Lecture 13 – Training Neural Networks

Prof. Maggie Makar

Announcements

mmakar@umich.edu

- Midterm-related
  - This is not a review session!
  - Piazza will be frozen at 2pm on Wednesday 3/6
  - There will be no quiz or discussions this week
  - Please look at the calendar for office hours
  - Recall your midterm location
  - **Bring your Mcard to the exam**
- HW2 solutions have been released
- Don't forget to fill out class midterm evaluations (by 3/6/2024)

Midterm: Preamble

EECS 445 — Introduction to Machine Learning: Sample Midterm

Winter 2024

Name: ______________________________ uniqname: ______________________________(1pt)

You have **110 minutes** to complete this exam (from the time you turn past this cover page to the time you make the last mark on any page other than this cover page). As indicated above, **filling in your name and uniqname is worth 1 point of the exam total**. This is a closed everything exam (including books, web, class notes, etc.) except for one double-sided 8.5×11 inch piece of paper with notes prepared by you. No electronic devices are allowed (this includes calculators, cellphones, etc.).
When you are finished, sign the honor code statement below.

"I have neither given nor received aid on this examination, nor have I concealed any violations of the honor code."

Signed: ______________________________

Midterm: additional instructions

1. **DO NOT DETACH PAGES FROM THE EXAM.** Failure to comply may result in point deductions.

6. If you think something about a question is open to interpretation, feel free to ask the course staff or make a note on the exam.

9. If you are still in the exam room within the last 10 minutes of the exam, you must remain seated inside the classroom until the end of the exam time.

10. Before you leave the exam room, **be sure to turn in your exam to the proctor** **<u>and</u>** **sign the sheet provided with your *uniqname* to confirm your submission.**

When solving the midterm

- Do not read into test design choices (e.g., number of pages, space allocated for the question)
- Read the instructions carefully
- Optional justifications are...optional!

Outline

- Recap: introduction to neural networks
- Neural networks as universal approximators
- Training neural networks
  - Stochastic gradient descent setup
  - Motivating backpropagation
  - Backpropagation updates
  - Practically: what does training look like?

Recall that Neural networks:

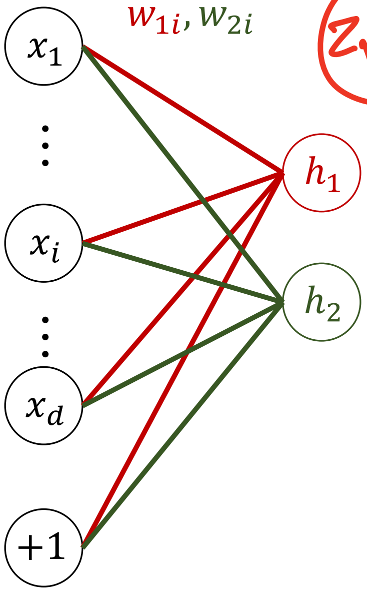
- Can be used for classification, regression, multi-class classification
- Learned, flexible feature mappings (aka representations)
- Flexibility comes from:
  - Non-linear transformations
  - Learnable parameters
  - Stacking and combining many representations

Two hidden neurons, one hidden layer

- $\bar{x} = [x_1, \ldots, x_d]^\top$

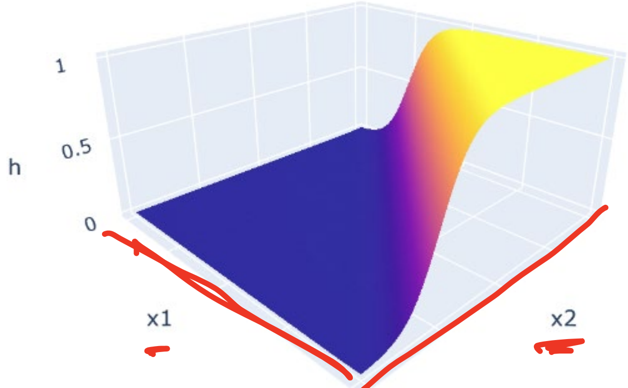
- $z_1 = \sum_i w_{1i} x_i + w_{10}$ (preactivation)
- $h_1 = g(z_1)$ (post activation)

