COSI 165B
Deep Learning

Chuxu Zhang
Computer Science Department
Brandeis University

2/10/2021

More of Last Lecture

Universal Approximation Theorem

A feedforward network with a linear output layer and at least one hidden layer with non-linear activation function (e.g., sigmoid activation function) can approximate any measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units.

Refs

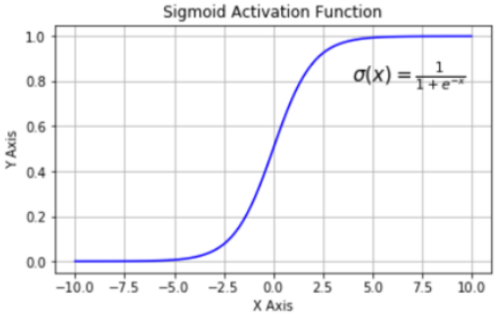
George Cybenko, Approximation by Superpositions of a Sigmoidal Function, Mathematics of control, signals and systems

Kurt Hornik et al., Multilayer Feedforward Networks are Universal Approximators, Neural networks

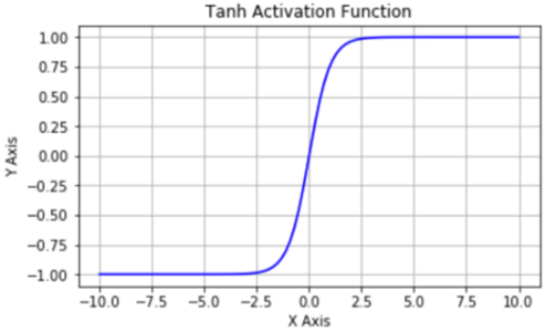
❑ Hidden Unit/Output Unit

Hidden Unit Activation

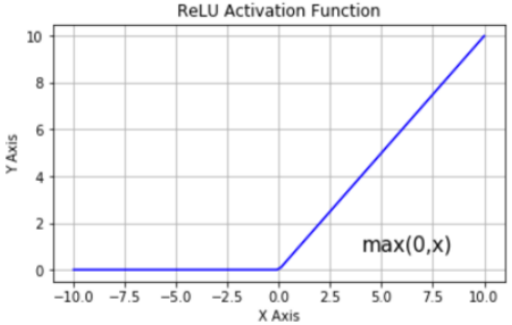
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



$$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max\{0, x\} = x\mathbf{1}_{x>0}$$



Output Unit

Linear

$$\hat{y} = \boldsymbol{W}^{\top}\boldsymbol{h} + \boldsymbol{b}.$$

Binary-label (Bernoulli)

