Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

http://deeplearning.stanford.edu/tutorial/ and then go to MultiLayerNeuralNetworks
http://neuralnetworksanddeeplearning.com/

Topic 6
Neural Networks Continued

INTRODUCTION TO MACHINE LEARNING

PROF. LINDA SELLIE

Some of these slides are from Prof. Rangan

Outline

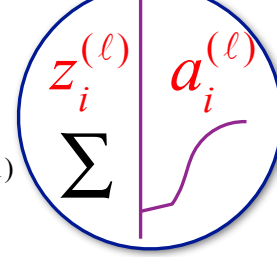
- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

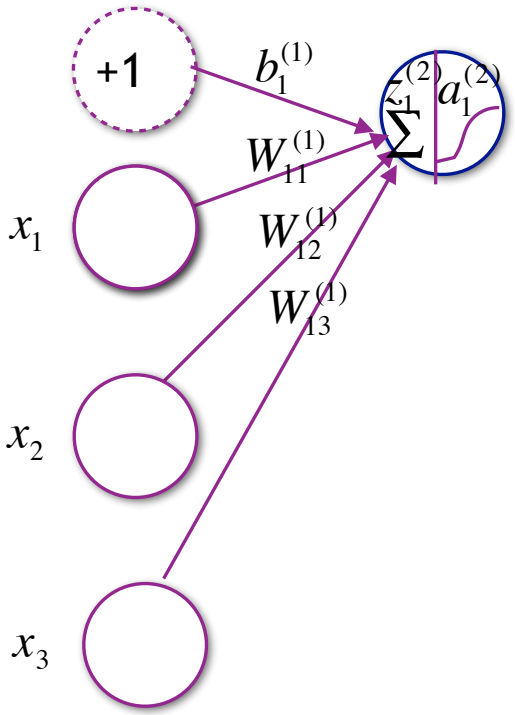
Neural Network Model & Notation

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

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)}$$

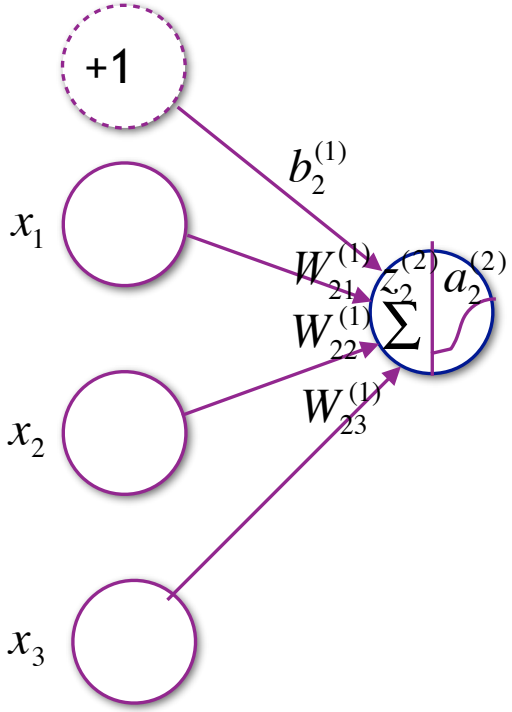
$$a_i^{(2)} = f(z_i^{(2)})$$





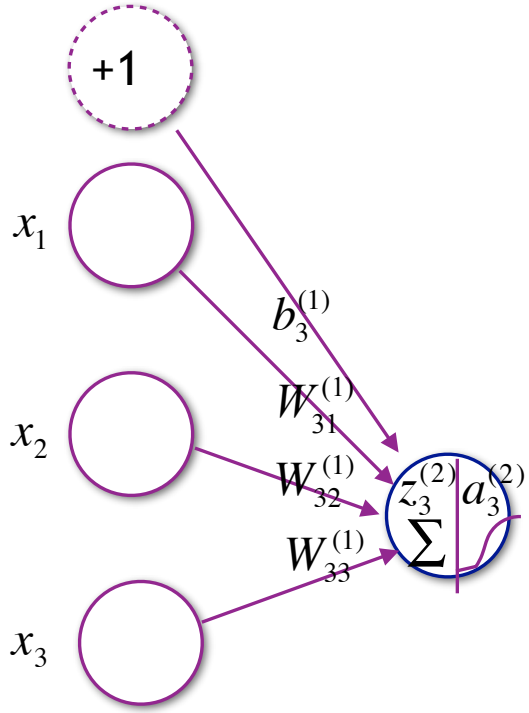
$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)}$$

$$a_1^{(2)} = f(z_1^{(2)})$$



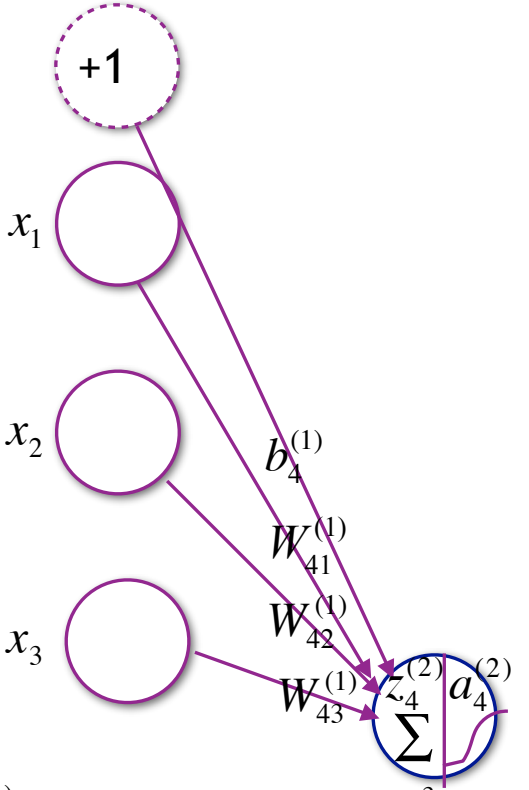
$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)}$$

$$a_2^{(2)} = f(z_2^{(2)})$$



$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)}$$

$$a_3^{(2)} = f(z_3^{(2)})$$



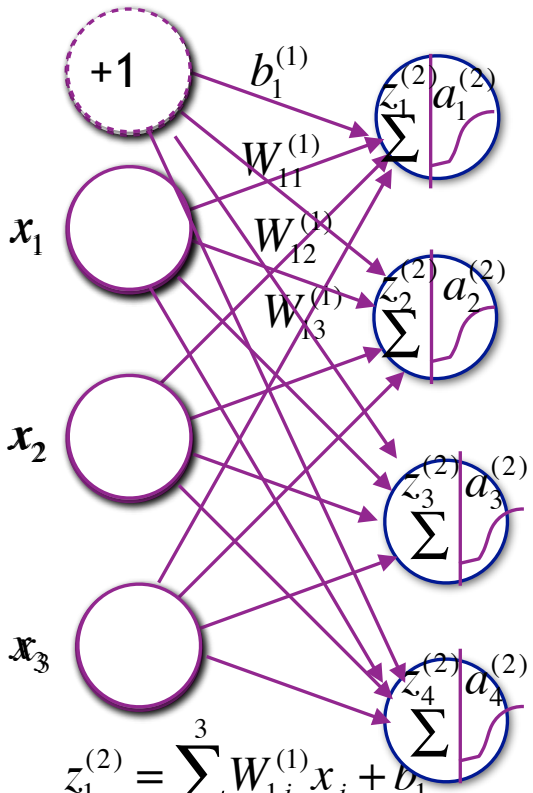
$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)}$$

$$a_4^{(2)} = f(z_4^{(2)})$$

Neural Network Model & Notation

$$z_i^{(2)} = \sum_{j=1}^{3} W_j^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

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



$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

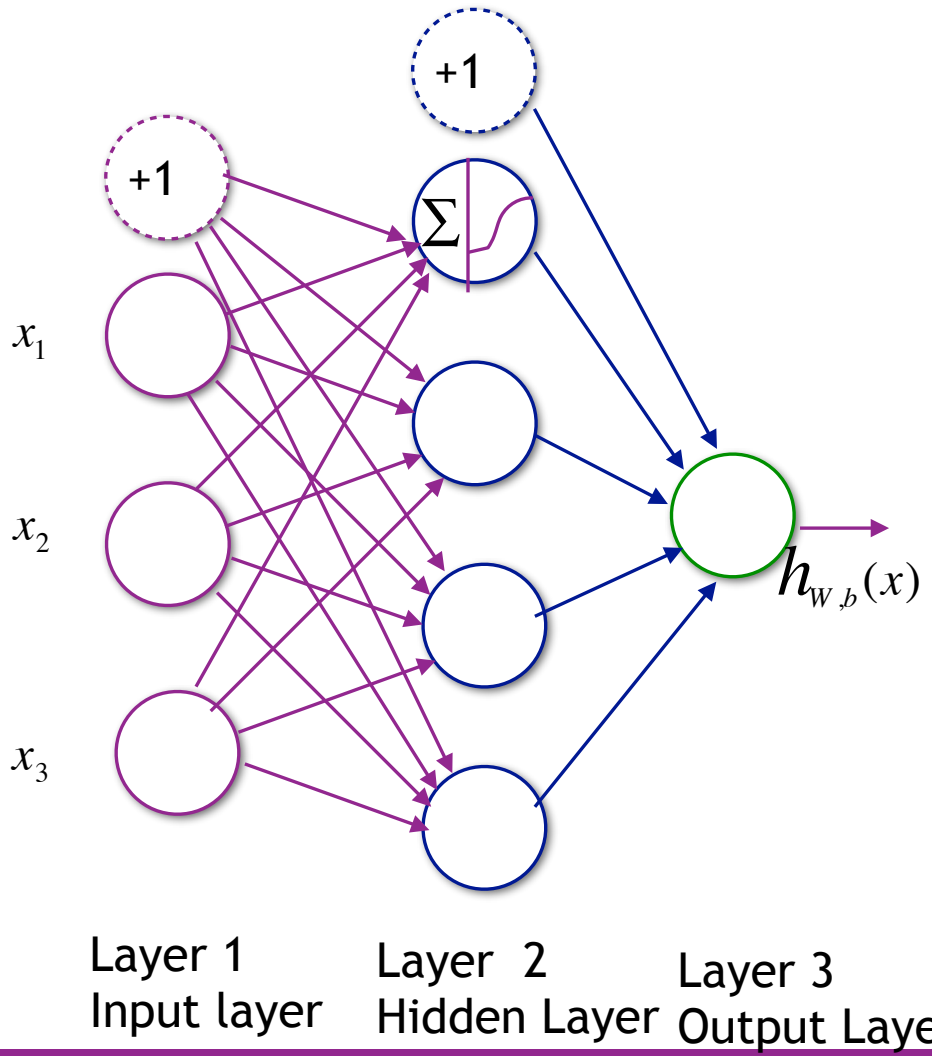
$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1 \qquad a_1^{(2)} = f(z_1^{(2)})$$

$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)} \qquad a_2^{(2)} = f(z_2^{(2)})$$

$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)} \qquad a_3^{(2)} = f(z_3^{(2)})$$

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)} \qquad a_4^{(2)} = f(z_4^{(2)})$$

Neural Network Model & Notation



- We can combine many of the simple "neurons" so the output of one is the input of another neuron
- Input layer units are set by **x**
- The input for all other nodes are typically the *weighted* sum of the **activation** on the links connected to this node
- The **activation** function is applied to the input to provide the value (we will use the sigmoid function)

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

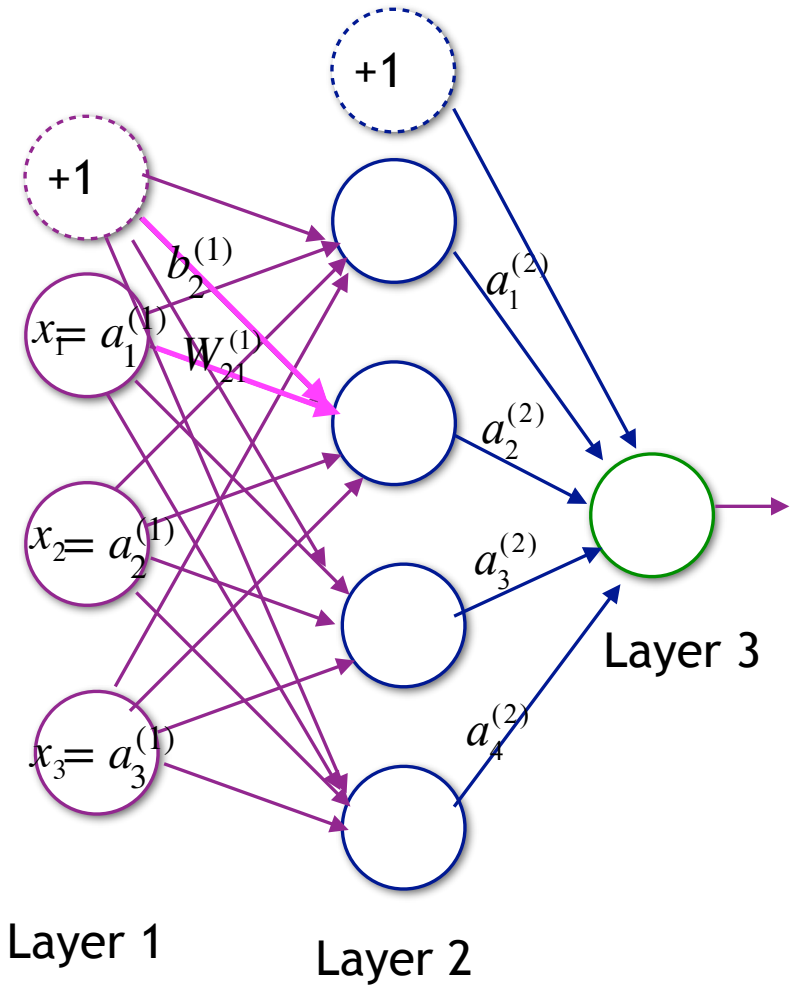
Neural Network Model & Notation

$W_{ij}^{(\ell)}$

$1 \le \ell < n_l$ layers

$1 \le j \le s_\ell$ inputs

$1 \le i \le s_{\ell+1}$ outputs



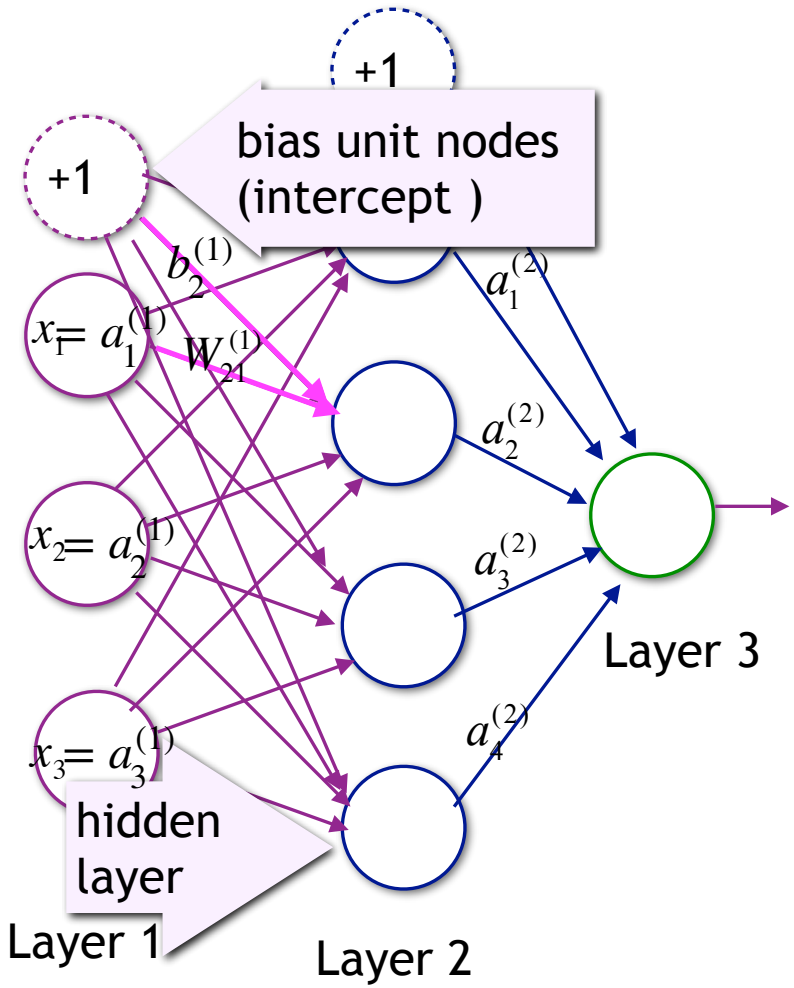
- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- n_ℓ is the number of layers in the network
- Each "neuron" has it own weights. We will store all the weights for one level together in a matrix $W^{(\ell)}$ such that $W_{ij}^{(1)}$ is the weight for leaving node j at level 1 and entering node i at level 2
- $b_i^{(1)}$ is the bias of unit i in level 2

$W^{(1)} \in \mathbb{R}^{4\times 3}$ $W^{(2)} \in \mathbb{R}^{1\times 4}$

Neural Network Model & Notation

$W_{ij}^{(\ell)}$

$1 \le \ell < n_l$ layers
$1 \le j \le s_\ell$ inputs
$1 \le i \le s_{\ell+1}$ outputs

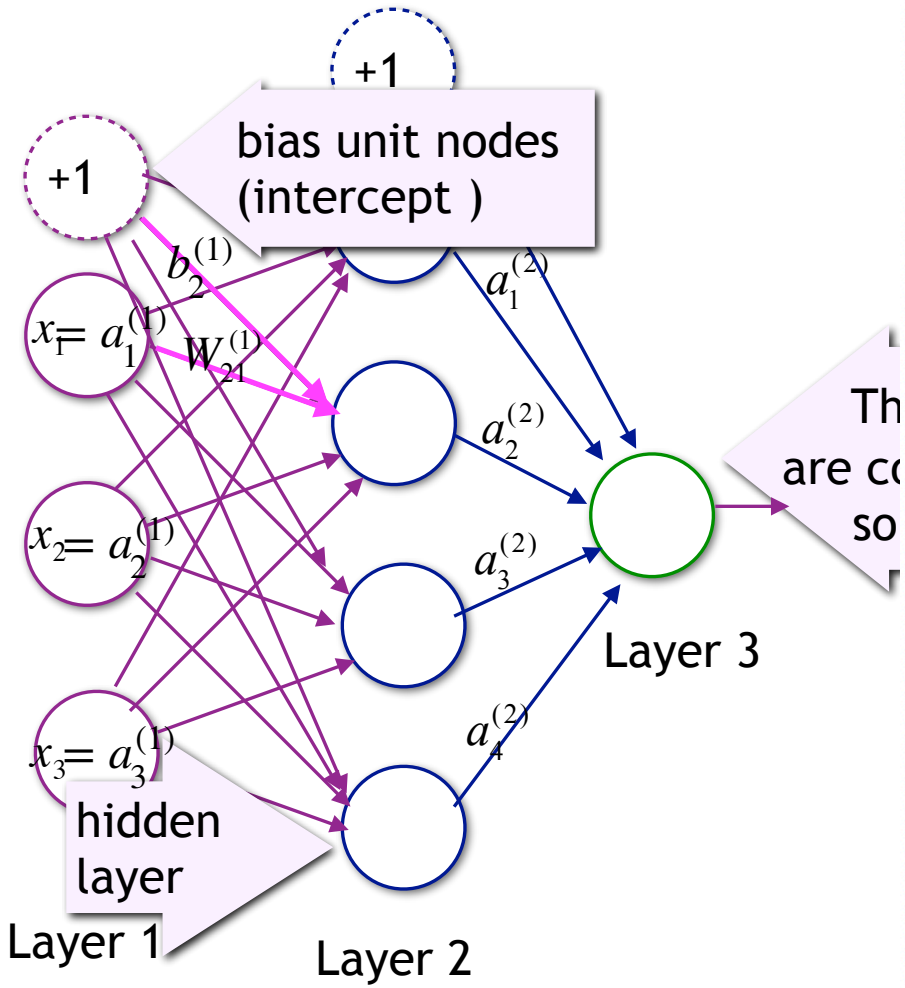


The number of layers is n_ℓ=3. We are counting the input layer - many sources do not count this layer

- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- n_ℓ is the number of layers in the network
- ts. We will together in a weight for ring node i at level 2
- $b_i^{(1)}$ is the bias of unit i in level 2

$W^{(1)} \in \mathbb{R}^{4\times 3}$ $W^{(2)} \in \mathbb{R}^{1\times 4}$

Neural Network Model & Notation



1. How many parameters to train in this neural network?

a) 0-3

b) 4-6

c) 7-10

d) 11-14

e) 15-18

f) 19-22

g) 23-26

h) more than 26

layers
inputs
outputs

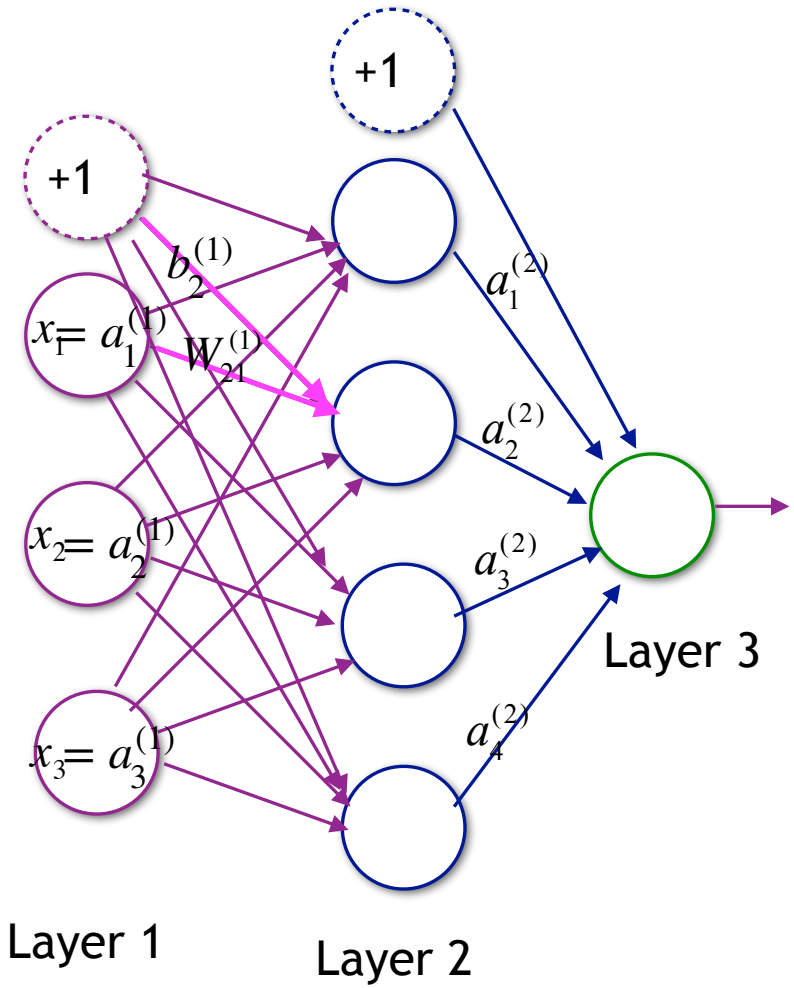
4 hidden

ork

e will
her in a
t for
ode i at

Neural Network Model & Notation

$$W_{ij}^{(\ell)} \begin{cases} 1 \le \ell < n_l & \text{layers} \\ 1 \le j \le s_\ell & \text{inputs} \\ 1 \le i \le s_{\ell+1} & \text{outputs} \end{cases}$$



input layer from input units

- The network has 3 input units (not 4), 4 hidden units (not 5) and 1 output unit
- n_ℓ is the number of layers in the network
- Each "neuron" has it own weights. We will store all the weights for one level together in a matrix $W^{(1)}$ such that $W_{ij}^{(1)}$ is the weight for leaving node j at level 1 and entering node i at level 2
- $b_i^{(1)}$ is the bias of unit i in level 2

$$W^{(1)} \in \mathbb{R}^{4\times 3} \qquad W^{(2)} \in \mathbb{R}^{1\times 4}$$

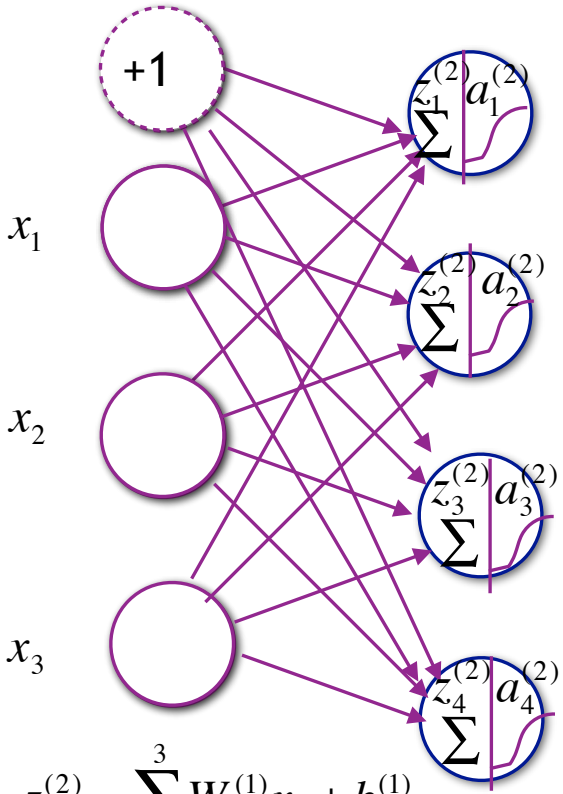
Let s_j be the number of nodes at layer j where we don't include the bias unit

Neural Network Model & Notation

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad a_i^{(2)} = f(z_i^{(2)})$$

$$z^{(2)} = W^{(1)} x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)}) = f(W^{(1)} x + b^{(1)})$$

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



$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix} = \begin{bmatrix} W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 \\ W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 \\ W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 \\ W_{41}^{(1)} x_1 + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)} \\ W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)} \\ W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)} \\ W_{41}^{(1)} x_1 + W_{42}^{(1)} x_2 + W_{43}^{(1)} x_3 + b_4^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix}$$

$$a^{(2)} = \begin{bmatrix} f(z_1^{(2)}) \\ f(z_2^{(2)}) \\ f(z_3^{(2)}) \\ f(z_4^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix}$$

$$z_1^{(2)} = \sum_{j=1}^{3} W_{1j}^{(1)} x_j + b_1^{(1)} \qquad a_1^{(2)} = f(z_1^{(2)})$$

$$z_2^{(2)} = \sum_{j=1}^{3} W_{2j}^{(1)} x_j + b_2^{(1)} \qquad a_2^{(2)} = f(z_2^{(2)})$$

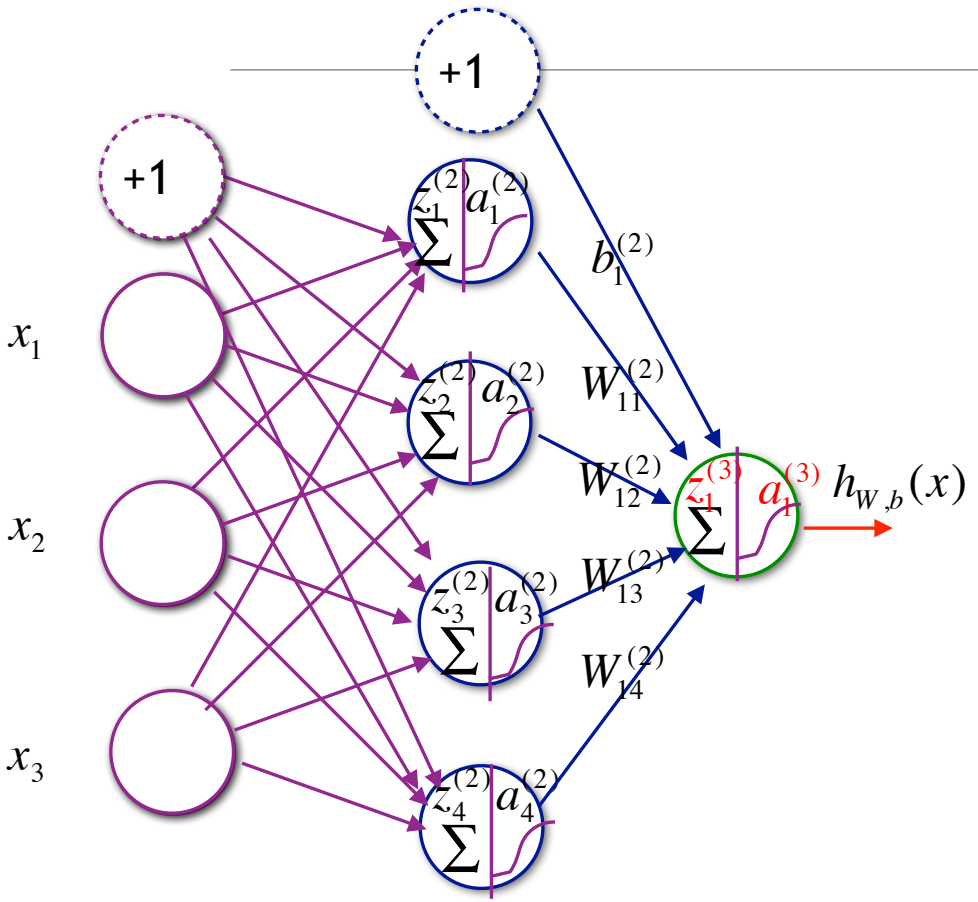
$$z_3^{(2)} = \sum_{j=1}^{3} W_{3j}^{(1)} x_j + b_3^{(1)} \qquad a_3^{(2)} = f(z_3^{(2)})$$

$$z_4^{(2)} = \sum_{j=1}^{3} W_{4j}^{(1)} x_j + b_4^{(1)} \qquad a_4^{(2)} = f(z_4^{(2)})$$

Neural Network Model & Notation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)})$$

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



$$\begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(3)} \end{bmatrix}$$

$$a^{(3)} = \begin{bmatrix} f(z_1^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(3)} \end{bmatrix}$$

$$z^{(3)} = W^{(2)} a^{(2)} + b^{(2)}$$

$$a^{(3)} = f(z^{(3)}) = f(W^{(2)} x + b^{(2)})$$

$$z_1^{(3)} = \sum_{j=1}^{4} W_{1j}^{(2)} a_j^{(2)} + b_1^{(2)}$$

$$a_1^{(3)} = f(z_1^{(3)})$$

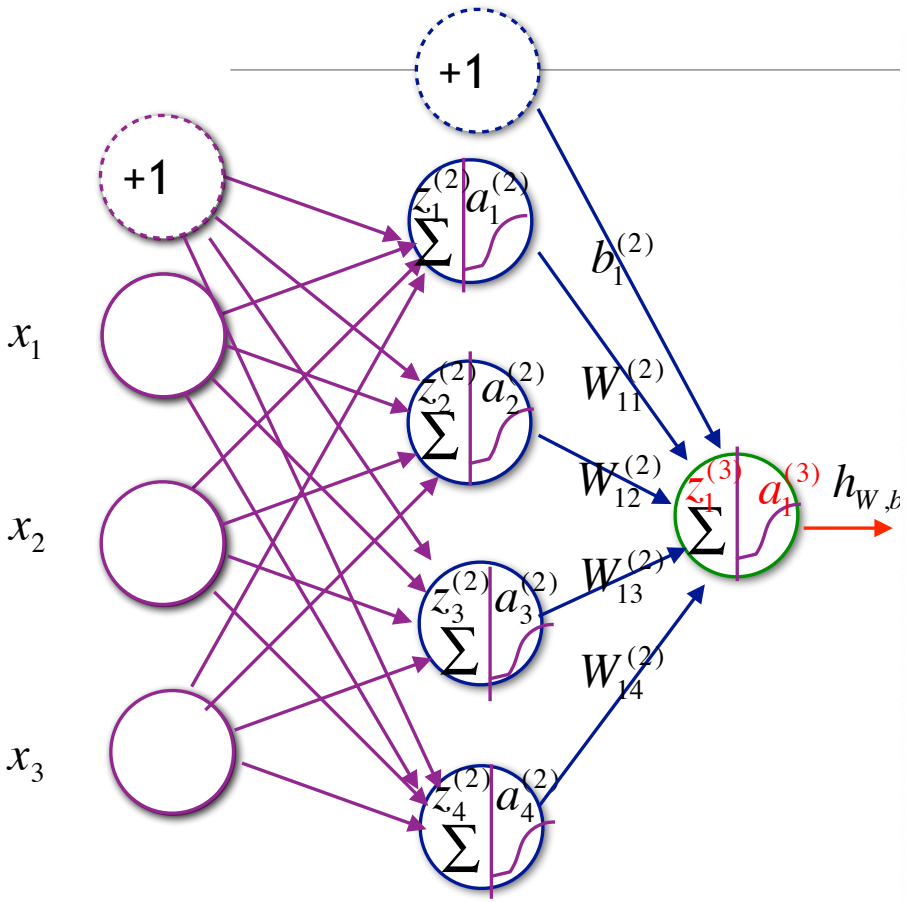
$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

Neural Network Model & Notation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$

$$a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)})$$

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



What is $a_3^{(2)}$?

a) One of the inputs to neurons on layer 2

b) One of the inputs to neurons on layer 3

c) activation value of 3rd neuron on layer 2

d) activation value of 2nd neuron on layer 3

e) activation value of 3rd neuron on layer 3

f) activation value of 2nd neuron on layer 3

$$+\left[b_1^{(2)}\right] = \left[z_1^{(3)}\right]$$

$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

Artificial Neural Networks

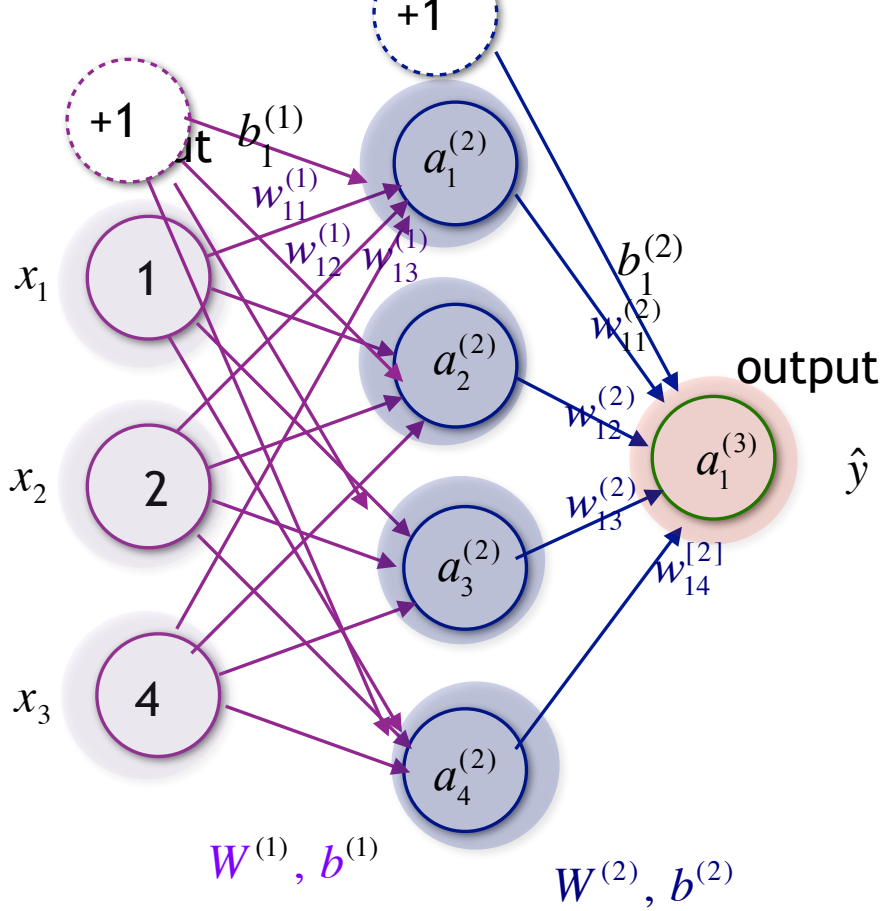
- Each node (we call the nodes ***neurons***) represents a linear function of its input, similar to logistic regression

$$z_1^{(2)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}$$
$$a_1^{(2)} = f(z_1^{(2)})$$

  - $W_{11}^{(1)}\ W_{12}^{(1)}\ W_{13}^{(1)}\ b_1^{(1)}$ are the ***weights*** (aka parameters)
  - $W^{(1)}$ is the matrix of weights that map values from input layer 1 to the first hidden layer (layer 2) ($W^{(\ell)}$ is the matrix that maps layer ℓ to layer ℓ+1)
  - $b^{(1)}$ is the *vector* of bias values for the neurons on layer 1

- $f^{(1)}$ is the "activation function" for layer 1. ($f^{(\ell)}$ is the activation function for layer ℓ)
- $a_j^{(2)}$ is the activation value for the j^{th} neuron of layer 2
  $a_j^{(\ell)}$ is the activation value for the j^{th} neuron of layer the ℓ

Let s_j is the number of nodes at layer j where we don't include the bias unit



$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \quad b^{(2)} = \begin{bmatrix} b_1^{(2)} \end{bmatrix}$$

