CS 171: Intro to ML and DM

Christian Shelton

UC Riverside

Slide Set 10: Neural Network I

Slides from CS 171

- From UC Riverside
  - CS 171: Introduction to Machine Learning and Data Mining
  - Professor Christian Shelton
- DO NOT REDISTRIBUTE
  - These slides contain copyrighted material (used with permission) from
    - Elements of Statistical Learning (Hastie, et al.)
    - Pattern Recognition and Machine Learning (Bishop)
    - An Introduction to Machine Learning (Kubat)
    - Machine Learning: A Probabilistic Perspective (Murphy)
  - For use only by enrolled students in the course

Non-linear methods

In

$$f(x) = x^\top w \qquad\qquad f(x) = \sigma(x^\top w)$$

replace x with $\phi(x)$:

$$f(x) = \phi(x)^\top w \qquad\qquad f(x) = \sigma(\phi(x)^\top w)$$

Non-linear methods

In

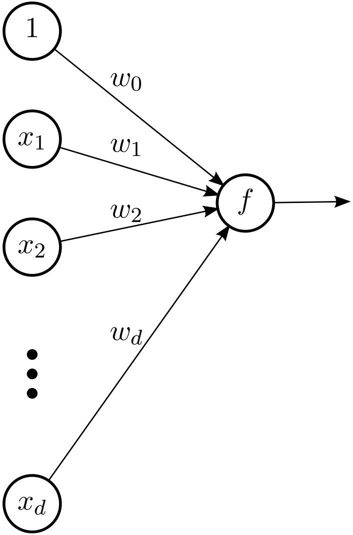
$$f(x) = x^\top w \qquad\qquad f(x) = \sigma(x^\top w)$$

replace x with $\phi(x)$:

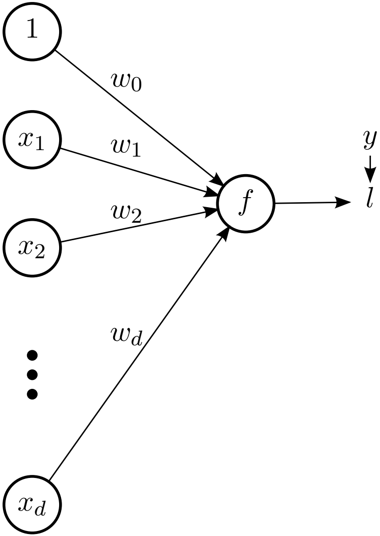
$$f(x) = \phi(x)^\top w \qquad\qquad f(x) = \sigma(\phi(x)^\top w)$$

If $\phi(x)$ selected by hand, we are done!

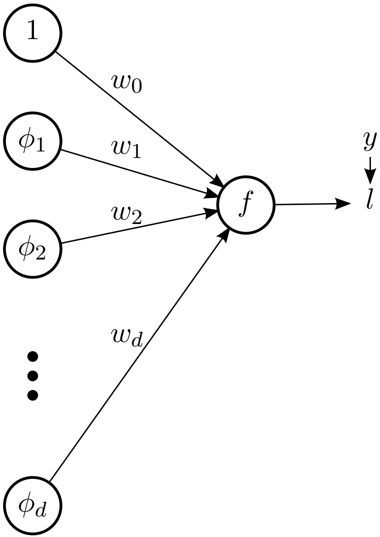
Learning Non-linearity



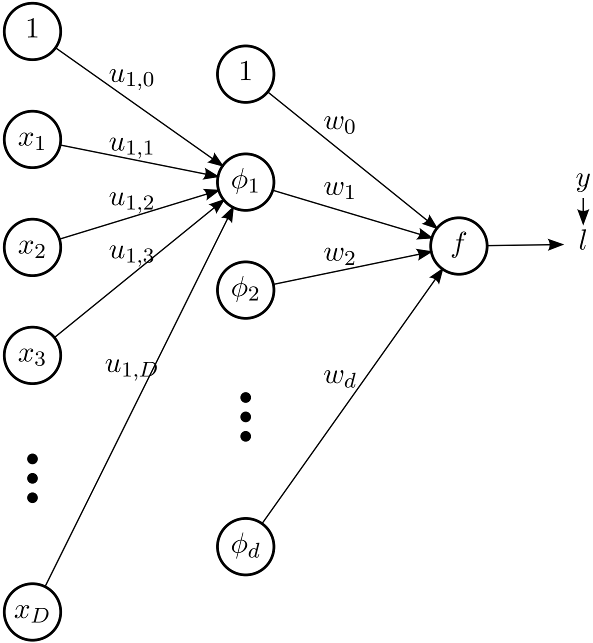
Learning Non-linearity



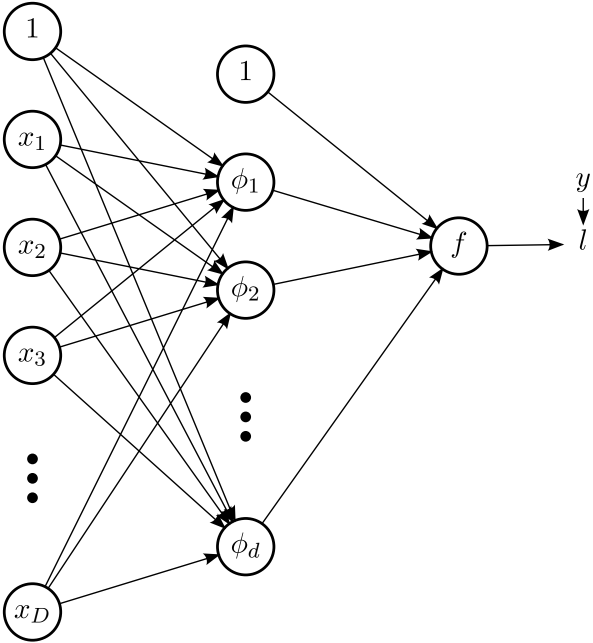
Learning Non-linearity



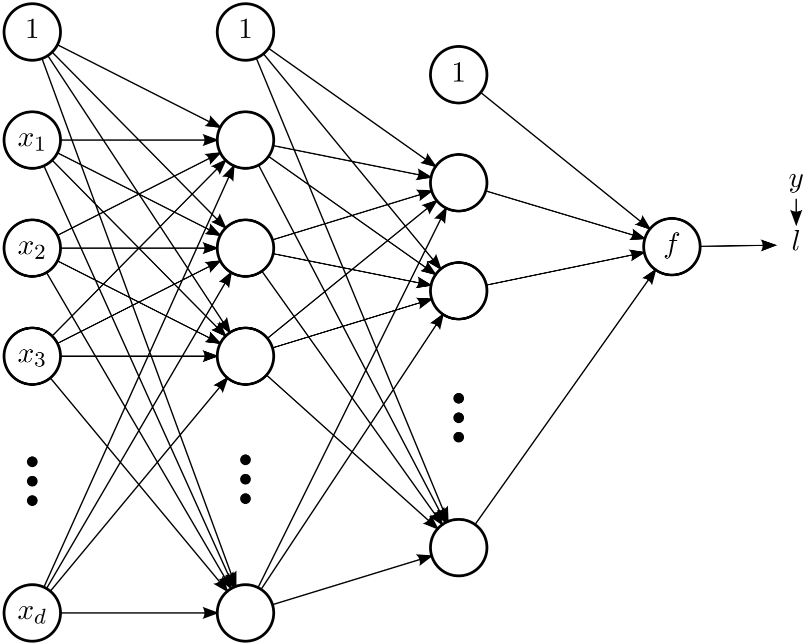
Learning Non-linearity



Learning Non-linearity



Learning Non-linearity



Neural Network Learning Algorithm

Stochastic Gradient Descent:

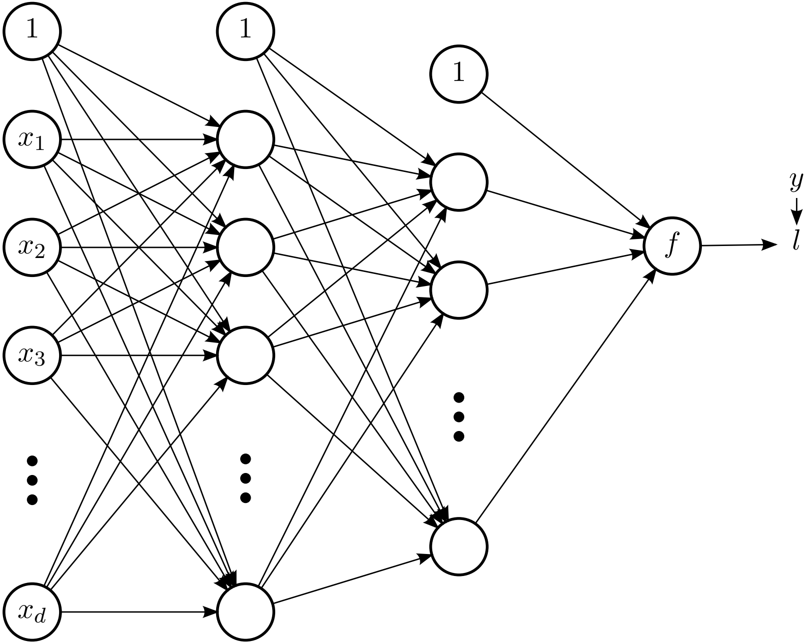
- Repeat until converged
  - For i from 1 to m (for each example)
    - Run forward propagation with x_i
    - Run backward propagation with y_i and results from forward-prop
    - Compute ∇_W for all weight layers, W, from forward-prop and backprop
    - Change weight layer W by $-\eta\nabla_W$

Neural Network Learning Algorithm

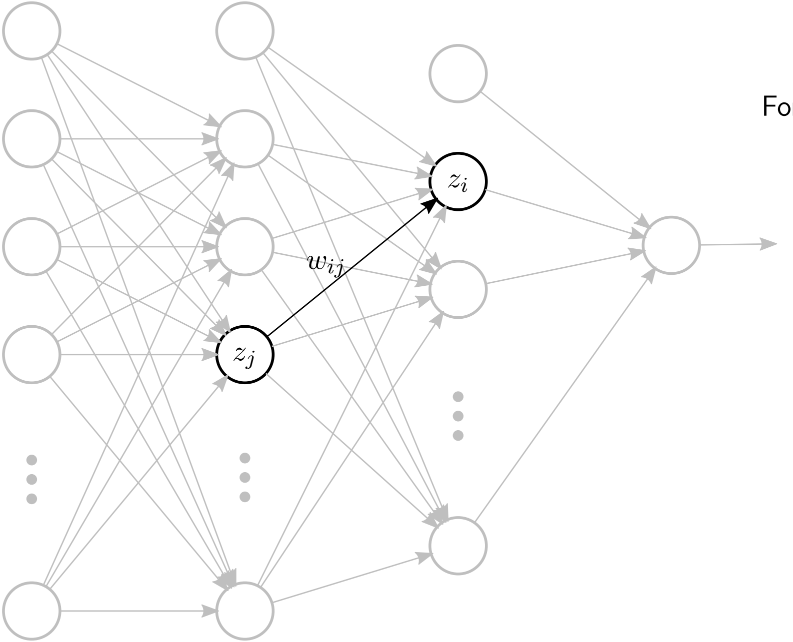
Gradient Descent:

- Repeat until converged
  - For i from 1 to m (for each example)
    - Run forward propagation with x_i
    - Run backward propagation with y_i and results from forward-prop
    - Compute ∇_W for all weight layers, W, from forward-prop and backprop
    - Add ∇_W to running sum, S_W for all weight layers
  - Change weight layer W by $-\eta S_W$

Learning Non-linearity



Learning Non-linearity

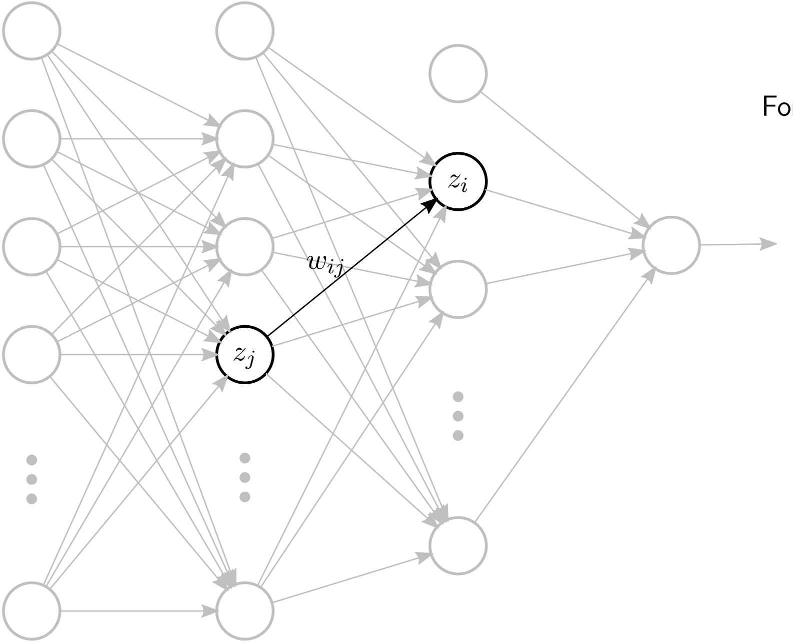


For a given x, y

$$z_i = g_i(a_i)$$

$$a_i = \sum_j w_{ij} z_j$$

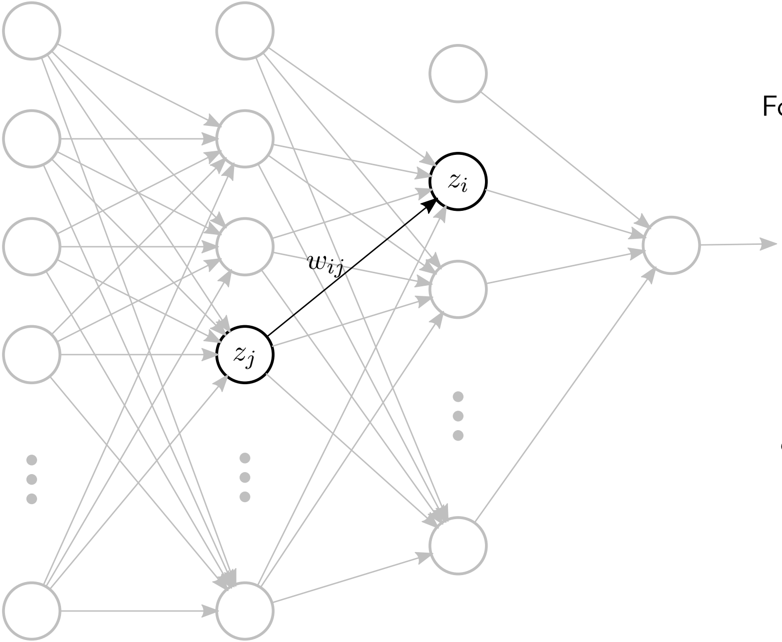
Learning Non-linearity



For a given x, y

$$
\begin{aligned}
z_i &= g_i(a_i) \\
a_i &= \sum_j w_{ij} z_j \\
\frac{\partial l}{\partial w_{ij}} &= \frac{\partial l}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \\
&= \delta_i z_j
\end{aligned}
$$

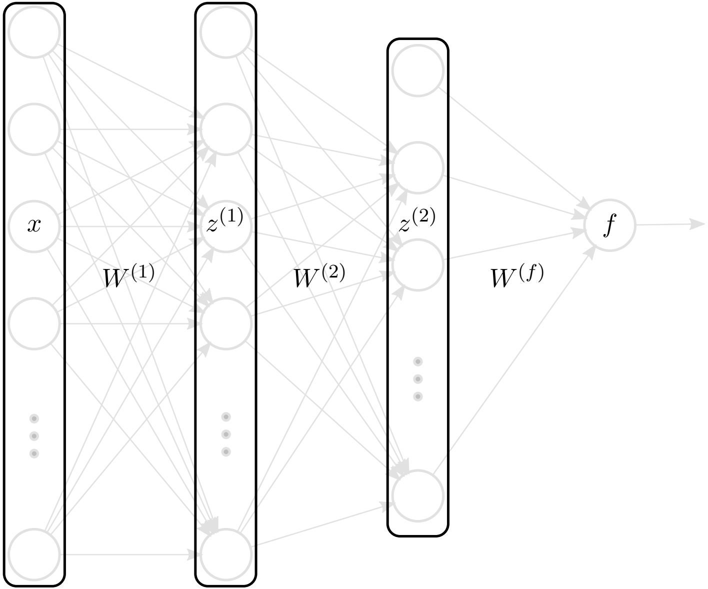
Learning Non-linearity



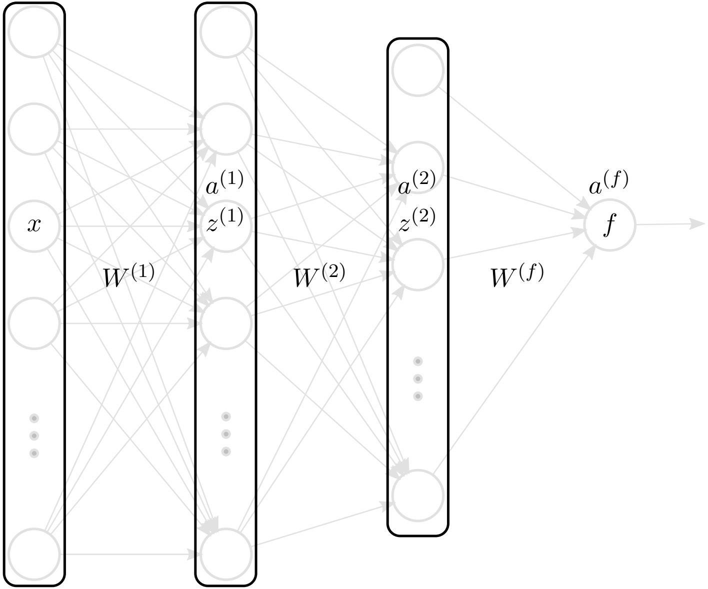
For a given x, y

$$\begin{aligned}
z_i &= g_i(a_i) \\
a_i &= \sum_j w_{ij} z_j \\
\frac{\partial l}{\partial w_{ij}} &= \frac{\partial l}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} \\
&= \delta_i z_j \\
\delta_j = \frac{\partial l}{\partial a_j} &= \frac{\partial l}{\partial z_j}\frac{\partial z_j}{\partial a_j} \\
&= \sum_i \frac{\partial l}{\partial a_i}\frac{\partial a_i}{\partial z_j}\frac{\partial z_j}{\partial a_j} \\
&= \sum_i \delta_i w_{ij} g'(a_j)
\end{aligned}$$

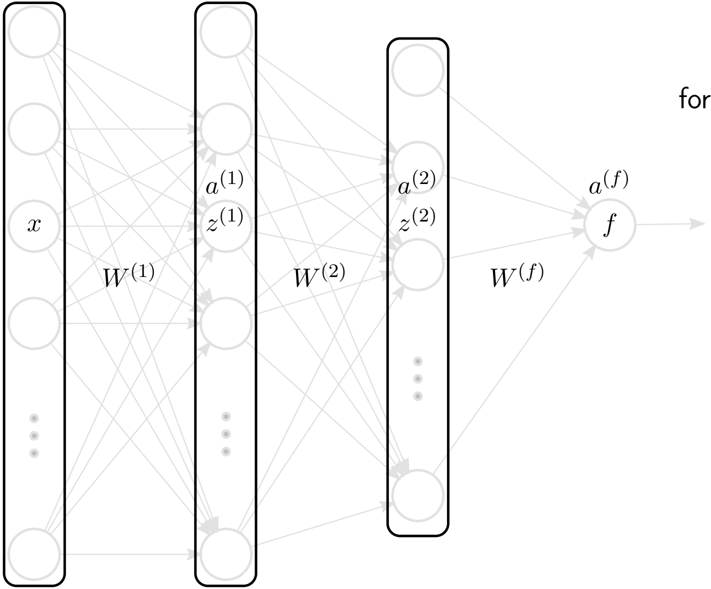
Layered Neural Network



Layered Neural Network



Layered Neural Network



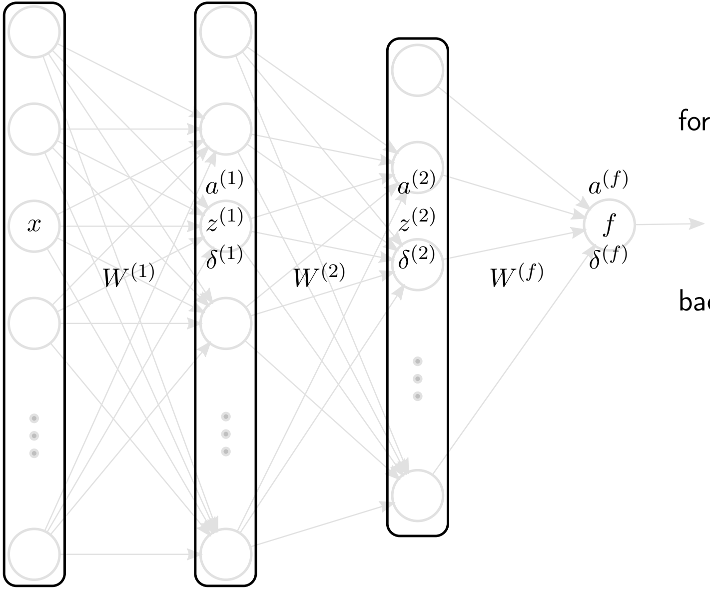
forward:

$$a^{(i)} = W^{(i)} z^{(i-1)}$$

$$z^{(i)} = g(a^{(i)})$$

(element-wise application of g)

