University of Colorado **Boulder**

Department of Computer Science

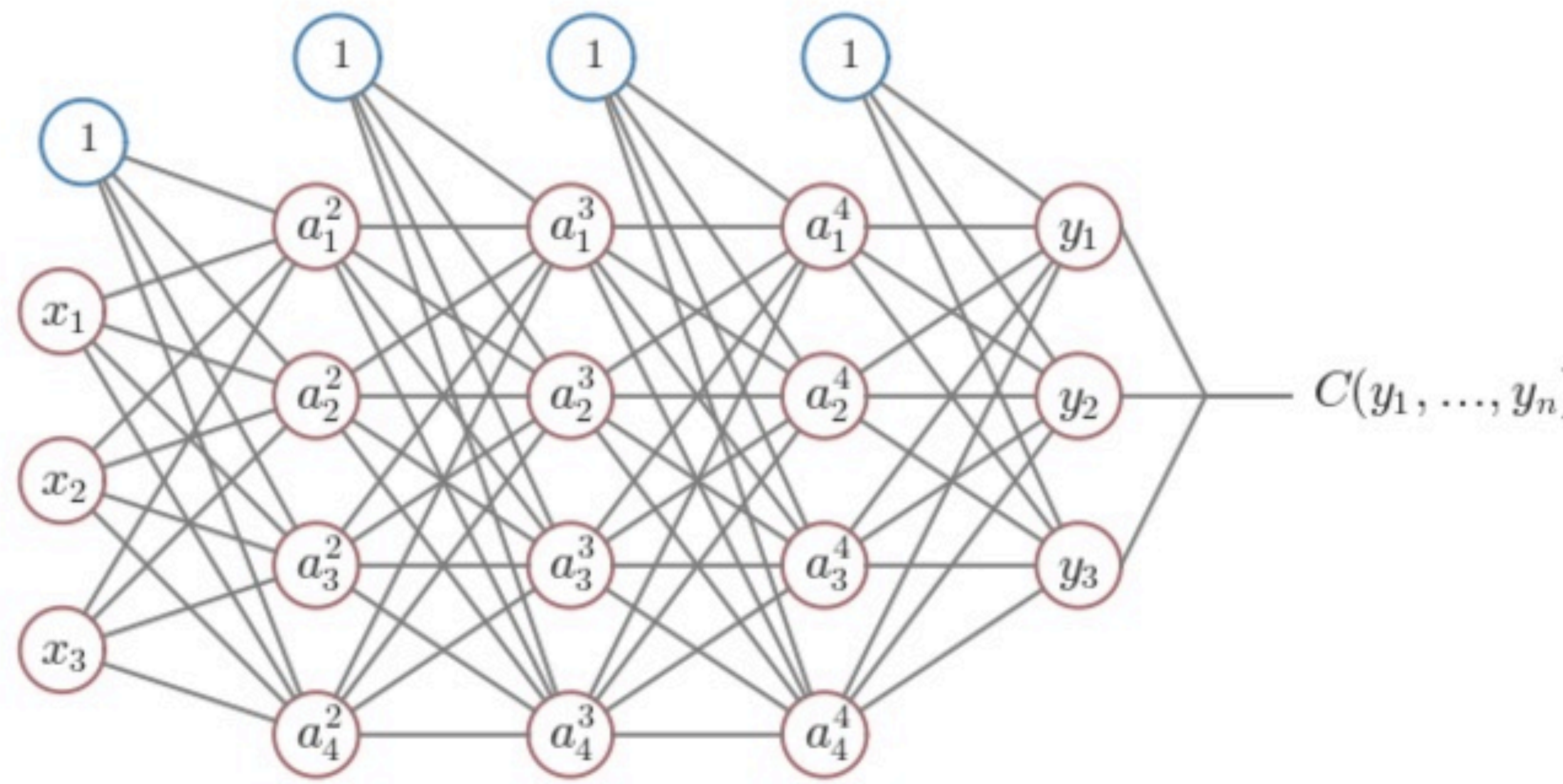
CSCI 5622: Machine Learning

Chris Ketelsen

Lecture 18:
Neural Networks Part 2

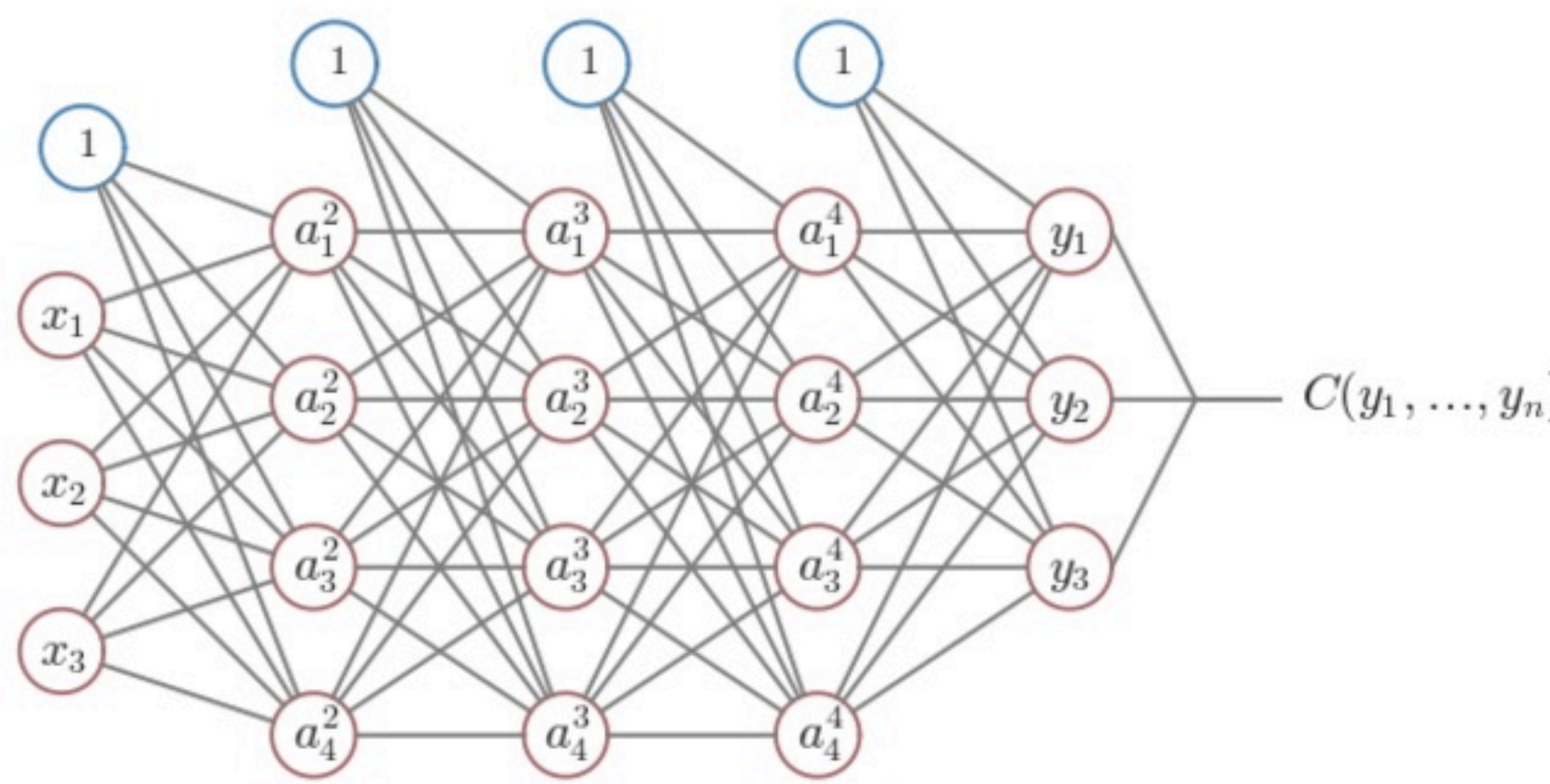
Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network



Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

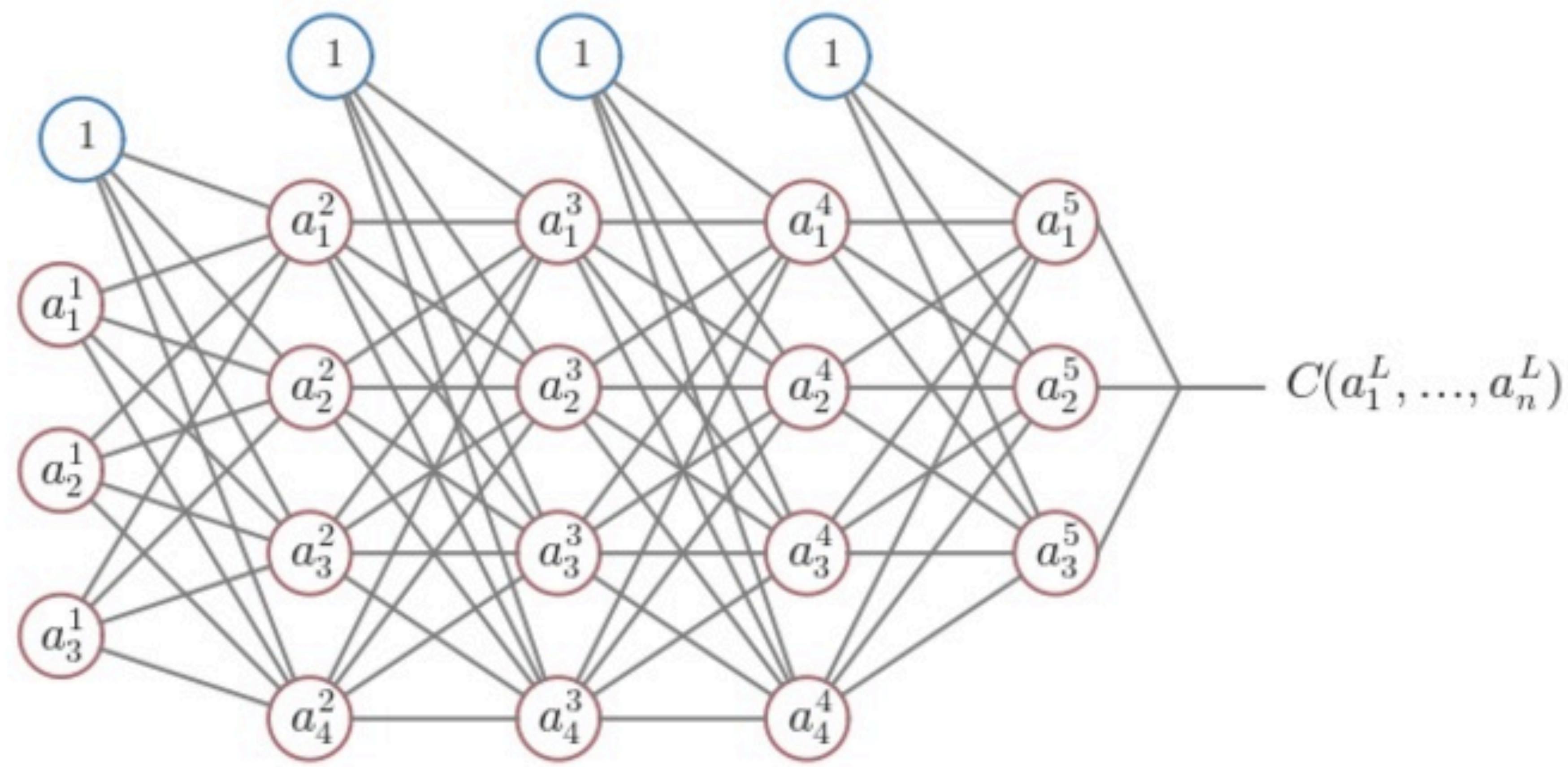


Up to us to choose:

- num. of hidden layers
- activation
- cost function

Feed-Forward Neural Network

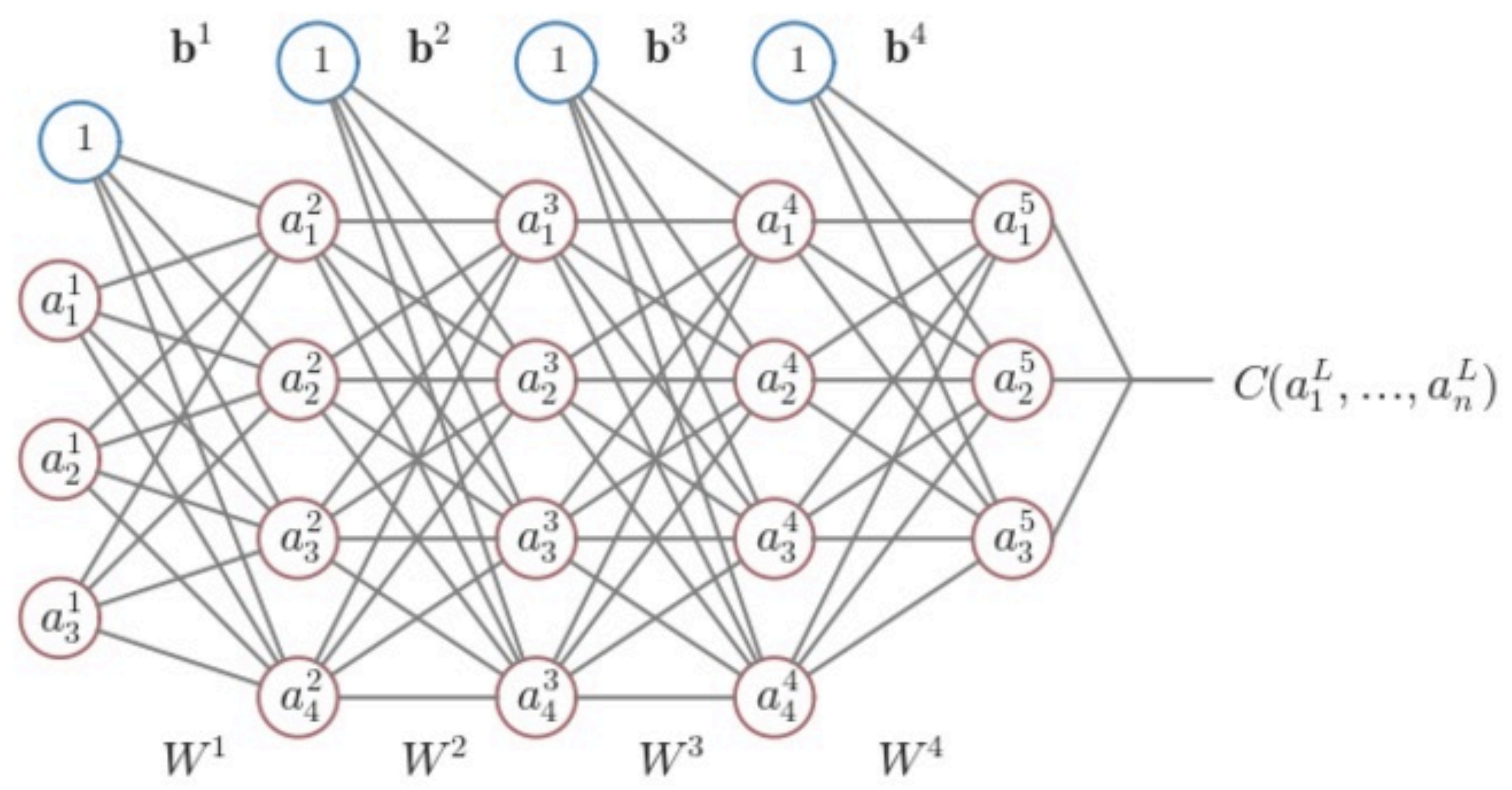
Last time we saw the generic feed-forward neural network



For simplicity, think of inputs and outputs as same as activations

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

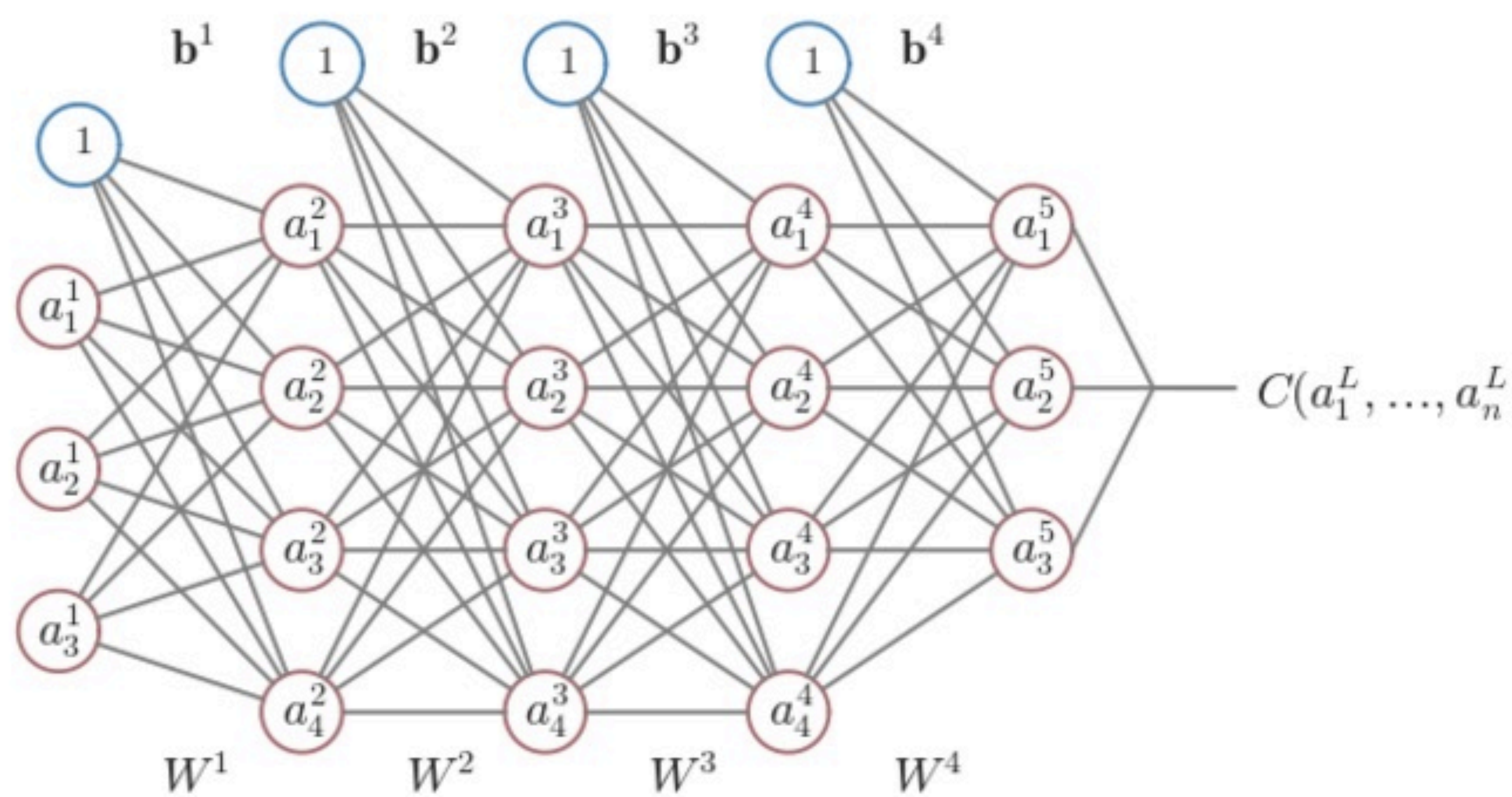


Inputs, outputs, intermediate variables, activations live on nodes

Weights and biases live on edges

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

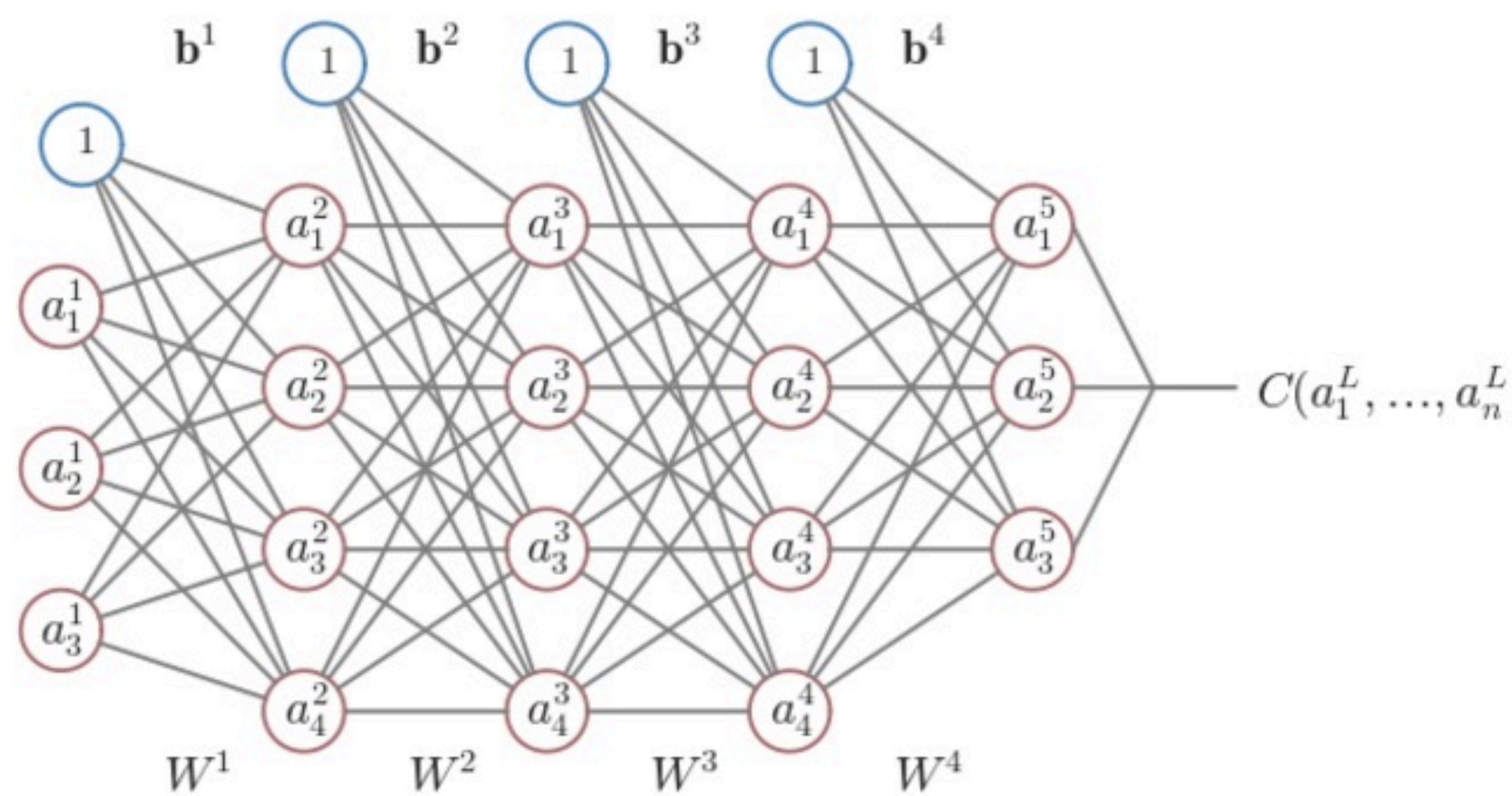


Store the weights connecting layer ℓ and $\ell + 1$ in W^ℓ .

