Neural Networks: Prediction (i.e. the forward pass)

Machine Learning

THE UNIVERSITY OF UTAH

Based on slides and material from Geoffrey Hinton, Richard Socher, Dan Roth, Yoav Goldberg, Shai Shalev-Shwartz and Shai Ben-David, and others

Neural Networks

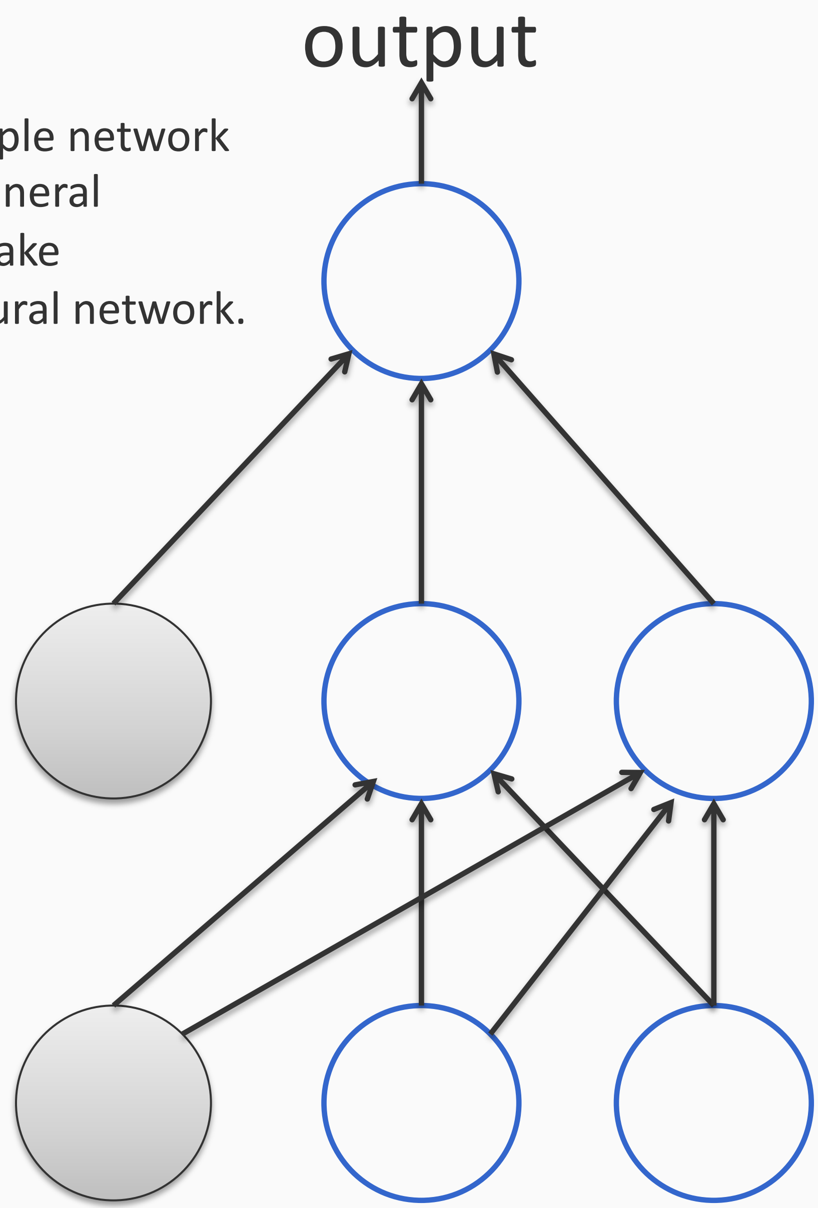
- What is a neural network?
- Predicting with a neural network
- Training neural networks
- Practical concerns

This lecture

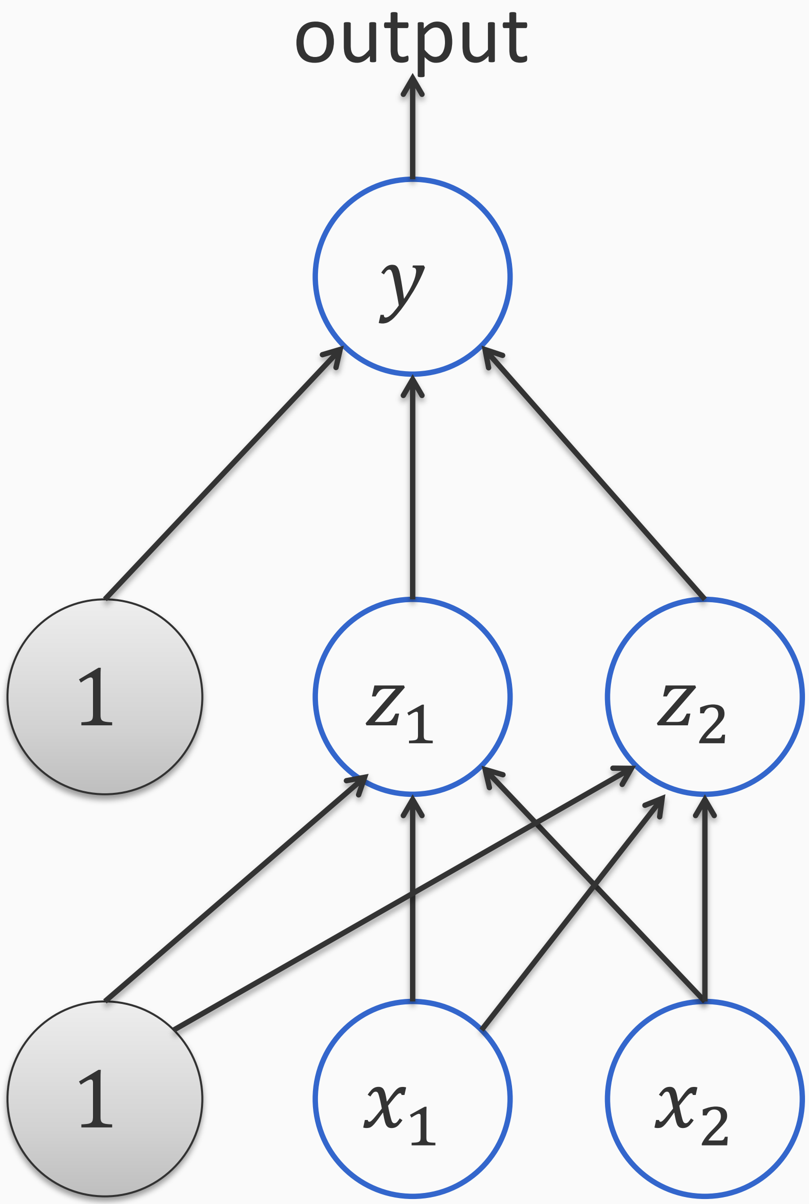
- What is a neural network?
- Predicting with a neural network
- Training neural networks
- Practical concerns

Let us consider an example network

We will use this example network as to introduce the general principle of how to make predictions with a neural network.