Artificial Neural Networks

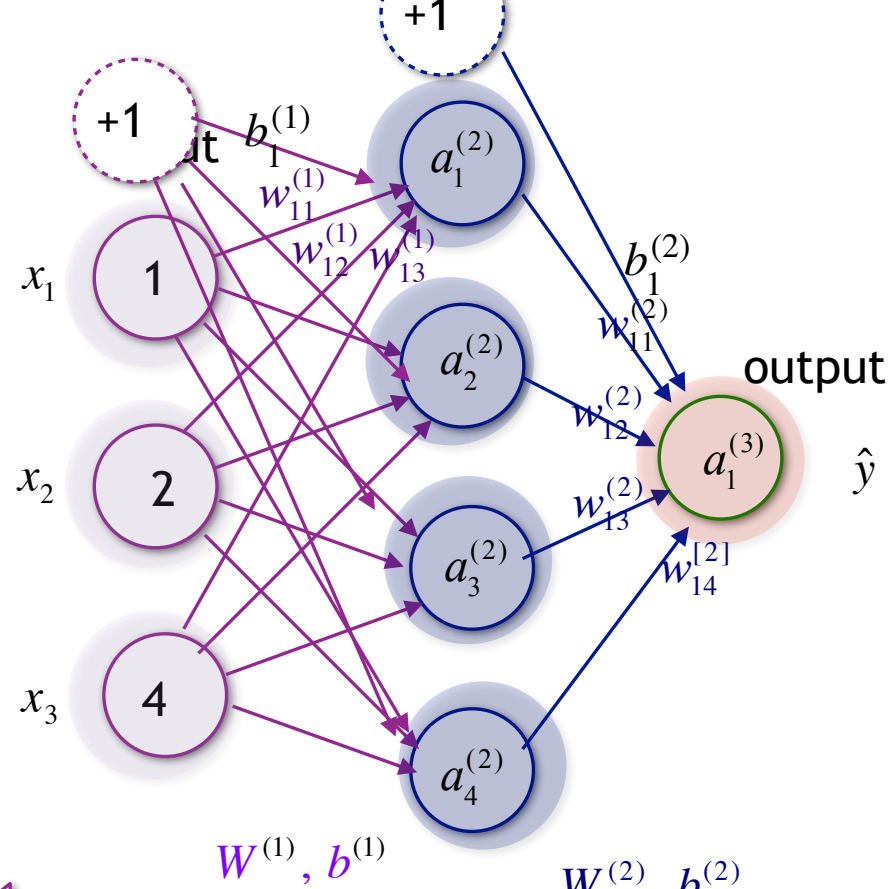
- Each node (we call the nodes ***neurons***) represents a linear function of its input, similar to logistic regression

$$z_1^{(2)} = W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}$$
$$a_1^{(2)} = f(z_1^{(2)})$$

  - $W_{11}^{(1)}\ W_{12}^{(1)}\ W_{13}^{(1)}\ b_1^{(1)}$ are the ***weights*** (aka parameters)
  - $W^{(1)}$ is the matrix of weights that map values from input layer 1 to the first hidden layer (layer 2) ($W^{(\ell)}$ is the matrix that maps layer ℓ to layer ℓ+1)
  - $b^{(1)}$ is the *vector* of bias values for the neurons on layer 1

- $f^{(1)}$ is the "activation function" for layer 1. ($f^{(\ell)}$ is the activation function for layer ℓ)
- $a_j^{(2)}$ is the activation value for the j^{th} neuron of layer 2
  $a_j^{(\ell)}$ is the activation value for the j^{th} neuron of layer the ℓ

Let s_j is the number of nodes at layer j where we don't include the bias unit



One example of an activation function is the logistic sigmoid

$$W = \begin{bmatrix} W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \qquad b = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \qquad b^{(2)} = \begin{bmatrix} b_1^{(2)} \end{bmatrix}$$

Outline

- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

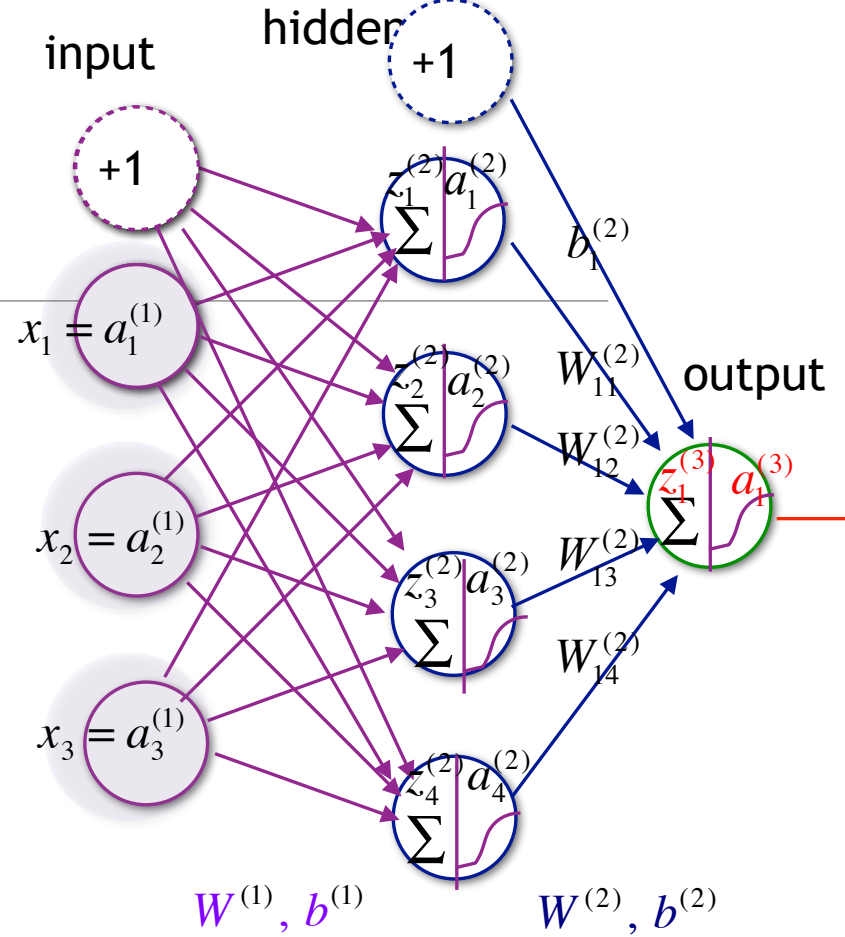
$f(z) = \frac{1}{1+\exp(-z)}$

Forward Propagation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)})$$

- Forward propagation:

$a^{(1)} = \mathbf{x}$
for ℓ = 1 to n_ℓ-1 do
$z^{(\ell+1)} = W^{(\ell)} a^{(\ell)} + b^{(\ell)}$
$a^{(\ell+1)} = f(z^{(\ell+1)})$
$\hat{y} = a^{(n_\ell)}$



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

Forward Propagation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)})$$

- Generalizing for an arbitrary neural network. Forward propagation:

$a^{(1)} = \mathbf{x}$

for ℓ = 1 to n_ℓ-1 do

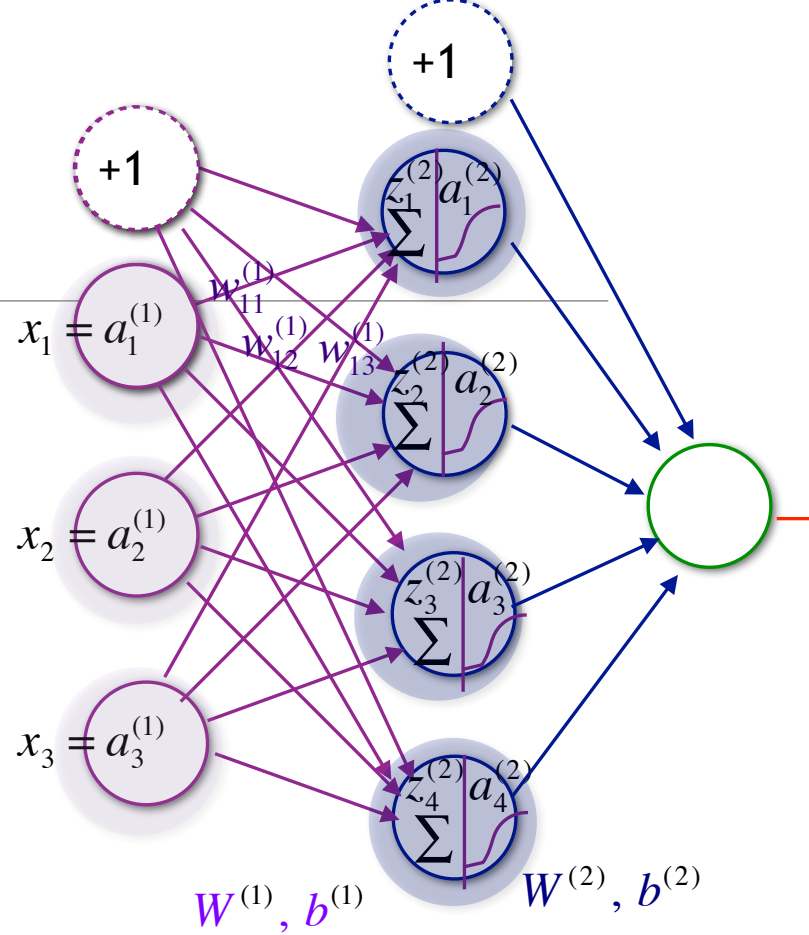
$z^{(\ell+1)} = W^{(\ell)} a^{(\ell)} + b^{(\ell)} = z^{(2)} =$

$$\begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} & W_{33}^{(1)} \\ W_{41}^{(1)} & W_{42}^{(1)} & W_{43}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \\ z_4^{(2)} \end{bmatrix}$$

$a^{(\ell+1)} = f(z^{(\ell+1)})$

$\hat{y} = a^{(n_\ell)}$

$$a^{(2)} = \begin{bmatrix} f(z_1^{(2)}) \\ f(z_2^{(2)}) \\ f(z_3^{(2)}) \\ f(z_4^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix}$$



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

Forward Propagation

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \qquad a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)})$$

- Generalizing for an arbitrary neural network. Forward propagation:

$a^{(1)} = \mathbf{x}$

for $\ell = 1$ to $n_\ell - 1$ do

$z^{(\ell+1)} = W^{(\ell)} a^{(\ell)} + b^{(\ell)}$

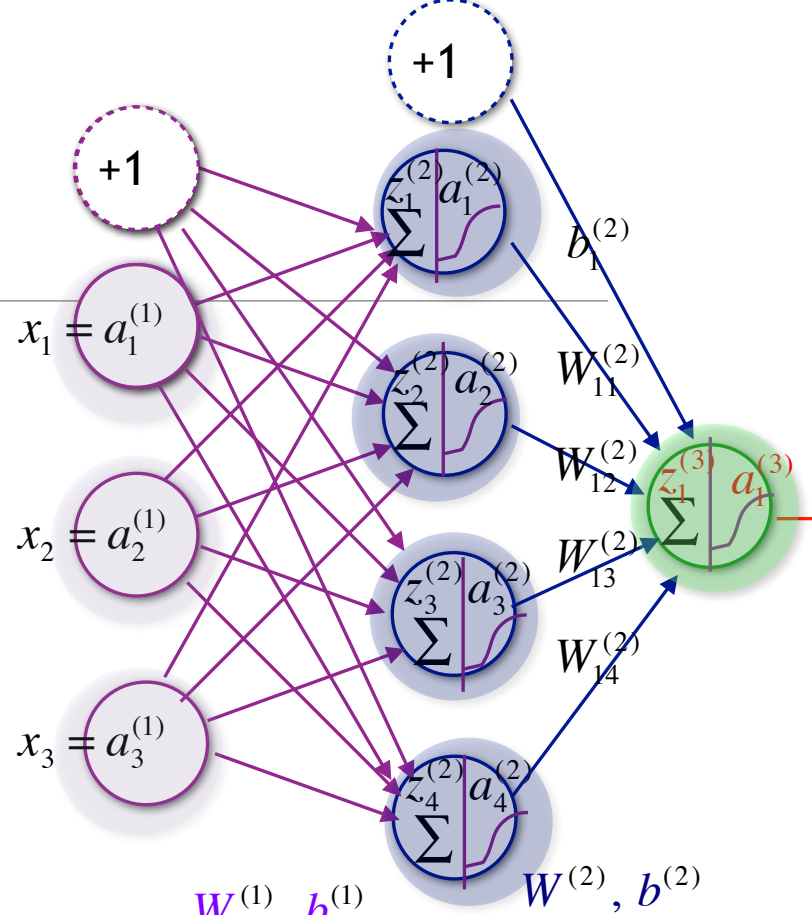
$a^{(\ell+1)} = f(z^{(\ell+1)})$

$\hat{y} = a^{(n_\ell)}$

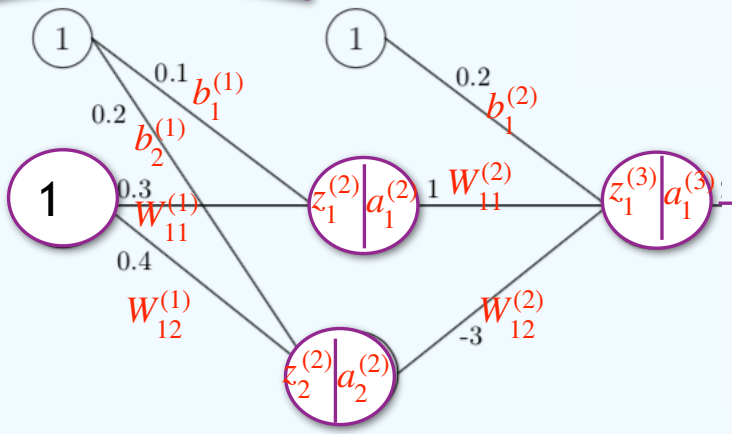
$$z^{(3)} = \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} & W_{13}^{(2)} & W_{14}^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(3)} \end{bmatrix}$$

$$a^{(3)} = \begin{bmatrix} f(z_1^{(2)}) \end{bmatrix} = \begin{bmatrix} a_1^{(3)} \end{bmatrix}$$

$$a^{(3)} = f(W^{(2)} f(W^{(1)} a^{(1)} + b^{(1)}) + b^{(2)})$$



Calculations done with rounded numbers...



$$h(\mathbf{x}) = \hat{y}_1 = a_1^{(3)}$$

$$\frac{1}{2}(y - \hat{y}_1)^2 = \frac{1}{2}(y - a_1^{(3)})^2$$

$$W^{(1)} = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix} \qquad W^{(2)} = \begin{bmatrix} 1 & -3 \end{bmatrix}$$

$$b^{(1)} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \qquad b^{(2)} = [0.2]$$

If x = 1 and y =1, forward propagation:

$$x = a^{(1)} = [1] \qquad z^{(2)} = \underbrace{\begin{bmatrix} 1*0.3+0.1 \\ 1*0.4+0.2 \end{bmatrix}}_{W^{(1)}a^{(1)}+b^{(1)}} \qquad a^{(2)} = \underbrace{\begin{bmatrix} 0.60 \\ 0.65 \end{bmatrix}}_{f(z^{(2)})} \qquad z^{(3)} = \underbrace{[1*0.60+(-3)*0.65+0.2]}_{W^{(2)}a^{(2)}+b^{(2)}} \qquad a^{(3)} = \underbrace{[\sigma(-1.15)]}_{f(z^{(3)})} = [0.24]$$

$$\underbrace{f(W^{(1)}a^{(1)}+b^{(1)})}_{f(W^{(2)}a^{(2)}+b^{(2)})}$$

$$f\left(W^{(2)}f\left(W^{(1)}a^{(1)}+b^{(1)}\right)+b^{(2)}\right)$$

$$J(W,b,\mathbf{x},y) = \frac{1}{2}(y - a_1^{(3)})^2 = \frac{1}{2}\left(y - f\left(W^{(2)}f(W^{(1)}a^{(1)}+b^{(1)})+b^{(2)}\right)\right)^2$$

Graph similar to graph in Learning from Data

What if we didn't use an activation function?

- Linear model… $z^{(3)} = a^{(3)} = \cancel{\sigma}(W^{(2)}\cancel{\sigma}(W^{(1)}a^{(1)} + b^{(1)}) + b^{(2)})$

To simplify the notation: Assume $b^{(i)} = 0$

$$z^{(2)} = W^{(1)}\mathbf{x}$$

$$z^{(3)} = W^{(2)}z^{(2)} = W^{(2)}W^{(1)}\mathbf{x}$$

If we added another layer

$$z^{(4)} = a^{(4)} = \cancel{\sigma}(W^{(3)}\cancel{\sigma}(W^{(2)}\cancel{\sigma}(W^{(1)}a^{(1)} + b^{(1)}) + b^{(2)}) + b^{(3)})$$

$$\tilde{W} = W^{(3)}W^{(2)}W^{(1)}$$

$$z^{(4)} = W^{(3)}a^{(3)} = W^{(3)}W^{(2)}a^{(2)} = W^{(3)}W^{(2)}W^{(1)}\mathbf{x} = \tilde{W}\mathbf{x}$$

Outline

- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

http://www.offconvex.org/2016/12/20/backprop/

How do we train a neural network?

We will use the values we computed in the forward pass

Alternatively, choose a different activation function on the last layer. We are keeping things simple in this lecture.