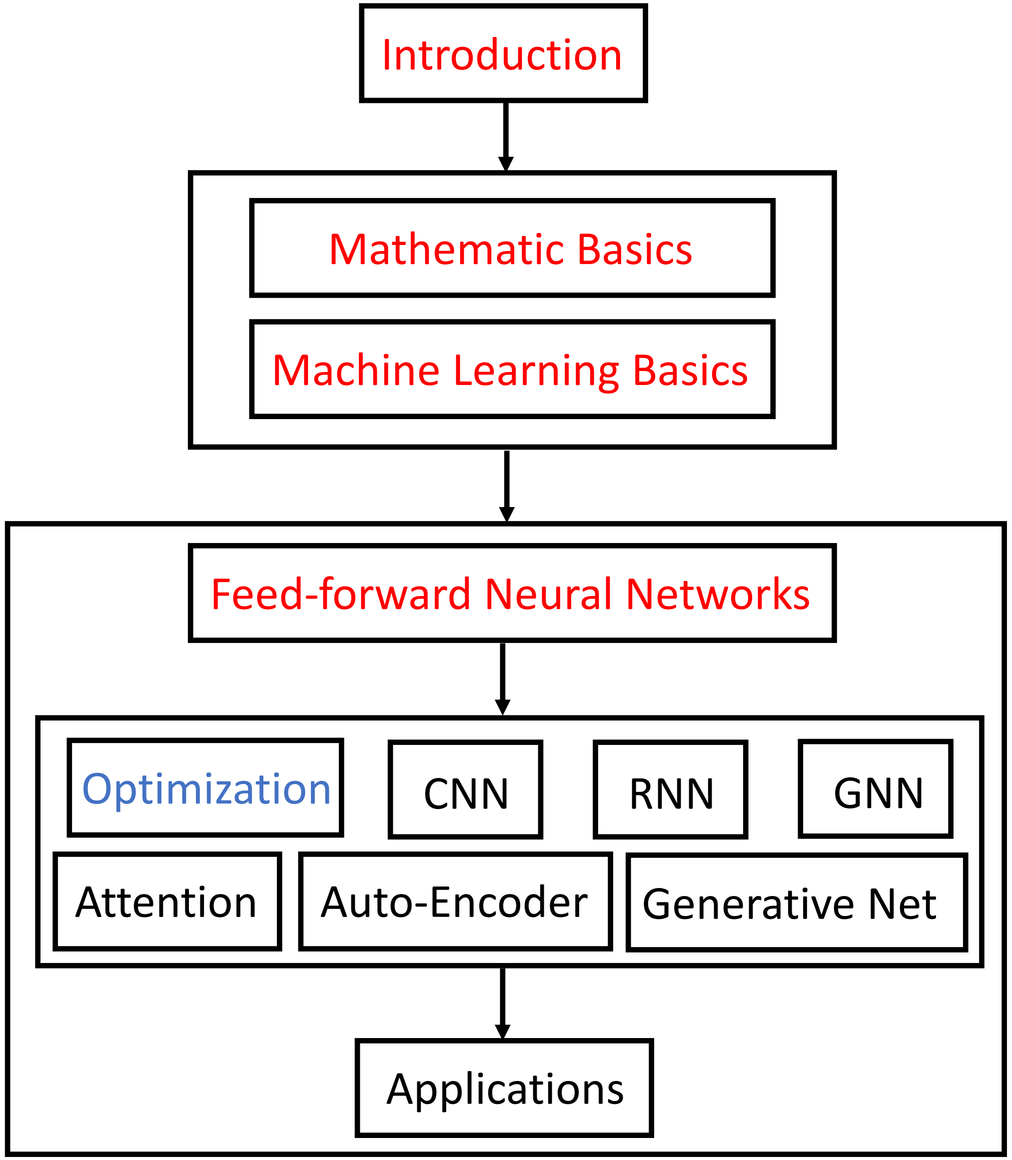
$$\hat{y} = \sigma\left(\boldsymbol{w}^{\top}\boldsymbol{h} + b\right)$$

Multi-label (Multinoulli)

$$\boldsymbol{z} = \boldsymbol{W}^{\top}\boldsymbol{h} + \boldsymbol{b},$$

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Structure of This Course



Brandeis UNIVERSITY

❑ Multi-layer Fully Connected Neural Network

$$
\begin{aligned}
a^{[1]} &= \mathrm{ReLU}(W^{[1]}x + b^{[1]}) \\
a^{[2]} &= \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]}) \\
&\cdots \\
a^{[r-1]} &= \mathrm{ReLU}(W^{[r-1]}a^{[r-2]} + b^{[r-1]}) \\
h_\theta(x) &= W^{[r]}a^{[r-1]} + b^{[r]}
\end{aligned}
$$

If $a^{[k]}$ has dimension m_k;

then the weight matrix $W^{[k]}$ should be of dimension $m_k \times m_{k-1}$, and the bias $b^{[k]} \in \mathbb{R}^{m_k}$. Moreover, $W^{[1]} \in \mathbb{R}^{m_1 \times d}$ and $W^{[r]} \in \mathbb{R}^{1 \times m_{r-1}}$.