- $z_2 = \sum_i w_{2i} x_i + w_{20}$
- $h_2 = g(z_2)$

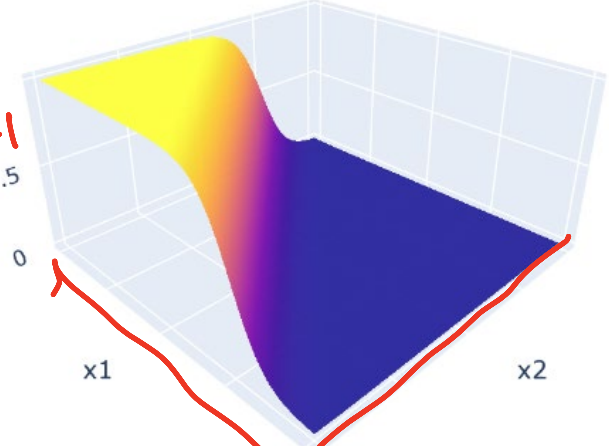


$z_1 \mid h_1$

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



h_1



h_2

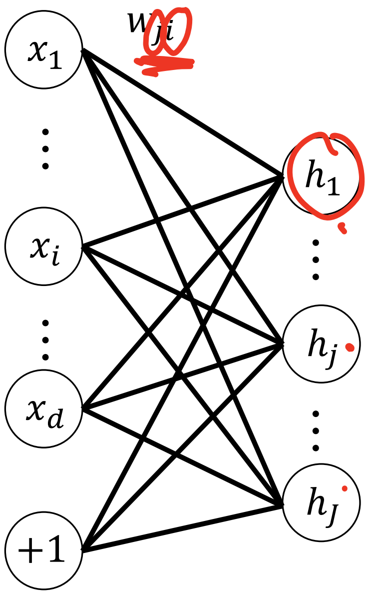
Many hidden neurons, one hidden layer

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

- $\bar{x} = [x_1, \dots, x_d]^\top$

For $j = 1, \dots, J$:

- $z_j = \sum_i w_{ji} x_i + w_{j0}$
- $h_j = g(z_j)$



$z_1 | h_1$

$z_j | h_j$

$\bar{x} \rightarrow$ $\boxed{}$

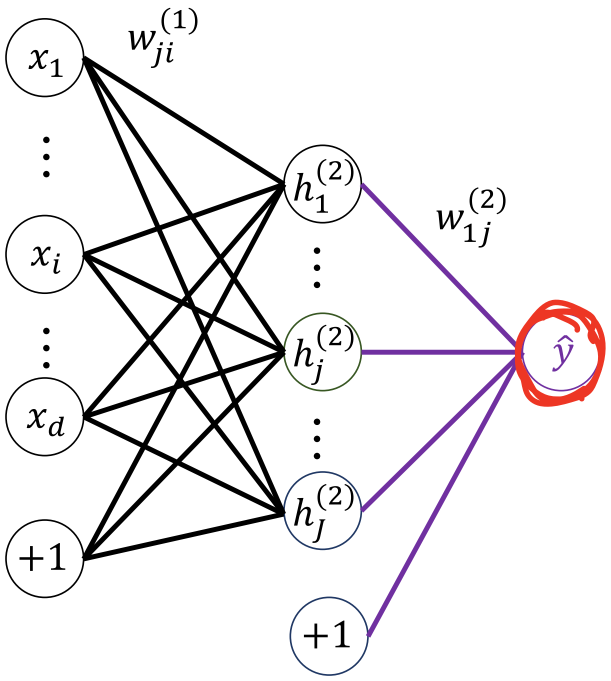
Neural network with 2 layers

- $\bar{x} = [x_1, \dots, x_d]^\top$

For $j = 1, \dots, J$:

- $z_j^{(2)} = \sum_i w_{ji}^{(1)} x_i + w_{j0}^{(1)}$
- $h_j^{(2)} = \sigma\left(z_j^{(2)}\right)$