ERROR BACK-PROPAGATION:

FIRST WE DO A FORWARD PROPAGATION AND COMPUTE THE z AND a VALUES THEN WE WORK BACKWARD AND COMPUTE THE GRADIENT OF THE LOSS WITH RESPECT TO THE WEIGHTS

SINCE ARE ARE USING THE SIGMOID ACTIVATION ON THE LAST LAYER, PREPROCESS THE TARGET VALUES TO BE IN [0,1]

Error/loss function

This lecture:

Squared error for single output: $J(W,b,\boldsymbol{x},y)=\frac{1}{2}(y-\hat{y})^2 \quad =\frac{1}{2}(y-a^{(n_\ell)})^2$

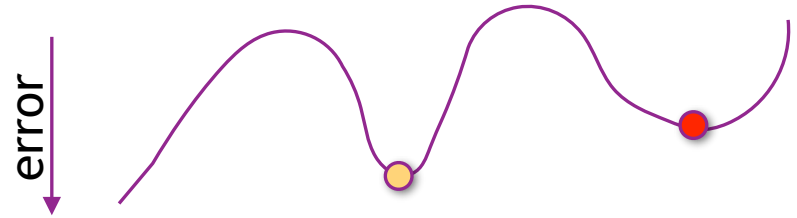
Many Alternatives, e.g:

Squared error for multiple outputs $J(W,b,\mathbf{x},y)=\frac{1}{2}\sum_{k\in K}(y_k-\hat{y}_k)^2=\frac{1}{2}\sum_{k\in K}(y_k-a_k^{(n_\ell)})^2$

Cross entropy cost for single output: $J(W,b,\mathbf{x},y)=-y\log(\hat{y})-(1-y)\log(1-\hat{y})$

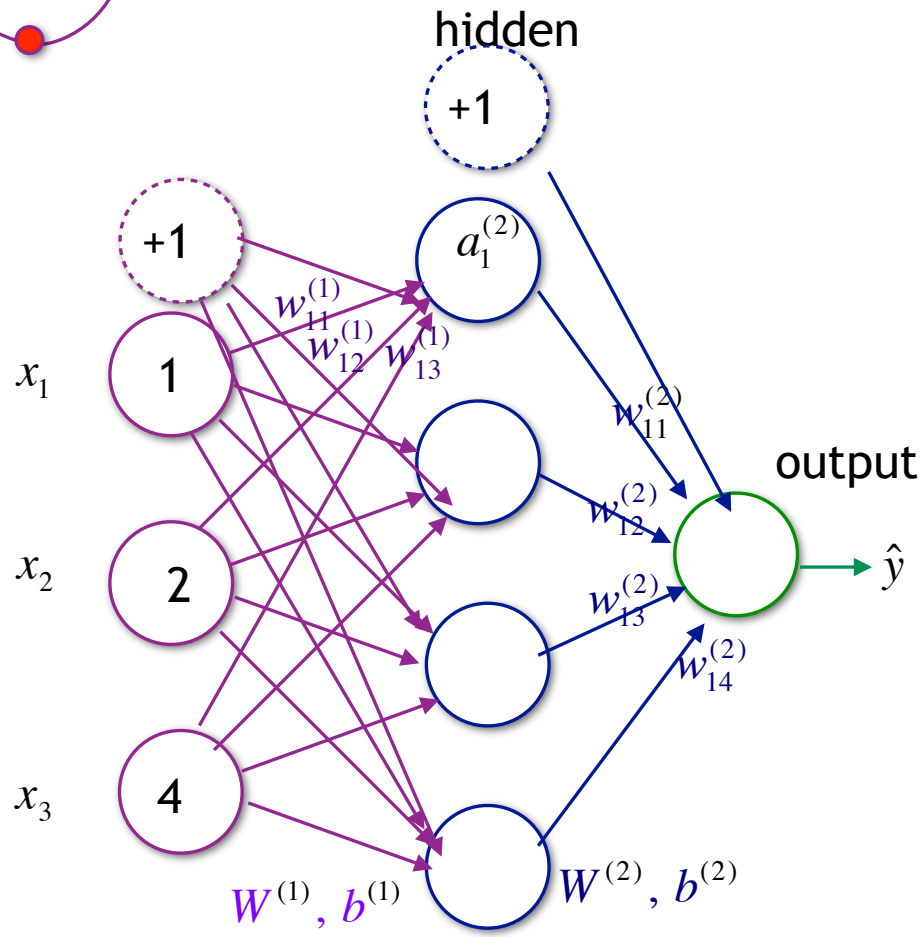
Cross entropy cost for multiple outputs: $J(W,b,\mathbf{x},y)=-\sum_{k\in K}y_k\log(\hat{y}_k)$

Problems!

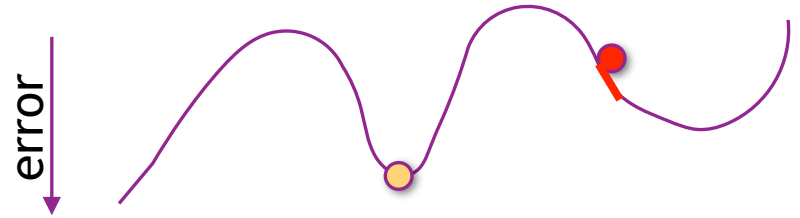


- We would like to use gradient descent to update the weights…
- Gradient descent worked before because we had a **convex** function… we no longer can assume our function is convex

- Solutions?
  - We can find the partial derivative for one level when the other levels are fixed
  - We can try different initial weights and retrain with different initial configurations

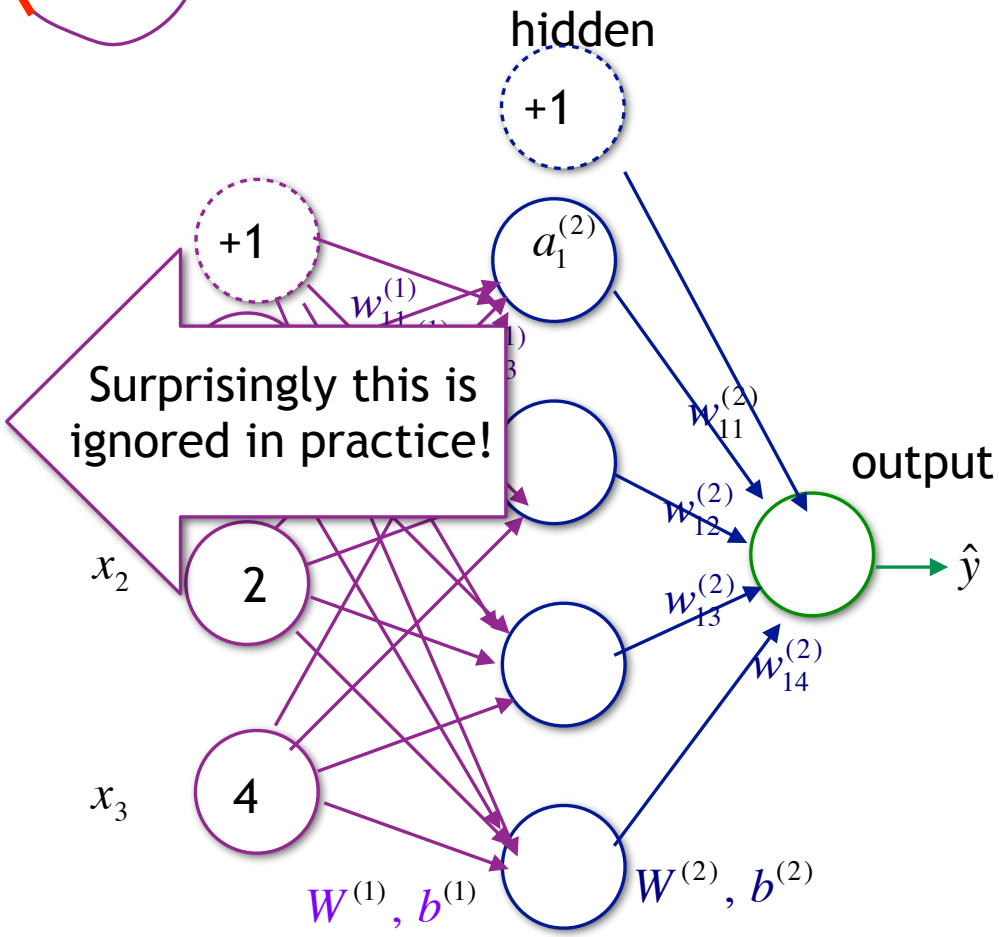


Problems!



- We would like to use gradient descent to update the weights…
- Gradient descent worked before because we had a **convex** function… we no longer can assume our function is convex

- Solutions?
  - We can find the partial derivative for one level when the other levels are fixed
  - We can try different initial weights and retrain with different initial configurations



Training a neural network

Local optimization: similar to logistic regression and linear regression

Our approach:

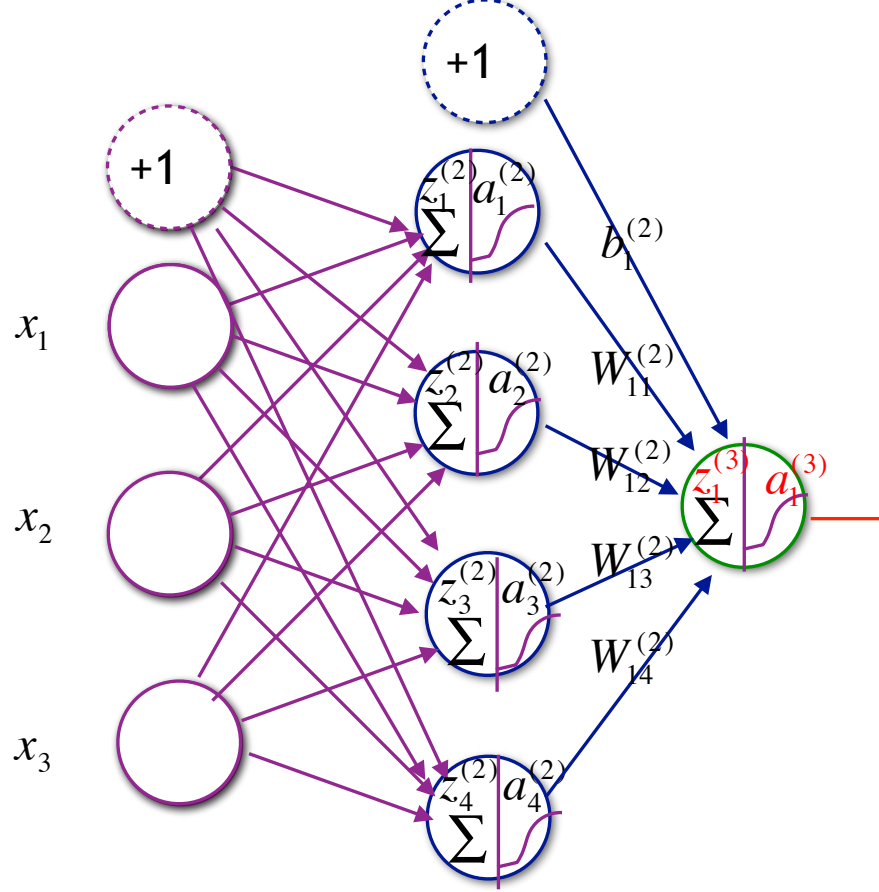
- First create a ***stochastic algorithm*** where we update the weights after observing only *one training example*
  - If the neural network correctly identifies (predicts) the example - no changes are made to the weights
  - If the neural network incorrectly identifies (predicts) the example - the weights are updated the reduce the error

How do you think we can update the weights?

$W_{new} = W_{old} - \alpha \nabla \text{error}$

$$W_{ij}^{(\ell)} = W_{ij}^{(\ell)} - \alpha \frac{\partial J(W, \mathbf{b}, \mathbf{x}, y)}{\partial W_{ij}^{(\ell)}}$$

$$b_i^{(\ell)} = b_i^{(\ell)} - \alpha \frac{\partial J(W, \mathbf{b}, \mathbf{x}, y)}{\partial b_i^{(\ell)}}$$



Gradient descent algorithm depends on the error function and the structure of the network

- Next we turn our stochastic algorithm into batch gradient descent

Stochastic gradient descent algorithm

Randomly initialize the biases and weights for each layer: $b^{(\ell)}\ W^{(\ell)}$

While iterations < iteration limit:

Randomly choose a training example: $(\mathbf{x}, y)$

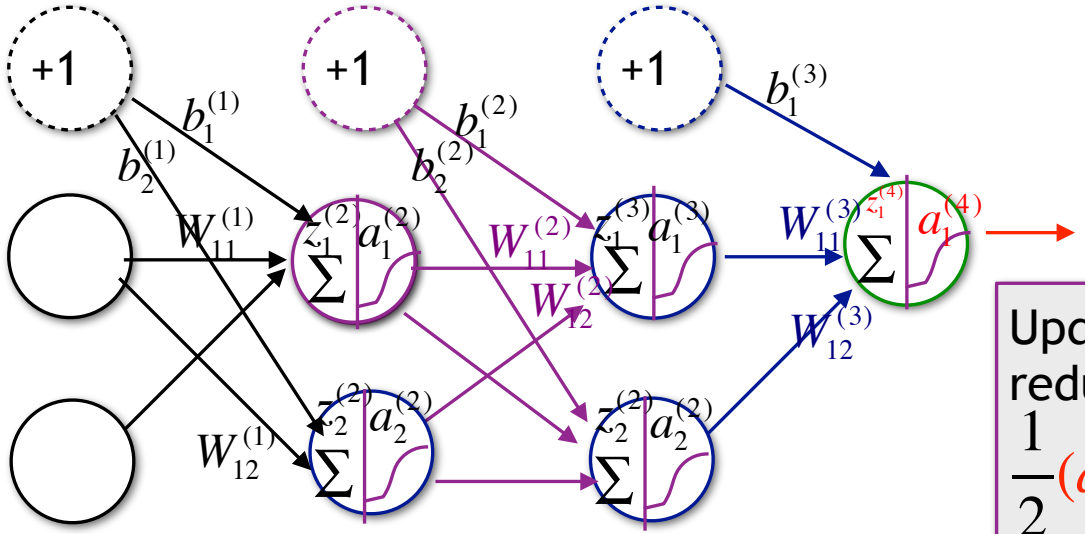
Run forward propagation and save for each level, $a^{(\ell)}$ and $z^{(\ell)}$

Run back propagation to compute the partial derivatives: $\nabla_{b^{(\ell)}} J(W, b, \mathbf{x}, y),\ \nabla_{W^{(\ell)}} J(W, b, \mathbf{x}, y)$

Perform a gradient descent step:

$$W^{(\ell)} = W^{(\ell)} - \alpha \nabla_{W^{(\ell)}} J(W, b, \mathbf{x}, y)$$

$$b^{(\ell)} = b^{(\ell)} - \alpha \nabla_{b^{(\ell)}} J(W, b, \mathbf{x}, y)$$



Update weight to reduce error:

$$\frac{1}{2}(a_1^{(4)} - y)^2$$

Another way to compute a partial derivative

https://colah.github.io/posts/2015-08-Backprop/

Computation graphs

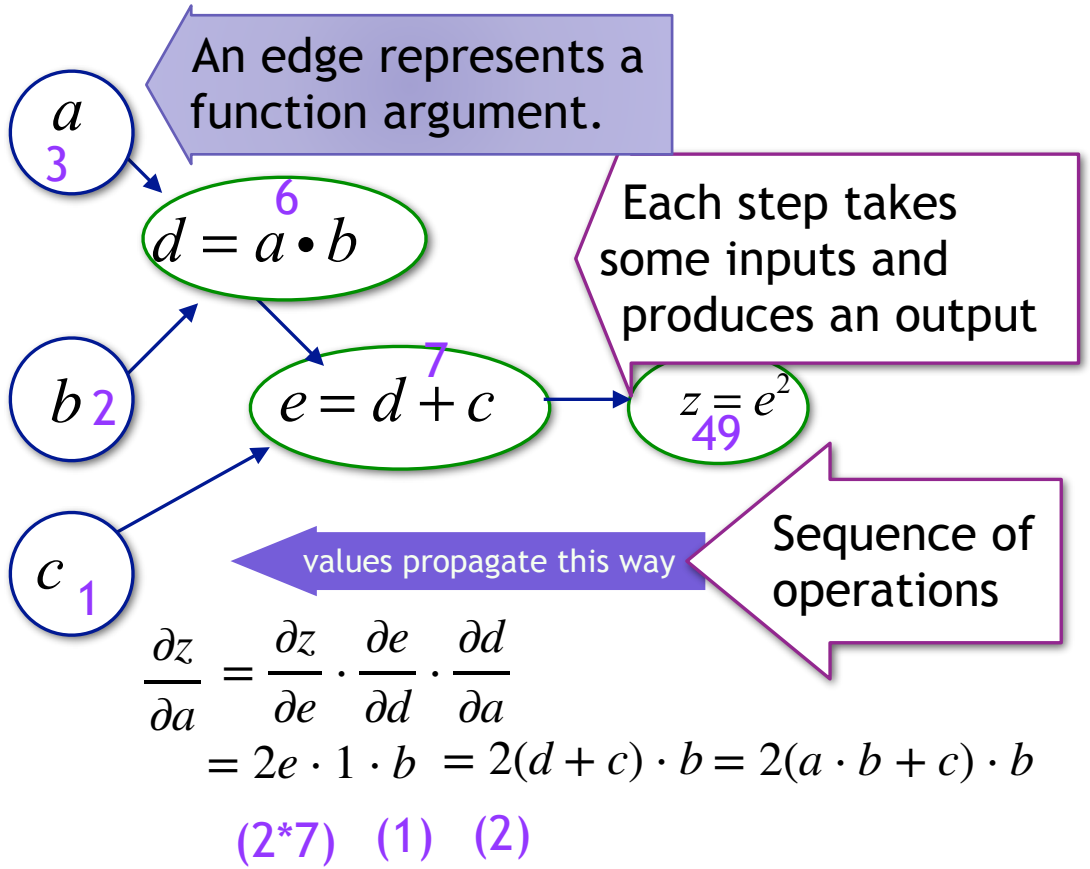
Represents a function as a graph

The derivative of $f(g(h(x)))$ is $f'(g(h(x)))g'(h(x))h'(x)$

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx}$$

- Directed acyclic graphs that show the dependency structure of the computation to be performed
- Computation graph breaks the function into steps we can compute the derivative
- Nodes
  - input values
  - functions for combining values
- Example: $(ab+c)^2$
  - input values: a,b,c
  - functions for combining values:

$$d = a \cdot b \qquad e = d + c \qquad z = e^2$$



$$\frac{\partial z}{\partial a} = \frac{\partial z}{\partial e} \cdot \frac{\partial e}{\partial d} \cdot \frac{\partial d}{\partial a}$$

$$= 2e \cdot 1 \cdot b \quad = 2(d+c) \cdot b = 2(a \cdot b + c) \cdot b$$

(2*7) (1) (2)

https://colah.github.io/posts/2015-08-Backprop/

Computation graphs

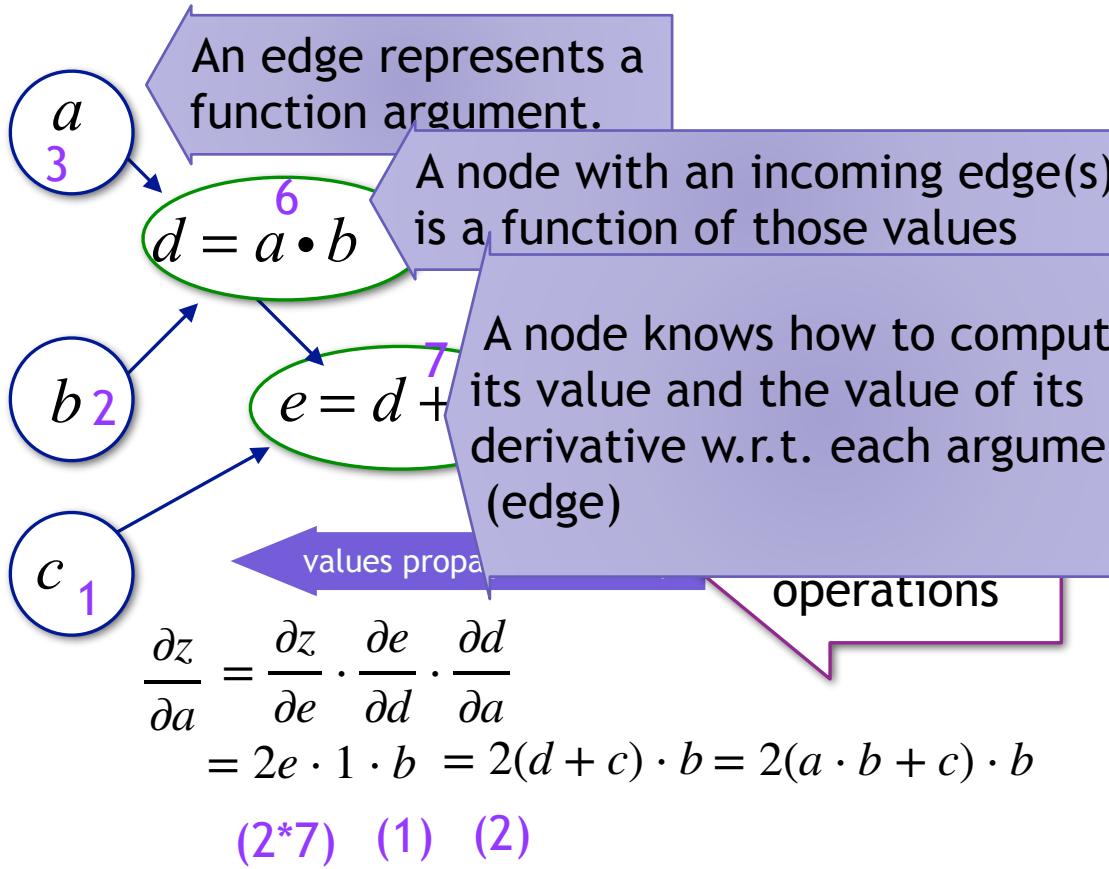
Represents a function as a graph

The derivative of $f(g(h(x)))$ is $f'(g(h(x)))g'(h(x))h'(x)$

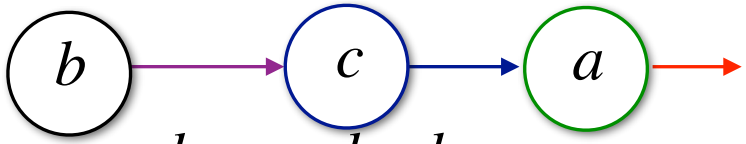
$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx}$$