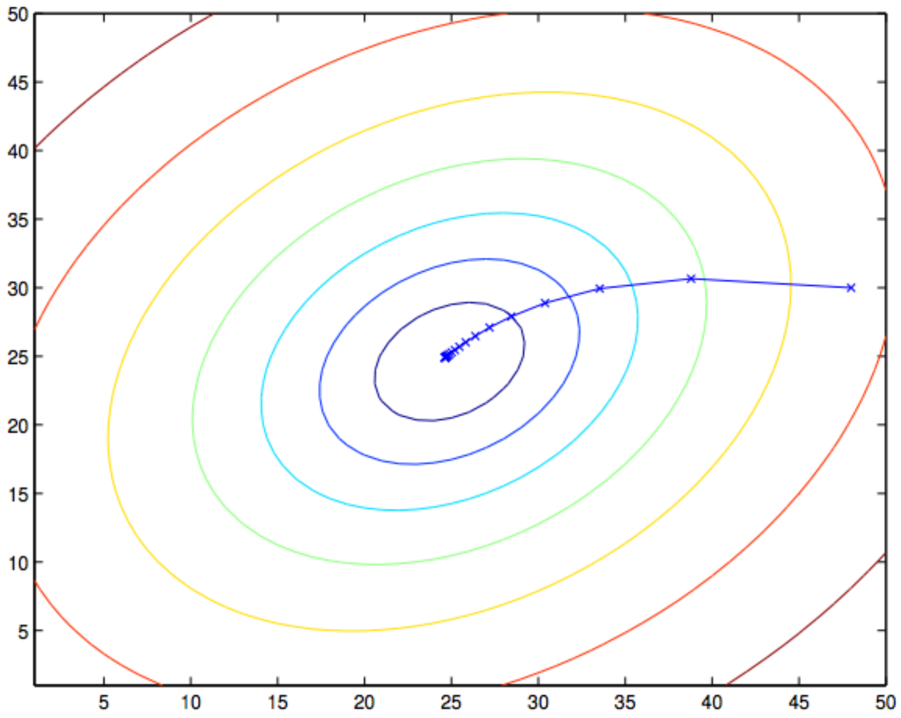
The total number of neurons in the network is $m_1 + \cdots + m_r$, and the total number of parameters in this network is $(d+1)m_1 + (m_1+1)m_2 + \cdots + (m_{r-1}+1)m_r$.

Gradient Descent

$$
\begin{aligned}
a^{[1]} &= \mathrm{ReLU}(W^{[1]}x + b^{[1]}) \\
a^{[2]} &= \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]}) \\
&\cdots \\
a^{[r-1]} &= \mathrm{ReLU}(W^{[r-1]}a^{[r-2]} + b^{[r-1]}) \\
h_\theta(x) &= W^{[r]}a^{[r-1]} + b^{[r]}
\end{aligned}
$$

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta).$$



Chain Rule

We first recall the chain rule in calculus. Suppose the variable J depends on the variables $\theta_1, \ldots, \theta_p$ via the intermediate variables $g_1, \ldots, g_k$:

$$g_j = g_j(\theta_1, \ldots, \theta_p), \forall j \in \{1, \cdots, k\}$$

$$J = J(g_1, \ldots, g_k)$$

$$\frac{\partial J}{\partial \theta_i} = \sum_{j=1}^{k} \frac{\partial J}{\partial g_j} \frac{\partial g_j}{\partial \theta_i}$$

Chain Rule: Example

$$J = J(g_1, \ldots, g_k)$$

$$\frac{\partial J}{\partial \theta_i} = \sum_{j=1}^{k} \frac{\partial J}{\partial g_j} \frac{\partial g_j}{\partial \theta_i}$$

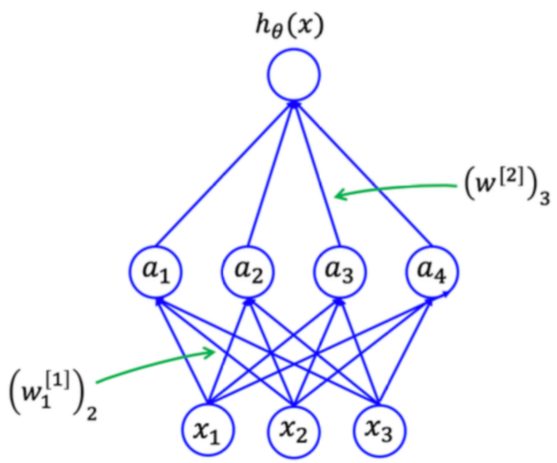
$$z = f(x, y) = 4x^2 + 3y^2,\ x = x(t) = \sin t,\ y = y(t) = \cos t$$

$$\frac{\partial z}{\partial x} = 8x$$
$$\frac{dx}{dt} = \cos t$$
$$\frac{\partial z}{\partial y} = 6y$$
$$\frac{dy}{dt} = -\sin t$$

$$\frac{dz}{dt} = \frac{\partial z}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial z}{\partial y} \cdot \frac{dy}{dt} = (8x)(\cos t) + (6y)(-\sin t) = 8x \cos t - 6y \sin t.$$

❑ Two-layer Neural Networks

$$z = W^{[1]}x + b^{[1]}$$
$$a = \mathrm{ReLU}(z)$$
$$h_\theta(x) \triangleq o = W^{[2]}a + b^{[2]}$$
$$J = \frac{1}{2}(y-o)^2$$