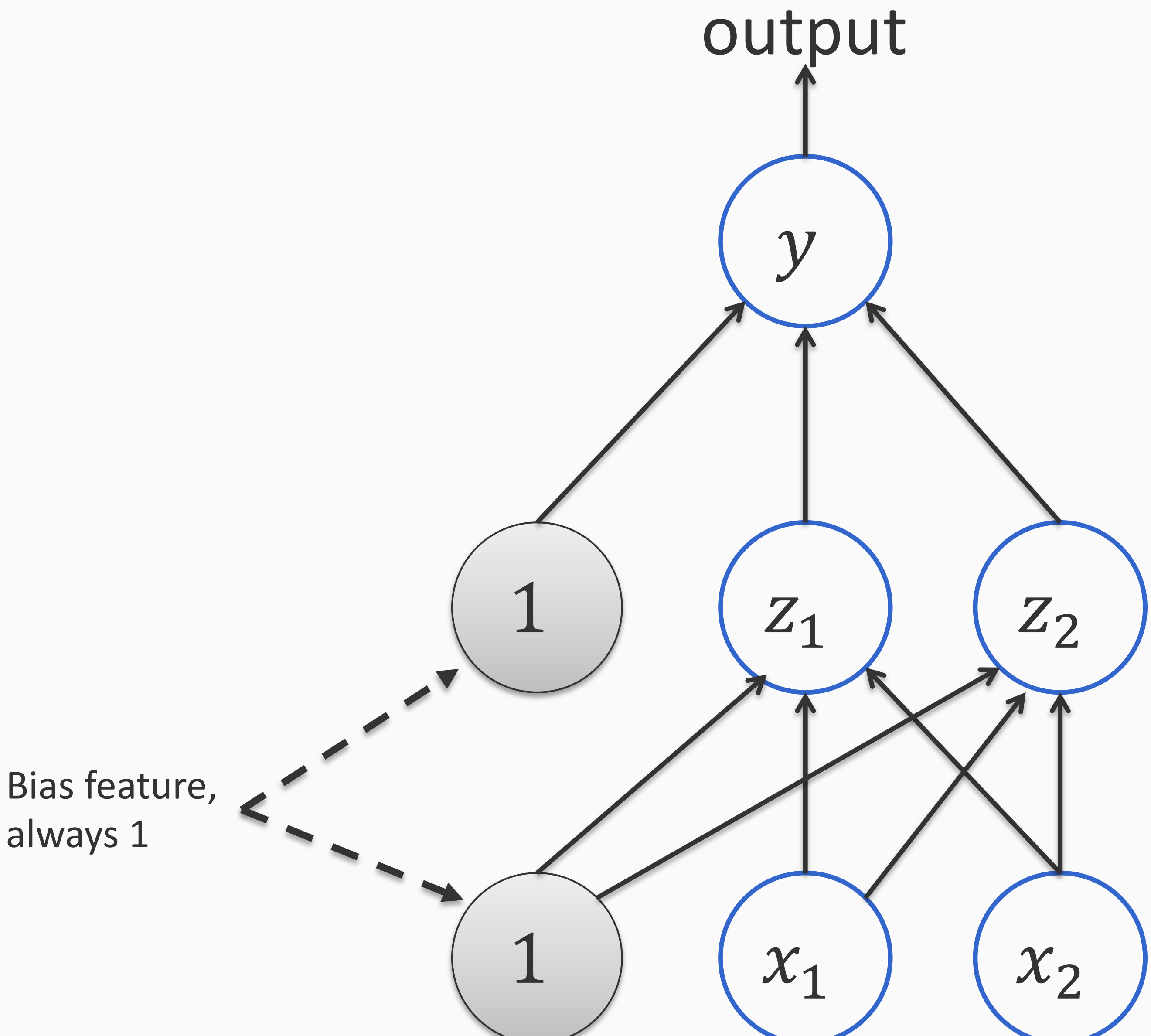
Let us consider an example network



Naming conventions for this example

- Inputs: x
- Hidden: z
- Output: y

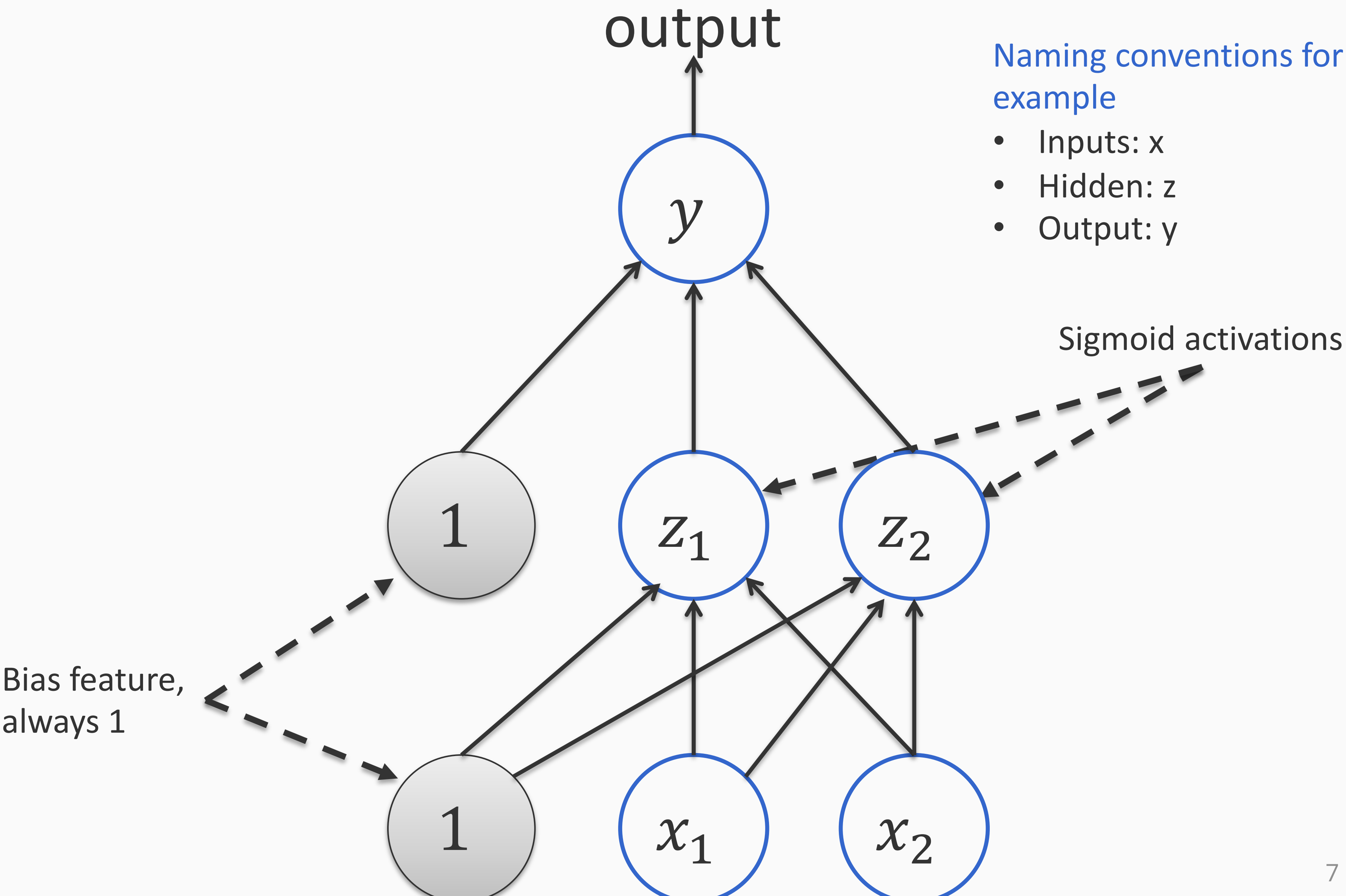
Let us consider an example network



Naming conventions for this example

- Inputs: x
- Hidden: z
- Output: y

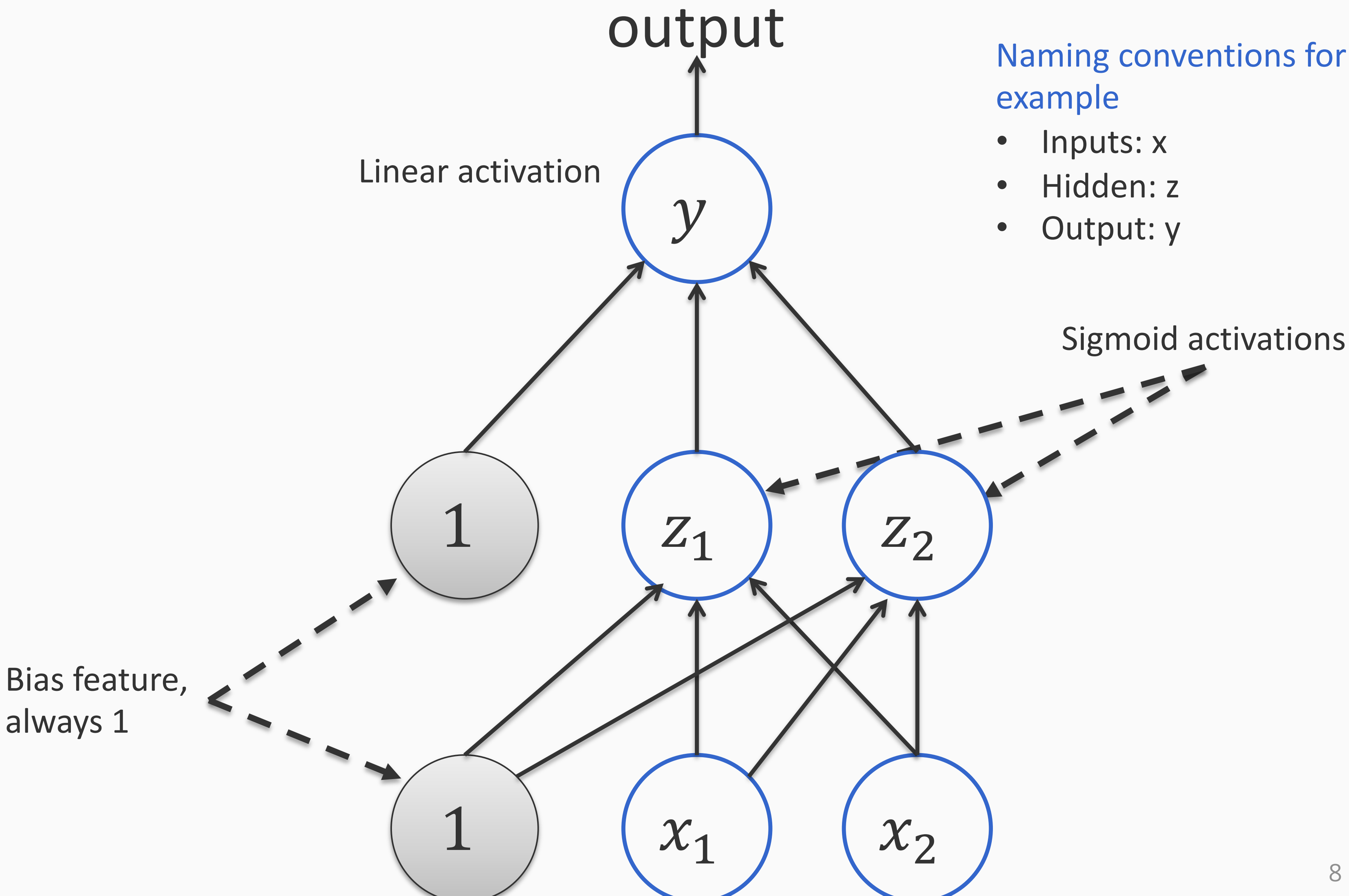
Let us consider an example network



Naming conventions for this example

- Inputs: x
- Hidden: z
- Output: y

