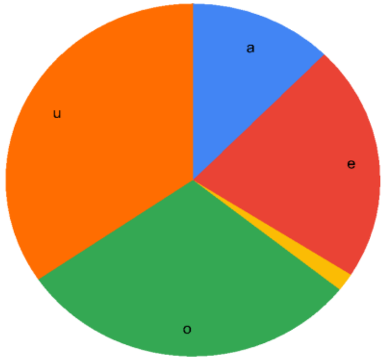
k=1: Bigram Model

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

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



$p(.|q)$: 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:

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

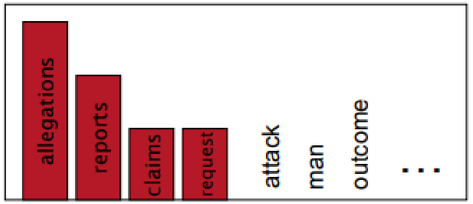
n-gram Training

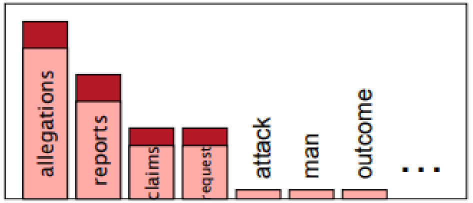
Issues:

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

- **1**. Multiply tiny numbers?
  - **Solution**: use logs; add instead of multiply
- **2.** n-grams with zero probability?
  - **Solution**: smoothing

$P(w|\text{denied the})$





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

Dan Klein

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(dog) using a unigram model without Laplace smoothing?

(ii) What is the P(my dog) using a unigram model without Laplace smoothing?

(iii) What is the P(my dog) using a bigram model without Laplace smoothing?

(iv) What is the P(dog | 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(dog) using a unigram model without Laplace smoothing?
(ii) What is the P(my dog) using a unigram model without Laplace smoothing?
(iii) What is the P(my dog) using a bigram model without Laplace smoothing?
(iv) What is the P(dog | my) for a bigram model without Laplace smoothing?

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

Representing Words

Remember value of random variables **(RVs)**

Easier to work with than objects like ‘dog’

Traditional representation: **one-hot vectors**

$$\text{dog} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- Dimension: # of words in vocabulary
- Relationships between words?

Smarter Representations

Distributional semantics: account for relationships

- Reps should be close/similar to other words that appear in a similar context

Dense vectors:

$$\text{dog} = \begin{bmatrix}0.13 & 0.87 & -0.23 & 0.46 & 0.87 & -0.31\end{bmatrix}^T$$

$$\text{cat} = \begin{bmatrix}0.07 & 1.03 & -0.43 & -0.21 & 1.11 & -0.34\end{bmatrix}^T$$

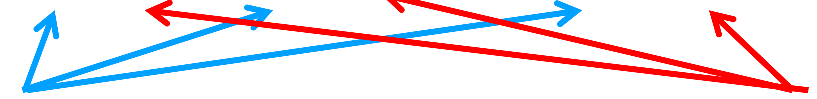
AKA **word embeddings**

Supervised/Unsupervised Learning

Supervised learning:

Make predictions, classify data, perform regression

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



Features / Covariates / Input

Labels / Outputs

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

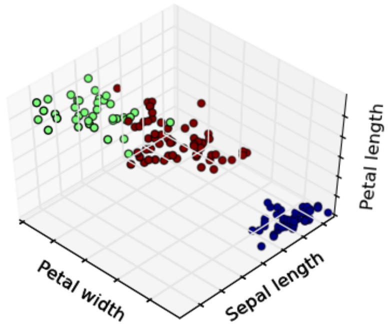
Supervised/Unsupervised Learning

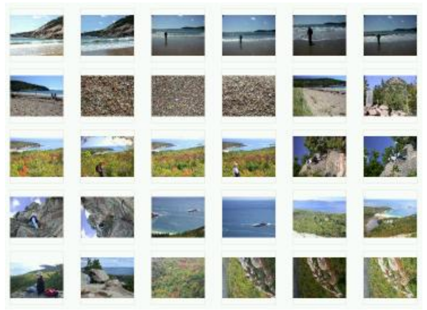
Unsupervised learning:

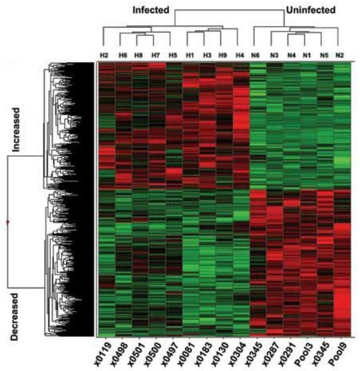
No labels; generally, won’t be making predictions

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

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



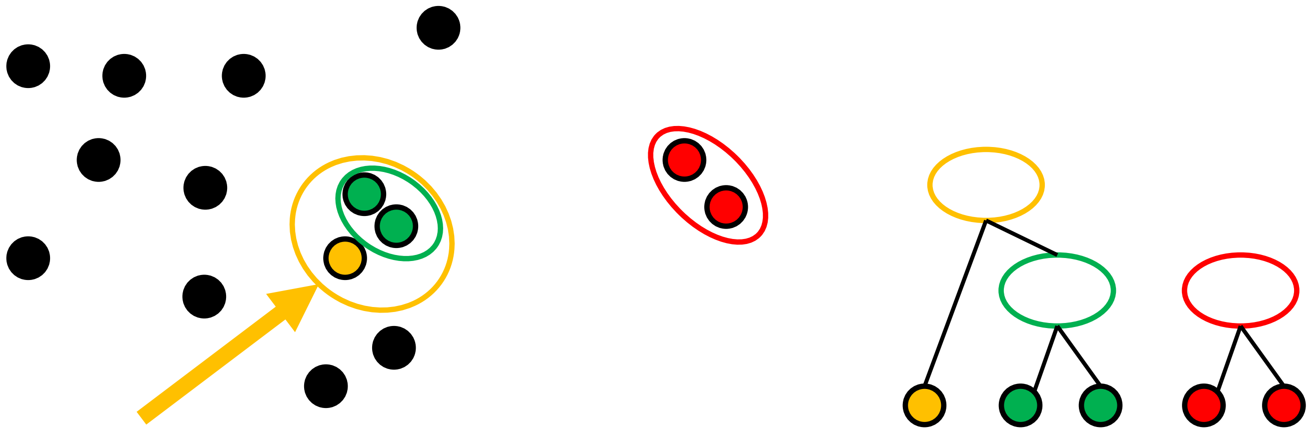




Mulvey and Gingold

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

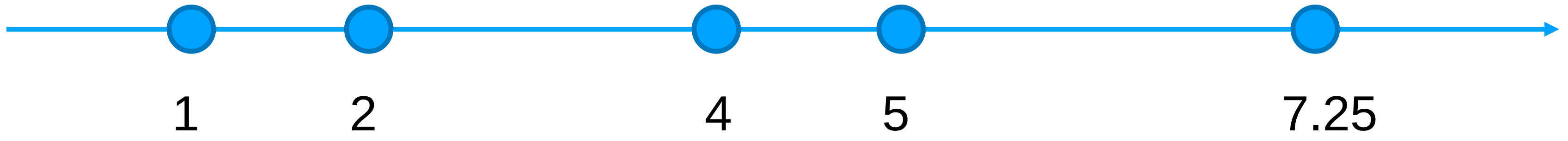
$$d(A, B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

Single-linkage Example

We’ll merge using single-linkage

1-dimensional vectors.