w_{ij}^ℓ is weight from node j in layer ℓ to node i in layer $\ell + 1$

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

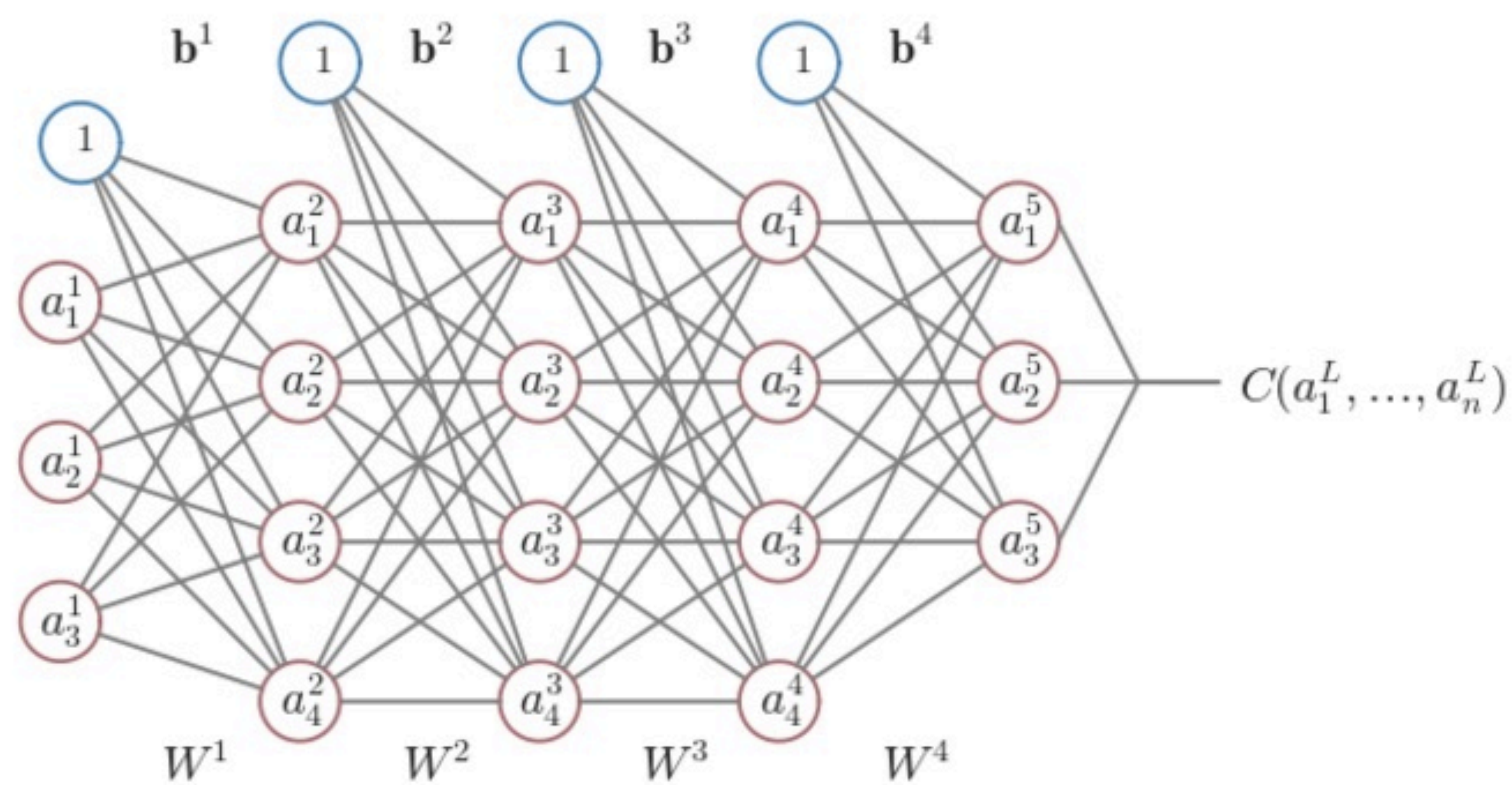


Store the biases for layer $\ell + 1$ in $\mathbf{b}^\ell$.

b_i^ℓ is bias on node i in layer $\ell + 1$

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

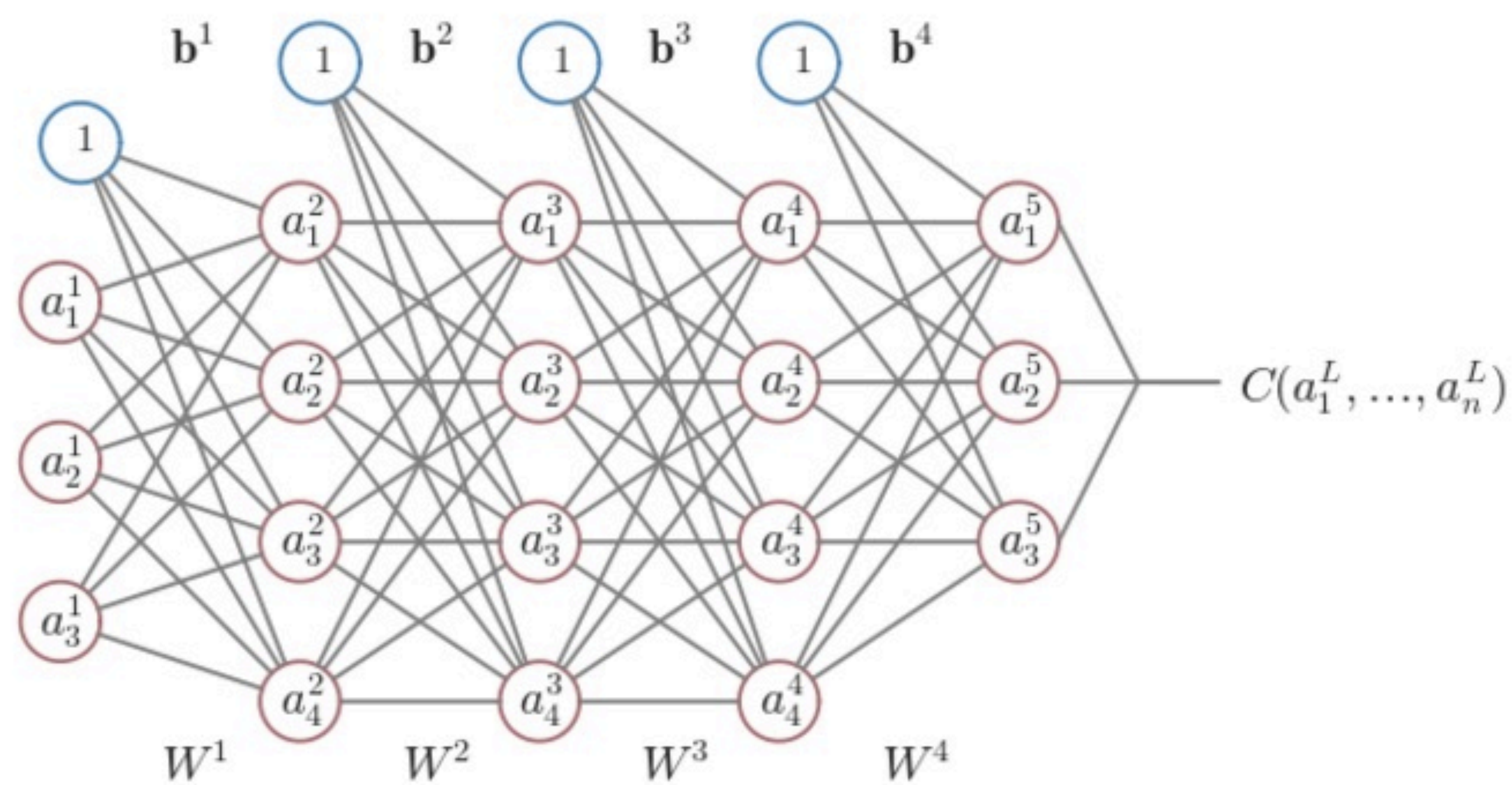


Activation of nodes in layer $\ell + 1$ given by

$$\mathbf{z}^{\ell+1} = W^{\ell}\mathbf{a}^{\ell} + \mathbf{b}^{\ell} \quad \text{then} \quad \mathbf{a}^{\ell+1} = g(\mathbf{z}^{\ell+1})$$

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

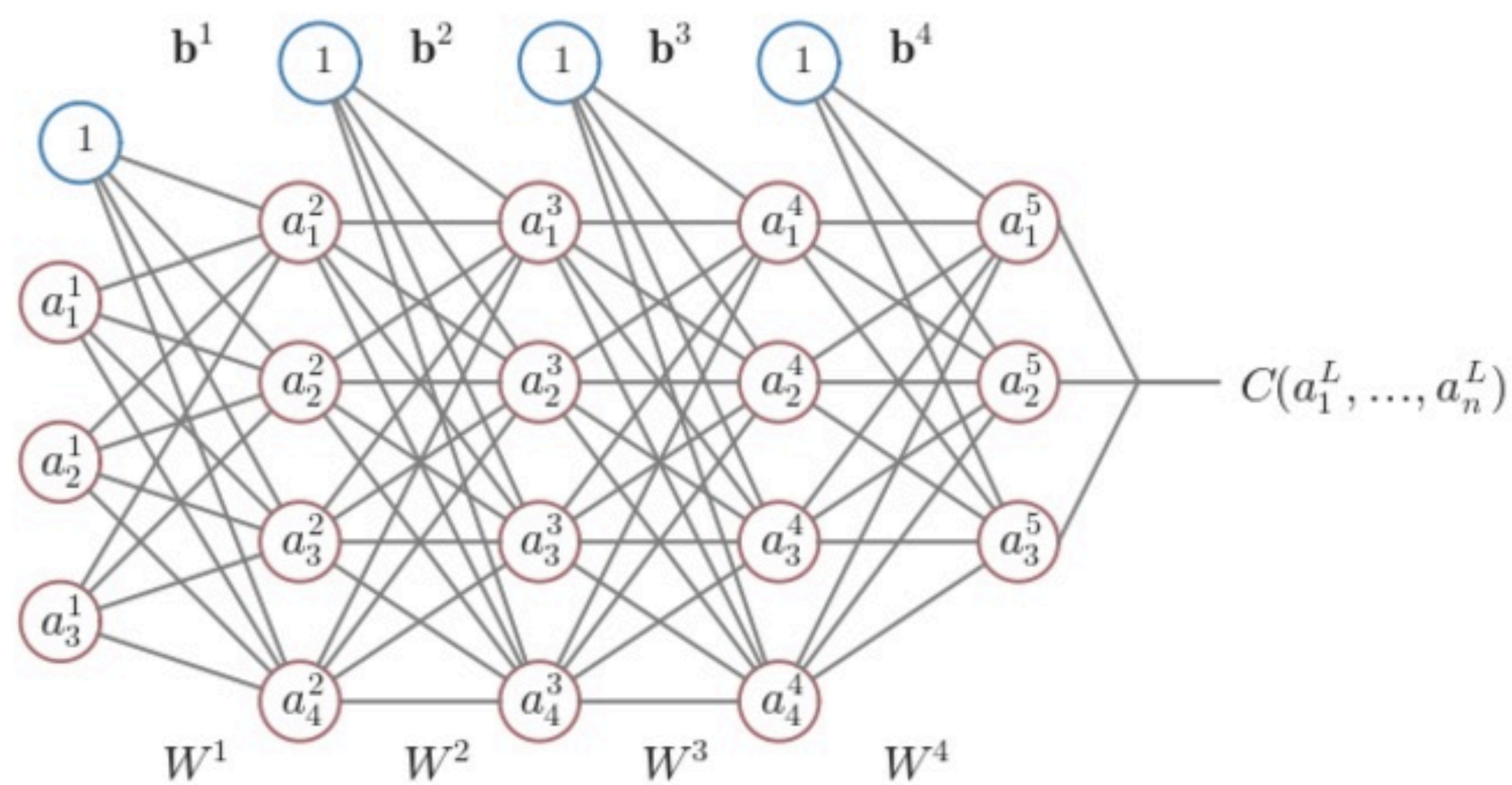


Given guess for the weights and biases, can evaluate the cost function by sequences of matrix multiplications, vector additions, and function evaluations

Want to perform SGD to optimize weights

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network

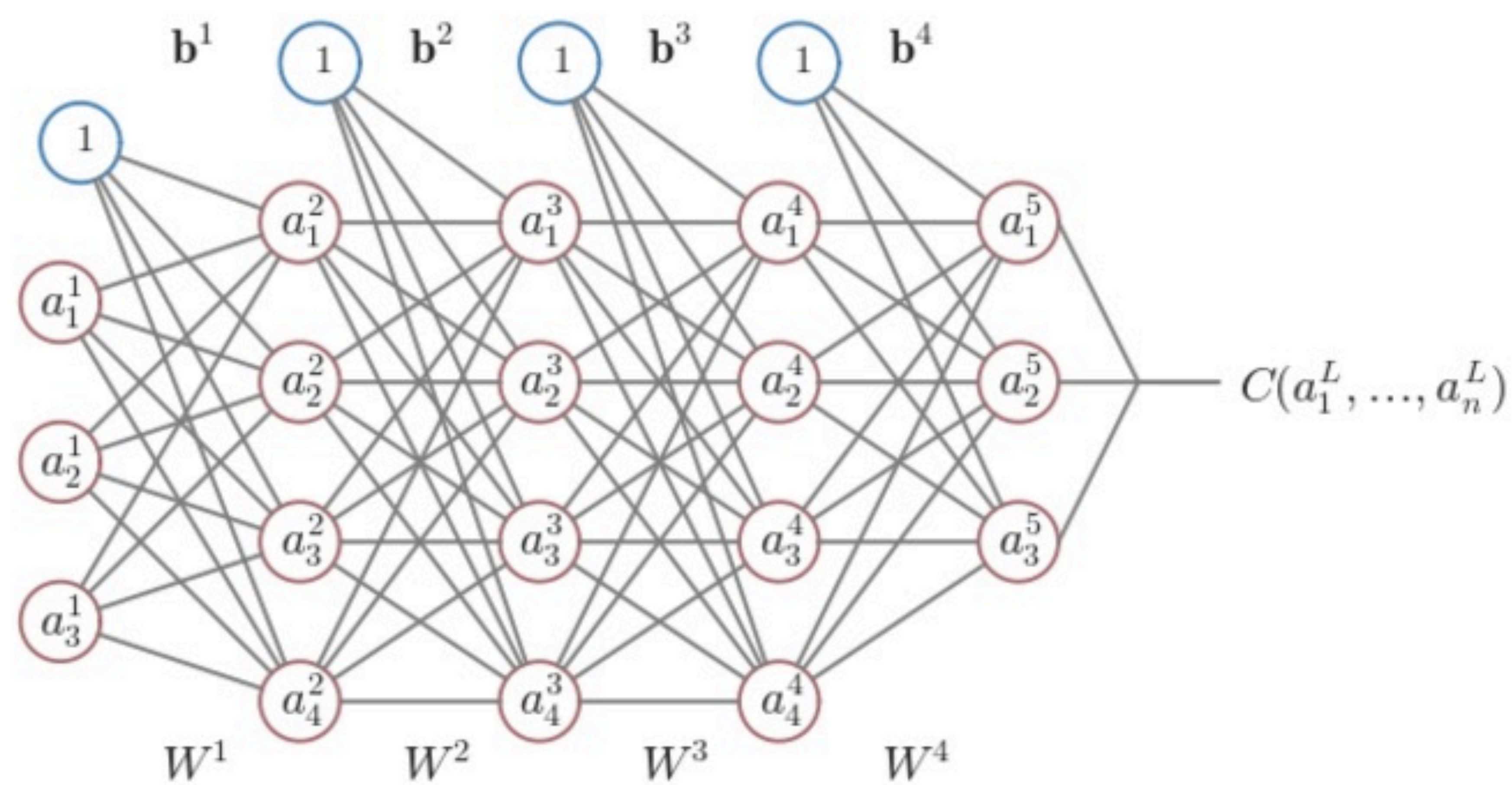


Standard SGD update

$$w_{ij}^{\ell} \leftarrow w_{ij}^{\ell} - \eta \frac{\partial C}{\partial w_{ij}^{\ell}} \qquad b_i^{\ell} \leftarrow b_i^{\ell} - \eta \frac{\partial C}{\partial b_i^{\ell}}$$

Feed-Forward Neural Network

Last time we saw the generic feed-forward neural network



Challenge: How the heck do we compute derivatives of C with respect to weights and biases?

Solution: Back Propagation

The Chain Rule

The chain rule allows us to take derivatives of nested functions

There are two forms of the Chain Rule

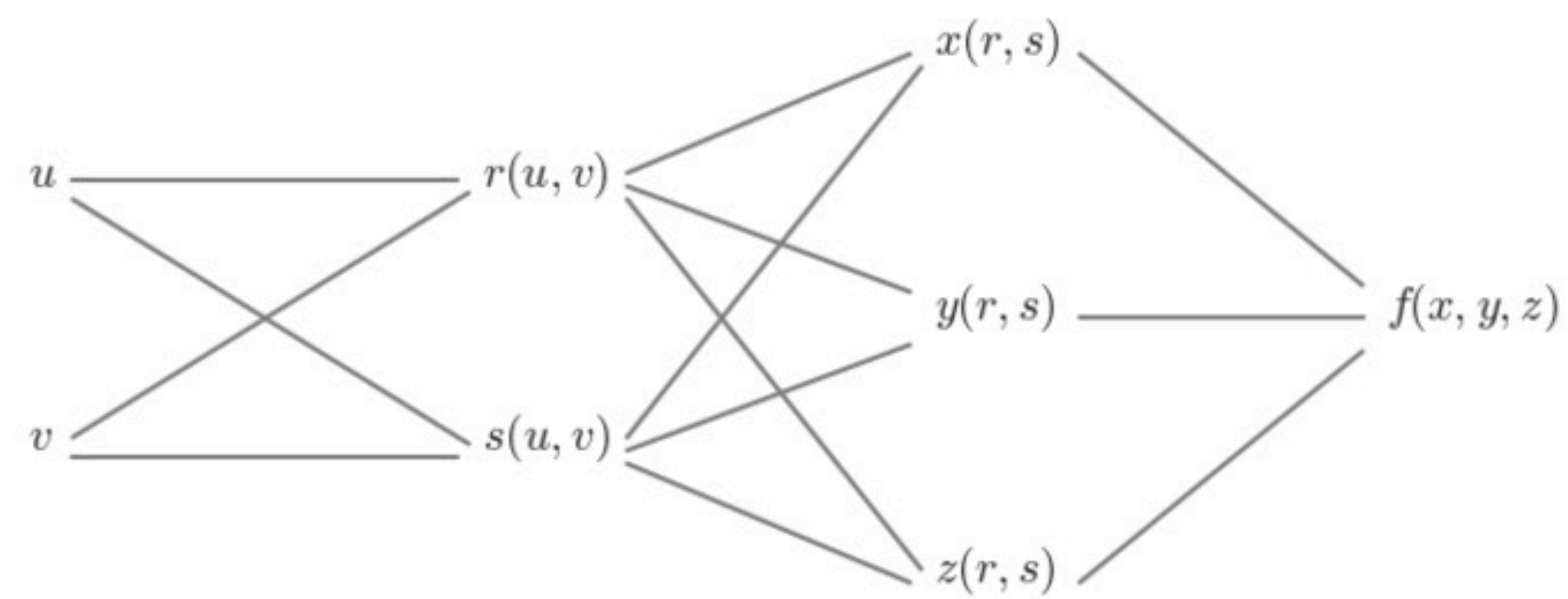
Baby Chain Rule:

$$\frac{d}{dx} f(g(x)) = f'(g(x))\, g'(x) = \frac{df}{dg} \frac{dg}{dx}$$

Example: $$\frac{d}{dx} \sin(x^2) = \cos(x^2)\, 2x$$

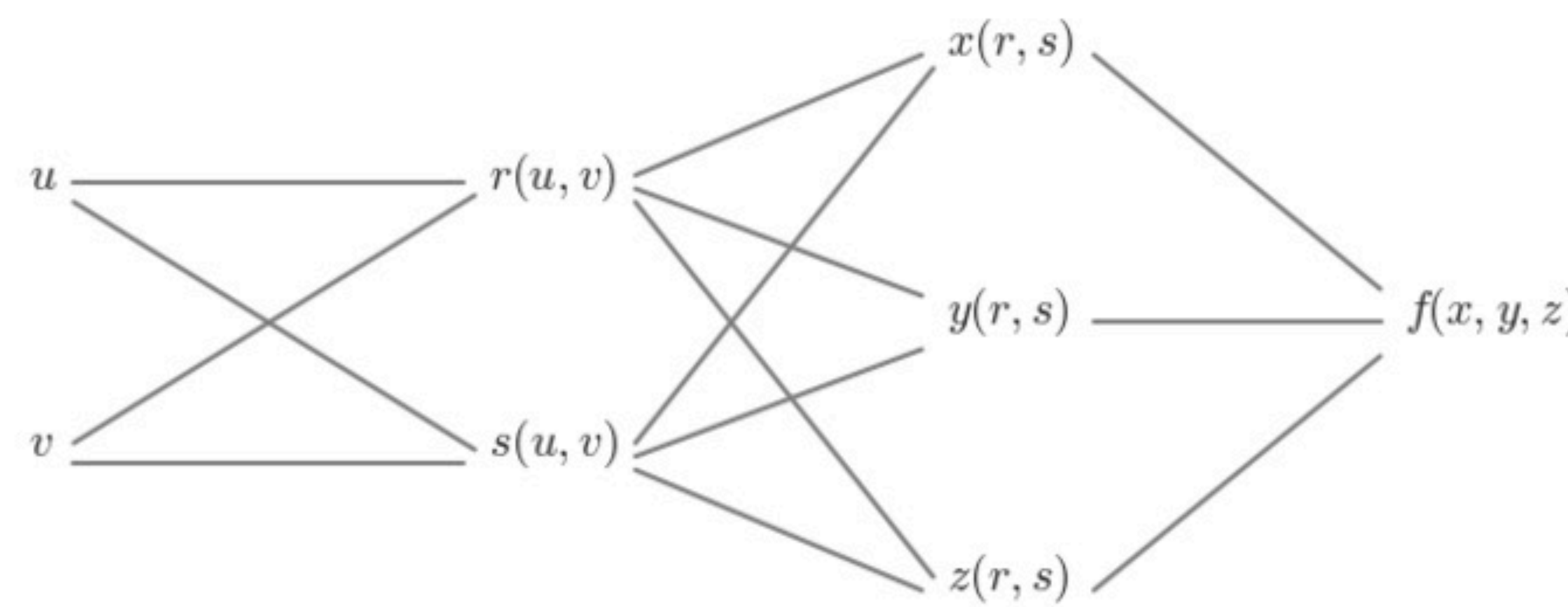
The Chain Rule

Full-Grown Adult Chain Rule:



The Chain Rule

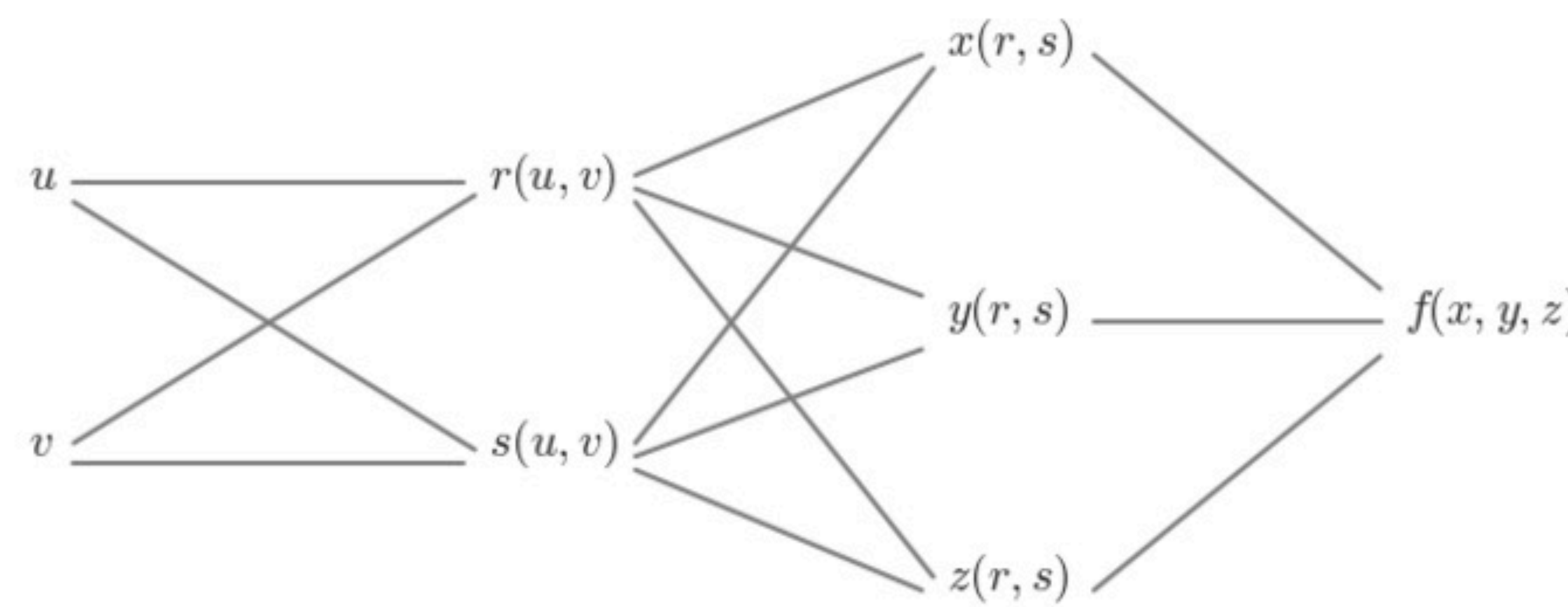
Full-Grown Adult Chain Rule:



Derivative of f with respect to x: $\dfrac{\partial f}{\partial x}$

The Chain Rule

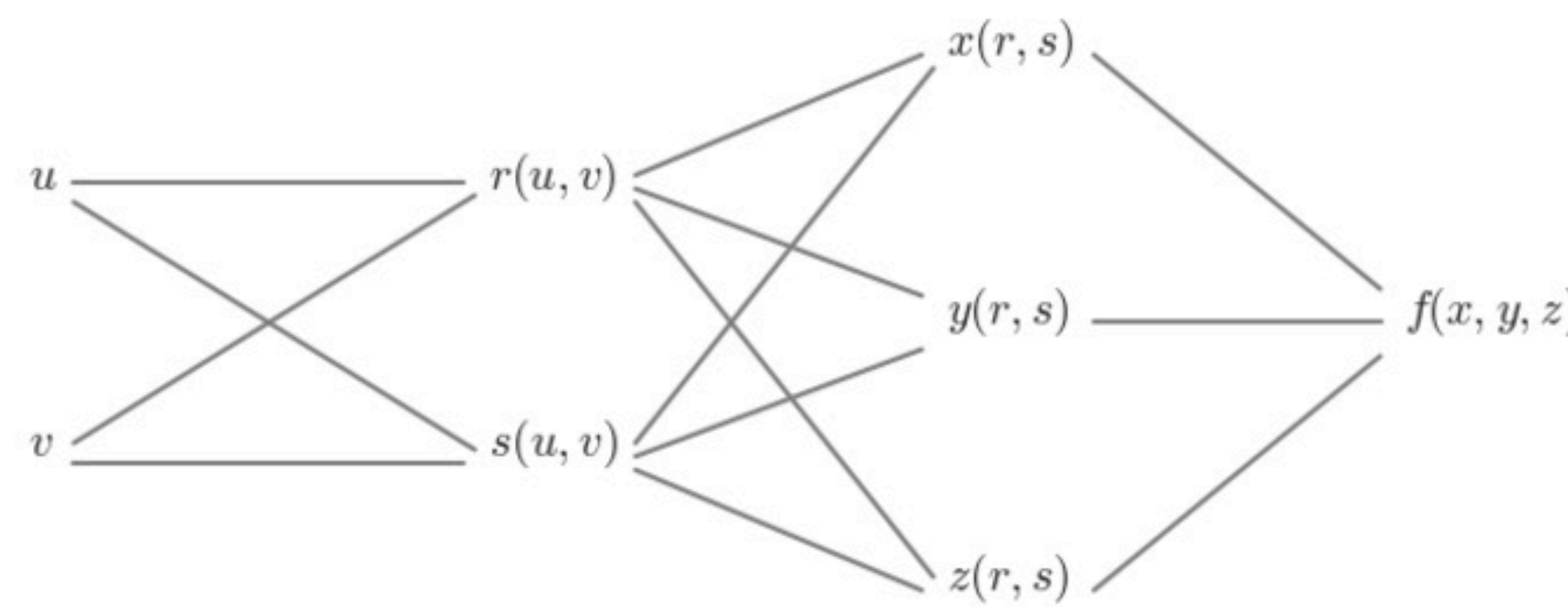
Full-Grown Adult Chain Rule:



Derivative of f with respect to y: $\dfrac{\partial f}{\partial y}$

The Chain Rule

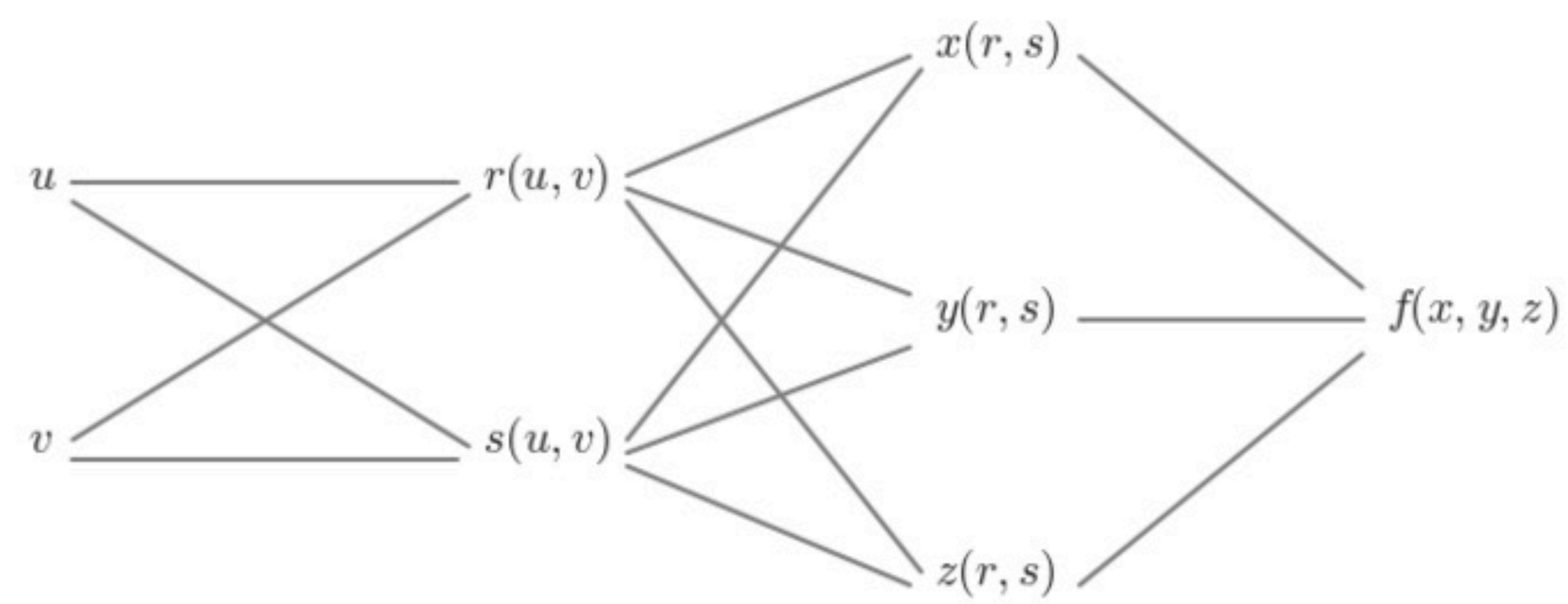
Full-Grown Adult Chain Rule:



Derivative of f with respect to z: $\dfrac{\partial f}{\partial z}$

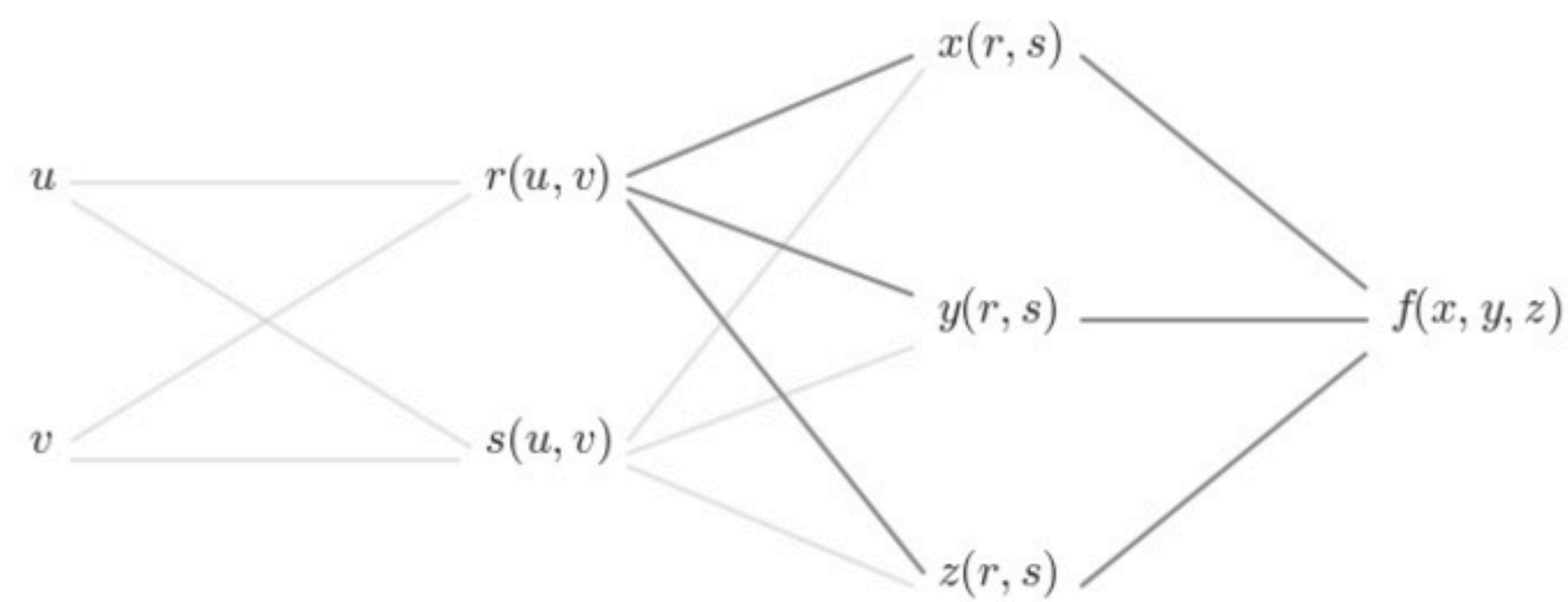
The Chain Rule

What is the derivative of f with respect to r ?



The Chain Rule

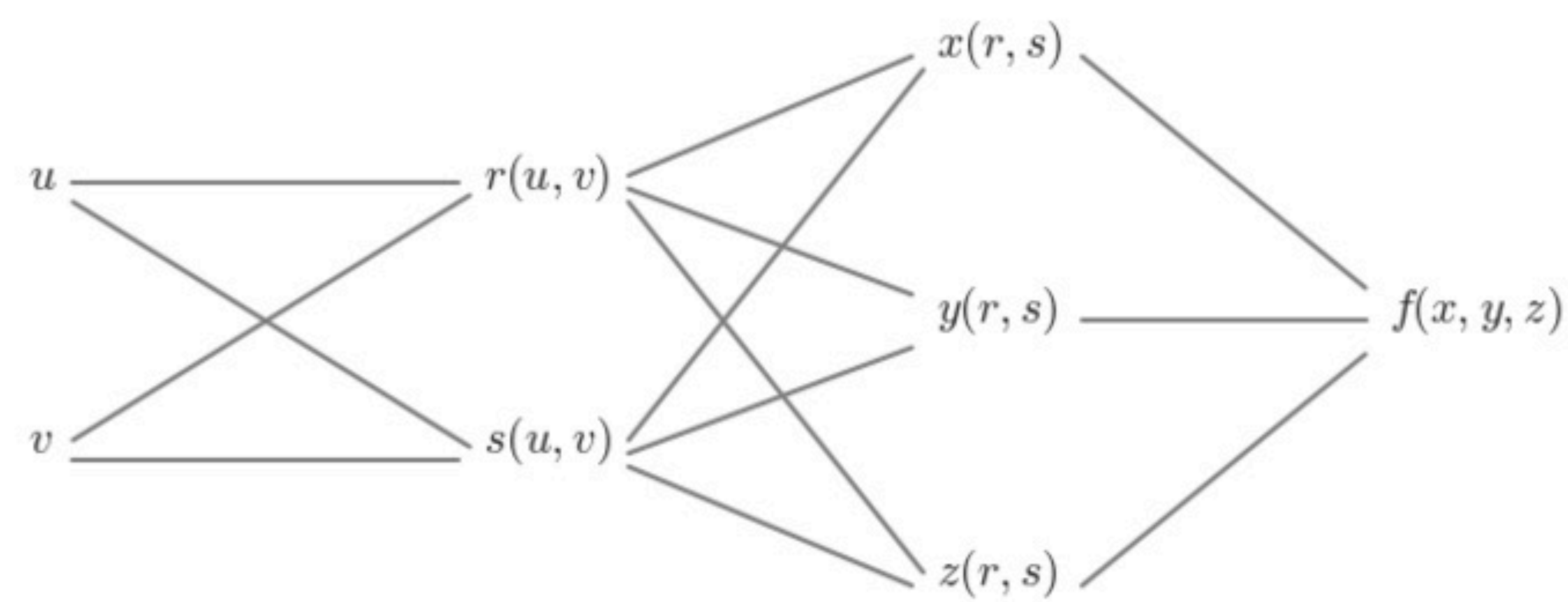
What is the derivative of f with respect to r ?



$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial r}$$

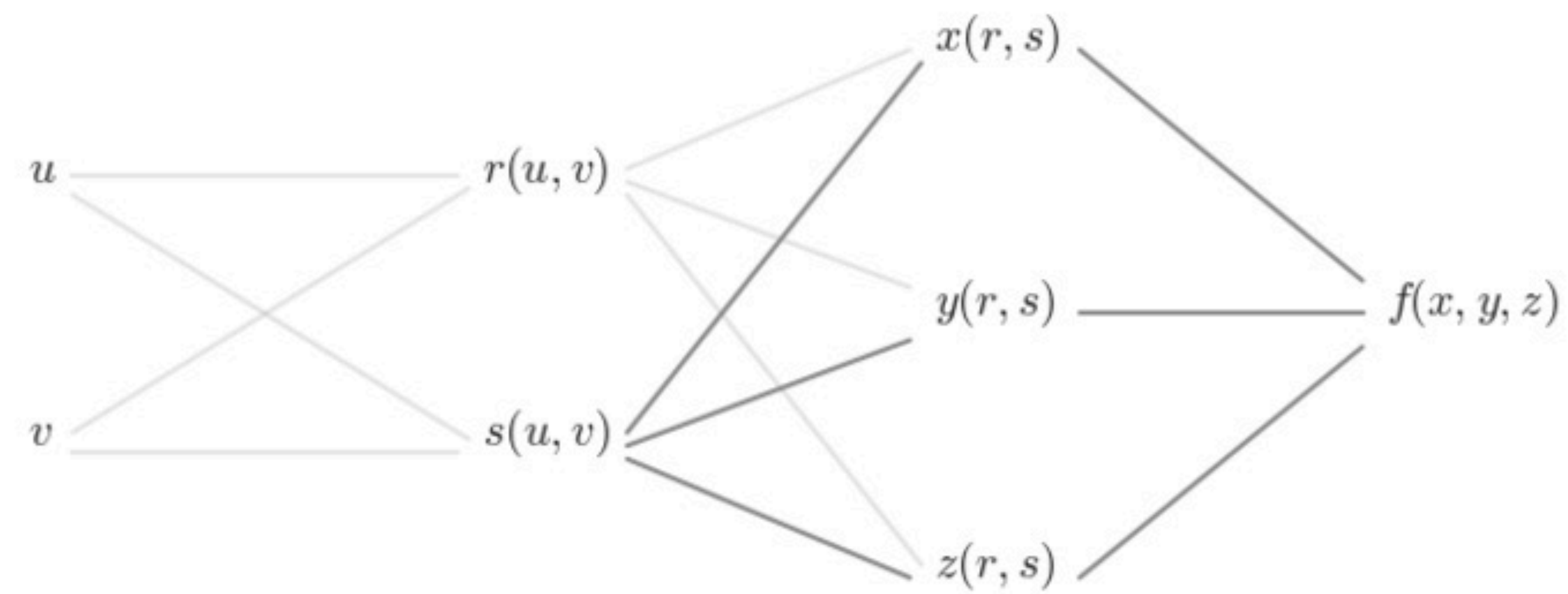
The Chain Rule

What is the derivative of f with respect to s ?



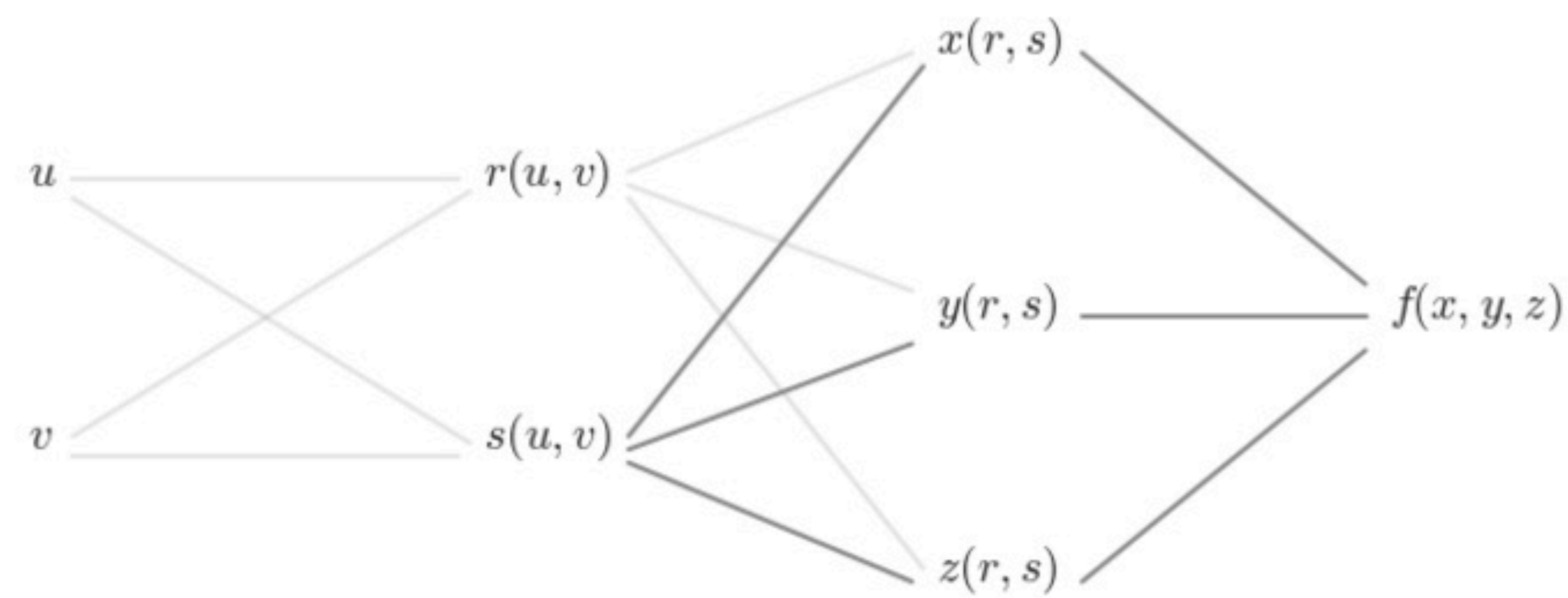
The Chain Rule

What is the derivative of f with respect to s ?



$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}$$

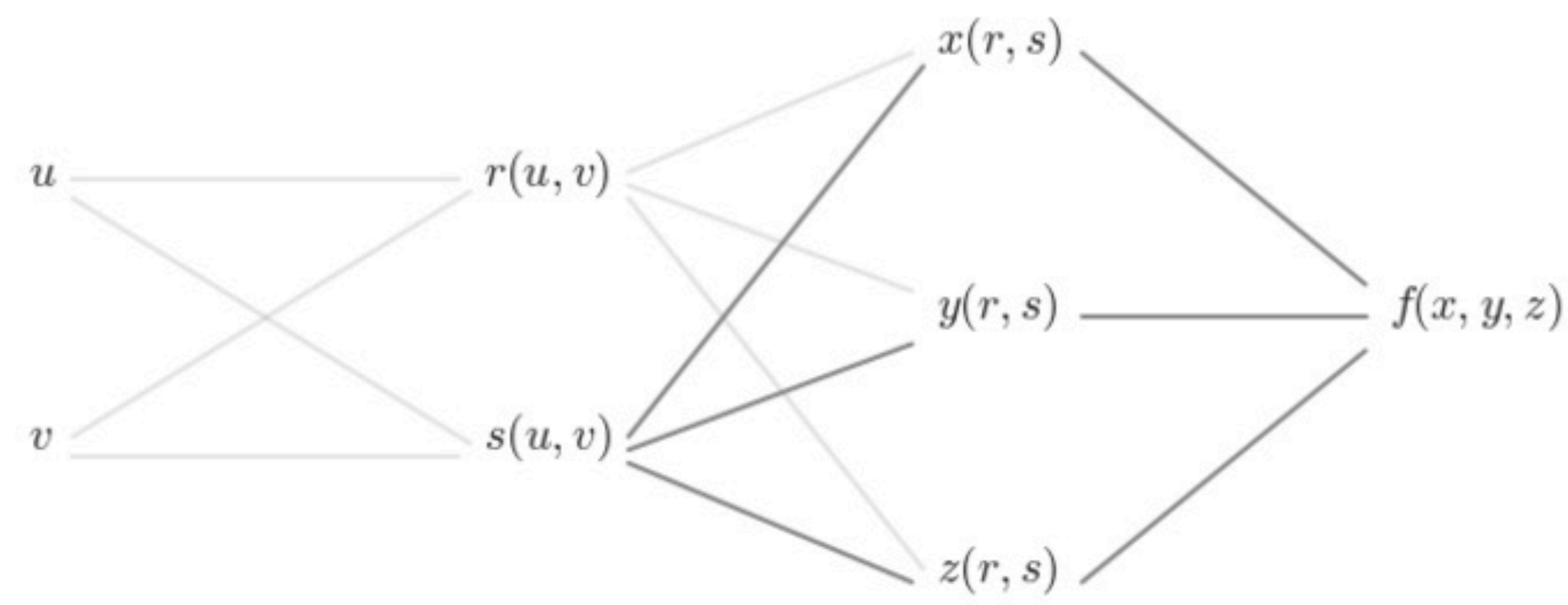
The Chain Rule



Example: Let $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\partial f / \partial s$