- Directed acyclic graphs that show the dependency structure of the computation to be performed
- Computation graph breaks the function into steps we can compute the derivative
- Nodes
  - input values
  - functions for combining values
- Example: $(ab+c)^2$
  - input values: a,b,c
  - functions for combining values:

$$d = a \cdot b \qquad e = d + c \qquad z = e^2$$



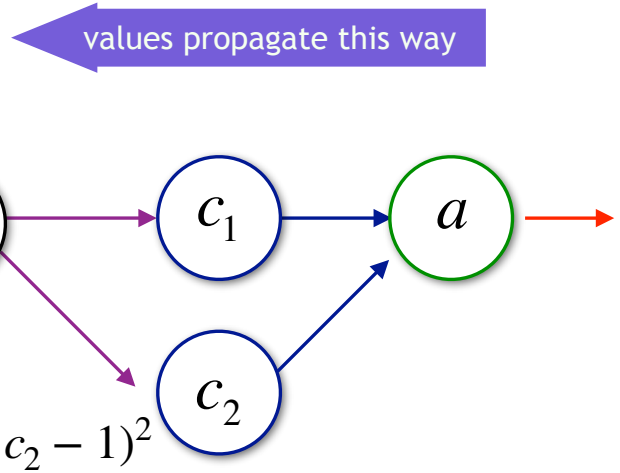
Basic Facts:



$$a = (c-1)^2$$

$$c = (qb+1)$$

$$\frac{da}{db} = \frac{da}{dc}\frac{dc}{db} = 2(c-1)q$$



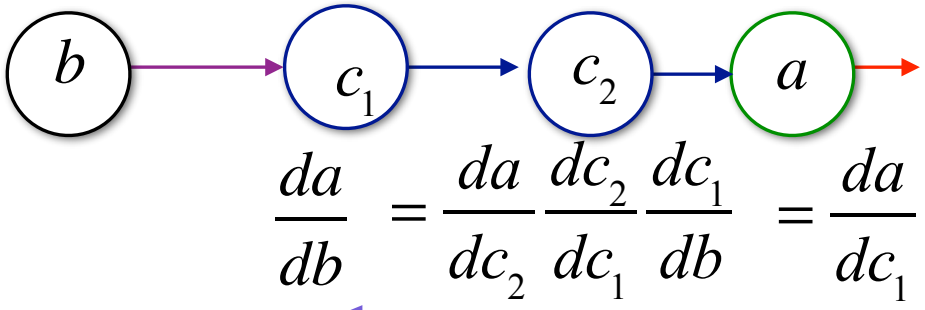
$$a = (c_1 + c_2 - 1)^2$$

$$c_1 = (q_1 b + 1)$$

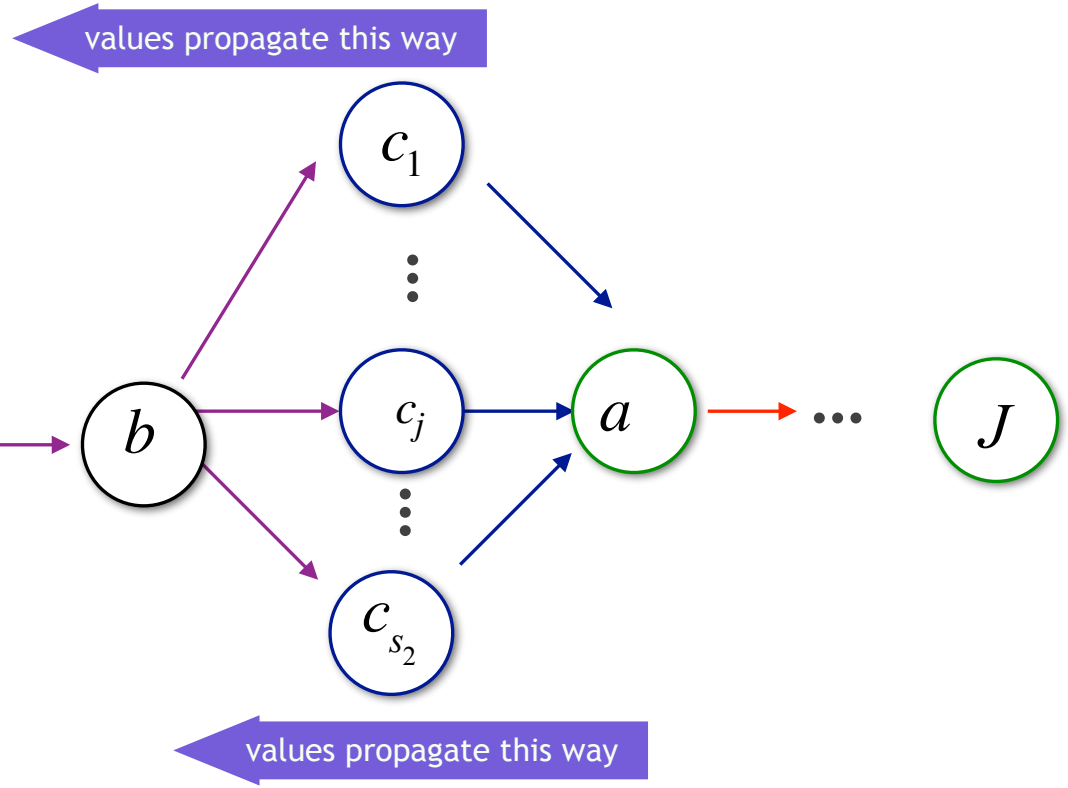
$$c_2 = (q_2 b + 1)$$

$$\frac{da}{db} = \frac{da}{dc_1}\frac{dc_1}{db} + \frac{da}{dc_2}\frac{dc_2}{db}$$

$$= 2(c_1 + c_2 - 1)\ q_1 + 2(c_1 + c_2 - 1)\ q_2$$



$$\frac{da}{db} = \frac{da}{dc_2}\frac{dc_2}{dc_1}\frac{dc_1}{db} = \frac{da}{dc_1}\frac{dc_1}{db}$$



$$\frac{da}{db} = \sum_{i=1}^{s_2} \frac{da}{dc_i}\frac{dc_i}{db}$$

$$\frac{dJ}{db} = \sum_{i=1}^{s_2} \frac{dJ}{dc_i}\frac{dc_i}{db}$$

$$\frac{dJ}{dw} = \frac{dJ}{db}\frac{db}{dw}$$

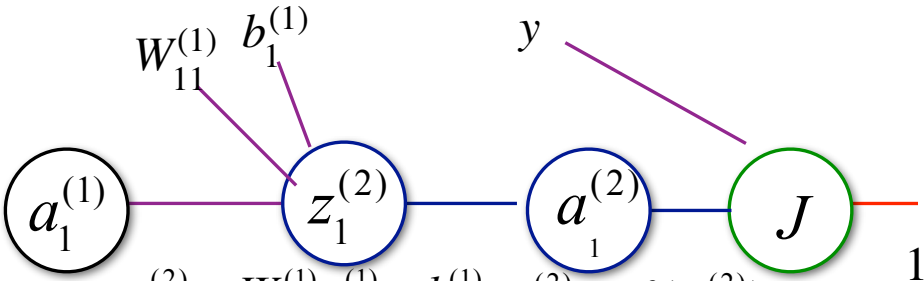
Partial derivatives

$$J = \frac{1}{2}(f(\underbrace{W_{11}^{(1)}a_1^{(1)} + b_1^{(1)}}_{z_1^{(2)}}) - y)^2$$

$$= \frac{1}{2}(f(z_1^{(2)}) - y)^2 = \frac{1}{2}(a_1^{(2)} - y)^2$$

The derivative of $f(g(h(x)))$ is $f'(g(h(x)))g'(h(x))h'(x)$

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx}$$



$$z_1^{(2)} = W_{11}^{(1)}a_1^{(1)} + b_1^{(1)} \quad a_1^{(2)} = f(z_1^{(2)}) \quad J = \frac{1}{2}(a_1^{(2)} - y)^2$$

$$\frac{\partial J}{\partial b_1^{(1)}} = \frac{\partial J}{\partial a_1^{(2)}}\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial b_1^{(1)}} = (a_1^{(2)} - y)\frac{\partial a^{(2)}}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial b_1^{(1)}}$$

values propagate this way

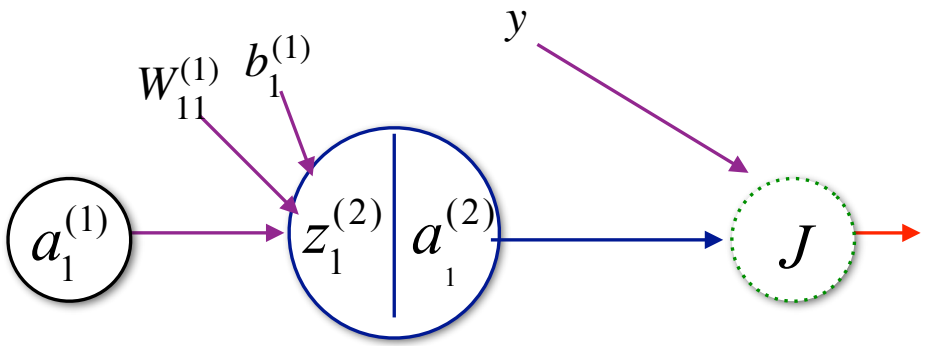
$$= (a_1^{(2)} - y)\frac{\partial f(z^{(2)})}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial b_1^{(1)}}$$

If $f(\cdot)$ is the logistic function $\frac{\partial f(z)}{\partial z} = f(z)(1 - f(z))$

$$= (a_1^{(2)} - y)f'(z^{(2)})\frac{\partial z_1^{(2)}}{\partial b_1^{(1)}}$$

$$= (\underbrace{a_1^{(2)} - y)f'(z^{(2)})}_{} \cdot 1$$

$$\frac{\partial J}{\partial W_{11}^{(1)}} = \frac{\partial J}{\partial a_1^{(2)}}\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} = \frac{\partial J}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} = (a_1^{(2)} - y)f'(z^{(2)}) \cdot a_1^{(1)}$$



$$a_1^{(2)} = f(z_1^{(2)}) \qquad J = \frac{1}{2}(a_1^{(2)} - y)^2$$

$$z_1^{(2)} = W_{11}^{(1)}a_1^{(1)} + b_1^{(1)}$$

$$\frac{dJ}{db_1^{(1)}} = \frac{dJ}{da_1^{(2)}}\frac{da_1^{(2)}}{dz_1^{(2)}}\frac{dz_1^{(2)}}{db_1^{(1)}}$$

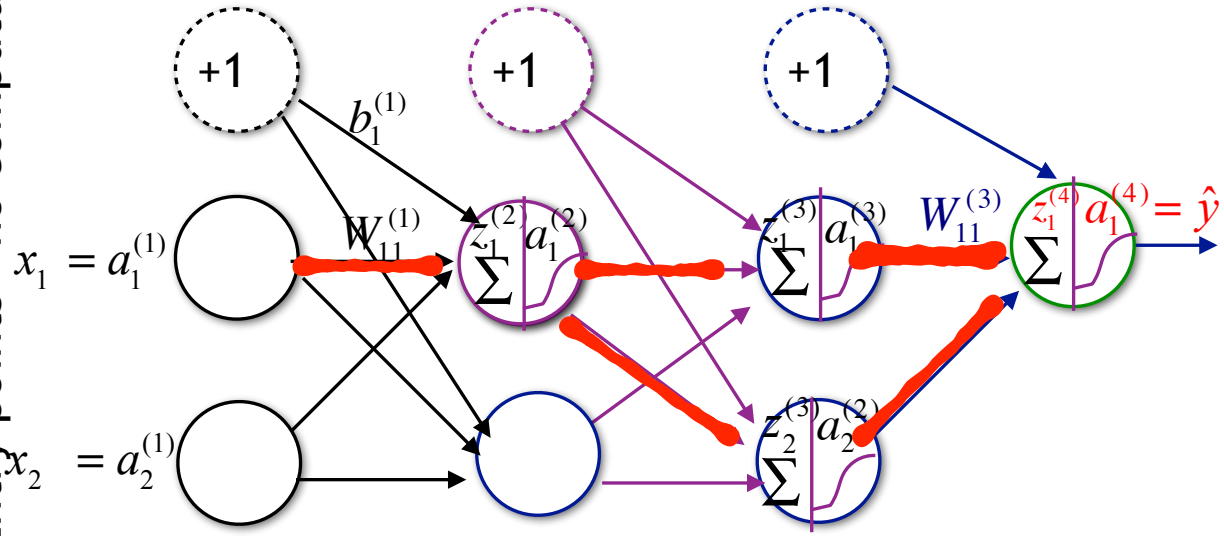
values propagate this way

$$\frac{\partial J}{\partial b_1^{(1)}} = (a_1^{(2)} - y)\ f'(z_1^{(2)})\ 1$$

It is a mess

For this lecture, all activations are: $f(z) = \frac{1}{1+e^{-z}}$

Entry points - no computation



Chain Rule

$$\frac{\partial J}{\partial W_{11}^{(1)}} = \frac{\partial J}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} = \frac{\partial J}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} + \frac{\partial J}{\partial z_2^{(3)}} \frac{\partial z_2^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}}$$

$$= \frac{\partial J}{\partial z_1^{(4)}} \frac{\partial z_1^{(4)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} + \frac{\partial J}{\partial z_1^{(4)}} \frac{\partial z_1^{(4)}}{\partial z_2^{(3)}} \frac{\partial z_2^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} = -(y - f(z_1^{(4)}))f'(z_1^{(4)}) \frac{\partial z_1^{(4)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}} + -(y - f(z_1^{(4)}))f'(z_1^{(4)}) \frac{\partial z_1^{(4)}}{\partial z_2^{(3)}} \frac{\partial z_2^{(3)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial W_{11}^{(1)}}$$

Its complicated and we will end up recomputing the same values again and again

Can we find a pattern to make this much easier?

How to update the weights...

Our update rule:

$W_{new} = W_{old} - \alpha \nabla \text{error}$

$$W_{ij}^{(\ell)} = W_{ij}^{(\ell)} - \alpha \frac{\partial J(W,b,\mathbf{x},y)}{\partial W_{ij}^{(\ell)}}$$

$$b_i^{(\ell)} = b_i^{(\ell)} - \alpha \frac{\partial J(W,b,\mathbf{x},y)}{\partial b_i^{(\ell)}}$$

$$\frac{\partial J}{\partial W_{11}^{(5)}} = \frac{\partial J}{\partial z_1^{(6)}} \frac{\partial z_1^{(6)}}{\partial W_{11}^{(5)}}$$

$$\frac{\partial J}{\partial W_{11}^{(4)}} = \frac{\partial J}{\partial z_1^{(5)}} \frac{\partial z_1^{(5)}}{\partial W_{11}^{(4)}}$$

$$\frac{\partial J}{\partial W_{11}^{(3)}} = \frac{\partial J}{\partial z_1^{(4)}} \frac{\partial z_1^{(4)}}{\partial W_{11}^{(3)}}$$

$$\frac{\partial J}{\partial W_{11}^{(2)}} = \frac{\partial J}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial W_{11}^{(2)}}$$

This rule works for all weights!