Initial: all points are clusters

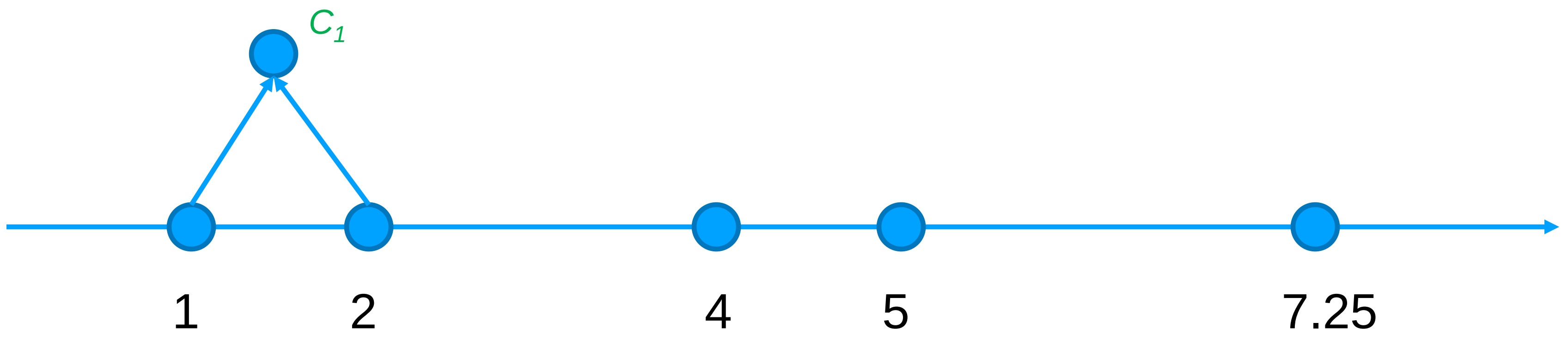


Single-linkage Example

We'll merge using single-linkage

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

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

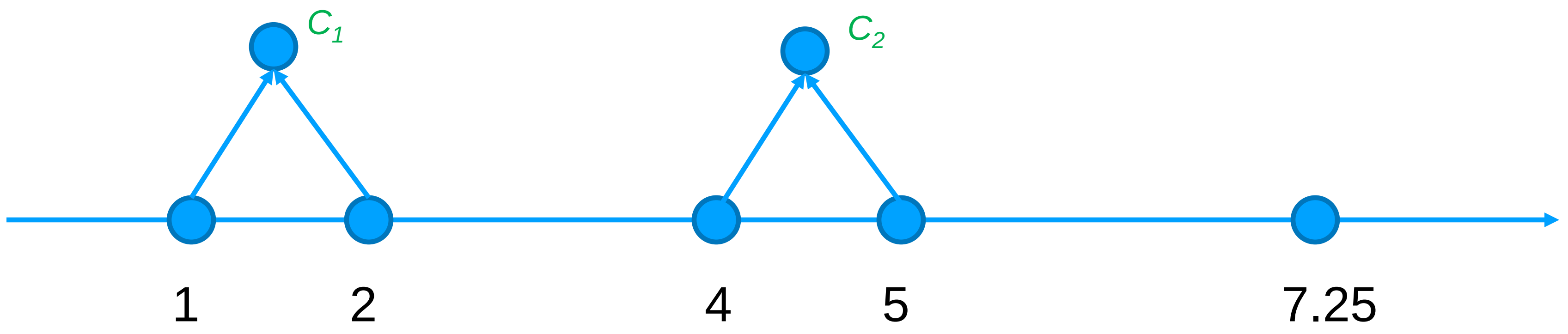


Single-linkage Example

Continue…

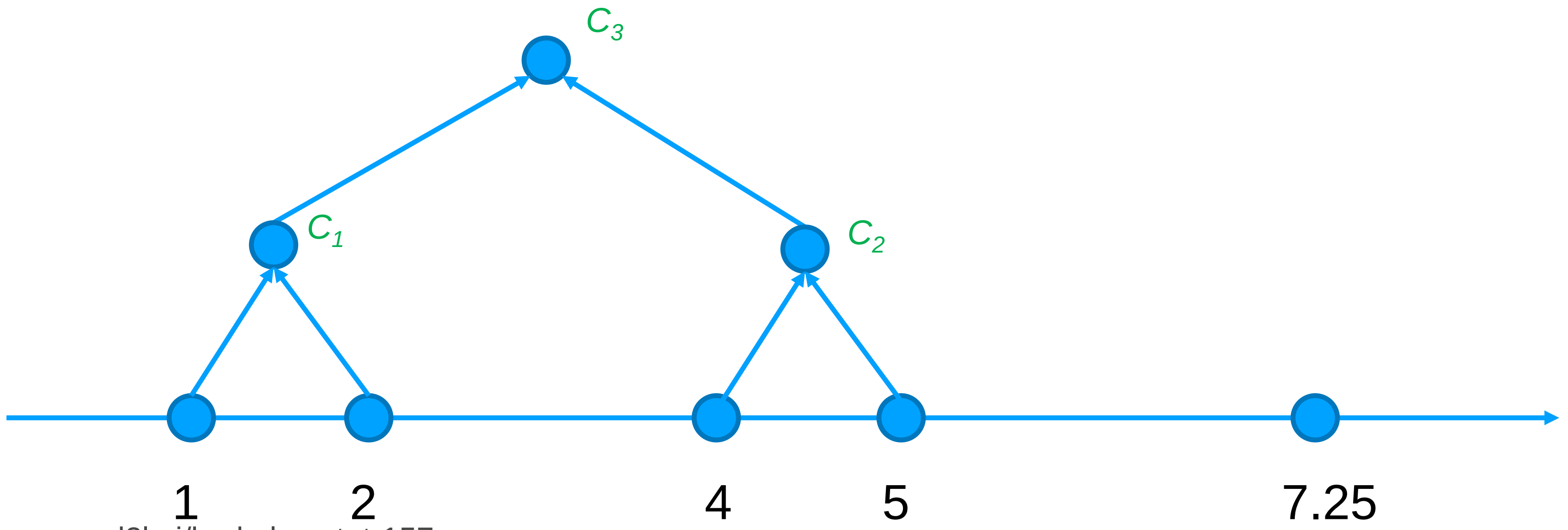
$$d(C_1, C_2) = d(2, 4) = 2$$

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

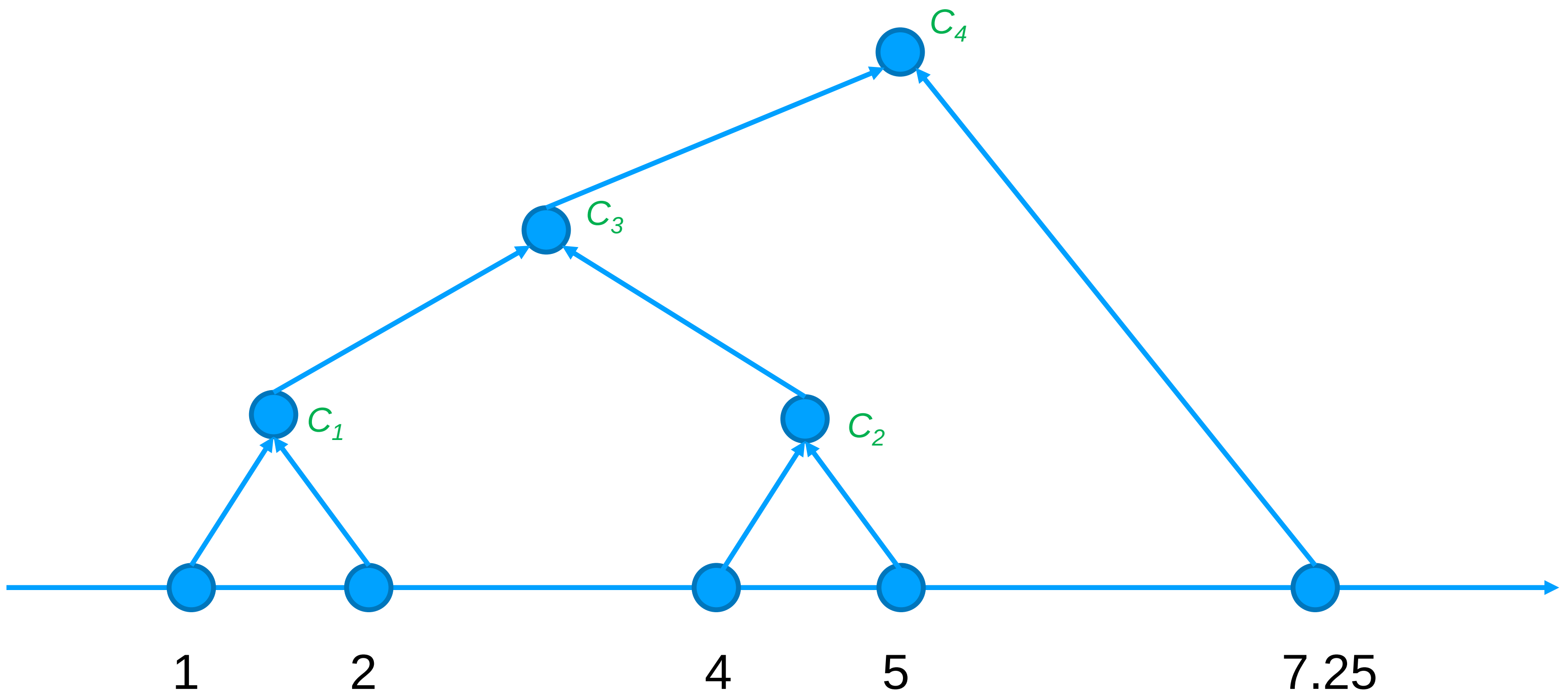


Single-linkage Example

Continue…



Single-linkage Example

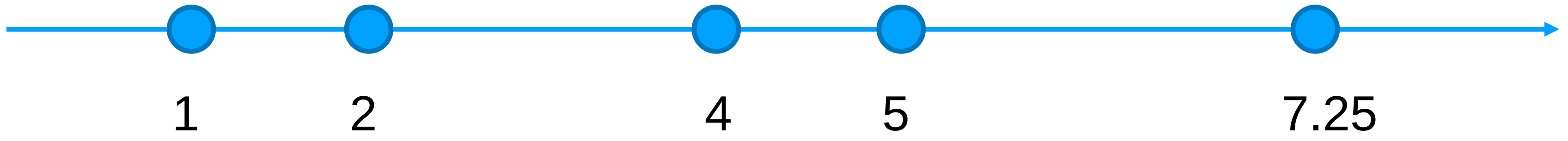


Complete-linkage Example

We’ll merge using complete-linkage

1-dimensional vectors.