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}$$

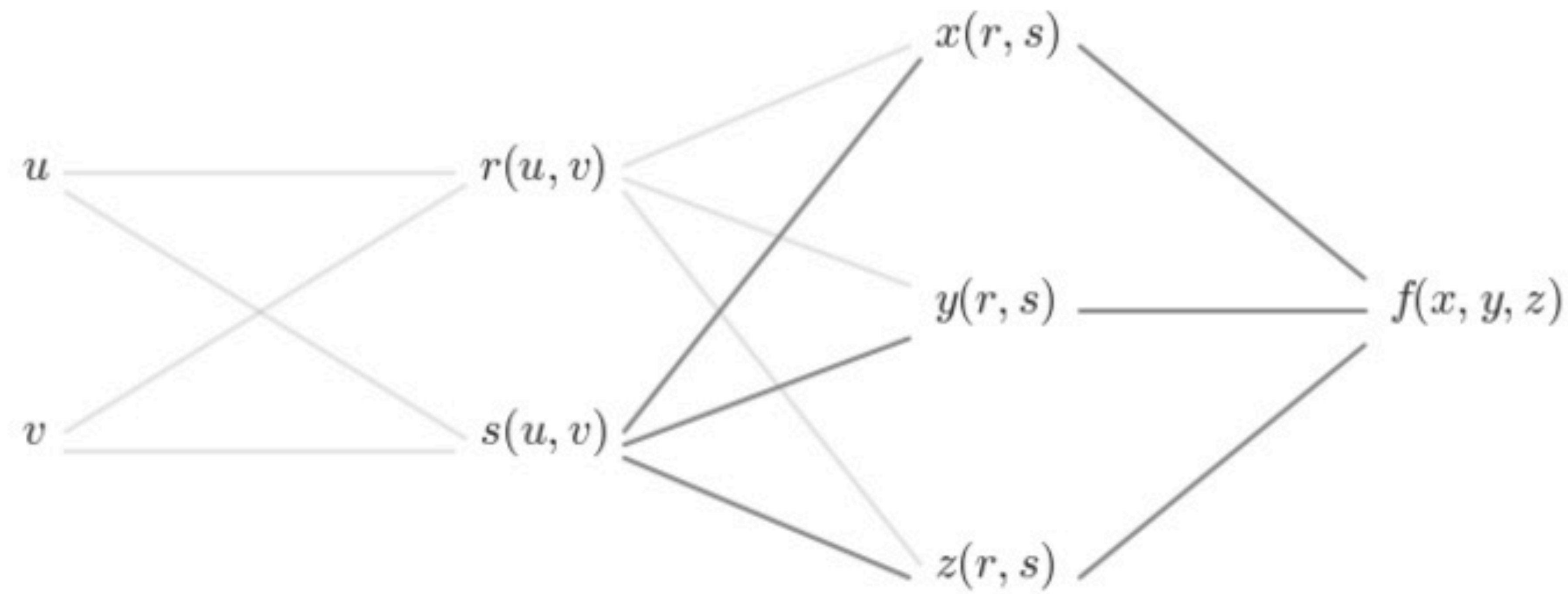
The Chain Rule



Example: Let $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\partial f / \partial s$

$$\frac{\partial f}{\partial s} = yz \cdot 0 + xz \cdot r + xy \cdot 1$$

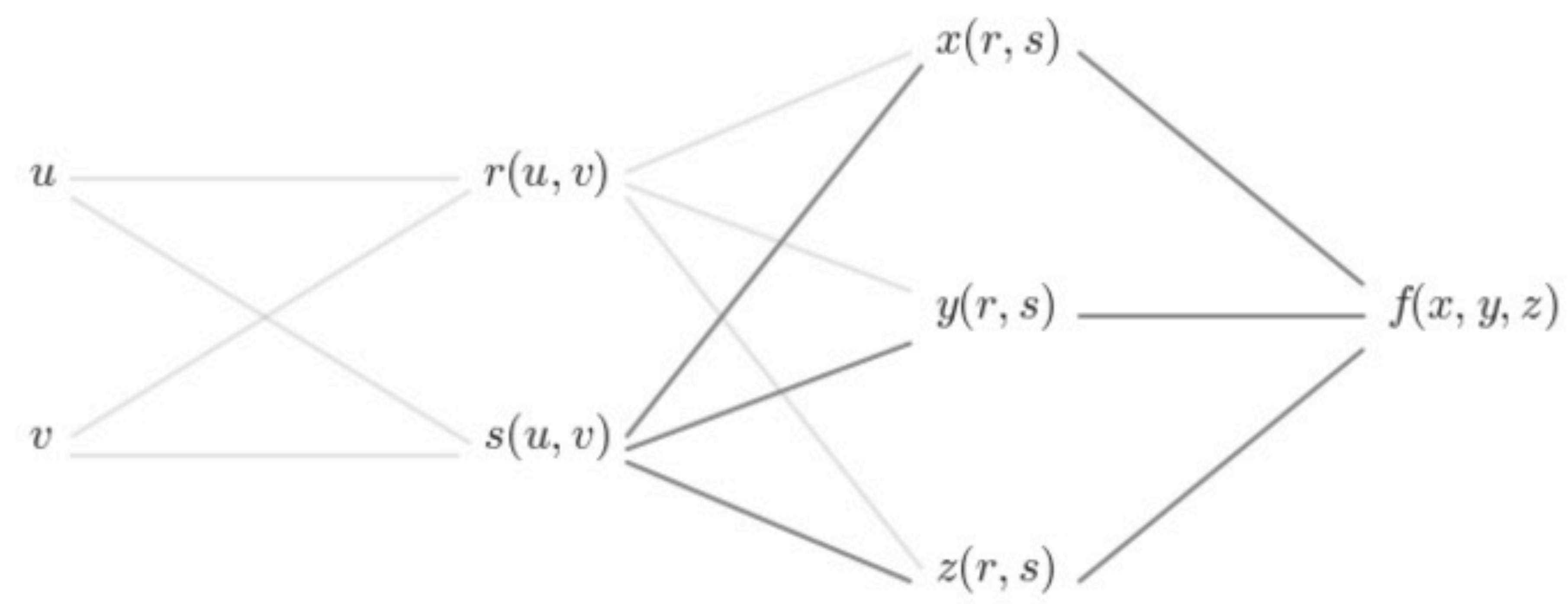
The Chain Rule



Example: Let $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\partial f / \partial s$

$$\frac{\partial f}{\partial s} = rs^2 \cdot 0 + rs \cdot r + r^2 s \cdot 1$$

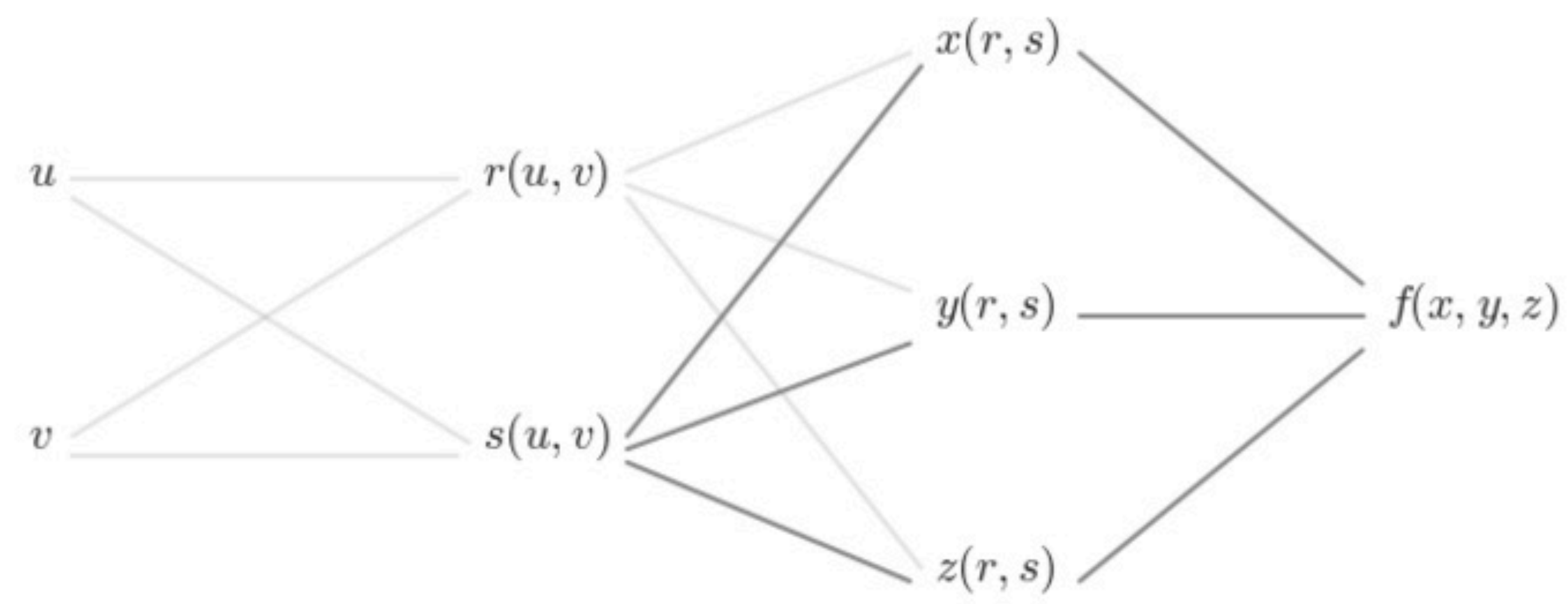
The Chain Rule



Example: Let $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\partial f / \partial s$

$$\frac{\partial f}{\partial s} = 2r^2 s$$

The Chain Rule

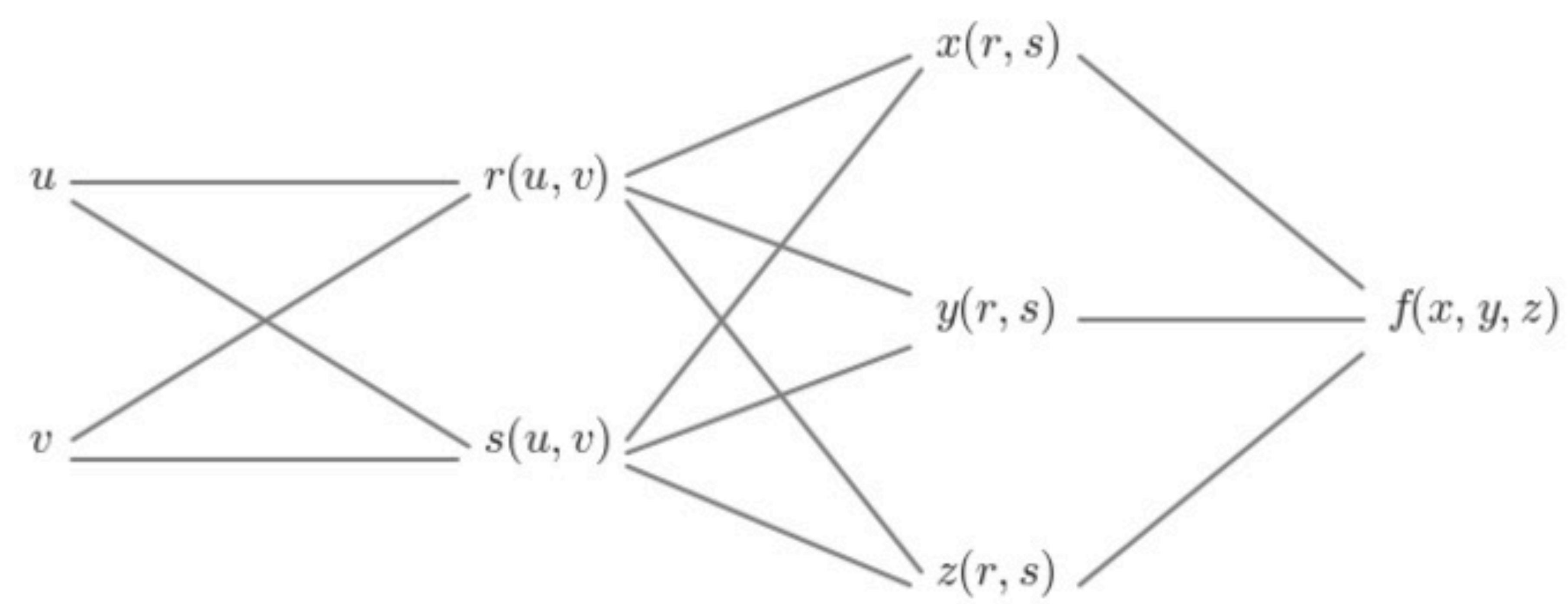


Example: Let $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\partial f / \partial s$

$$f(r,s) = r \cdot rs \cdot s = r^2 s^2 \quad \Rightarrow \quad \frac{\partial f}{\partial s} = 2r^2 s \ \checkmark$$

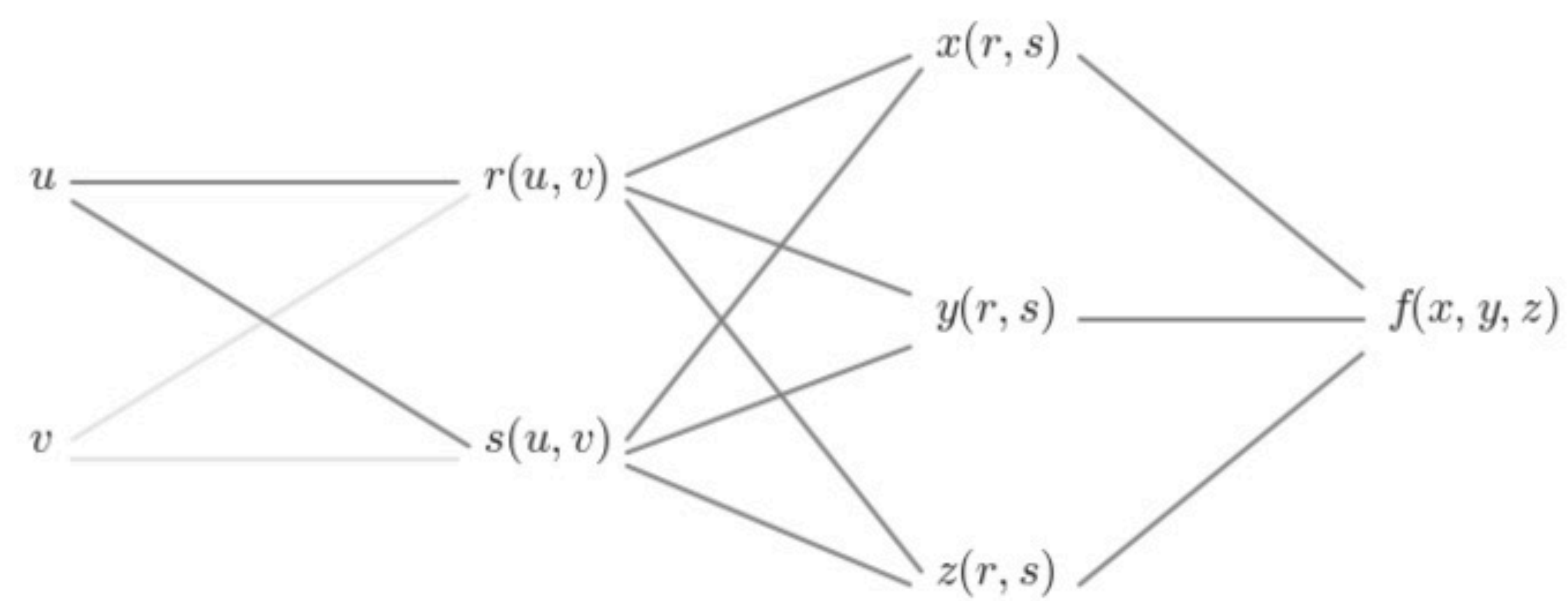
The Chain Rule

What is the derivative of f with respect to u ?



The Chain Rule

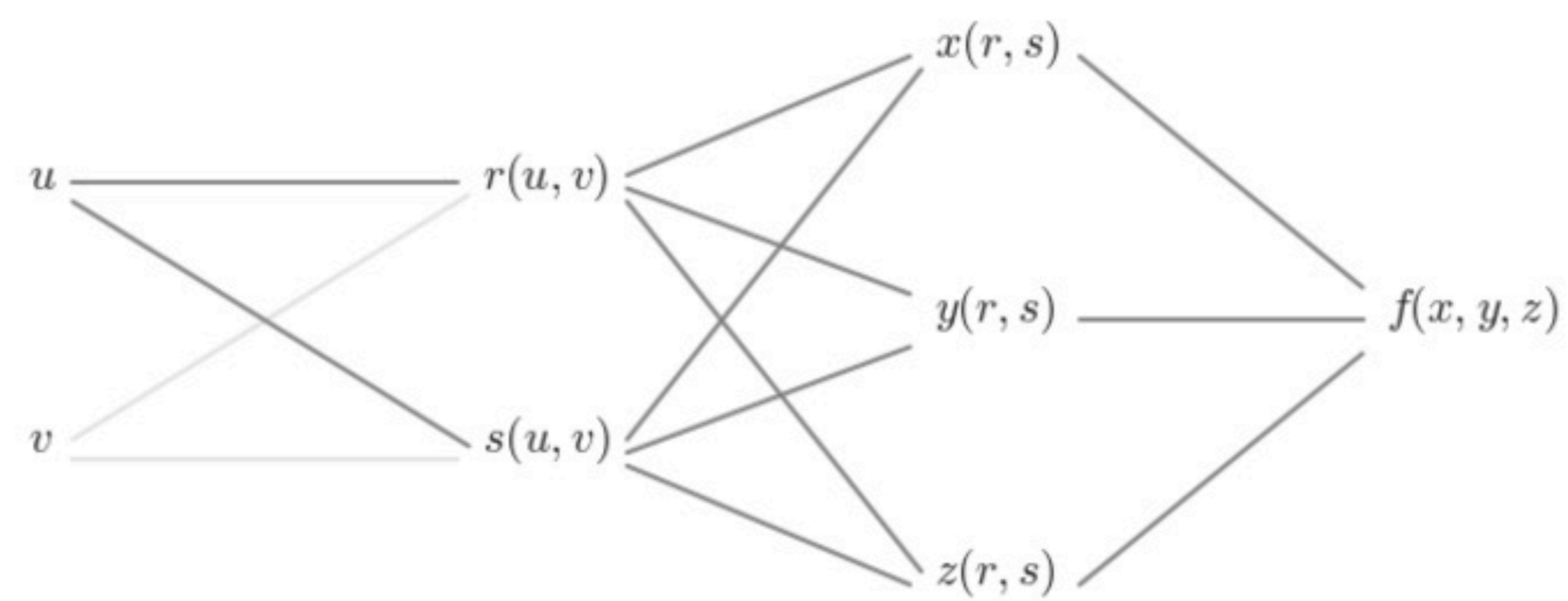
What is the derivative of f with respect to u ?



$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial x}\frac{\partial x}{\partial s}\frac{\partial s}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial u} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial u}$$

The Chain Rule

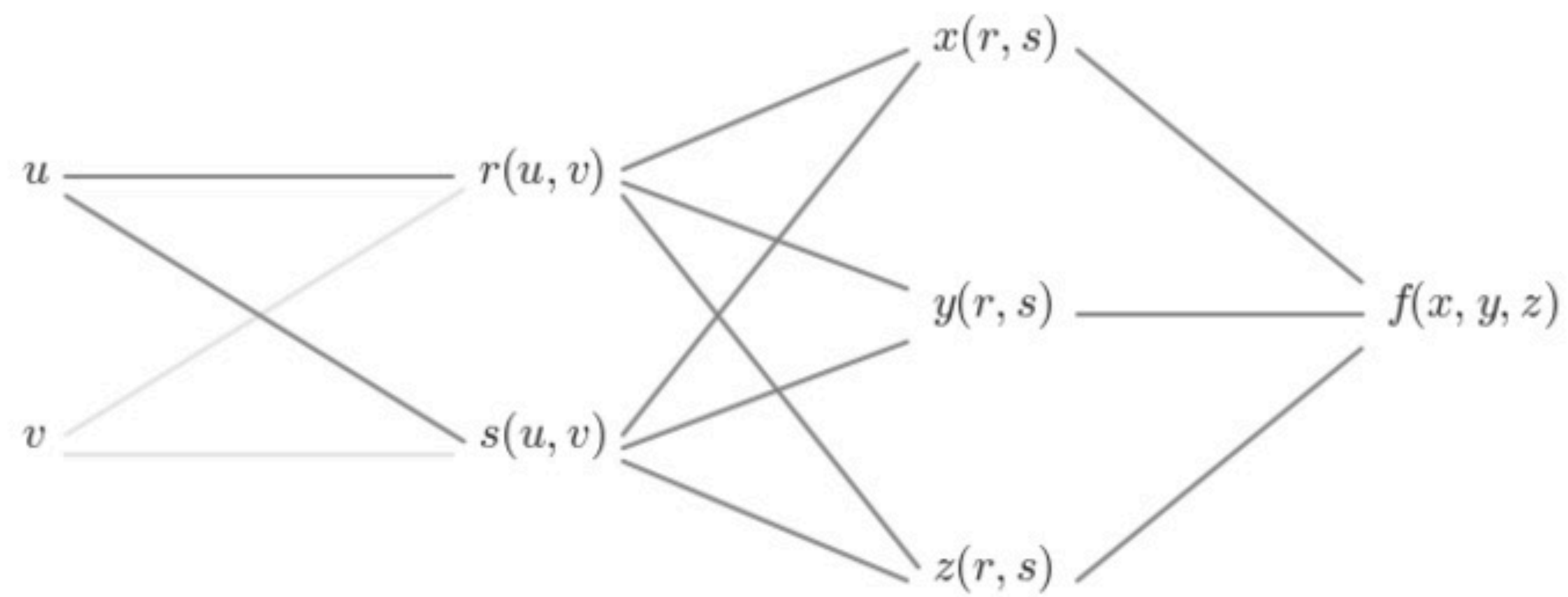
What is the derivative of f with respect to u ?



$$\frac{\partial f}{\partial u} = \left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial r}\right)\frac{\partial r}{\partial u} + \left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}\right)\frac{\partial s}{\partial u}$$

The Chain Rule

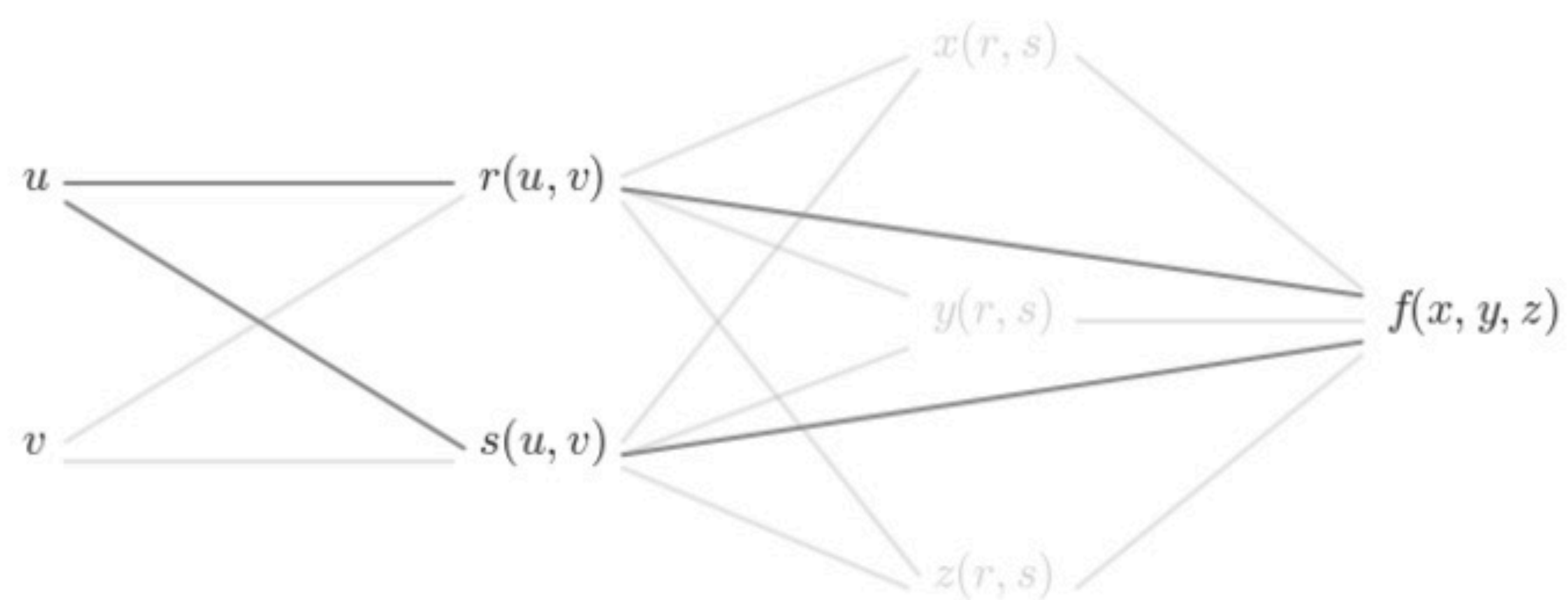
What is the derivative of f with respect to u ?



