



THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound



A decorative horizontal bar at the bottom of the slide, consisting of five colored segments: light green, medium green, cyan, light blue, and purple.

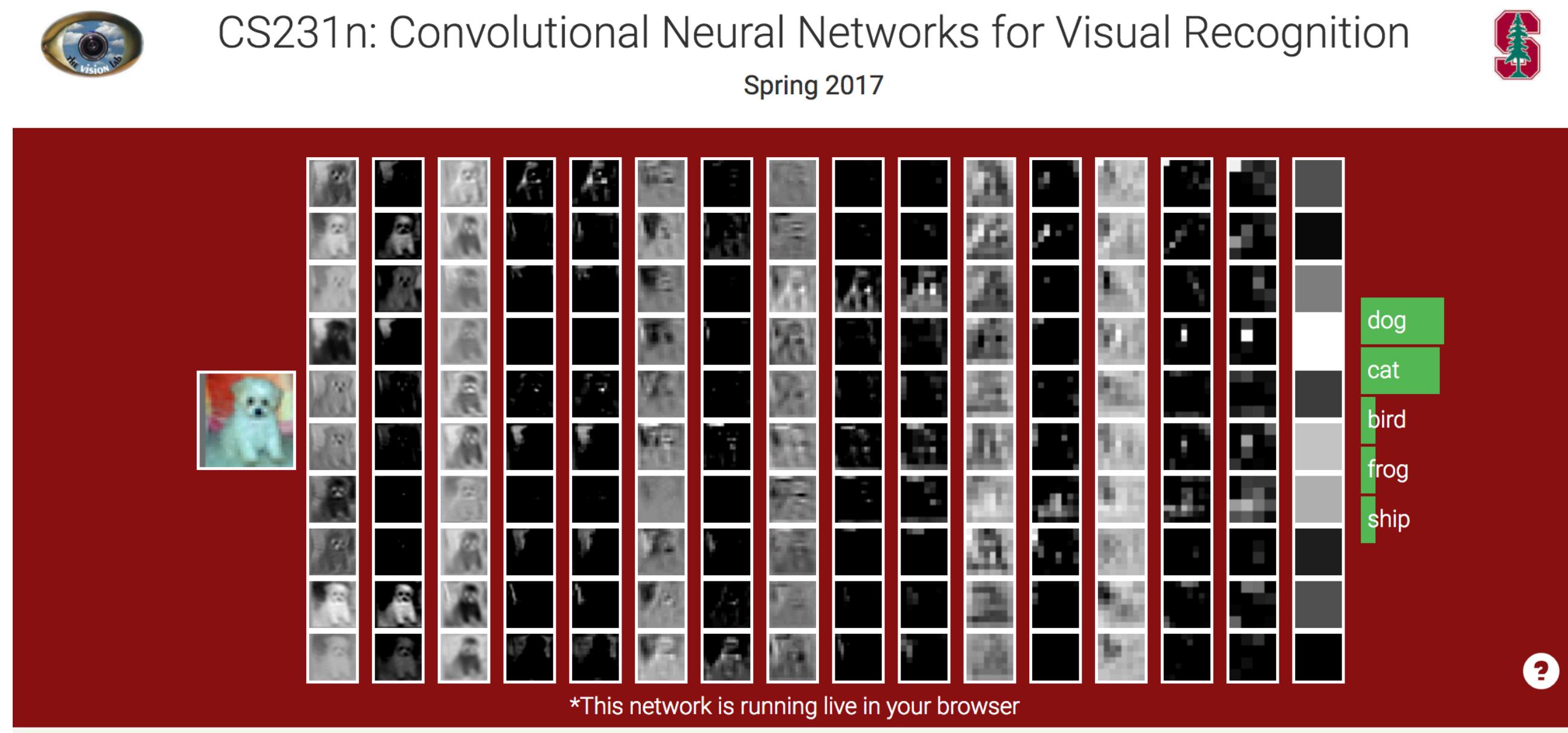
Lecture 2: Introduction to Deep Learning

Course Logistics

- Update on **course registrations** — 51 students registered
- **Piazza** — 38 students signed up
 - piazza.com/ubc.ca/winterterm22021/cpsc532s2012020w
 - Access code: **cpsc532s**
- **Assignment 0** is out (for practice only, no credit)
- **Assignment 1** is out (due date Thursday, Jan 21 @ 11:59pm)
- Mine and TA office hours will be posted by **today**

Introduction to Deep Learning

There is a **lot packed** into today's lecture (excerpts from a few lectures of CS231n)



if you want more details, check out CS231n lectures on-line

Covering: foundations and most important aspects of supervised DNNs

Not-covering: neuroscience background of deep learning, optimization (CPSC 340 & CPSC 540), and not a lot of theoretical underpinning



Linear regression (review)

		Inputs (features)					Outputs	
Training Set		production costs	promotional costs	genre of the movie	box office first week	total book sales	total revenue USA	total revenue international
  	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$	
	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$	
	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$	

Linear regression (review)

		Inputs (features)					Outputs	
		production costs	promotional costs	genre of the movie	box office first week	total book sales	total revenue USA	total revenue international
Training Set	Movie Poster	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
		$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
		$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$
Testing Set	Movie Poster	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$		
		$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$x_5^{(5)}$		

Linear regression (review)

		Inputs (features)					Outputs	
		production costs	promotional costs	genre of the movie	box office first week	total book sales	total revenue USA	total revenue international
Training Set	Movie Poster	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
		$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
		$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$
Testing Set	Movie Poster	$x_1^{(4)}$	$x_2^{(4)}$	$x_3^{(4)}$	$x_4^{(4)}$	$x_5^{(4)}$	$\hat{y}_j = \sum_i w_{ji}x_i + b_j$	
		$x_1^{(5)}$	$x_2^{(5)}$	$x_3^{(5)}$	$x_4^{(5)}$	$x_5^{(5)}$		

Linear regression (review)

$$\hat{y}_j = \sum_i w_{ji}x_i + b_j$$

each output is a linear combination of inputs plus bias, easier to write in **matrix form**:

$$\hat{\mathbf{y}} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$

Linear regression (review)

$$\hat{y}_j = \sum_i w_{ji}x_i + b_j$$

each output is a linear combination of inputs plus bias, easier to write in **matrix form**:

$$\hat{\mathbf{y}} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$

Key to accurate prediction is
learning parameters to minimize
discrepancy with historical data

$$D_{train} = \{(\mathbf{x}^{(d)}, \mathbf{y}^{(d)})\}$$

Inputs (features)					Outputs		
Training Set	production costs	promotional costs	genre of the movie	box office first week	total book sales	total revenue USA	total revenue international
	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	$x_4^{(1)}$	$x_5^{(1)}$	$y_1^{(1)}$	$y_2^{(1)}$
	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	$x_4^{(2)}$	$x_5^{(2)}$	$y_1^{(2)}$	$y_2^{(2)}$
	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	$x_4^{(3)}$	$x_5^{(3)}$	$y_1^{(3)}$	$y_2^{(3)}$

*slide adopted from V. Ordonex

Linear regression (review)

$$\hat{y}_j = \sum_i w_{ji}x_i + b_j$$

each output is a linear combination of inputs plus bias, easier to write in **matrix form**:

$$\hat{\mathbf{y}} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$

Key to accurate prediction is
learning parameters to minimize
discrepancy with historical data

$$D_{train} = \{(\mathbf{x}^{(d)}, \mathbf{y}^{(d)})\}$$

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \frac{|D_{train}|}{d=1} l(\hat{\mathbf{y}}^{(d)}, \mathbf{y}^{(d)})$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Linear regression (review)

$$\hat{y}_j = \sum_i w_{ji}x_i + b_j$$

each output is a linear combination of inputs plus bias, easier to write in **matrix form**:

$$\hat{\mathbf{y}} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$

Key to accurate prediction is
learning parameters to minimize
discrepancy with historical data

$$D_{train} = \{(\mathbf{x}^{(d)}, \mathbf{y}^{(d)})\}$$

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \|\hat{\mathbf{y}}^{(d)} - \mathbf{y}^{(d)}\|^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Linear regression (review) – Learning /w Least Squares

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Solution:

Linear regression (review) – Learning /w Least Squares

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Solution:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

Linear regression (review) – Learning /w Least Squares

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Solution:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2 = 0$$

Linear regression (review) – Learning /w Least Squares

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Solution:

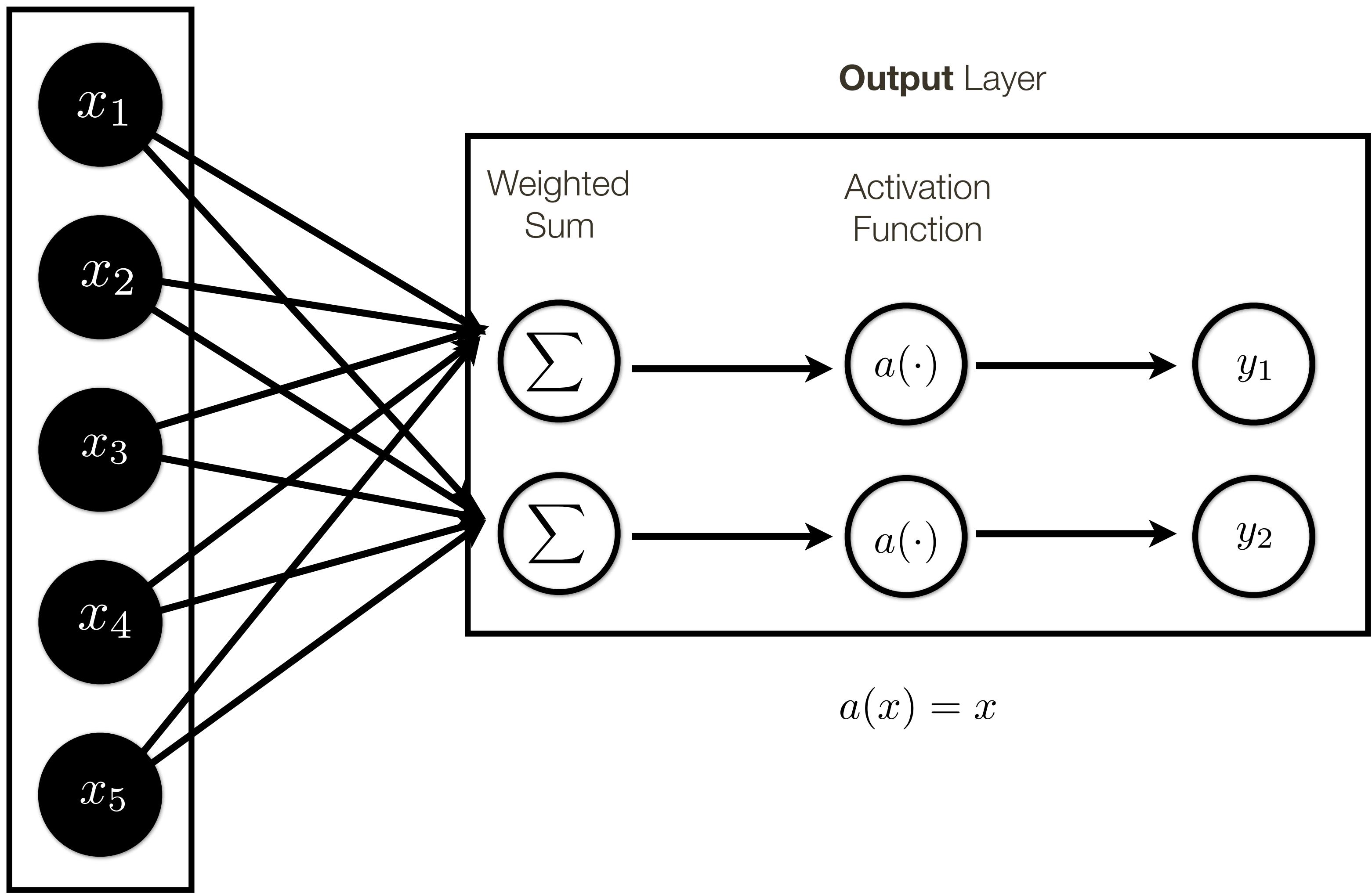
$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2$$

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left\| \mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} - \mathbf{y}^{(d)} \right\|^2 = 0$$

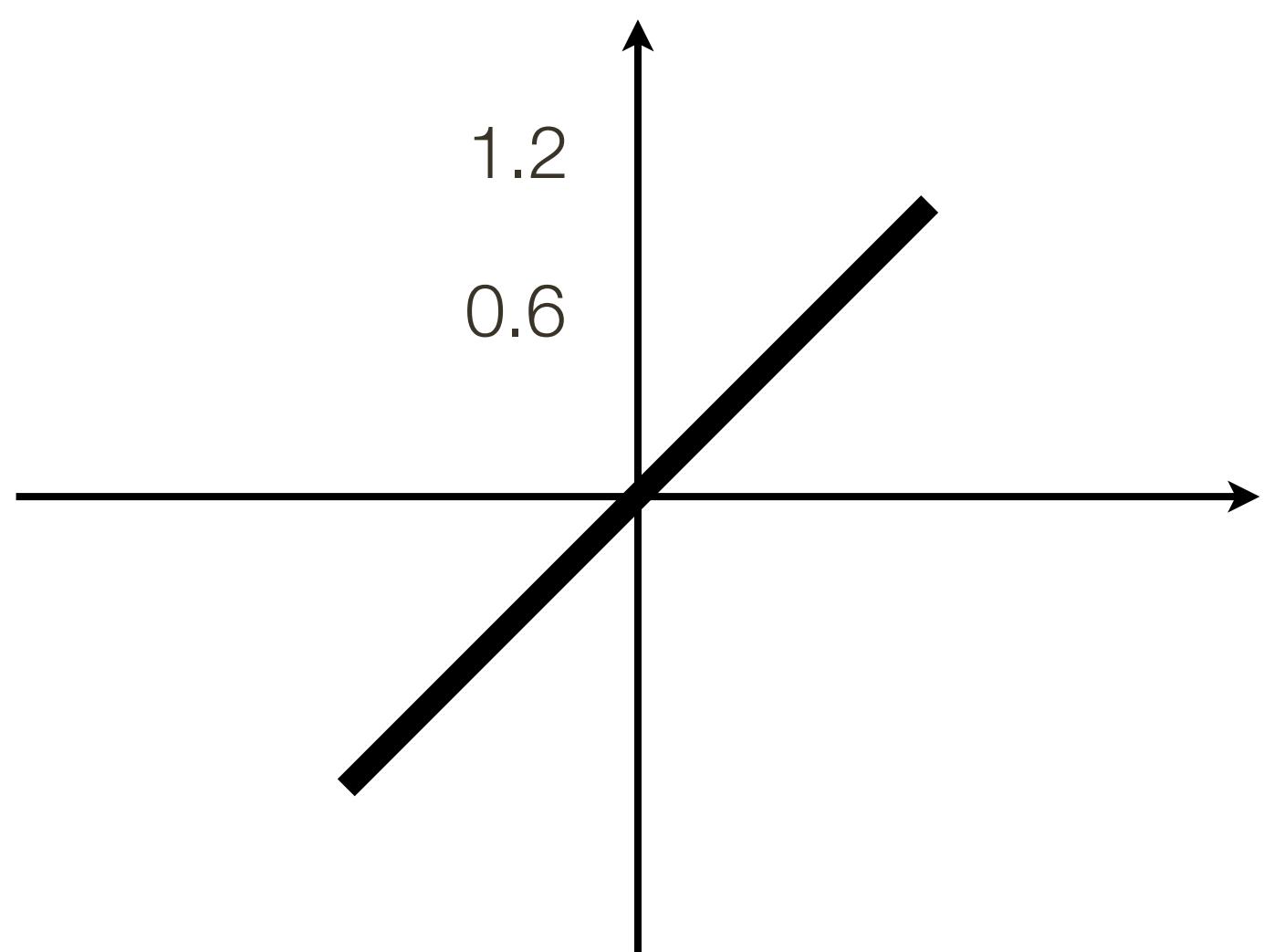
after some operations $\longrightarrow \mathbf{W}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$

One-layer Neural Network

Input Layer



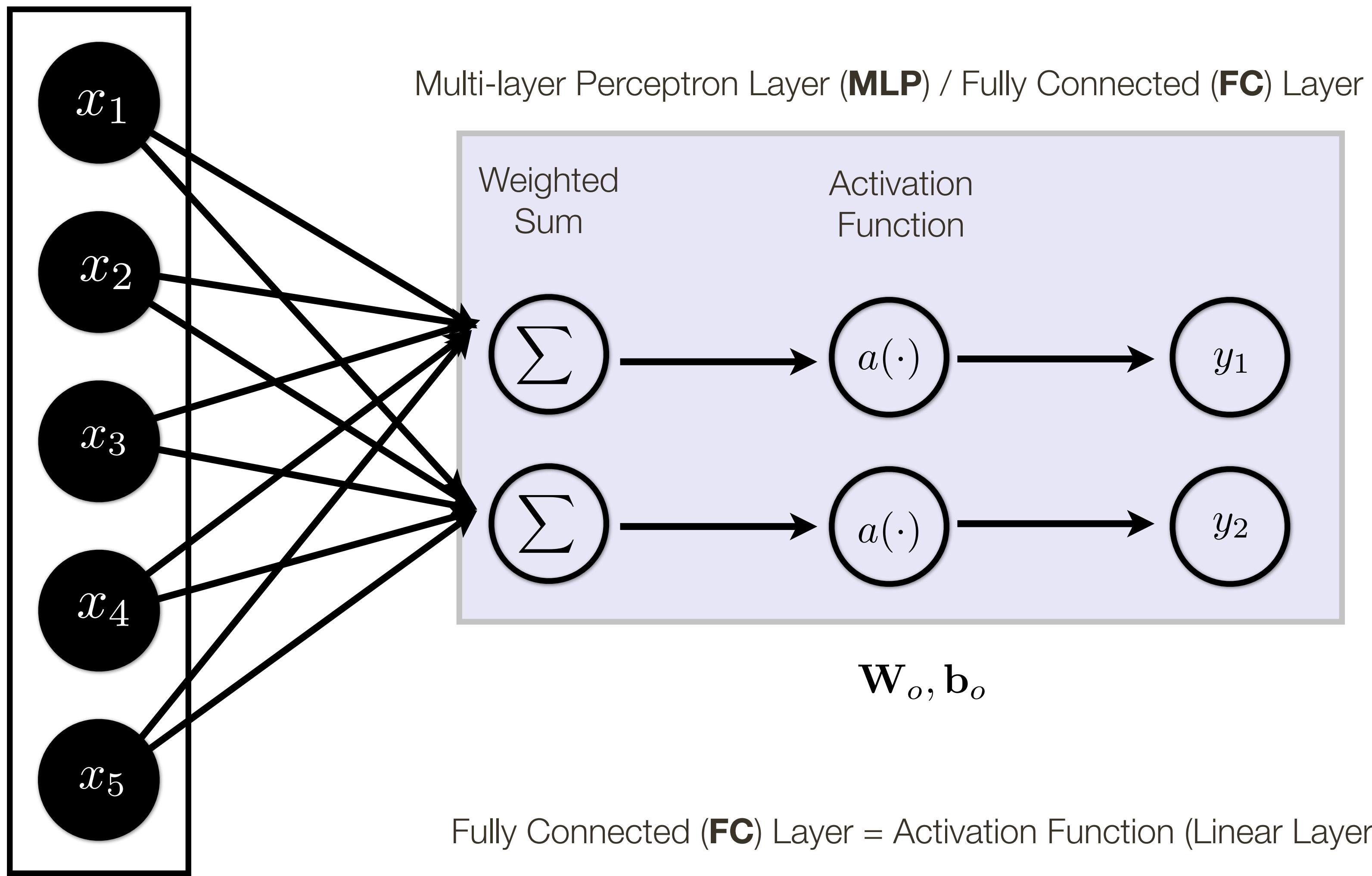
$$a(x) = x$$



Linear Activation

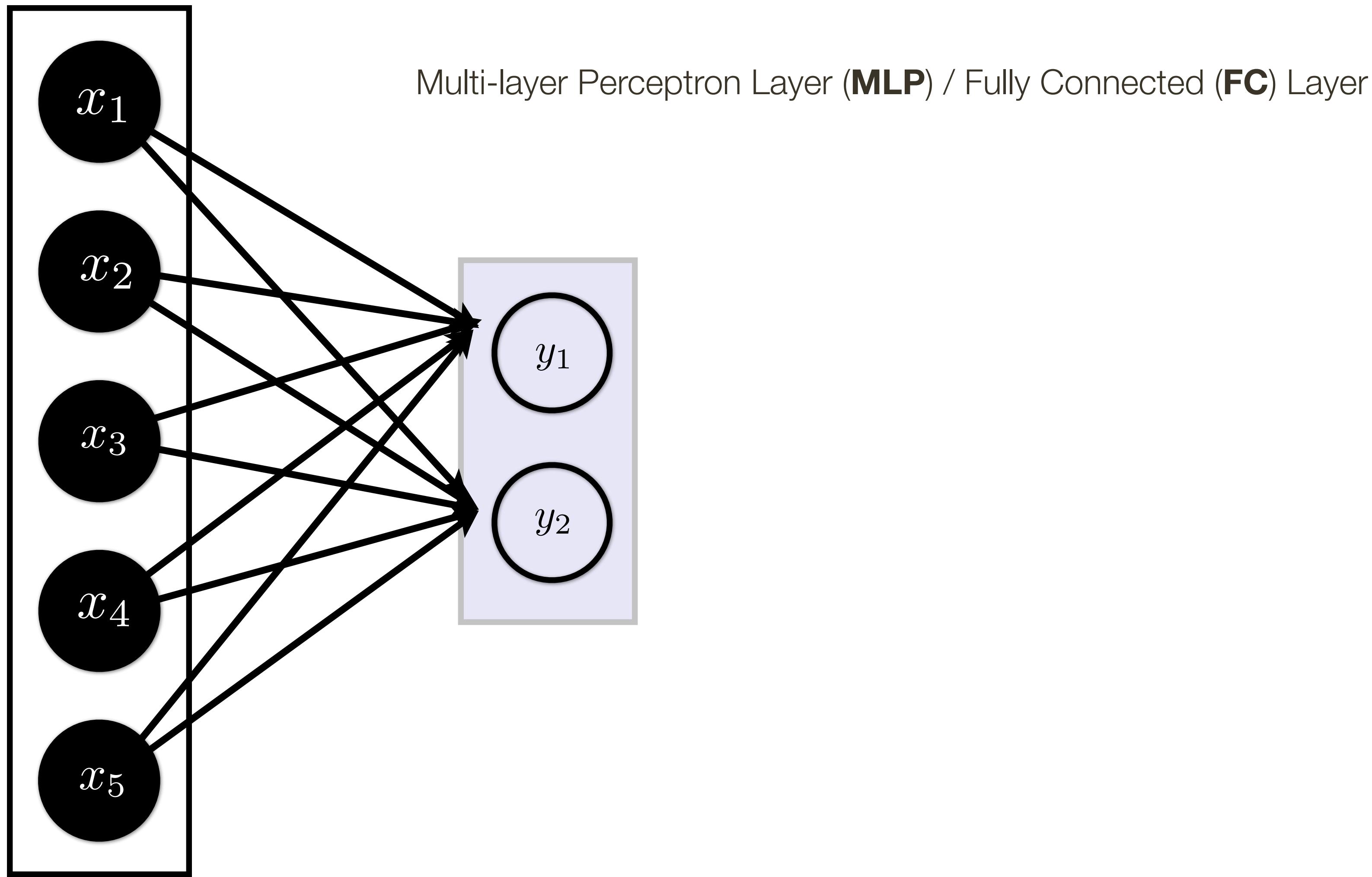
One-layer Neural Network

Input Layer

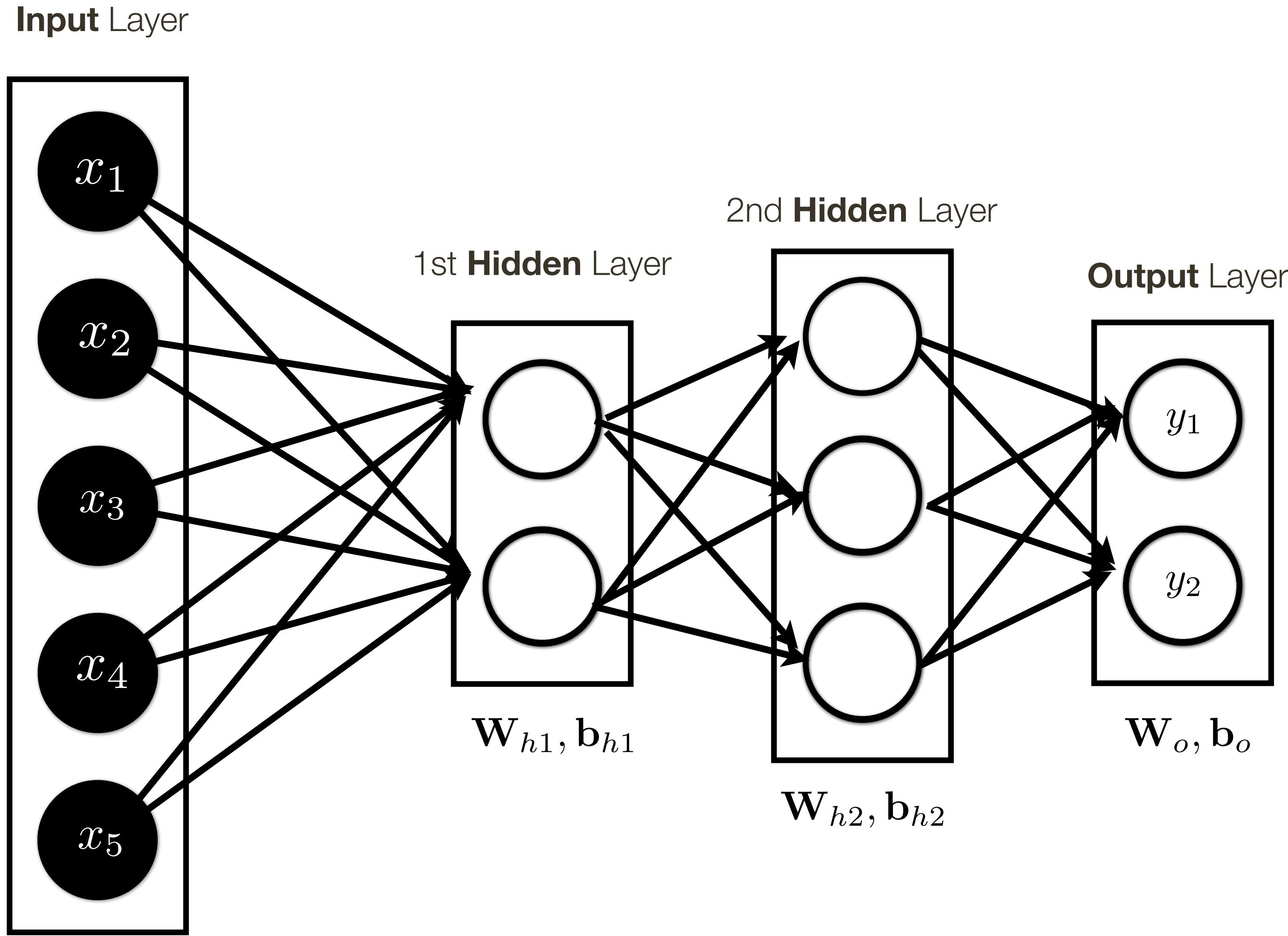


One-layer Neural Network

Input Layer



Multi-layer Neural Network



Neural Network Intuition

Question: What is a Neural Network?

Answer: Complex mapping from an input (vector) to an output (vector)

Neural Network Intuition

Question: What is a Neural Network?

Answer: Complex mapping from an input (vector) to an output (vector)

Question: What class of functions should be considered for this mapping?

Answer: Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next ...

Neural Network Intuition

Question: What is a Neural Network?

Answer: Complex mapping from an input (vector) to an output (vector)

Question: What class of functions should be considered for this mapping?

Answer: Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next ...

Question: What does a hidden unit do?

Answer: It can be thought of as classifier or a feature.

Neural Network Intuition

Question: What is a Neural Network?

Answer: Complex mapping from an input (vector) to an output (vector)

Question: What class of functions should be considered for this mapping?

Answer: Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next ...

Question: What does a hidden unit do?