Let us consider an example network



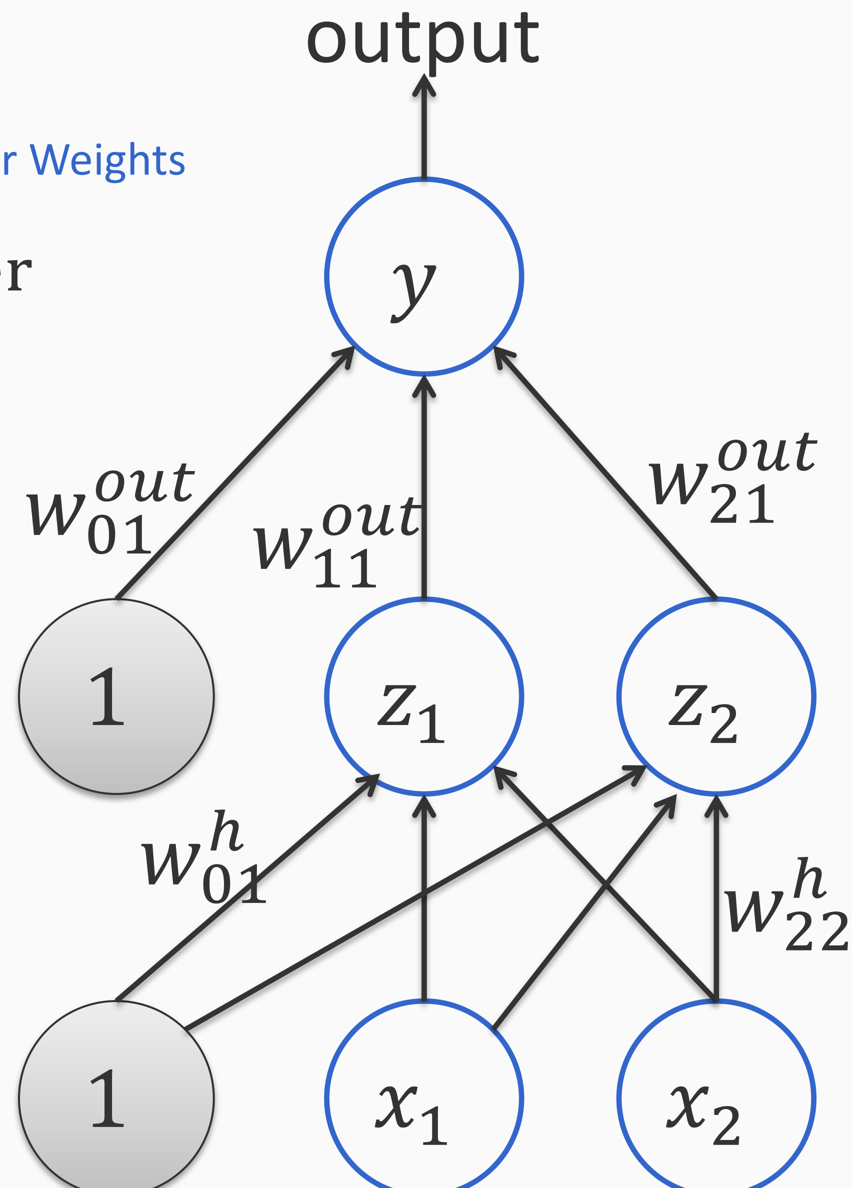
Naming conventions for this example

- Inputs: x
- Hidden: z
- Output: y

Let us consider an example network

Naming Convention for Weights

$$w_{\text{from,to}}^{\text{target layer}}$$



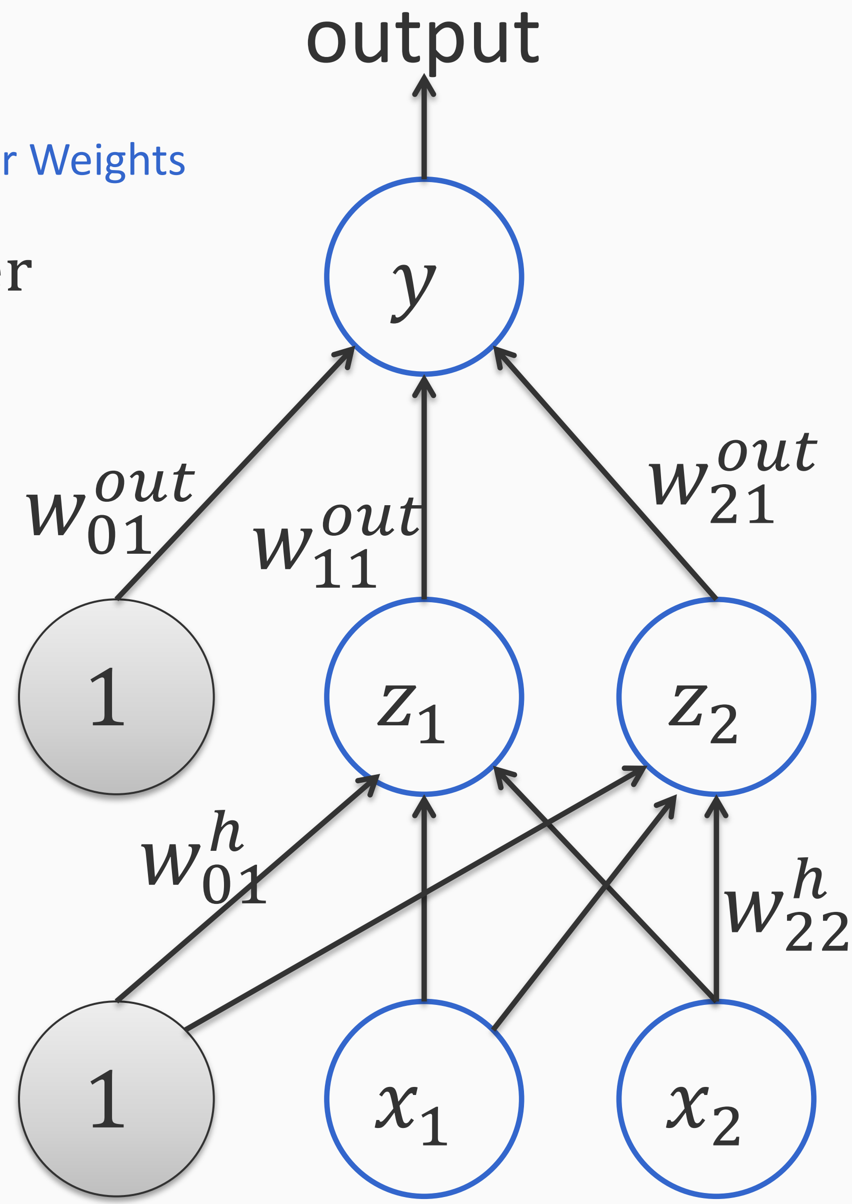
Let us consider an example network

Naming Convention for Weights

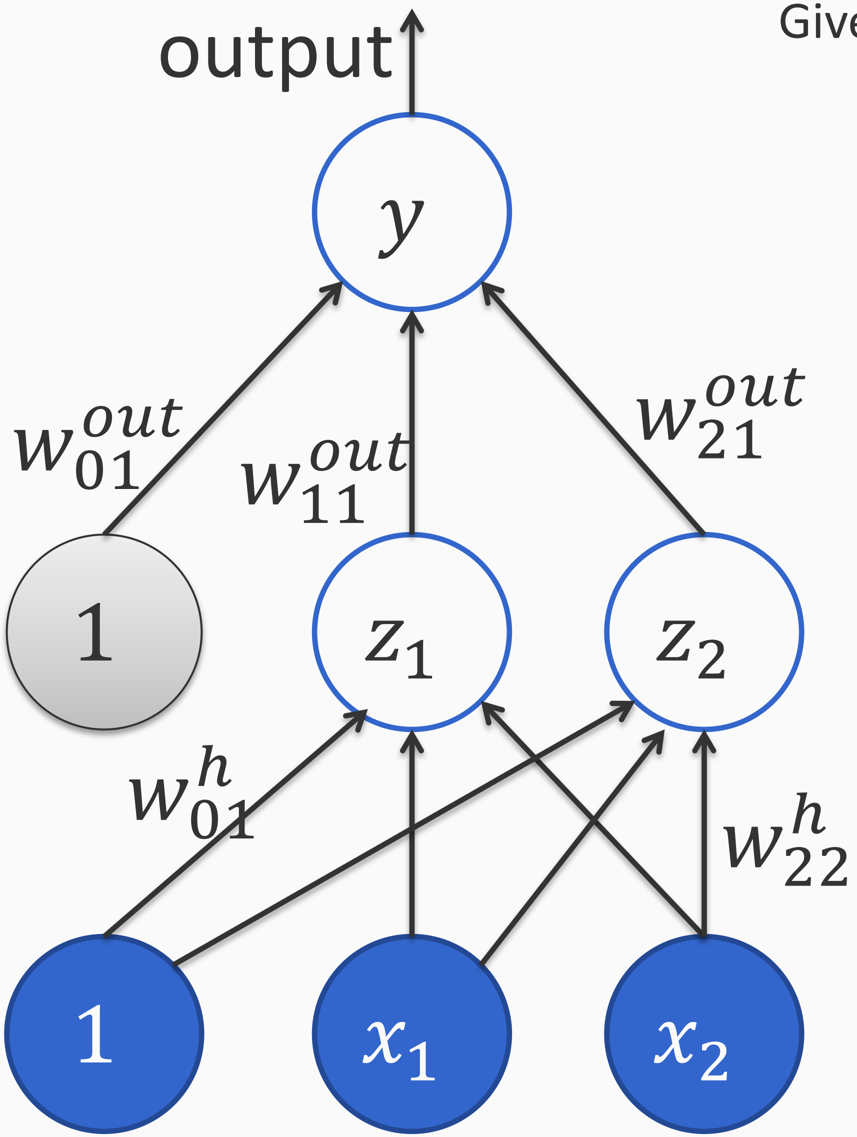
$$w_{\text{from,to}}^{\text{target-layer}}$$