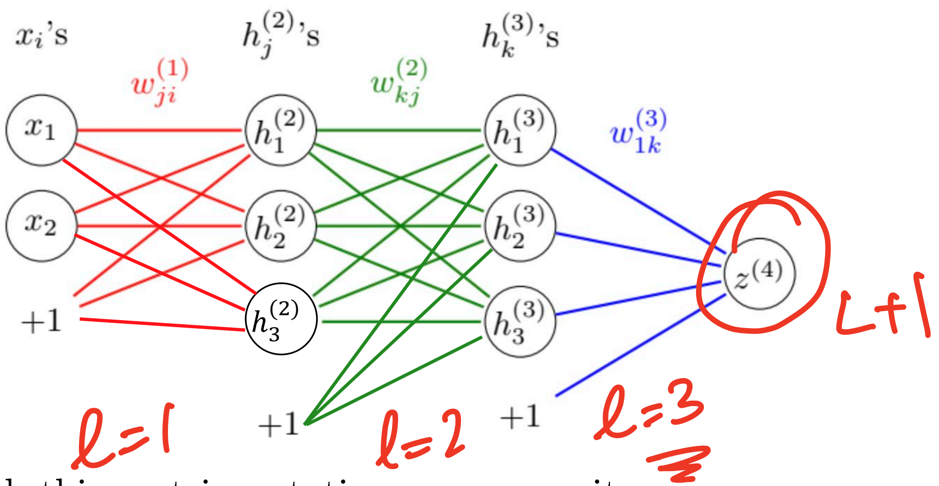
- $z_j^{(3)} = \sum_j w_{1j}^{(2)} h_j^{(2)} + w_{j0}^{(2)}$
  $:= \hat{y}$



sign $\left(z_j^{(3)}\right)$

$z_j^{(3)} = z_1^{(3)} = \hat{y}$

Recap: Matrix notation



$$z_1^{(2)} = w_{11}^{(1)}\, x_1 + w_{12}^{(1)} x_2 + w_{10}^{(1)}$$

$$z_2^{(2)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{20}^{(1)}$$

$$z_3^{(2)} = w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{30}^{(1)}$$

$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} w_{10}^{(1)} \\ w_{20}^{(1)} \\ w_{30}^{(1)} \end{bmatrix}$$

$$\bar{z}^{(2)} = W^{(1)}\, \bar{x} + \bar{b}^{(1)}$$

$$\bar{h}^{(2)} = g(\bar{z}^{(2)})$$

elementwise application g g

$$\bar{h}^{(2)} = \begin{bmatrix} g(z_1^{(2)}) \\ g(z_2^{(2)}) \\ \vdots \end{bmatrix}$$

With this matrix notation, we can write:

$\bar{h}^{(1)} = \bar{x}$

$\bar{z}^{(2)} = W^{(1)}\, \bar{h}^{(1)} + \bar{b}^{(1)}$

$\bar{h}^{(2)} = g\left(\bar{z}^{(2)}\right)$

$\bar{z}^{(3)} = W^{(2)}\, \bar{h}^{(2)} + \bar{b}^{(2)}$

…

$\bar{z}^{(j+1)} = W^{(j)}\, \bar{h}^{(j)} + \bar{b}^{(j)}$

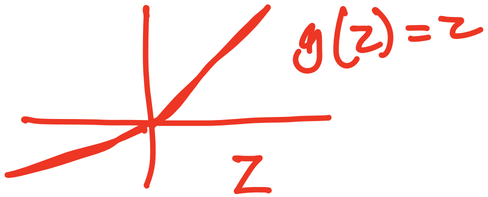
$\bar{h}^{(j+1)} = g\left(\bar{z}^{(j+1)}\right)$