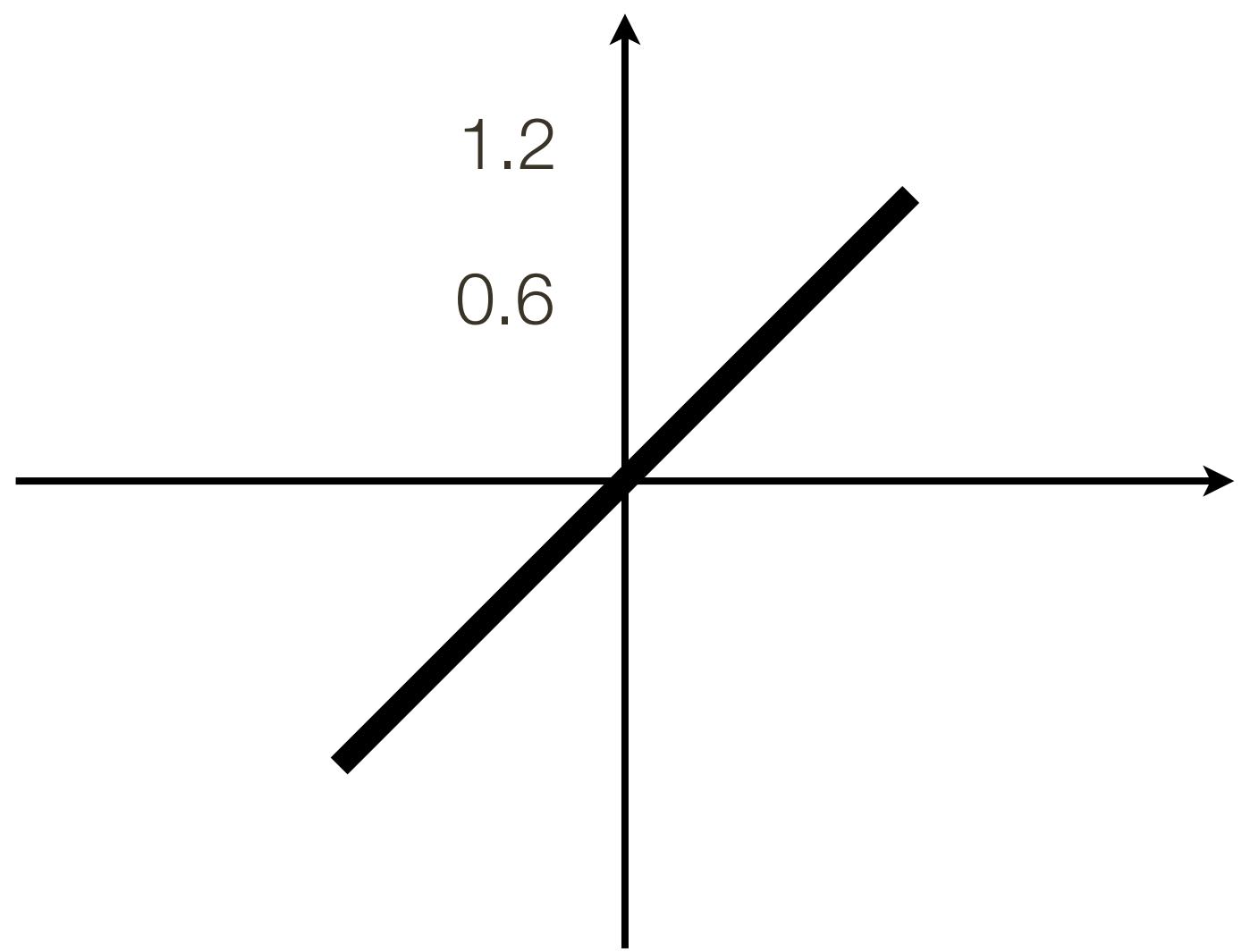
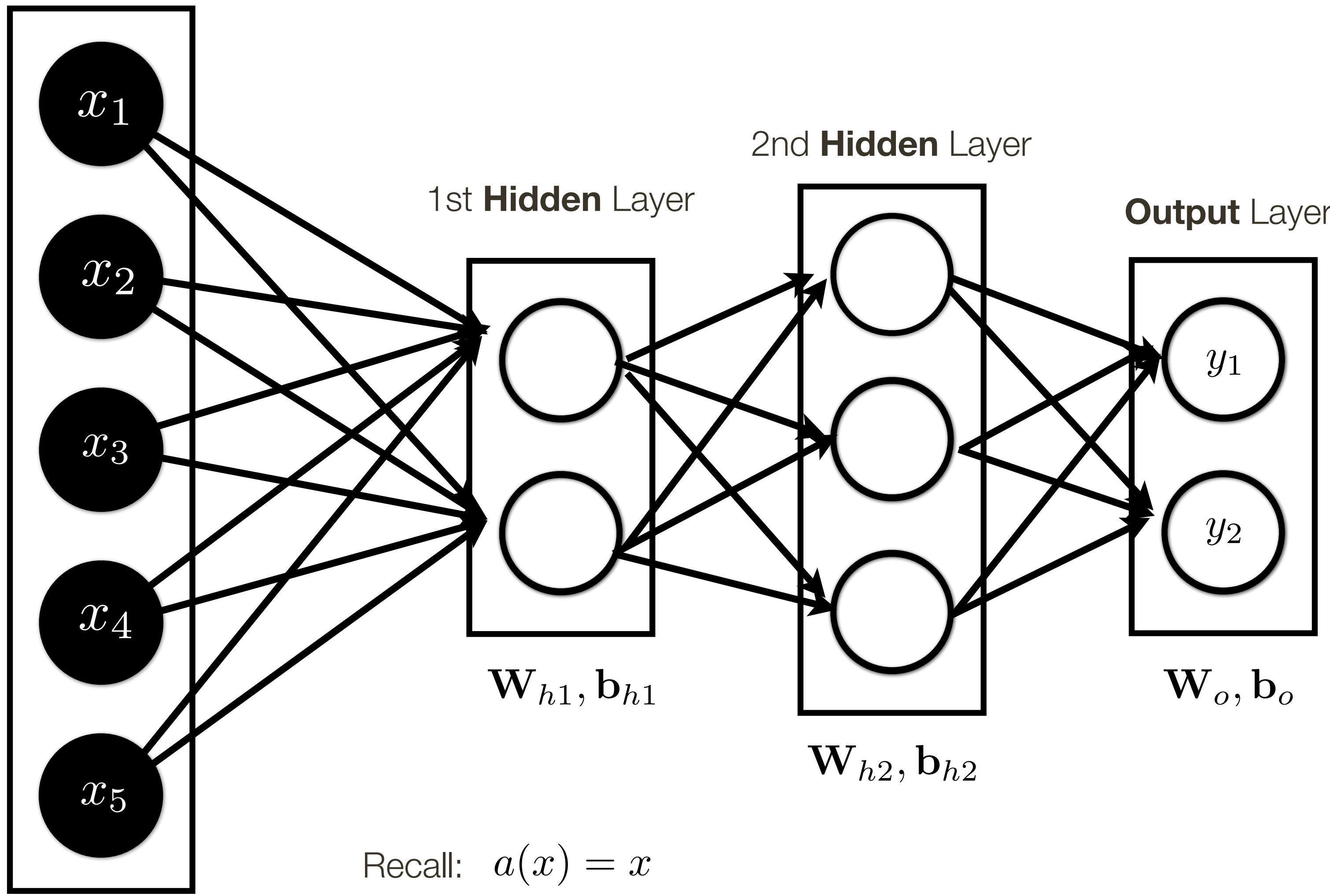
Answer: It can be thought of as classifier or a feature.

Question: Why have many layers?

Answer: 1) More layers = more complex functional mapping
2) More efficient due to distributed representation

Multi-layer Neural Network

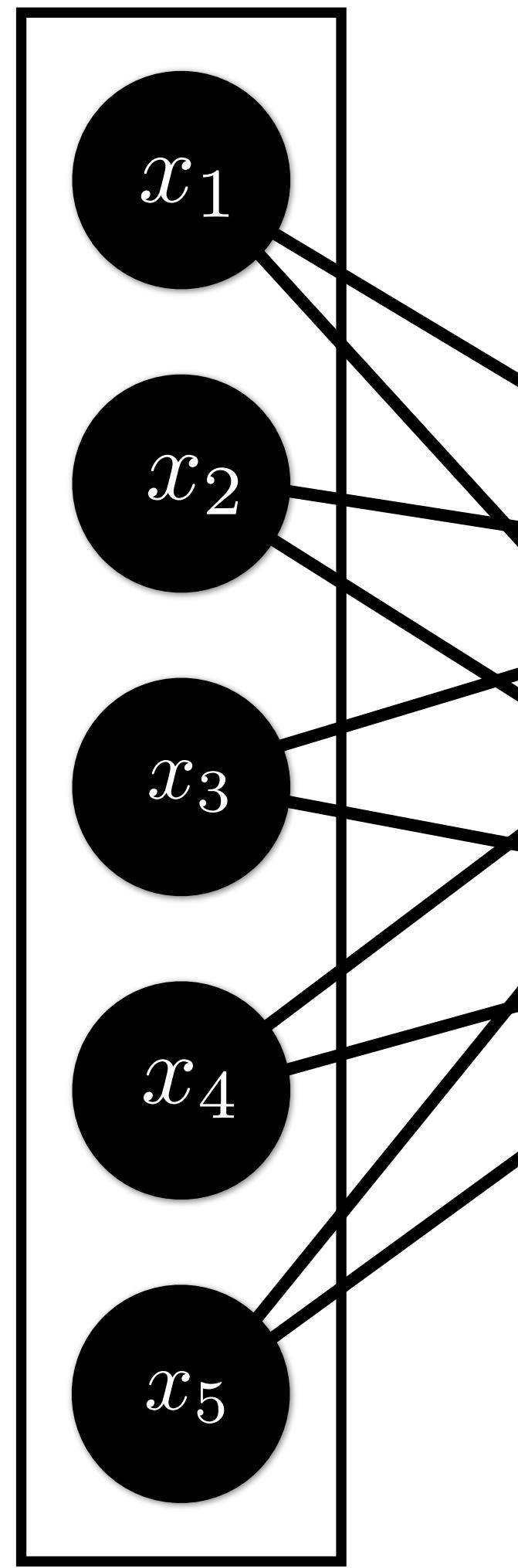
Input Layer



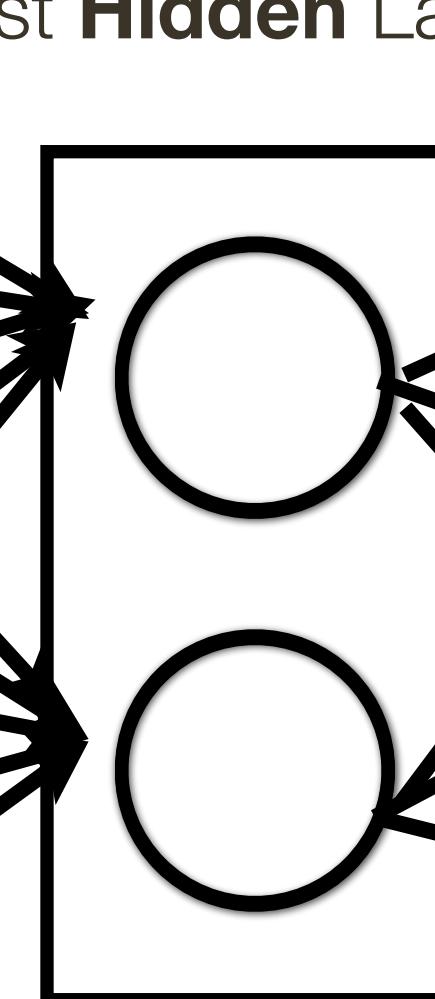
Linear Activation

Multi-layer Neural Network

Input Layer

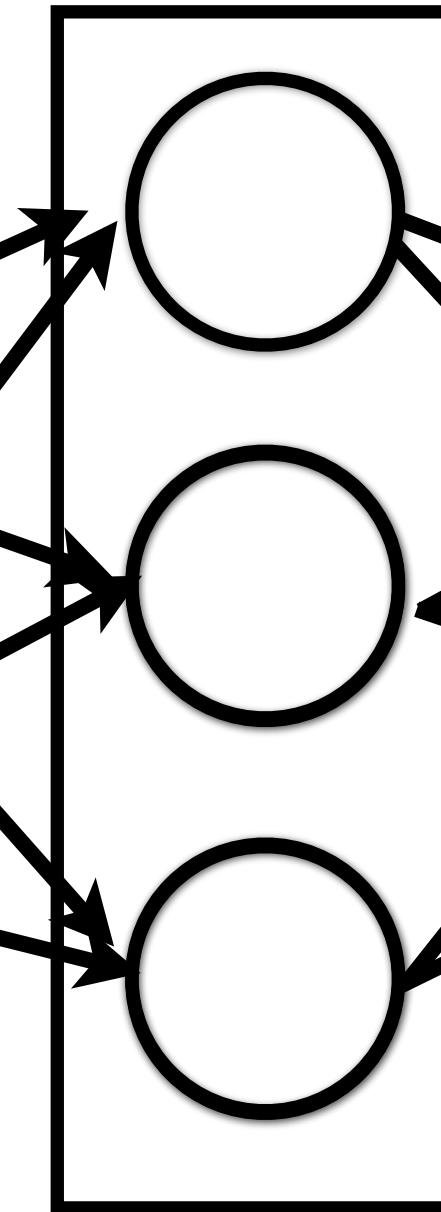


1st Hidden Layer



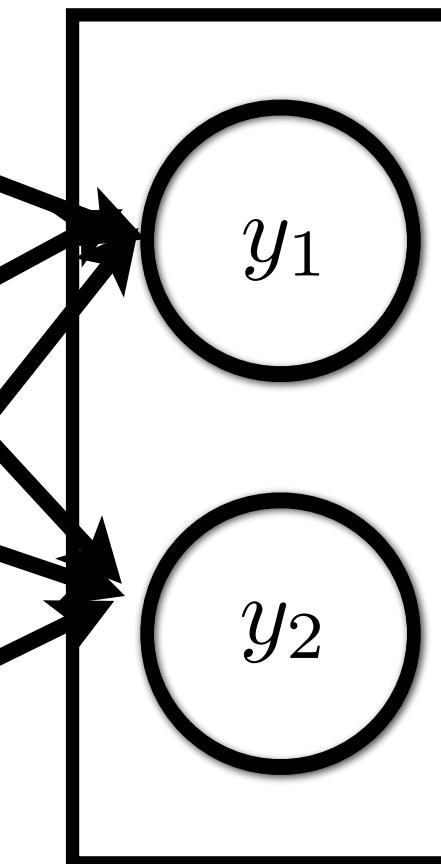
$\mathbf{W}_{h1}, \mathbf{b}_{h1}$

2nd Hidden Layer



$\mathbf{W}_{h2}, \mathbf{b}_{h2}$

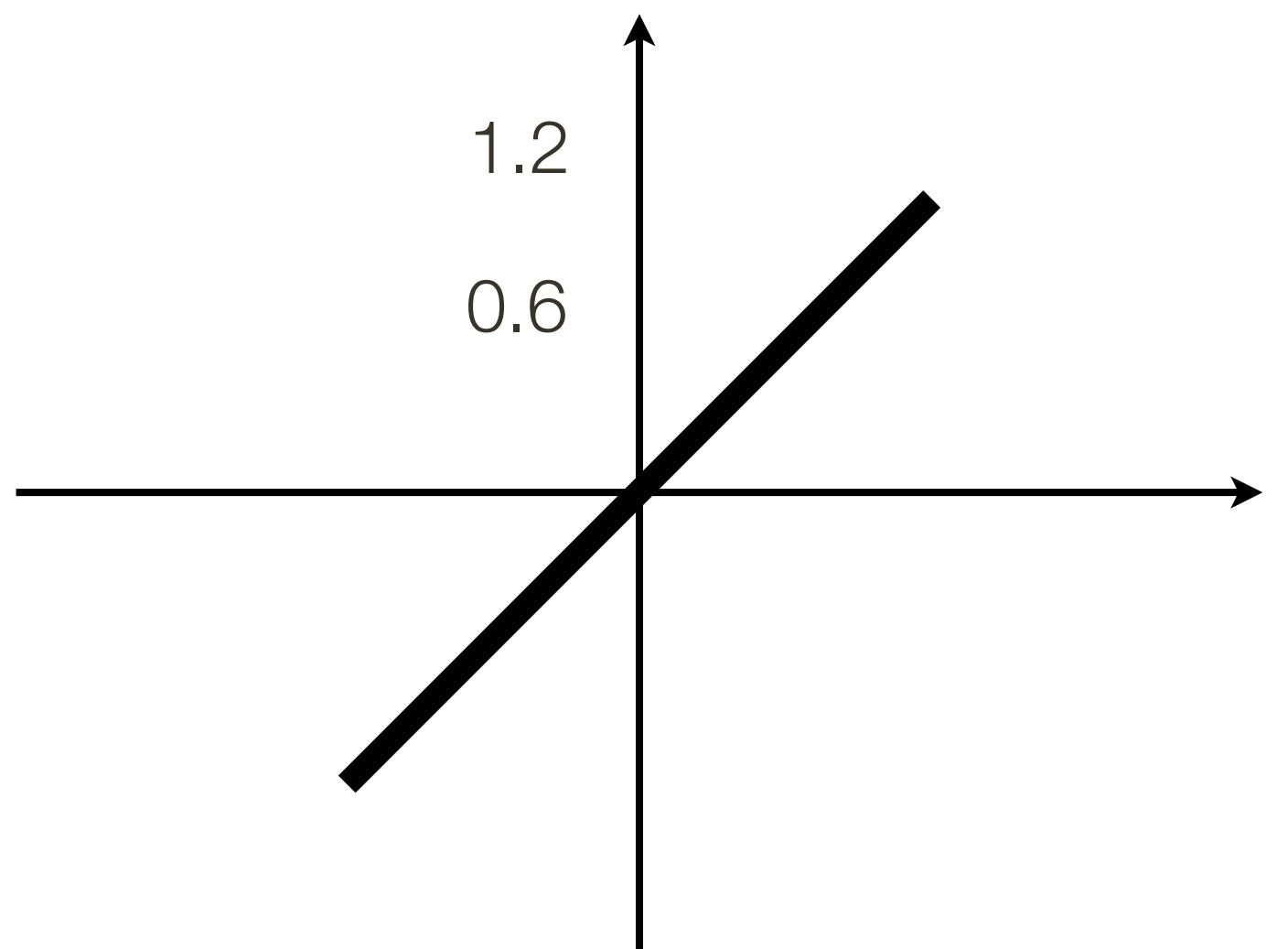
Output Layer



$\mathbf{W}_o, \mathbf{b}_o$

Why?

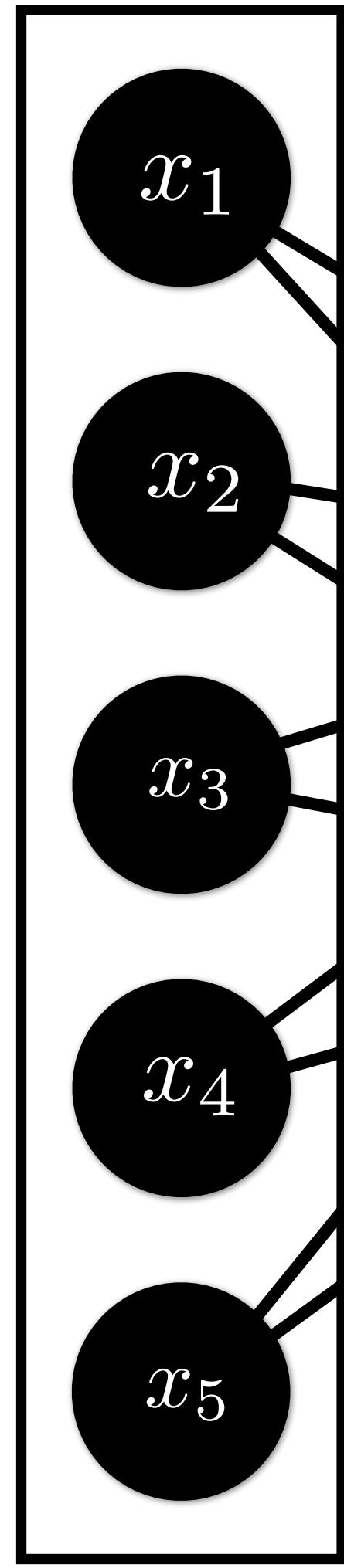
Recall: $a(x) = x$



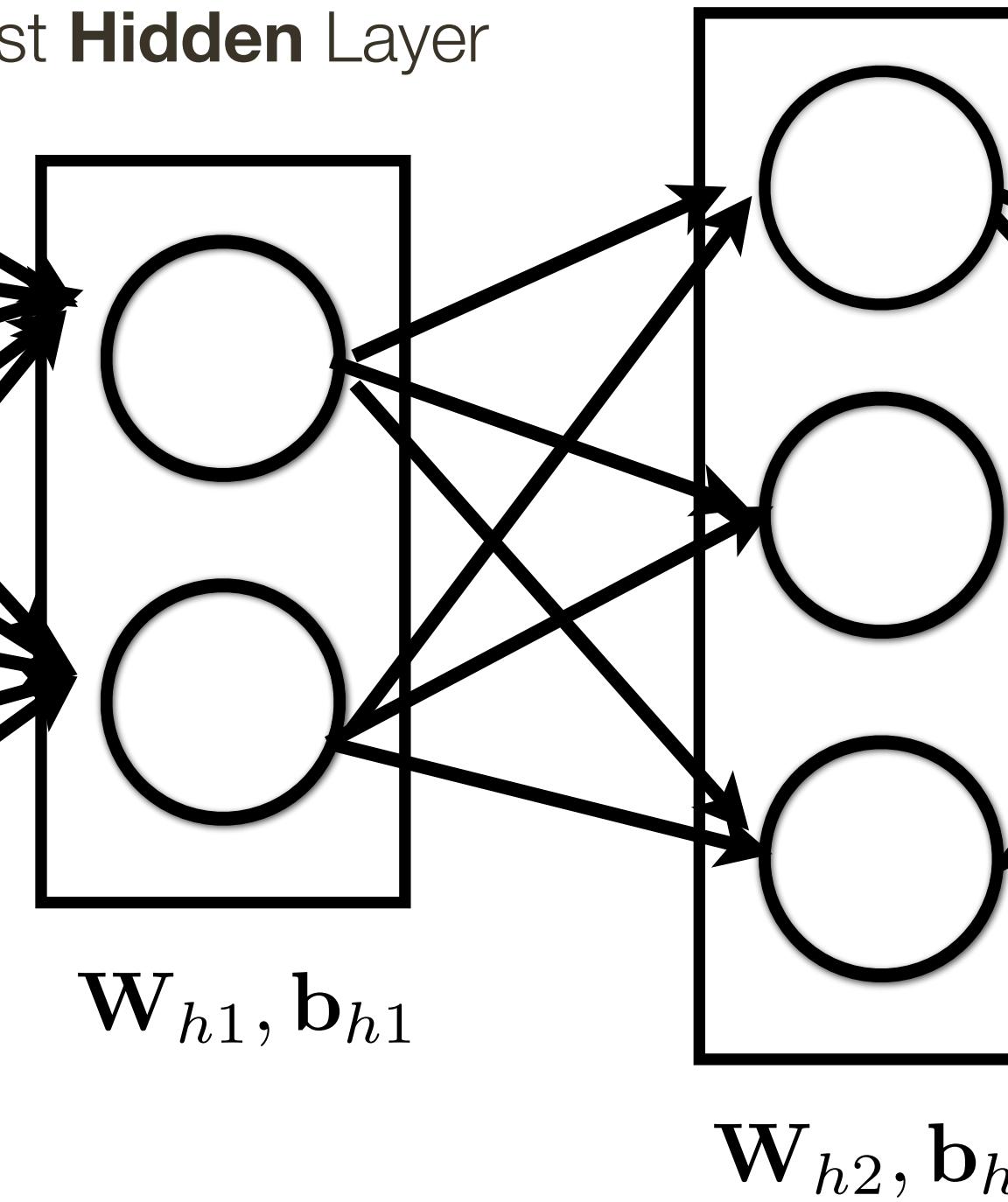
Linear Activation

Multi-layer Neural Network

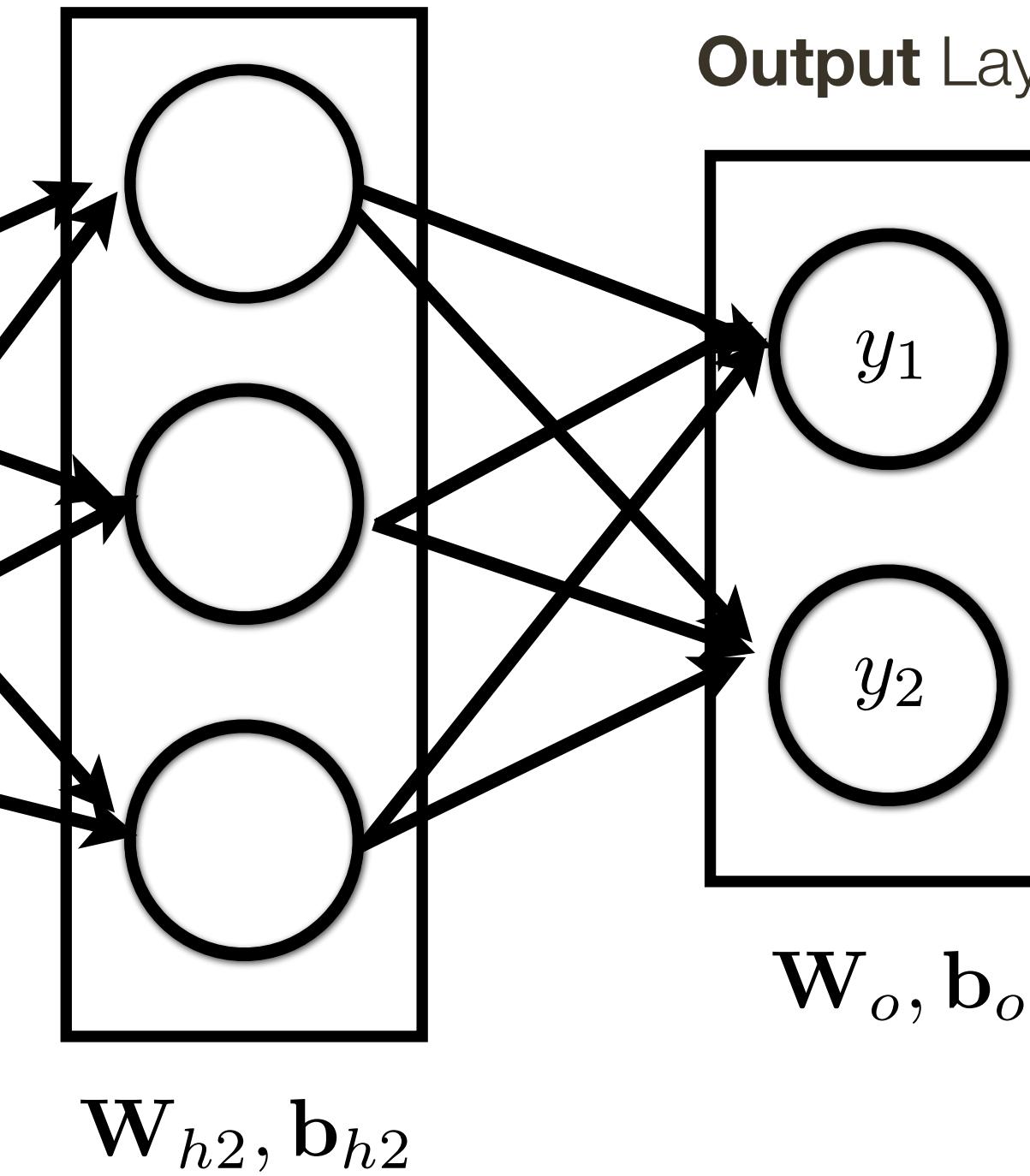
Input Layer



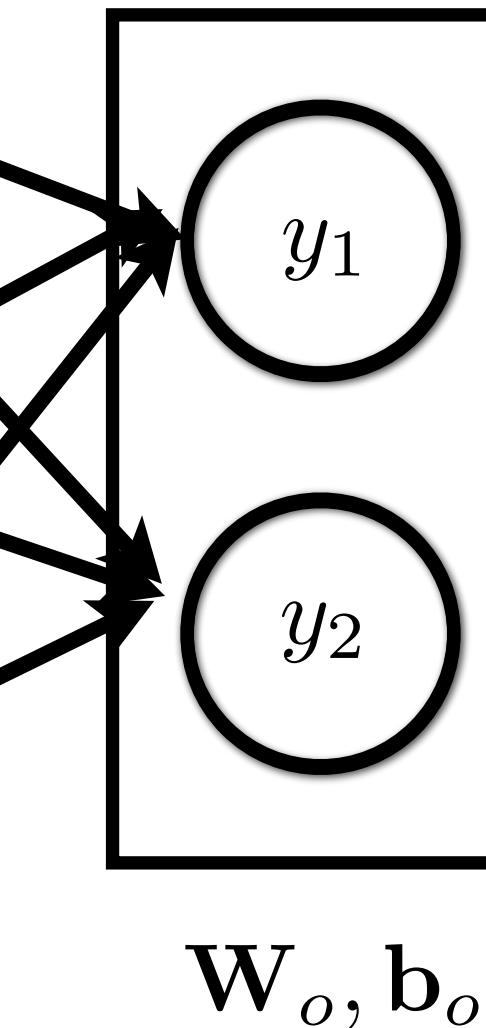
1st Hidden Layer



2nd Hidden Layer



Output Layer



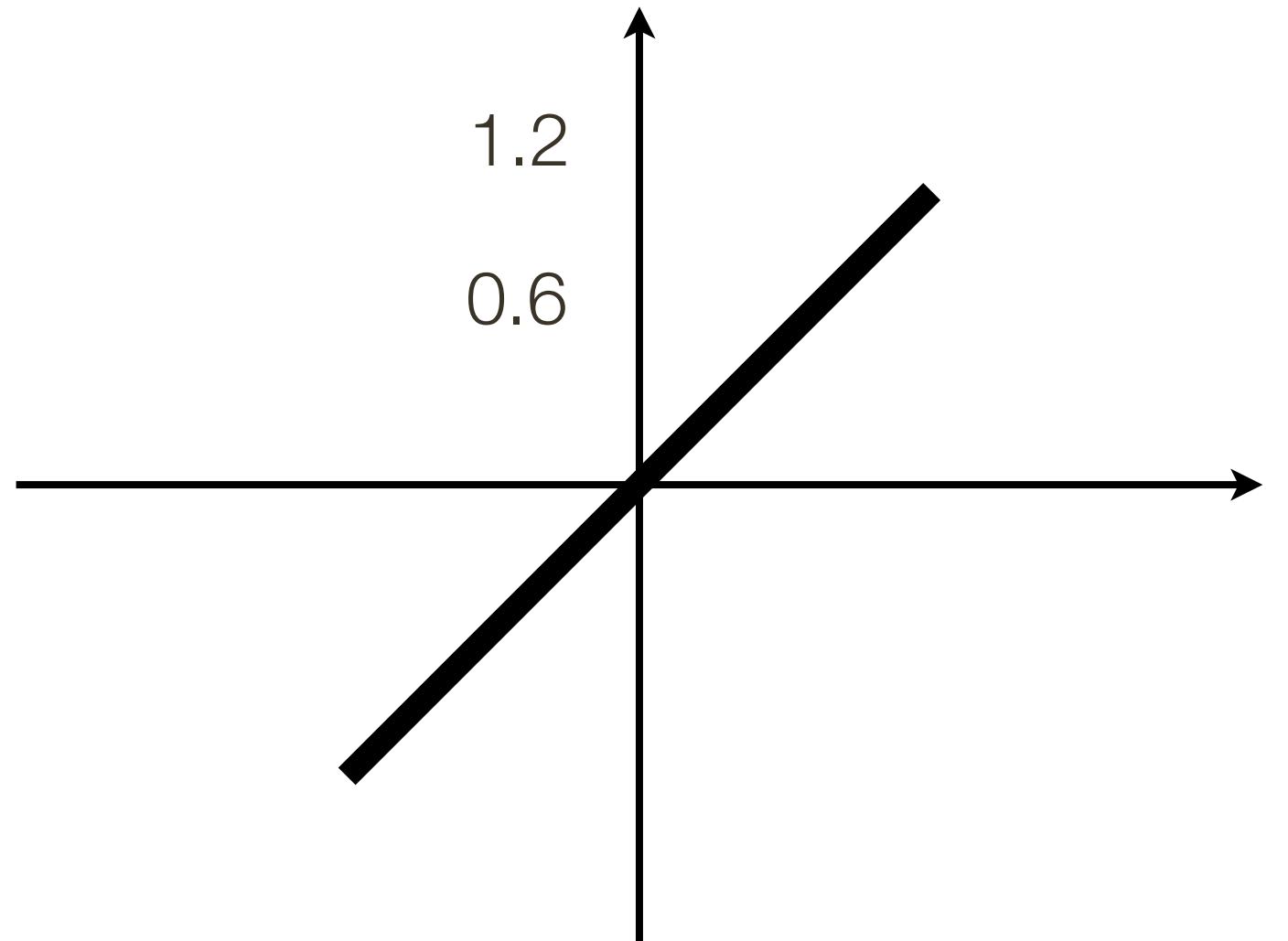
Recall: $a(x) = x$

Why?