$$w_{01}^{out}$$

From neuron #0 to neuron #1 in output layer

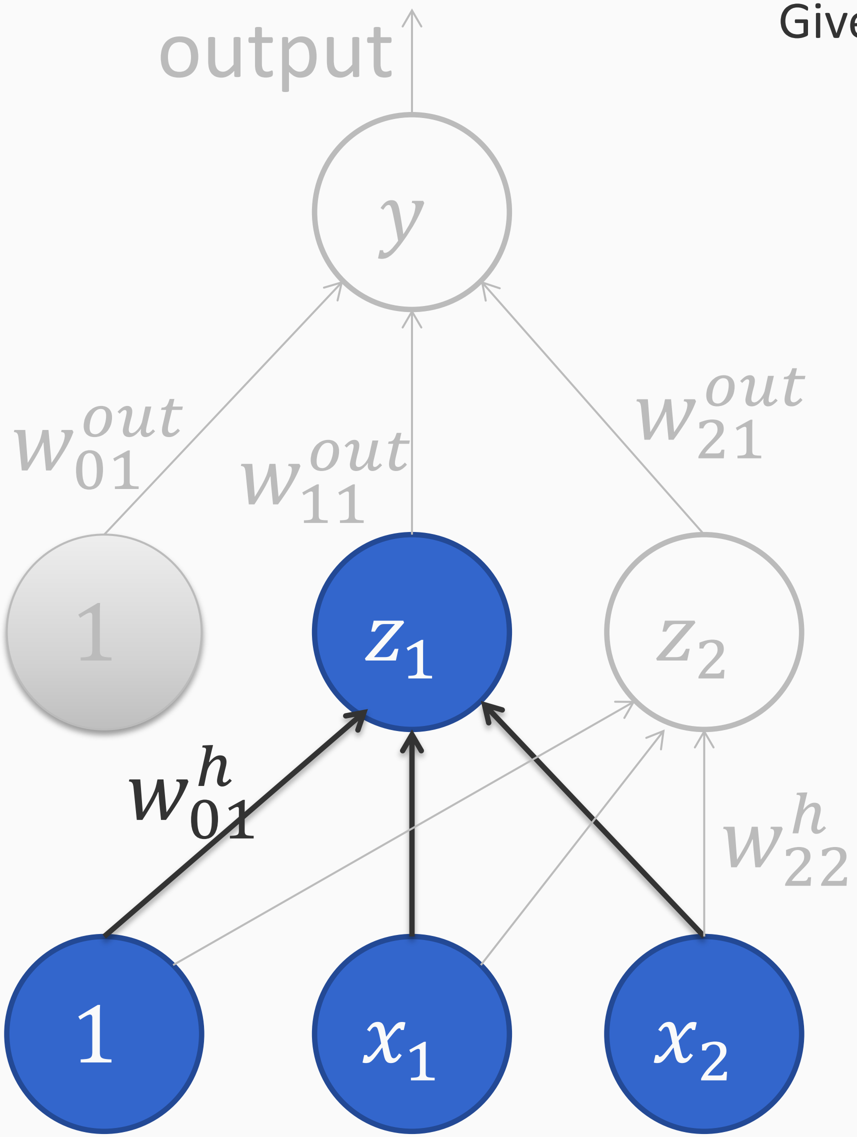


How to predict: The forward pass



Given an input **x**, how is the output predicted

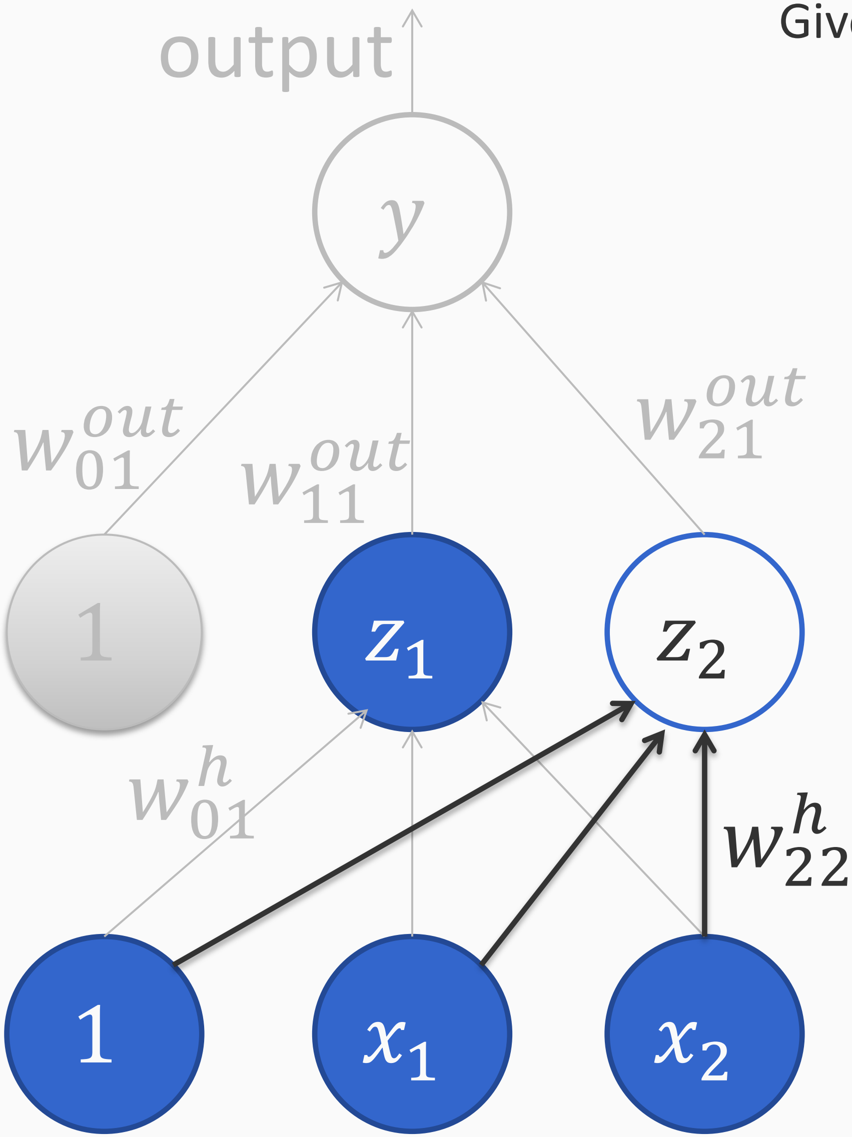
The forward pass



Given an input **x**, how is the output predicted

$$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$$

The forward pass

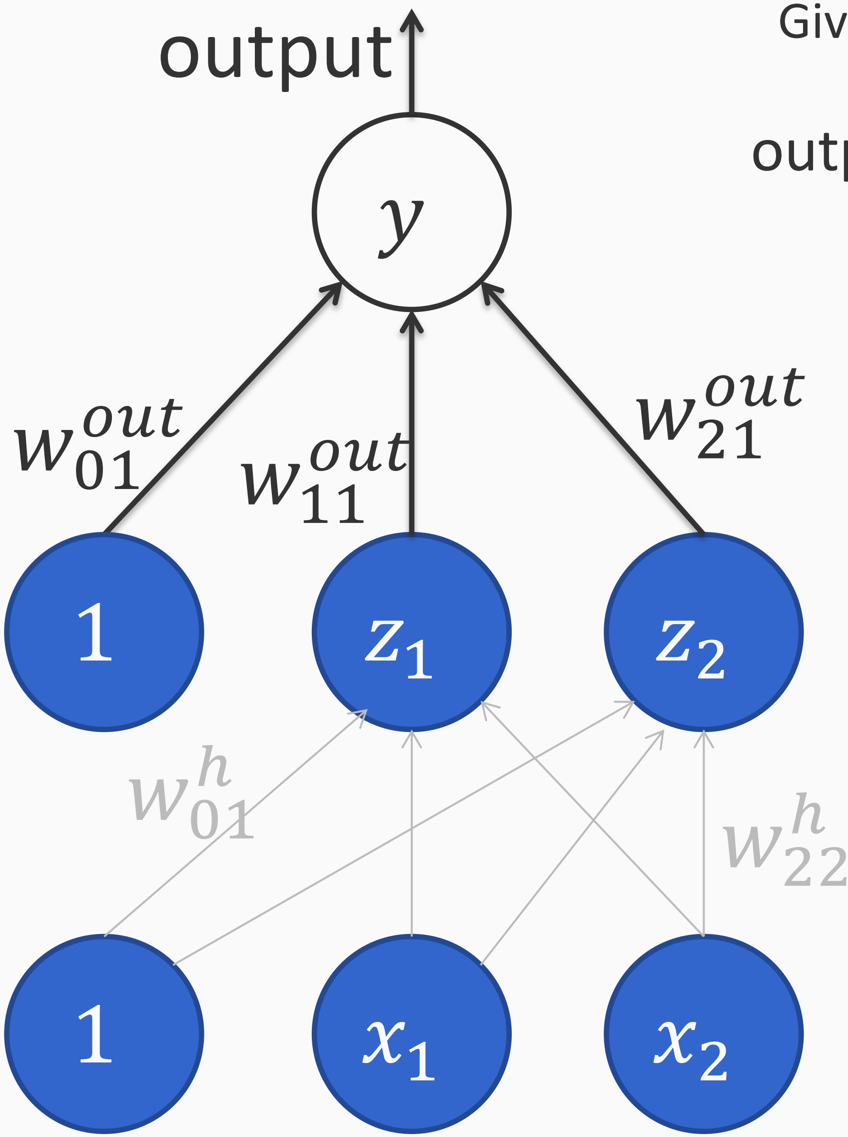