Layered Neural Network



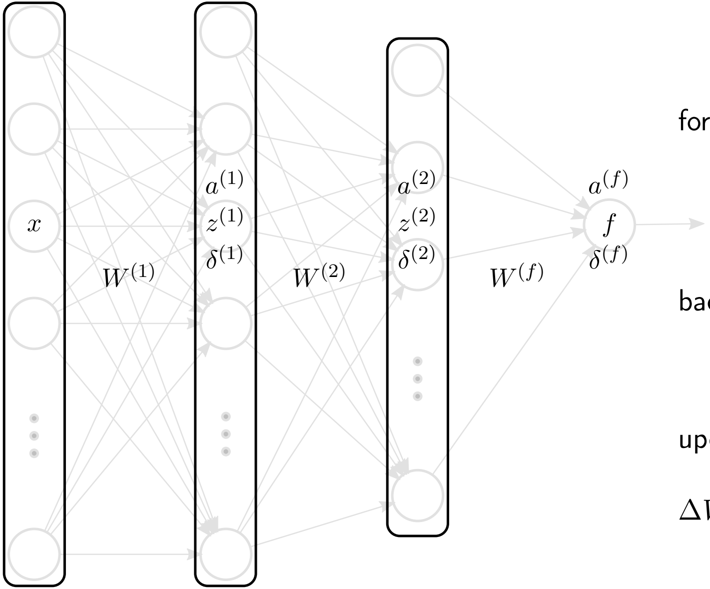
forward:

$$a^{(i)} = W^{(i)} z^{(i-1)}$$
$$z^{(i)} = g(a^{(i)})$$

backward:

$$\delta^{(i)} = g'(a^{(i)}) \odot \left({W^{(i+1)}}^\top \delta^{(i+1)} \right)$$

Layered Neural Network



forward:

$$a^{(i)} = W^{(i)} z^{(i-1)}$$
$$z^{(i)} = g(a^{(i)})$$

backward:

$$\delta^{(i)} = g'(a^{(i)}) \odot \left({W^{(i+1)}}^\top \delta^{(i+1)} \right)$$

update:

$$\Delta W^{(i)} = -\eta \delta^{(i)} {z^{(i-1)}}^\top$$

Output Layer

For regression:

$$g_f(a) = a$$

$$\delta^{(f)} = (f - y)$$

$$l(f, y) = \frac{1}{2}(f - y)^2$$

Output Layer

For regression:

$$g_f(a) = a \qquad\qquad l(f, y) = \frac{1}{2}(f - y)^2$$

$$\delta^{(f)} = (f - y)$$

For binary classification, $y \in \{0, 1\}$:

$$g_f(a) = \sigma(a) \qquad\qquad l(f, y) = -(y \log(f) + (1 - y) \log(1 - f))$$

$$\delta^{(f)} = (f - y)$$

Non-linearities

Historically most common:

$$g(a) = \sigma(a) = 1/(1 + e^{-a})$$

$$g'(a) = g(a)\,(1 - g(a))$$

Non-linearities

Historically most common:

$$g(a) = \sigma(a) = 1/(1 + e^{-a})$$
$$g'(a) = g(a)\,(1 - g(a))$$

(we will always use this in this course)

Non-linearities

Historically most common:

$$g(a) = \sigma(a) = 1/(1 + e^{-a})$$

$$g'(a) = g(a)\,(1 - g(a))$$

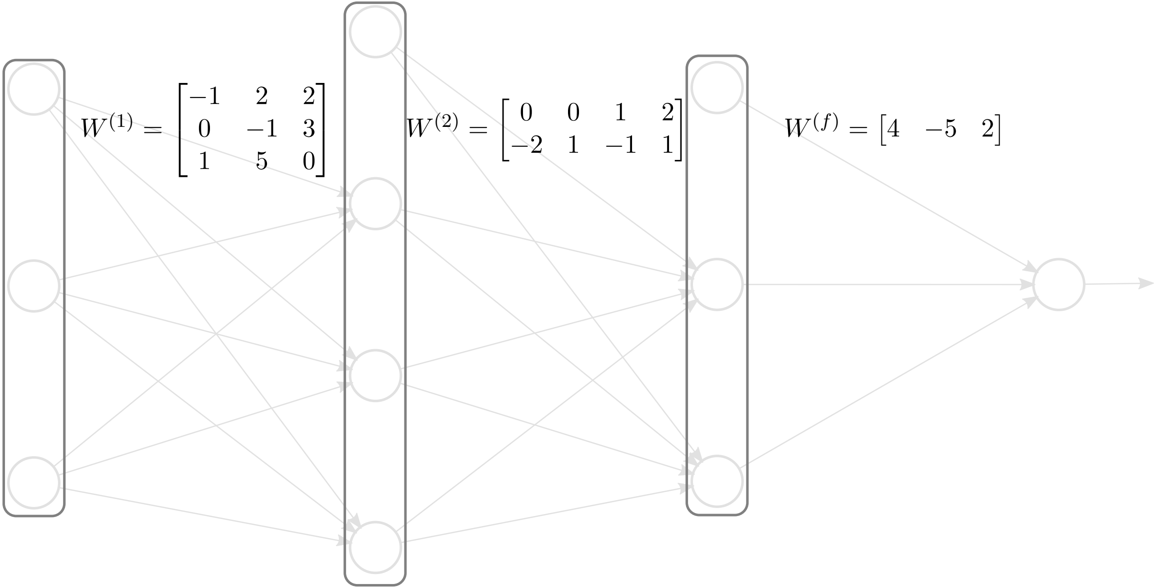
(we will always use this in this course)

Now common (for hidden units):