Initial: all points are clusters

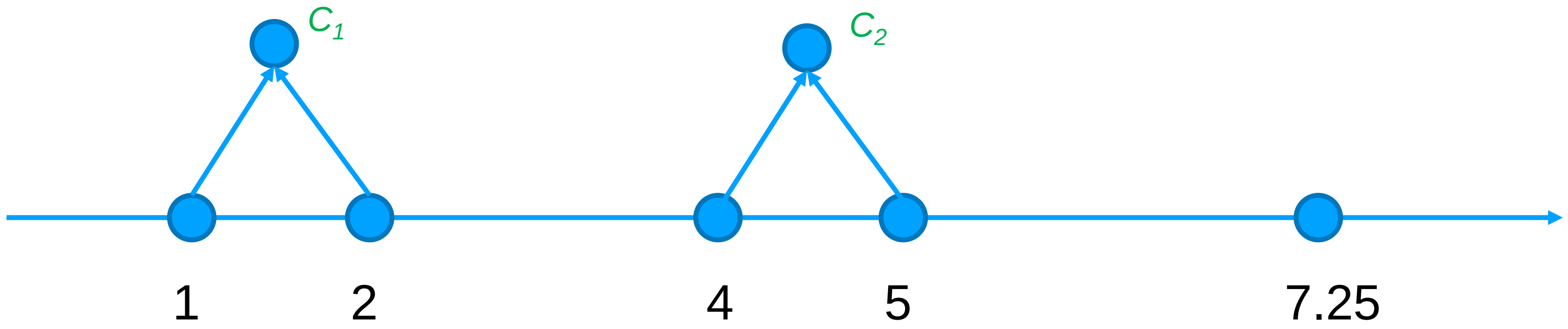


Complete-linkage Example

Beginning is the same…

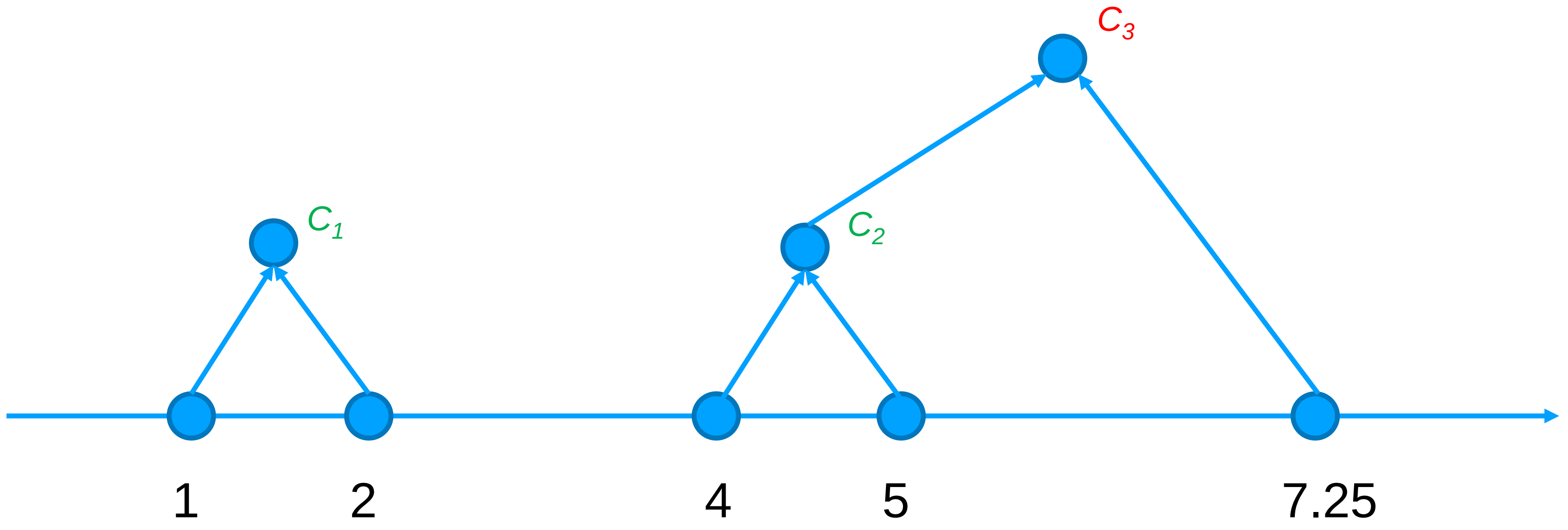
$$d(C_1, C_2) = d(1, 5) = 4$$

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

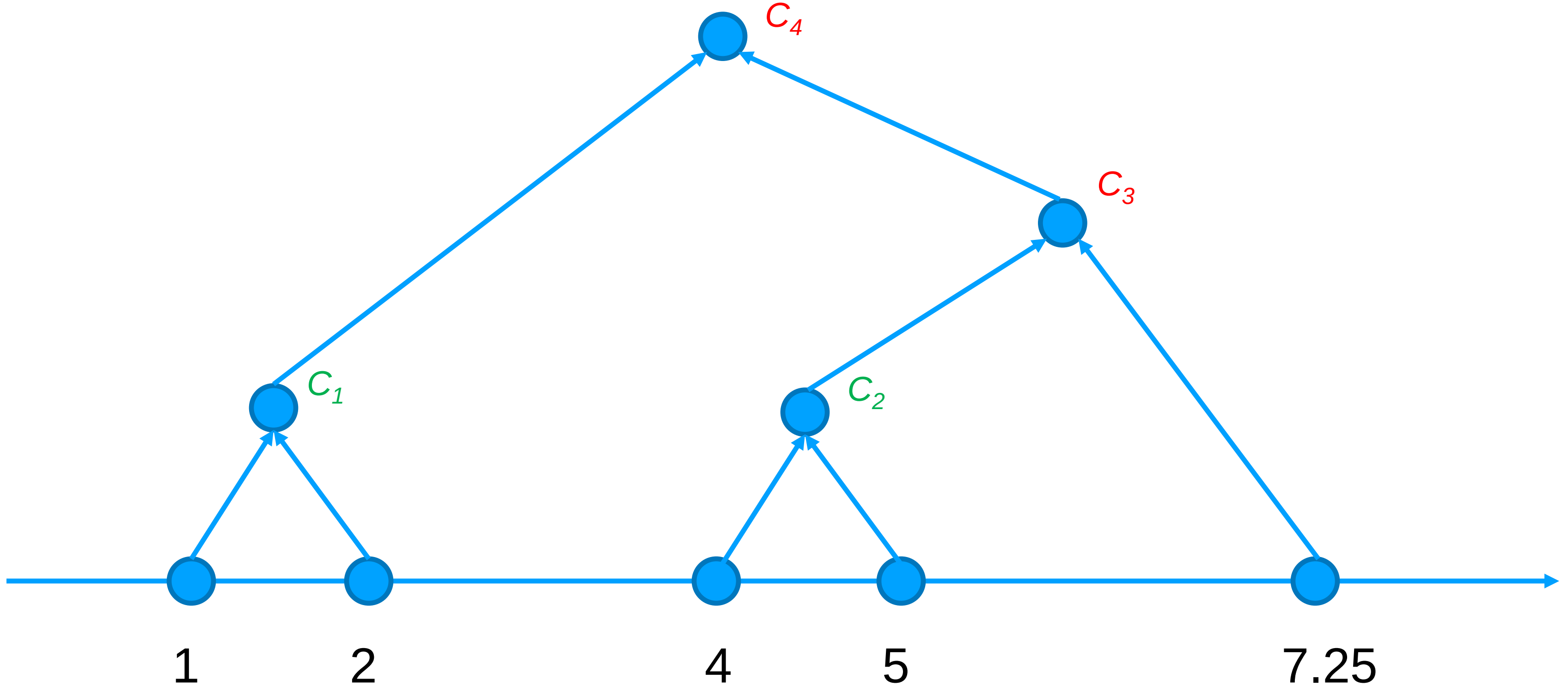


Complete-linkage Example

Now we diverge:



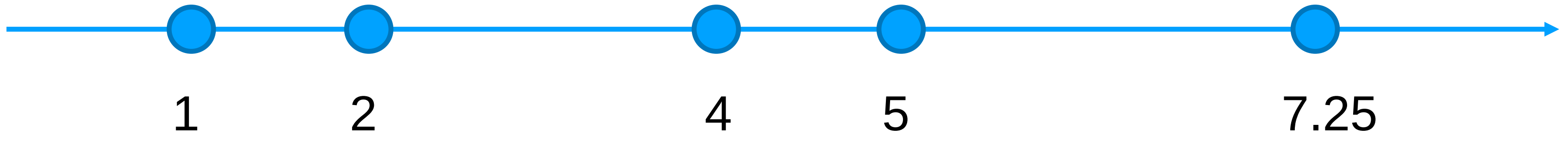
Complete-linkage Example



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?

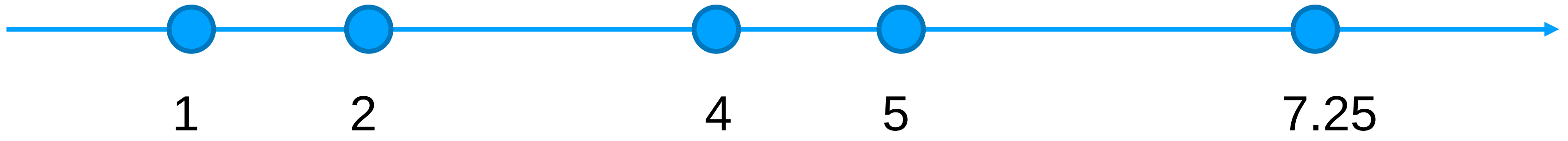
A. {1}, {2,4,5,7.25}

B. {1,2}, {4, 5, 7.25}

C. {1,2,4}, {5, 7.25}

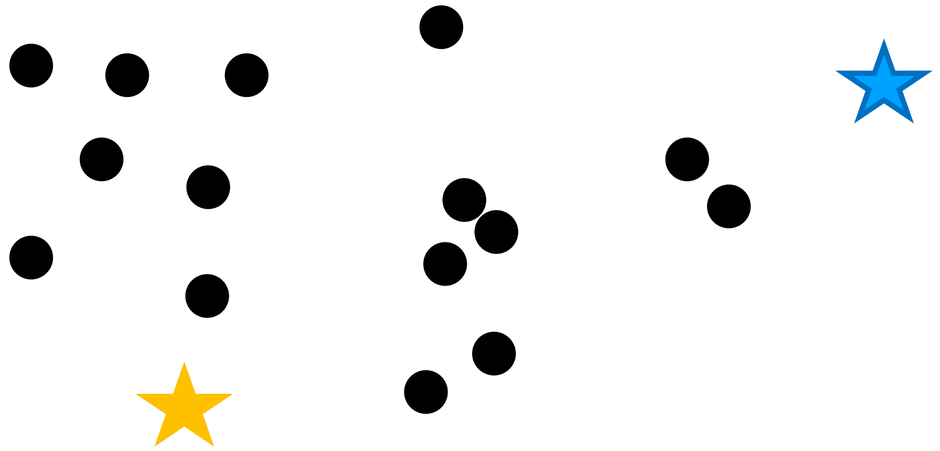
D. {1,2,4,5}, {7.25}

Iteration 1: merge 1 and 2
Iteration 2: merge 4 and 5
Iteration 3: Now we have clusters {1,2}, {4,5}, {7.25}.
distance({1,2}, {4,5})= 3
distance({4,5}, {7.25}) = 2.75
distance({1,2}, {7.25}) is clearly larger than the above two.
So average linkage will merge {4,5} and {7.25}