Computing $\frac{\partial J}{\partial W^{[2]}}$

$$\frac{\partial J}{\partial W_i^{[2]}} = \frac{\partial J}{\partial o} \cdot \frac{\partial o}{\partial W_i^{[2]}}$$

$$= (o-y) \cdot \frac{\partial o}{\partial W_i^{[2]}} \qquad \text{(because } o = \textstyle\sum_{i=1}^m W_i^{[2]} a_i + b^{[2]}\text{)}$$

$$= (o-y) \cdot a_i$$ ⟵ error (difference) multiplies input

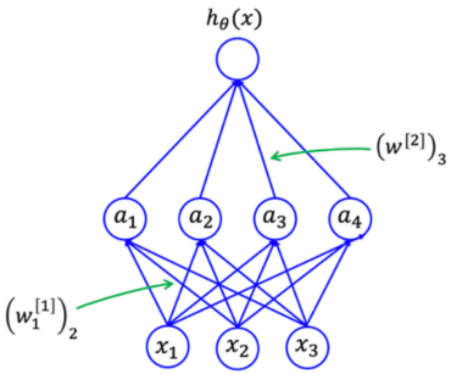
Vectorized notation $\frac{\partial J}{\partial W^{[2]}} = (o-y) \cdot a^\top \in \mathbb{R}^{1\times m}$

$$\frac{\partial J}{\partial b^{[2]}} = (o-y) \in \mathbb{R}$$ ⟵ error (difference)

Backpropagation

❑ Two-layer Neural Networks

$$z = W^{[1]}x + b^{[1]}$$
$$a = \mathrm{ReLU}(z)$$
$$h_\theta(x) \triangleq o = W^{[2]}a + b^{[2]}$$
$$J = \frac{1}{2}(y-o)^2$$



Computing $\frac{\partial J}{\partial W^{[2]}}$

$$\frac{\partial J}{\partial W_i^{[2]}} = \frac{\partial J}{\partial o} \cdot \frac{\partial o}{\partial W_i^{[2]}}$$
$$= (o-y) \cdot \frac{\partial o}{\partial W_i^{[2]}} \qquad \text{(because } o = \textstyle\sum_{i=1}^m W_i^{[2]} a_i + b^{[2]}\text{)}$$
$$= (o-y) \cdot a_i$$

Vectorized notation $\frac{\partial J}{\partial W^{[2]}} = (o-y) \cdot a^\top \in \mathbb{R}^{1\times m}$

$$\frac{\partial J}{\partial b^{[2]}} = (o-y) \in \mathbb{R}$$

Claim-1

$$z = Wu + b$$
$$J = J(z)$$

$$\frac{\partial J}{\partial W} = \frac{\partial J}{\partial z} \cdot u^\top$$
$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial z}$$

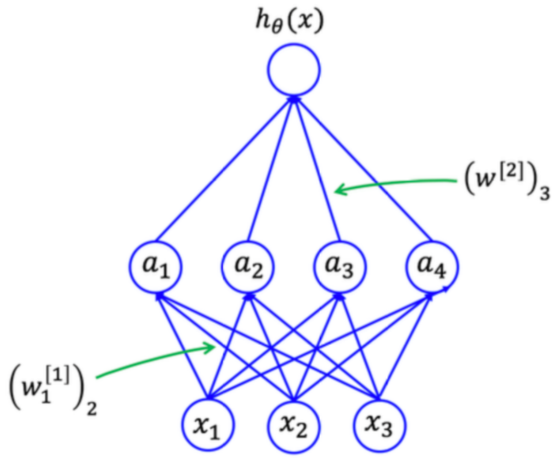
Two-layer Neural Networks

$$z = W^{[1]}x + b^{[1]}$$
$$a = \mathrm{ReLU}(z)$$
$$h_\theta(x) \triangleq o = W^{[2]}a + b^{[2]}$$
$$J = \frac{1}{2}(y-o)^2$$

Computing $\frac{\partial J}{\partial W^{[1]}}$

$$\frac{\partial J}{\partial W_{ij}^{[1]}} = \frac{\partial J}{\partial z_i} \cdot \frac{\partial z_i}{\partial W_{ij}^{[1]}} \quad \left(\text{because } z_i = \textstyle\sum_{k=1}^{d} W_{ik}^{[1]} x_k + b_i^{[1]}\right)$$
$$= \frac{\partial J}{\partial z_i} \cdot x_j$$