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

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

Layered Neural Network Example

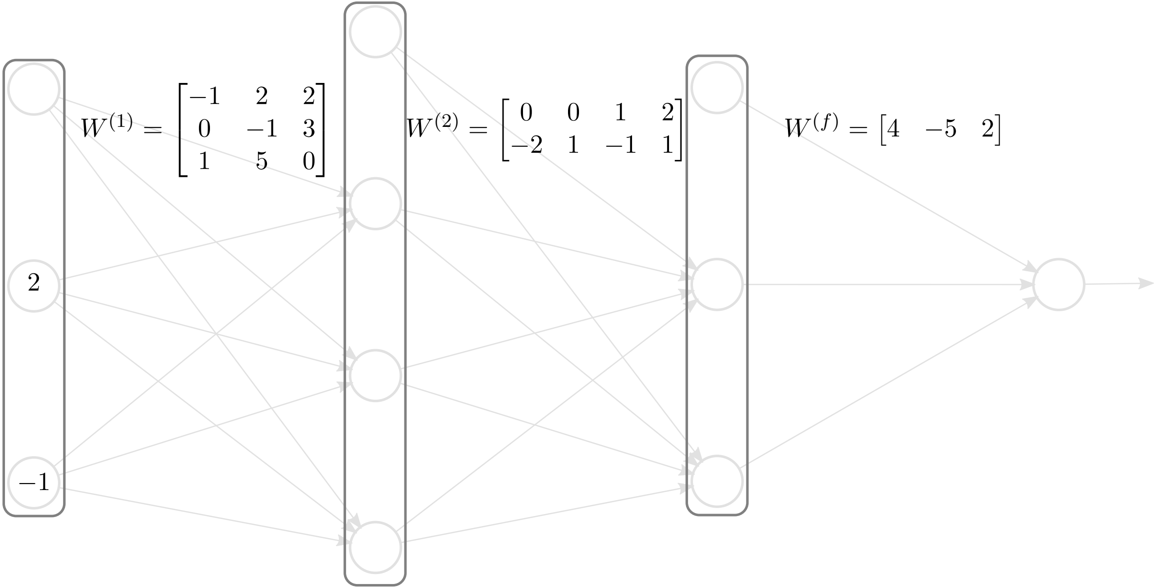


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

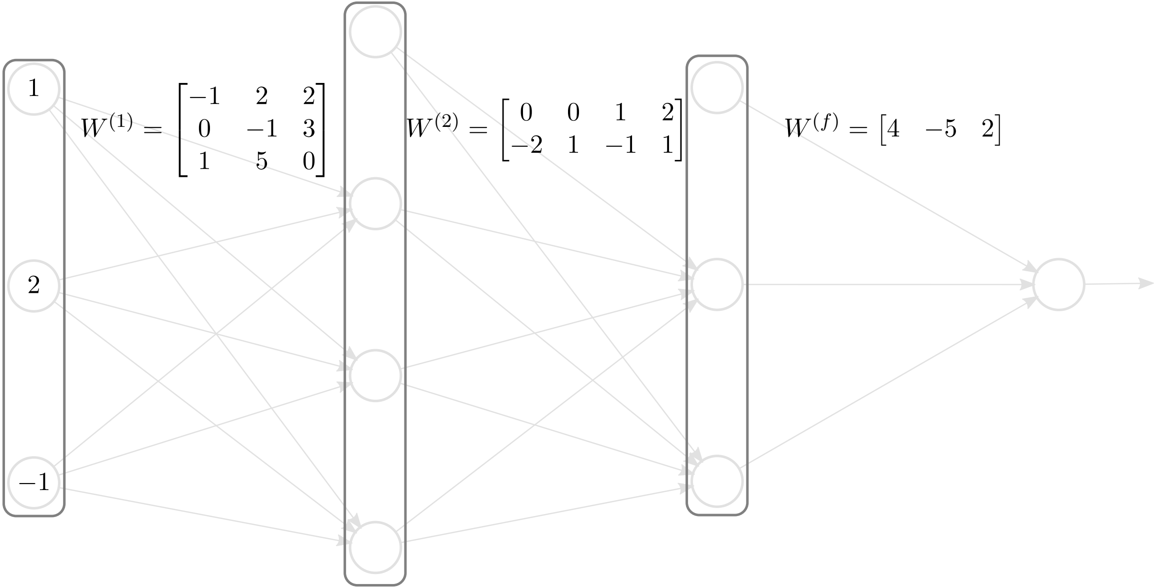


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

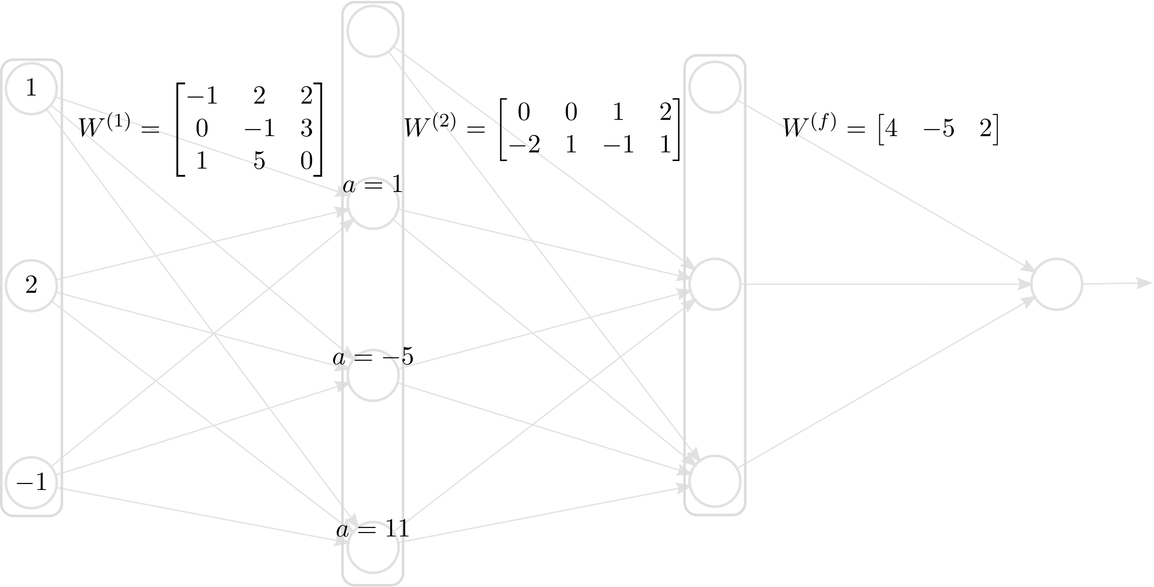


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

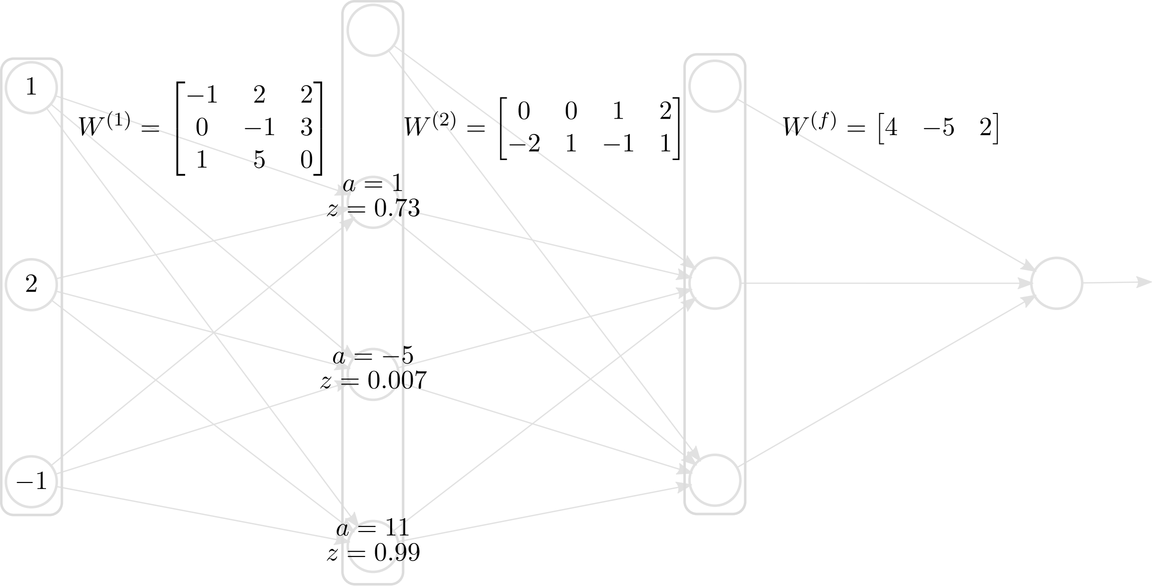


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

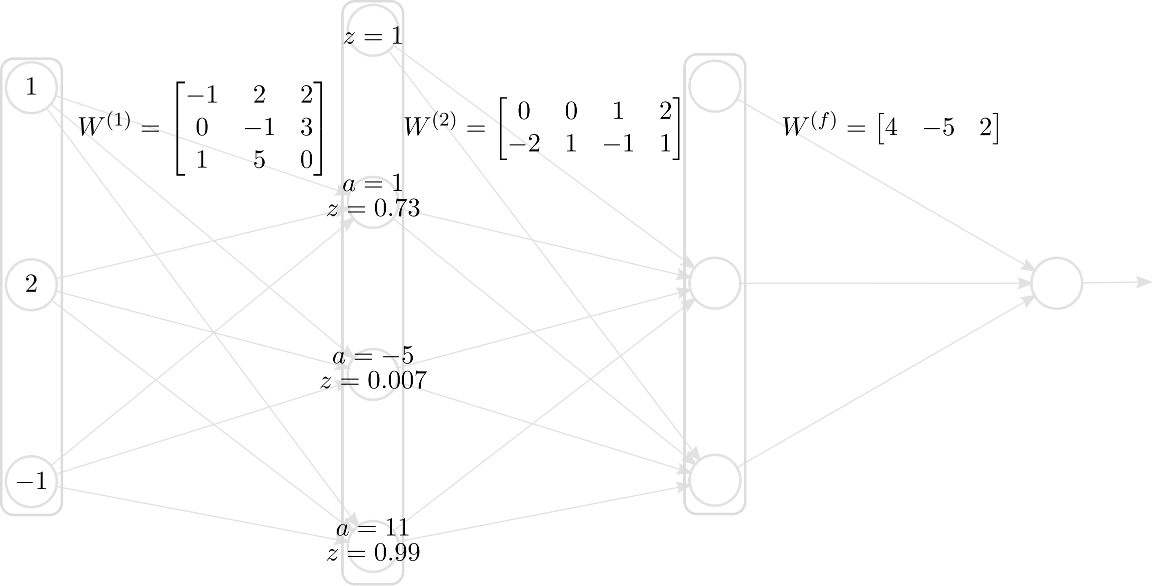


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

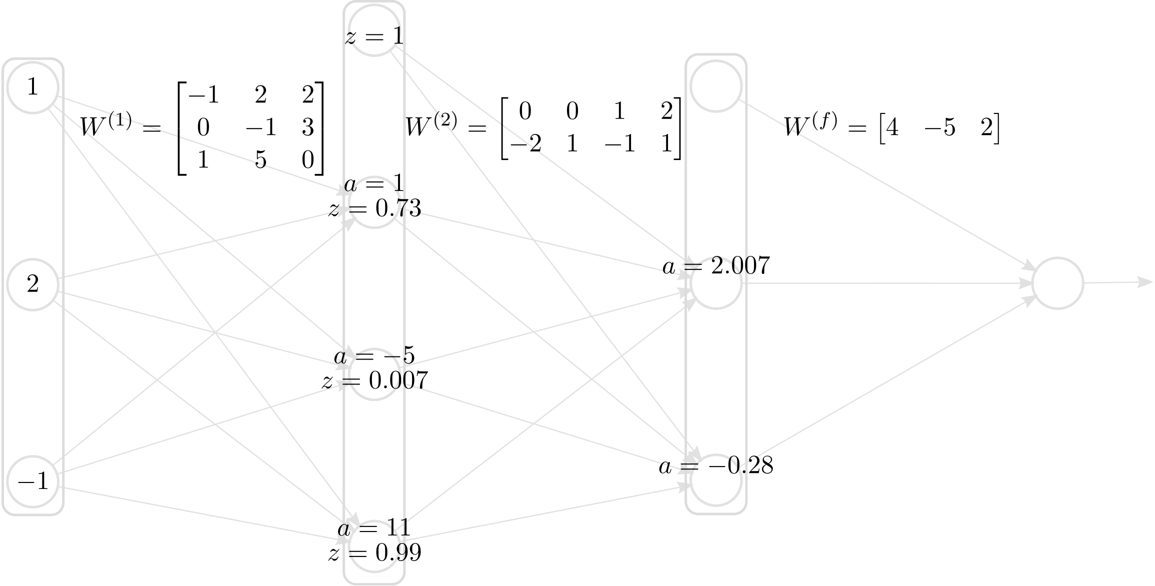


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

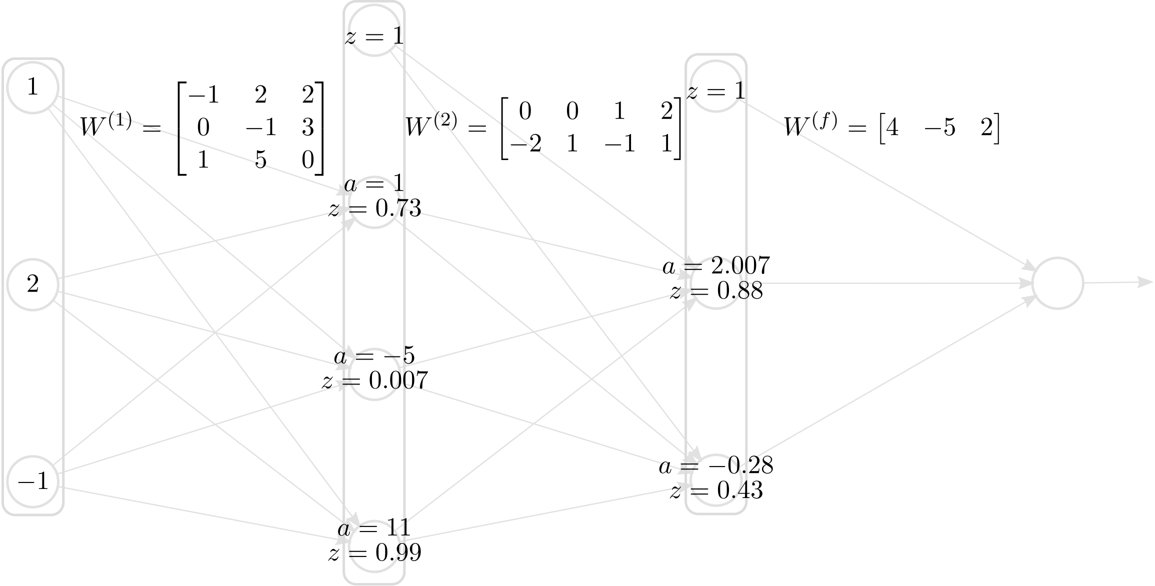