Given an input **x**, how is the output predicted

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

$$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$$

The forward pass



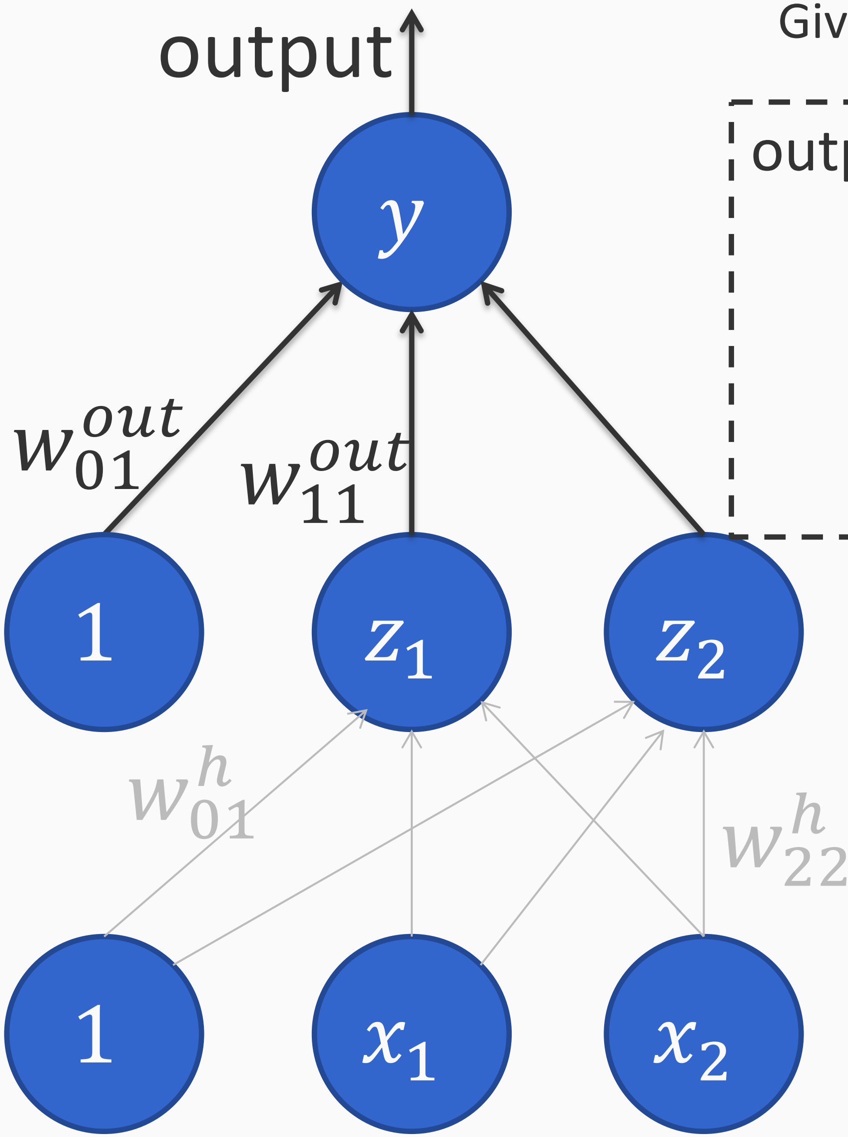
Given an input **x**, how is the output predicted

output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

The forward pass



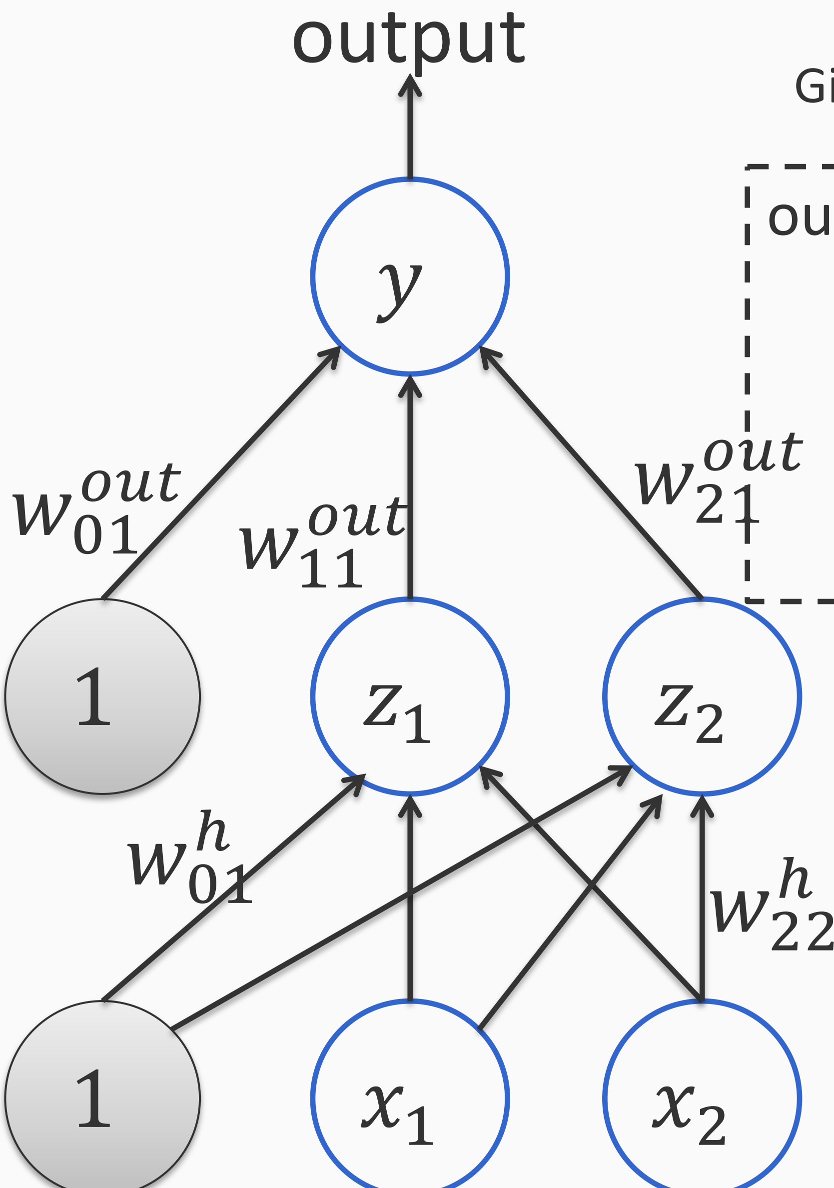
Given an input **x**, how is the output predicted