…

$\bar{z}^{(L+1)} = W^{(L)}\, \bar{h}^{(L)} + \bar{b}^{(L)}$

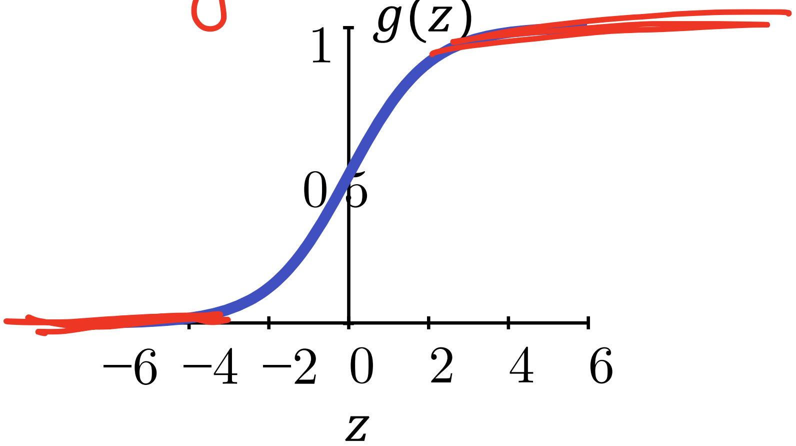
$\bar{h}^{(L+1)} = g\left(\bar{z}^{(L+1)}\right)$

Activation functions, g



Sigmoid

Range = [0,1]

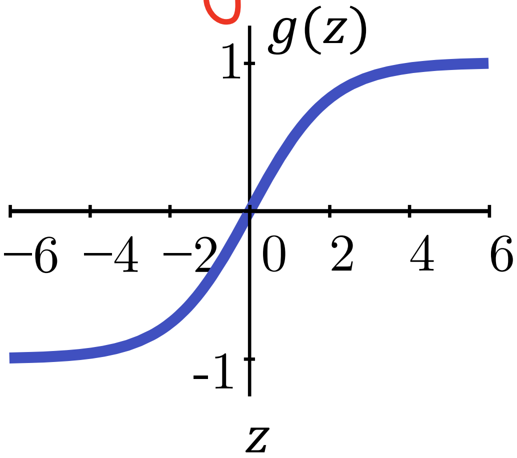


$$g(z) = \frac{1}{1 + e^{-z}}$$

tanh

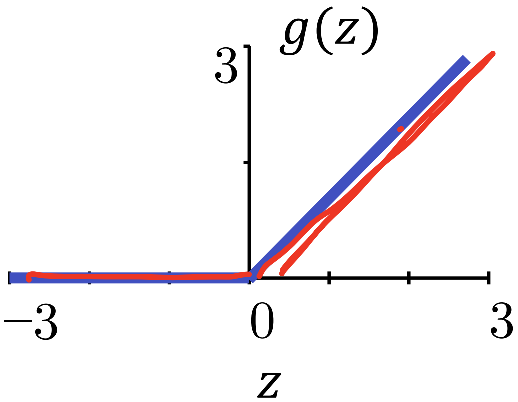
Range = [-1, +1]

g(z) = az



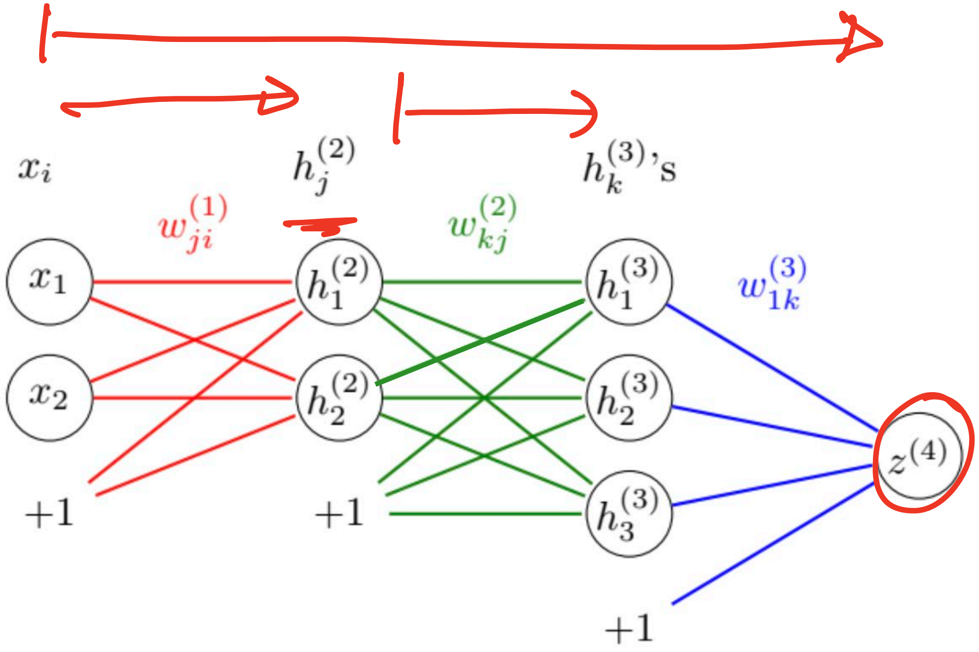
$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