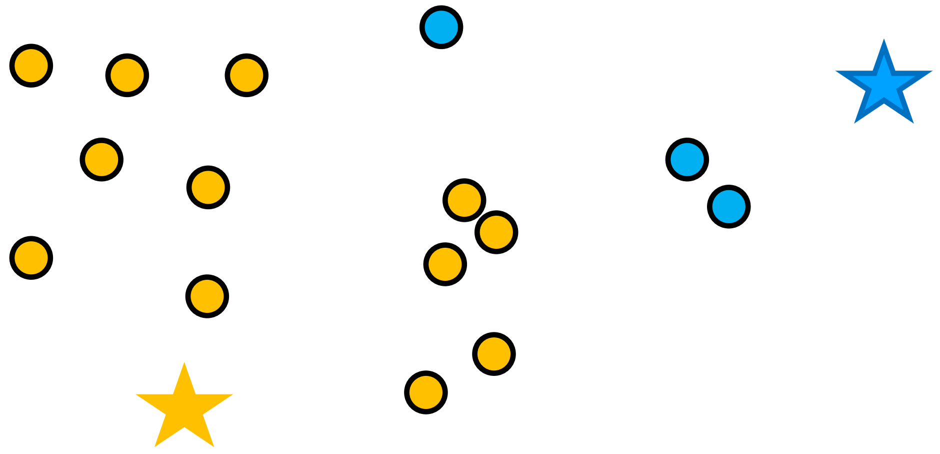
K-Means Clustering

Steps: **1**. Randomly pick k cluster centers



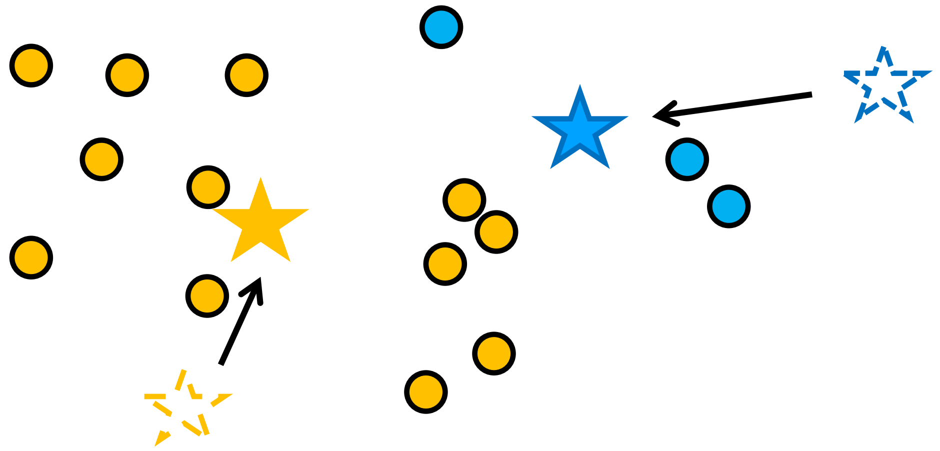
K-Means Clustering

2. Find closest center for each point



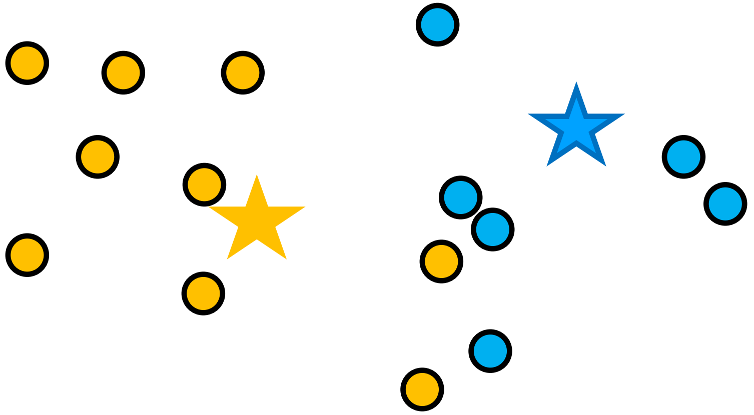
K-Means Clustering

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?

The optimization problem of k-means

- What is k-means trying to optimize?

$$\min_{c,y} \sum_{i=1}^{n} ||x_i - c_{y(x_i)}||^2$$

Questions on k-means

- Will k-means stop (converge)? Yes

Given a fixed dataset and a fixed number of clusters, there are only a **finite number of ways** to assign data points to clusters.

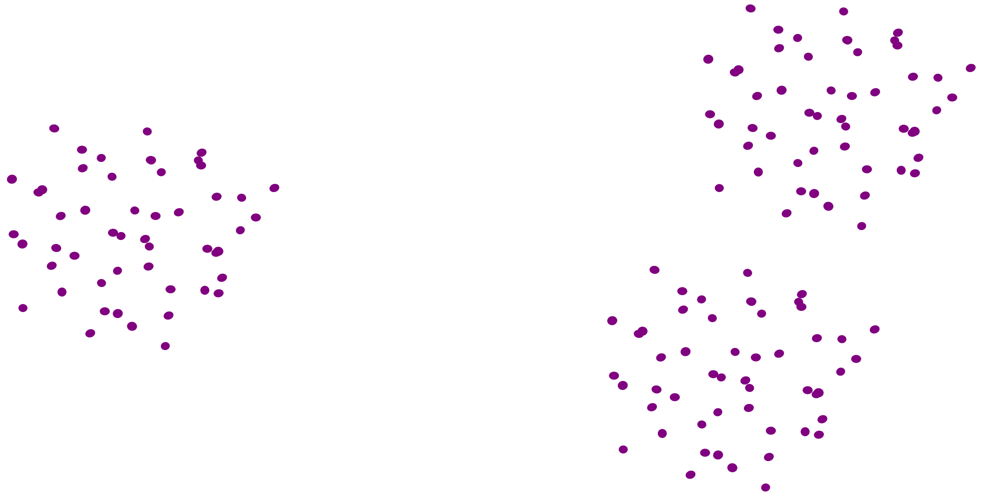
Each iteration consists of:

- Assignment Step: Assign each data point to the closest centroid.
- Update Step: Recompute centroids as the mean of assigned points.

These steps **always reduce or keep the same** the objective function (sum of squared distance), ensuring termination.

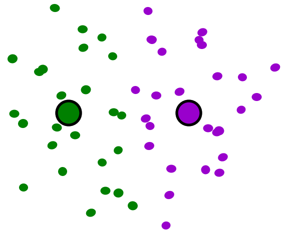
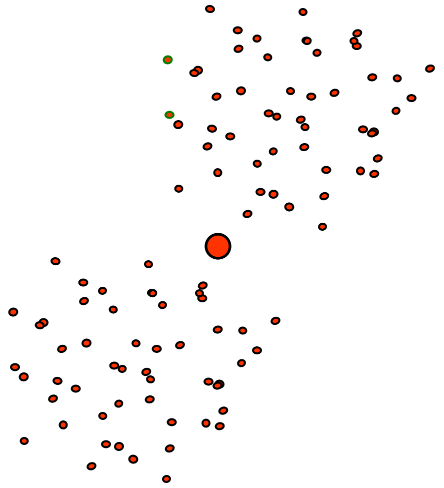
Questions on k-means

- Will it find a global or local optimum? (sadly no guarantee)



Questions on k-means

- Will it find a global or local optimum? (sadly no guarantee)

Questions on k-means

How many clusters should we use?

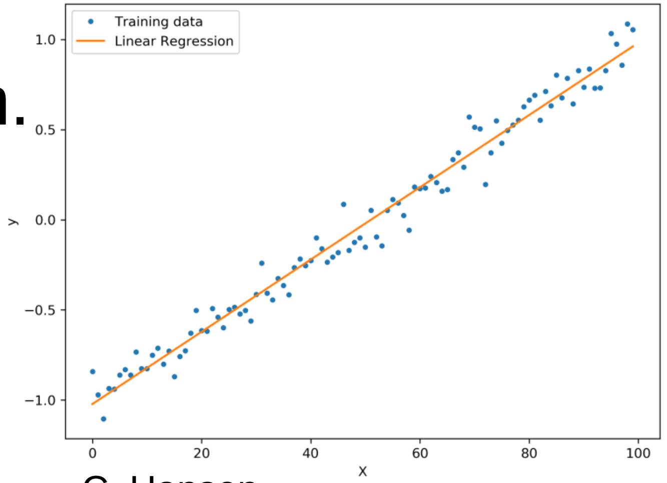
- Difficult problem
- Domain knowledge
- Elbow Method
  - Compute the within-cluster sum of squares (WCSS) for different values of k.
  - Plot WCSS vs. k and look for the **elbow point** where the reduction in WCSS slows down. The optimal k is typically at this elbow.

Linear Regression

Simplest type of regression problem.

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

- *x*'s are vectors, *y*'s are scalars.
- "**Linear**": predict a linear combination of x components + intercept



C. Hansen

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

Want: parameters θ

Linear Regression Setup

Problem Setup

Goal: figure out how to minimize square loss

Let’s organize it. Train set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

- Since $f(x) = \theta_0 + x^T\theta$, use a notational trick by augmenting feature vector with a constant dimension of 1:

$$x = \begin{bmatrix} 1 \\ x \end{bmatrix}$$

- Then, with this one more dimension we can write (θ contains θ_0 now) $f(x) = x^T\theta$

Linear Regression Setup

Problem Setup

Train set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$

Take train features and make it a n*(d+1) matrix, and y a vector:

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

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”

How Good are the Estimated Parameters?