$$\mathbf{W}_o (\mathbf{W}_{h2} (\mathbf{W}_{h1} \mathbf{x} + \mathbf{b}_{h1}) + \mathbf{b}_{h2}) + \mathbf{b}_o =$$

$$\frac{[\mathbf{W}_o \mathbf{W}_{h1} \mathbf{W}_{h2}] \mathbf{x} + [\mathbf{W}_o \mathbf{W}_{h1} \mathbf{b}_{h1} + \mathbf{W}_o \mathbf{b}_{h2} + \mathbf{b}_o]}{\mathbf{W}'}$$

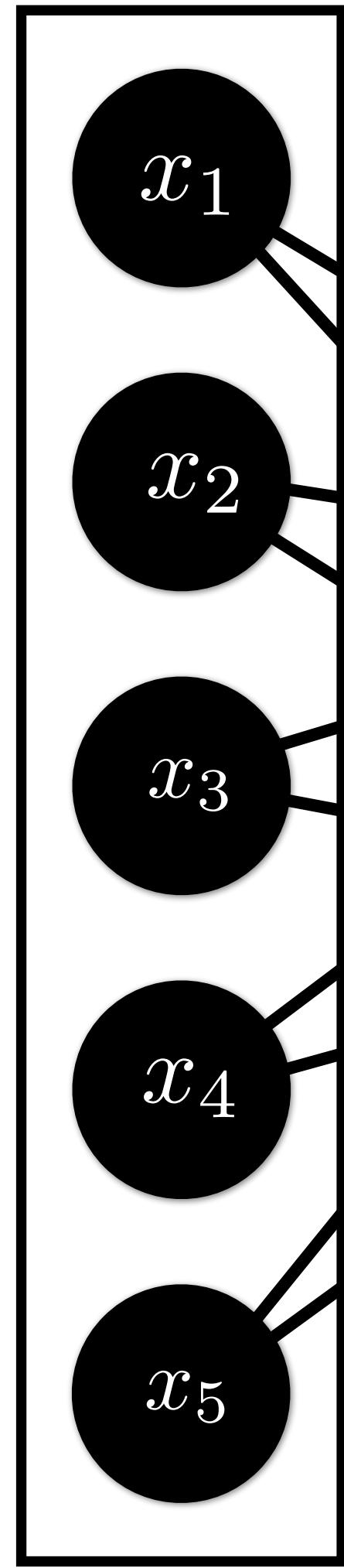
\mathbf{b}'



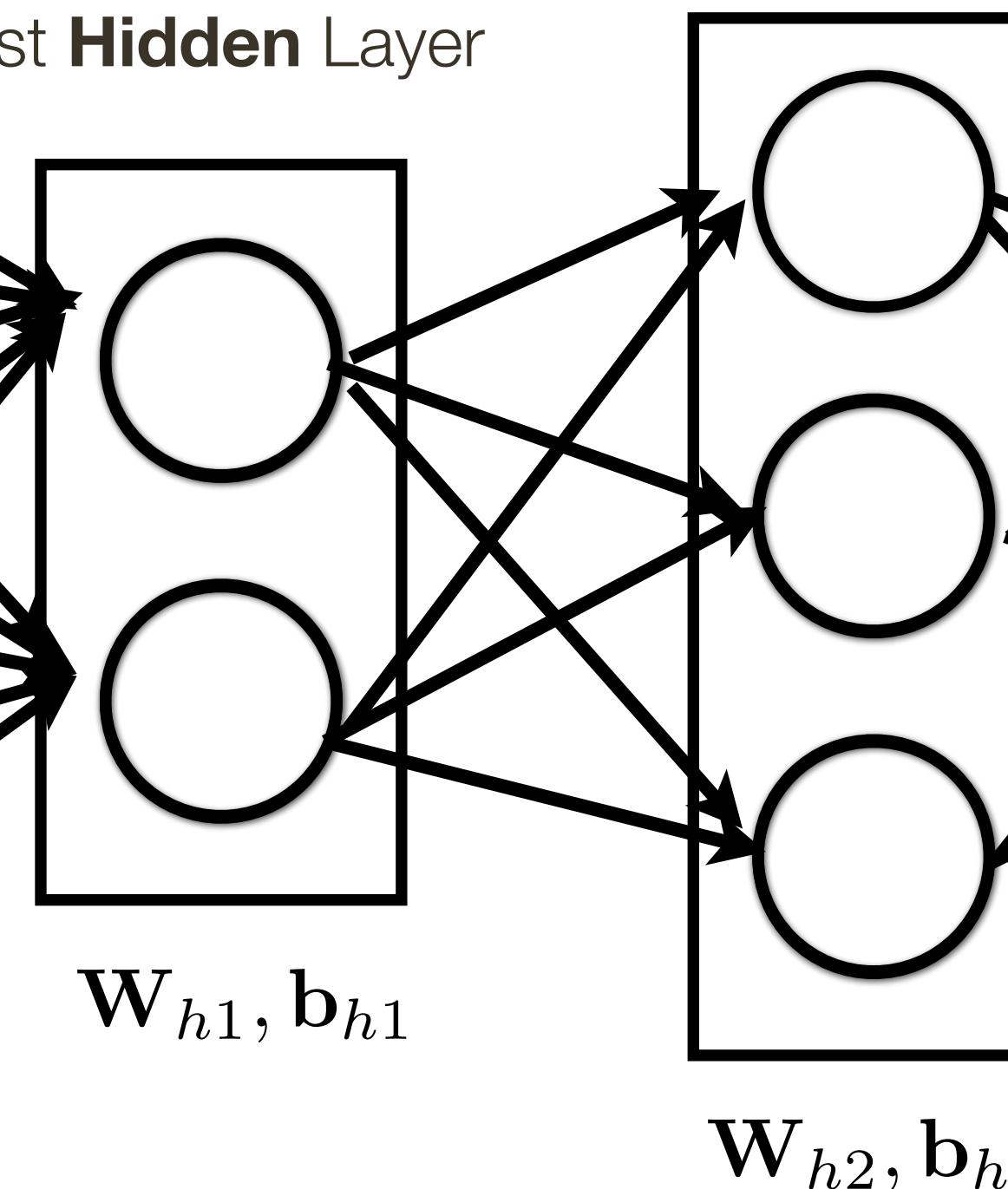
Linear Activation

Multi-layer Neural Network

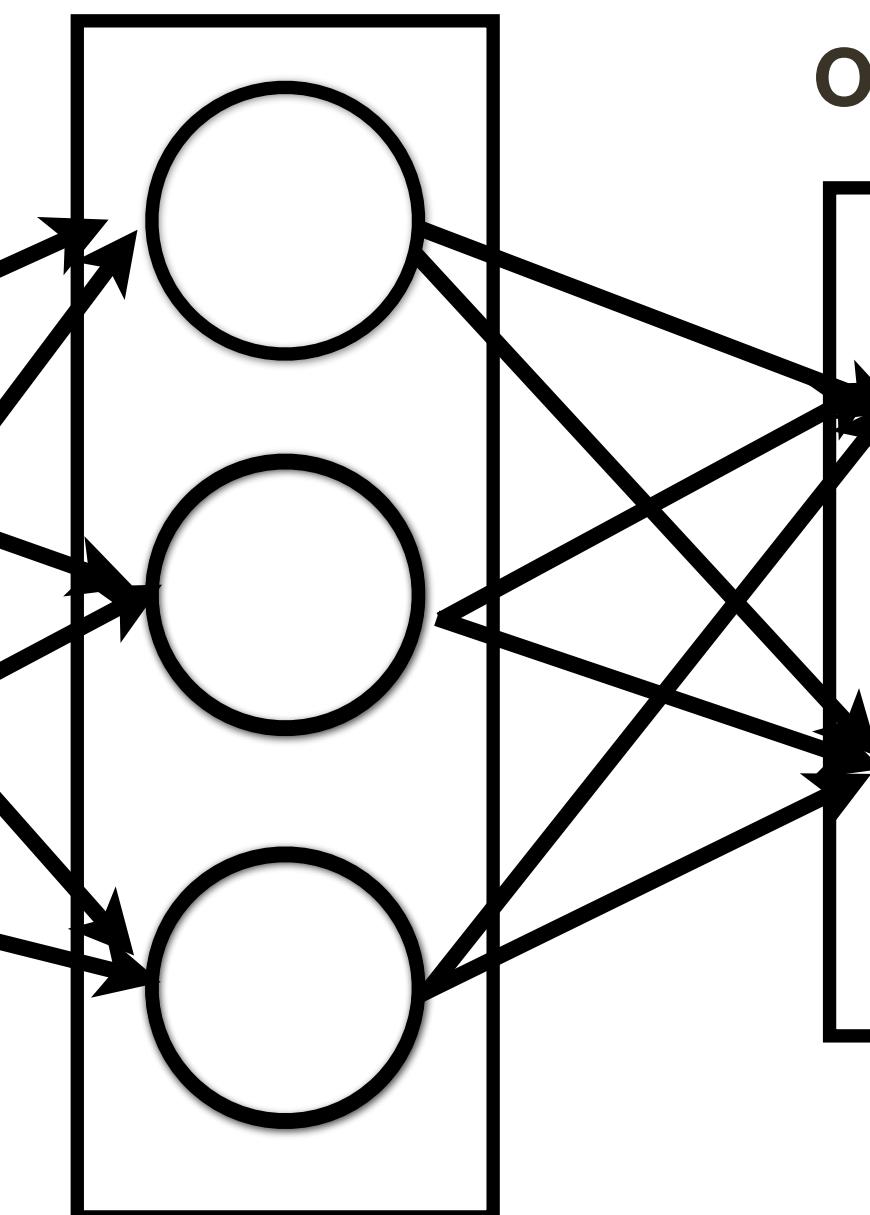
Input Layer



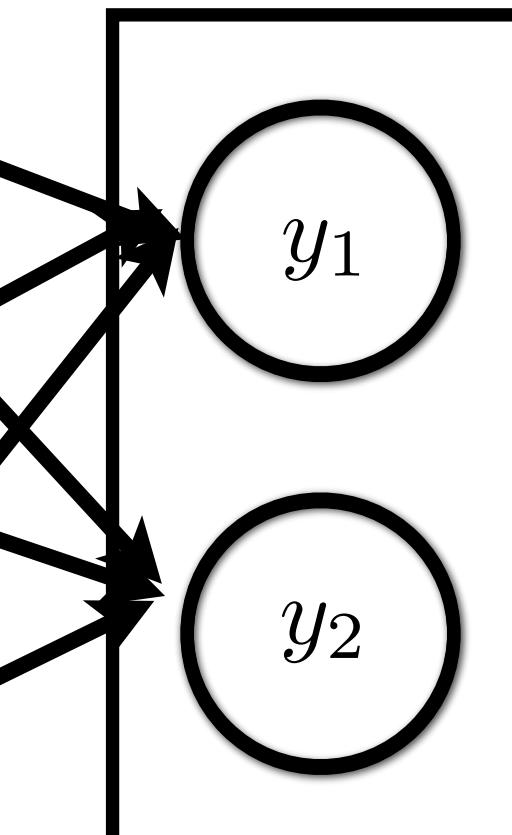
1st Hidden Layer



2nd Hidden Layer



Output Layer

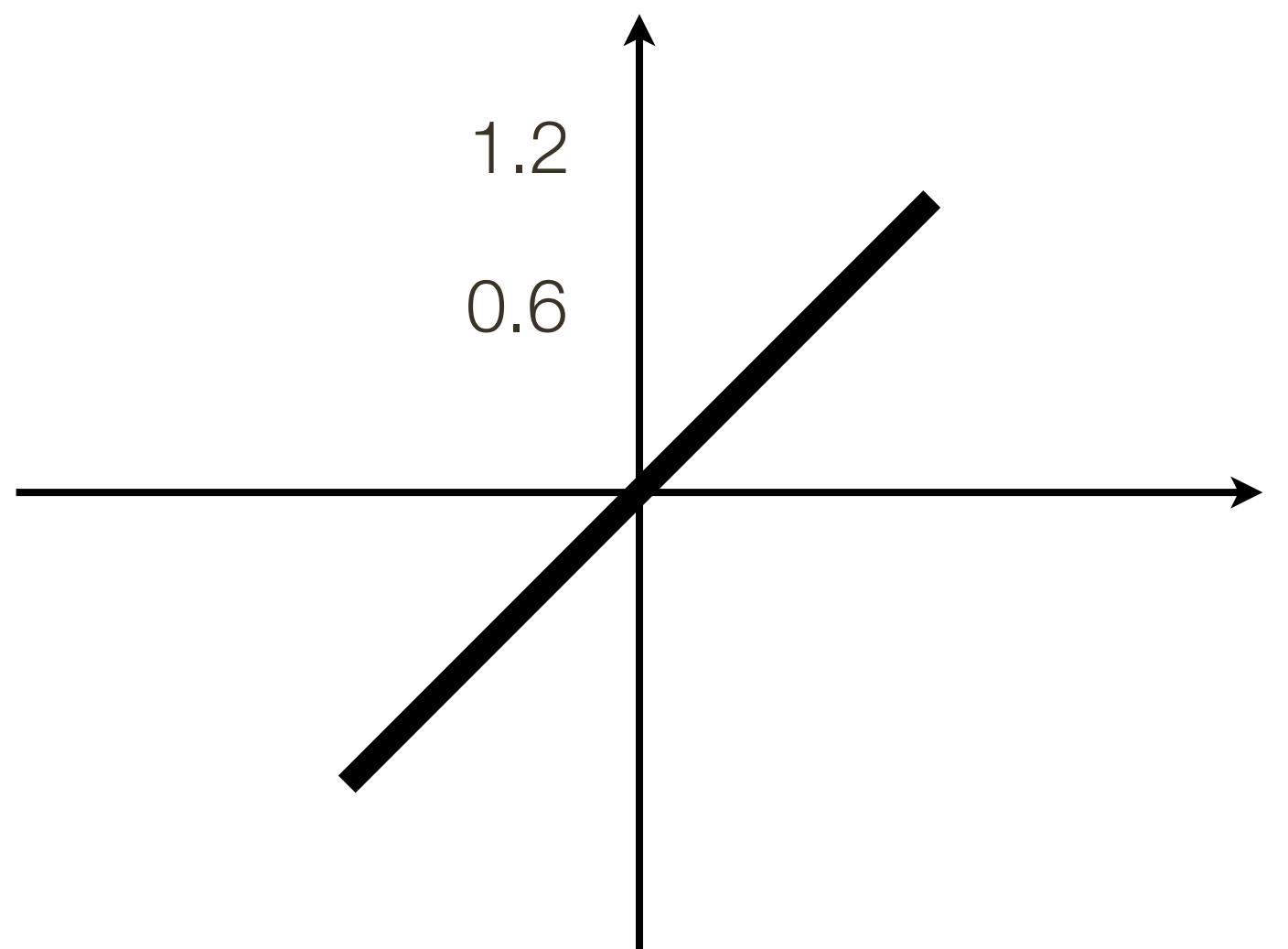


Why?

$$\mathbf{W}_o (\mathbf{W}_{h2} (\mathbf{W}_{h1} \mathbf{x} + \mathbf{b}_{h1}) + \mathbf{b}_{h2}) + \mathbf{b}_o =$$

$$\frac{[\mathbf{W}_o \mathbf{W}_{h1} \mathbf{W}_{h2}] \mathbf{x} + [\mathbf{W}_o \mathbf{W}_{h1} \mathbf{b}_{h1} + \mathbf{W}_o \mathbf{b}_{h2} + \mathbf{b}_o]}{\mathbf{W}'}$$

\mathbf{b}'

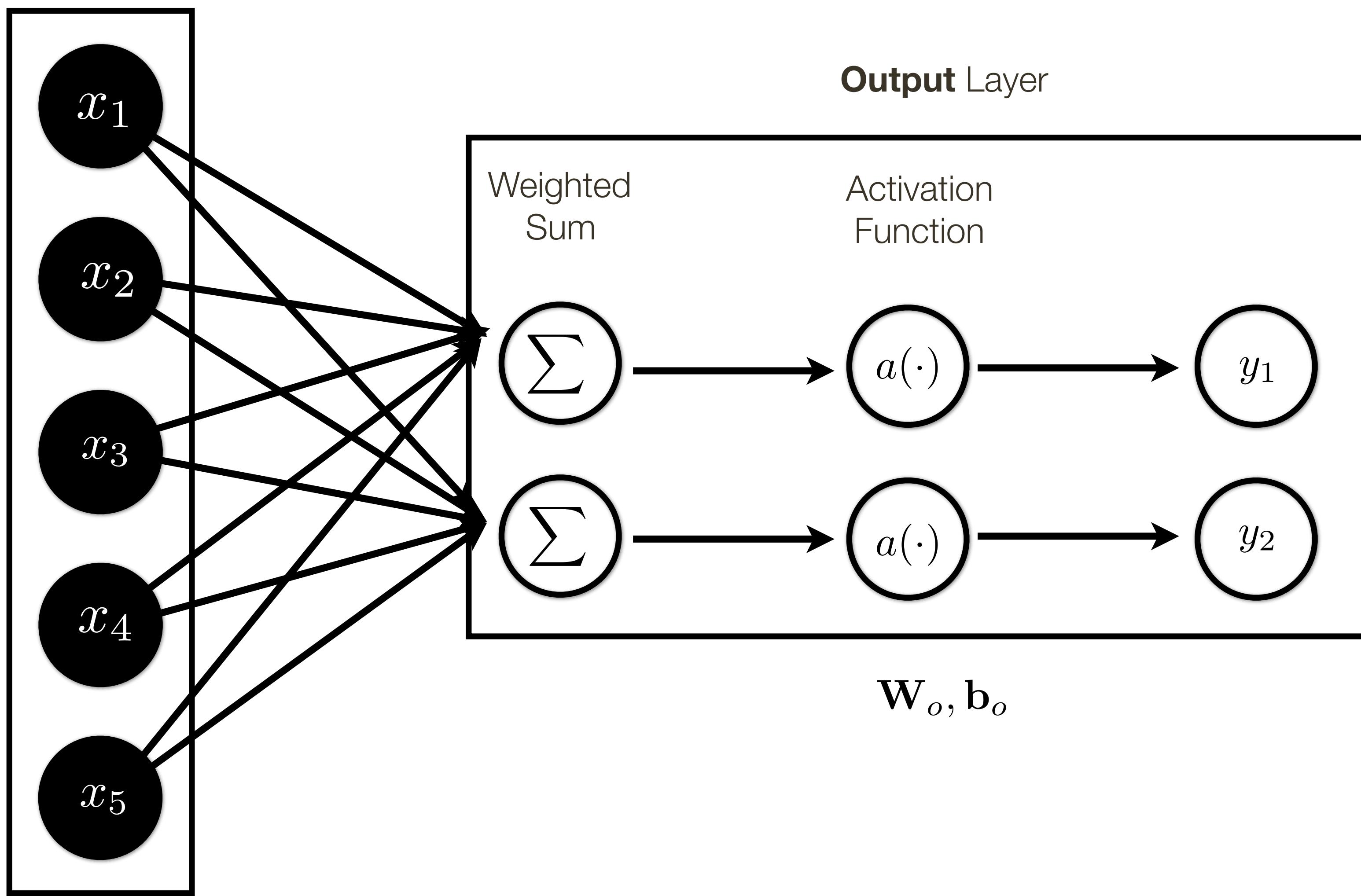


Linear Activation

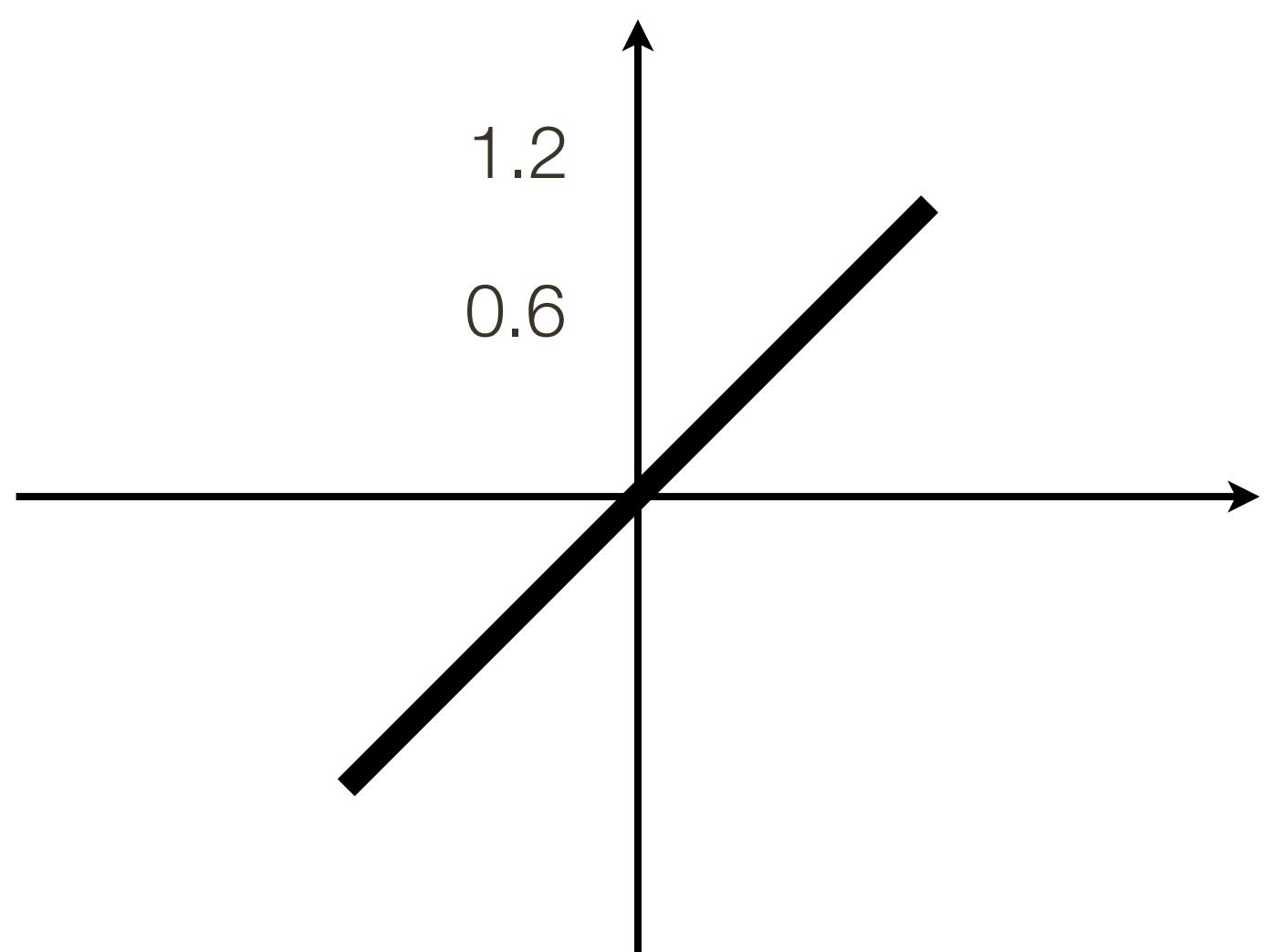
Recall: $a(x) = x \Rightarrow$ entire neural network is linear, which is **not expressive**