output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

The forward pass



Given an input **x**, how is the output predicted

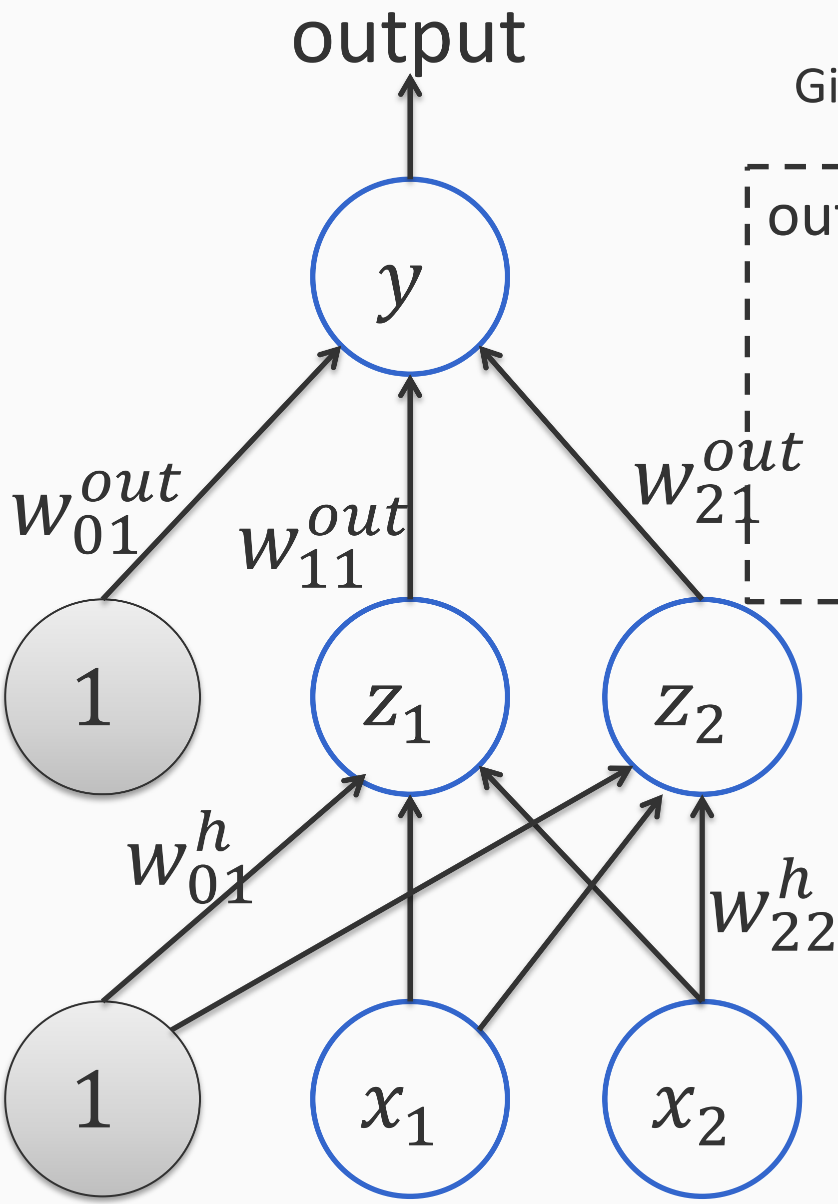
output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

In general, before visiting (i.e. computing) the value of a node, visit all nodes that serve as inputs to it.

The forward pass



Given an input **x**, how is the output predicted

output $y = w_{01}^{o} + w_{11}^{o} z_1 + w_{21}^{o} z_2$

$$z_2 = \sigma(w_{02}^{h} + w_{12}^{h} x_1 + w_{22}^{h} x_2)$$

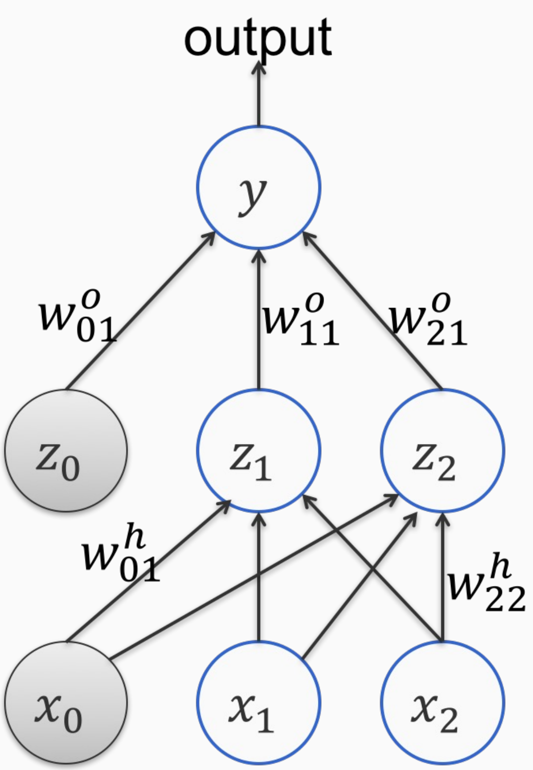
$$z_1 = \sigma(w_{01}^{h} + w_{11}^{h} x_1 + w_{21}^{h} x_2)$$

In general, before visiting (i.e. computing) the value of a node, visit all nodes that serve as inputs to it.

Questions?

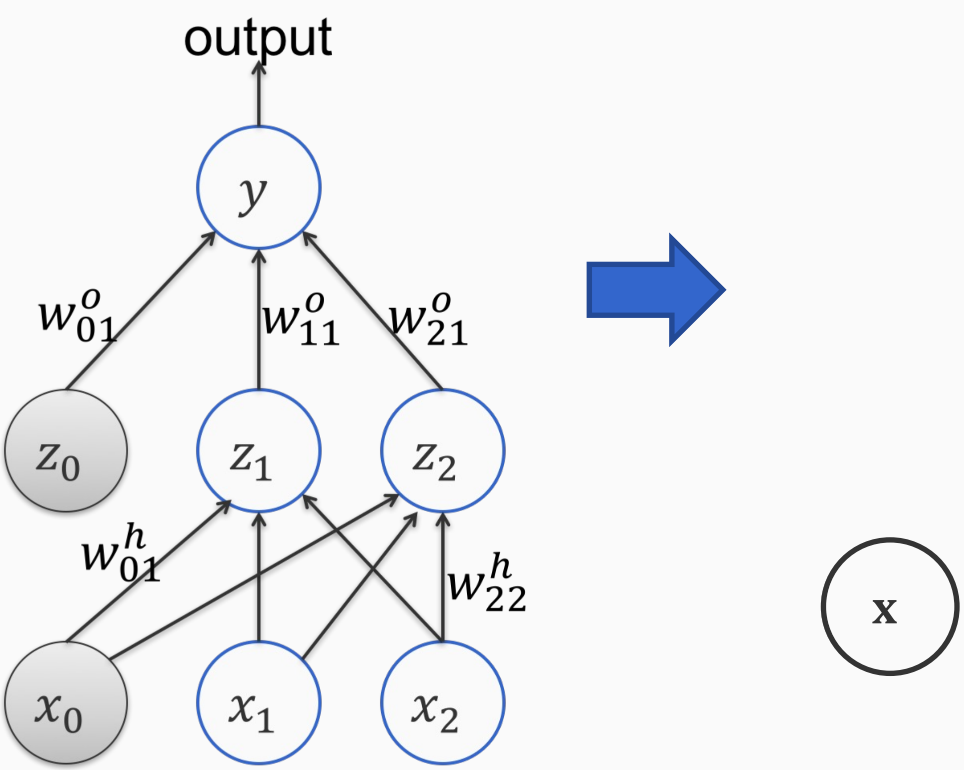
A notational convenience

Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).