ReLU



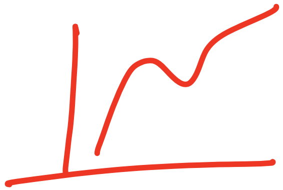
$$g(z) = \max(0, z)$$

Describing a neural network



- 3 layer NN with 2 hidden layers
  - # of layers = # of layers of trainable weights
- Fully connected layers/Dense
  - Every node in previous layer is connected to every node in the current layer
- Architecture: Describe NN
  - The number of hidden layers, nodes per hidden layer, and the way layers are connected
- Forward propagation
  - Propagating values from input to output layer

Universal approximation

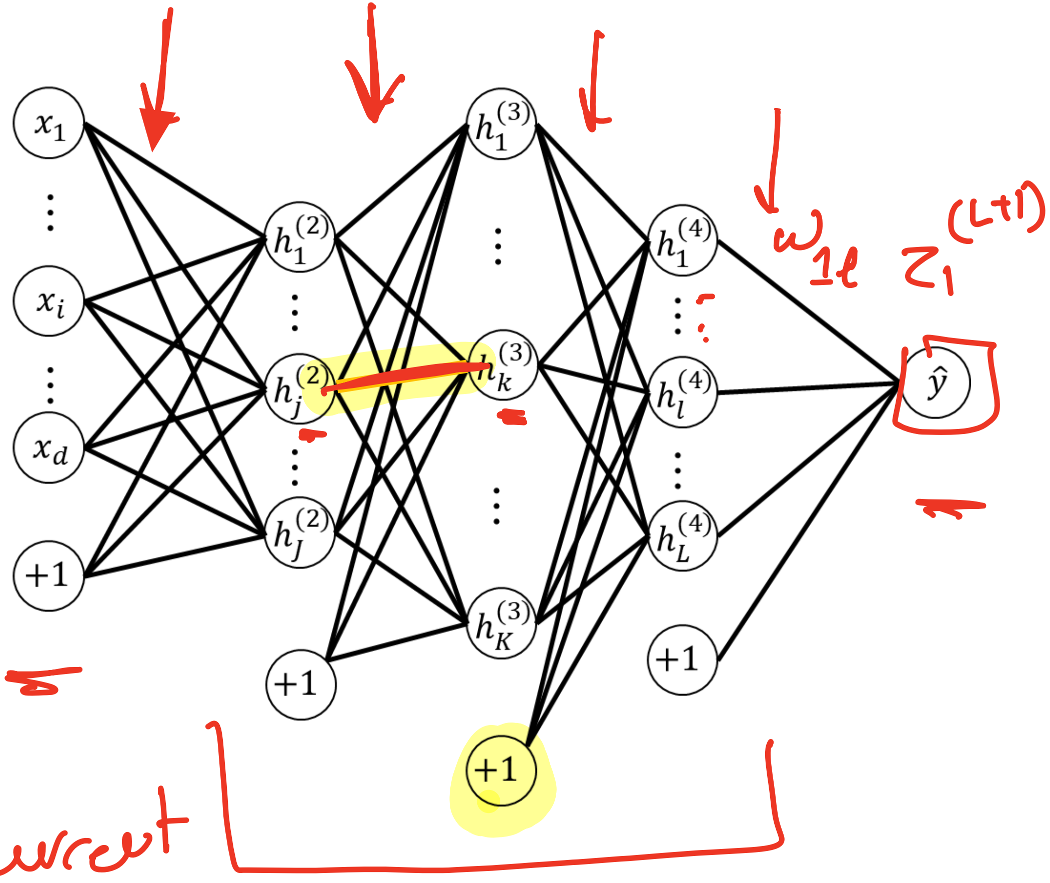


- Universal approximation theorem (Hornik, 1991)
  - A two layer NN can approximate any continuous function arbitrarily well, given a sufficiently large number of hidden neurons
- But this doesn't say anything about efficiency
  - The number of neurons can be huge
  - There might not be an efficient learning algorithm to find the necessary parameter values
- Empirically, having a large number of layers is helpful, but there is a diminishing return