Vectorized notation $\frac{\partial J}{\partial W^{[1]}} = \frac{\partial J}{\partial z} \cdot x^\top$



Computing $\frac{\partial J}{\partial z}$

$$\frac{\partial J}{\partial z_i} = \frac{\partial J}{\partial a_i}\frac{\partial a_i}{\partial z_i}$$
$$= \frac{\partial J}{\partial a_i} \cdot 1\{z_i \geq 0\}$$

Computing $\frac{\partial J}{\partial a}$

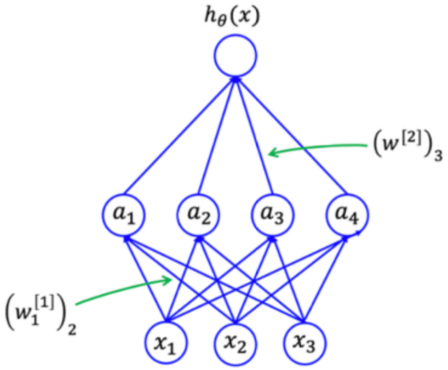
$$\frac{\partial J}{\partial a_i} = \frac{\partial J}{\partial o}\frac{\partial o}{\partial a_i} \quad \left(\text{because } o = \textstyle\sum_{i=1}^{m} W_i^{[2]} a_i + b^{[2]}\right)$$
$$= (o-y) \cdot W_i^{[2]}$$

Vectorized notation

$$\frac{\partial J}{\partial a} = {W^{[2]}}^\top \cdot (o-y)$$

❑ Two-layer Neural Networks

$$z = W^{[1]}x + b^{[1]}$$
$$a = \mathrm{ReLU}(z)$$
$$h_\theta(x) \triangleq o = W^{[2]}a + b^{[2]}$$
$$J = \frac{1}{2}(y-o)^2$$



Computing $\frac{\partial J}{\partial W^{[1]}}$

$$\frac{\partial J}{\partial W_{ij}^{[1]}} = \frac{\partial J}{\partial z_i} \cdot \frac{\partial z_i}{\partial W_{ij}^{[1]}} \quad \left(\text{because } z_i = \textstyle\sum_{k=1}^{d} W_{ik}^{[1]} x_k + b_i^{[1]}\right)$$
$$= \frac{\partial J}{\partial z_i} \cdot x_j$$

Vectorized notation $\frac{\partial J}{\partial W^{[1]}} = \frac{\partial J}{\partial z} \cdot x^\top$

Computing $\frac{\partial J}{\partial z}$

$$\frac{\partial J}{\partial z_i} = \frac{\partial J}{\partial a_i}\frac{\partial a_i}{\partial z_i}$$
$$= \frac{\partial J}{\partial a_i} \cdot 1\{z_i \geq 0\}$$

Computing $\frac{\partial J}{\partial a}$

$$\frac{\partial J}{\partial a_i} = \frac{\partial J}{\partial o}\frac{\partial o}{\partial a_i} \quad \left(\text{because } o = \textstyle\sum_{i=1}^{m} W_i^{[2]} a_i + b^{[2]}\right)$$
$$= (o-y) \cdot W_i^{[2]}$$

Vectorized notation

$$\frac{\partial J}{\partial a} = W^{[2]\top} \cdot (o-y)$$

Claim-2

$$a = \sigma(z),$$
$$J = J(a)$$
$$\frac{\partial J}{\partial z} = \frac{\partial J}{\partial a} \odot \sigma'(z)$$

Claim-3

$$v = Wu + b$$
$$J = J(v)$$
$$\frac{\partial J}{\partial u} = W^\top \frac{\partial J}{\partial v}$$

Brandeis UNIVERSITY

Two-layer Neural Networks

1: Compute the values of $z \in \mathbb{R}^m$, $a \in \mathbb{R}^m$, and $o \in \mathbb{R}$

2: Compute

$$\delta^{[2]} \triangleq \frac{\partial J}{\partial o} = (o - y) \in \mathbb{R}$$

$$\delta^{[1]} \triangleq \frac{\partial J}{\partial z} = ({W^{[2]}}^\top (o - y)) \odot 1\{z \geq 0\} \in \mathbb{R}^{m \times 1}$$