One-layer Neural Network

Input Layer



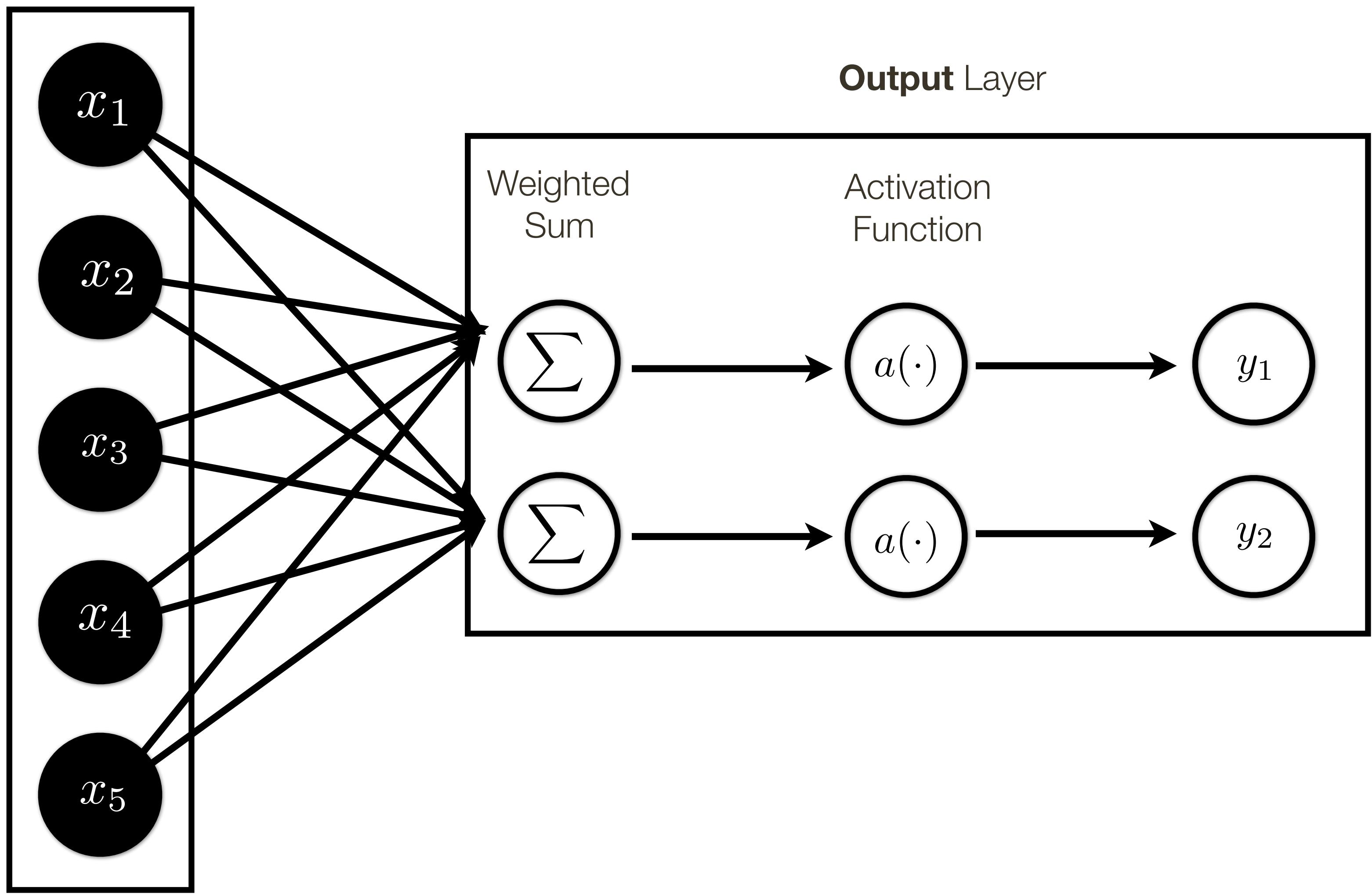
$$a(x) = x$$



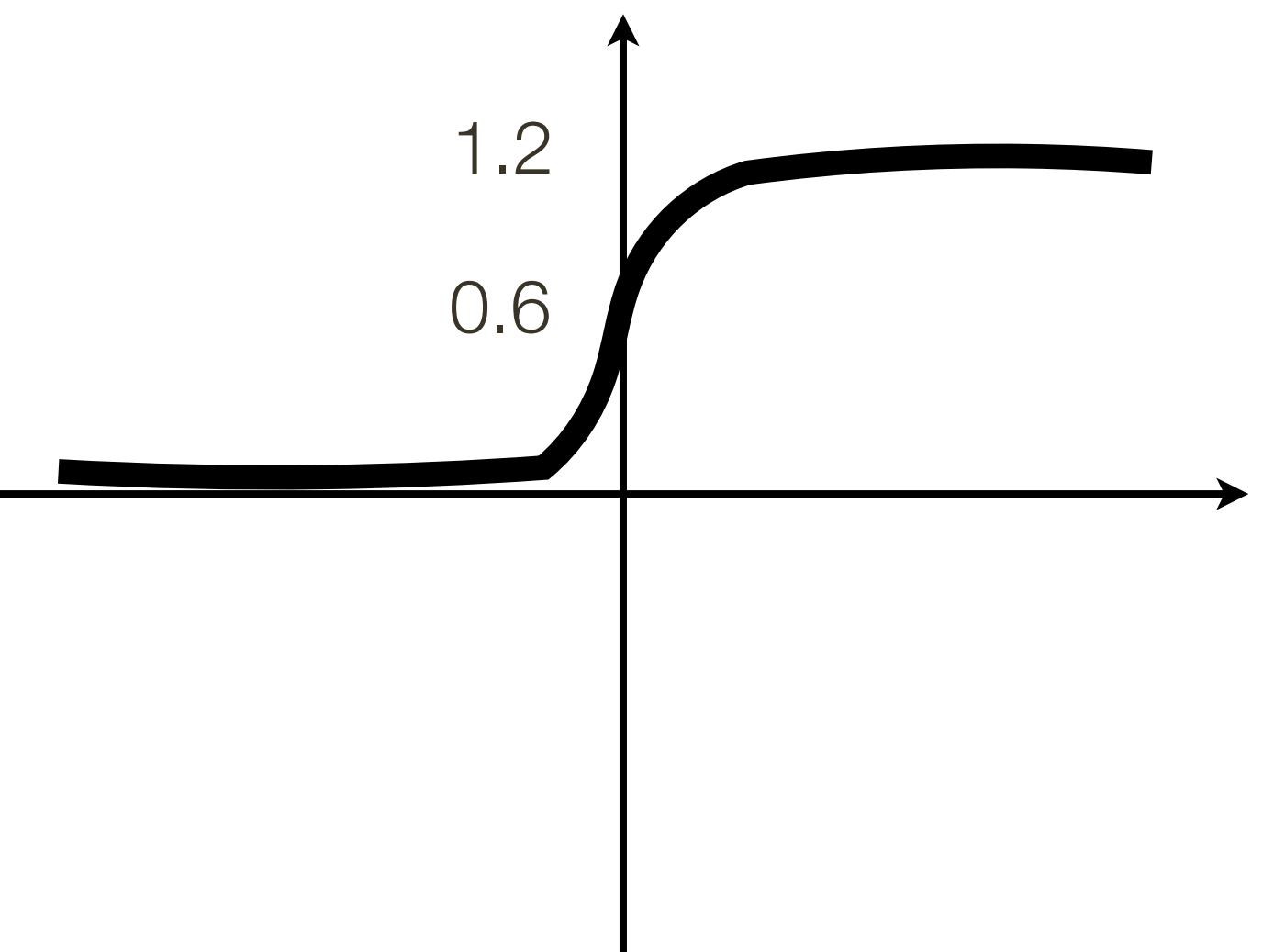
Linear Activation

One-layer Neural Network

Input Layer



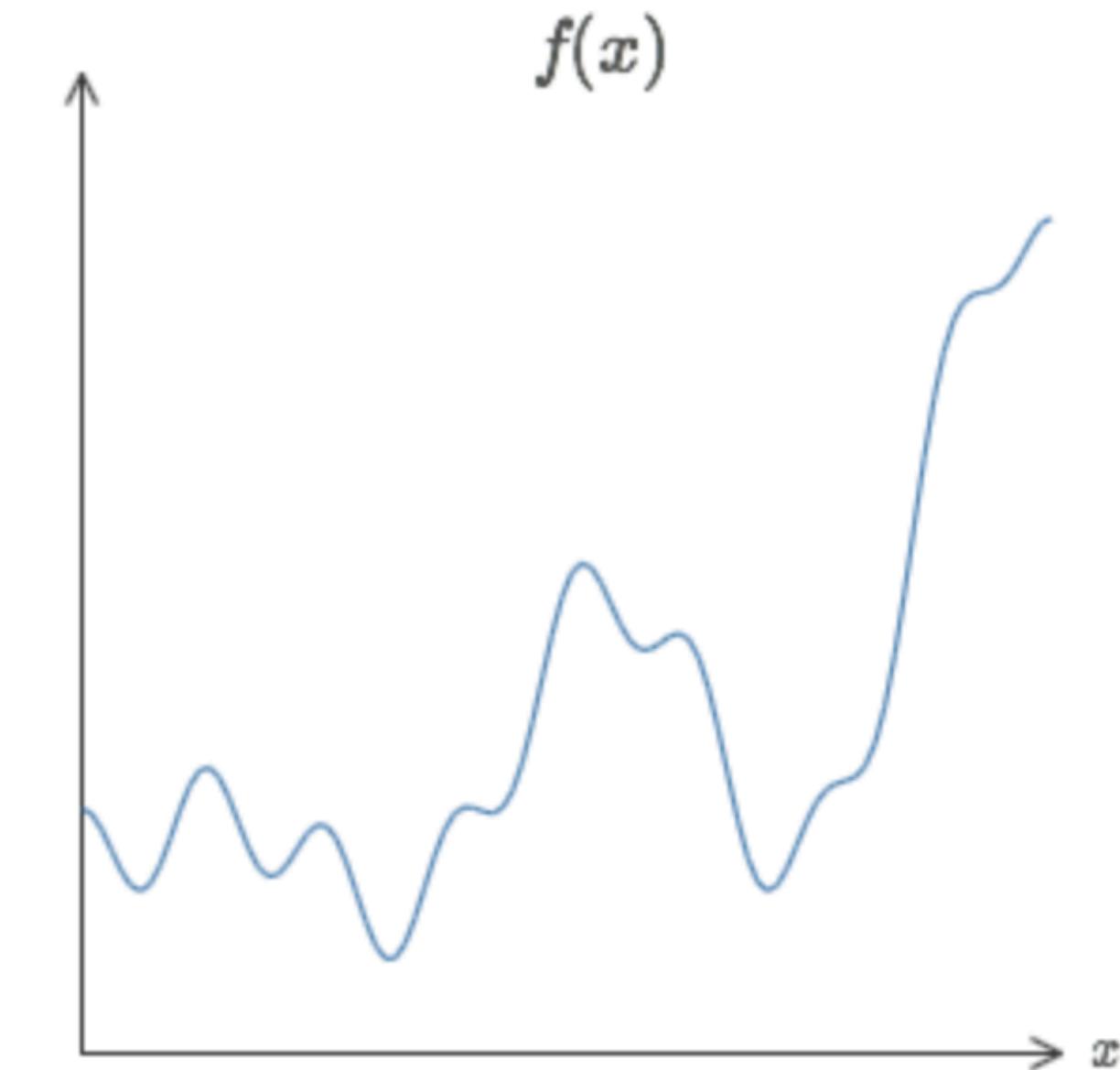
$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



Sigmoid Activation

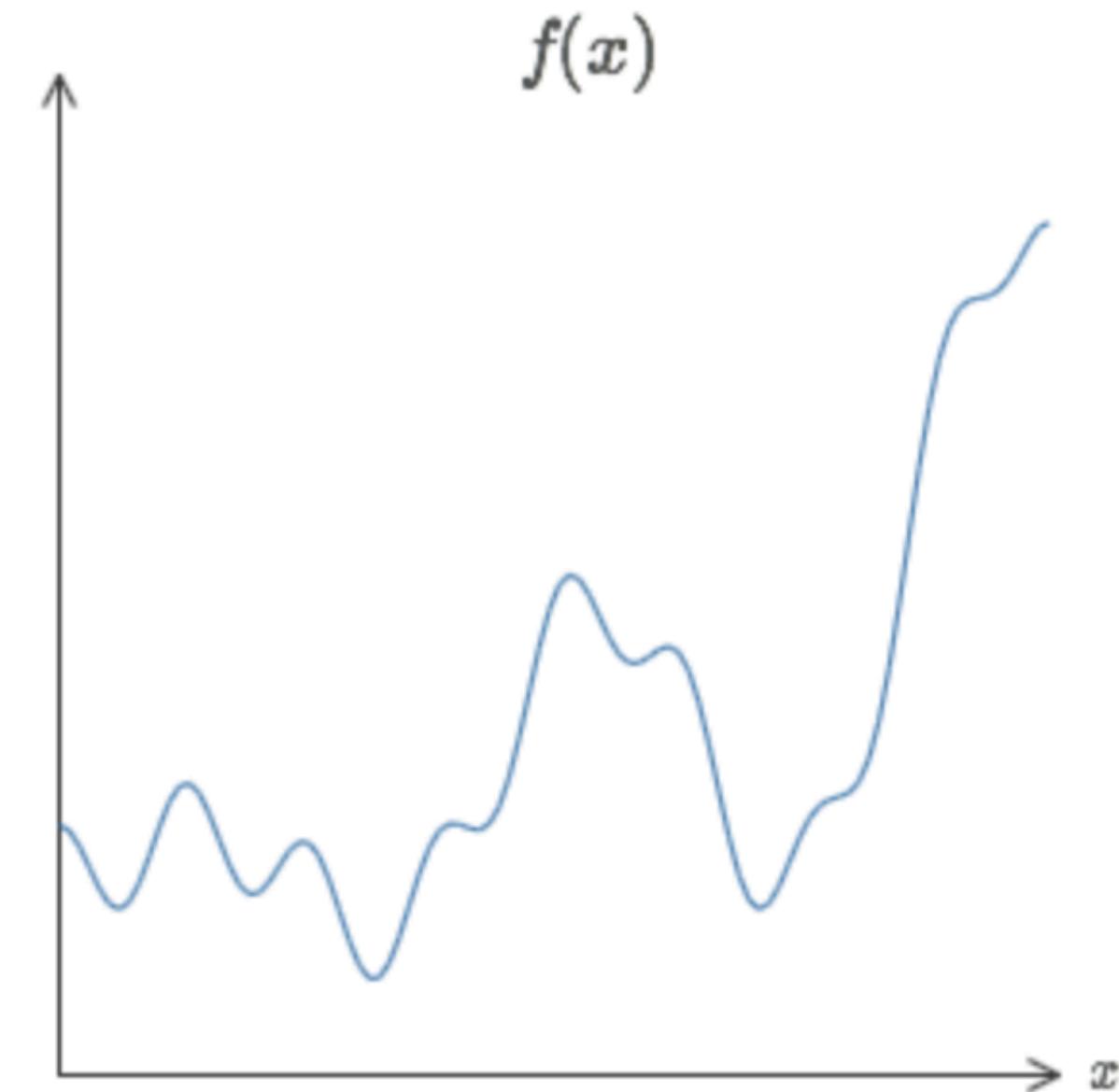
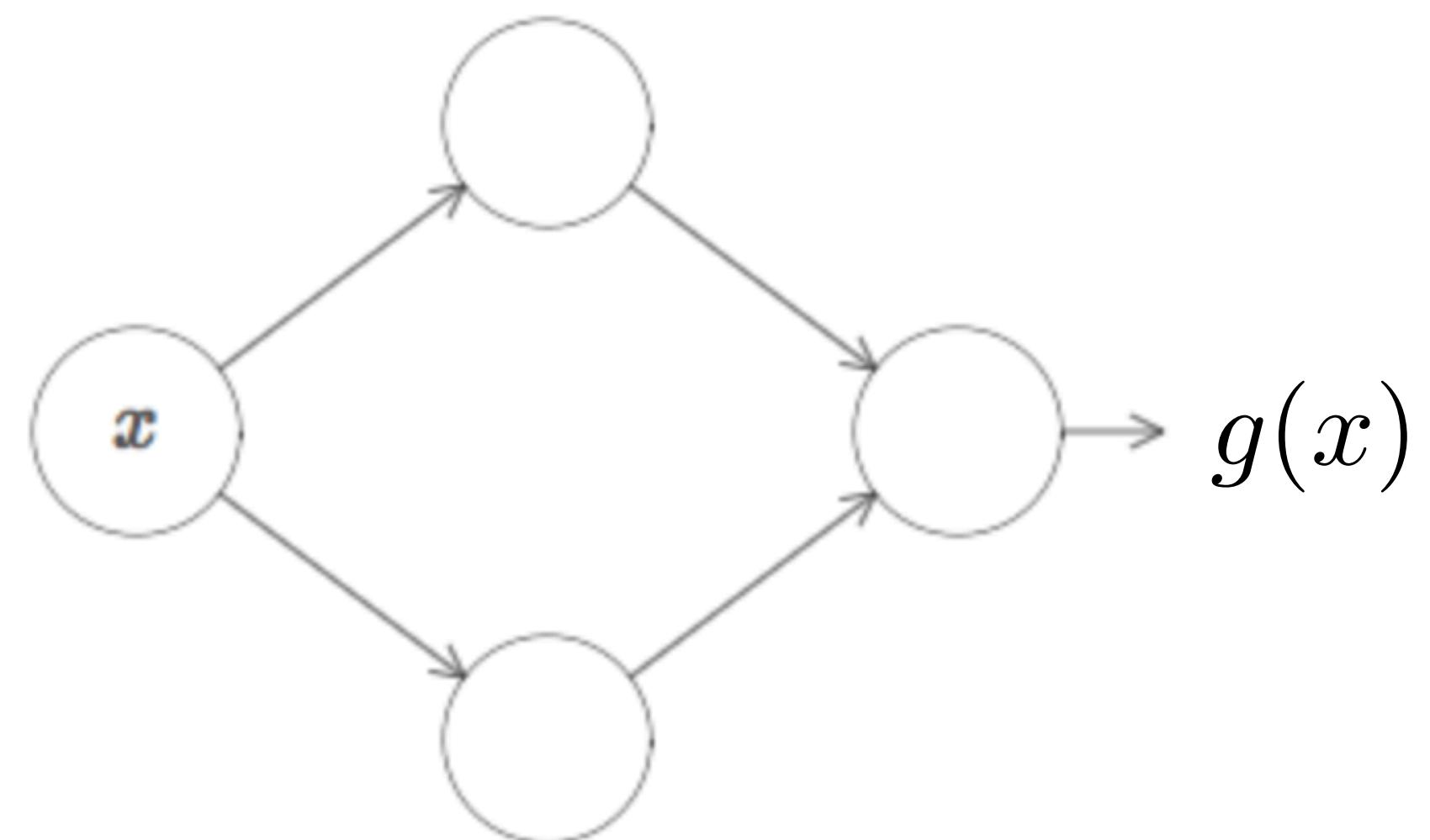
Light Theory: Neural Network as Universal Approximator

Neural network can arbitrarily approximate *any* **continuous** function for every value of possible inputs



Light Theory: Neural Network as Universal Approximator

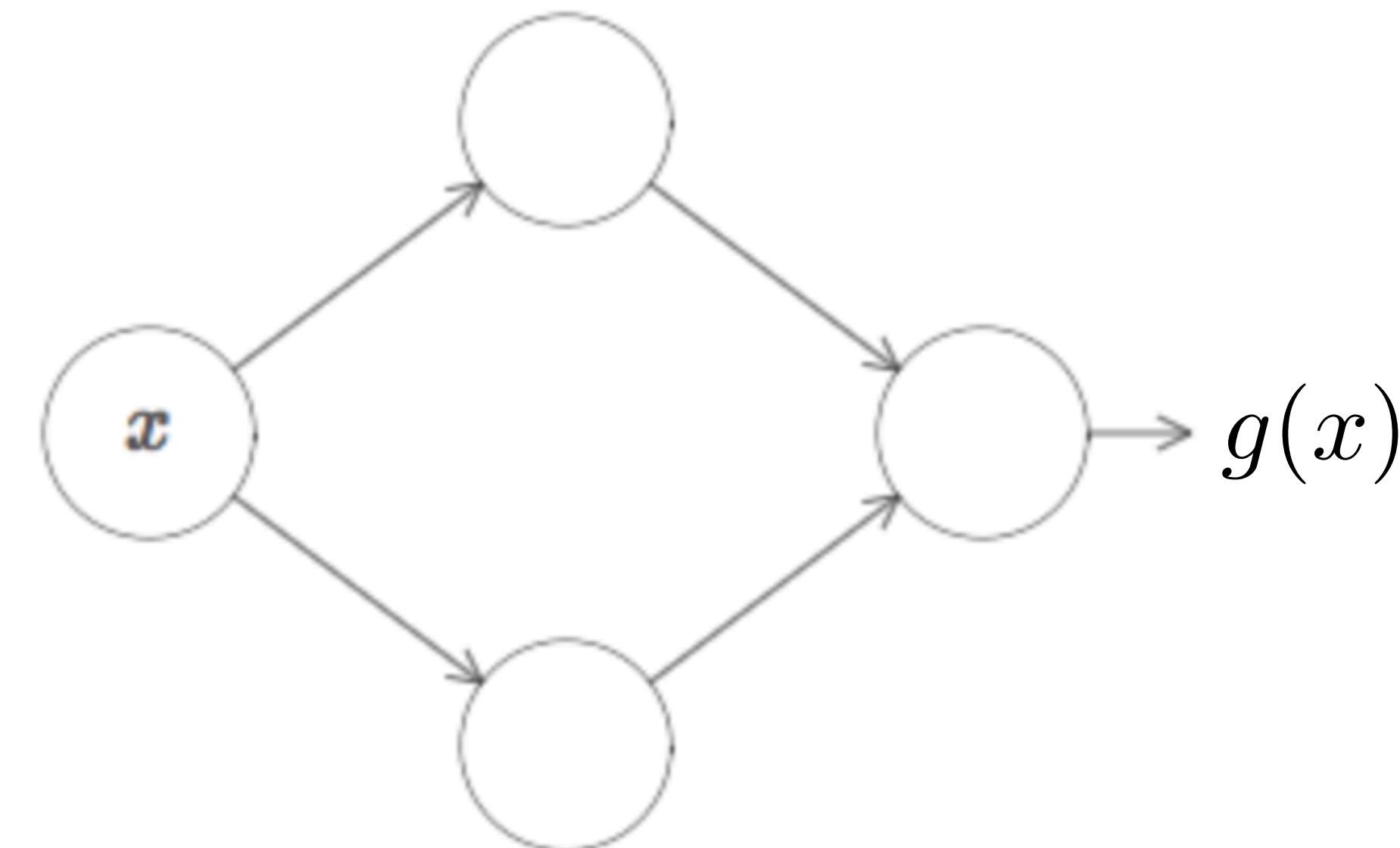
Neural network can arbitrarily approximate any **continuous** function for every value of possible inputs



The guarantee is that by using enough hidden neurons we can always find a neural network whose output $g(x)$ satisfies $|g(x) - f(x)| < \epsilon$ for an arbitrarily small ϵ

Light Theory: Neural Network as Universal Approximator

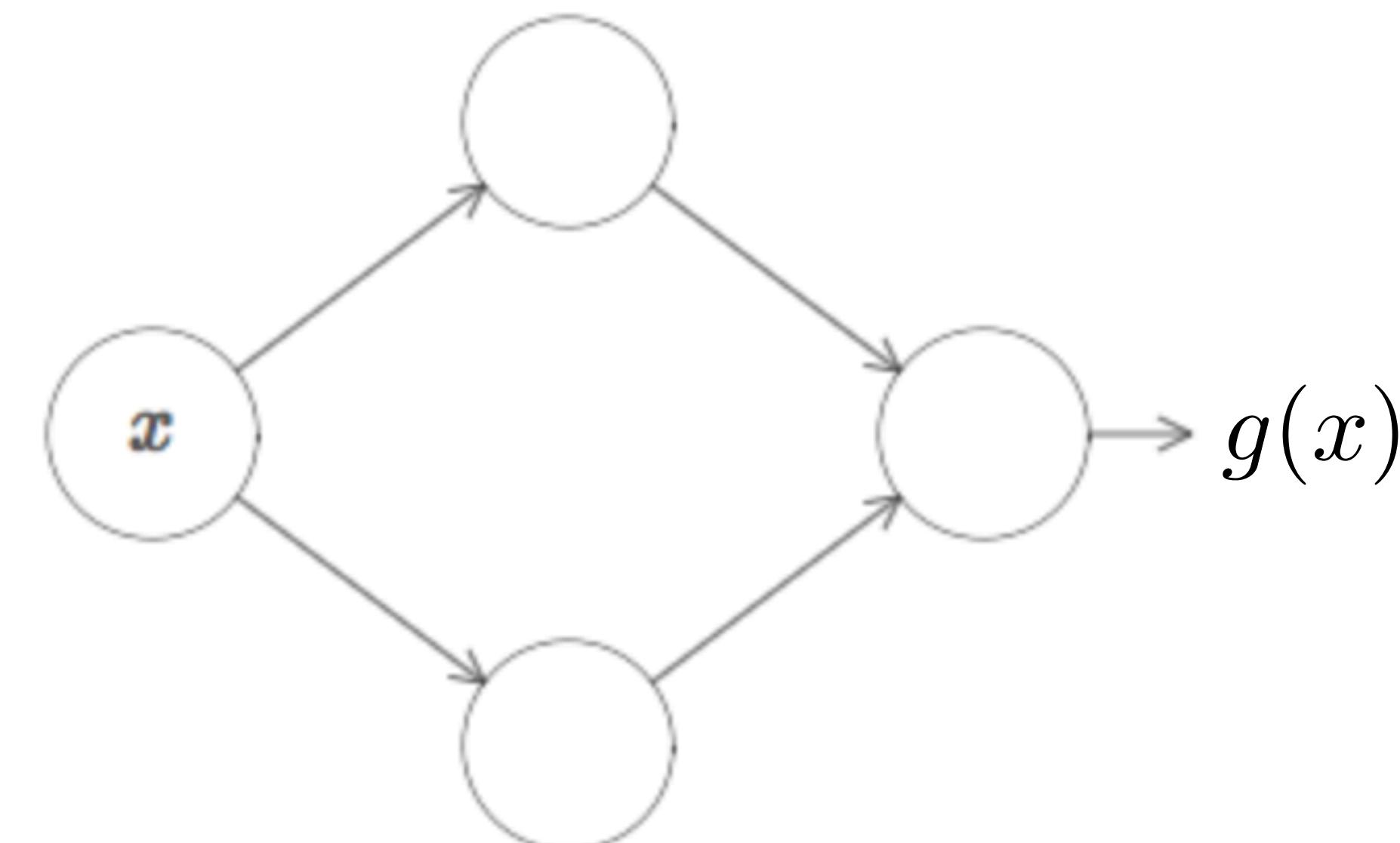
Lets start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)



Light Theory: Neural Network as Universal Approximator

Lets start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)

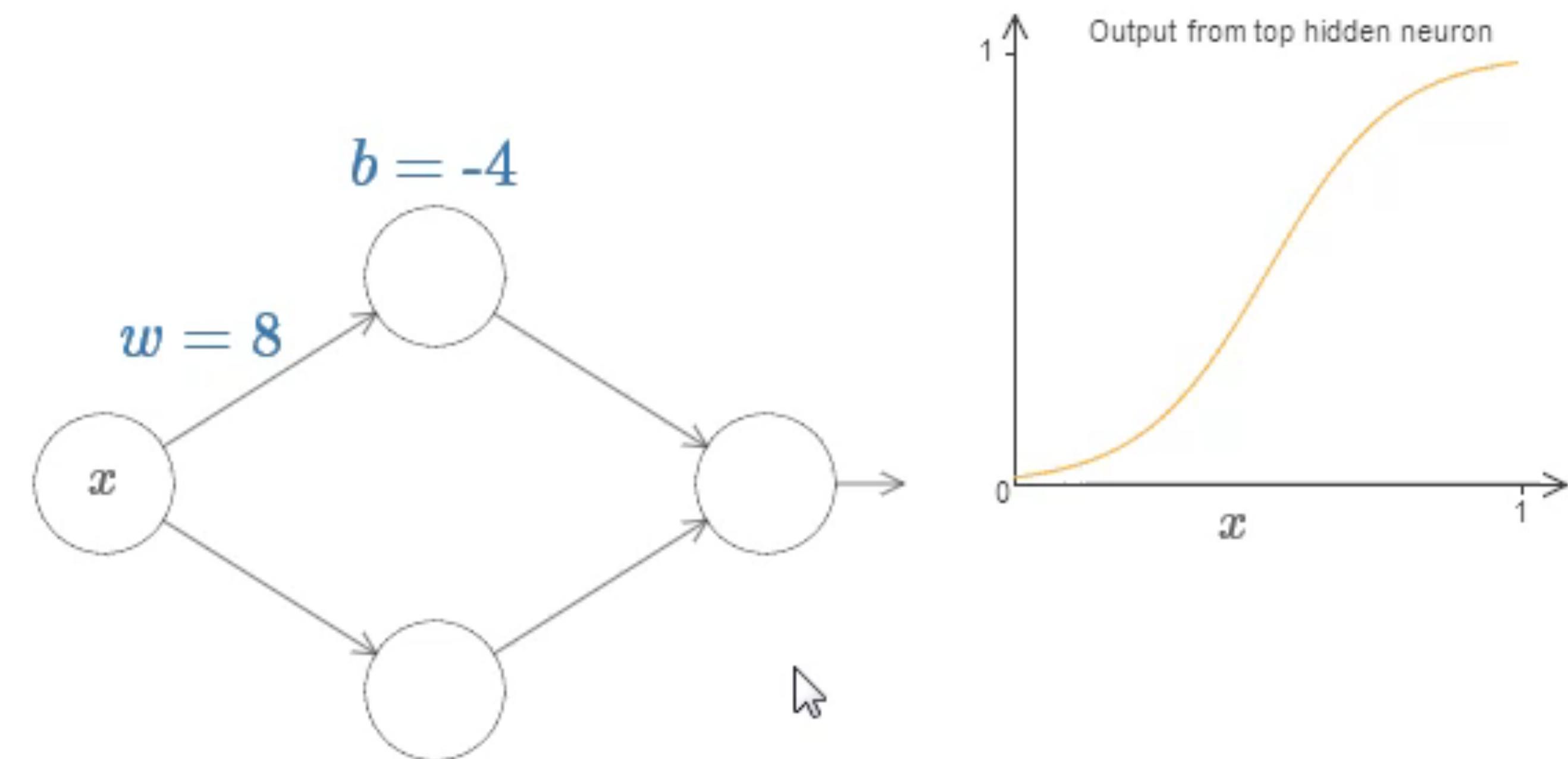
Let's look at output of this (hidden) neuron as a function of parameters (weight, bias)



Light Theory: Neural Network as Universal Approximator

Lets start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)