$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial s}\frac{\partial s}{\partial u}$$

The Chain Rule

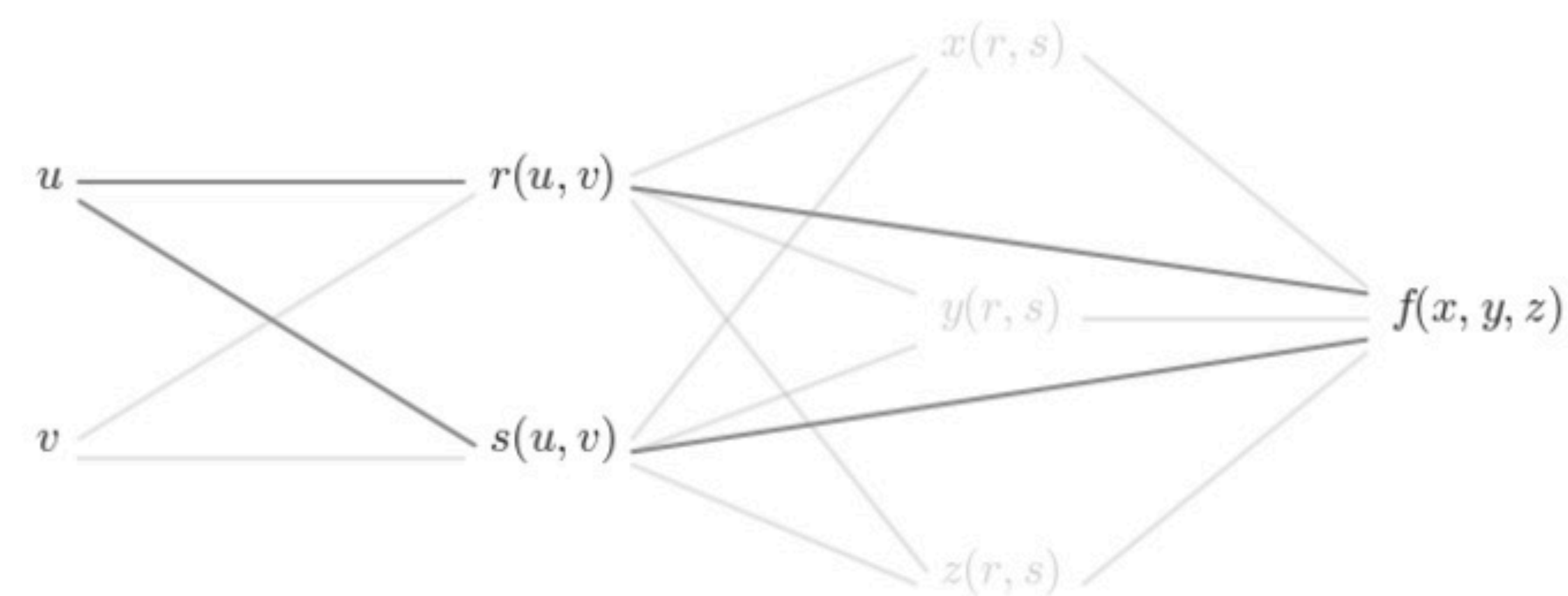
What is the derivative of f with respect to u ?



$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial s}\frac{\partial s}{\partial u}$$

The Chain Rule

What is the derivative of f with respect to u ?

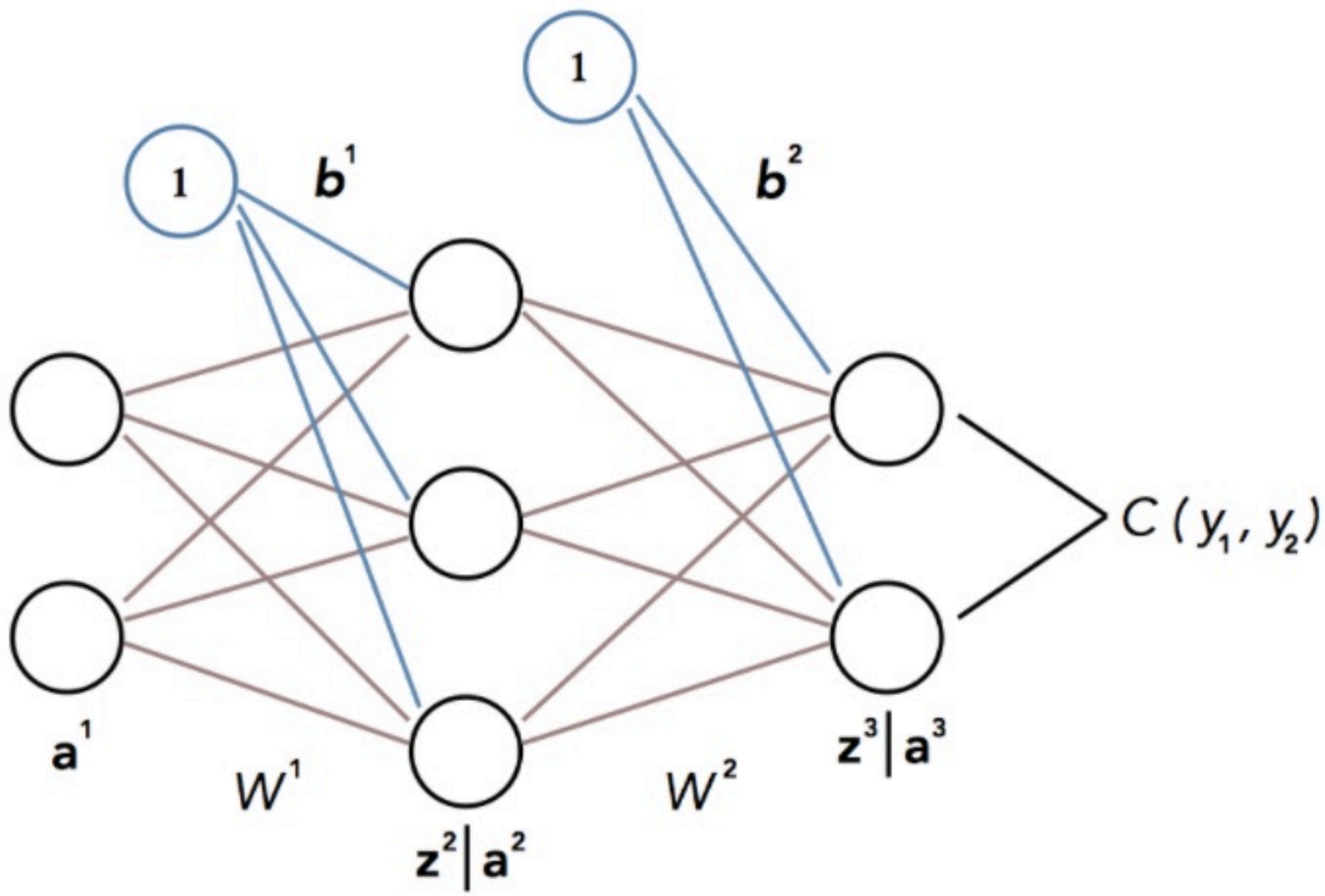


Crux: If you know derivative of objective w.r.t. intermediate value in the chain, can eliminate everything in between

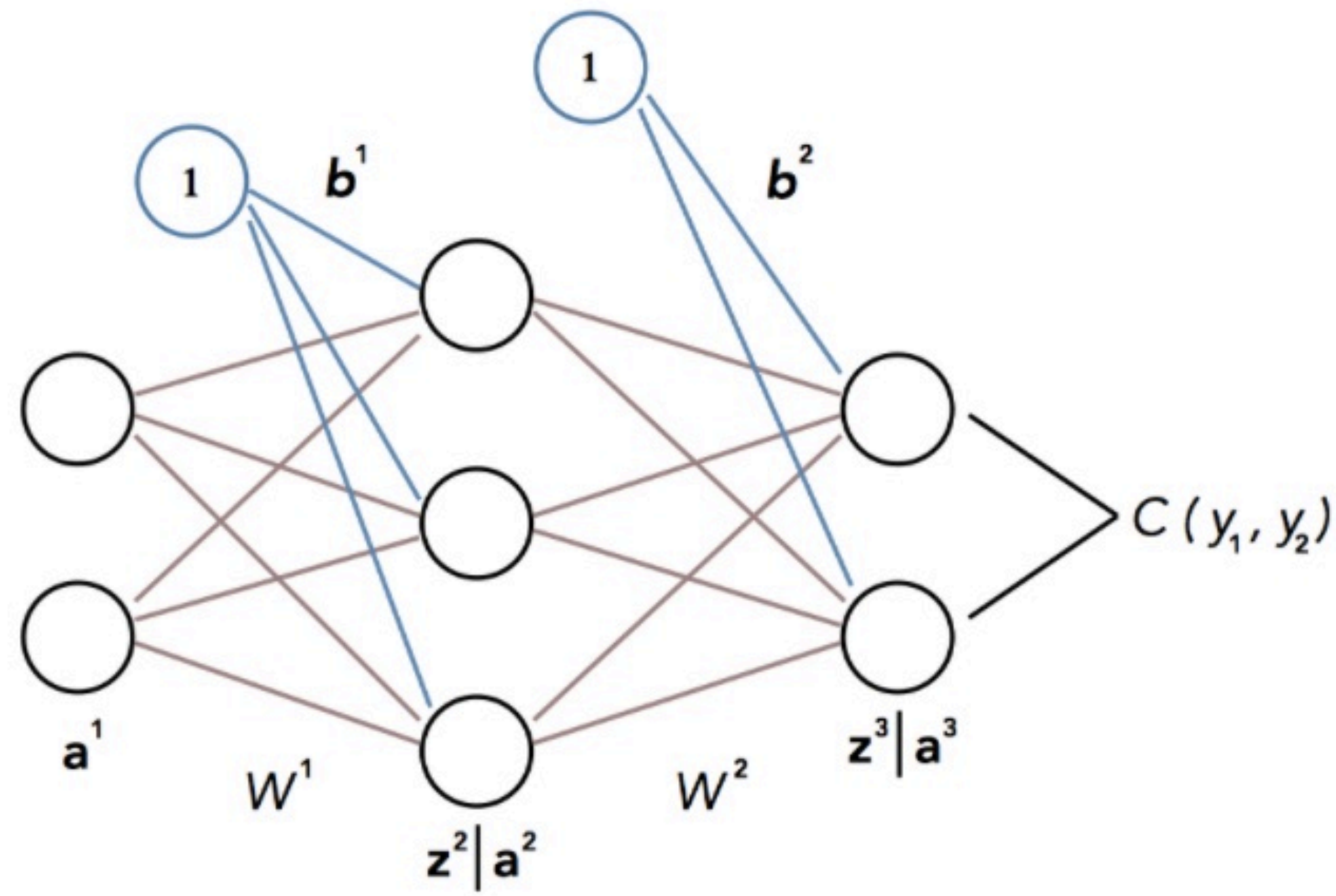
This is the cornerstone of the Back Propagation algorithm

Back Propagation

For the derivation, we'll consider a simplified network



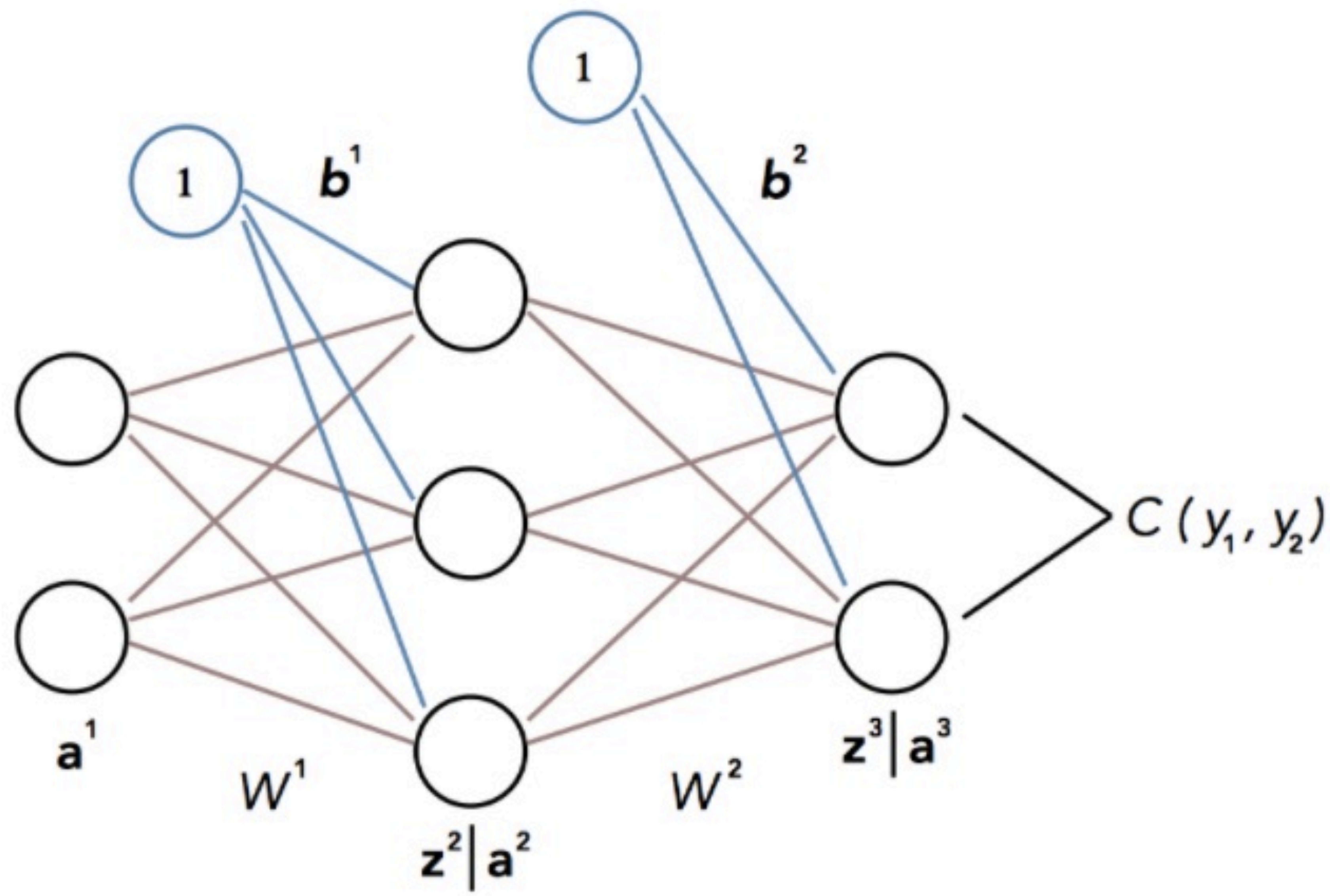
Back Propagation



We want to use the Chain Rule to compute partial derivatives of C w.r.t. the weights and the biases

$$\frac{\partial C}{\partial w_{ij}^{\ell}}, \quad \frac{\partial C}{\partial b_i^{\ell}} \quad \text{for } \ell = 1, 2$$

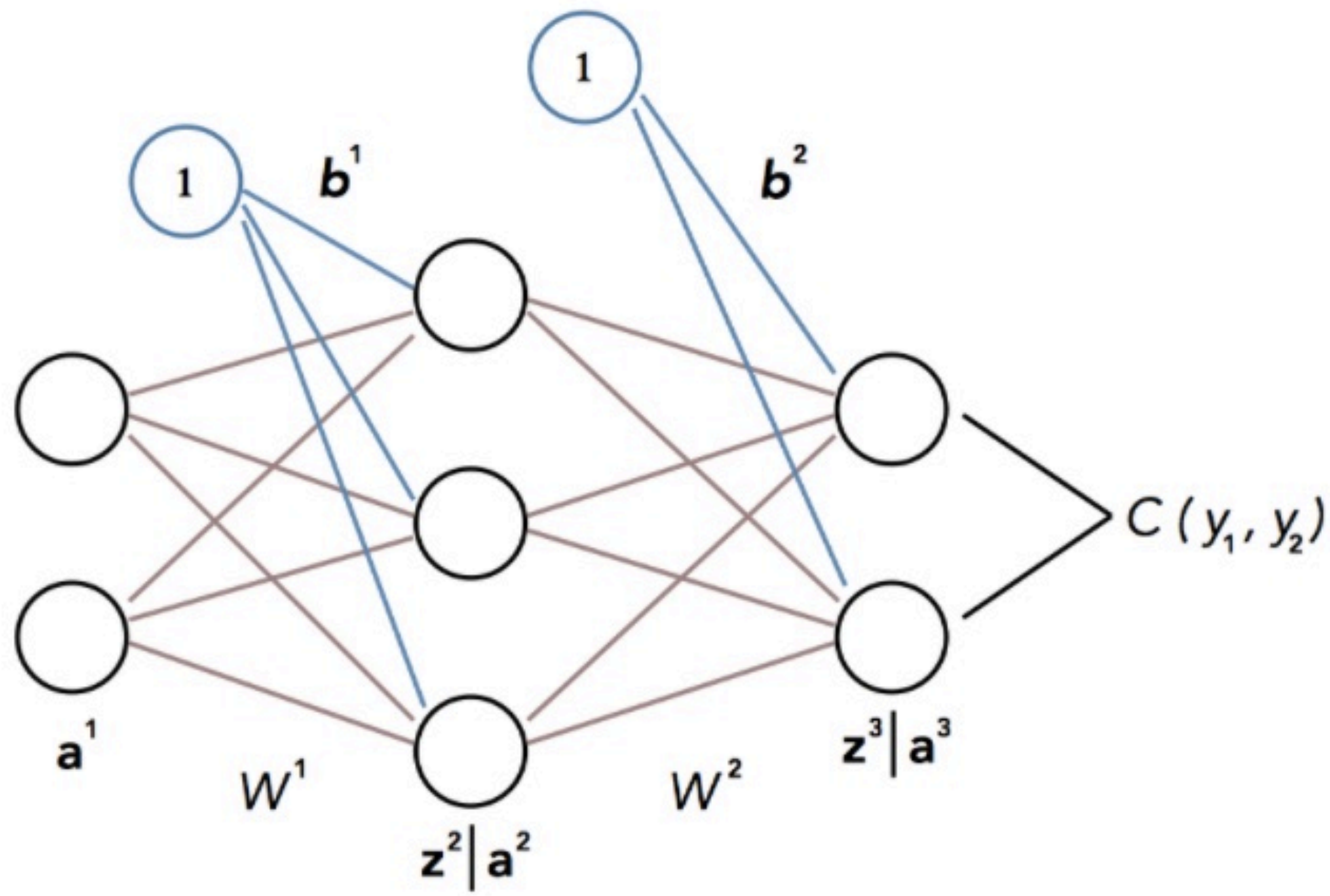
Back Propagation



We need to choose an intermediate term that lives on the nodes, that we can easily compute derivative with respect to

Could choose $\mathbf{a}$'s, but we'll choose $\mathbf{z}$'s because math is easier

Back Propagation

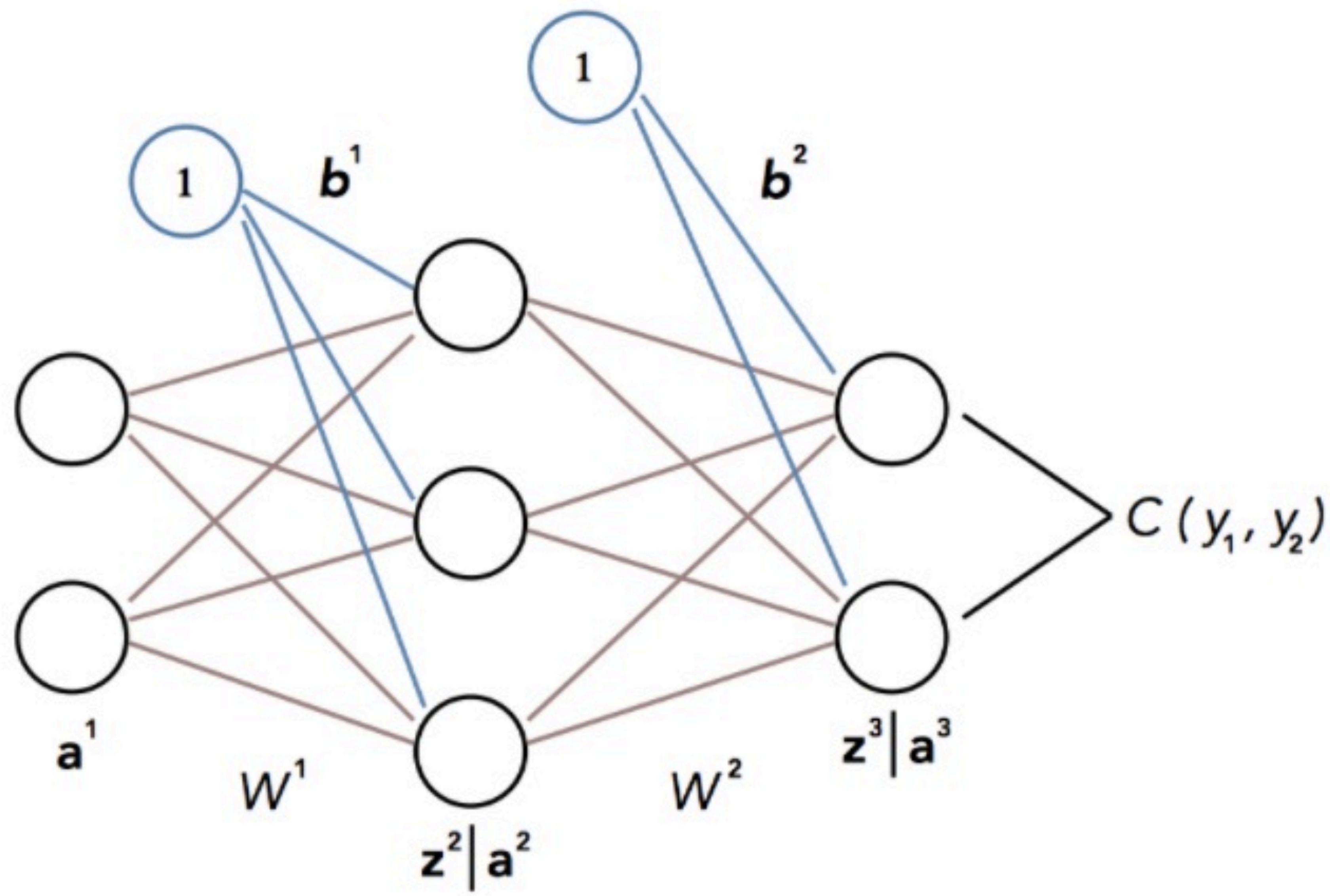


For easier notation, we'll represent partials of C w.r.t. z by δ

$$\delta_i^\ell = \frac{\partial C}{\partial z_i^\ell}, \quad \delta^\ell \text{ denotes the vector quantity}$$

Note: Vector δ^ℓ same size as vectors $\mathbf{z}^\ell$ and $\mathbf{a}^\ell$

Back Propagation



Let's compute δ^3 for output layer $L = 3$:

$$\delta_i^3 = \frac{\partial C}{\partial z_i^3} = \text{(using the chain rule)} = \frac{\partial C}{\partial a_i^3} \frac{da_i^3}{dz_i^3}$$

Back Propagation

So we have $\delta_i^3 = \frac{\partial C}{\partial z_i^3} = \frac{\partial C}{\partial a_i^3} \frac{da_i^3}{dz_i^3}$ where $i = 1, 2$ for this architecture

Need nicer expressions for two terms on right

We know that $a_i^3 = g(z_i^3)$, so $\dfrac{da_i^3}{dz_i^3} = g'(z_i^3)$

Updating our expression for δ_i^3 we have $\delta_i^3 = \frac{\partial C}{\partial a_i^3} g'(z_i^3)$

Note that the first partial derivative is simply the i^{th} entry in the gradient vector of C with respect to the outputs $\mathbf{a}^3$

If we write this gradient vector as $\nabla_{\mathbf{a}^3} C$ then we can come up with a nice vectorized expression

Back Propagation

The elementwise deltas are $\delta_i^3 = \dfrac{\partial C}{\partial a_i^3} g'(z_i^3)$

which we can combine into a vector formula

$$\delta^3 = \nabla_{\mathbf{a}^3} C \odot g'(\mathbf{z}^3)$$

where $g'(\mathbf{z}^3)$ is the derivative of the activation function applied elementwise to $\mathbf{z}^3$

The symbol $\odot$ indicates element-wise multiplication of vectors

This expression δ^ℓ is valid for any differentiable cost function C and activation function g.

In practice we compute these derivatives symbolically.

Back Propagation

The elementwise deltas are $\delta_i^3 = \dfrac{\partial C}{\partial a_i^3} g'(z_i^3)$

which we can combine into a vector formula

$$\delta^3 = \nabla_{\mathbf{a}^3} C \odot g'(\mathbf{z}^3)$$

Important: Notice that computing δ requires knowing activities

This means that before we can compute derivatives for SGD through Back Prop, we first have to Forward Prop the training example through the network!