more accuracy
& can efficiently find optimal NN

Your turn

- How many layers? 4
- How many hidden layers? 3
- How many neurons are in hidden layer 3?
- Let $w_{ab}^{(c)}$ denote the highlighted weight. What are the values for a, b and c? (you may use letters here)



$w_{kj}^{(2)}$

$k \to$ neuron in next layer

$j \to$ neuron in current layer.

TL;DPA:

1. Recapped NN notation and organization

2. Recapped matrix notation of NNs

3. NNs are so flexible they can approximate any function...with some caveats.

Training neural networks: Setup

- Training data

$$S_n = \{(\bar{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$$

$$\bar{x} \in \mathbb{R}^d$$

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

- Neural network

$$\bar{\theta} = [w_{10}^{(1)}, w_{11}^{(1)}, \ldots, w_{JK}^{(L)}]$$

$$h(\bar{x}; \bar{\theta}) = z^{(L+1)} = z_1^{(L+1)}$$

Loss(y z)

Loss(y, z)

- Loss function

$$J(\bar{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y^{(i)}, h(\bar{x}^{(i)}; \bar{\theta}))$$

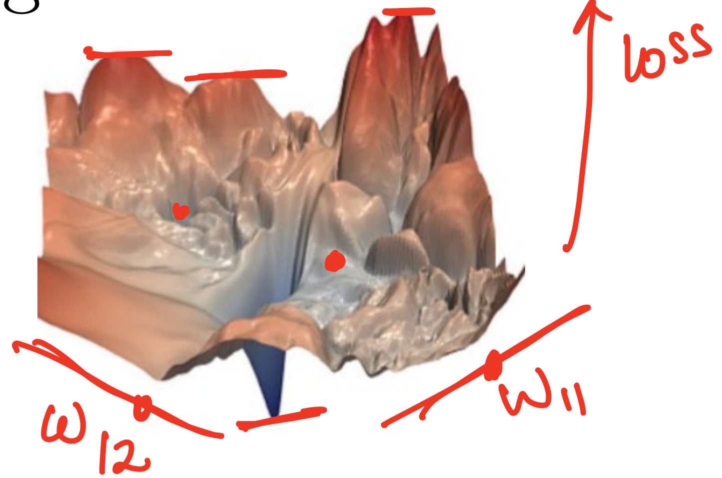
target label

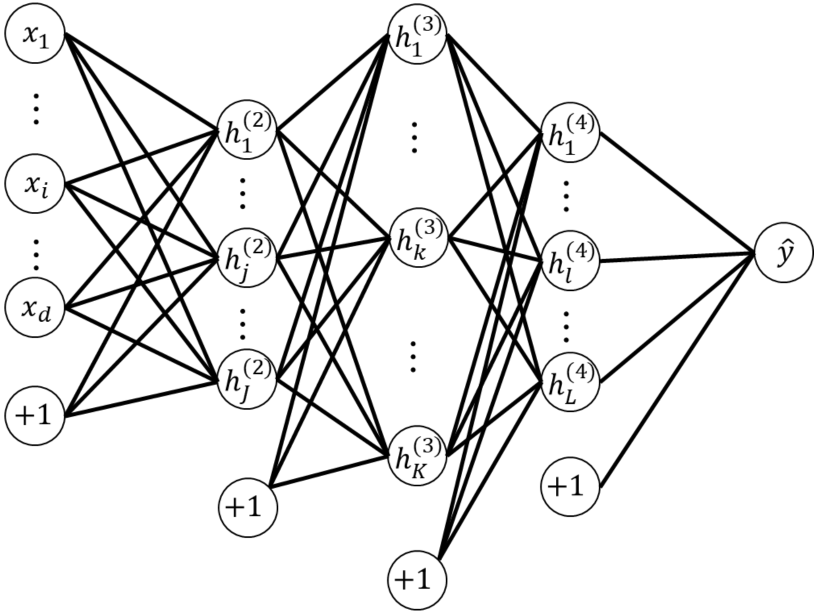
predicted label

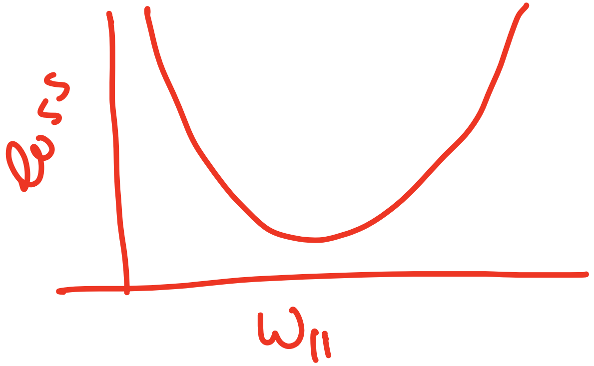
Training neural networks: challenges

GD

1. Non-convex loss → no closed form sol
2. Typically uses large datasets
3.







Training neural networks: stochastic gradient descent

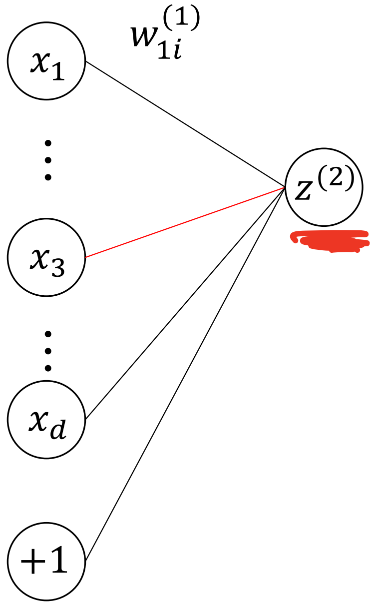