A notational convenience

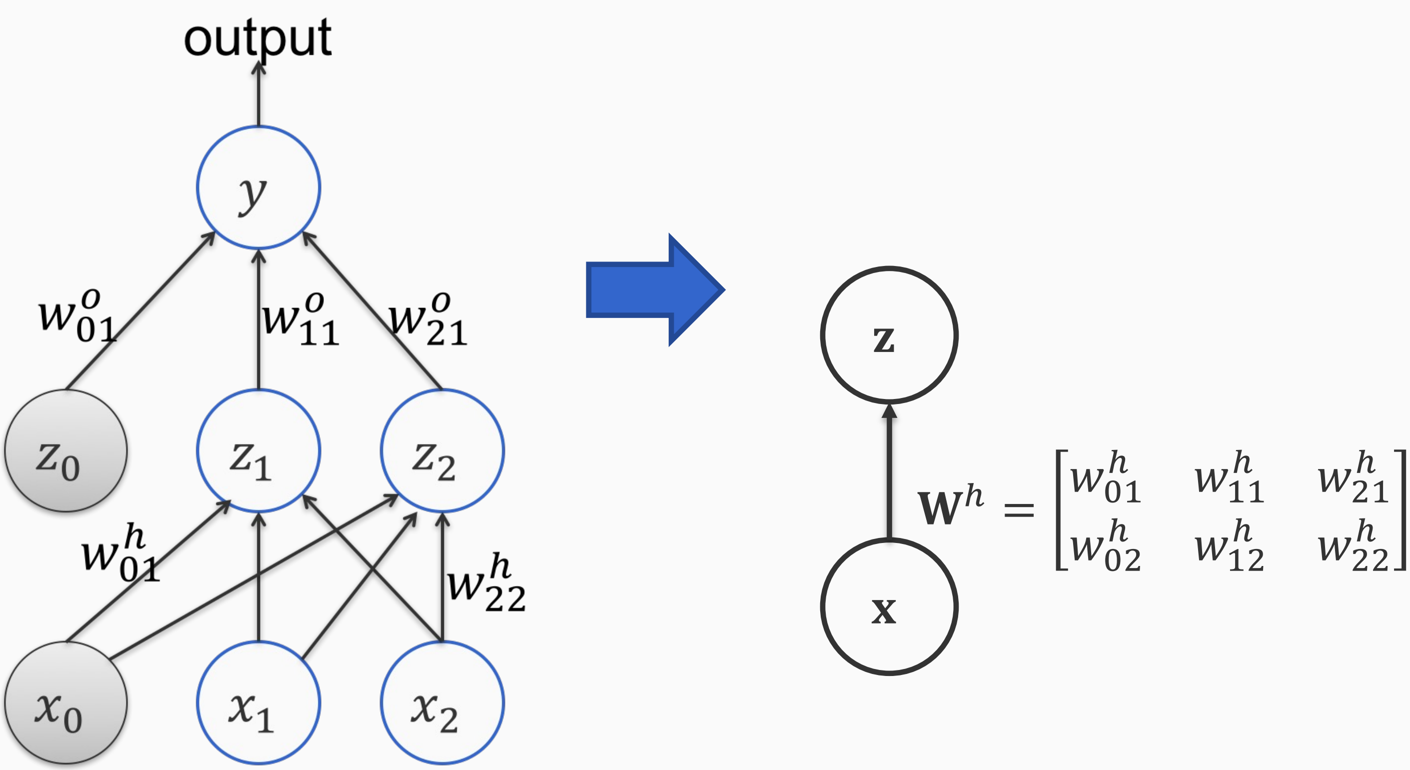
Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).



Represents $[x_0, x_1, x_2]$

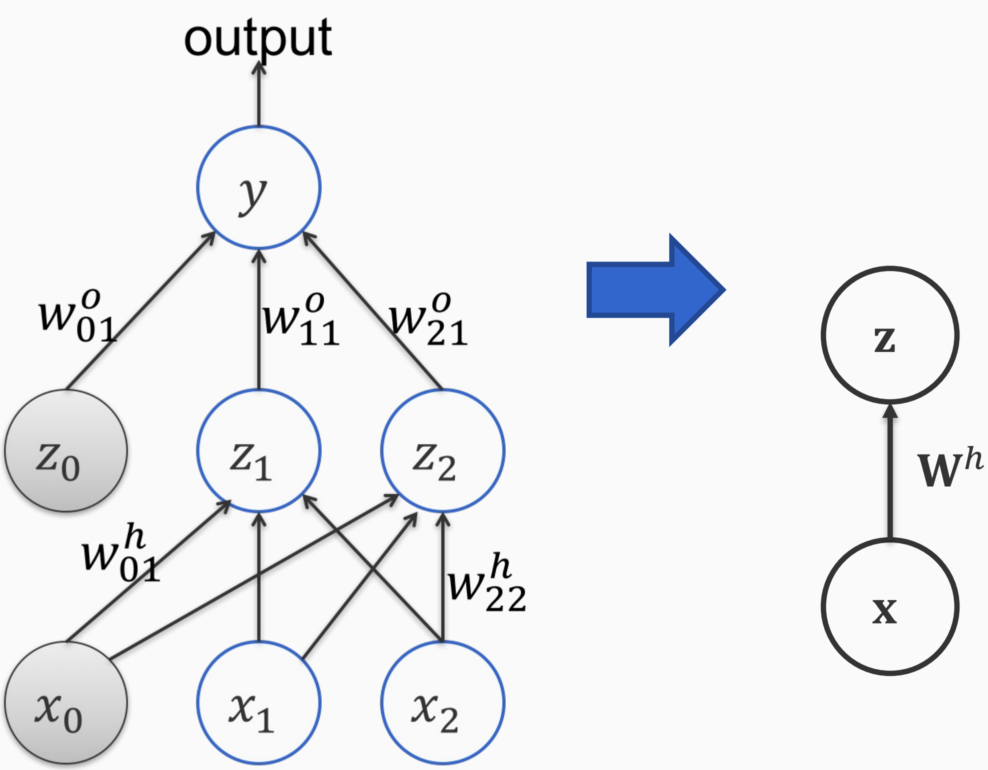
A notational convenience

Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).



A notational convenience

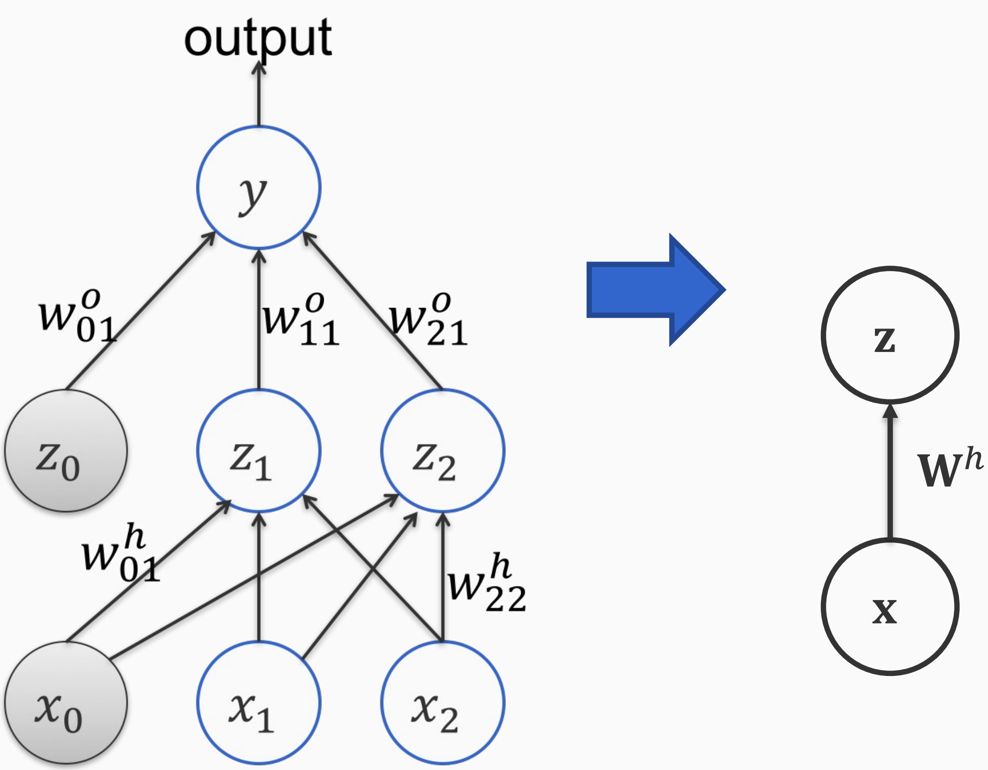
Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).



$$\mathbf{z} = \sigma(\mathbf{W}^h \mathbf{x})$$

A notational convenience

Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).