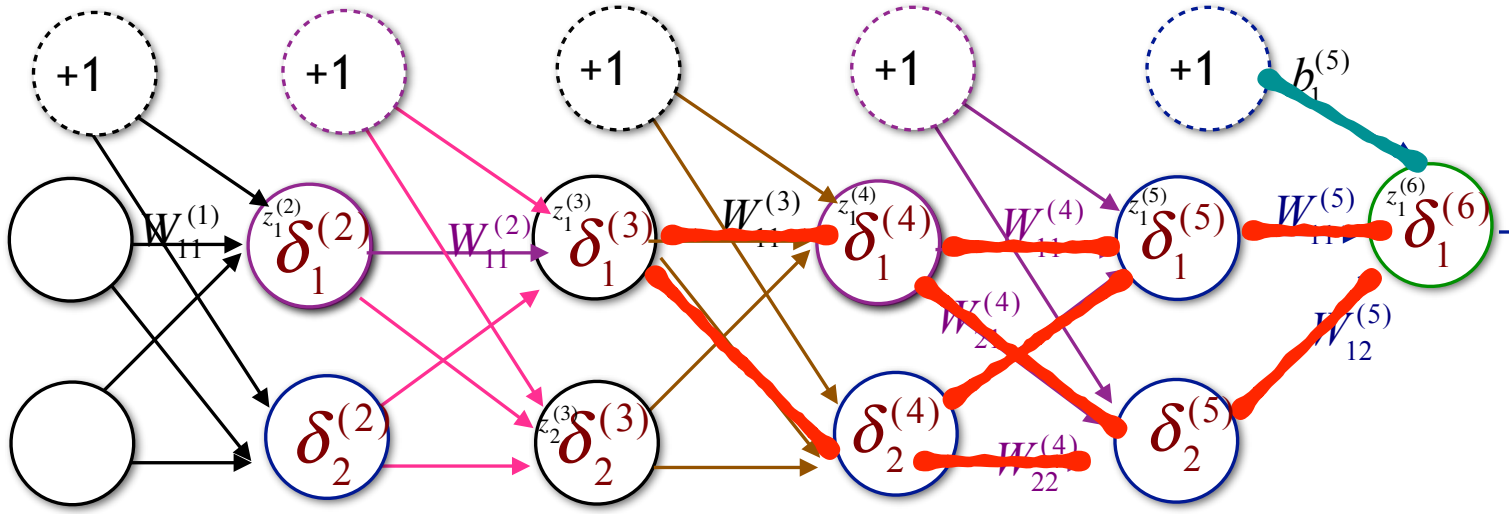
$$\frac{\partial J}{\partial W_{ij}^{(\ell)}} = \frac{\partial J}{\partial z_i^{(\ell+1)}} \frac{\partial z_i^{(\ell+1)}}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell+1)} a_j^{(\ell)}$$

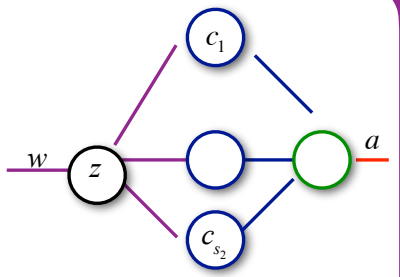
($a_j^{(\ell)}$ over the second factor)

$$\frac{\partial J}{\partial b_i^{(\ell)}} = \frac{\partial J}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial b_i^{(\ell)}} = \delta_i^{(\ell+1)}$$

(1 over the second factor)

$$\delta_j^{(\ell)} = \frac{\partial J}{\partial z_j^{(\ell)}}$$





$$\frac{da}{dw} = \frac{da}{dz}\frac{dz}{dw}$$

values propagate this way

$$z_i^{(\ell+1)} = W_{i1}^{(\ell)} a_1^{(\ell)} + \cdots + W_{ij}^{(\ell)} a_j^{(\ell)} + \cdots W_{is_\ell}^{(\ell)} a_{s_\ell}^{(\ell)} + b_i^{(\ell)}$$

How to update the weights…

Our update rule:

$W_{new} = W_{old} - \alpha \nabla \text{error}$

$$W_{ij}^{(\ell)} = W_{ij}^{(\ell)} - \alpha \frac{\partial J(W,b,\mathbf{x},y)}{\partial W_{ij}^{(\ell)}}$$

$$b_i^{(\ell)} = b_i^{(\ell)} - \alpha \frac{\partial J(W,b,\mathbf{x},y)}{\partial b_i^{(\ell)}}$$

$$\frac{\partial J}{\partial W_{11}^{(5)}} = \delta_1^{(6)} \frac{\partial z_1^{(6)}}{\partial W_{11}^{(5)}}$$

$$\frac{\partial J}{\partial W_{11}^{(4)}} = \delta_1^{(5)} \frac{\partial z_1^{(5)}}{\partial W_{11}^{(4)}}$$

$$\frac{\partial J}{\partial W_{11}^{(3)}} = \delta_1^{(4)} \frac{\partial z_1^{(4)}}{\partial W_{11}^{(3)}}$$

$$\frac{\partial J}{\partial W_{11}^{(2)}} = \delta_1^{(3)} \frac{\partial z_1^{(3)}}{\partial W_{11}^{(2)}}$$

This rule works for all weights!

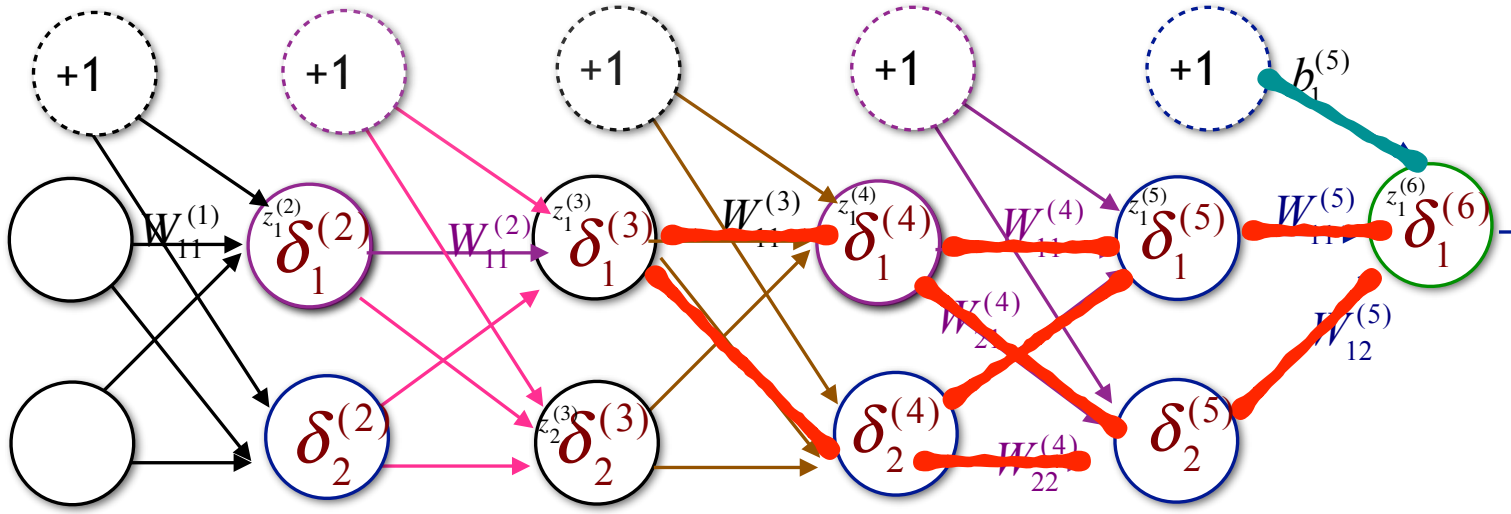
$$\frac{\partial J}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell+1)} \frac{\partial z_i^{(\ell+1)}}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell+1)} a_j^{(\ell)}$$

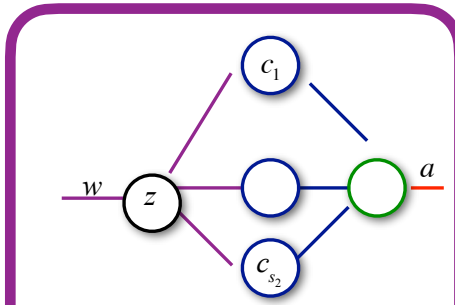
($a_j^{(\ell)}$ = activation of unit j in layer ℓ)

$$\frac{\partial J}{\partial b_i^{(\ell)}} = \frac{\partial J}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial b_i^{(\ell)}} = \delta_i^{(\ell+1)}$$

(1 = $\frac{\partial z_i^{(l+1)}}{\partial b_i^{(\ell)}}$)

$$\delta_j^{(\ell)} = \frac{\partial J}{\partial z_j^{(\ell)}}$$



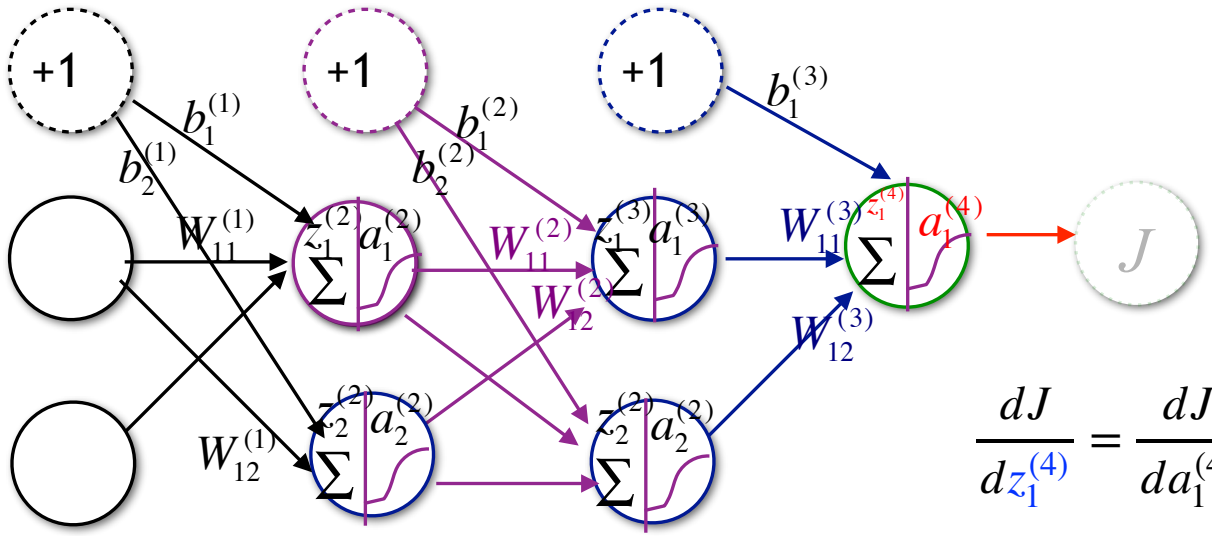


$$\frac{da}{dw} = \frac{da}{dz}\frac{dz}{dw}$$

values propagate this way

$$z_i^{(\ell+1)} = W_{i1}^{(\ell)} a_1^{(\ell)} + \cdots + W_{ij}^{(\ell)} a_j^{(\ell)} + \cdots W_{is_\ell}^{(\ell)} a_{s_\ell}^{(\ell)} + b_i^{(\ell)}$$

Computing δ for the last layer



Update weight to reduce error:

$$\frac{1}{2}(a_1^{(4)} - y)^2$$

$$\frac{dJ}{dz_1^{(4)}} = \frac{dJ}{da_1^{(4)}} \frac{da_1^{(4)}}{dz_1^{(4)}}$$

$$= (a_1^{(4)} - y) \frac{\partial a^{(4)}}{\partial z_1^{(4)}}$$

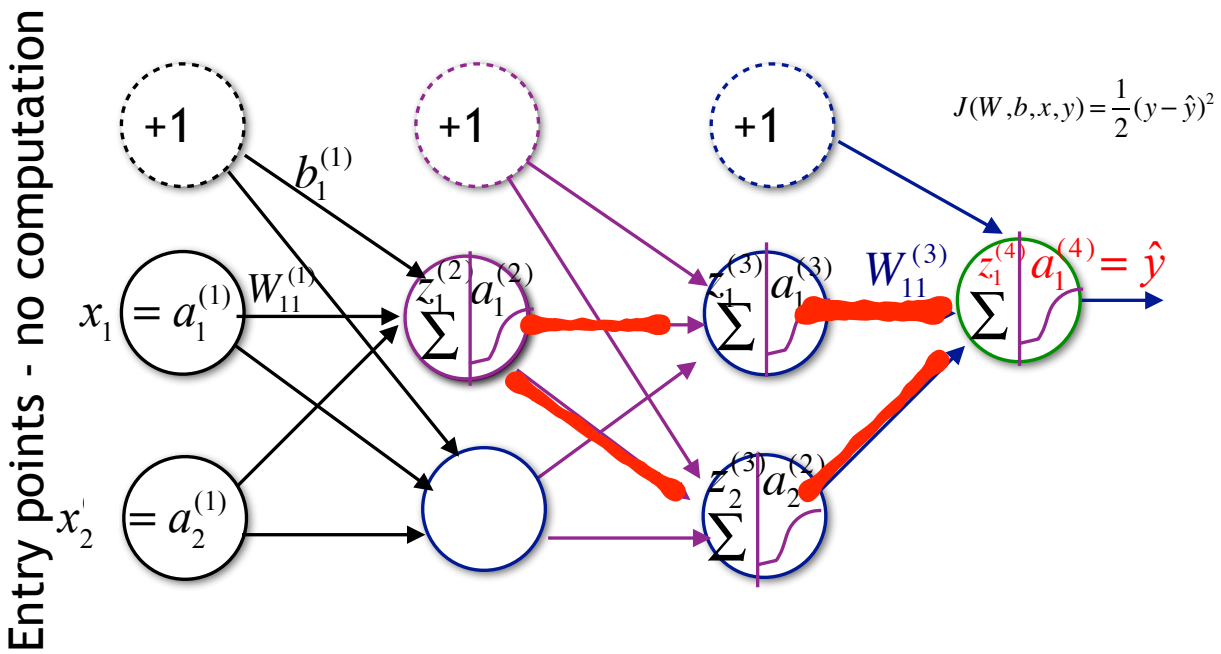
$$= (a_1^{(4)} - y) \frac{\partial f(z^{(4)})}{\partial z_1^{(4)}}$$

$$= \underbrace{(a_1^{(4)} - y) f'(z^{(4)})}_{\frac{\partial J}{\partial z_1^{(4)}}}$$

Small example to build intuition: computing $\delta_1^{(2)}$ for the graph below

$$\delta_1^{(2)} = \frac{\partial J(W,b,\mathbf{x},y)}{\partial z_1^{(2)}} = \frac{\partial J}{\partial z_1^{(3)}}\frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} + \frac{\partial J}{\partial z_2^{(3)}}\frac{\partial z_2^{(3)}}{\partial z_1^{(2)}}$$

$$= \sum_{i=1}^{2} \frac{\partial J}{\partial z_i^{(3)}}\frac{\partial z_i^{(3)}}{\partial z_1^{(2)}} = \sum_{i=1}^{s_{\ell+1}} \frac{\partial J}{\partial z_i^{(3)}} W_{i1} f'(z_i^{(2)})$$



$$\frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} = W_{11}^{(2)}\frac{\partial f(z_1^{(2)})}{\partial z_1^{(2)}} = W_{11} f'(z_1^{(2)})$$

$$\frac{\partial z_2^{(3)}}{\partial z_1^{(2)}} = W_{21}^{(2)}\frac{\partial f(z_2^{(2)})}{\partial z_1^{(2)}} = W_{21}^{(2)} f'(z_2^{(2)})$$

$$z_1^{(3)} = W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + b_1^{(2)}$$

$$z_2^{(3)} = W_{21}^{(2)} a_1^{(2)} + W_{22}^{(2)} a_2^{(2)} + b_2^{(2)}$$

$$z_1^{(3)} = W_{11}^{(2)} f(z_1^{(2)}) + W_{12}^{(2)} f(z_2^{(2)}) + b_1^{(2)}$$

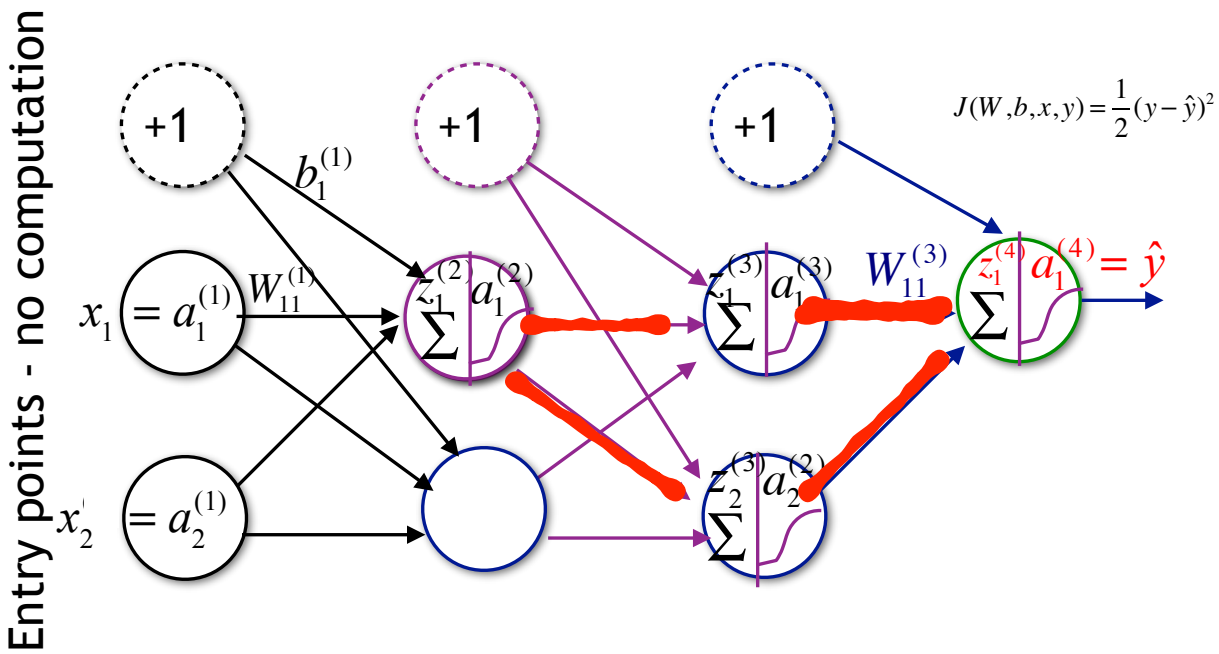
$$z_2^{(3)} = W_{21}^{(2)} f(z_1^{(2)}) + W_{22}^{(2)} f(z_2^{(2)}) + b_2^{(2)}$$

Slide added after class

Small example to build intuition: computing $\delta_1^{(2)}$ for the graph below

$$\delta_1^{(2)} = \frac{\partial J(W,b,\mathbf{x},y)}{\partial z_1^{(2)}} = \delta_1^{(3)} \frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} + \delta_2^{(3)} \frac{\partial z_2^{(3)}}{\partial z_1^{(2)}}$$

$$= \sum_{i=1}^{2} \delta_i^{(3)} \frac{\partial z_i^{(3)}}{\partial z_1^{(2)}} = \sum_{i=1}^{s_{\ell+1}} \delta_i^{(3)} W_{i1} f'(z_i^{(2)})$$



$$\frac{\partial z_1^{(3)}}{\partial z_1^{(2)}} = W_{11}^{(2)} \frac{\partial f(z_1^{(2)})}{\partial z_1^{(2)}} = W_{11} f'(z_1^{(2)})$$

$$\frac{\partial z_2^{(3)}}{\partial z_1^{(2)}} = W_{21}^{(2)} \frac{\partial f(z_2^{(2)})}{\partial z_1^{(2)}} = W_{21}^{(2)} f'(z_2^{(2)})$$

$$z_1^{(3)} = W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + b_1^{(2)}$$

$$z_2^{(3)} = W_{21}^{(2)} a_1^{(2)} + W_{22}^{(2)} a_2^{(2)} + b_2^{(2)}$$

$$z_1^{(3)} = W_{11}^{(2)} f(z_1^{(2)}) + W_{12}^{(2)} f(z_2^{(2)}) + b_1^{(2)}$$

$$z_2^{(3)} = W_{21}^{(2)} f(z_1^{(2)}) + W_{22}^{(2)} f(z_2^{(2)}) + b_2^{(2)}$$

Slide added after class

Computing $\delta_j^{(\ell)}$ for $\ell < n_\ell$

$$\delta_j^{(\ell)} = \frac{\partial J(W,b,\mathbf{x},y)}{\partial z_j^{(\ell)}} = \sum_{i=1}^{s_{\ell+1}} \frac{\partial J}{\partial z_i^{(\ell+1)}} \frac{\partial z_i^{(\ell+1)}}{\partial z_j^{(\ell)}} = \sum_{i=1}^{s_{\ell+1}} \frac{\partial J}{\partial z_i^{(\ell+1)}} W_{ij} f'(z_j^{(\ell)})$$

$$= \frac{\partial J}{\partial z_1^{(\ell+1)}} \frac{\partial z_1^{(\ell+1)}}{\partial z_j^{(\ell)}} + \frac{\partial J}{\partial z_2^{(\ell+1)}} \frac{\partial z_2^{(\ell+1)}}{\partial z_j^{(\ell)}} + \cdots + \frac{\partial J}{\partial z_{s_{\ell+1}}^{(\ell+1)}} \frac{\partial z_{s_{\ell+1}}^{(\ell+1)}}{\partial z_j^{(\ell)}}$$