1. Initialize weights at small random values $\bar{\theta}^{(0)} \sim \text{Gaussian}(0, \sigma^2)$

shuffle datapoints

2. Sample one data point
3. Update the weights as follows

$$\underbrace{\bar{\theta}^{(k+1)}}_{\text{value @ next iteration}} = \underbrace{\bar{\theta}^{(k)}}_{\text{value @ current iteration.}} - \underbrace{\eta_k}_{\substack{\text{magnitude} \\ \text{dynamic learning rate}}} \underbrace{\nabla_{\bar{\theta}} \, \text{Loss}(y^{(i)}, h(\bar{x}; \bar{\theta}))}_{\text{direction}}$$

Is SGD enough? Single layer updates



- Single layer NN with no non-linearities, and hinge loss
- Goal: get parameter updates for all $\bar{\theta} = [w_{10}^{(1)}, w_{11}^{(1)}, \ldots, w_{1d}^{(1)}]$
- Focus on one component of $\bar{\theta}$. E.g., $w_{13}^{(1)}$

$$\begin{aligned}\text{Loss}(y, h(\bar{x};\bar{\theta})) &= \max\{1 - yh(\bar{x};\bar{\theta}), 0\} \\ &= \max\{1 - yz^{(2)}, 0\} \\ &= \max\{1 - y(\sum_i^d w_{1i}^{(1)} x_i + w_{10}^{(1)}), 0\}\end{aligned}$$

- Derivative wrt to $w_{13}^{(1)}$

$$\frac{\partial \text{Loss}(y, h(\bar{x},\bar{\theta}))}{\partial w_{13}^{(1)}} = \begin{cases} -yx_3, & \text{if } 1 - yz^{(2)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$= -y x_3 [[1 - yz^{(2)} > 0]] + 0 [[1 - yz^{(2)} < 0]]$

iteration #

layer #

- Update for the $k+1$ SGD iteration:

$$w_{13}^{(1,k+1)} = w_{13}^{(1,k)} + \eta_k y x_3 [[1 - yz^{(2)} > 0]]$$

value according to $\bar{\theta}^{(k)}$