Now we have parameters $\hat{\theta} = (X^T X)^{-1} X^T y$

How good are they?

Predictions are $f(x_i) = \hat{\theta}^T x_i = ((X^T X)^{-1} X^T y)^T x_i$

Errors ("residuals")

$$|y_i - f(x_i)| = |y_i - \hat{\theta}^T x_i| = |y_i - ((X^T X)^{-1} X^T y)^T x_i|$$

If data is linear, residuals are 0. Almost never the case!

Mean squared error on a test set $\frac{1}{m} \sum_{i=n+1}^{n+m} \left(\hat{\theta}^T x_i - y_i\right)^2$

Linear Regression → Classification?

What if we want the same idea, but *y* is 0 or 1?

Need to convert the $\theta^T x$ to a probability in [0,1]

$$p(y=1|x) = \frac{1}{1+\exp(-\theta^T x)}$$

Logistic function

Why does this work?

If $\theta^T x$ is really big, $\exp(-\theta^T x)$ is really small → *p* close to 1

If really negative exp is huge → *p* close to 0

"Logistic Regression"

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:

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

$$\hat{y}_2 = 2 \ * \beta_1 + 3 * \beta_2 + 1 * \beta_3 = 8$$
$$\ell(\hat{y}_1, y_1) = (8 \ - 4)^2 = 16$$

MSE = (16 + 9) / 2 = 25 / 2

Review: Perceptron

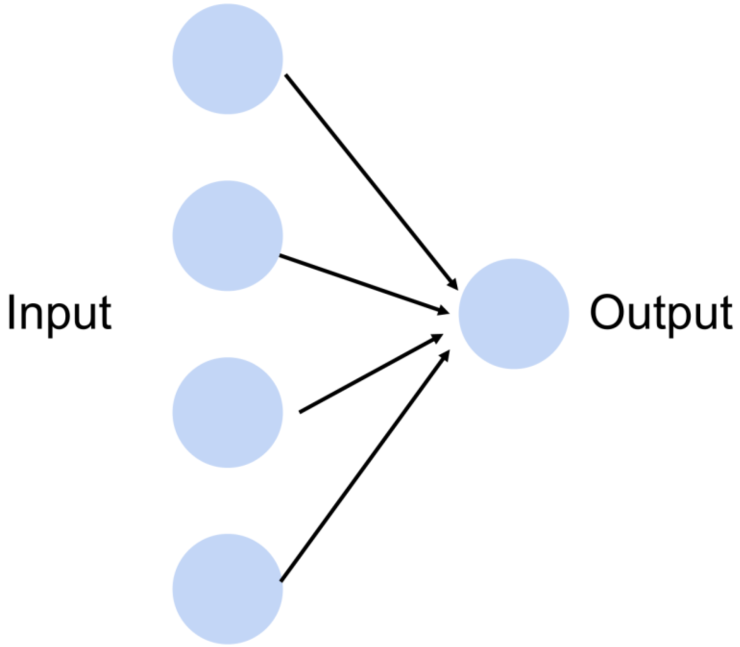
- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b , perceptron outputs:

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

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

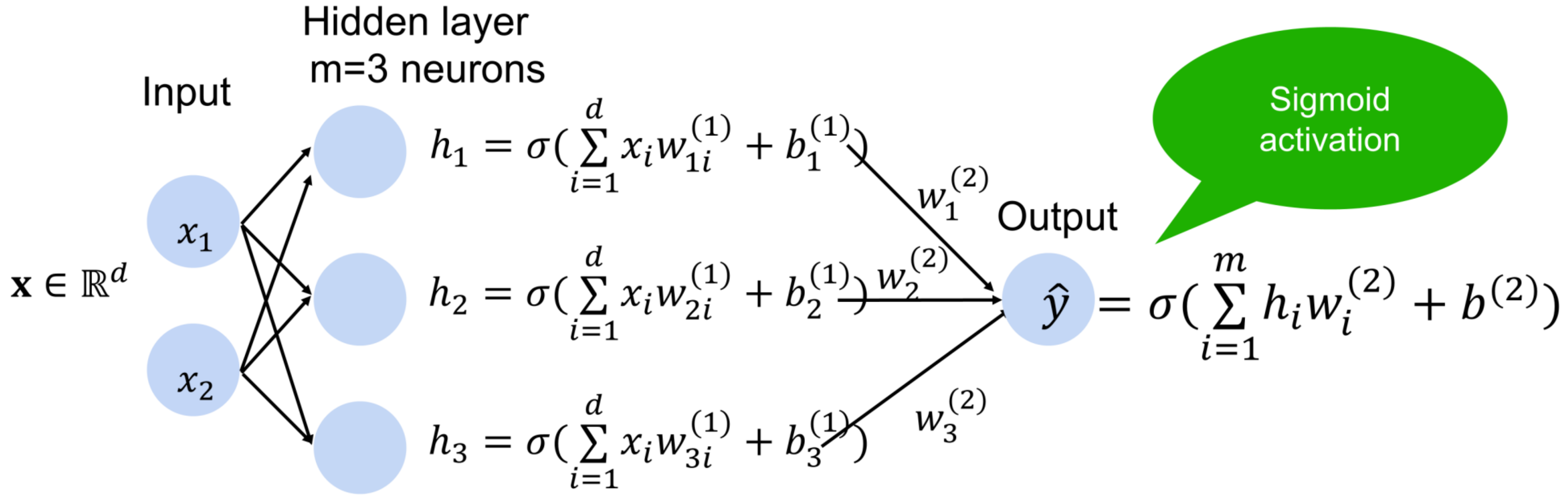
Activation function

Cats vs. dogs?



Multi-layer perceptron: Example

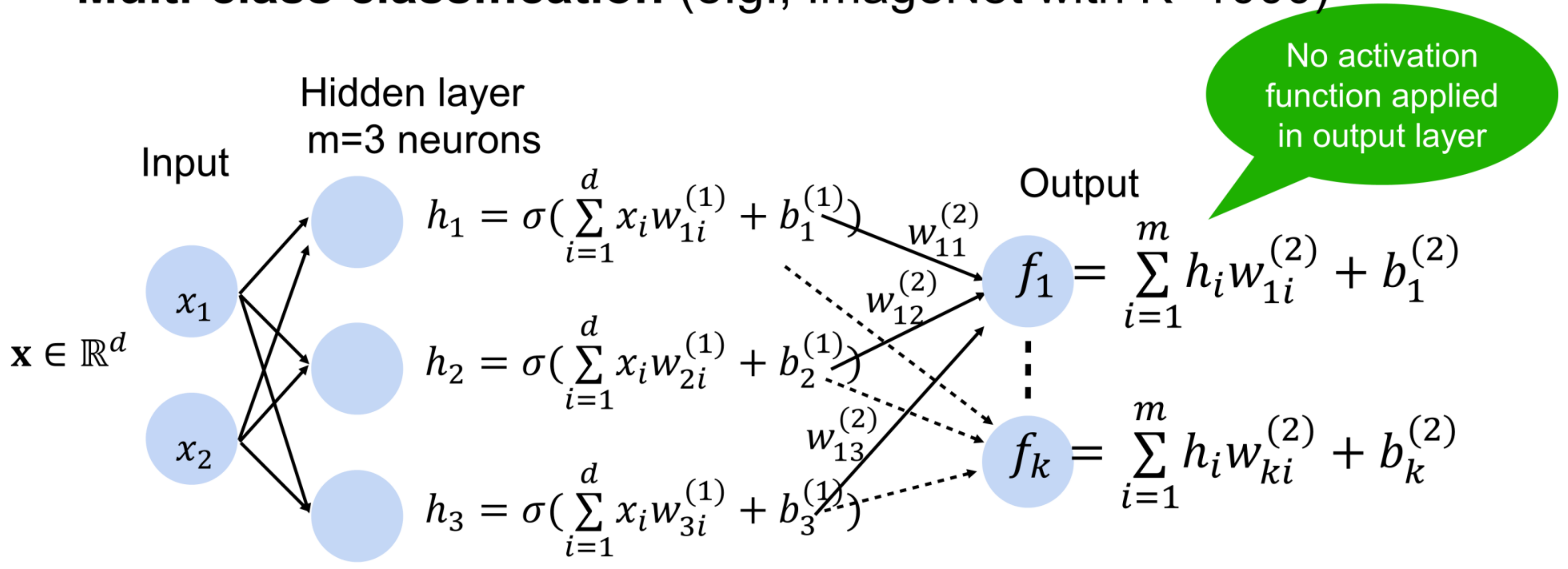
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2


Hidden layer
m=3 neurons
Input
x_1
x_2
$\mathbf{x} \in \mathbb{R}^d$
$h_1 = \sigma(\sum_{i=1}^{d} x_i w_{1i}^{(1)} + b_1^{(1)})$
$h_2 = \sigma(\sum_{i=1}^{d} x_i w_{2i}^{(1)} + b_2^{(1)})$
$h_3 = \sigma(\sum_{i=1}^{d} x_i w_{3i}^{(1)} + b_3^{(1)})$
$w_1^{(2)}$
$w_2^{(2)}$
$w_3^{(2)}$
Output
$\hat{y} = \sigma(\sum_{i=1}^{m} h_i w_i^{(2)} + b^{(2)})$
Sigmoid activation