Back Propagation

Example: Suppose we're in regression setting for the given toy architecture and we choose a sigmoid activation function:

$$C = \frac{1}{2} \sum_j (y_j - a_j^3)^2 \quad \text{and} \quad g(z) = \text{sigm}(z)$$

The derivative of C w.r.t. to an output activation a_i^3 is given by

$$\frac{\partial C}{\partial a_i^3} = 2 \cdot \frac{1}{2} (y_i - a_i^3)(-1) = (a_i^3 - y_i)$$

Since g is a sigmoid we have $g'(z_i^3) = g(z_i^3)(1 - g(z_i^3))$

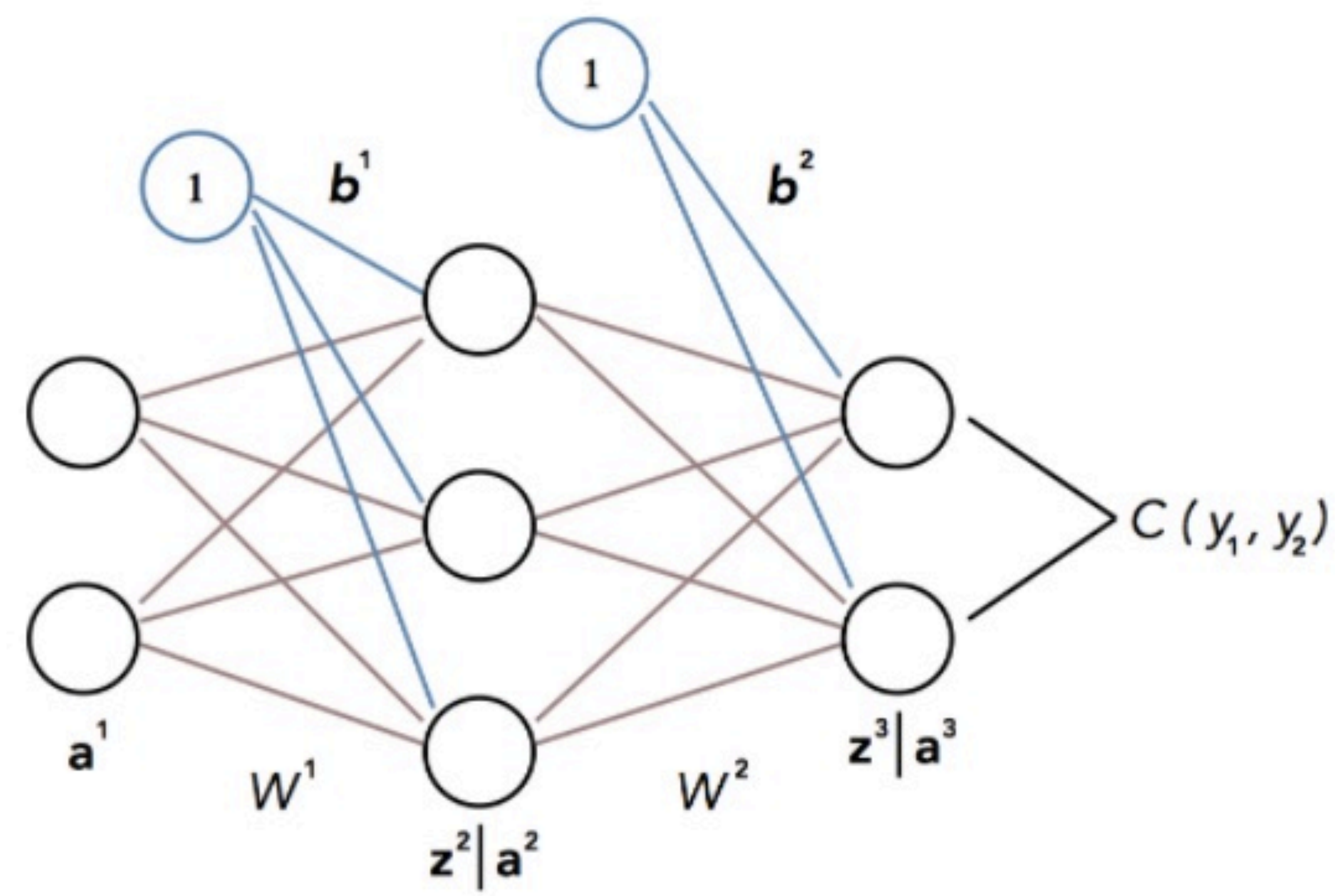
Thus the δ-vector for the output layer can be computed as

$$\delta^3 = (\mathbf{a}^3 - \mathbf{y}) \odot g(\mathbf{z}^3) \odot (1 - g(\mathbf{z}^3))$$

Back Propagation

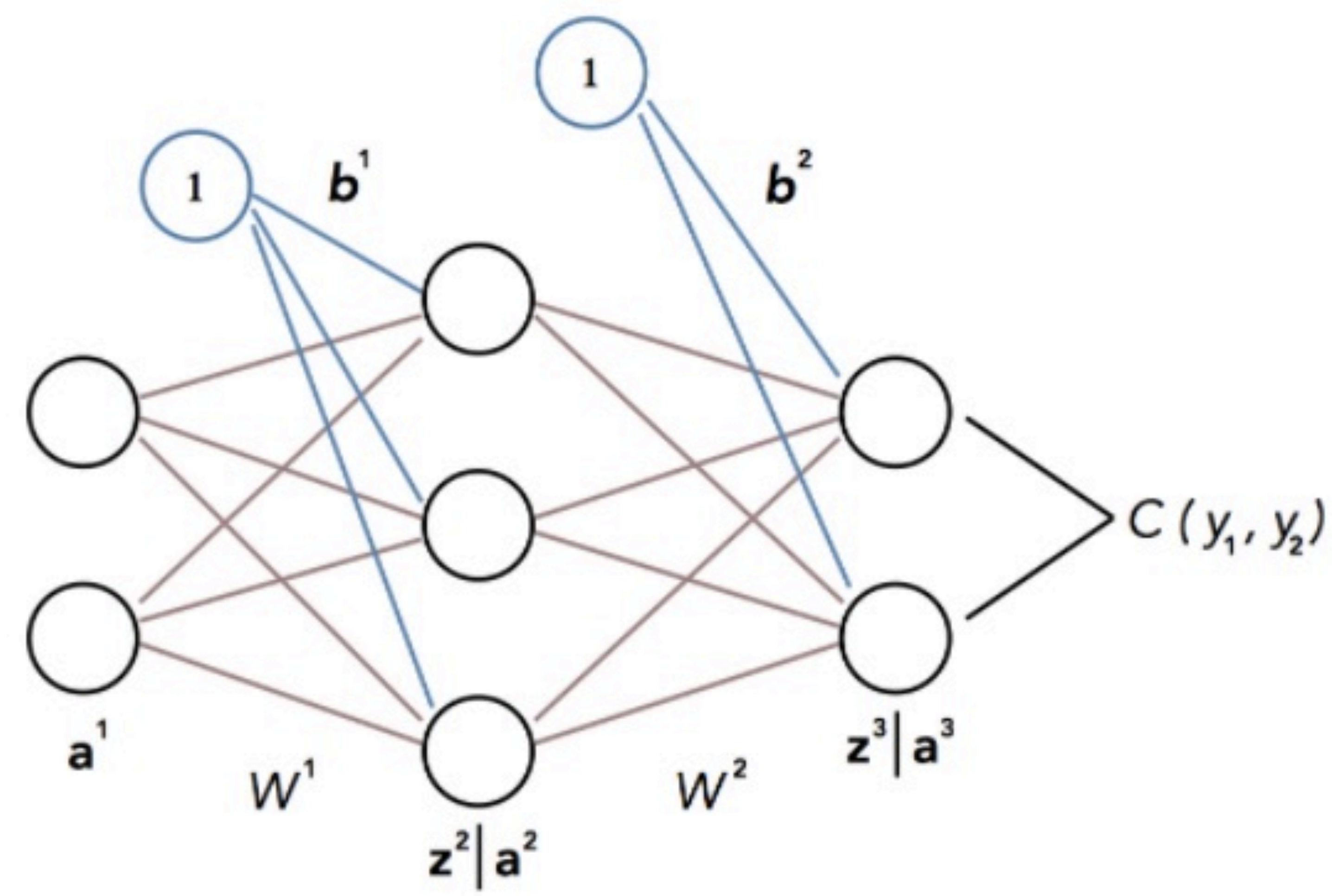
OK Great! Now we can easily-ish compute δ^3 for the output layer

But really we're after partials w.r.t. to weights and biases



Note that every weight connected to a node in the output layer depends on a single δ_i^3

Back Propagation

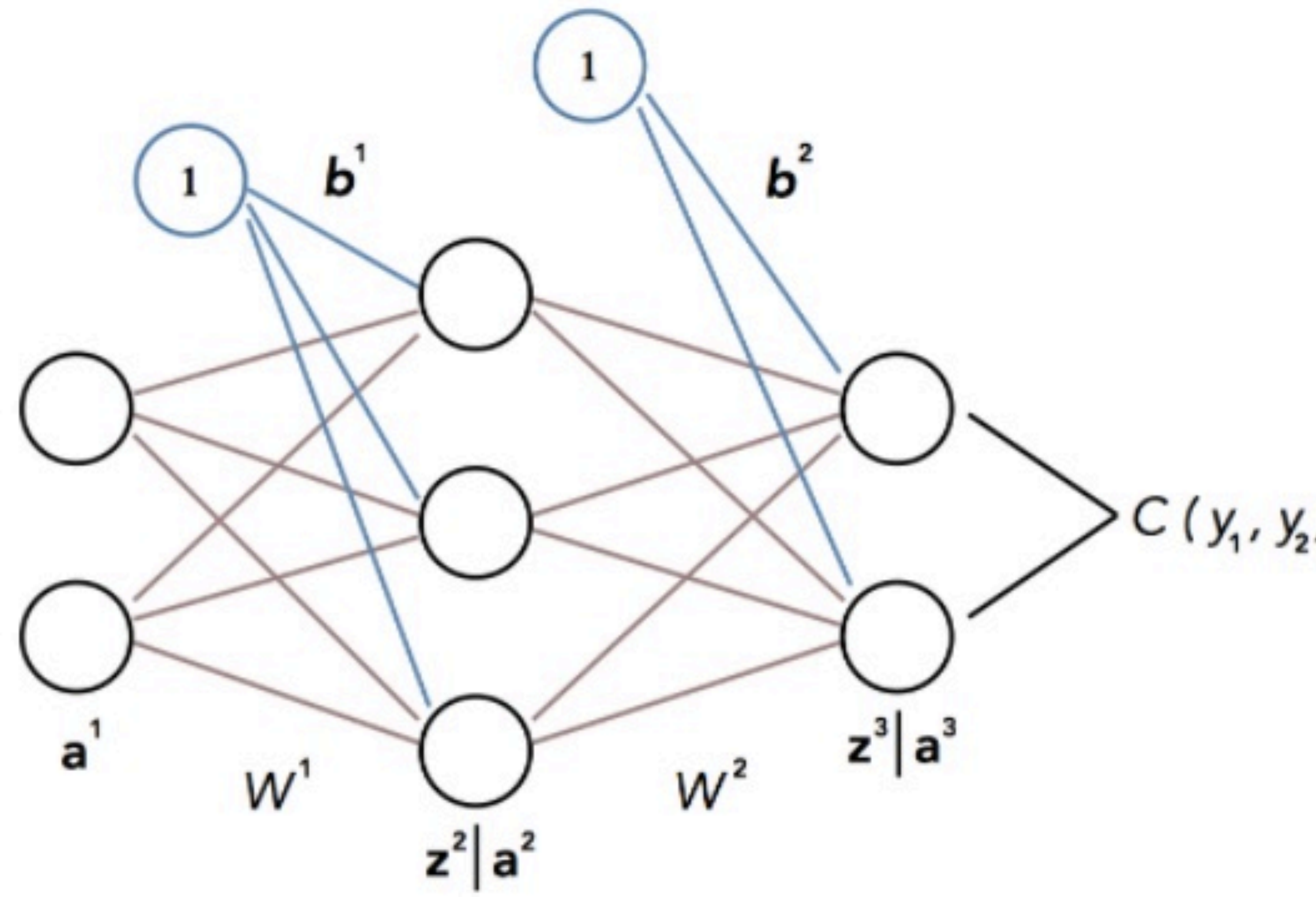


Let's take derivative w.r.t. w_{21}^2, we have $\frac{\partial C}{\partial w_{21}^2} = \frac{\partial C}{\partial z_2^3} \frac{\partial z_2^3}{\partial w_{21}^2} = \delta_2^3 \frac{\partial z_2^3}{\partial w_{21}^2}$

Recall that $\mathbf{z}^3 = W^2 \mathbf{a}^2 + \mathbf{b}^2$ and the 2^{nd} entry of $\mathbf{z}^3$ is

$$z_2^3 = \mathbf{w_{21}^2} a_1^2 + w_{22}^2 a_2^2 + w_{23}^2 a_3^2 + b_2^2$$

Back Propagation

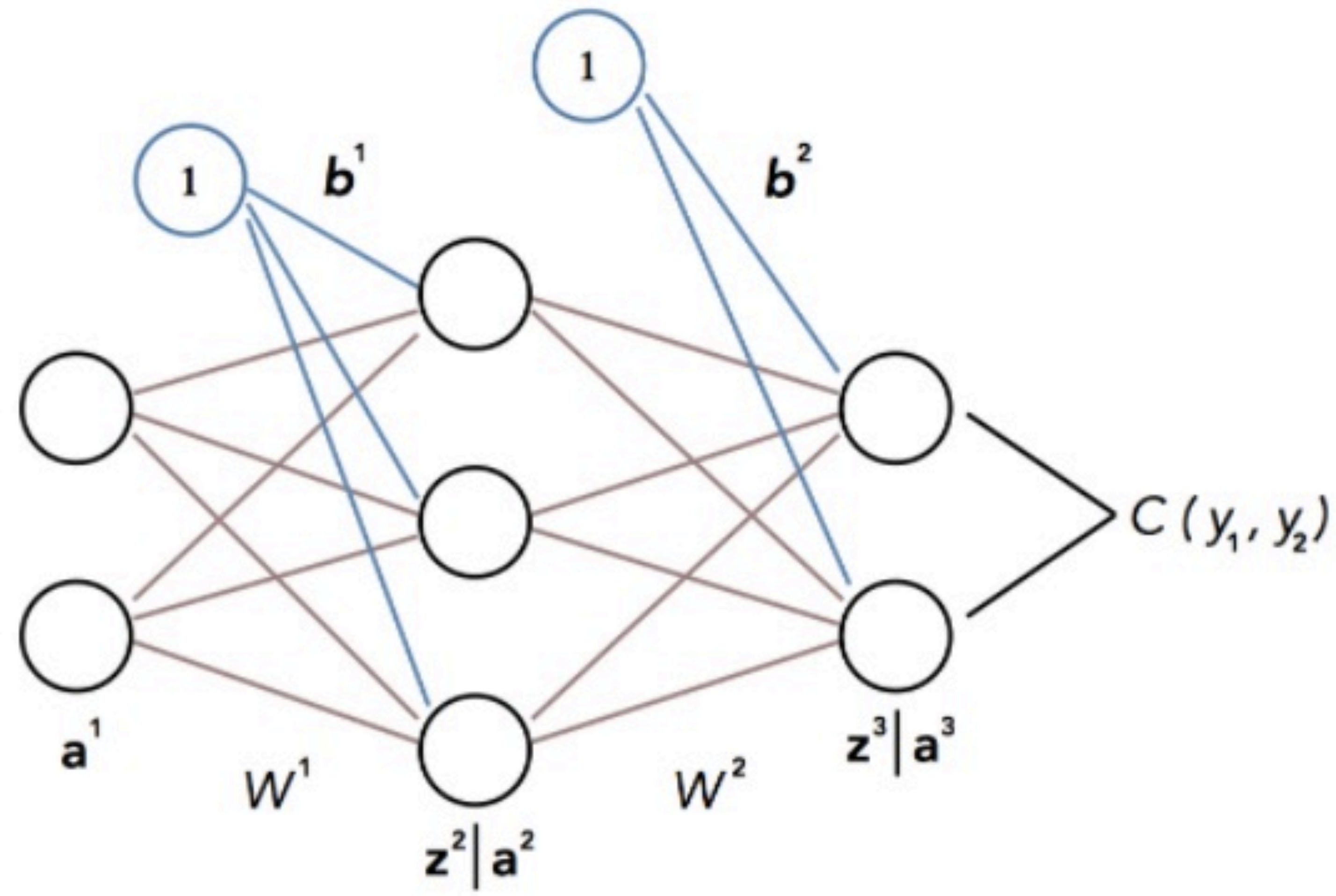


Let's take derivative w.r.t. w_{21}^2, we have $\frac{\partial C}{\partial w_{21}^2} = \frac{\partial C}{\partial z_2^3}\frac{\partial z_2^3}{\partial w_{21}^2} = \delta_2^3 \frac{\partial z_2^3}{\partial w_{21}^2}$

Taking derivative w.r.t. w_{21}^2 gives $\frac{\partial z_2^3}{\partial w_{21}^2} = a_1^2$

And plugging this in gives $\frac{\partial C}{\partial w_{21}^2} = \delta_2^3 a_1^2$

Back Propagation



For the general derivative of C wrt w^2_{ij} we have

$$\frac{\partial C}{\partial w^2_{ij}} = \delta^3_i a^2_j \quad \text{for } \; i = 1,2 \; \text{ and } \; j = 1,2,3$$

Back Propagation

Let's make the notation a little more practical

We have weights stored in a matrix $W^2 = \begin{bmatrix} w^2_{11} & w^2_{12} & w^2_{13} \\ w^2_{21} & w^2_{22} & w^2_{23} \end{bmatrix}$

Let's store the derivatives in a matrix of the same shape

$$\frac{\partial C}{\partial W^2} = \begin{bmatrix} \frac{\partial C}{\partial w^2_{11}} & \frac{\partial C}{\partial w^2_{12}} & \frac{\partial C}{\partial w^2_{13}} \\ \frac{\partial C}{\partial w^2_{21}} & \frac{\partial C}{\partial w^2_{22}} & \frac{\partial C}{\partial w^2_{23}} \end{bmatrix} = \begin{bmatrix} \delta^3_1 a^2_1 & \delta^3_1 a^2_2 & \delta^3_1 a^2_3 \\ \delta^3_2 a^2_1 & \delta^3_2 a^2_2 & \delta^3_2 a^2_3 \end{bmatrix}$$

Finally, we can write this as an outer-product of vectors δ^3 and $\mathbf{a}^2$

$$\frac{\partial C}{\partial W^2} = \delta^3 (\mathbf{a}^2)^T$$

Back Propagation

So, after forward prop have all of the activity vectors $\mathbf{z}^\ell$ and all of the activation vectors $\mathbf{a}^\ell$

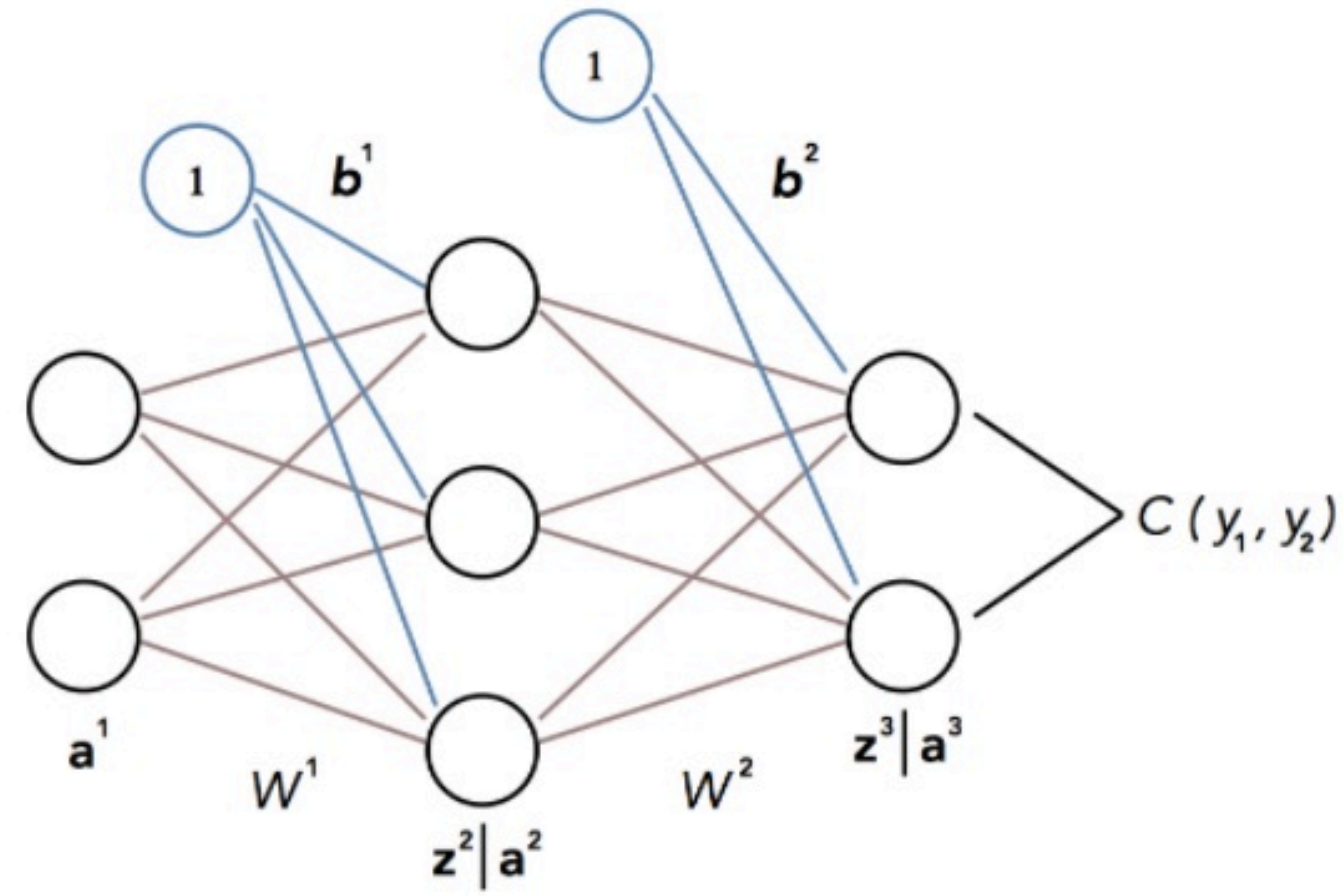
The derivatives of C w.r.t. the weights connected to the output layer are given by the matrix

$$\frac{\partial C}{\partial W^2} = \delta^3 (\mathbf{a}^2)^T$$

Notice that if we've hardcoded the derivate of C and the derivative of the activation function, then this update can be done will all vectorized operations

OK, cool! But what about the biases $\mathbf{b}^2$?