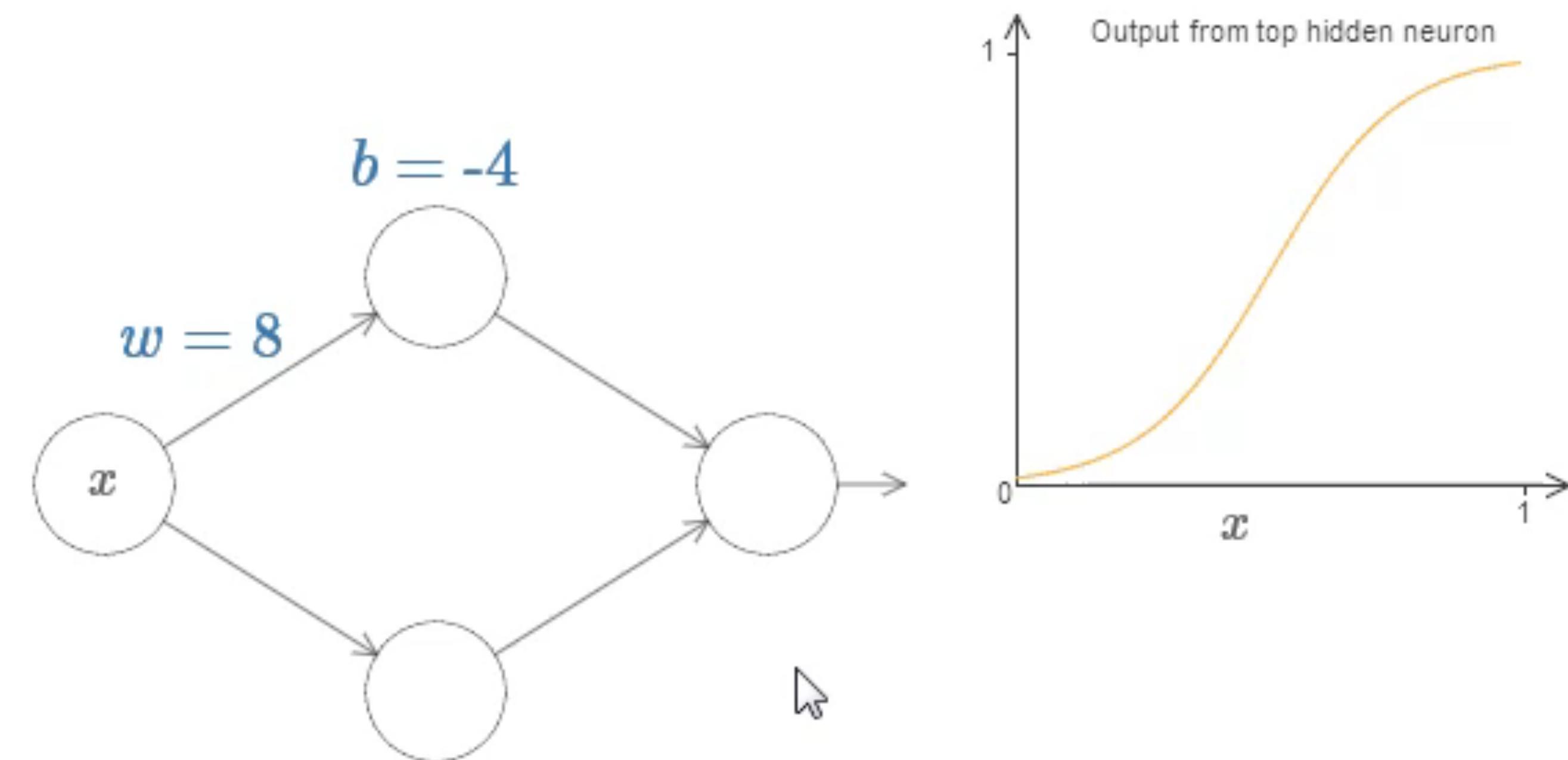
Let's look at output of this (hidden) neuron as a function of parameters (weight, bias)



Light Theory: Neural Network as Universal Approximator

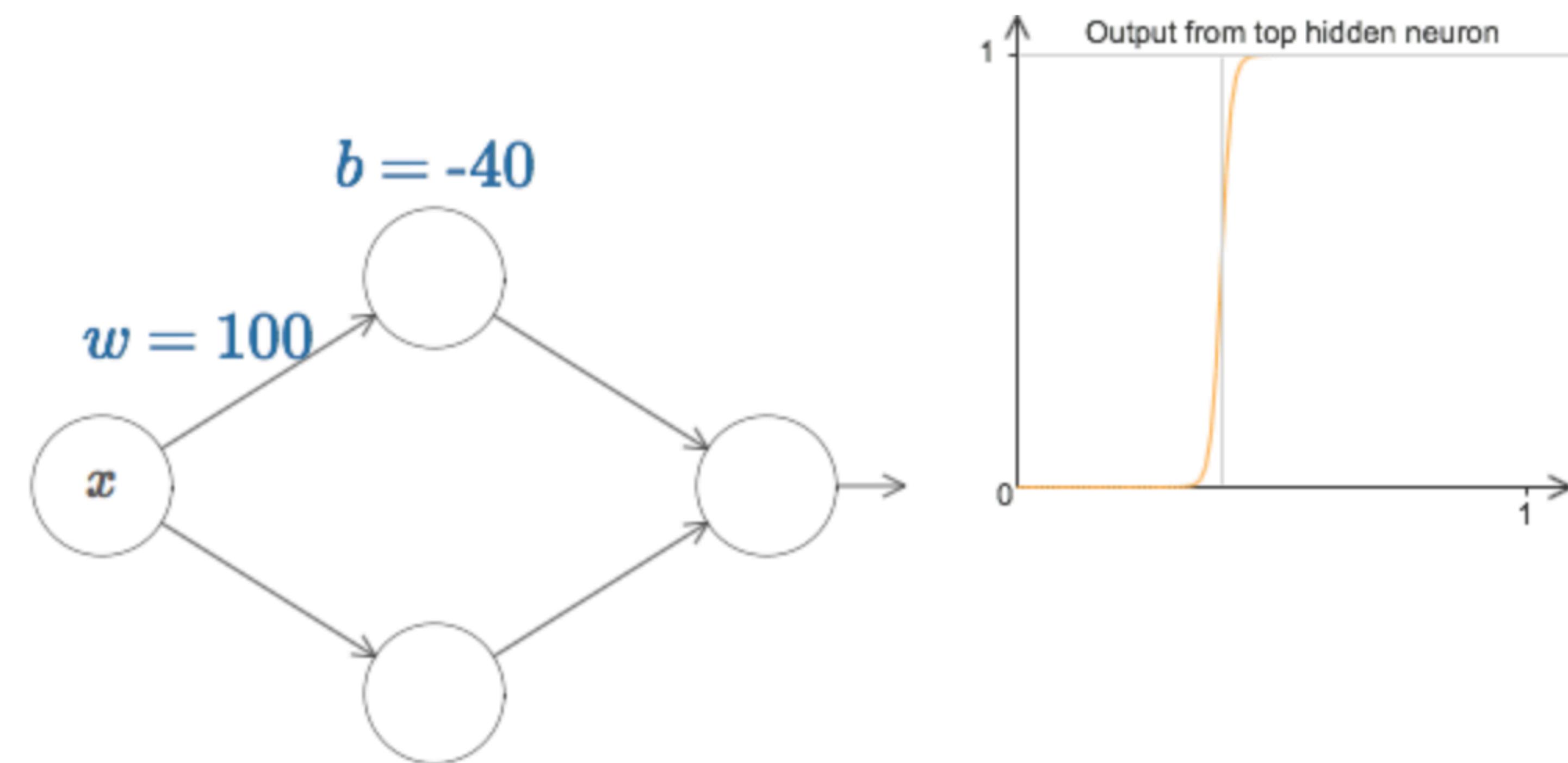
Lets start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)

Let's look at output of this (hidden) neuron as a function of parameters (weight, bias)



Light Theory: Neural Network as Universal Approximator

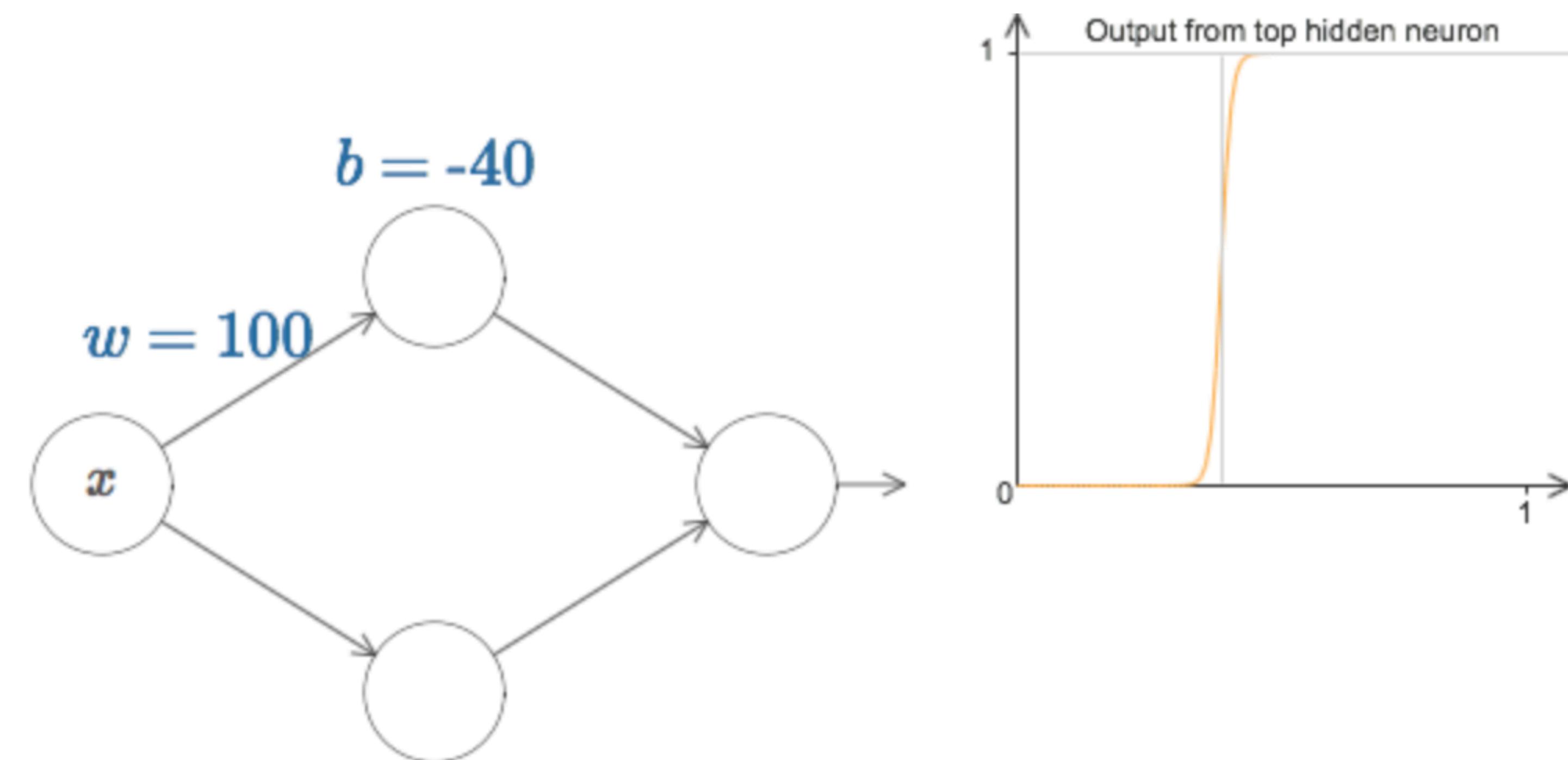
By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function



Light Theory: Neural Network as Universal Approximator

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

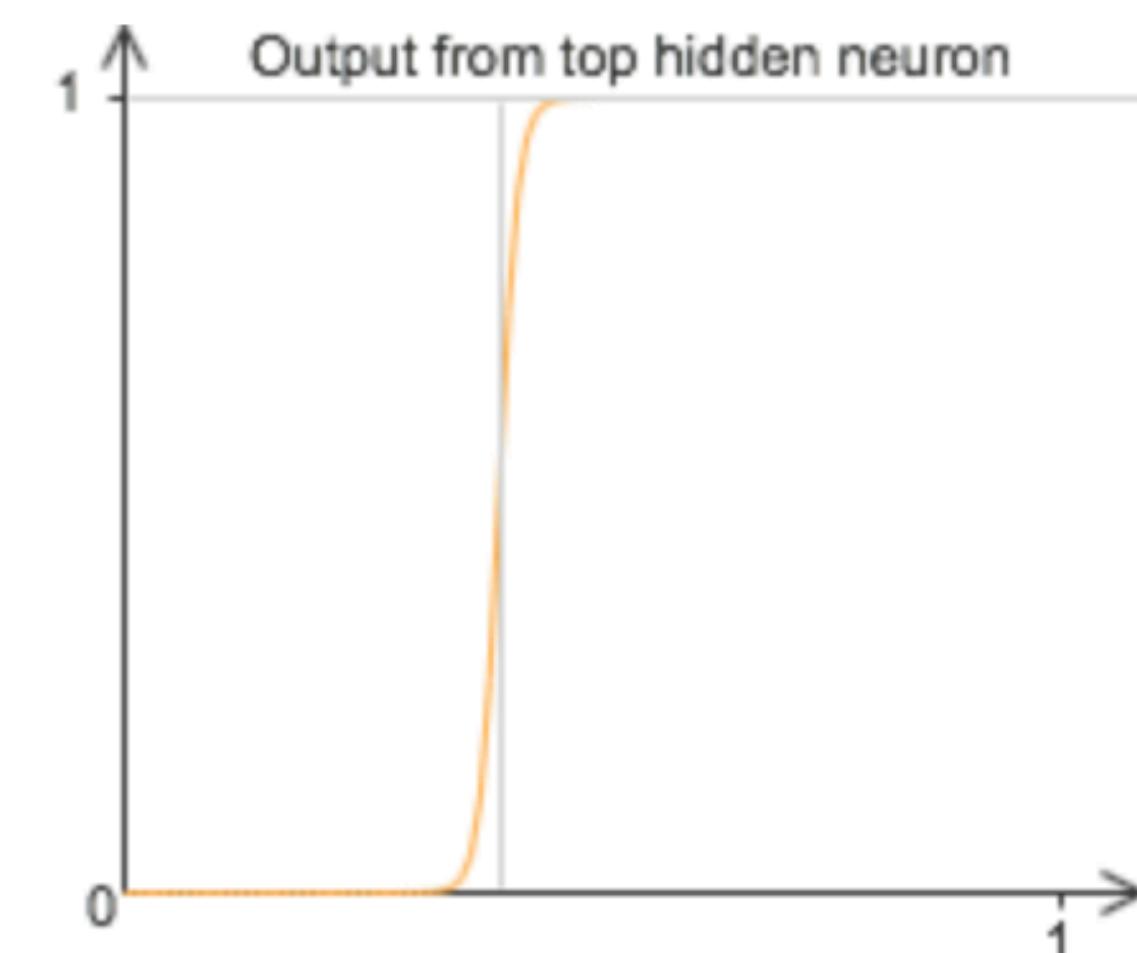
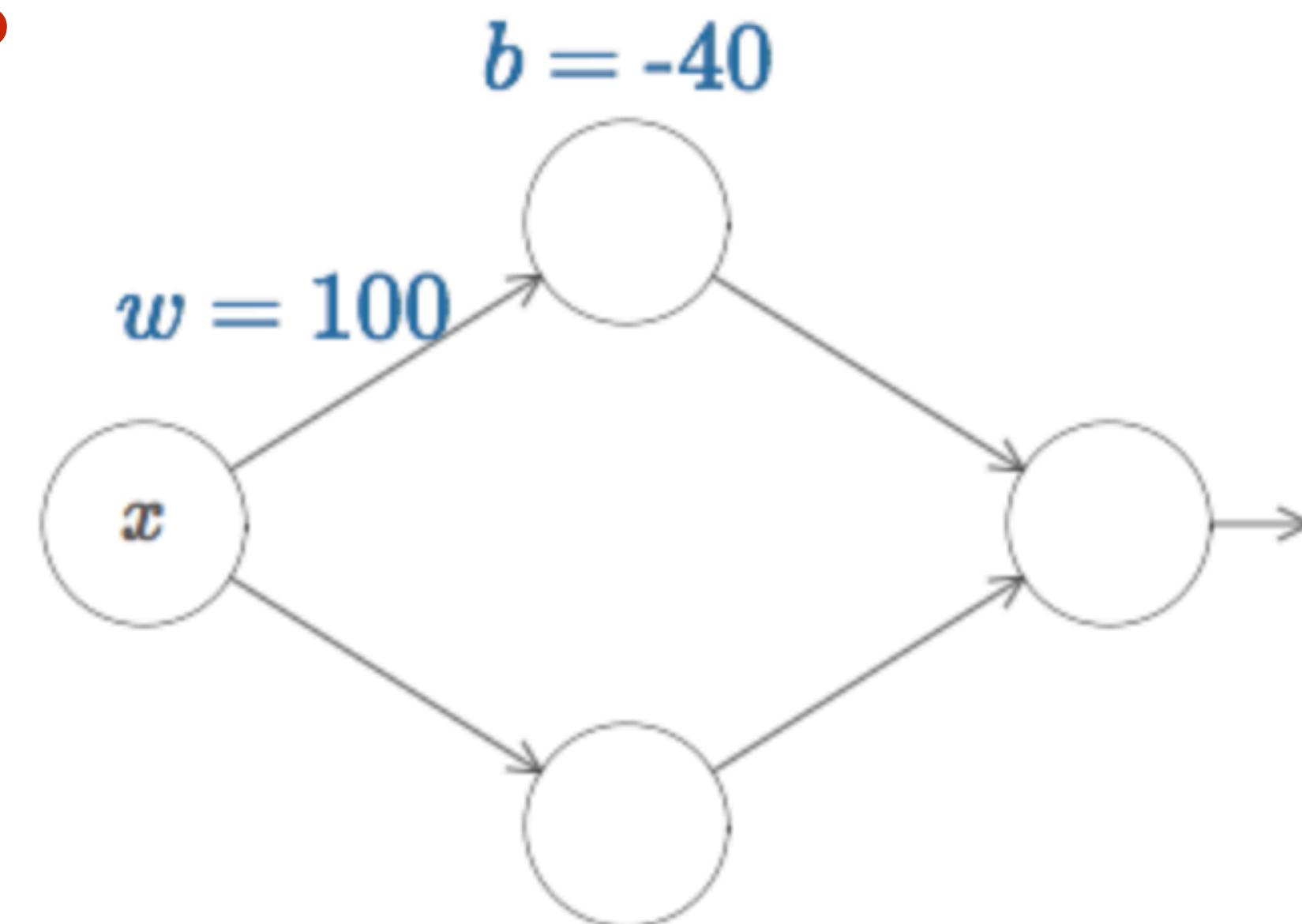


Light Theory: Neural Network as Universal Approximator

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

Location of the step?



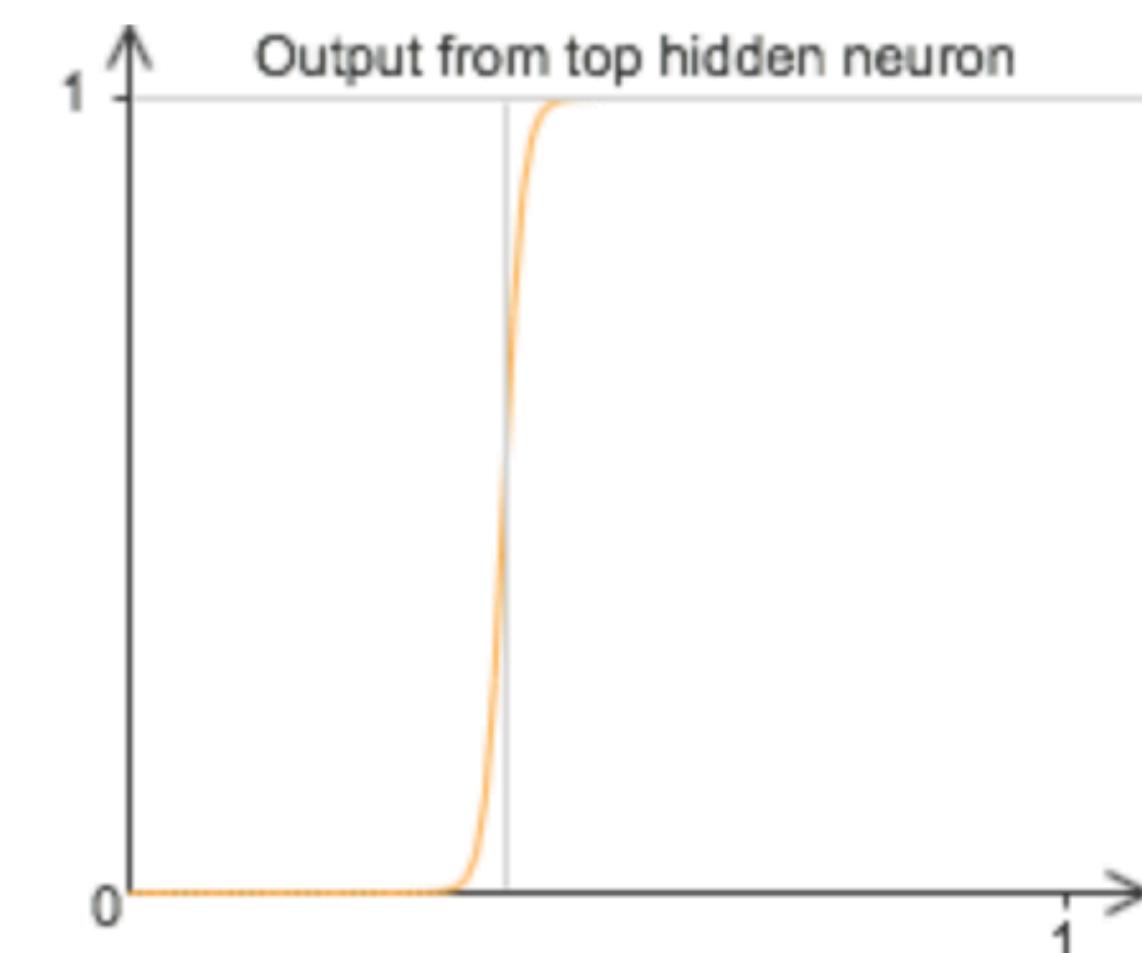
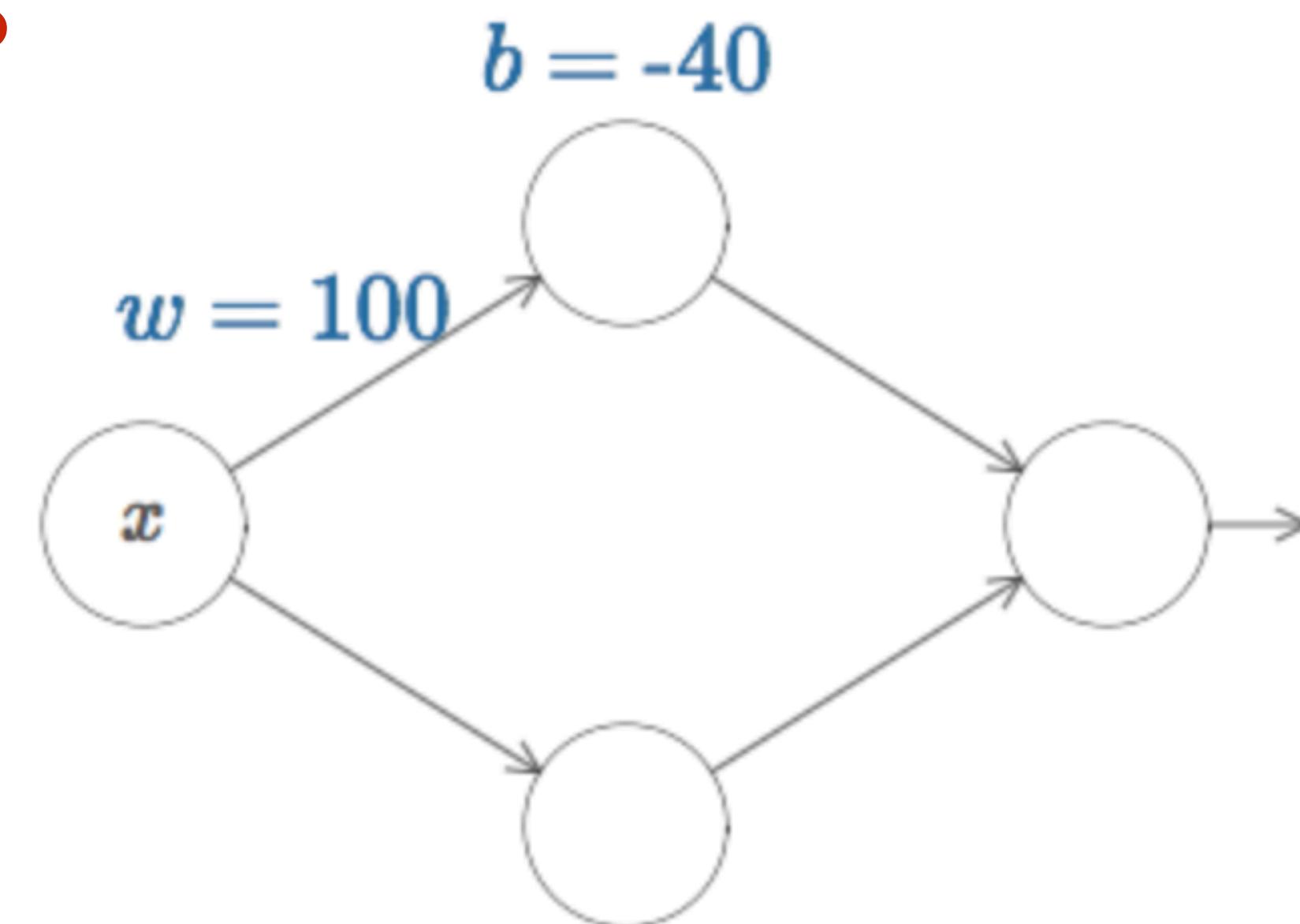
Light Theory: Neural Network as Universal Approximator

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

Location of the step?

$$s = -\frac{b}{w}$$



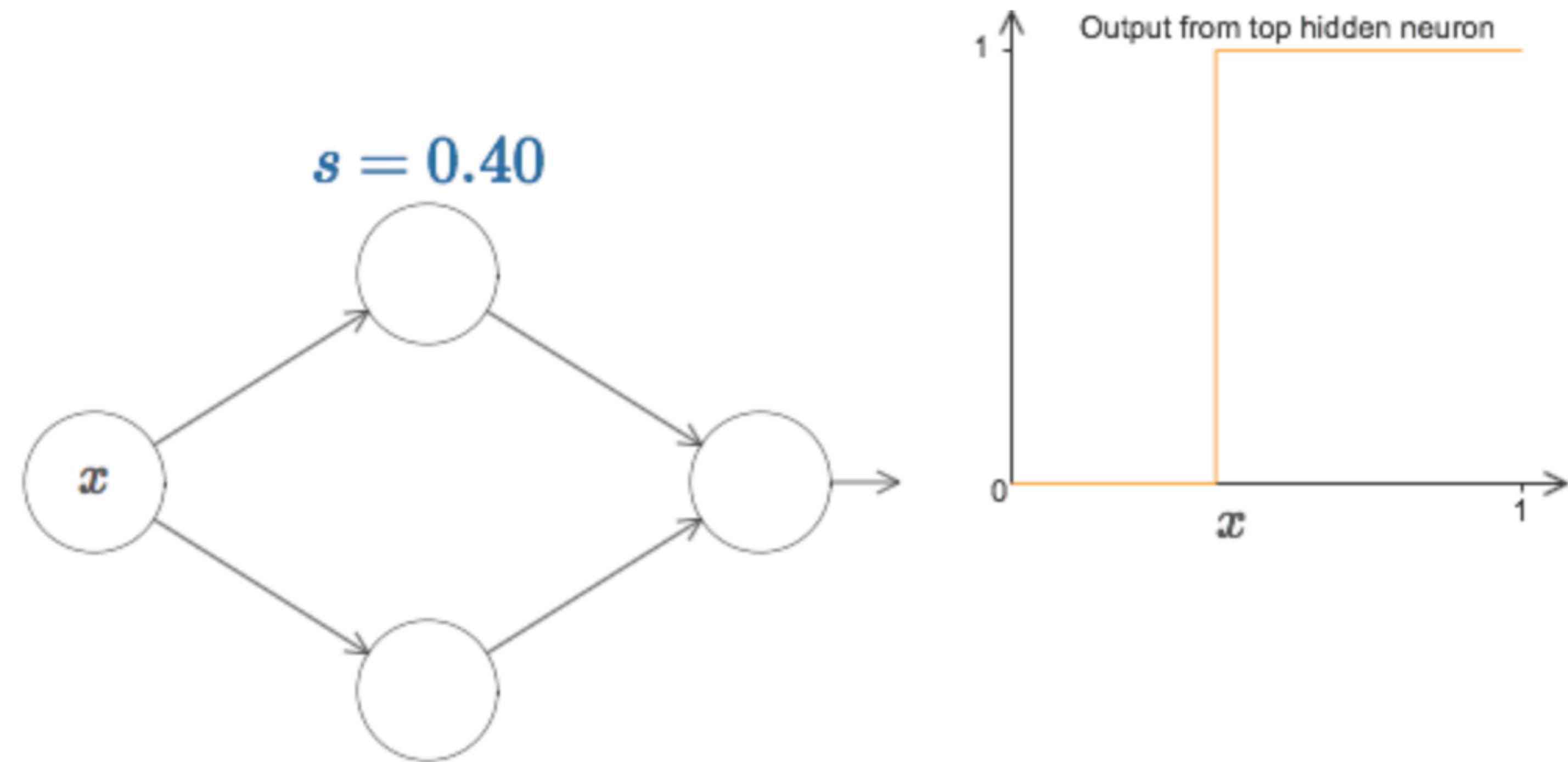
Light Theory: Neural Network as Universal Approximator