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation

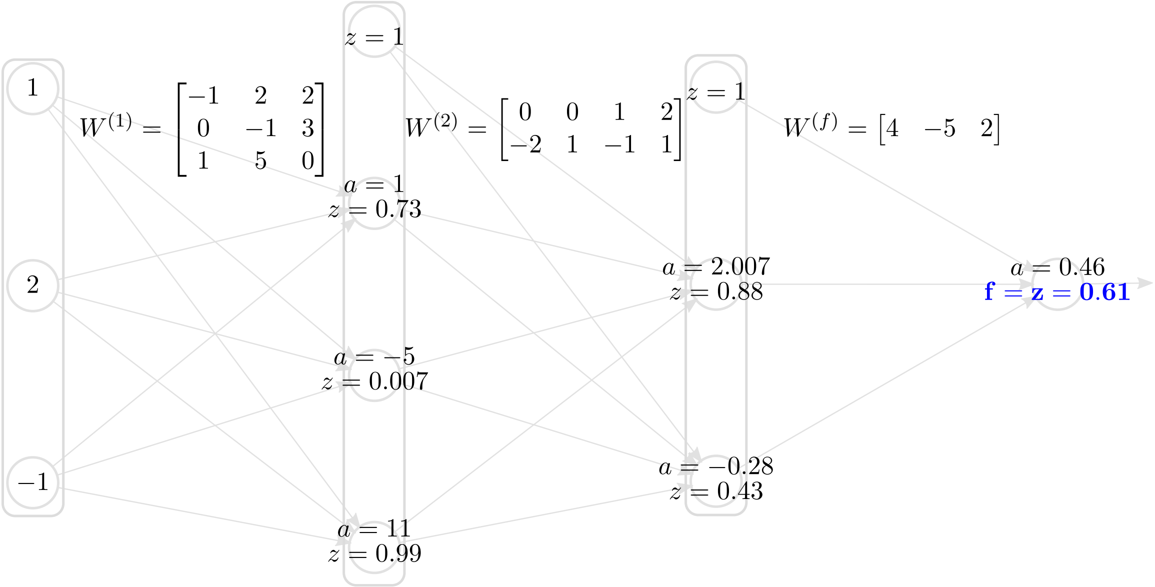


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, forward propagation



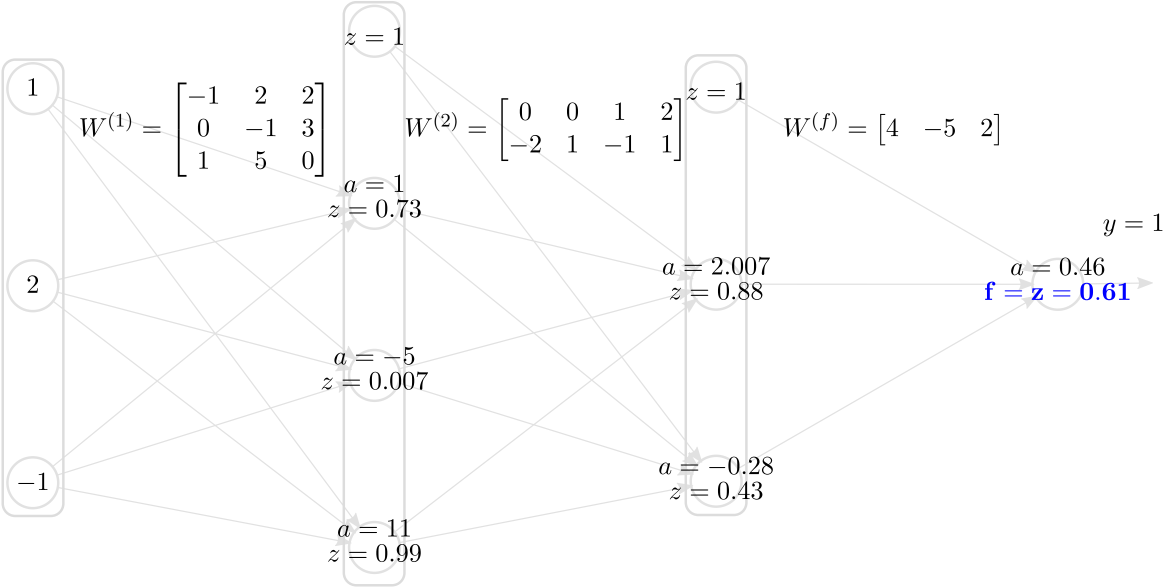
$$\mathbf{f = z = 0.61}$$

$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, backward propagation

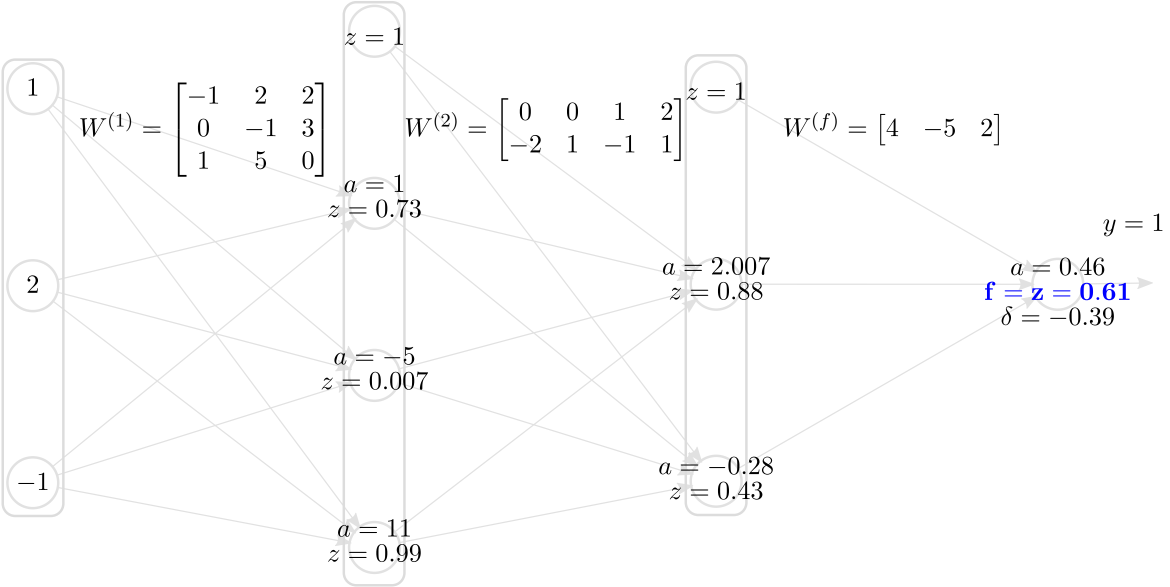


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, backward propagation

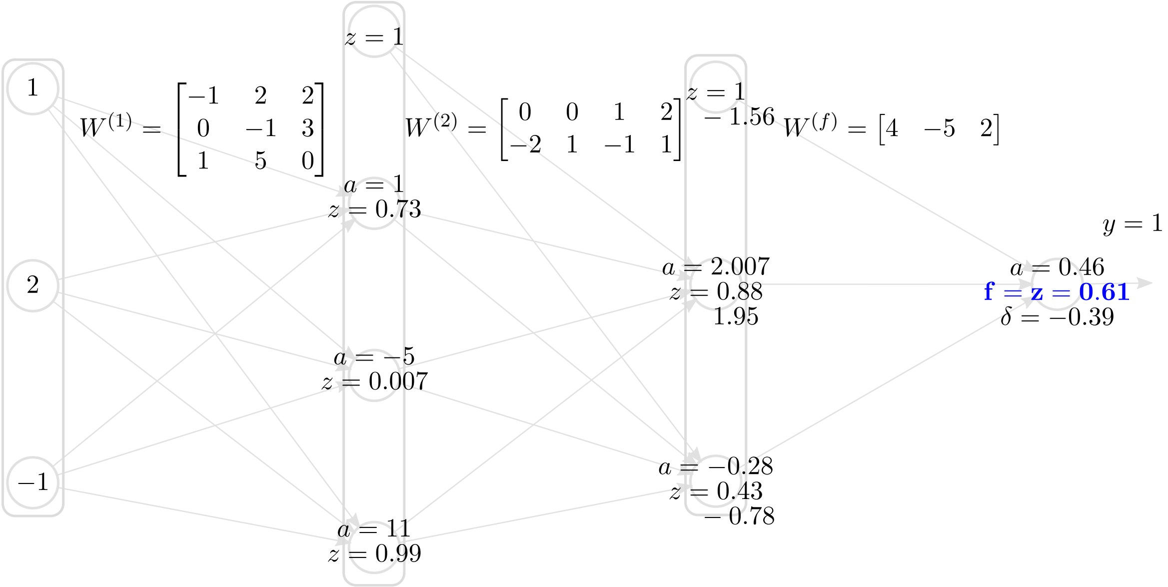


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, backward propagation



1

2

-1

$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$z = 1$

$a = 1$
$z = 0.73$

$a = -5$
$z = 0.007$

$a = 11$
$z = 0.99$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$z = 1$
-1.56