Neural network for K-way classification

- K outputs in the final layer

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



No activation function applied in output layer
Hidden layer m=3 neurons
Input
Output
$\mathbf{x} \in \mathbb{R}^d$
x_1
x_2
$h_1 = \sigma(\sum_{i=1}^{d} x_i w_{1i}^{(1)} + b_1^{(1)})$
$h_2 = \sigma(\sum_{i=1}^{d} x_i w_{2i}^{(1)} + b_2^{(1)})$
$h_3 = \sigma(\sum_{i=1}^{d} x_i w_{3i}^{(1)} + b_3^{(1)})$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$w_{13}^{(2)}$
$f_1 = \sum_{i=1}^{m} h_i w_{1i}^{(2)} + b_1^{(2)}$
$f_k = \sum_{i=1}^{m} h_i w_{ki}^{(2)} + b_k^{(2)}$

More complicated neural networks: multiple hidden layers

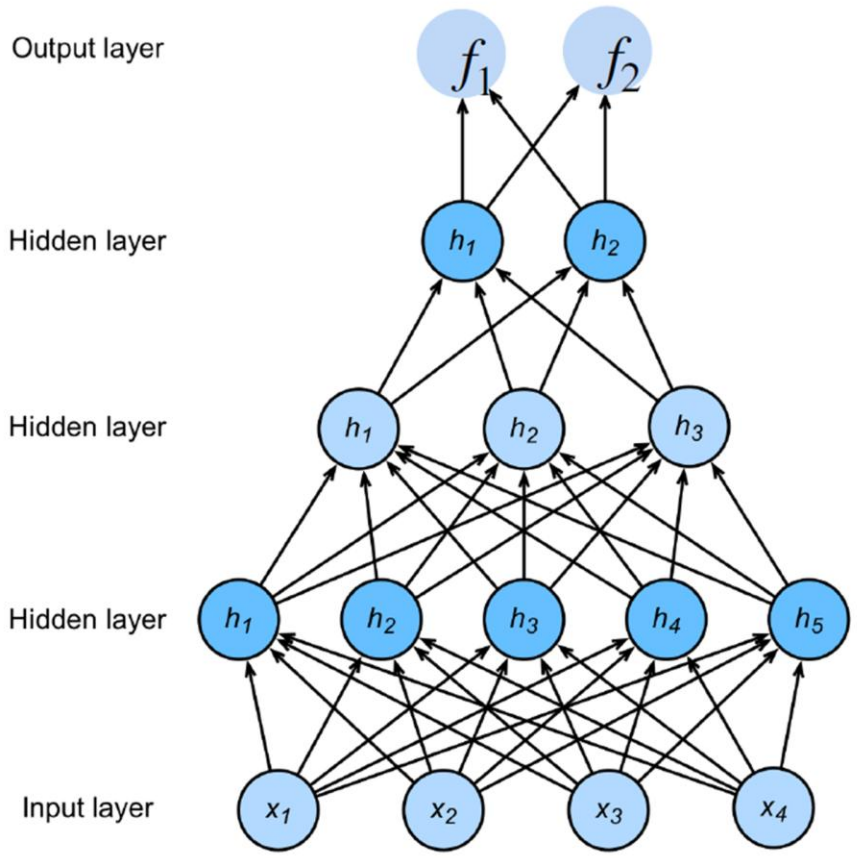
$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

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

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

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

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



How to train a neural network? Binary classification

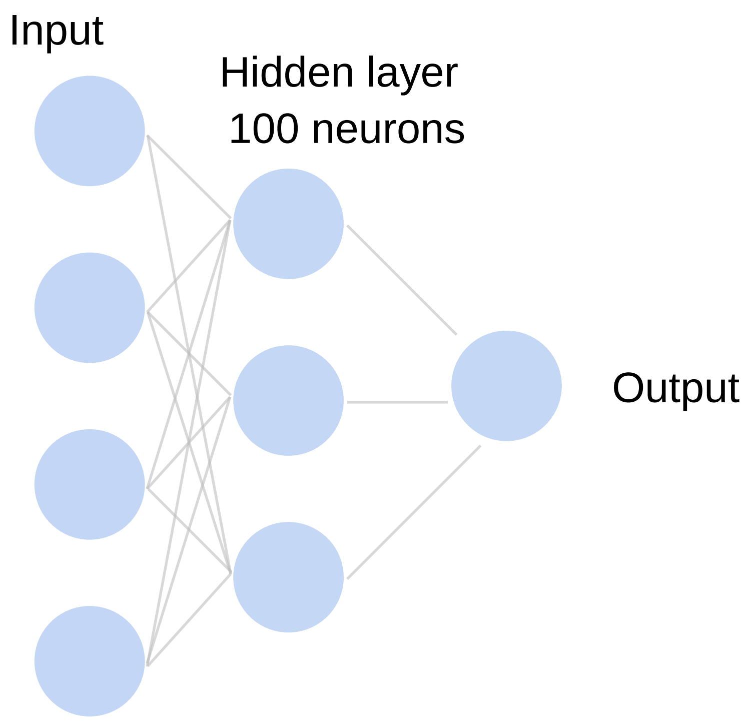
Loss function: $$\frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$$

Per-sample loss:

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

$\uparrow$

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? Multiclass

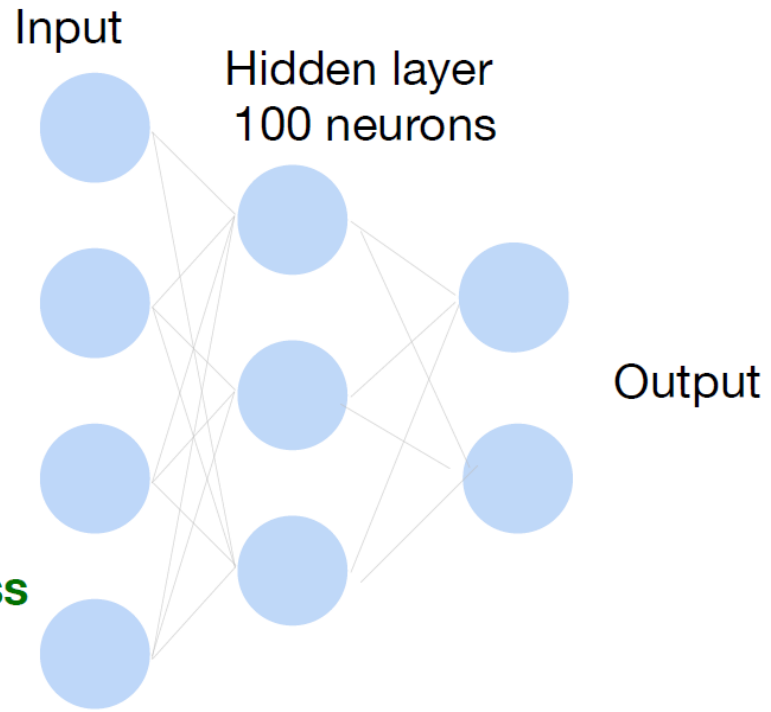
Loss function: $\frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$

Per-sample loss:

$$\ell(\mathbf{x}, y) = \sum_{k=1}^{K} - Y_k \log p_k = - \log p_y$$

where Y is one-hot encoding of y

Also known as cross-entropy loss or softmax loss

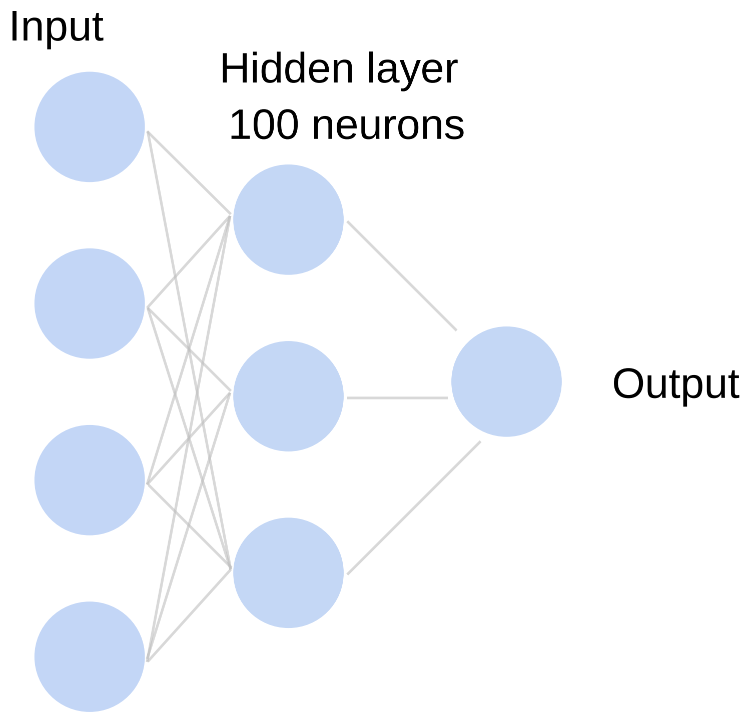


How to train a neural network?

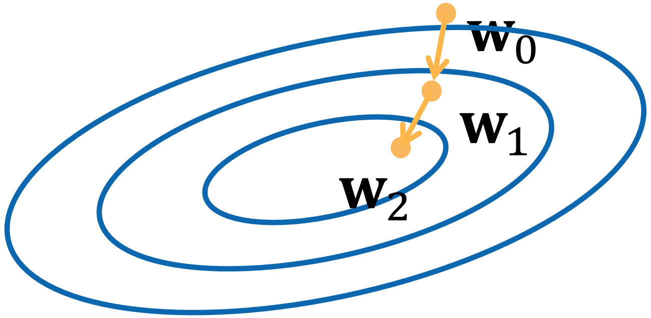
Update the weights W to minimize the loss function

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

Use gradient descent!