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function

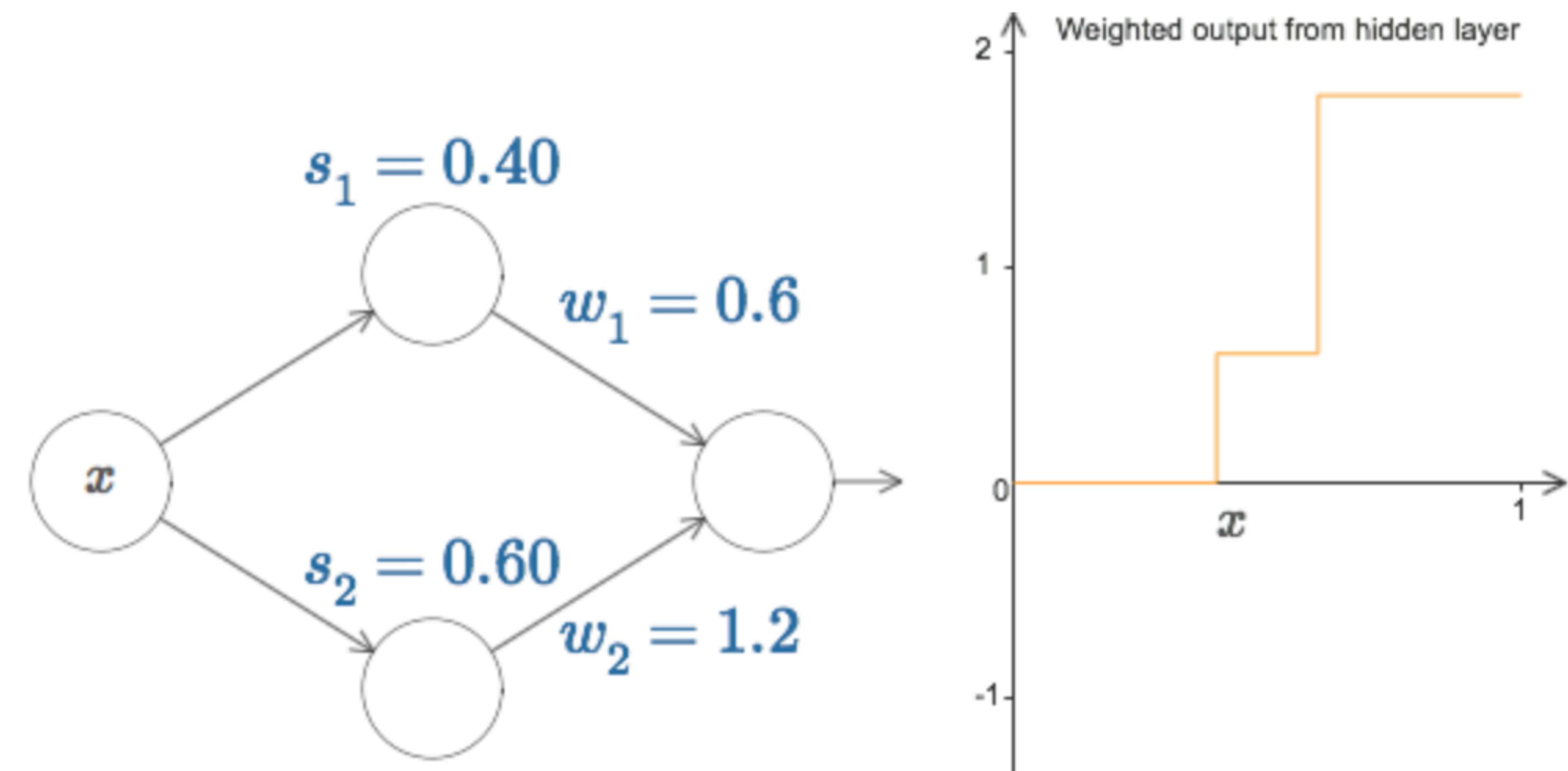
Location of the step?

$$s = -\frac{b}{w}$$



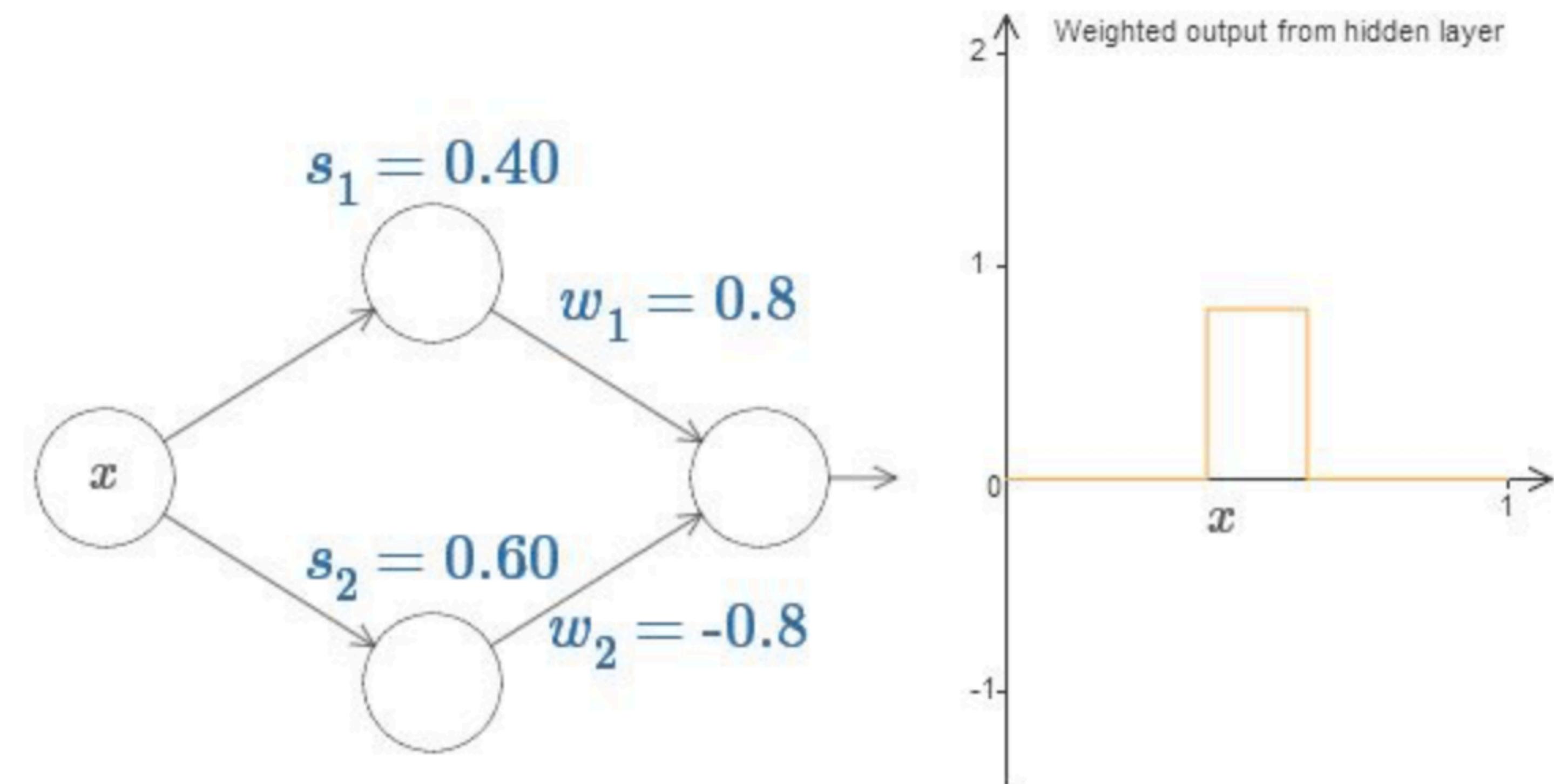
Light Theory: Neural Network as Universal Approximator

The output neuron is a **weighted combination of step functions** (assuming bias for that layer is 0)



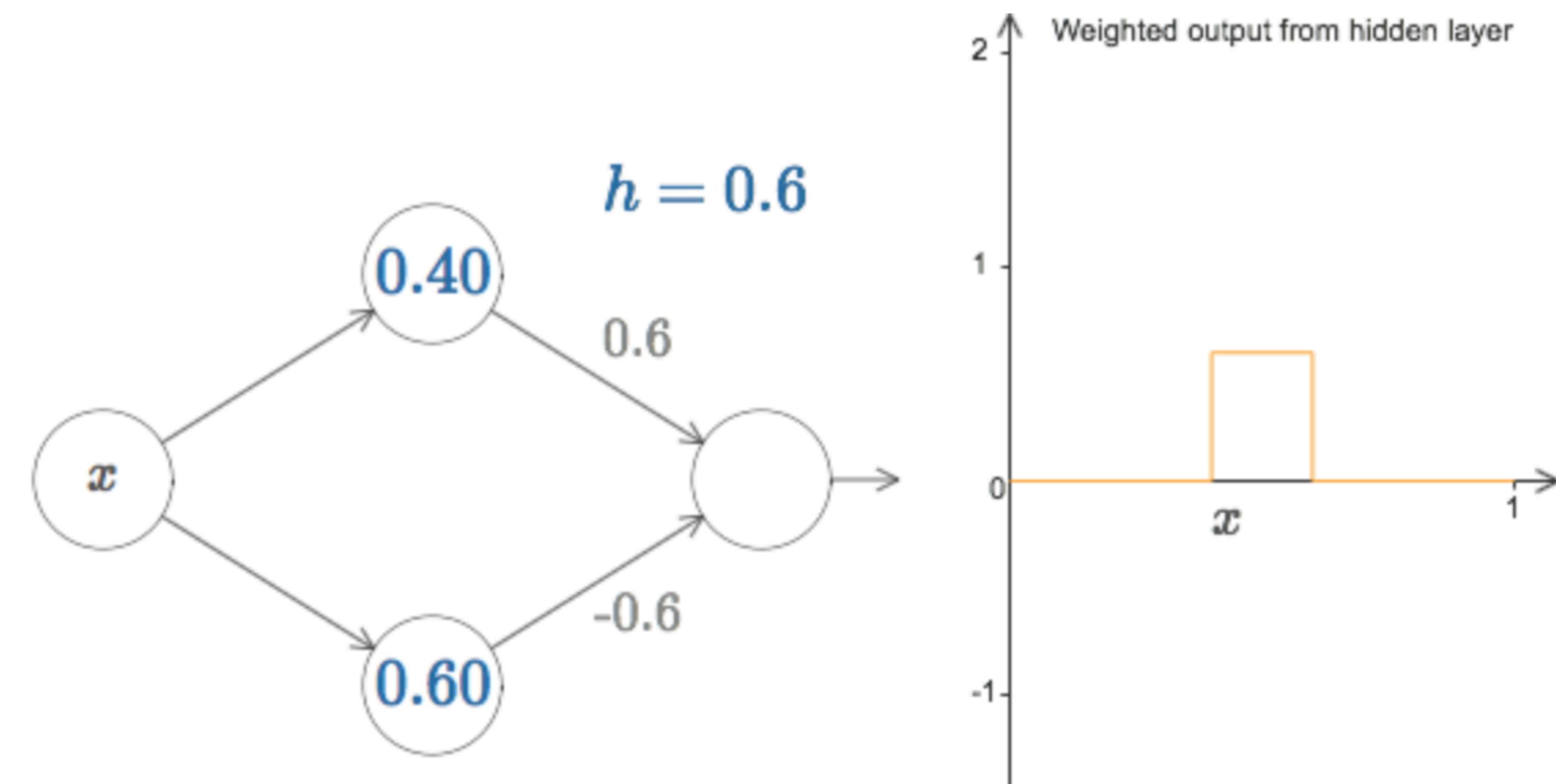
Light Theory: Neural Network as Universal Approximator

The output neuron is a **weighted combination of step functions** (assuming bias for that layer is 0)

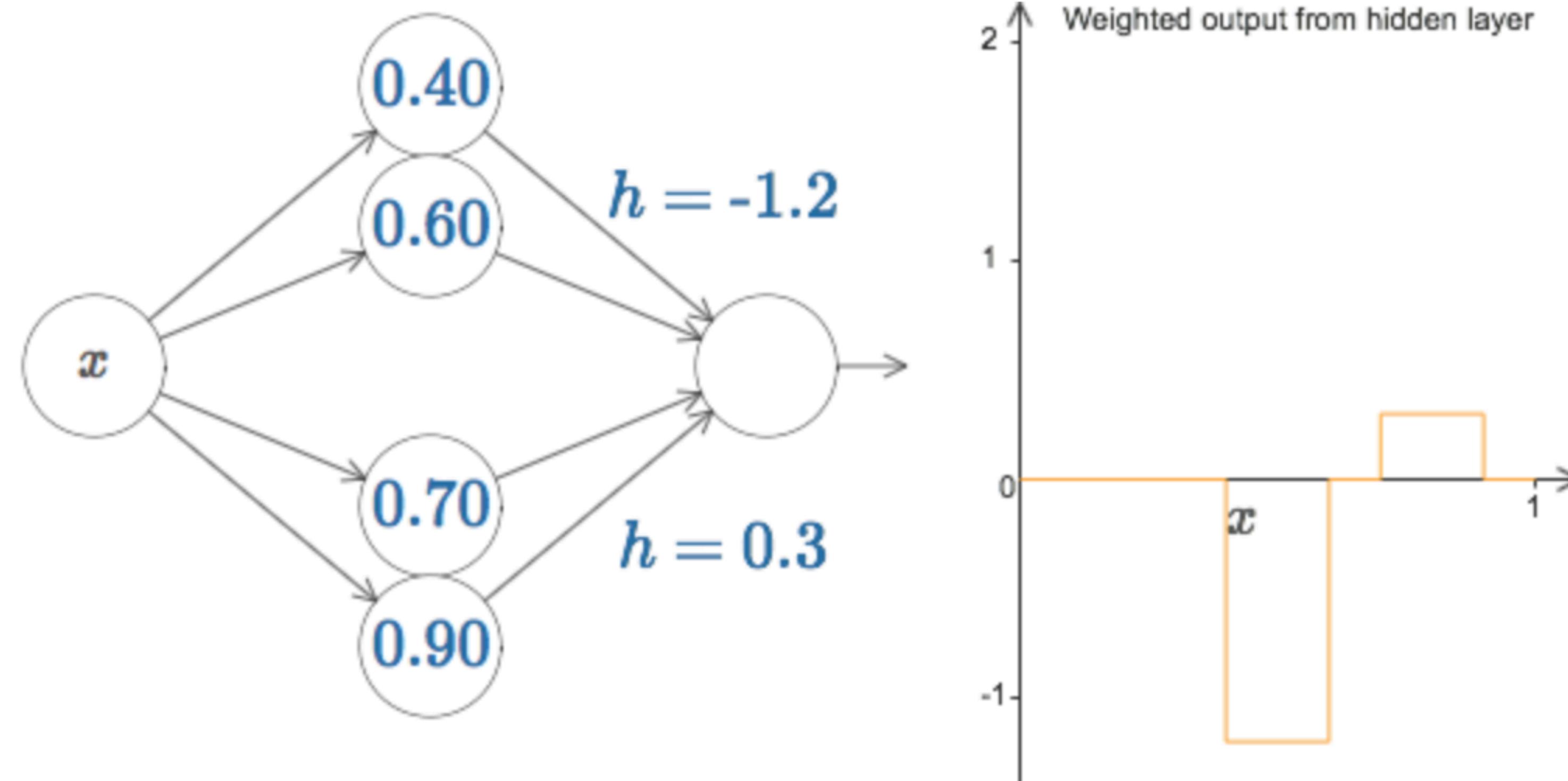


Light Theory: Neural Network as Universal Approximator

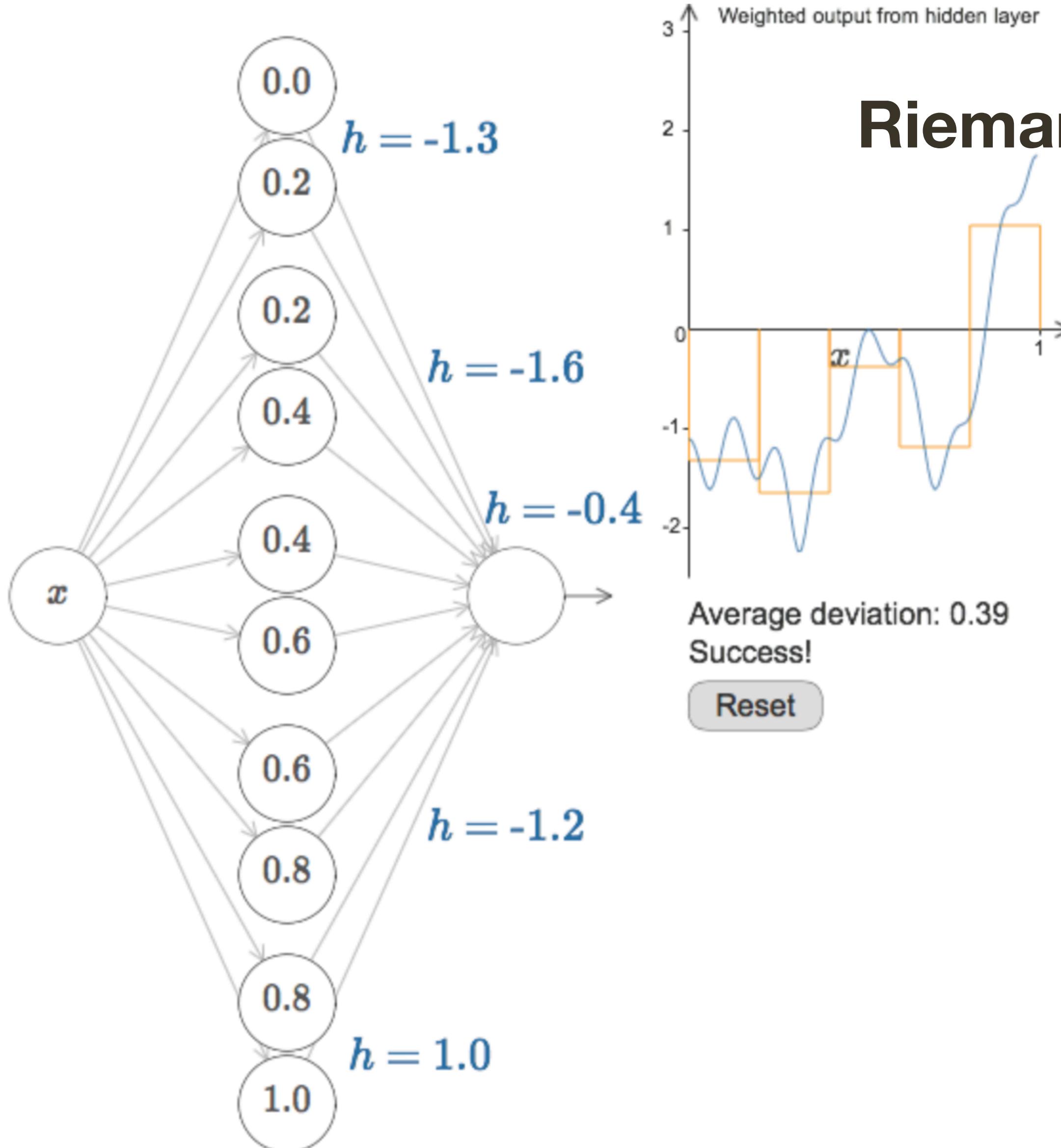
The output neuron is a **weighted combination of step functions** (assuming bias for that layer is 0)



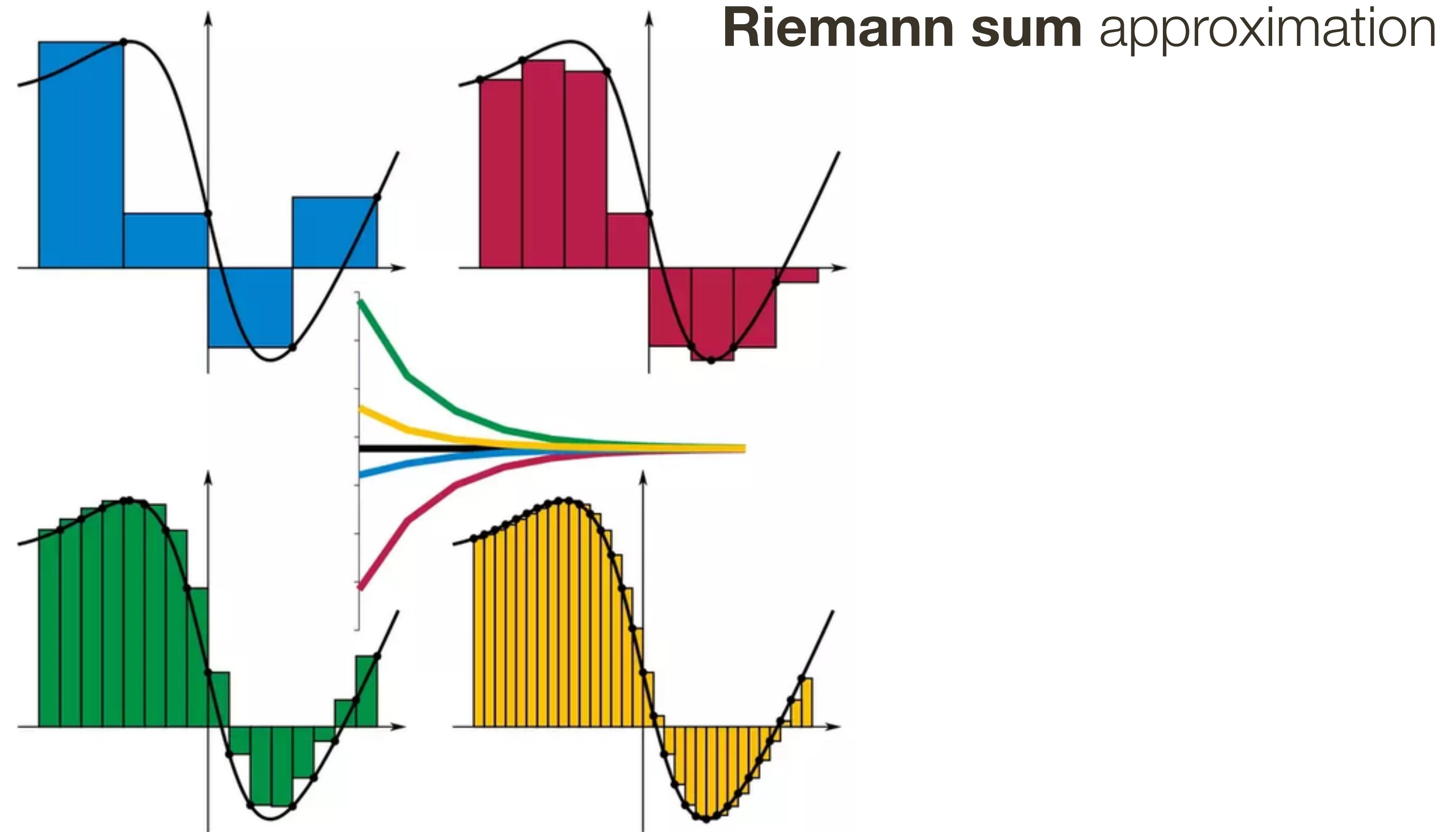
Light Theory: Neural Network as Universal Approximator



Light Theory: Neural Network as Universal Approximator



Light Theory: Neural Network as Universal Approximator



Light Theory: Neural Network as Universal Approximator

Conditions needed for proof to hold: Activation function needs to be well defined

$$\lim_{x \rightarrow \infty} a(x) = A$$

$$\lim_{x \rightarrow -\infty} a(x) = B$$

$$A \neq B$$

Light Theory: Neural Network as Universal Approximator

Conditions needed for proof to hold: Activation function needs to be well defined

$$\lim_{x \rightarrow \infty} a(x) = A$$

$$\lim_{x \rightarrow -\infty} a(x) = B$$

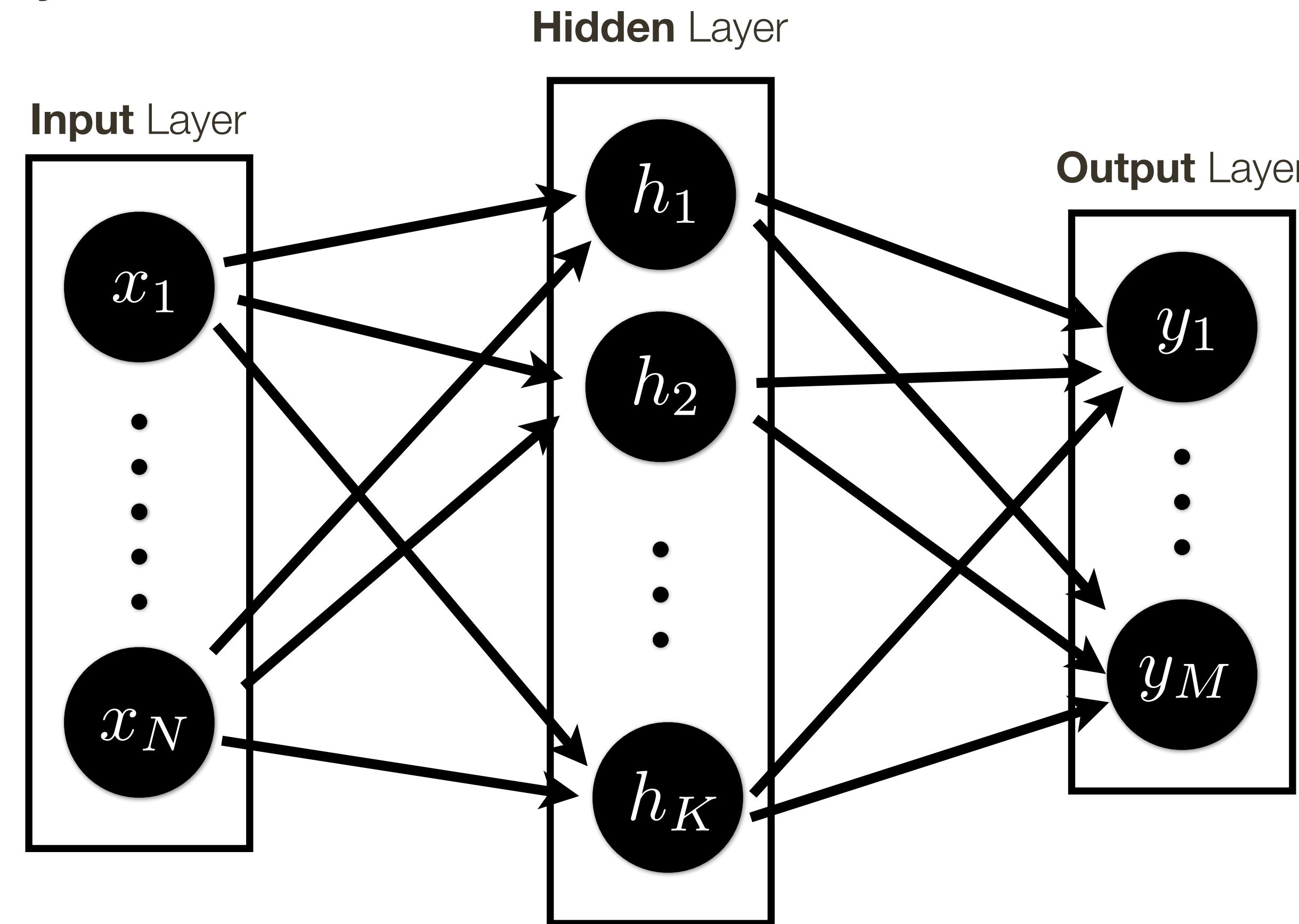
$$A \neq B$$

Note: This gives us another way to provably say that linear activation function cannot produce a neural network which is an universal approximator.

Light Theory: Neural Network as Universal Approximator

Universal Approximation Theorem: Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity.