$a = 2.007$
$z = 0.88$
1.95

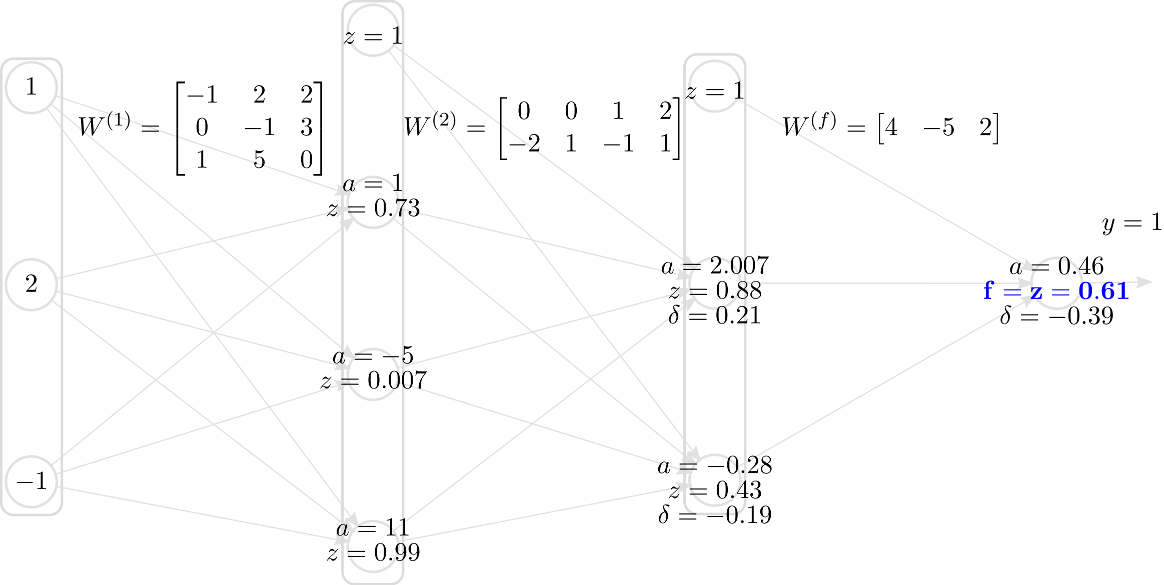
$a = -0.28$
$z = 0.43$
-0.78

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

$y = 1$

$a = 0.46$
$\mathbf{f = z = 0.61}$
$\delta = -0.39$

Layered Neural Network Example, backward propagation

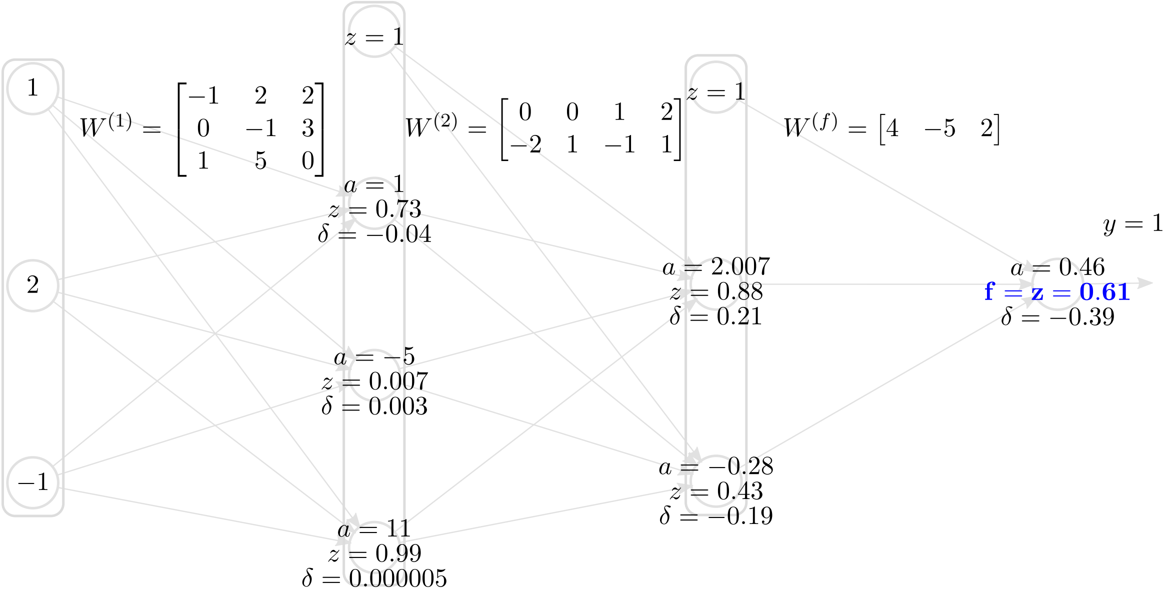


$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

Layered Neural Network Example, backward propagation



1

2

-1

$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$z = 1$

$a = 1$
$z = 0.73$
$\delta = -0.04$

$a = -5$
$z = 0.007$
$\delta = 0.003$

$a = 11$
$z = 0.99$
$\delta = 0.000005$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$z = 1$

$a = 2.007$
$z = 0.88$
$\delta = 0.21$

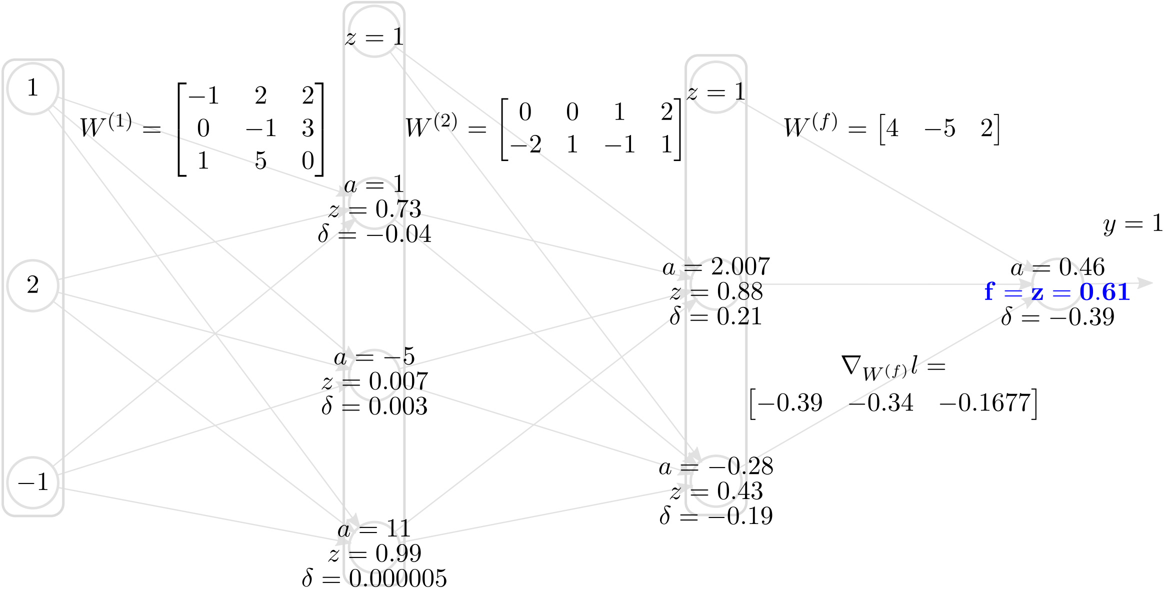
$a = -0.28$
$z = 0.43$
$\delta = -0.19$

$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

$y = 1$

$a = 0.46$
$\mathbf{f = z = 0.61}$
$\delta = -0.39$

Layered Neural Network Example, backward propagation



1

2

-1

$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$z = 1$

$a = 1$
$z = 0.73$
$\delta = -0.04$

$a = -5$
$z = 0.007$
$\delta = 0.003$

$a = 11$
$z = 0.99$
$\delta = 0.000005$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$z = 1$

$a = 2.007$
$z = 0.88$
$\delta = 0.21$

$a = -0.28$
$z = 0.43$
$\delta = -0.19$

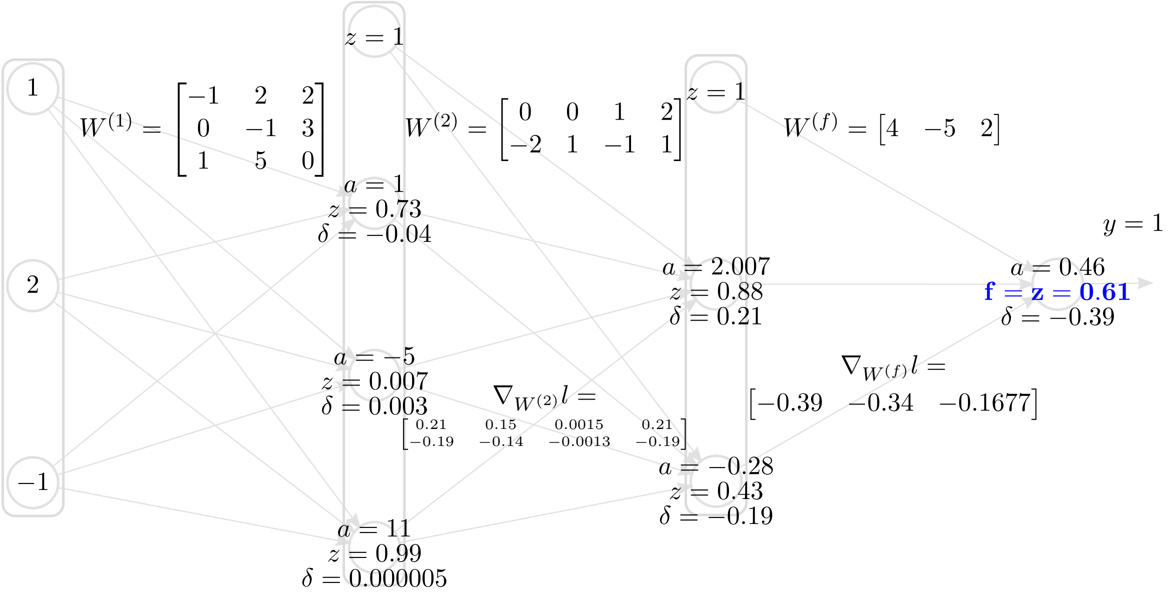
$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

$y = 1$

$a = 0.46$
$\mathbf{f = z = 0.61}$
$\delta = -0.39$

$$\nabla_{W^{(f)}} l = \begin{bmatrix} -0.39 & -0.34 & -0.1677 \end{bmatrix}$$

Layered Neural Network Example, backward propagation



1

2

-1

$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$z = 1$

$a = 1$
$z = 0.73$
$\delta = -0.04$

$a = -5$
$z = 0.007$
$\delta = 0.003$

$a = 11$
$z = 0.99$
$\delta = 0.000005$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

$$\nabla_{W^{(2)}} l = \begin{bmatrix} 0.21 & 0.15 & 0.0015 & 0.21 \\ -0.19 & -0.14 & -0.0013 & -0.19 \end{bmatrix}$$