3: Compute

$$\frac{\partial J}{\partial W^{[2]}} = \delta^{[2]} a^\top \in \mathbb{R}^{1 \times m}$$

$$\frac{\partial J}{\partial b^{[2]}} = \delta^{[2]} \in \mathbb{R}$$

$$\frac{\partial J}{\partial W^{[1]}} = \delta^{[1]} x^\top \in \mathbb{R}^{m \times d}$$

$$\frac{\partial J}{\partial b^{[1]}} = \delta^{[1]} \in \mathbb{R}^m$$

Computing $\frac{\partial J}{\partial W^{[2]}}$

$$\frac{\partial J}{\partial W_i^{[2]}} = \frac{\partial J}{\partial o} \cdot \frac{\partial o}{\partial W_i^{[2]}}$$

$$= (o - y) \cdot \frac{\partial o}{\partial W_i^{[2]}} \qquad \left(\text{because } o = \sum_{i=1}^m W_i^{[2]} a_i + b^{[2]}\right)$$

$$= (o - y) \cdot a_i$$ ← error (difference) multiplies input

Vectorized notation $\frac{\partial J}{\partial W^{[2]}} = (o - y) \cdot a^\top \in \mathbb{R}^{1 \times m}$

$$\frac{\partial J}{\partial b^{[2]}} = (o - y) \in \mathbb{R}$$ ← error (difference)

Computing $\frac{\partial J}{\partial W^{[1]}}$

$$\frac{\partial J}{\partial W_{ij}^{[1]}} = \frac{\partial J}{\partial z_i} \cdot \frac{\partial z_i}{\partial W_{ij}^{[1]}} \qquad \left(\text{because } z_i = \sum_{k=1}^d W_{ik}^{[1]} x_k + b_i^{[1]}\right)$$

$$= \frac{\partial J}{\partial z_i} \cdot x_j$$

Vectorized notation $\frac{\partial J}{\partial W^{[1]}} = \frac{\partial J}{\partial z} \cdot x^\top$

Computing $\frac{\partial J}{\partial z}$

$$\frac{\partial J}{\partial z_i} = \frac{\partial J}{\partial a_i} \frac{\partial a_i}{\partial z_i}$$

$$= \frac{\partial J}{\partial a_i} \cdot 1\{z_i \geq 0\}$$

Computing $\frac{\partial J}{\partial a}$

$$\frac{\partial J}{\partial a_i} = \frac{\partial J}{\partial o} \frac{\partial o}{\partial a_i} \qquad \left(\text{because } o = \sum_{i=1}^m W_i^{[2]} a_i + b^{[2]}\right)$$

$$= (o - y) \cdot W_i^{[2]}$$

Vectorized notation

$$\frac{\partial J}{\partial a} = {W^{[2]}}^\top \cdot (o - y)$$

Backpropagation

❑ Multi-layer Neural Networks

$$a^{[0]} = x$$
$$a^{[1]} = \mathrm{ReLU}(W^{[1]}a^{[0]} + b^{[1]})$$
$$a^{[2]} = \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]})$$
$$\cdots$$
$$a^{[r-1]} = \mathrm{ReLU}(W^{[r-1]}a^{[r-2]} + b^{[r-1]})$$
$$a^{[r]} = z^{[r]} = W^{[r]}a^{[r-1]} + b^{[r]}$$
$$J = \frac{1}{2}(a^{[r]} - y)^2$$

$$z^{[k]} = W^{[k]}a^{[k-1]} + b^{[k]}$$
$$J = J(z^{[k]})$$
$$\frac{\partial J}{\partial W^{[k]}} = \frac{\partial J}{\partial z^{[k]}} \cdot a^{[k-1]^\top}$$
$$\frac{\partial J}{\partial b^{[k]}} = \frac{\partial J}{\partial z^{[k]}}$$

$$\delta^{[k]} \triangleq \frac{\partial J}{\partial z^{[k]}} \longleftarrow \text{error term}$$

$$\delta^{[r]} \triangleq \frac{\partial J}{\partial z^{[r]}} = (z^{[r]} - y)$$

(k = r: output layer)

Claim-1

$$z = Wu + b$$
$$J = J(z)$$

$$\frac{\partial J}{\partial W} = \frac{\partial J}{\partial z} \cdot u^\top$$
$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial z}$$

Backpropagation

❑ Multi-layer Neural Networks

$$a^{[0]} = x$$
$$a^{[1]} = \mathrm{ReLU}(W^{[1]}a^{[0]} + b^{[1]})$$
$$a^{[2]} = \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]})$$
$$\cdots$$
$$a^{[r-1]} = \mathrm{ReLU}(W^{[r-1]}a^{[r-2]} + b^{[r-1]})$$
$$a^{[r]} = z^{[r]} = W^{[r]}a^{[r-1]} + b^{[r]}$$
$$J = \frac{1}{2}(a^{[r]} - y)^2$$