Back Propagation

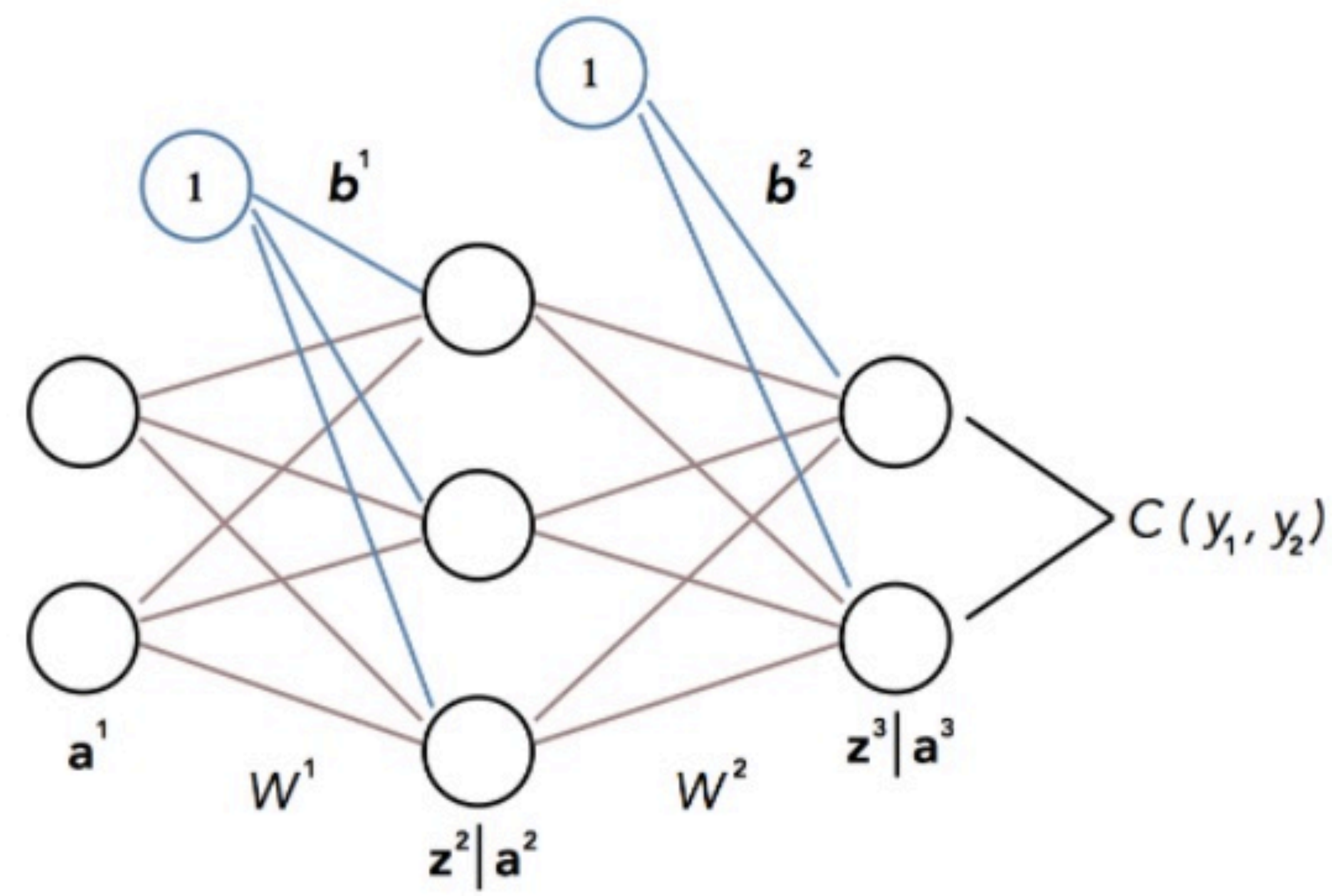


Let's take derivative w.r.t. b_1^2, we have $\frac{\partial C}{\partial b_1^2} = \frac{\partial C}{\partial z_1^3}\frac{\partial z_1^3}{\partial b_1^2} = \delta_1^3 \frac{\partial z_1^3}{\partial b_1^2}$

Recall that $\mathbf{z}^3 = W^2\mathbf{a}^2 + \mathbf{b}^2$ and the 1^{st} entry of $\mathbf{z}^3$ is

$$z_1^3 = \mathbf{w}^{\mathbf{2}}_{\mathbf{11}} a_1^2 + w_{12}^2 a_2^2 + w_{13}^2 a_3^2 + b_1^1$$

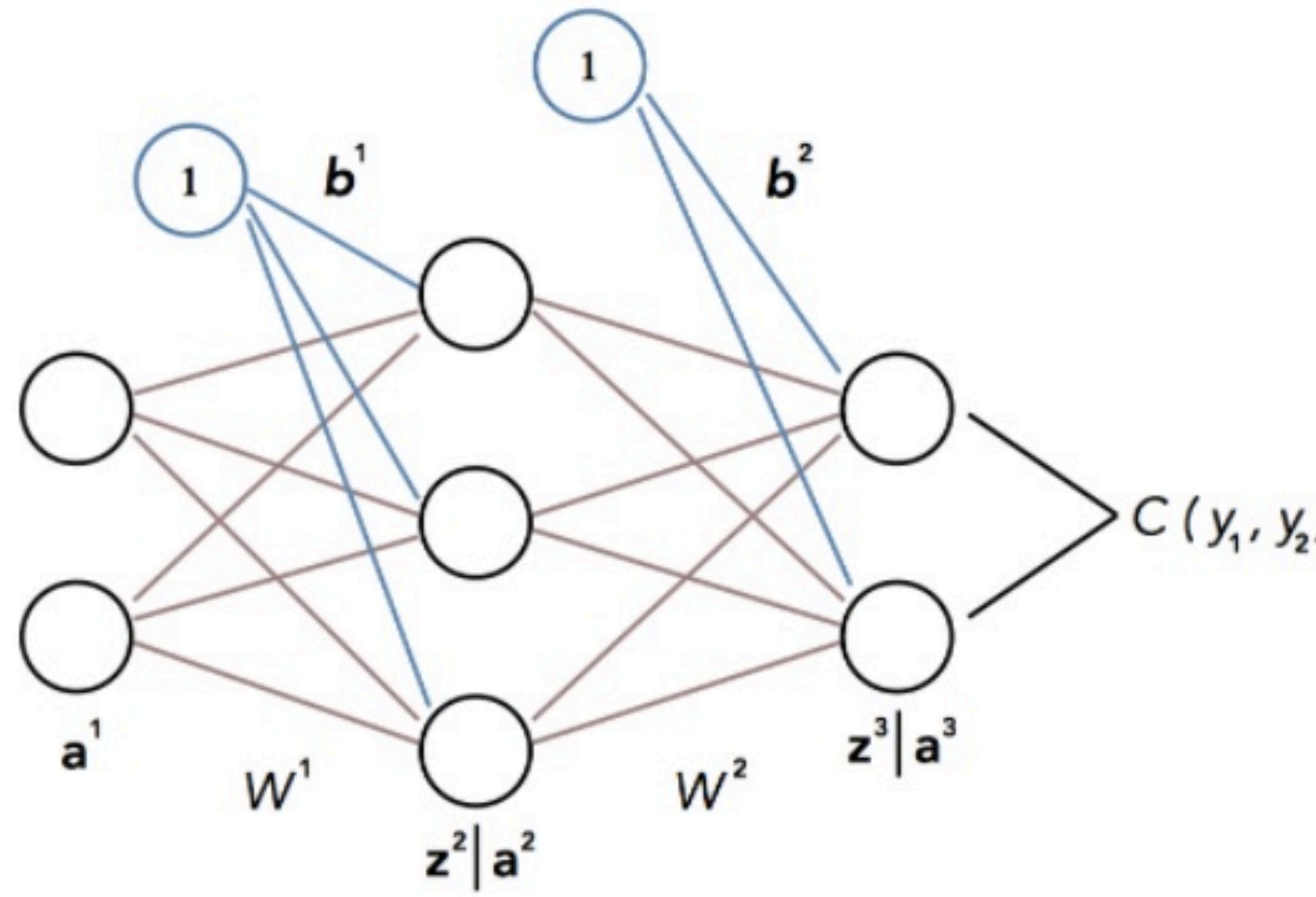
Back Propagation



Let's take derivative w.r.t. b_1^2, we have $\frac{\partial C}{\partial b_1^2} = \frac{\partial C}{\partial z_1^3} \frac{\partial z_1^3}{\partial b_1^2} = \delta_1^3 \frac{\partial z_1^3}{\partial b_1^2}$

Taking derivative w.r.t. b_1^2 gives $\frac{\partial z_1^3}{\partial b_1^2} = 1$, and plugged in $\frac{\partial C}{\partial b_1^2} = \delta_1^3$

Back Propagation



For the general derivative of C w.r.t. b_i^2 we have for $i = 1, 2$

$$\frac{\partial C}{\partial b_i^2} = \delta_i^3 \quad \text{and vectorized} \quad \frac{\partial C}{\partial \mathbf{b}^2} = \delta^3$$

Back Propagation

Intermediate Summary:

For a given training example $\mathbf{x}_k$, perform forward propagation to get activities $\mathbf{z}^\ell$ and activations $\mathbf{a}^\ell$ on each layer

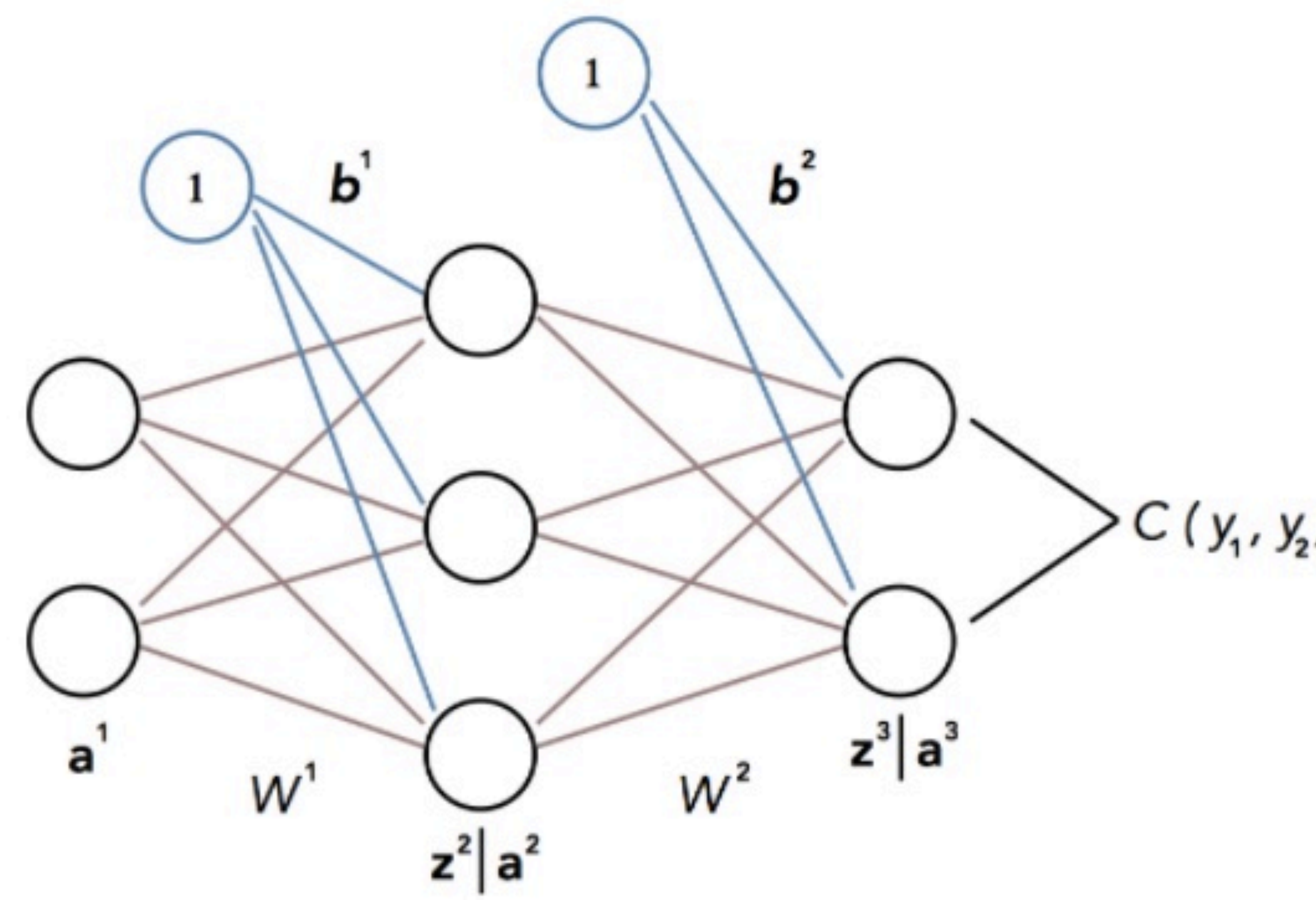
Then to get the partial derivatives we:

1. Compute $\delta^3 = \nabla_{\mathbf{a}^3} C \odot g'(\mathbf{z}^3)$
2. Compute $\dfrac{\partial C}{\partial W^2} = \delta^3 (\mathbf{a}^2)^T$ and $\dfrac{\partial C}{\partial \mathbf{b}^2} = \delta^3$

OK, that wasn't so bad! We found very simple expressions for the derivatives with respect to the weights in the last hidden layer!

Problem: How do we do the weights W^1 and biases $\mathbf{b}^1$?

Back Propagation



The formulas for W^2 and $\mathbf{b}^2$ were nice because we knew δ^3

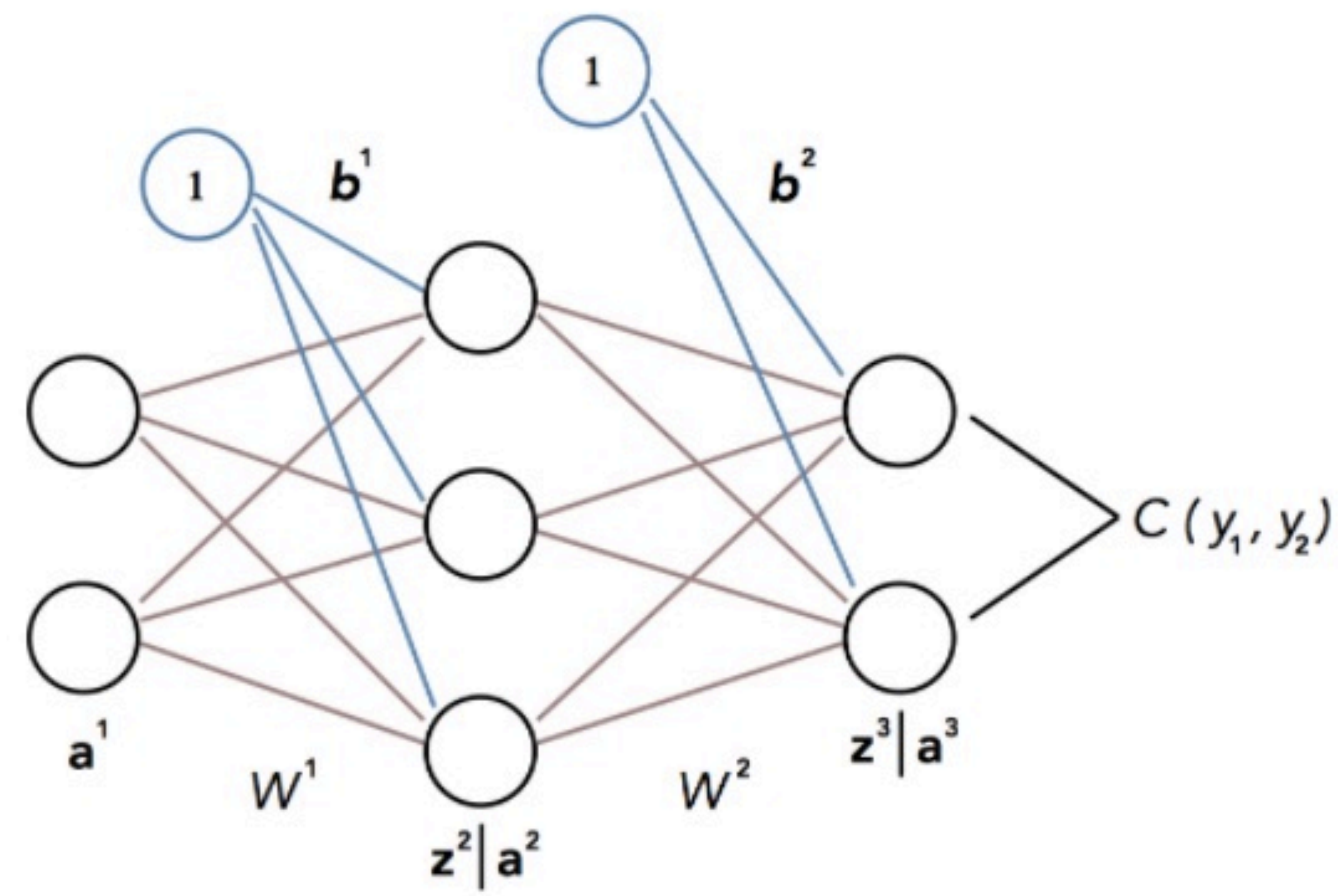
Seems like formulas for W^1 and $\mathbf{b}^1$ would be nice if we knew δ^2

But the relationship between C and $\mathbf{z}^2$ is really complicated because of multiple passes through the activation functions :(

Back Propagation

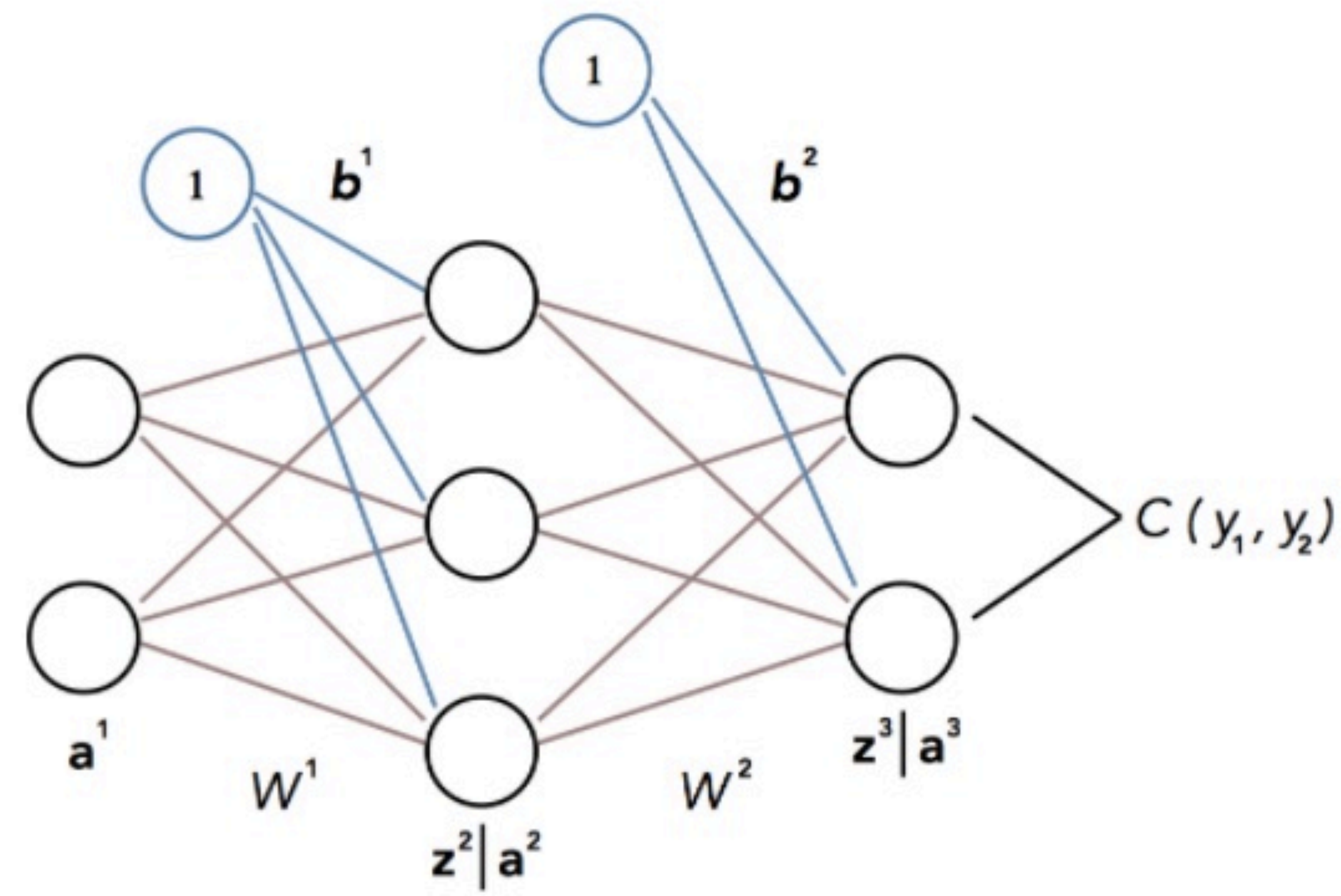
But it's OK! There's a nice-ish process for **propagating** δ's **backwards** through the layers

Notice that δ_2^2 depends on both δ_1^3 and δ_2^3



Back Propagation

Notice that δ_2^2 depends on both δ_1^3 and δ_2^3

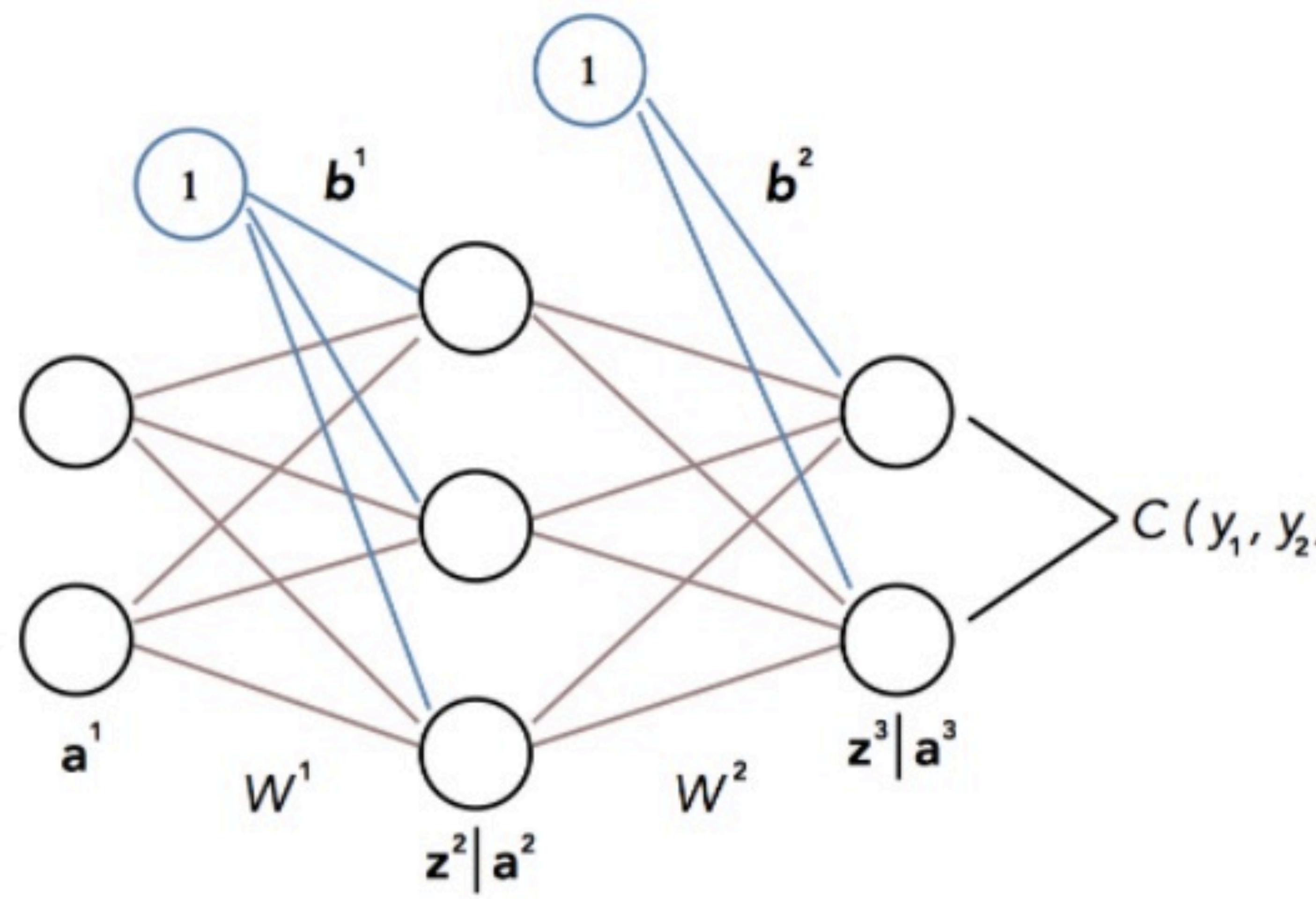


By the Adult Chain Rule:

$$\delta_2^2 = \frac{\partial C}{\partial z_2^2} = \frac{\partial C}{\partial z_1^3}\frac{\partial z_1^3}{\partial z_2^2} + \frac{\partial C}{\partial z_2^3}\frac{\partial z_2^3}{\partial z_2^2}$$