$z = 1$

$a = 2.007$
$z = 0.88$
$\delta = 0.21$

$a = -0.28$
$z = 0.43$
$\delta = -0.19$

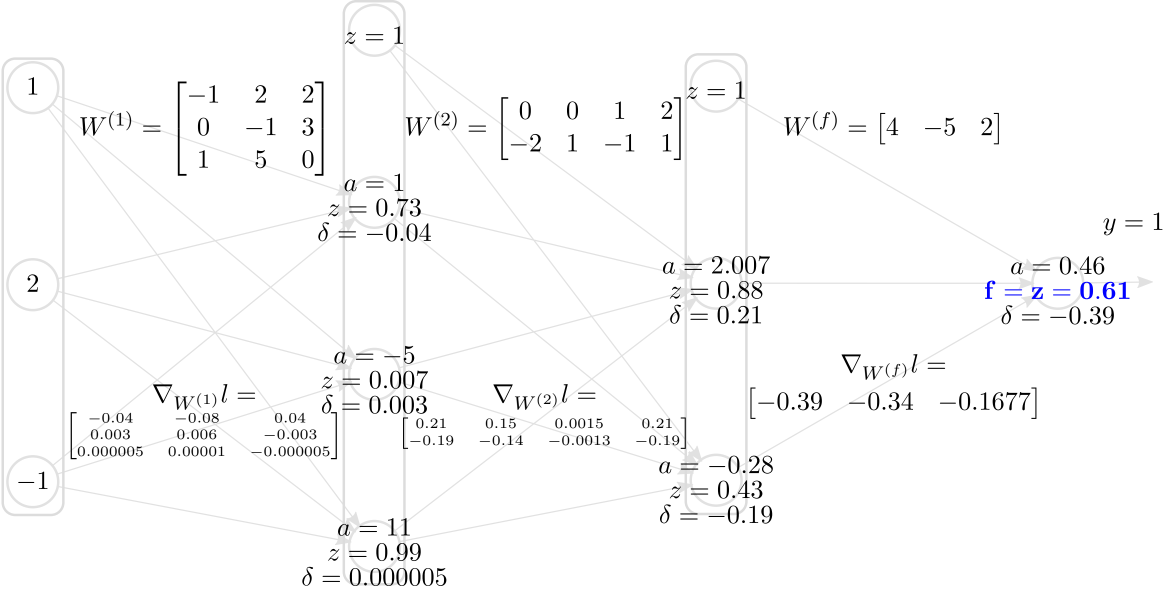
$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

$$\nabla_{W^{(f)}} l = \begin{bmatrix} -0.39 & -0.34 & -0.1677 \end{bmatrix}$$

$y = 1$

$a = 0.46$
$\mathbf{f = z = 0.61}$
$\delta = -0.39$

Layered Neural Network Example, backward propagation



$$W^{(1)} = \begin{bmatrix} -1 & 2 & 2 \\ 0 & -1 & 3 \\ 1 & 5 & 0 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ -2 & 1 & -1 & 1 \end{bmatrix}$$

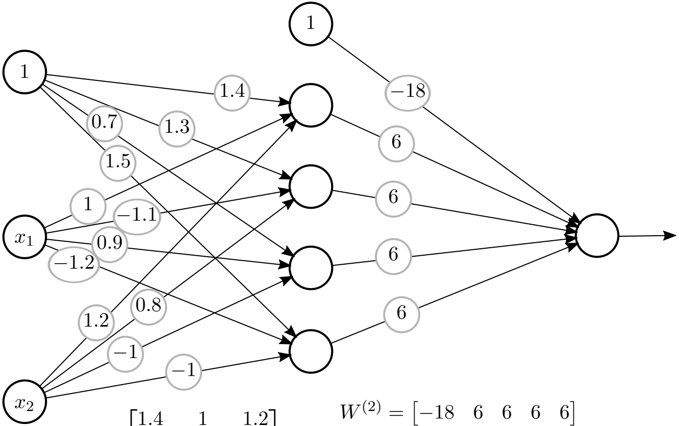
$$W^{(f)} = \begin{bmatrix} 4 & -5 & 2 \end{bmatrix}$$

$$\nabla_{W^{(1)}} l = \begin{bmatrix} -0.04 & -0.08 & 0.04 \\ 0.003 & 0.006 & -0.003 \\ 0.000005 & 0.00001 & -0.000005 \end{bmatrix}$$

$$\nabla_{W^{(2)}} l = \begin{bmatrix} 0.21 & 0.15 & 0.0015 & 0.21 \\ -0.19 & -0.14 & -0.0013 & -0.19 \end{bmatrix}$$

$$\nabla_{W^{(f)}} l = \begin{bmatrix} -0.39 & -0.34 & -0.1677 \end{bmatrix}$$

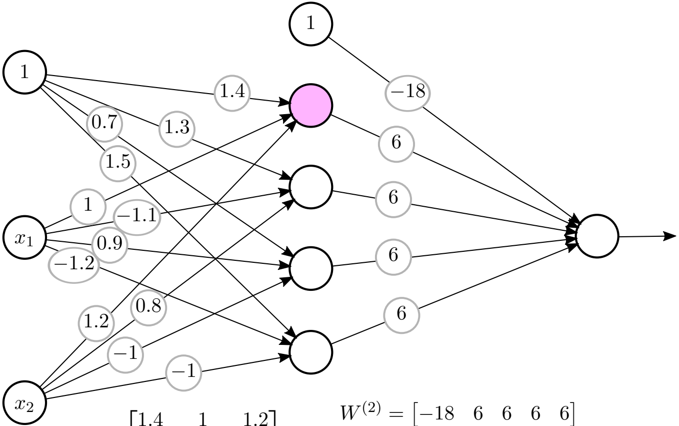
Function Approximation



$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

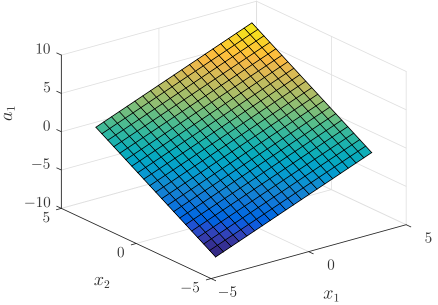
$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Function Approximation

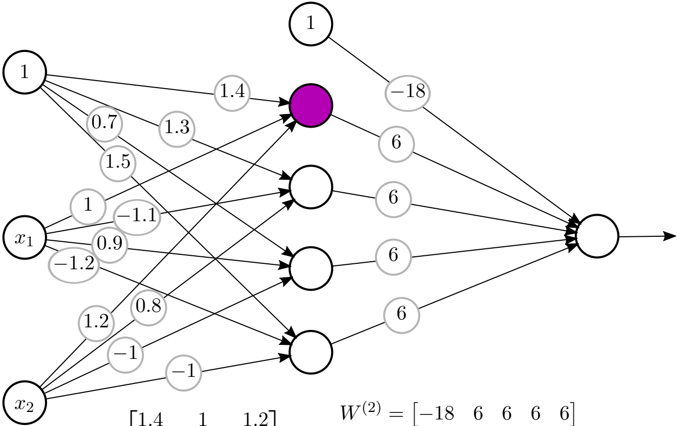


$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$

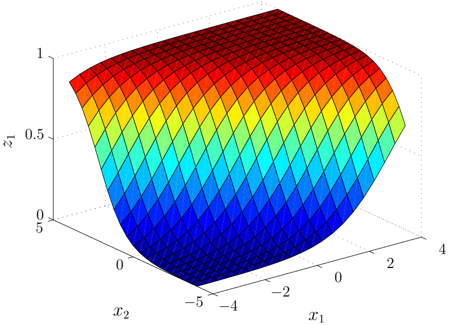


Function Approximation

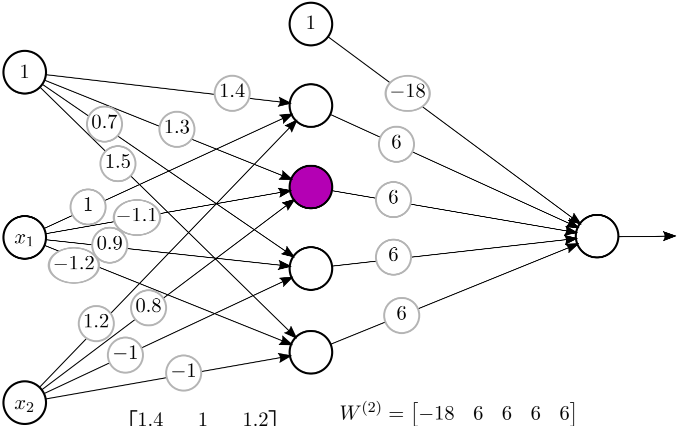


$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$

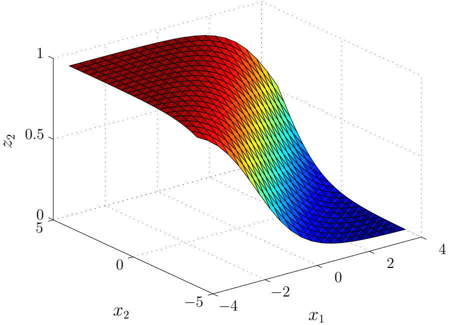


Function Approximation

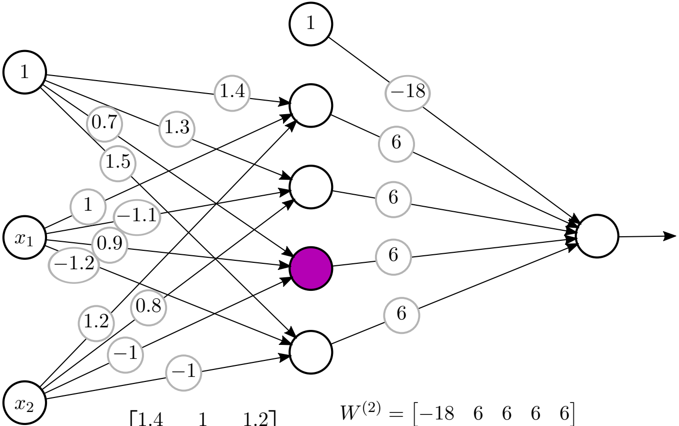


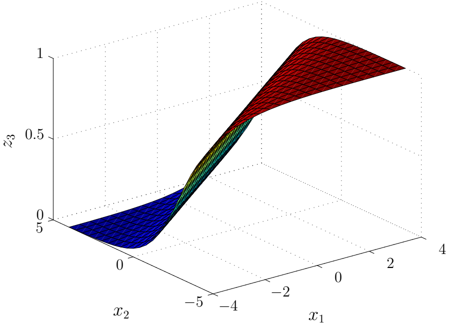
$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$



Function Approximation

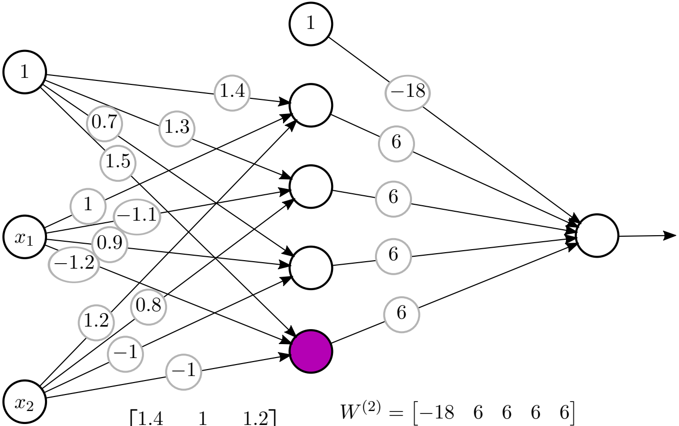




$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

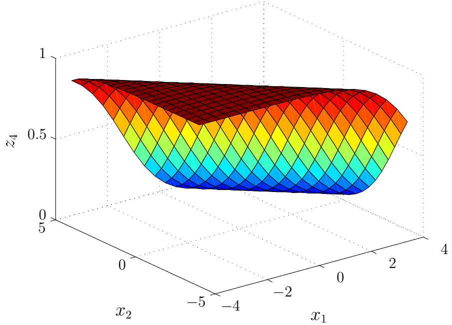
$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Function Approximation