Gradient Descent



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

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

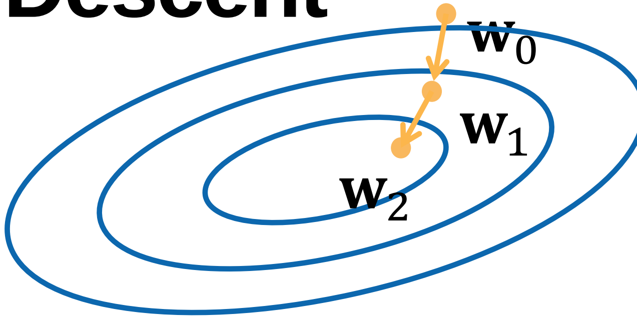
D can be very large. Expensive per iteration

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

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

- Repeat until converges

Minibatch Stochastic Gradient Descent

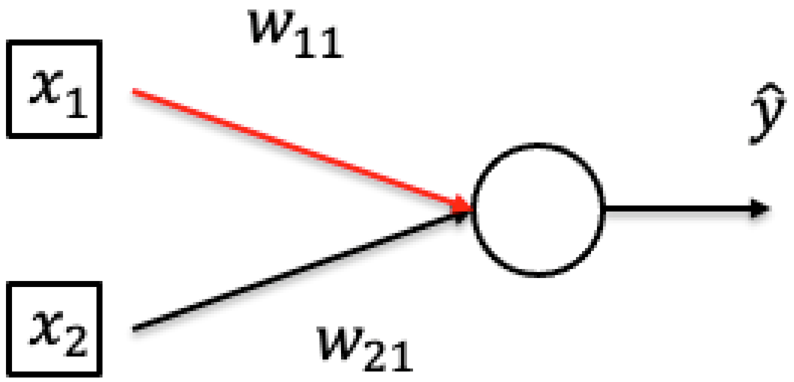


- Choose a learning rate $\alpha > 0$
- Initialize the model parameters w_0
- For t =1,2,…
  - **Randomly sample a subset (mini-batch)** $B \subset D$Update parameters:

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

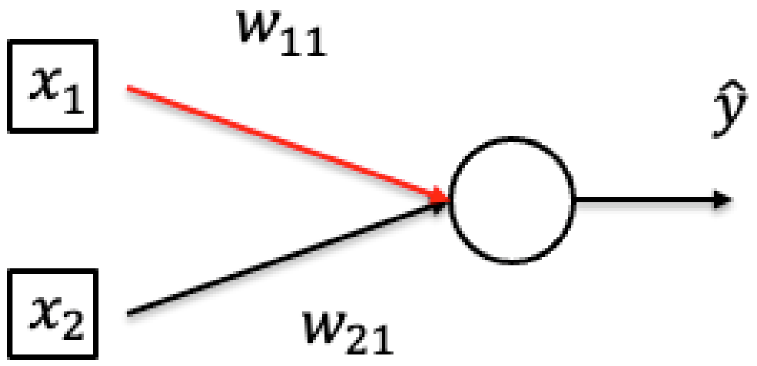
- Repeat until converges

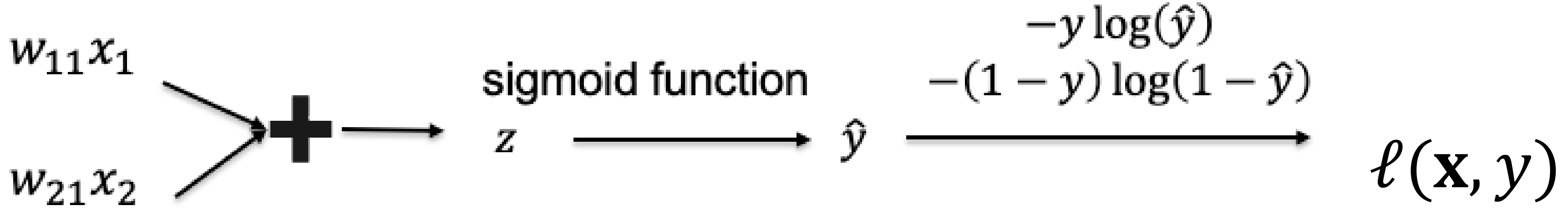
Calculate Gradient (on one data point)



- Want to compute $\frac{\partial \ell(\mathbf{x}, y)}{\partial w_{11}}$
- Data point: $((x_1, x_2), y)$

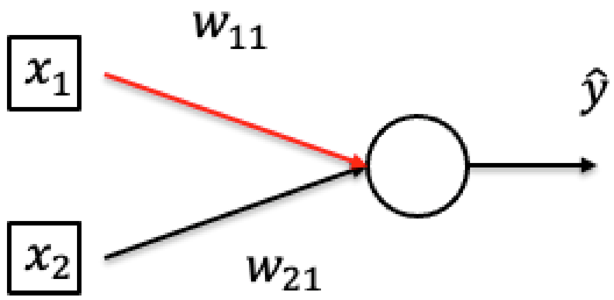
Calculate Gradient (on one data point)





Use chain rule!

Calculate Gradient (on one data point)

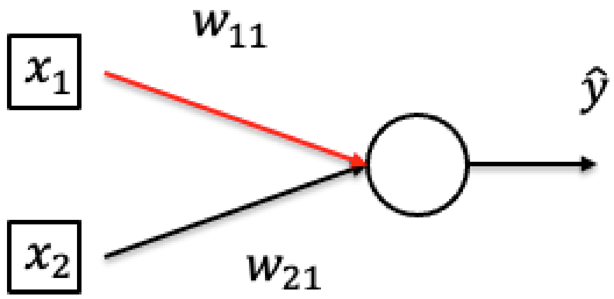


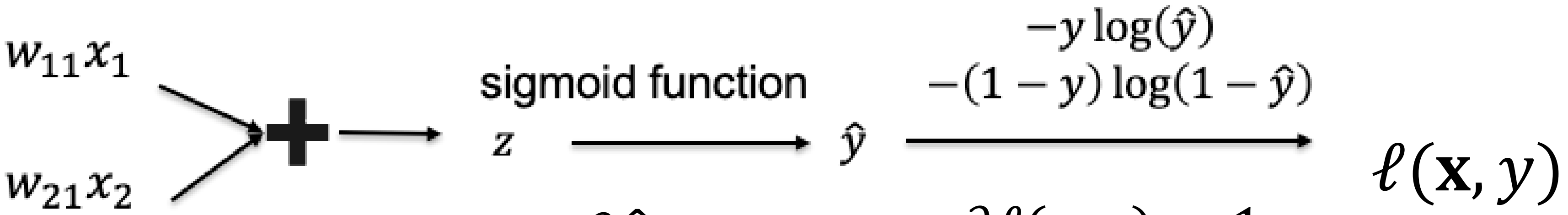
$$\begin{tikzcd}
w_{11}x_1 \arrow[dr] & & & & \\
& + \arrow[r] & z \arrow[r, "\text{sigmoid function}"] & \hat{y} \arrow[r, "\substack{-y\log(\hat{y}) \\ -(1-y)\log(1-\hat{y})}"] & \ell(\mathbf{x}, y) \\
w_{21}x_2 \arrow[ur] & & & &
\end{tikzcd}$$

$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) \qquad \frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- By chain rule: $\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{11}}$

Calculate Gradient (on one data point)

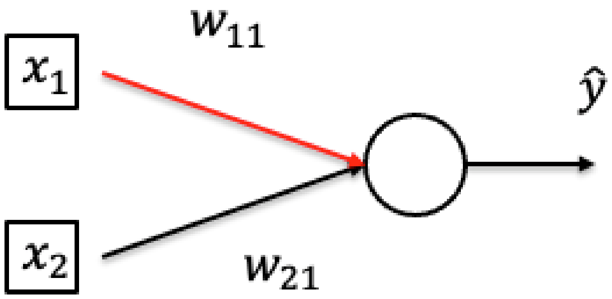


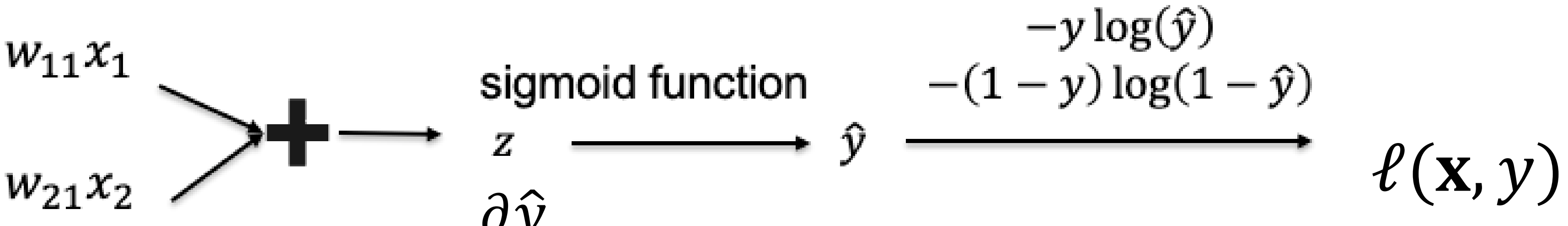


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) \qquad \frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- By chain rule: $\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z}$ x_1

Calculate Gradient (on one data point)

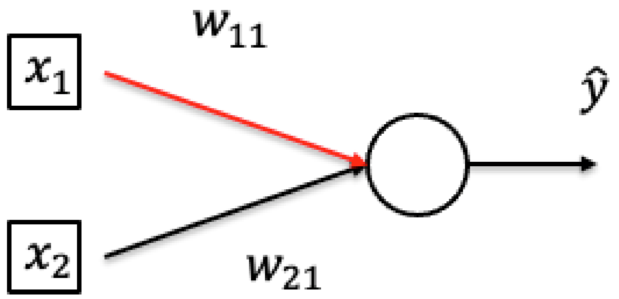




$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \;\; \hat{y}(1-\hat{y})x_1$

Calculate Gradient (on one data point)