[Hornik et al., 1989]



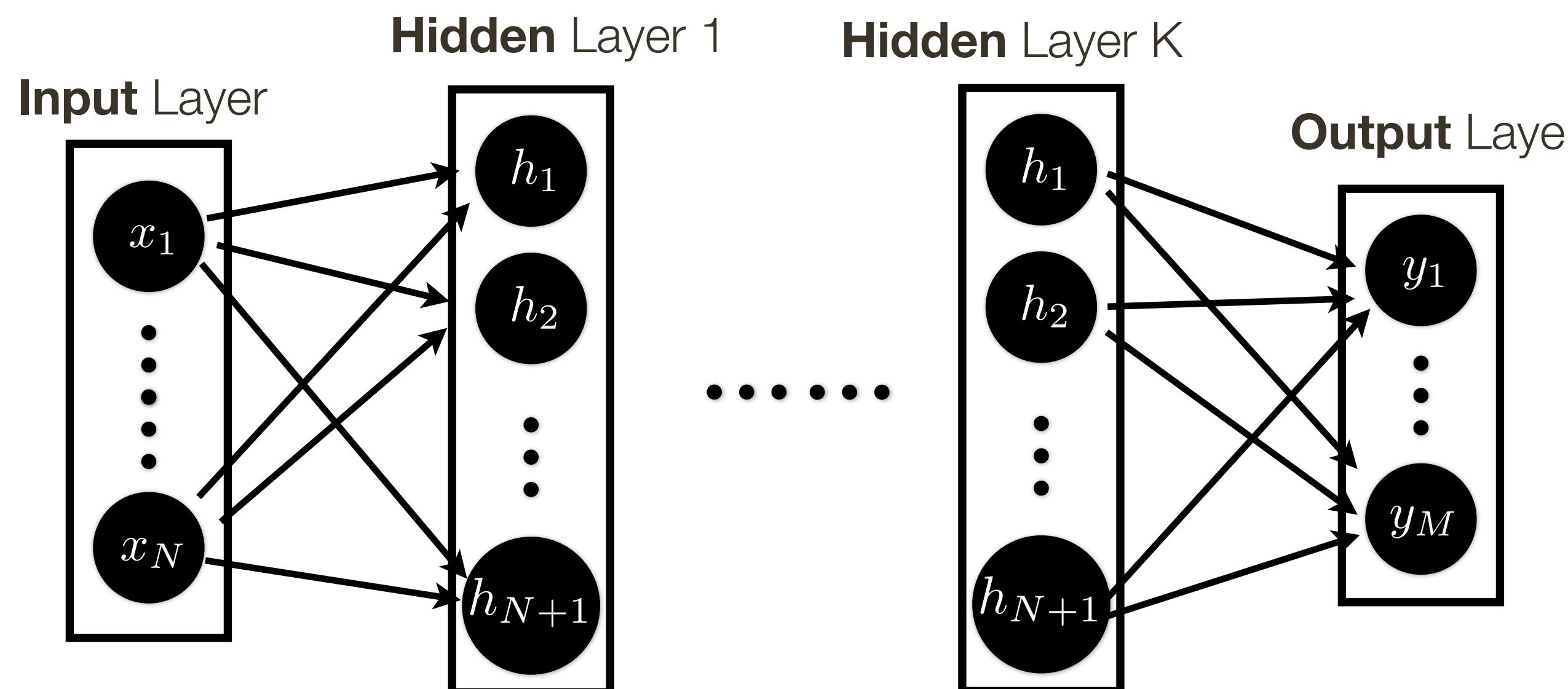
Light Theory: Neural Network as Universal Approximator

Universal Approximation Theorem: Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity.

[Hornik *et al.*, 1989]

Universal Approximation Theorem (revised): A network of infinite depth with a hidden layer of size $d + 1$ neurons, where d is the dimension of the input space, can approximate any continuous function.

[Lu *et al.*, NIPS 2017]



Light Theory: Neural Network as Universal Approximator

Universal Approximation Theorem: Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity.

[Hornik *et al.*, 1989]

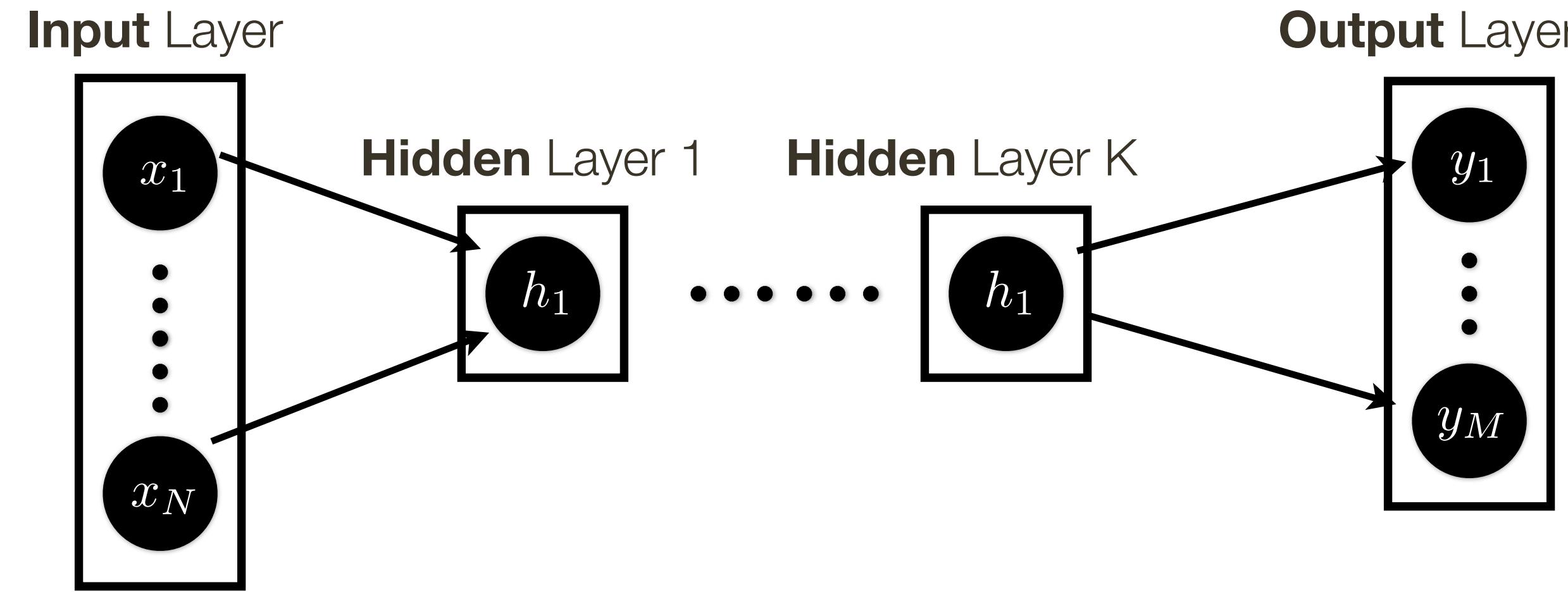
Universal Approximation Theorem (revised): A network of infinite depth with a hidden layer of size $d + 1$ neurons, where d is the dimension of the input space, can approximate any continuous function.

[Lu *et al.*, NIPS 2017]

Universal Approximation Theorem (further revised): ResNet with a single hidden unit and infinite depth can approximate any continuous function.

[Lin and Jegelka, NIPS 2018]

Light Theory: Neural Network as Universal Approximator

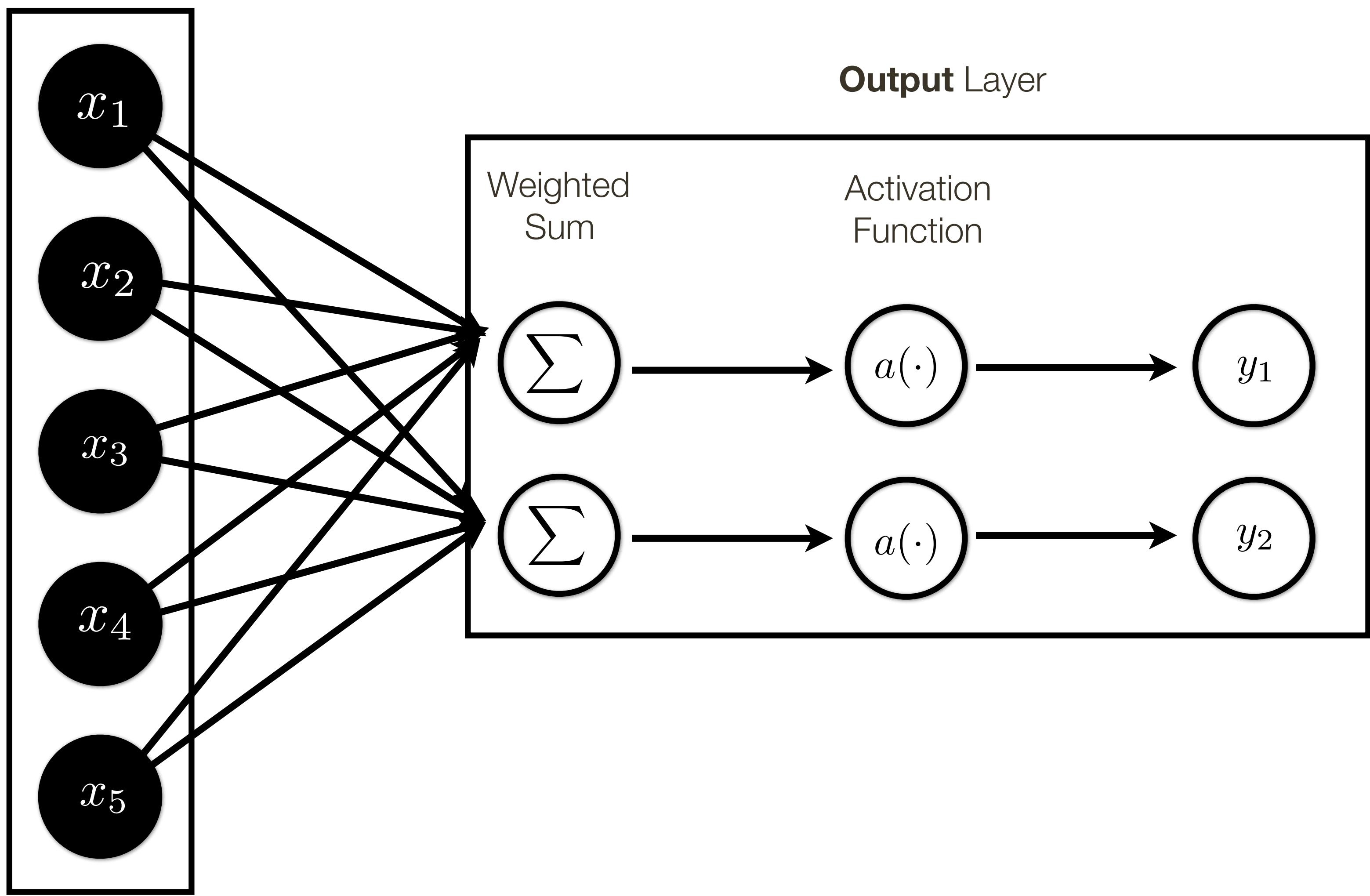


Universal Approximation Theorem (further revised): ResNet with a single hidden unit and infinite depth can approximate any continuous function.

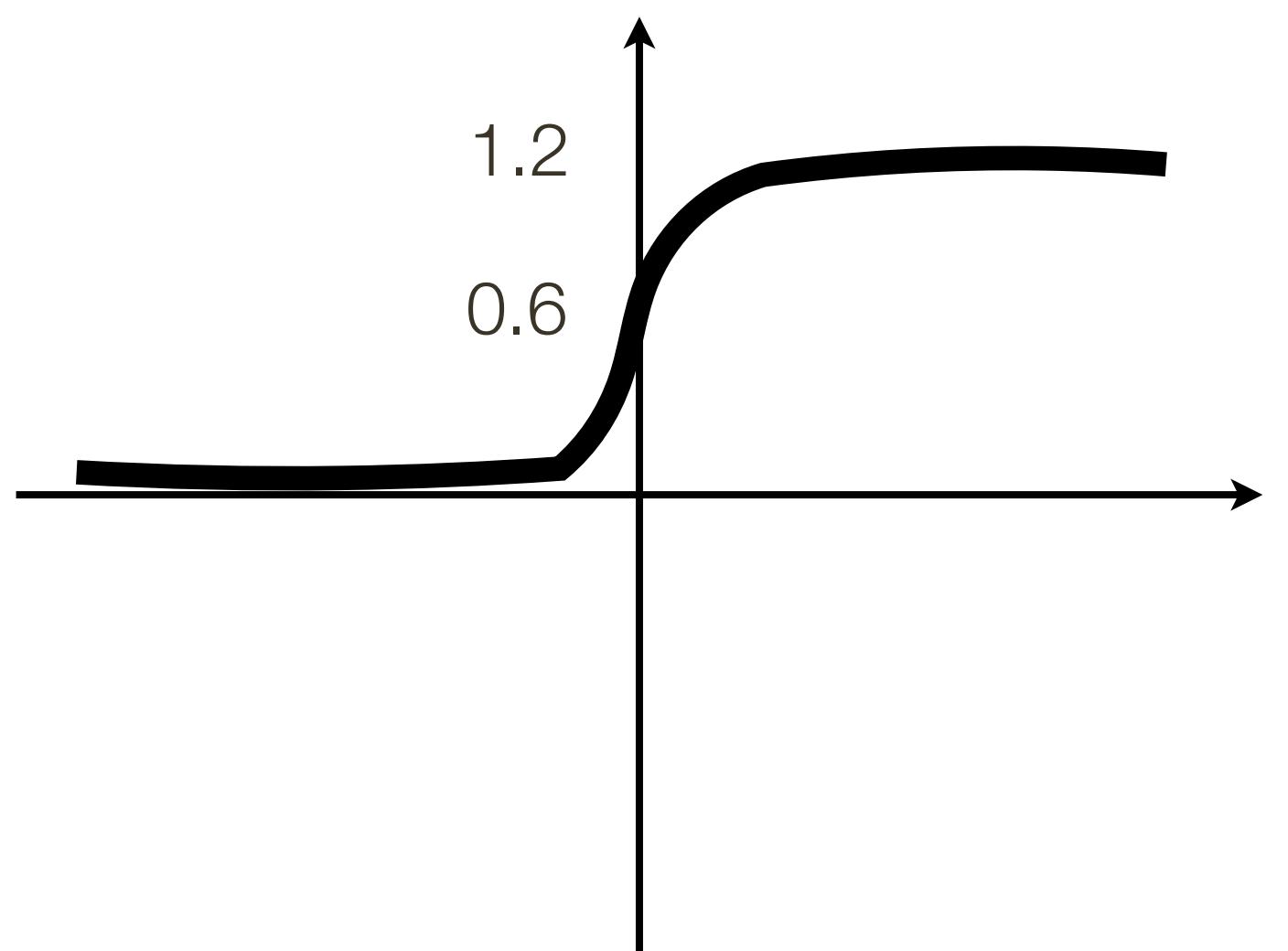
[Lin and Jegelka, NIPS 2018]

One-layer Neural Network

Input Layer



$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



Sigmoid Activation

Learning Parameters of One-layer Neural Network

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Learning Parameters of One-layer Neural Network

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Solution:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2 = 0$$

Learning Parameters of One-layer Neural Network

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

$$\mathbf{W}^*, \mathbf{b}^* = \arg \min \mathcal{L}(\mathbf{W}, \mathbf{b})$$

Solution:

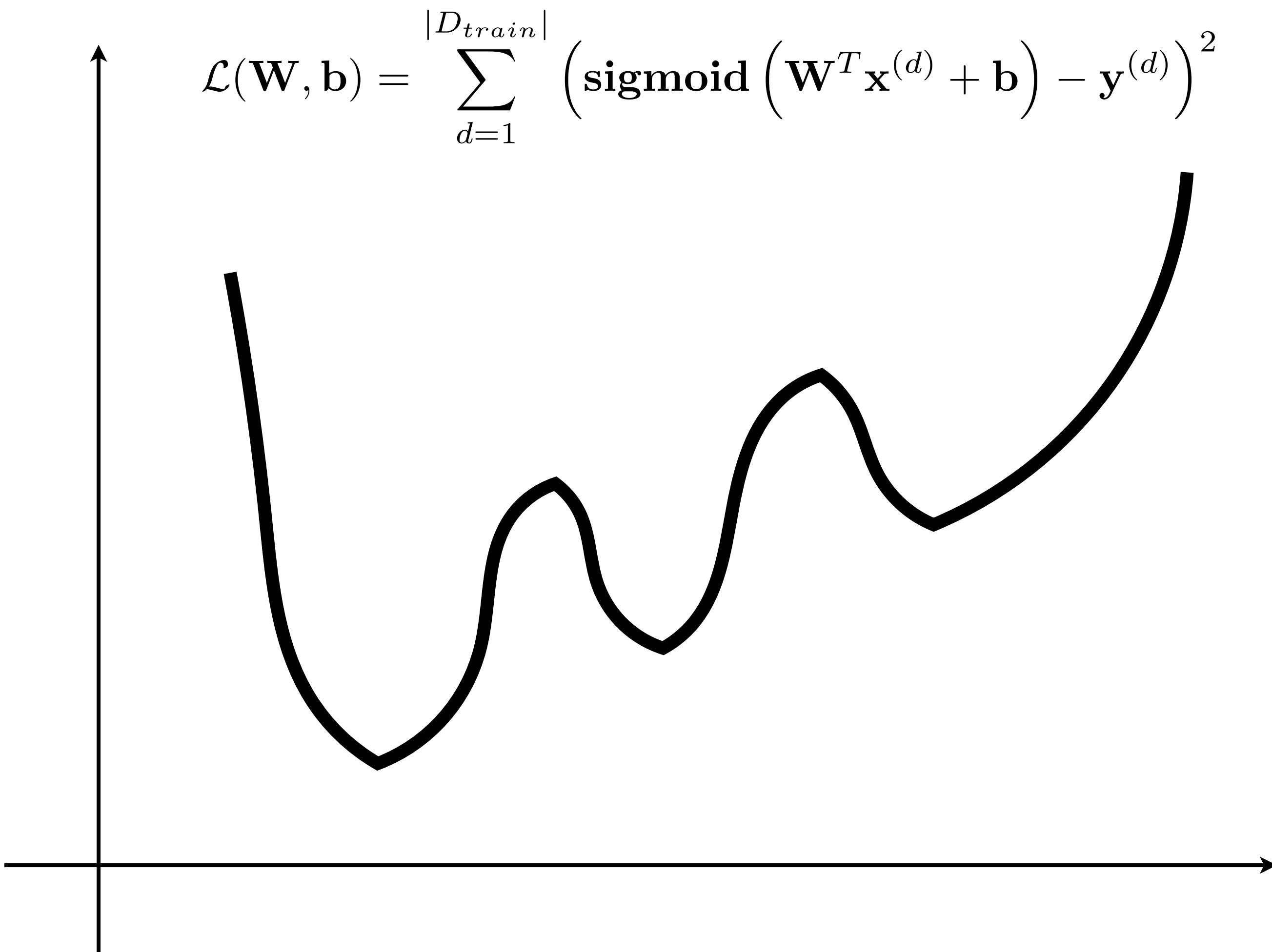
$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2 = 0$$

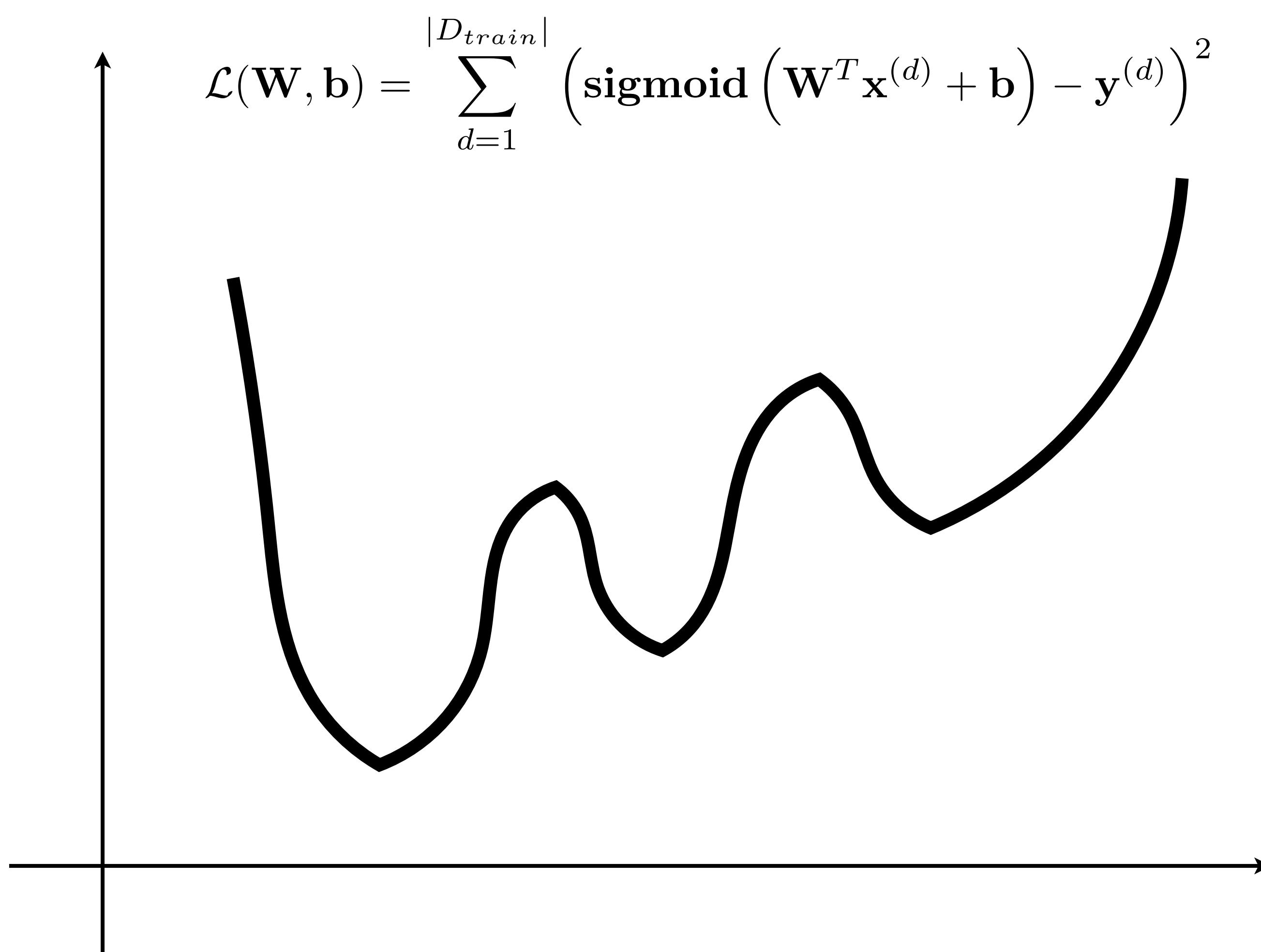
Problem: No closed form solution

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = 0$$

Gradient Descent (review)

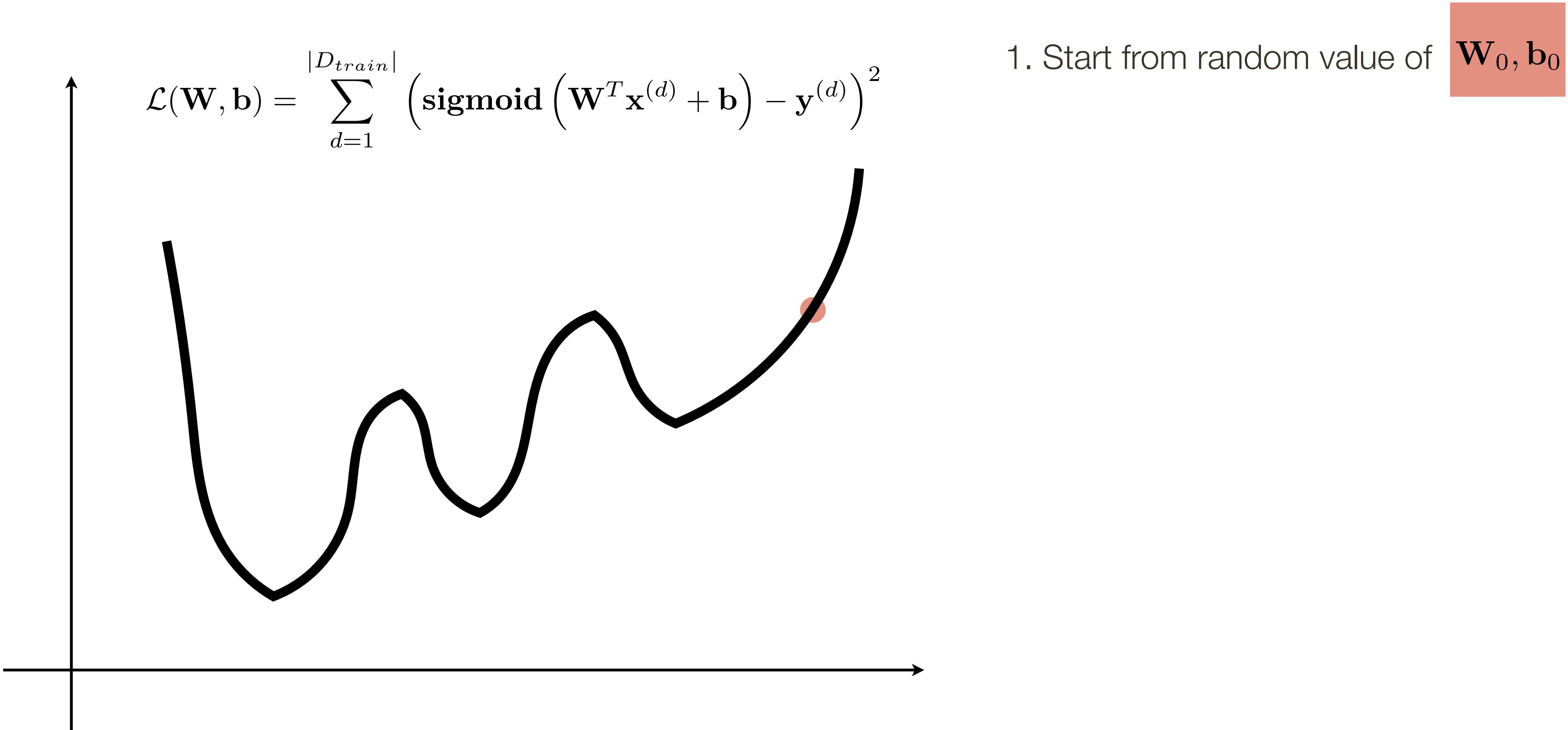


Gradient Descent (review)



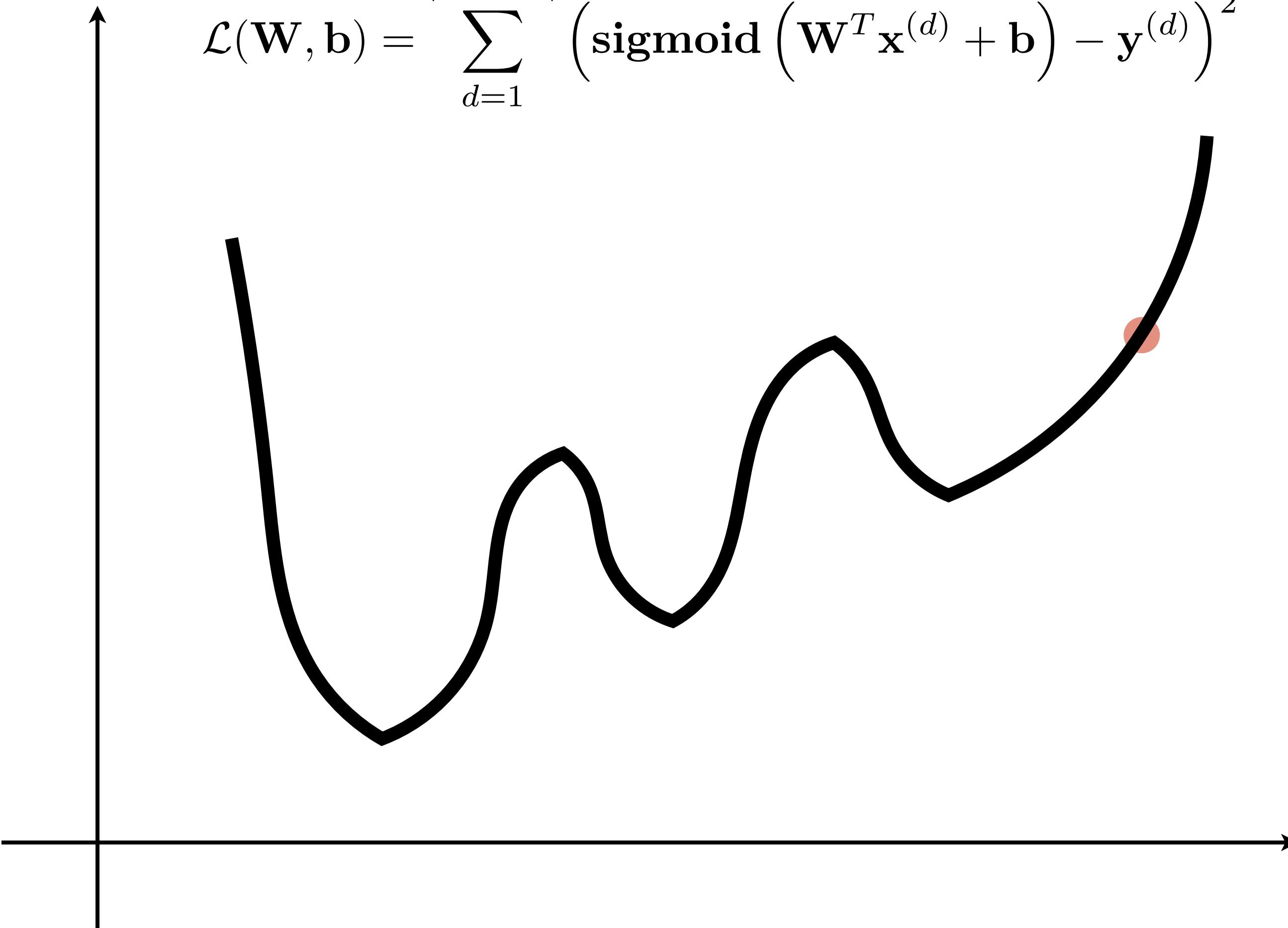
1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