Back Propagation

Notice that δ_2^2 depends on both δ_1^3 and δ_2^3



By the Adult Chain Rule:

$$\delta_2^2 = \frac{\partial C}{\partial z_2^2} = \delta_1^3 \frac{\partial z_1^3}{\partial z_2^2} + \delta_2^3 \frac{\partial z_2^3}{\partial z_2^2}$$

Back Propagation

$$\delta_2^2 = \frac{\partial C}{\partial z_2^2} = \delta_1^3 \frac{\partial z_1^3}{\partial z_2^2} + \delta_2^3 \frac{\partial z_2^3}{\partial z_2^2}$$

Now we just need to compute derivatives of the level 3 activities w.r.t. to the level 2 activities

Recall that $\mathbf{z}^3 = W^2\mathbf{a}^2 + \mathbf{b}^2$, the i^{th} entry of which is given by

$$z_i^3 = w_{i1}^2 a_1^2 + w_{i2}^2 a_2^2 + w_{i3}^2 a_3^2 + b_i^2$$

Back Propagation

$$\delta_2^2 = \frac{\partial C}{\partial z_2^2} = \delta_1^3 \frac{\partial z_1^3}{\partial z_2^2} + \delta_2^3 \frac{\partial z_2^3}{\partial z_2^2}$$

Now we just need to compute derivatives of the level 3 activities w.r.t. to the level 2 activities

Recall that $\mathbf{z}^3 = W^2 \mathbf{a}^2 + \mathbf{b}^2$, the i^{th} entry of which is given by

$$z_i^3 = w_{i1}^2 a_1^2 + w_{i2}^2 a_2^2 + w_{i3}^2 a_3^2 + b_i^2$$

But the activations are just the activities run through g

$$z_i^3 = w_{i1}^2 g(z_1^2) + w_{i2}^2 g(z_2^2) + w_{i3}^2 g(z_3^2) + b_i^2$$

Back Propagation

$$\delta_2^2 = \frac{\partial C}{\partial z_2^2} = \delta_1^3 \frac{\partial z_1^3}{\partial z_2^2} + \delta_2^3 \frac{\partial z_2^3}{\partial z_2^2}$$

Now we just need to compute derivatives of the level 3 activities w.r.t. to the level 2 activities

Recall that $\mathbf{z}^3 = W^2 \mathbf{a}^2 + \mathbf{b}^2$, the i^{th} entry of which is given by

$$z_i^3 = w_{i1}^2 g(z_1^2) + w_{i2}^2 g(z_2^2) + w_{i3}^2 g(z_3^2) + b_i^2$$

Taking the derivative $\dfrac{\partial z_i^3}{\partial z_2^2} = w_{i2}^2 g'(z_2^2)$, and plugging in gives

$$\delta_2^2 = \frac{\partial C}{\partial z_2^2} = \delta_1^3 \, w_{12}^2 \, g'(z_2^2) + \delta_2^3 \, w_{22}^2 \, g'(z_2^2)$$

Back Propagation

If we do this for each of the 3 δ_i^2's something nice happens

(**EFY**: Work out δ_1^2 and δ_3^2 for yourself)

$$
\begin{aligned}
\delta_1^2 &= \delta_1^3 \, w_{11}^2 \, g'(z_1^2) + \delta_2^3 \, w_{21}^2 \, g'(z_1^2) \\
\delta_2^2 &= \delta_1^3 \, w_{12}^2 \, g'(z_2^2) + \delta_2^3 \, w_{22}^2 \, g'(z_2^2) \\
\delta_3^2 &= \delta_1^3 \, w_{13}^2 \, g'(z_3^2) + \delta_2^3 \, w_{23}^2 \, g'(z_3^2)
\end{aligned}
$$

Notice that each row of the system gets multiplied by $g'(z_i^2)$, so let's factor those out

Back Propagation

If we do this for each of the 3 δ_i^2's something nice happens

(**EFY**: Work out δ_1^2 and δ_3^2 for yourself)

$$\begin{aligned}
\delta_1^2 &= (\delta_1^3 \, w_{11}^2 + \delta_2^3 \, w_{21}^2) \cdot g'(z_1^2) \\
\delta_2^2 &= (\delta_1^3 \, w_{12}^2 + \delta_2^3 \, w_{22}^2) \cdot g'(z_2^2) \\
\delta_3^2 &= (\delta_1^3 \, w_{13}^2 + \delta_2^3 \, w_{23}^2) \cdot g'(z_3^2)
\end{aligned}$$

Remember that δ^3 and W^2 looked as follows

$$\delta^3 = \begin{bmatrix} \delta_1^3 \\ \delta_2^3 \end{bmatrix}, \qquad W^2 = \begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \end{bmatrix}$$

Do you see δ^3 and W^2 lurking anywhere in the above system?

Back Propagation

If we do this for each of the 3 δ_i^2's something nice happens

(**EFY**: Work out δ_1^2 and δ_3^2 for yourself)

$$
\begin{aligned}
\delta_1^2 &= (\delta_1^3 \, w_{11}^2 + \delta_2^3 \, w_{21}^2) \cdot g'(z_1^2) \\
\delta_2^2 &= (\delta_1^3 \, w_{12}^2 + \delta_2^3 \, w_{22}^2) \cdot g'(z_2^2) \\
\delta_3^2 &= (\delta_1^3 \, w_{13}^2 + \delta_2^3 \, w_{23}^2) \cdot g'(z_3^2)
\end{aligned}
$$

Does this help?

$$
(W^2)^T = \begin{bmatrix} w_{11}^2 & w_{21}^2 \\ w_{12}^2 & w_{22}^2 \\ w_{13}^2 & w_{23}^2 \end{bmatrix}, \qquad \delta^3 = \begin{bmatrix} \delta_1^3 \\ \delta_2^3 \end{bmatrix}
$$

Back Propagation

If we do this for each of the 3 δ_i^2's something nice happens

(**EFY**: Work out δ_1^2 and δ_3^2 for yourself)

$$
\begin{aligned}
\delta_1^2 &= (\delta_1^3\, w_{11}^2 + \delta_2^3\, w_{21}^2) \cdot g'(z_1^2) \\
\delta_2^2 &= (\delta_1^3\, w_{12}^2 + \delta_2^3\, w_{22}^2) \cdot g'(z_2^2) \\
\delta_3^2 &= (\delta_1^3\, w_{13}^2 + \delta_2^3\, w_{23}^2) \cdot g'(z_3^2)
\end{aligned}
$$

How about now?

$$
\begin{bmatrix} \delta_1^2 \\ \delta_2^2 \\ \delta_3^2 \end{bmatrix} = \begin{bmatrix} w_{11}^2 & w_{21}^2 \\ w_{12}^2 & w_{22}^2 \\ w_{13}^2 & w_{23}^2 \end{bmatrix} \begin{bmatrix} \delta_1^3 \\ \delta_2^3 \end{bmatrix} \odot \begin{bmatrix} g'(z_1^2) \\ g'(z_2^2) \\ g'(z_3^2) \end{bmatrix} = (W^2)^T \delta^3 \odot g'(\mathbf{z}^2)
$$

Back Propagation

OK Great!

- With δ^3 we can easily compute δ^2

Then we can compute derivatives of C w.r.t. weights W^1 and biases $\mathbf{b}^1$ exactly the way we did for W^2 and $\mathbf{b}^2$

1. Compute $\delta^2 = (W^2)^T \delta^3 \odot g'(\mathbf{z}^2)$
2. Compute $\dfrac{\partial C}{\partial W^1} = \delta^2 (\mathbf{a}^1)^T$ and $\dfrac{\partial C}{\partial \mathbf{b}^1} = \delta^2$

We've worked this out for a simple network with one hidden layer.

Nothing we've done assumed anything about the number of layers, so we can use the same procedure for any number of layers

Back Propagation

For a general L-layer network we have the following:

$\delta^L = \nabla_a C \odot g'(\mathbf{z}^L)$ # Compute δ's on output layer

For $\ell = L - 1, \ldots, 2$

$$\frac{\partial C}{\partial W^\ell} = \delta^{\ell+1}(a^\ell)^T \quad \text{\# Compute weight derivatives}$$

$$\frac{\partial C}{\partial b^\ell} = \delta^{\ell+1} \quad \text{\# Compute bias derivatives}$$

$$\delta^\ell = \left(W^\ell\right)^T \delta^{\ell+1} \odot g'(\mathbf{z}^\ell) \quad \text{\# Back prop } \delta\text{'s to previous layer}$$

Then SGD update: $W^\ell \leftarrow W^\ell - \eta \frac{\partial C}{\partial W^\ell}, \quad \mathbf{b}^\ell \leftarrow \mathbf{b}^\ell - \eta \frac{\partial C}{\partial \mathbf{b}^\ell}$

Training a Feed-Forward Neural Network

Given initial guess for weights and biases.

Loop over each training example in random order:

1. Forward Propagate to get activations on each layer
2. Back Propagate to get derivatives
3. Update weights and biases via Stochastic Gradient Descent
4. Rinse and Repeat

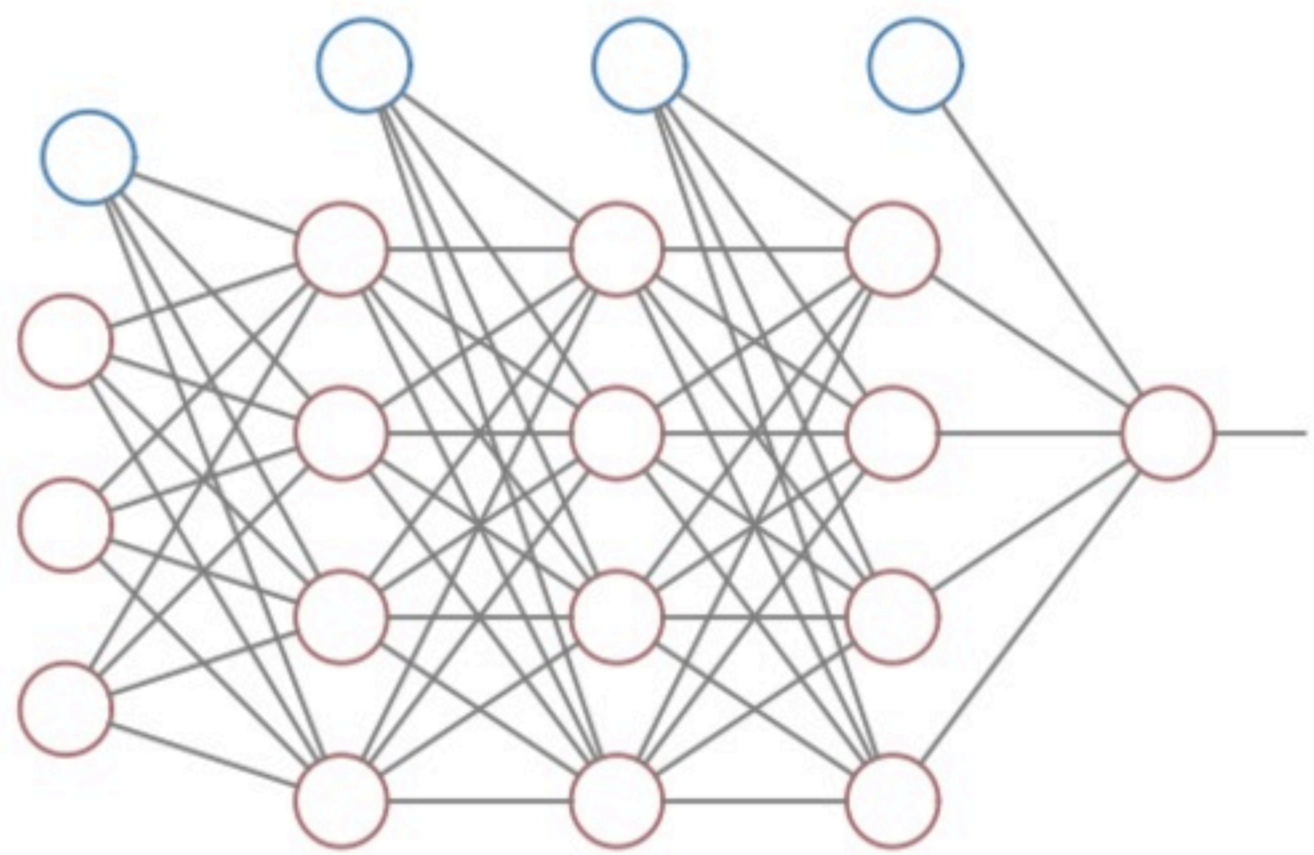
Training a Feed-Forward Neural Network

Looming Questions:

1. Should I worry about local minima?
2. How do we initialize weights and biases?
3. When do we stop?
4. Do I need to do regularization?
5. Can I batch this?

Training a Feed-Forward Neural Network

Should I worry about local minima?

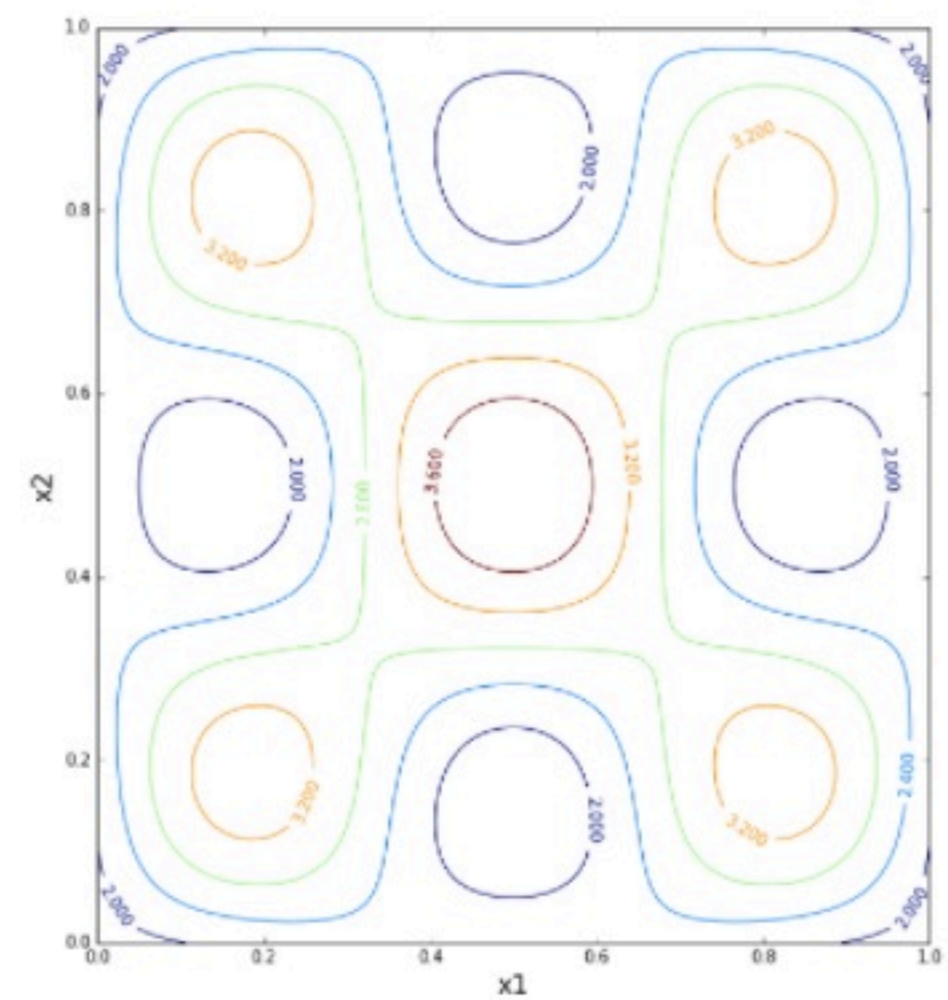


The cost function for a multilayer NN is decidedly non-convex

Swapping rows in Ws lead to same value of cost function

Training a Feed-Forward Neural Network

Should I worry about local minima?

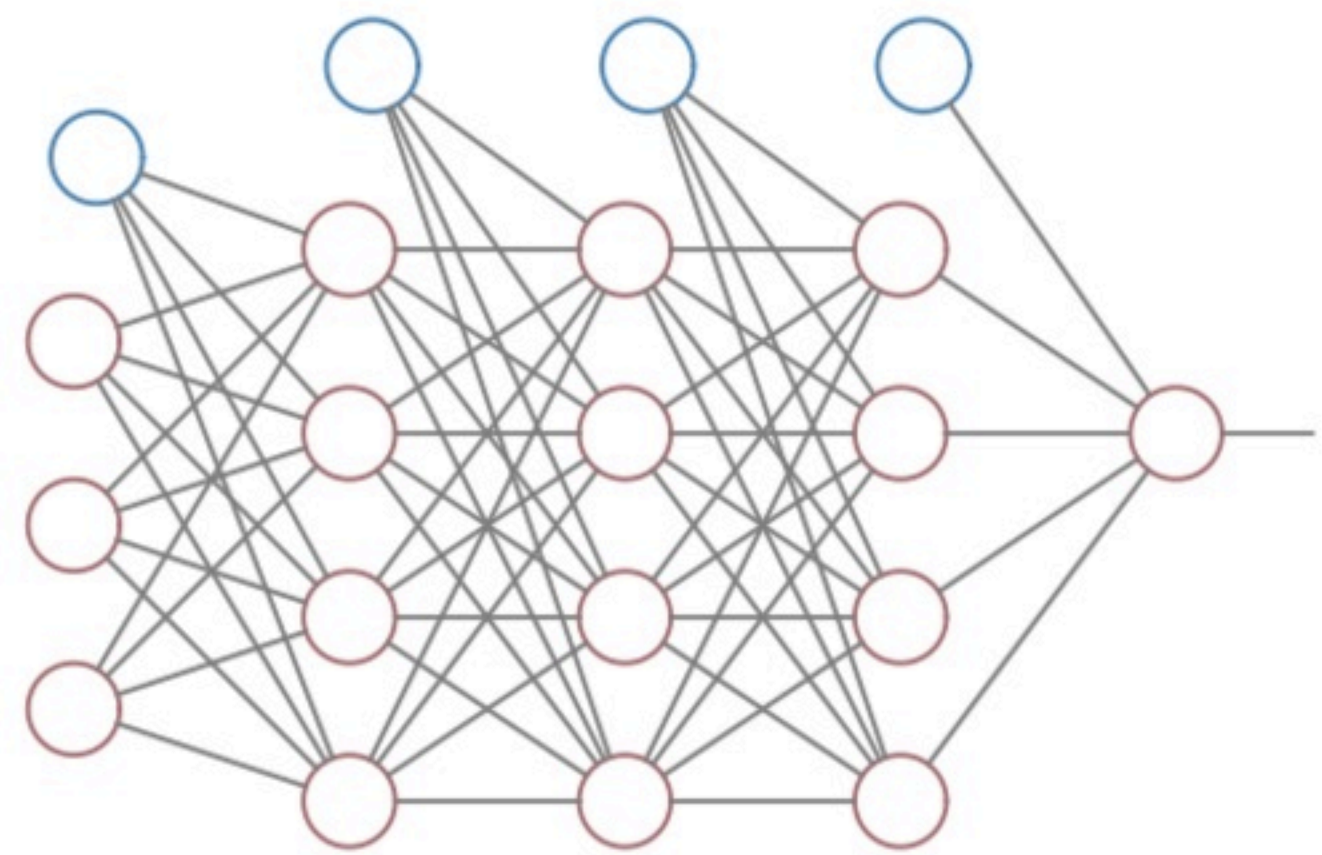


Small networks may have some bad local minima

Saddle points more of a problem for training

Training a Feed-Forward Neural Network

How should we initialize weights?



Initializing with all zeros would cause locking

Usually initialize with small normal random values

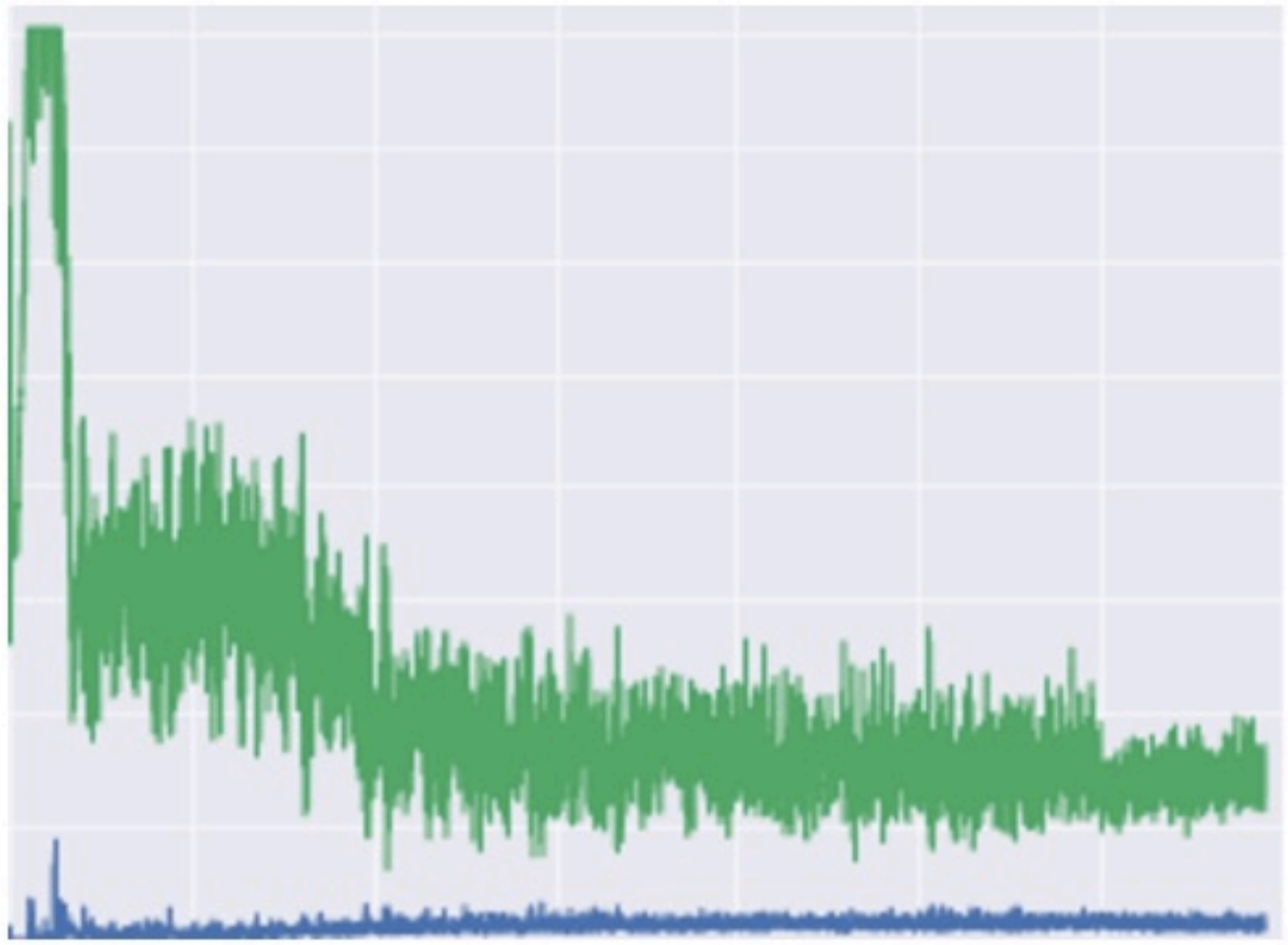
Training a Feed-Forward Neural Network

When do we stop?

In general, the same way you do in any SGD training

Monitor loss function on training and validation set

Stop when validation set stops improving



Training a Feed-Forward Neural Network

Should we do regularization?

Definitely in some form

Can use the usual L_2 -regularization by modifying cost function

$$C = C_0 + \frac{\lambda}{2} \sum_w w^2$$

Other popular forms of regularization:

- Early Stopping
- Dropout

Training a Feed-Forward Neural Network

Can I batch this?

Yes, and you probably should

We'll discuss the details of this in class and you'll explore it in the Homework

In Class

- **Your Questions!**
- Write our first back-prop code
- Talk more implementation details

Acknowledgements

Many of the slides in this lecture were adopted from Mike Mozer

The treatment of backprop was taken from Michael Neilsen's online book *Neural Networks and Deep Learning* available at neuralnetworksanddeeplearning.com

In Class

In Class

In Class

In Class

In Class