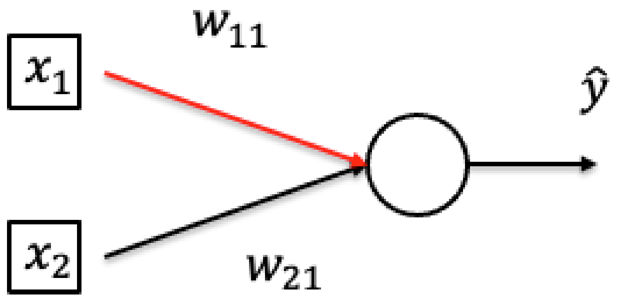


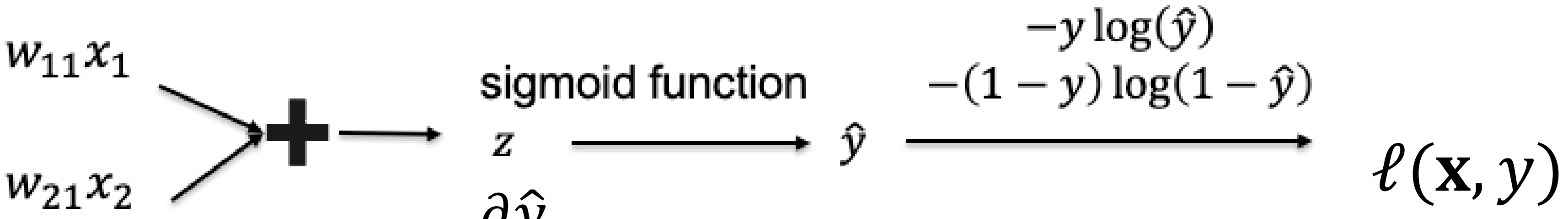
$$\begin{tikzcd}
w_{11}x_1 \arrow[dr] & & & & \\
& + \arrow[r] & z \arrow[r, "\text{sigmoid function}"] & \hat{y} \arrow[r, "\substack{-y\log(\hat{y}) \\ -(1-y)\log(1-\hat{y})}"] & \ell(\mathbf{x}, y) \\
w_{21}x_2 \arrow[ur] & & & &
\end{tikzcd}$$

$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $\dfrac{\partial l}{\partial w_{11}} = \left(\dfrac{1-y}{1-\hat{y}} - \dfrac{y}{\hat{y}}\right)\hat{y}(1-\hat{y})x_1$

Calculate Gradient (on one data point)

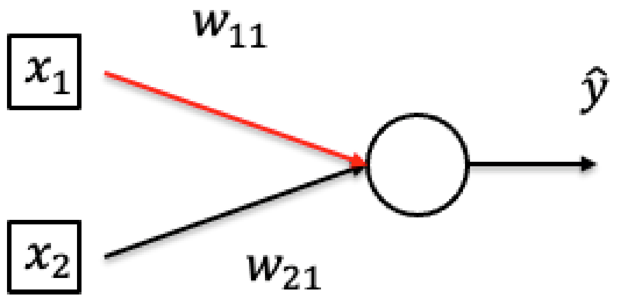


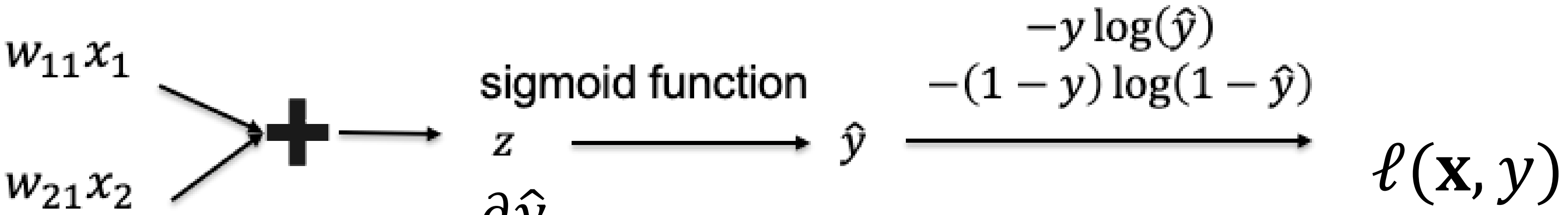


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $\frac{\partial l}{\partial w_{11}} = (\hat{y} - y)x_1$

Calculate Gradient (on one data point)

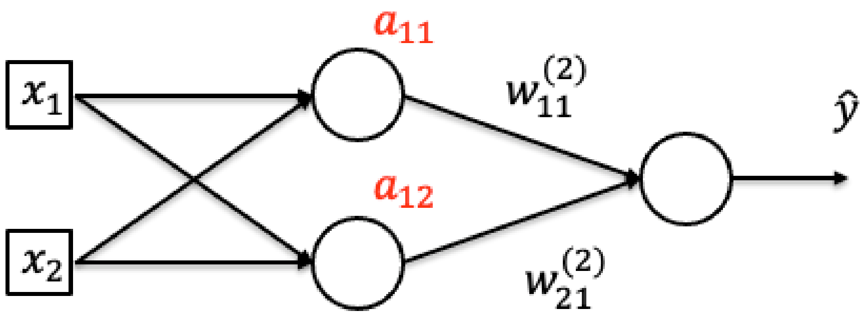




$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule $\frac{\partial l}{\partial x_1} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} w_{11} = (\hat{y} - y) w_{11}$

Calculate Gradient (on one data point)



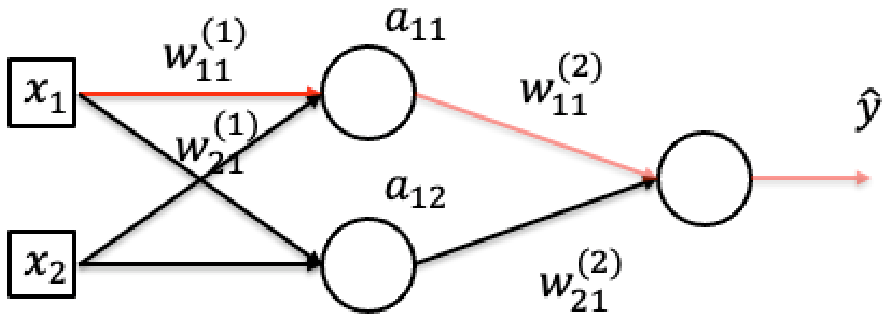
Make it deeper

$$\begin{tikzcd}
w_{11}^{(2)} a_{11} \arrow[dr] & & & & \\
& + \arrow[r] & z \arrow[r, "\text{sigmoid function}"] & \hat{y} \arrow[r, "{-y\log(\hat{y}) \atop -(1-y)\log(1-\hat{y})}"] & \ell(\mathbf{x}, y) \\
w_{21}^{(2)} a_{12} \arrow[ur] & & & &
\end{tikzcd}$$

$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain ru $\frac{\partial l}{\partial a_{11}} = (\hat{y}-y)w_{11}^{(2)}, \ \frac{\partial l}{\partial a_{12}} = (\hat{y}-y)w_{21}^{(2)}$

Calculate Gradient (on one data point)

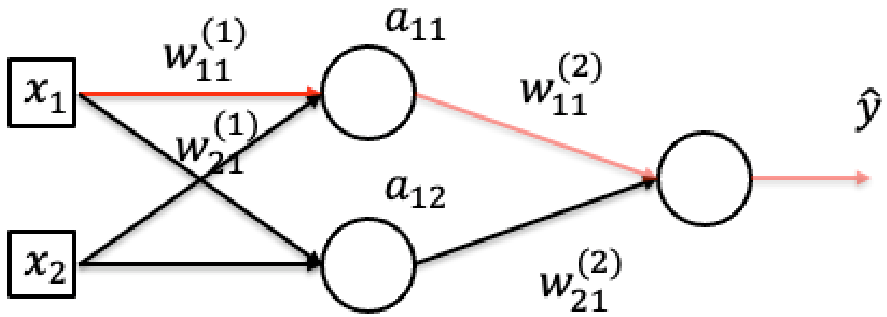


$$\begin{tikzcd}
w_{11}^{(1)} x_1 \arrow[dr] & & & & \\
& \boldsymbol{+} \arrow[r] & z_{11} \arrow[r, "\sigma(z_{11})"] & a_{11} \arrow[r] & l(\boldsymbol{x}, y) \\
w_{21}^{(1)} x_2 \arrow[ur] & & & &
\end{tikzcd}$$

$$\frac{\partial a_{11}}{\partial z_{11}} = \sigma'(z_{11}) \qquad \frac{\partial l}{\partial a_{11}} = (\hat{y} - y) w_{11}^{(2)}$$

- By chain rule: $\dfrac{\partial l}{\partial w_{11}^{(1)}} = \dfrac{\partial l}{\partial a_{11}} \dfrac{\partial a_{11}}{\partial w_{11}^{(1)}} = (\hat{y} - y) w_{11}^{(2)} \dfrac{\partial a_{11}}{\partial w_{11}^{(1)}}$

Calculate Gradient (on one data point)



$$\begin{tikzcd}
w^{(1)}_{11}x_1 \arrow[dr] & & & & \\
& + \arrow[r] & z_{11} \arrow[r, "\sigma(z_{11})", "\frac{\partial a_{11}}{\partial z_{11}} = \sigma'(z_{11})"'] & a_{11} \arrow[r, "", "\frac{\partial l}{\partial a_{11}} = (\hat{y}-y)w^{(2)}_{11}"'] & l(\boldsymbol{x}, y) \\
w^{(1)}_{21}x_2 \arrow[ur] & & & &
\end{tikzcd}$$

- By chain rule $\dfrac{\partial l}{\partial w^{(1)}_{11}} = \dfrac{\partial l}{\partial a_{11}} \dfrac{\partial a_{11}}{\partial w^{(1)}_{11}} = (\hat{y}-y)w^{(2)}_{11} a_{11}(1-a_{11})x_1$

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?

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

Thanks!