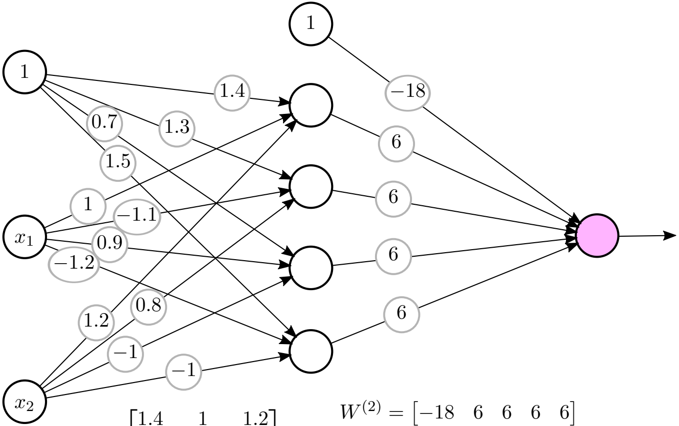


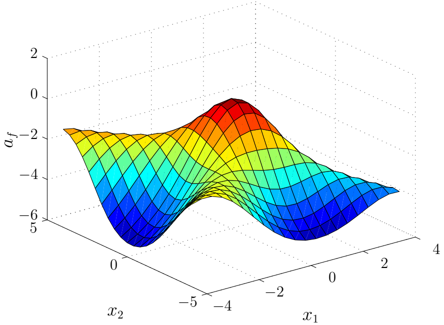
$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$



Function Approximation

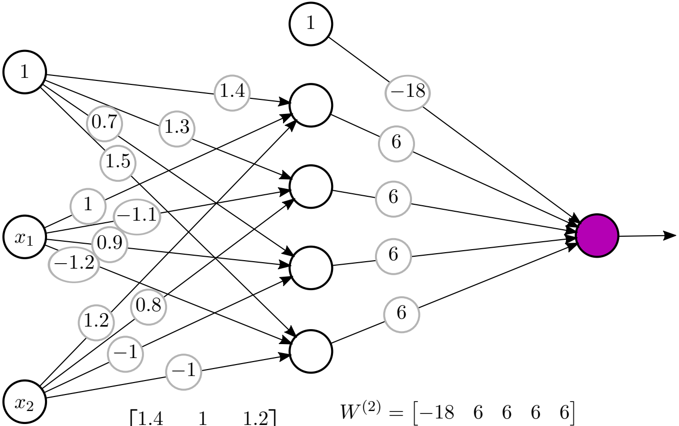


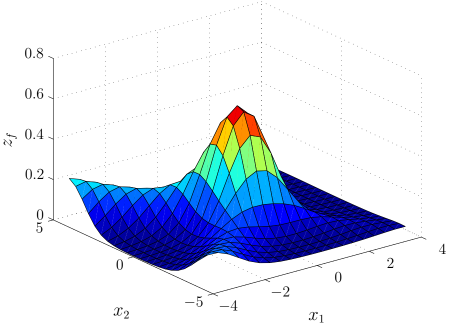


$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Function Approximation





$$W^{(1)} = \begin{bmatrix} 1.4 & 1 & 1.2 \\ 1.3 & -1.1 & 0.8 \\ 0.7 & 0.9 & -1 \\ 1.5 & -1.2 & -1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} -18 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Hidden Units

How many layers?

Hidden Units

How many layers?

- Historically, 2 (1 layer of hidden units)
- 2 can approximate any function (with enough hidden units)

Hidden Units

How many layers?

- Historically, 2 (1 layer of hidden units)
- 2 can approximate any function (with enough hidden units)
- More cause problems with minimization

Hidden Units

How many layers?

- Historically, 2 (1 layer of hidden units)
- 2 can approximate any function (with enough hidden units)
- More cause problems with minimization
- Now common to have many (deep learning)

Hidden Units

How many layers?

- Historically, 2 (1 layer of hidden units)
- 2 can approximate any function (with enough hidden units)
- More cause problems with minimization
- Now common to have many (deep learning)

How many units?

- Too many can cause overfitting...
- ...see next slide

Hidden Units

How many layers?

- Historically, 2 (1 layer of hidden units)
- 2 can approximate any function (with enough hidden units)
- More cause problems with minimization
- Now common to have many (deep learning)

How many units?

- Too many can cause overfitting...
- ...see next slide
- Usually err on side of too many

Overfitting

Start (stochastic) gradient descent:

- With weights near 0
- But, random!

Overfitting

Start (stochastic) gradient descent:

- With weights near 0
- But, random!

Add regularization to loss:

$$L = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \sum_{ijk} \left(w_{ij}^{(k)}\right)^2$$

Overfitting

Start (stochastic) gradient descent:

- With weights near 0
- But, random!

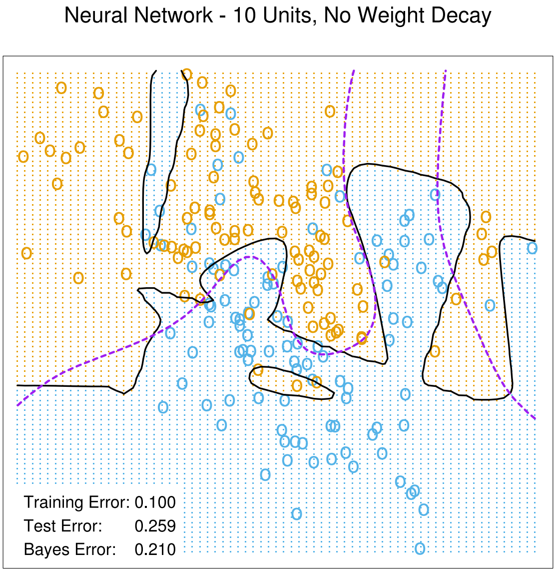
Add regularization to loss:

$$L = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \sum_{ijk} \left(w_{ij}^{(k)}\right)^2$$

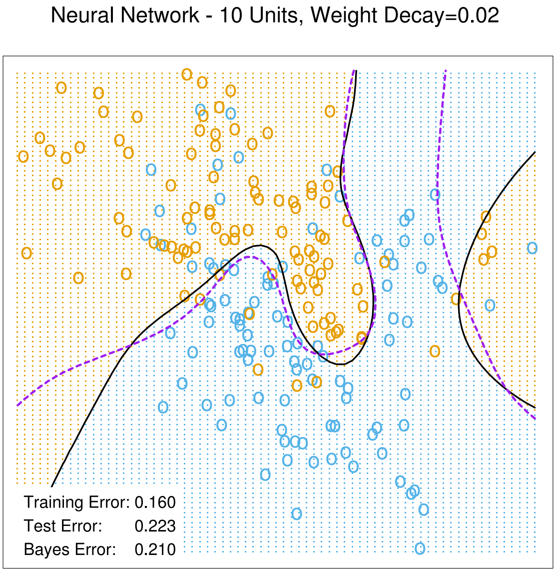
Same as adding $-\eta 2\lambda w_{ij}^{(k)}$ to update of $w_{ij}^{(k)}$

Called “weight decay”

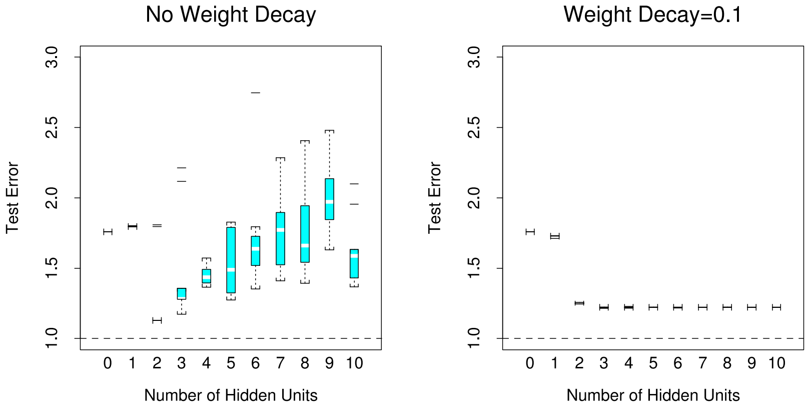
Example


Neural Network - 10 Units, No Weight Decay
Training Error: 0.100
Test Error: 0.259
Bayes Error: 0.210

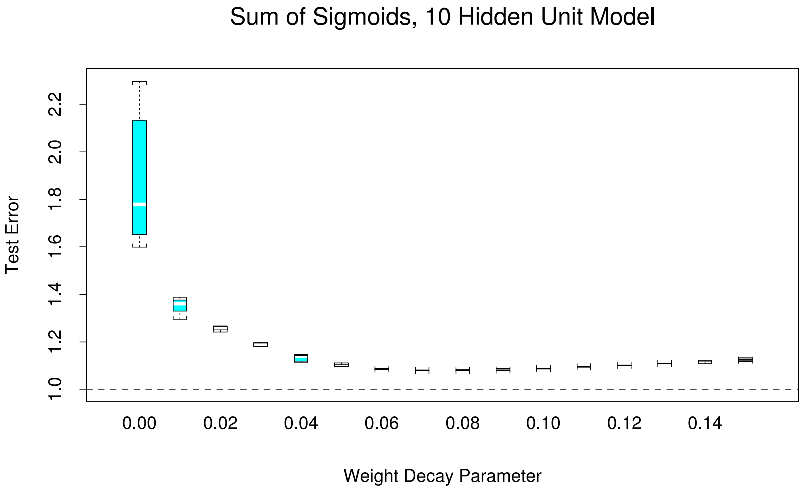
Example


Neural Network - 10 Units, Weight Decay=0.02
Training Error: 0.160
Test Error: 0.223
Bayes Error: 0.210

Sum of 2 Sigmoids



Sum of 2 Sigmoids



Optimization

Either batch (gradient descent) or online (stochastic gradient descent) used.

Optimization

Either batch (gradient descent) or online (stochastic gradient descent) used.

Many many local minima and saddle points...

Optimization

Either batch (gradient descent) or online (stochastic gradient descent) used.

Many many local minima and saddle points… use random restart

Optimization

Either batch (gradient descent) or online (stochastic gradient descent) used.

Many many local minima and saddle points... use random restart

Normalize data