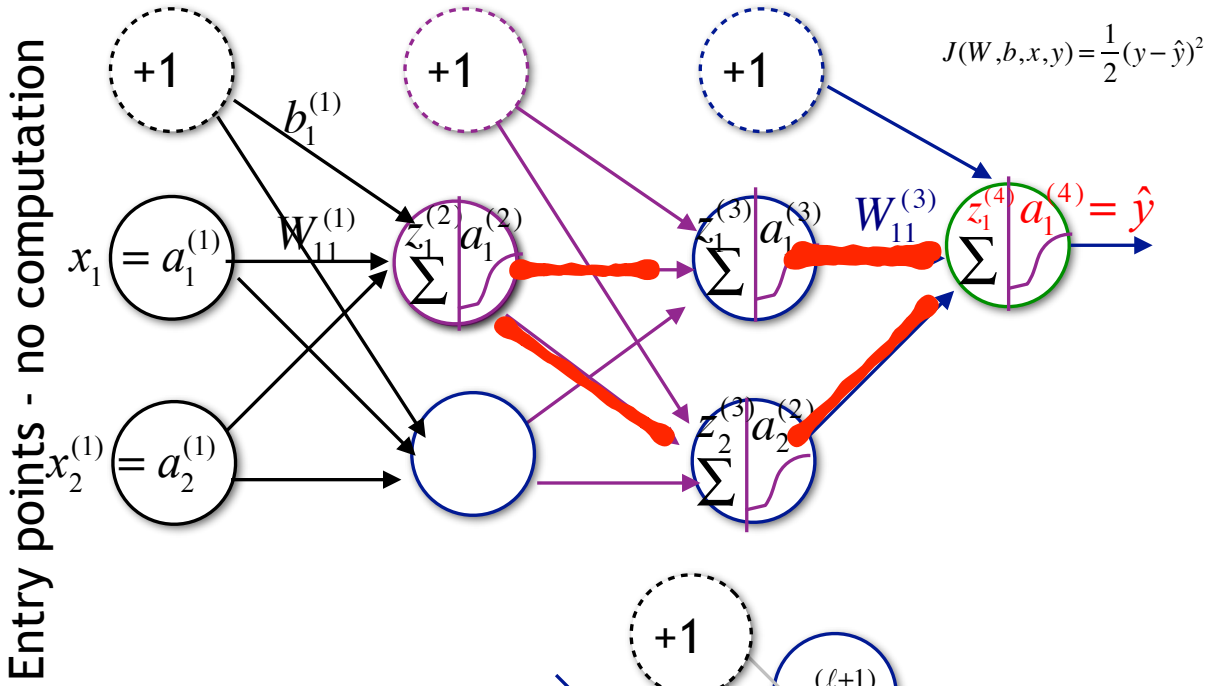
$$\frac{\partial z_i^{(\ell+1)}}{\partial z_j^{(\ell)}} = W_{ij}^{(\ell)} \frac{\partial f(z_j^{(\ell)})}{\partial z_j^{(\ell)}} = W_{ij}^{\ell} f'(z_j^{(\ell)})$$

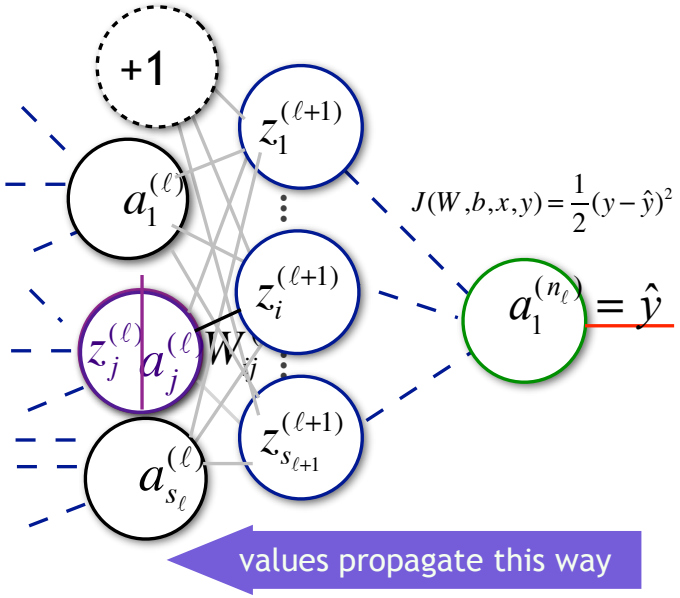
$$z_i^{(\ell+1)} = \sum_{k=1}^{s_\ell} W_{ik}^{(\ell)} a_k^{(\ell)} + b_i^{(\ell)}$$

$$= \sum_{k=1}^{s_\ell} W_{ik}^{(\ell)} f(z_k^{(\ell)}) + b_i^{(\ell)}$$

$$= W_{i1}^{(\ell)} f(z_1^{(\ell)}) + \cdots + W_{ij}^{(\ell)} f(z_j^{(\ell)}) + \cdots + W_{is_\ell}^{(\ell)} f(z_{s_\ell}^{(\ell)}) + b_i^{(\ell)}$$

Entry points - no computation





Computing $\delta_j^{(\ell)}$ for $j < n_\ell$

$$\delta_j^{(\ell)} = \frac{\partial J(W,b,\mathbf{x},y)}{\partial z_j^{(\ell)}} = \sum_{i=1}^{s_{\ell+1}} \delta_i^{(\ell+1)} \frac{\partial z_i^{(\ell+1)}}{\partial z_j^{(\ell)}} = \sum_{i=1}^{s_{\ell+1}} \delta_i^{(\ell+1)} W_{ij} f'(z_j^{(\ell)})$$

$$= \frac{\partial J}{\partial z_1^{(\ell+1)}} \frac{\partial z_1^{(\ell+1)}}{\partial z_j^{(\ell)}} + \frac{\partial J}{\partial z_2^{(\ell+1)}} \frac{\partial z_2^{(\ell+1)}}{\partial z_j^{(\ell)}} + \cdots + \frac{\partial J}{\partial z_{s_{\ell+1}}^{(\ell+1)}} \frac{\partial z_{s_{\ell+1}}^{(\ell+1)}}{\partial z_j^{(\ell)}}$$

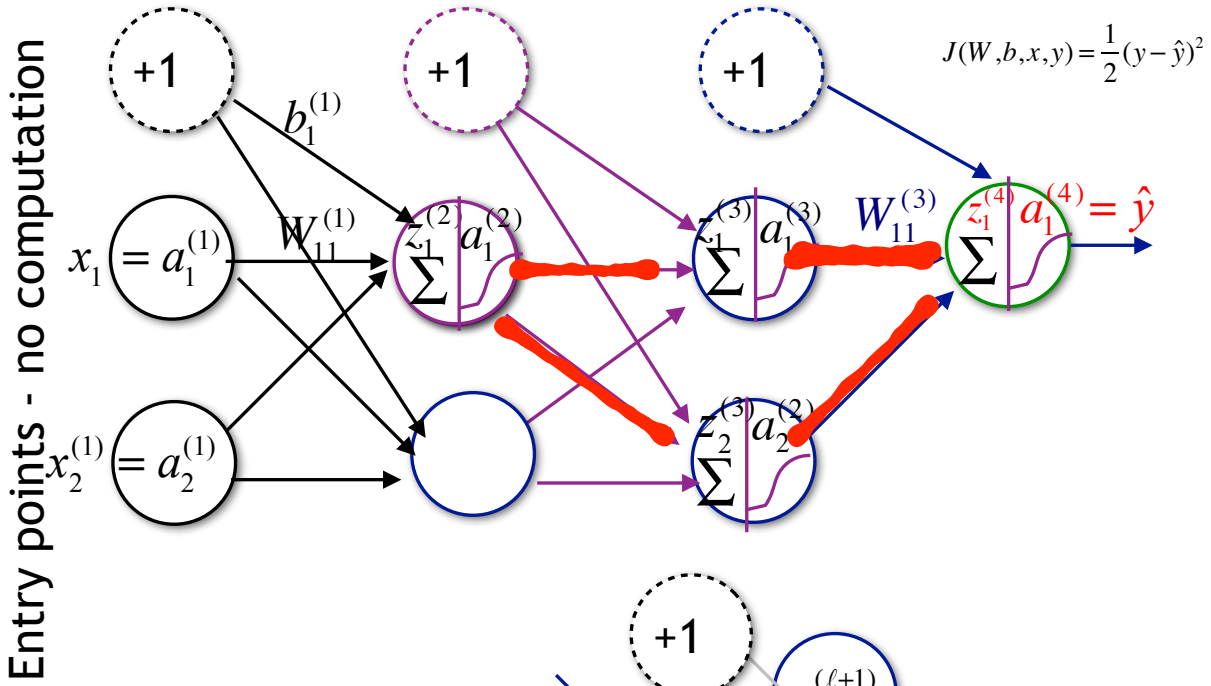
$$\frac{\partial z_i^{(\ell+1)}}{\partial z_j^{(\ell)}} = W_{ij}^{(\ell)} \frac{\partial f(z_j^{(\ell)})}{\partial z_j^{(\ell)}} = W_{ij}^{\ell} f'(z_j^{(\ell)})$$

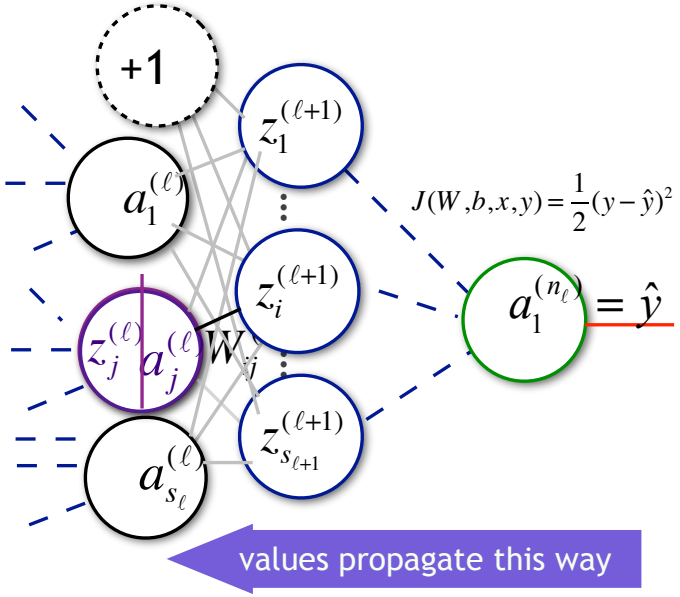
$$z_i^{(\ell+1)} = \sum_{k=1}^{s_\ell} W_{ik}^{(\ell)} a_k^{(\ell)} + b_i^{(\ell)}$$

$$= \sum_{k=1}^{s_\ell} W_{ik}^{(\ell)} f(z_k^{(\ell)}) + b_i^{(\ell)}$$

$$= W_{i1}^{(\ell)} f(z_1^{(\ell)}) + \cdots + W_{ij}^{(\ell)} f(z_j^{(\ell)}) + \cdots + W_{is_\ell}^{(\ell)} f(z_{s_\ell}^{(\ell)}) + b_i^{(\ell)}$$

Entry points - no computation





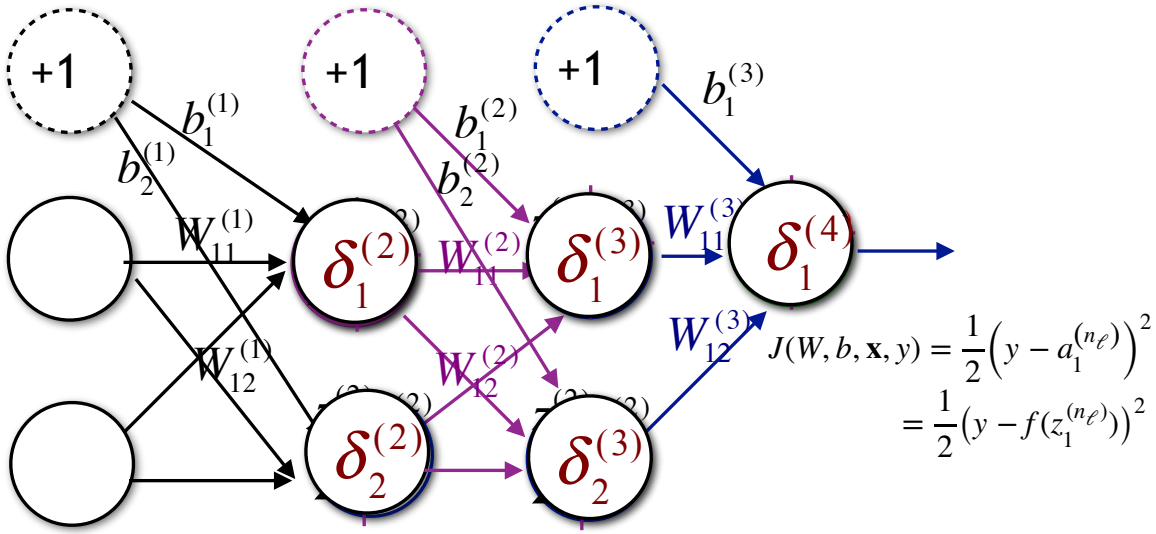
Putting the pieces together

$$\delta_i^{(\ell)} = \frac{dJ}{dz_i^{(\ell)}}$$

Chain rule from previous slide:

$$\frac{\partial J}{\partial W_{ij}^{(\ell)}} = \frac{\partial J}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell+1)} a_j^{(\ell)}$$

$$\frac{\partial J}{\partial b_i^{(\ell)}} = \frac{\partial J}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial b_i^{(\ell)}} = \delta_i^{(\ell+1)}$$



We first compute $\delta_1^{(4)}$, then we compute $\delta_1^{(3)}$ and $\delta_2^{(3)}$, then we compute $\delta_1^{(2)}$ and $\delta_2^{(2)}$.

What is $\delta_i^{(\ell)}$?

For the output layer:

$$\delta_j^{(n_\ell)} = \frac{\partial J}{\partial z_j^{(n_\ell)}}$$

I added a subscript in case there was more than one output neuron

$$= -(y_j - f(z_j^{(n_\ell)}) f'(z_j^{(n_\ell)})$$

For the other layers:

$$\delta_j^{(\ell)} = \frac{\partial J}{\partial z_j^{(\ell)}}$$

$$= \sum_{i=1}^{s_{\ell+1}} \delta_i^{(\ell+1)} W_{ij}^{(\ell)} f'(z_j^{(\ell)})$$

$$\delta_j^{(n_\ell)} = \frac{dJ}{dz_j^{(n_\ell)}} = (f(z_j^{(n_\ell)}) - y) f'(z_j^{(n_\ell)})$$

$$\delta_j^{(\ell)} = \frac{dJ}{dz_j^{(\ell)}} = \sum_{i=1}^{s_{\ell+1}} \delta_i^{(\ell+1)} W_{ij}^{(\ell)} f'(z_j^{(\ell)})$$

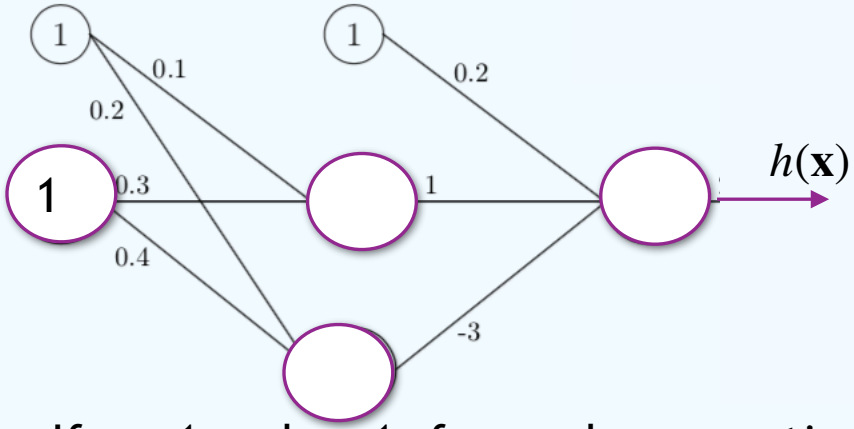
$$\frac{df(z)}{dz} = f(z)(1 - f(z))$$

Here we choose $f(z) = \sigma(z)$ to be the logistic function

$$\frac{dJ}{db_i^{(\ell)}} = \delta_i^{(\ell+1)}$$

$$W_{ij}^{(\ell)} = W_{ij}^{(\ell)} - \alpha \frac{\partial J(W, b, \mathbf{x}, y)}{\partial W_{ij}^{(\ell)}}$$

$$\frac{\partial J(W, b, \mathbf{x}, y)}{\partial b_i^{(\ell)}}$$



$$W^{(1)} = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix} \qquad W^{(2)} = [1 \quad -3]$$

$$b^{(1)} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} \qquad b^{(2)} = [0.2]$$

If x = 1 and y =1, forward propagation:

$$x = a^{(1)} = [1] \qquad z^{(2)} = \begin{bmatrix} 1 * 0.3 + 0.1 \\ 1 * 0.4 + 0.2 \end{bmatrix} \qquad a^{(2)} = \begin{bmatrix} 0.60 \\ 0.65 \end{bmatrix} \qquad z^{(3)} = [1 * 0.60 + (-3) * 0.65 + 0.2] \qquad a^{(3)} = [\sigma(-1.15)]$$

$$= [0.24]$$

Backward propagation:

$$\delta^{(3)} = [(0.24 - 1)(.24)(1 - .24)] = [-0.14]$$

$$\delta_1^{(2)} = \frac{\partial J}{\partial z_1^{(2)}} = \delta_1^{(3)} W_{11}^{(2)} f'(z_1^{(2)}) = [-0.14 * 1(0.60)(1 - 0.60)] = [-0.03]$$

$$\frac{\partial J}{\partial W_{11}^{(2)}} = \delta_1^{(3)} a_1^{(2)} = [-0.14 * 0.60] \qquad \frac{\partial J}{\partial W_{11}^{(1)}} = \delta_1^{(2)} a_1^{(1)} = [-0.03 * 1] = (f(z_j^{(n_\ell)}) - y) f'(z_j^{(n_\ell)})$$

Calculations done with rounded numbers...

Similar to example in Learning from Data

45

Regularization

$$J(W,b,\mathbf{x},y) = \frac{1}{2}(y-\hat{y})^2 + \frac{\lambda}{2}\sum_{k=1}^{n_\ell-1}\sum_{j=1}^{s_\ell}\sum_{i=1}^{s_{\ell+1}}(W_{ij}^{(k)})^2$$

Adding L2 regularization

$$\frac{\partial J(W,b,\mathbf{x},y)}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell+1)} a_j^{(\ell)} + \lambda(W_{ij}^{(\ell)})$$

$$\frac{\partial J(W,b,\mathbf{x},y)}{\partial b_i^{(\ell)}} = \delta_i^{(\ell+1)}$$

Activation functions

If we use a ***sigmoid*** activation: $f(z) = \frac{1}{1+e^{-z}}$ then $\frac{df(z)}{dz} = f(z)(1-f(z))$

If we use a ***relu*** activation: $f(z) = \max(0,z)$ then $\frac{df(z)}{dz} = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$

Batch Gradient Descent

$$J(W,b) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}-y)^2 = \frac{1}{N}\sum_{i=1}^{N}J(W,b,\mathbf{x},y) + \frac{\lambda}{2}\sum_{k=1}^{n_\ell-1}\sum_{j=1}^{s_\ell}\sum_{i=1}^{s_{\ell+1}}(W_{ij}^{(k)})^2$$

Outline

- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

Implementing Gradient Descent!