(k < r: hidden layer)

$$\delta^{[k]} \triangleq \frac{\partial J}{\partial z^{[k]}} = \frac{\partial J}{\partial a^{[k]}} \odot \mathrm{ReLU}'(z^{[k]})$$

Claim-2

$$a = \sigma(z),$$
$$J = J(a)$$
$$\frac{\partial J}{\partial z} = \frac{\partial J}{\partial a} \odot \sigma'(z)$$

$$z^{[k+1]} = W^{[k+1]}a^{[k]} + b^{[k+1]}$$
$$J = J(z^{[k+1]})$$
$$\frac{\partial J}{\partial a^{[k]}} = {W^{[k+1]}}^\top \frac{\partial J}{\partial z^{[k+1]}}$$
$$\delta^{[k]} = \left({W^{[k+1]}}^\top \frac{\partial J}{\partial z^{[k+1]}}\right) \odot \mathrm{ReLU}'(z^{[k]})$$
$$= \left({W^{[k+1]}}^\top \delta^{[k+1]}\right) \odot \mathrm{ReLU}'(z^{[k]})$$

Claim-3

$$v = Wu + b$$
$$J = J(v)$$
$$\frac{\partial J}{\partial u} = W^\top \frac{\partial J}{\partial v}$$

Brandeis University

❑ Multi-layer Neural Networks

.

1: Compute and store the values of $a^{[k]}$'s and $z^{[k]}$'s for $k = 1, \ldots, r$, and J. ▷ This is often called the "forward pass"
.

2: .

3: **for** $k = r$ to 1 **do** ▷ This is often called the "backward pass"

4: **if** $k = r$ **then**

5: compute $\delta^{[r]} \triangleq \frac{\partial J}{\partial z^{[r]}}$

6: **else**

7: compute

$$\delta^{[k]} \triangleq \frac{\partial J}{\partial z^{[k]}} = \left(W^{[k+1]^\top} \delta^{[k+1]} \right) \odot \mathrm{ReLU}'(z^{[k]})$$

8: Compute

$$\frac{\partial J}{\partial W^{[k]}} = \delta^{[k]} a^{[k-1]^\top}$$
$$\frac{\partial J}{\partial b^{[k]}} = \delta^{[k]}$$

$$z^{[k+1]} = W^{[k+1]} a^{[k]} + b^{[k+1]}$$
$$J = J(z^{[k+1]})$$
$$\frac{\partial J}{\partial a^{[k]}} = W^{[k+1]^\top} \frac{\partial J}{\partial z^{[k+1]}}$$
$$\delta^{[k]} = \left(W^{[k+1]^\top} \frac{\partial J}{\partial z^{[k+1]}} \right) \odot \mathrm{ReLU}'(z^{[k]})$$
$$= \left(W^{[k+1]^\top} \delta^{[k+1]} \right) \odot \mathrm{ReLU}'(z^{[k]})$$

Brandeis UNIVERSITY

Multi-layer Neural Networks

1: Compute the values of $z \in \mathbb{R}^m$, $a \in \mathbb{R}^m$, and $o \in \mathbb{R}$
2: Compute

$$\delta^{[2]} \triangleq \frac{\partial J}{\partial o} = (o - y) \in \mathbb{R}$$
$$\delta^{[1]} \triangleq \frac{\partial J}{\partial z} = (W^{[2]^\top}(o - y)) \odot 1\{z \geq 0\} \in \mathbb{R}^{m \times 1}$$

(by eqn. (3.12) and (3.13))

3: Compute

$$\frac{\partial J}{\partial W^{[2]}} = \delta^{[2]} a^\top \in \mathbb{R}^{1 \times m} \quad \text{(by eqn. (3.5))}$$
$$\frac{\partial J}{\partial b^{[2]}} = \delta^{[2]} \in \mathbb{R} \quad \text{(by eqn. (3.6))}$$
$$\frac{\partial J}{\partial W^{[1]}} = \delta^{[1]} x^\top \in \mathbb{R}^{m \times d} \quad \text{(by eqn. (3.7))}$$
$$\frac{\partial J}{\partial b^{[1]}} = \delta^{[1]} \in \mathbb{R}^m \quad \text{(as an exercise)}$$