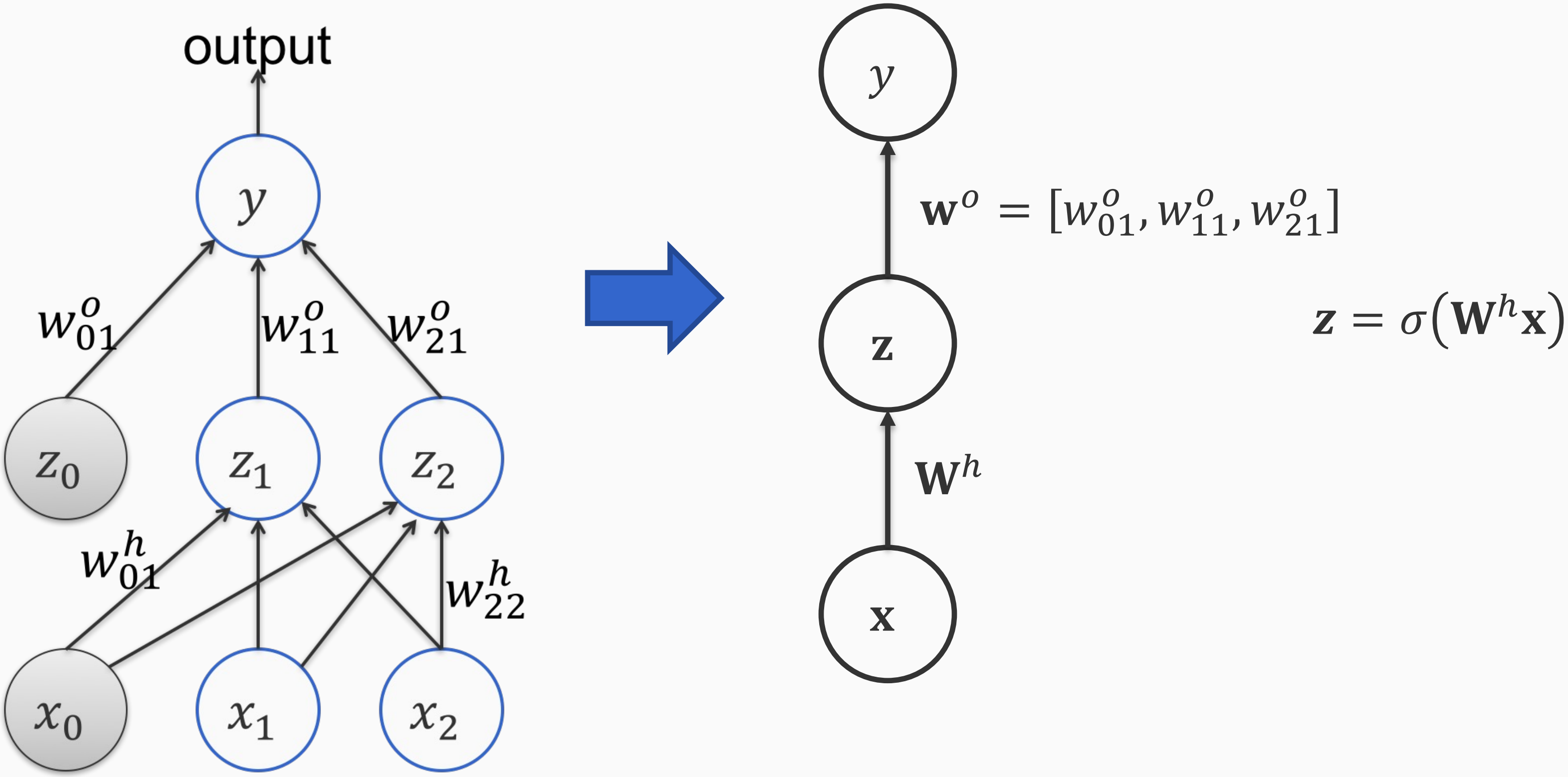


$$\mathbf{z} = \sigma(\mathbf{W}^h \mathbf{x})$$

Each element of $\mathbf{z}$ is z_i, and is generated by the sigmoid activation to each element of $\mathbf{W}^h \mathbf{x}$.

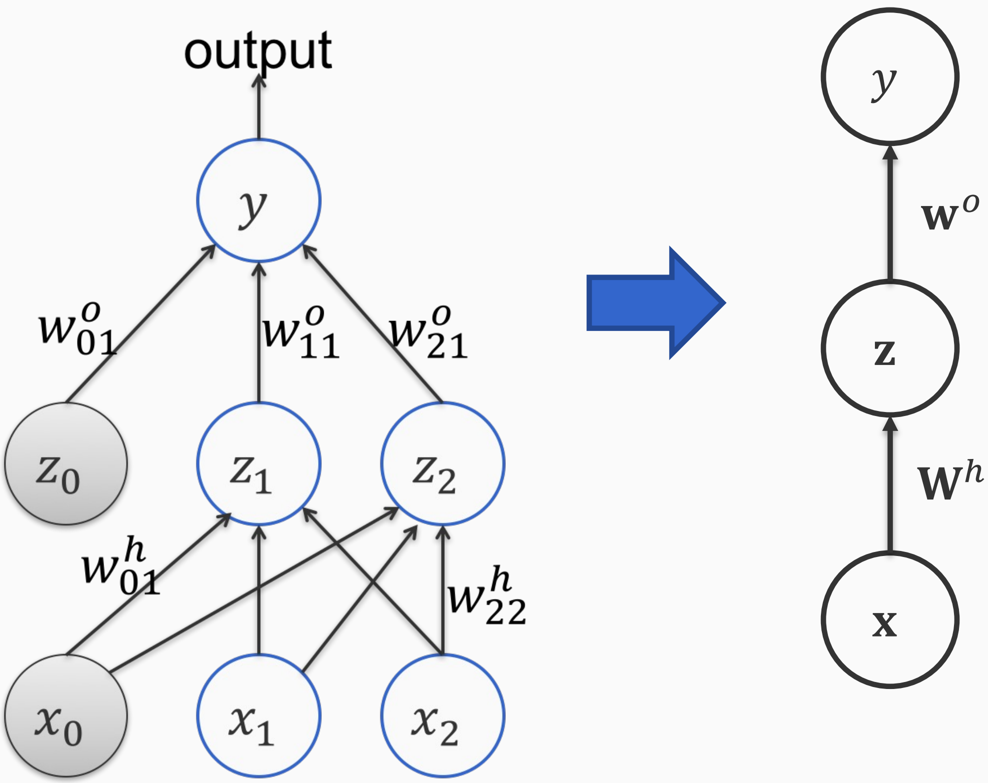
A notational convenience

Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).



A notational convenience

Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).

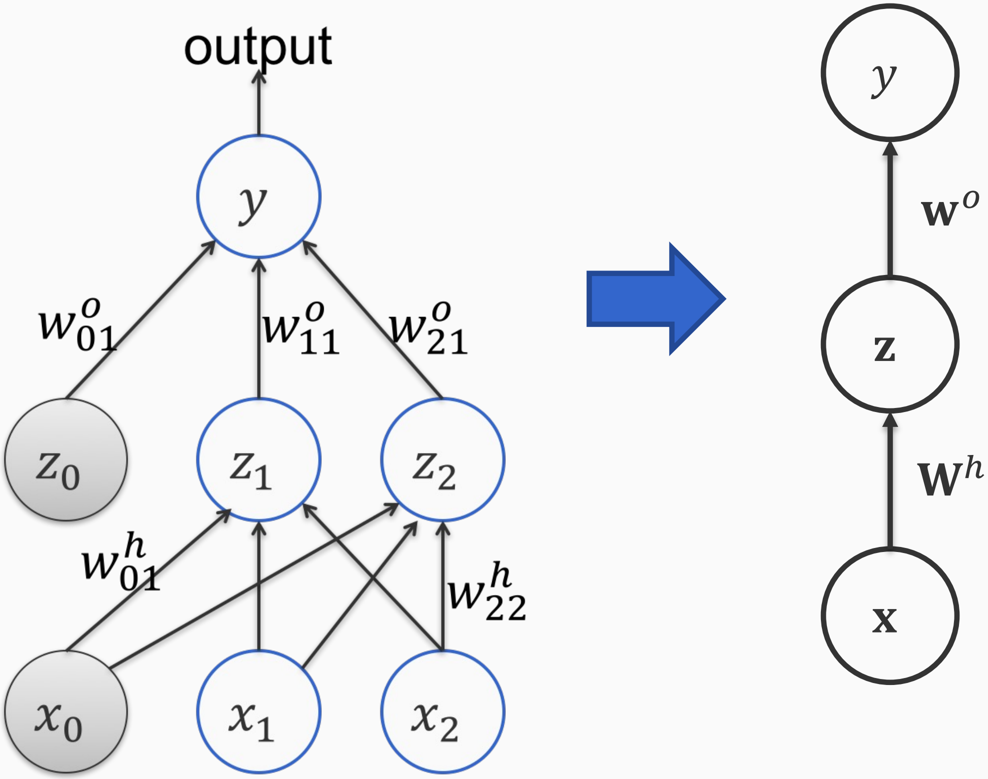


$$y = \mathbf{w}^o\mathbf{z}$$

$$\mathbf{z} = \sigma(\mathbf{W}^h\mathbf{x})$$

A notational convenience

Commonly nodes in the networks represent not only single numbers (e.g. features, outputs) but also entire *vectors* (an array of numbers), *matrices* (a 2d array of numbers) or *tensors* (an n-dimensional array of numbers).



$$y = \mathbf{w}^o \mathbf{z}$$

No activation because the output is defined to be linear

$$\mathbf{z} = \sigma(\mathbf{W}^h \mathbf{x})$$

Side note: Why tensors?

The notational convenience allows us to:

1. Build complicated neural network architectures without the cognitive load

2. Write code that operates on vectors, matrices, tensors directly

3. Design and use accelerators for matrix algebra (e.g. GPUs)