Outline

- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

To efficiently implement the algorithm we use vectorization

Matrix-Vector Notation: $\delta_j^{(\ell)} = \frac{dJ}{dz_j^{(\ell)}}$

Hadamard product: element-wise product operator

$a = b \bullet c \quad a_i = b_i c_i$

$f'([z_1, z_2, z_3]^T) = [f'(z_1), f'(z_2), f'(z_3)]^T$

Our formulas

The output layer:

$$\delta_j^{(n_\ell)} = \frac{\partial J}{\partial z_j^{(n_\ell)}} = -(y_j - f(z_j^{(n_\ell)}))f'(z_j^{(n_\ell)})$$

The other layers:

$$\delta_j^{(\ell)} = \frac{\partial J}{\partial z_j^{(\ell)}} = \sum_{i=1}^{s_{\ell+1}} \delta_i^{(\ell+1)} W_{ij}^{(\ell)} f'(z_j^{(\ell)})$$

Vectorized formulas

$$\delta^{(n_\ell)} = -(y - f(z^{(n_\ell)})) \bullet f'(z^{(n_\ell)})$$

$$W^{(\ell)} = \begin{bmatrix} W_{11}^{(\ell)} & W_{12}^{(\ell)} & \cdots & W_{1s_\ell}^{(\ell)} \\ W_{21}^{(\ell)} & W_{22}^{(\ell)} & & W_{2s_\ell}^{(\ell)} \\ \vdots & & & \vdots \\ W_{s_{\ell+1}1}^{(\ell)} & W_{s_{\ell+1}2}^{(\ell)} & \cdots & W_{s_{\ell+1}s_\ell}^{(\ell)} \end{bmatrix}$$

$$\delta^{(\ell)} = \left(\left(W^{(\ell)}\right)^T \delta^{(\ell+1)}\right) \bullet f'(z^{(\ell)})$$

$$\left(W^{(\ell)}\right)^T = \begin{bmatrix} W_{11}^{(\ell)} & W_{21}^{(\ell)} & \cdots & W_{s_{\ell+1}1}^{(\ell)} \\ W_{12}^{(\ell)} & W_{22}^{(\ell)} & & W_{s_{\ell+1}2}^{(\ell)} \\ \vdots & & & \vdots \\ W_{1s_\ell}^{(\ell)} & W_{2s_\ell}^{(\ell)} & \cdots & W_{s_{\ell+1}s_\ell}^{(\ell)} \end{bmatrix}$$

$$\delta^{(\ell)} = \begin{bmatrix} \delta_1^{(\ell)} \\ \delta_2^{(\ell)} \\ \vdots \\ \delta_{s_\ell}^{(\ell)} \end{bmatrix} = \left[\begin{bmatrix} W_{11}^{(\ell)} & W_{21}^{(\ell)} & \cdots & W_{s_{\ell+1}1}^{(\ell)} \\ W_{12}^{(\ell)} & W_{22}^{(\ell)} & & W_{s_{\ell+1}2}^{(\ell)} \\ \vdots & & & \vdots \\ W_{1s_\ell}^{(\ell)} & W_{2s_\ell}^{(\ell)} & \cdots & W_{s_{\ell+1}s_\ell}^{(\ell)} \end{bmatrix} \begin{bmatrix} \delta_1^{(\ell+1)} \\ \delta_2^{(\ell+1)} \\ \vdots \\ \delta_{s_{\ell+1}}^{(\ell+1)} \end{bmatrix}\right] \bullet \begin{bmatrix} f'(z_1^{(\ell)}) \\ f'(z_2^{(\ell)}) \\ \vdots \\ f'(z_{s_\ell}^{(\ell)}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{s_{\ell+1}} W_{i1}^{(\ell)} \delta_i^{(\ell+1)} \\ \sum_{i=1}^{s_{\ell+1}} W_{i2}^{(\ell)} \delta_i^{(\ell+1)} \\ \vdots \\ \sum_{i=1}^{s_{\ell+1}} W_{is_{\ell+1}}^{(\ell)} \delta_i^{(\ell+1)} \end{bmatrix} \bullet \begin{bmatrix} f'(z_1^{(\ell)}) \\ f'(z_2^{(\ell)}) \\ \vdots \\ f'(z_{s_\ell}^{(\ell)}) \end{bmatrix}$$

Matrix-Vector Notation:

Definition: $\delta_j^{(\ell)} = \frac{dJ}{dz_j^{(\ell)}}$ $\frac{dJ}{dW_{ij}^{\ell}} = \delta_i^{(\ell+1)} a_j^{(\ell)}$ $\frac{dJ}{db_i^{\ell}} = \delta_i^{(\ell+1)}$

Hadamard product: element-wise product operator $a = b \bullet c$ $a_i = b_i c_i$ $f'([z_1, z_2, z_3]^T) = [f'(z_1), f'(z_2), f'(z_3)]^T$

The partial derivatives:

Our formulas

$$\frac{\partial J}{\partial W_{ij}^{(\ell)}} = \delta_i^{(\ell+1)} a_j^{(\ell)}$$

$$\frac{dJ}{db_i^{(\ell)}} = \delta_i^{(\ell+1)}$$

Vectorized formulas

$$\frac{\partial J}{\partial W^{(\ell)}} = \delta^{(\ell+1)} \left(a^{(\ell)}\right)^T$$

$$\frac{dJ}{db^{(\ell)}} = \delta^{(\ell+1)}$$

$$a^{(\ell)} = \begin{bmatrix} a_1^{(\ell)} \\ a_2^{(\ell)} \\ \vdots \\ a_{s_\ell}^{(\ell)} \end{bmatrix} \quad \delta^{(\ell+1)} = \begin{bmatrix} \delta_1^{(\ell+1)} \\ \delta_2^{(\ell+1)} \\ \vdots \\ \delta_{s_{\ell+1}}^{(\ell+1)} \end{bmatrix} \quad W^{(\ell)} = \begin{bmatrix} W_{11}^{(\ell)} & W_{12}^{(\ell)} & \cdots & W_{1s_\ell}^{(\ell)} \\ W_{21}^{(\ell)} & W_{22}^{(\ell)} & & W_{2s_\ell}^{(\ell)} \\ \vdots & & & \vdots \\ W_{s_{\ell+1}1}^{(\ell)} & W_{s_{\ell+1}2}^{(\ell)} & \cdots & W_{s_{\ell+1}s_\ell}^{(\ell)} \end{bmatrix}$$

$$\frac{\partial J}{\partial W^{(\ell)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(\ell)}} & \frac{\partial J}{\partial W_{12}^{(\ell)}} & \cdots & \frac{\partial J}{\partial W_{1s_\ell}^{(\ell)}} \\ \frac{\partial J}{\partial W_{21}^{(\ell)}} & \frac{\partial J}{\partial W_{22}^{(\ell)}} & & \frac{\partial J}{\partial W_{2s_\ell}^{(\ell)}} \\ \vdots & & & \vdots \\ \frac{\partial J}{\partial W_{s_{\ell+1}1}^{(\ell)}} & \frac{\partial J}{\partial W_{s_{\ell+1}2}^{(\ell)}} & \cdots & \frac{\partial J}{\partial W_{s_{\ell+1}s_\ell}^{(\ell)}} \end{bmatrix} = \begin{bmatrix} \delta_1^{(\ell+1)} \\ \delta_2^{(\ell+1)} \\ \vdots \\ \delta_{s_{\ell+1}}^{(\ell+1)} \end{bmatrix} \begin{bmatrix} a_1^{(\ell)} & a_2^{(\ell)} & \cdots & a_{s_\ell}^{(\ell)} \end{bmatrix}$$

How do we go from stochastic to batch gradient descent?

Our update rules: $W_{ij}^{(\ell)} = W_{ij}^{(\ell)} - \alpha \frac{1}{N} \frac{\partial J(W,b)}{\partial W_{ij}^{(\ell)}}$ and $b_i^{(\ell)} = b_i^{(\ell)} - \alpha \frac{1}{N} \frac{\partial J(W,b)}{\partial b_i^{(\ell)}}$

$$\frac{\partial J(W,b)}{\partial W_{ij}^{(\ell)}} = \sum_{i=1}^{N} \frac{\partial J(W,b,x^{(i)},y^{(i)})}{\partial W_{ij}^{(\ell)}} = \frac{1}{2} \sum_{i=1}^{N} \frac{\partial (y^{(i)} - \hat{y}^{(i)})^2}{\partial W_{ij}^{(\ell)}}$$

$$\Delta W^{(\ell)} = \sum \frac{\partial J(W,b,\boldsymbol{x},y)}{\partial W^{(\ell)}} = \begin{bmatrix} \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{11}^{(\ell)}} & \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{12}^{(\ell)}} & \cdots & \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{1s_\ell}^{(\ell)}} \\ \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{21}^{(\ell)}} & \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{22}^{(\ell)}} & & \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{2s_\ell}^{(\ell)}} \\ \vdots & & & \vdots \\ \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{s_{\ell+1}1}^{(\ell)}} & \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{s_{\ell+1}2}^{(\ell)}} & \cdots & \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial W_{s_{\ell+1}s_\ell}^{(\ell)}} \end{bmatrix}$$

$$\Delta b^{(\ell)} = \sum \frac{\partial J(W,b,\boldsymbol{x},y)}{\partial b^{(\ell)}} = \begin{bmatrix} \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial z_1^{(\ell)}} \\ \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial z_2^{(\ell)}} \\ \vdots \\ \sum_{i=1}^{N} \frac{\partial J(W,b,\boldsymbol{x}^{(i)},y^{(i)})}{\partial z_{s_{\ell+1}}^{(\ell)}} \end{bmatrix}$$

We just need to chose the notation $\Delta W^{(\ell)}$, $\Delta b^{(\ell)}$

$$W^{(\ell)} = W^{(\ell)} - \alpha \frac{1}{N} \Delta W^{(\ell)}$$

$$b^{(\ell)} = b^{(\ell)} - \alpha \frac{1}{N} \Delta b^{(\ell)}$$

The batch gradient descent algorithm

Algorithm and code adapted from http://adventuresinmachinelearning.com/neural-networks-tutorial/

Randomly initialize the weights for each layer: $W^{(\ell)}$

While iterations < iteration limit:

$\Delta W^{(\ell)} = 0$

$\Delta b^{(\ell)} = 0$

For i = 1 to N:

run forward propagation and save for each level, the values $a^{(\ell)}$ $\mathbf{z}^{(\ell)}$

run back-propagation to calculate $\delta^{(\ell)}$ values for levels 2 through n_ℓ

Update $\Delta W^{(\ell)}$ $\Delta b^{(\ell)}$ for each level $\Delta W^{(\ell)} = \Delta W^{(\ell)} + \delta^{(\ell+1)}\left(a^{(\ell)}\right)^T$

$\Delta b^{(\ell)} = \Delta b^{(\ell)} + \delta^{(\ell+1)}$

Perform a gradient descent step

$$W^{(\ell)} = W^{(\ell)} - \alpha \frac{1}{N} \Delta W^{(\ell)}$$

$$b^{(\ell)} = b^{(\ell)} - \alpha \frac{1}{N} \Delta b^{(\ell)}$$

Outline

- ❑ Introduction to neurons
- ❑ Nonlinear classifiers from linear features
- ❑ Neural networks notation
- ❑ Pseudocode for prediction
- ❑ Training a neural network
- ❑ Implementing gradient descent for neural networks
  - Vectorization
  - Pseudocode
- ❑ Preprocessing
- ❑ Initialization
- ❑ Activations

The batch gradient descent algorithm

Algorithm and code adapted from http://adventuresinmachinelearning.com/neural-networks-tutorial/

Randomly initialize the weights for each layer: $W^{(\ell)}$

While iterations < iteration limit:

$\Delta W^{(\ell)} = 0$

$\Delta b^{(\ell)} = 0$

For i = 1 to N:

run forward propagation and save for each level, the values $a^{(\ell)}$ $\mathbf{z}^{(\ell)}$

run back-propagation to calculate $\delta^{(\ell)}$ values for levels 2 through n_ℓ

Update $\Delta W^{(\ell)}$ $\Delta b^{(\ell)}$ for each level $\Delta W^{(\ell)} = \Delta W^{(\ell)} + \delta^{(\ell+1)} \left(a^{(\ell)}\right)^T$

$\Delta b^{(\ell)} = \Delta b^{(\ell)} + \delta^{(\ell+1)}$

Perform a gradient descent step

$$W^{(\ell)} = W^{(\ell)} - \alpha \frac{1}{N} \Delta W^{(\ell)}$$

$$b^{(\ell)} = b^{(\ell)} - \alpha \frac{1}{N} \Delta b^{(\ell)}$$

Random Initialization of Parameters

nn_structure = [3, 4, 3]

What are the dimensions of $W^{(1)}$?

a) 1x1
b) 3x4
c) 4x3
d) 3x3

Random Initialization of Parameters

```
nn_structure = [3, 4, 3]
```

$W^{(1)}$

```
[[0.65, 0.18, 0.89],
 [0.06, 0.74, 0.72],
 [0.92, 0.29, 0.71],
 [0.39, 0.41, 0.10]]
```

$b^{(1)}$

```
0.07,
0.35,
0.60,
0.50
```

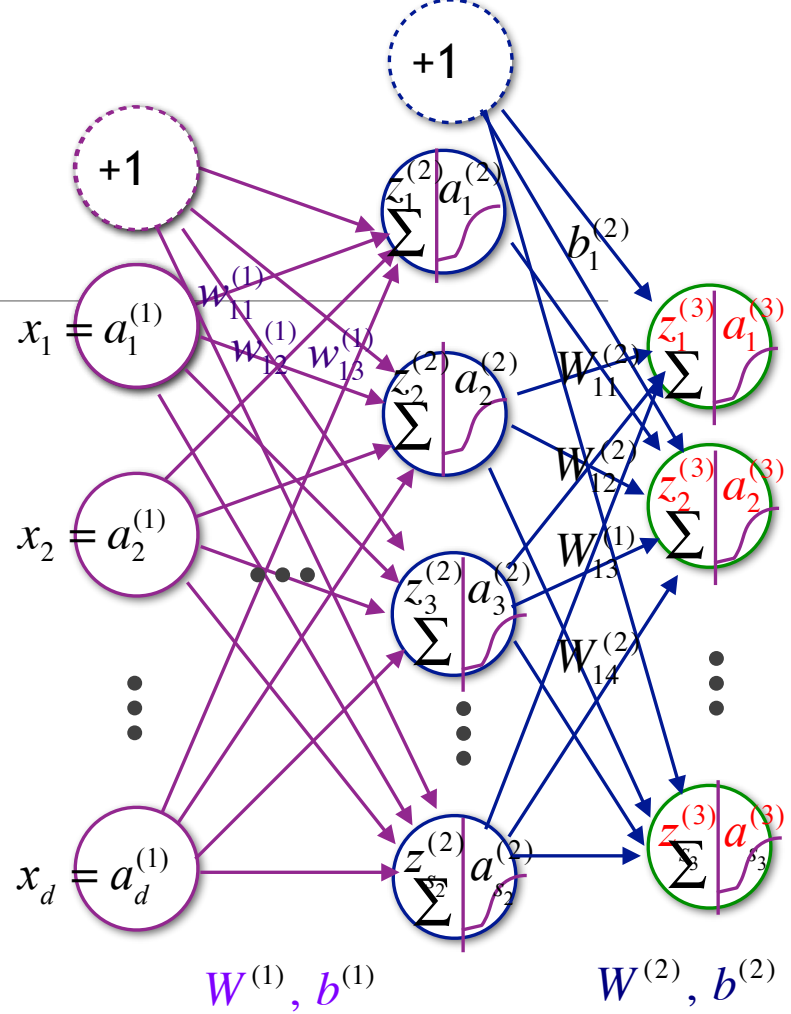
$W^{(2)}$

```
[[0.45, 0.64, 0.10, 0.82],
 [0.30, 0.25, 0.39, 0.36],
 [0.78, 0.60, 0.16, 0.07]]
```

$b^{(2)}$

```
0.14,
0.64,
0.62
```

```
def setup_and_init_weights(nn_structure):
    W = {}
    b = {}
    for l in range(1, len(nn_structure)):
        W[l] = r.random_sample((nn_structure[l], nn_structure[l-1]))
        b[l] = r.random_sample((nn_structure[l],))
    return W, b
```



Array - we represent it as column
Numpy represents it as a row

The gradient descent algorithm

Algorithm and code adapted from http://adventuresinmachinelearning.com/neural-networks-tutorial/

Randomly initialize the weights for each layer: $W^{(\ell)}$

While iterations < iteration limit:

$$\Delta W^{(\ell)} = 0$$
$$\Delta b^{(\ell)} = 0$$

For i = 1 to N:

run forward propagation and save for each level, the values $a^{(\ell)}$ $\mathbf{z}^{(\ell)}$
run back-propagation to calculate $\delta^{(\ell)}$ values for levels 2 through n_ℓ
Update $\Delta W^{(\ell)}$ $\Delta b^{(\ell)}$ for each level $\Delta W^{(\ell)} = \Delta W^{(\ell)} + \delta^{(\ell+1)}\left(a^{(\ell)}\right)^T$

$$\Delta b^{(\ell)} = \Delta b^{(\ell)} + \delta^{(\ell+1)}$$

Perform a gradient descent step

$$W^{(\ell)} = W^{(\ell)} - \alpha \frac{1}{N} \Delta W^{(\ell)}$$

$$b^{(\ell)} = b^{(\ell)} - \alpha \frac{1}{N} \Delta b^{(\ell)}$$

Initialization of ∇W and ∇b to zero

nn_structure = [3, 4, 3]

```
[[0., 0., 0.],   [0.,
 [0., 0., 0.],    0.,
 [0., 0., 0.],    0.,
 [0., 0., 0.]]    0.]
```

$\nabla W^{(1)}$ $\nabla b^{(1)}$

```
[[0., 0., 0., 0.],   [0.,
 [0., 0., 0., 0.],    0.,
 [0., 0., 0., 0.]]    0.]
```

$\nabla W^{(2)}$ $\nabla b^{(2)}$

```
def init_tri_values(nn_structure):
    tri_W = {}
    tri_b = {}
    for l in range(1, len(nn_structure)):
        tri_W[l] = np.zeros((nn_structure[l], nn_structure[l-1]))
        tri_b[l] = np.zeros((nn_structure[l],))
    return tri_W, tri_b
```