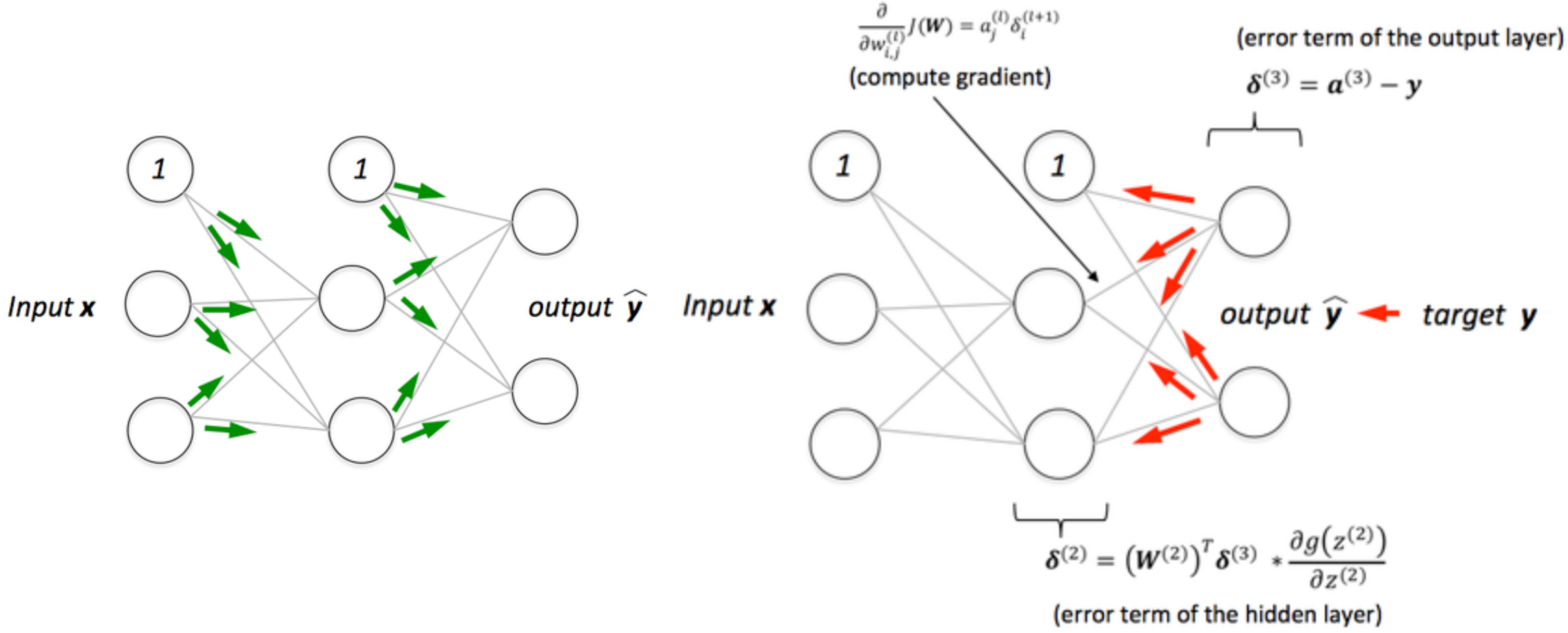
1: Compute and store the values of $a^{[k]}$'s and $z^{[k]}$'s for $k = 1, \ldots, r$, and J.
. ▷ This is often called the "forward pass"
2: .
3: **for** $k = r$ to 1 **do** ▷ This is often called the "backward pass"
4: **if** $k = r$ **then**
5: compute $\delta^{[r]} \triangleq \frac{\partial J}{\partial z^{[r]}}$
6: **else**
7: compute

$$\delta^{[k]} \triangleq \frac{\partial J}{\partial z^{[k]}} = \left(W^{[k+1]^\top} \delta^{[k+1]}\right) \odot \mathrm{ReLU}'(z^{[k]})$$

8: Compute

$$\frac{\partial J}{\partial W^{[k]}} = \delta^{[k]} a^{[k-1]^\top}$$
$$\frac{\partial J}{\partial b^{[k]}} = \delta^{[k]}$$

❑ Visualization Example



In backpropagation, we "simply" backpropagate the error to update model parameters.

Backpropagation

- Homework-1 will be released in weekend (2 weeks time)
- Deep learning tool: **Pytorch**/Tensorflow (TA lecture, next Wed.)

Q & A