Gradient Descent (review)



1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

Gradient Descent (review)



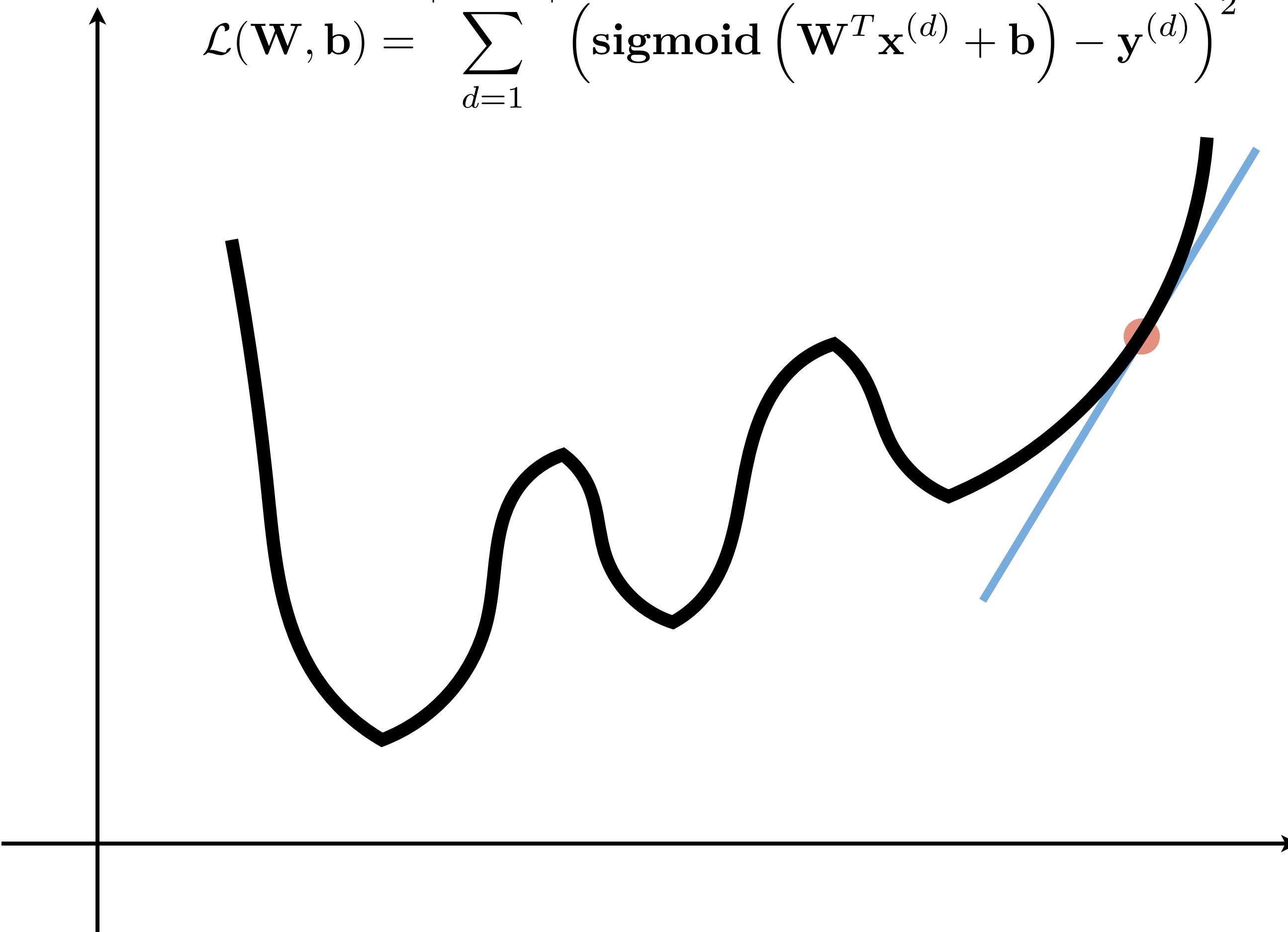
1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

For $k = 0$ to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

$$\nabla \mathcal{L}(\mathbf{W}, \mathbf{b})|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

Gradient Descent (review)



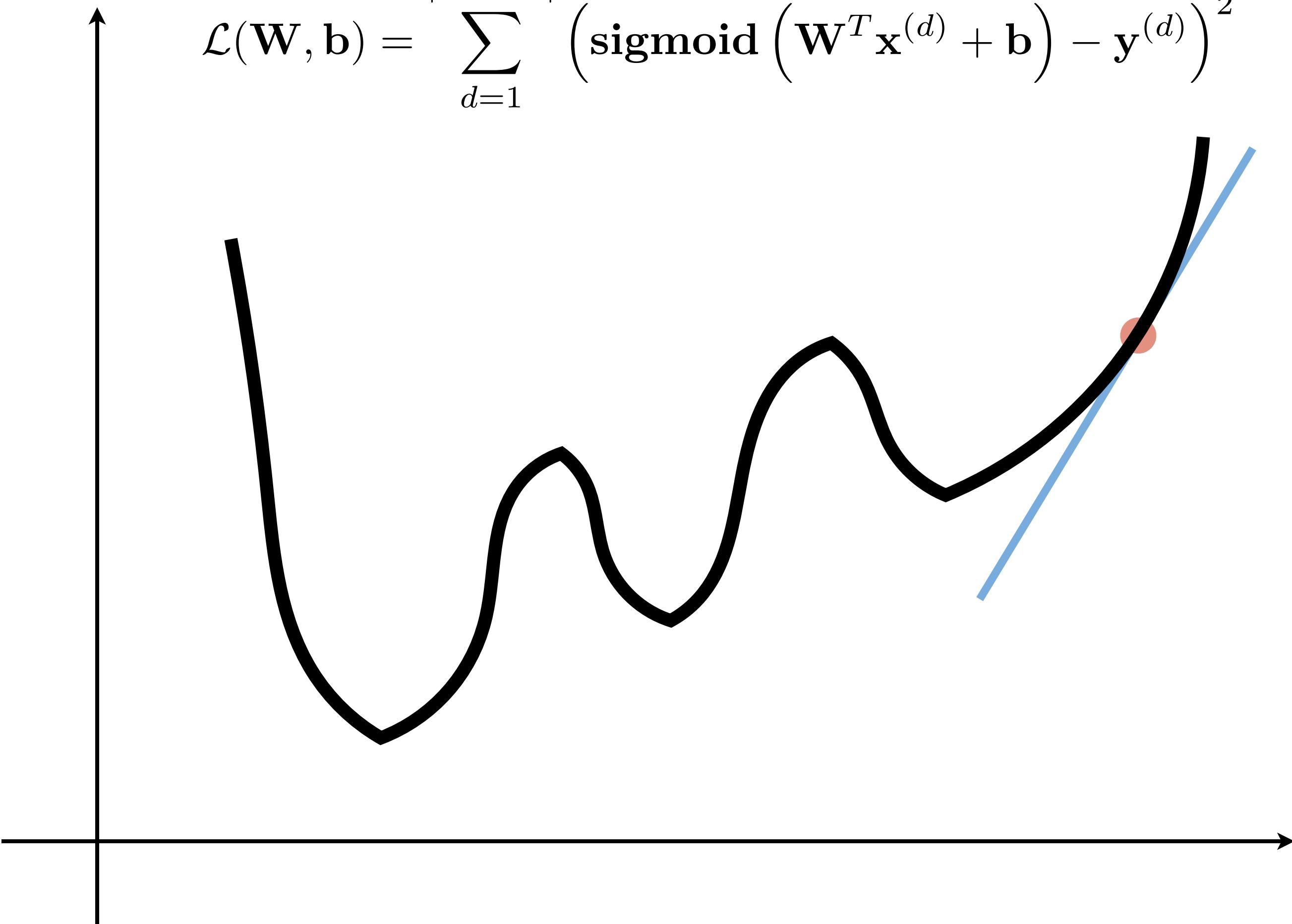
1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

For $k = 0$ to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

$$\nabla \mathcal{L}(\mathbf{W}, \mathbf{b})|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

Gradient Descent (review)



1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

For $k = 0$ to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

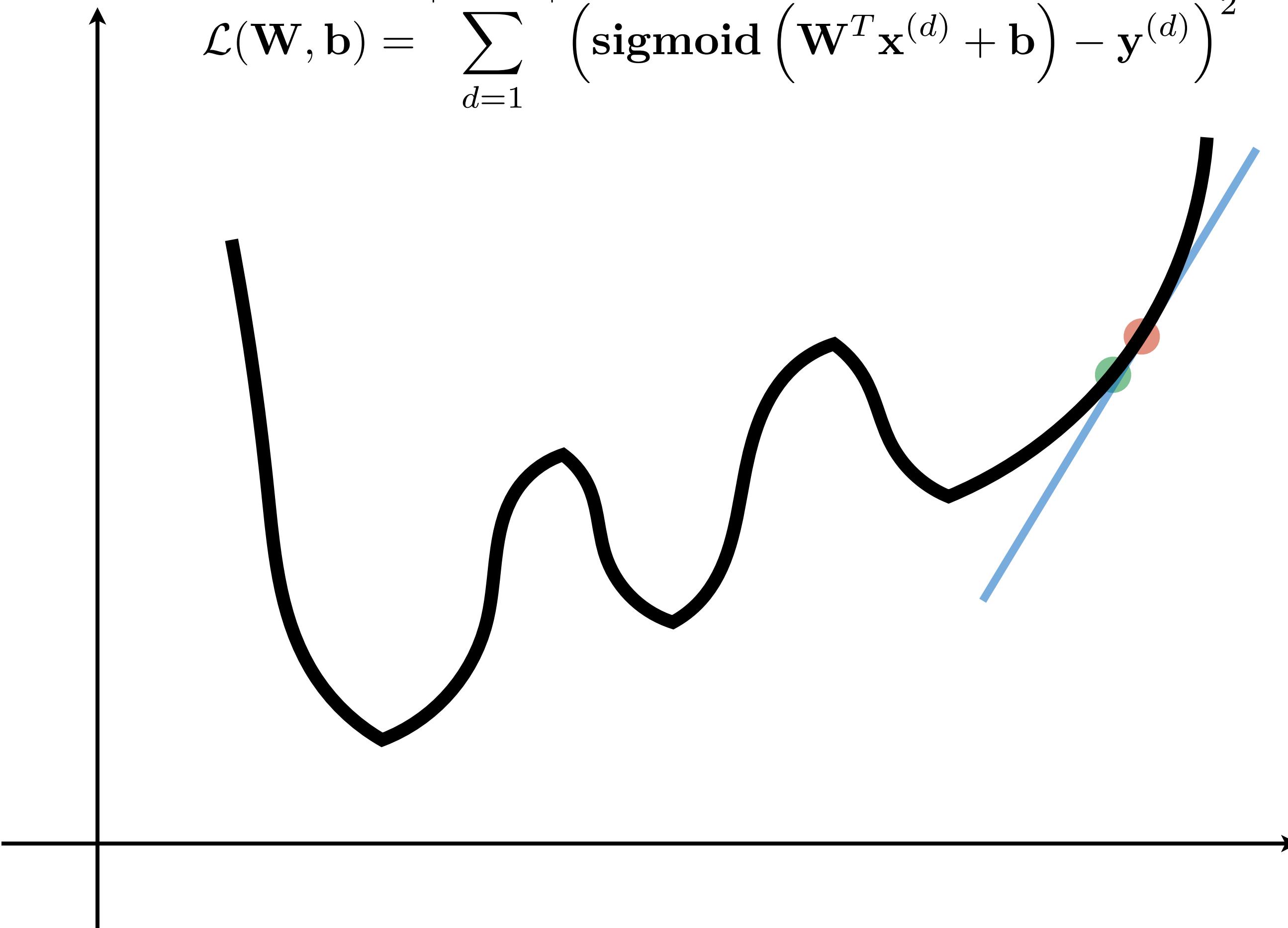
$$\nabla \mathcal{L}(\mathbf{W}, \mathbf{b})|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

3. Re-estimate the parameters

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

$$\mathbf{b}_{k+1} = \mathbf{b}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{b}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

Gradient Descent (review)



1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

For $k = 0$ to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

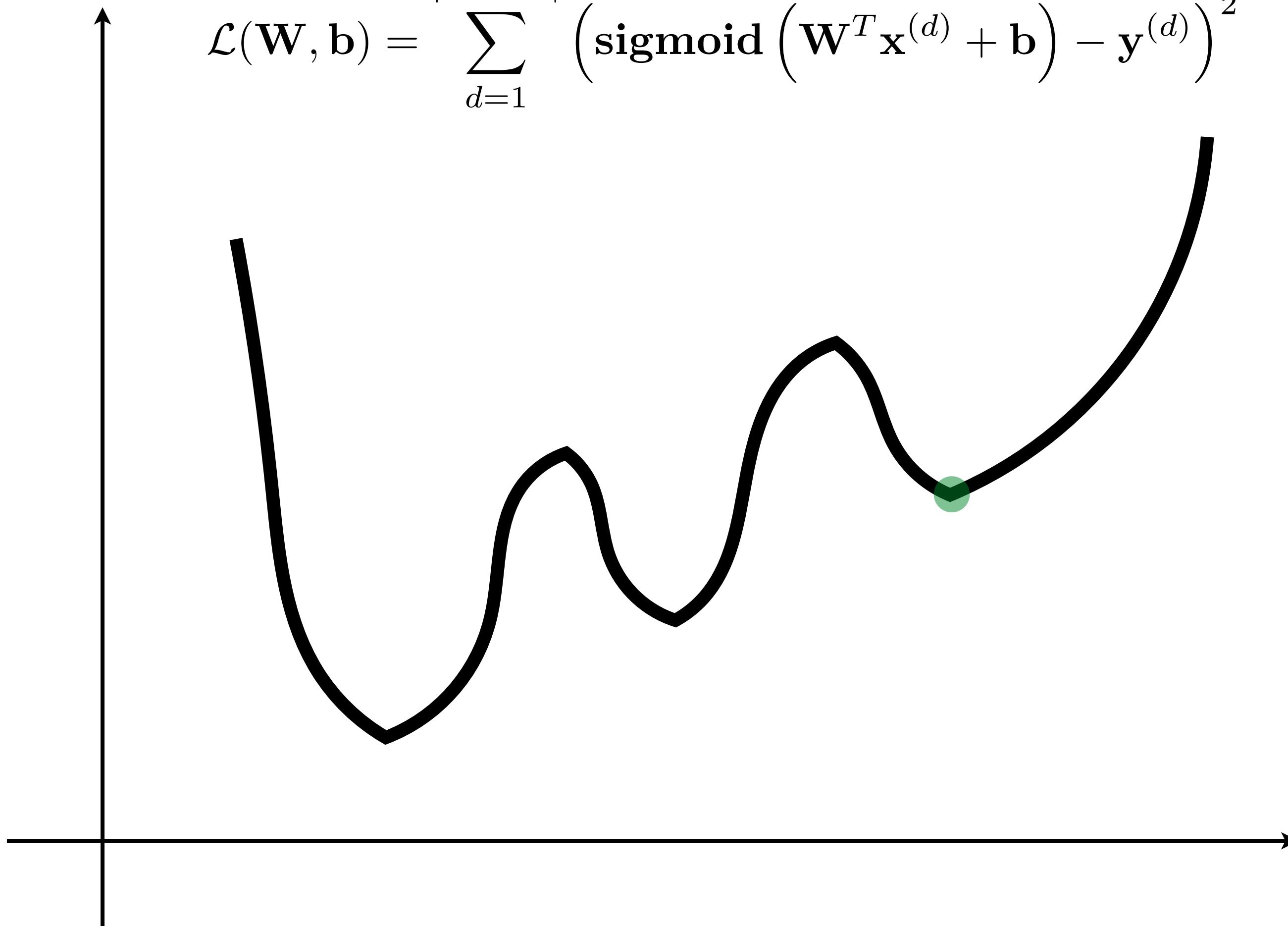
$$\nabla \mathcal{L}(\mathbf{W}, \mathbf{b})|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

3. Re-estimate the parameters

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

$$\mathbf{b}_{k+1} = \mathbf{b}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{b}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

Gradient Descent (review)



1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$

For $k = 0$ to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

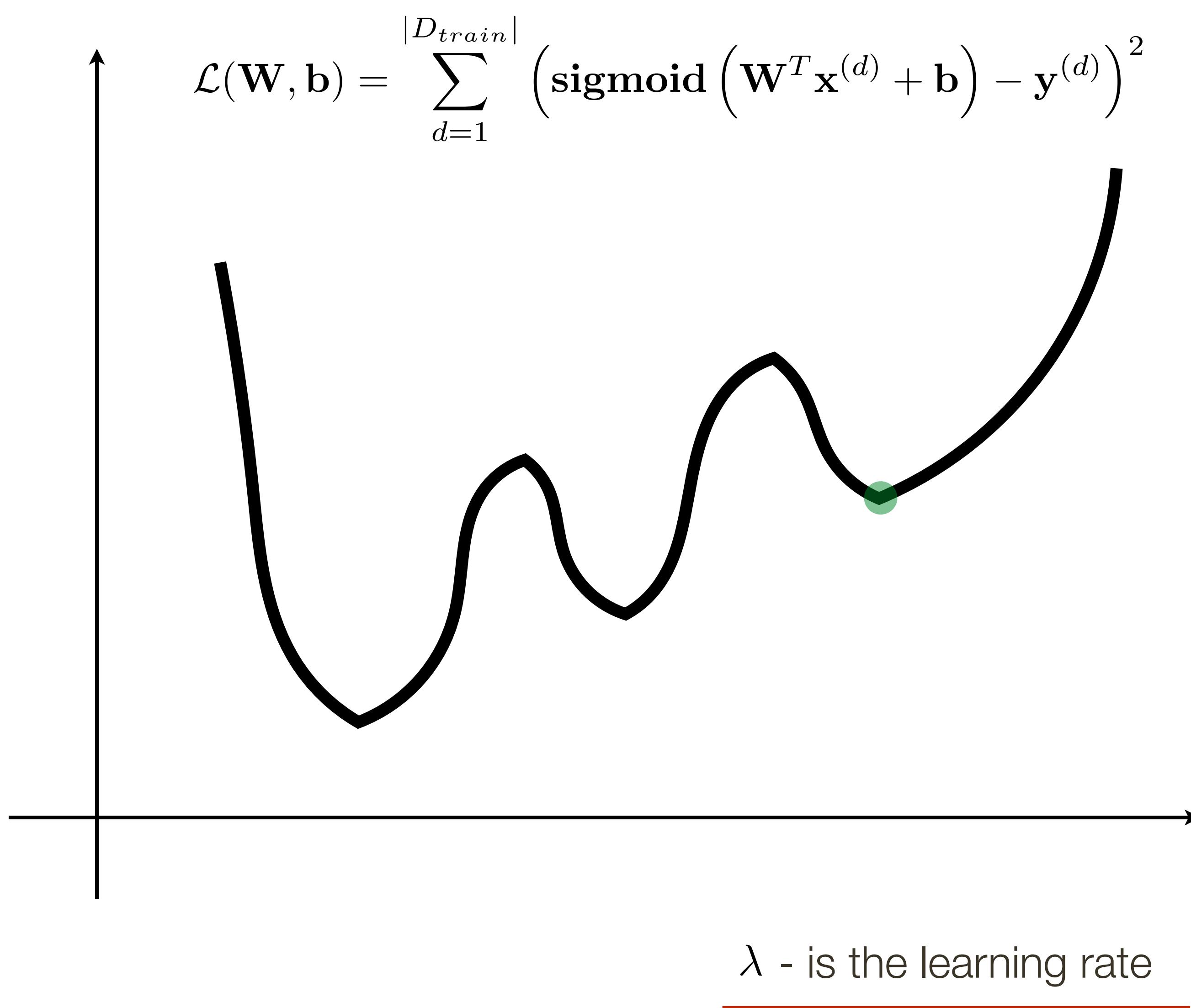
$$\nabla \mathcal{L}(\mathbf{W}, \mathbf{b})|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

3. Re-estimate the parameters

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

$$\mathbf{b}_{k+1} = \mathbf{b}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{b}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

Gradient Descent (review)



1. Start from random value of $\mathbf{W}_0, \mathbf{b}_0$
- For $k = 0$ to max number of iterations
2. Compute gradient of the loss with respect to previous (initial) parameters:
3. Re-estimate the parameters

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

$$\mathbf{b}_{k+1} = \mathbf{b}_k - \lambda \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{b}} \Big|_{\mathbf{W}=\mathbf{W}_k, \mathbf{b}=\mathbf{b}_k}$$

Stochastic Gradient Descent (review)

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

Stochastic Gradient Descent (review)

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

Problem: For large datasets computing sum is expensive

Stochastic Gradient Descent (review)

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

Problem: For large datasets computing sum is expensive

Solution: Compute approximate gradient with mini-batches of much smaller size (as little as 1-example sometimes)

Stochastic Gradient Descent (review)

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{train}|} \left(\text{sigmoid} \left(\mathbf{W}^T \mathbf{x}^{(d)} + \mathbf{b} \right) - \mathbf{y}^{(d)} \right)^2$$

Problem: For large datasets computing sum is expensive

Solution: Compute approximate gradient with mini-batches of much smaller size (as little as 1-example sometimes)

Problem: How do we compute the actual gradient?

Numerical Differentiation

$\mathbf{1}_i$ - Vector of all zeros, except for one 1 in i-th location

We can approximate the gradient numerically, using:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{1}_i) - f(\mathbf{x})}{h}$$

Numerical Differentiation

$\mathbf{1}_i$ - Vector of all zeros, except for one 1 in i-th location

We can approximate the gradient numerically, using:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{1}_i) - f(\mathbf{x})}{h}$$

Even better, we can use central differencing:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{1}_i) - f(\mathbf{x} - h\mathbf{1}_i)}{2h}$$

Numerical Differentiation

$\mathbf{1}_i$ - Vector of all zeros, except for one 1 in i-th location

We can approximate the gradient numerically, using:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{1}_i) - f(\mathbf{x})}{h}$$

Even better, we can use central differencing:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{1}_i) - f(\mathbf{x} - h\mathbf{1}_i)}{2h}$$

However, both of these suffer from rounding errors and are not good enough for learning (they are very good tools for checking the correctness of implementation though, e.g., use $h = 0.000001$).

Numerical Differentiation

$\mathbf{1}_i$ - Vector of all zeros, except for one 1 in i-th location

$\mathbf{1}_{ij}$ - Matrix of all zeros, except for one 1 in (i,j)-th location

We can approximate the gradient numerically, using:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ij}} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W} + h\mathbf{1}_{ij}, \mathbf{b}) - \mathcal{L}(\mathbf{W}, \mathbf{b})}{h}$$

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial b_j} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W}, \mathbf{b} + h\mathbf{1}_j) - \mathcal{L}(\mathbf{W}, \mathbf{b})}{h}$$

Even better, we can use central differencing:

$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial w_{ij}} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W} + h\mathbf{1}_{ij}, \mathbf{b}) - \mathcal{L}(\mathbf{W} - h\mathbf{1}_{ij}, \mathbf{b})}{2h}$$

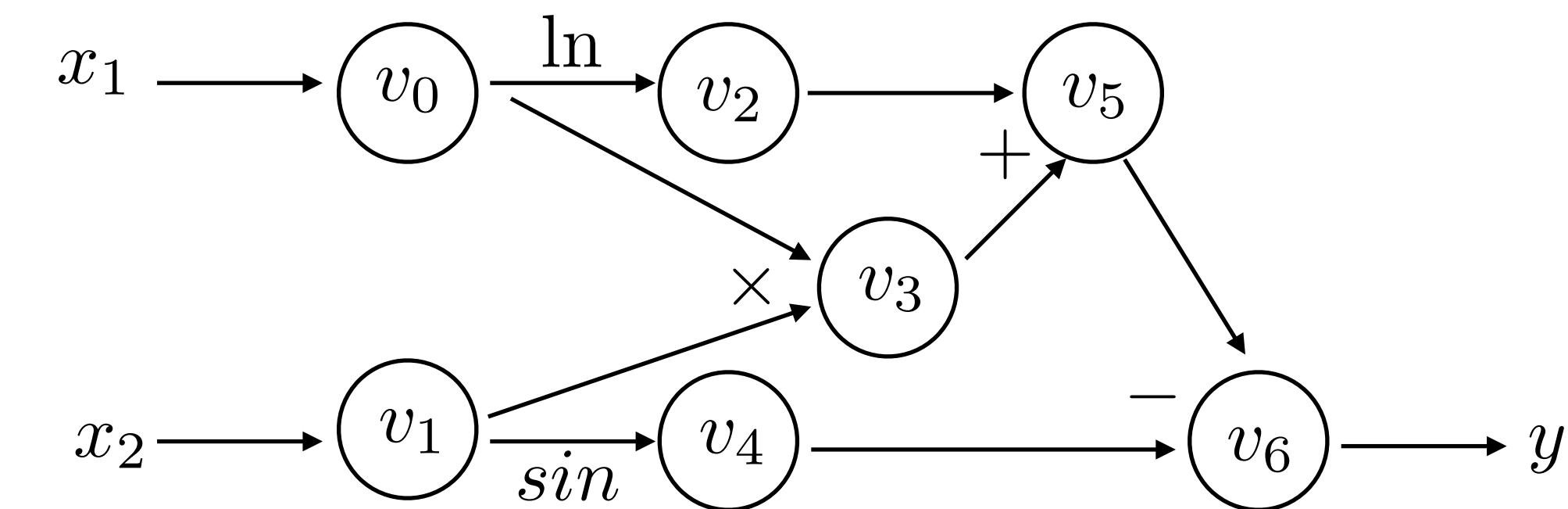
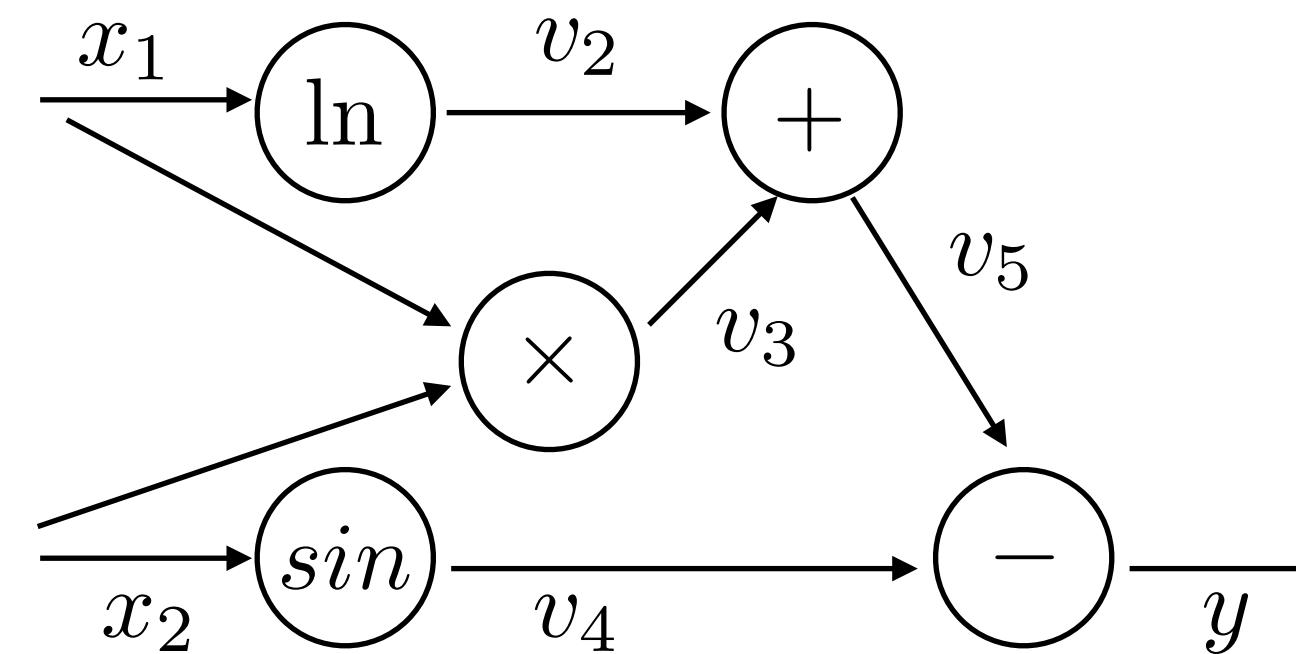
$$\frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial b_j} \approx \lim_{h \rightarrow 0} \frac{\mathcal{L}(\mathbf{W}, \mathbf{b} + h\mathbf{1}_j) - \mathcal{L}(\mathbf{W}, \mathbf{b} - h\mathbf{1}_j)}{2h}$$

However, both of these suffer from rounding errors and are not good enough for learning (they are very good tools for checking the correctness of implementation though, e.g., use $h = 0.000001$).

Symbolic Differentiation

$$y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$$

Input function is represented as **computational graph** (a symbolic tree)



Implements differentiation rules for composite functions:

Sum Rule

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

Product Rule

$$\frac{d(f(x) \cdot g(x))}{dx} = \frac{df(x)}{dx}g(x) + f(x)\frac{dg(x)}{dx}$$

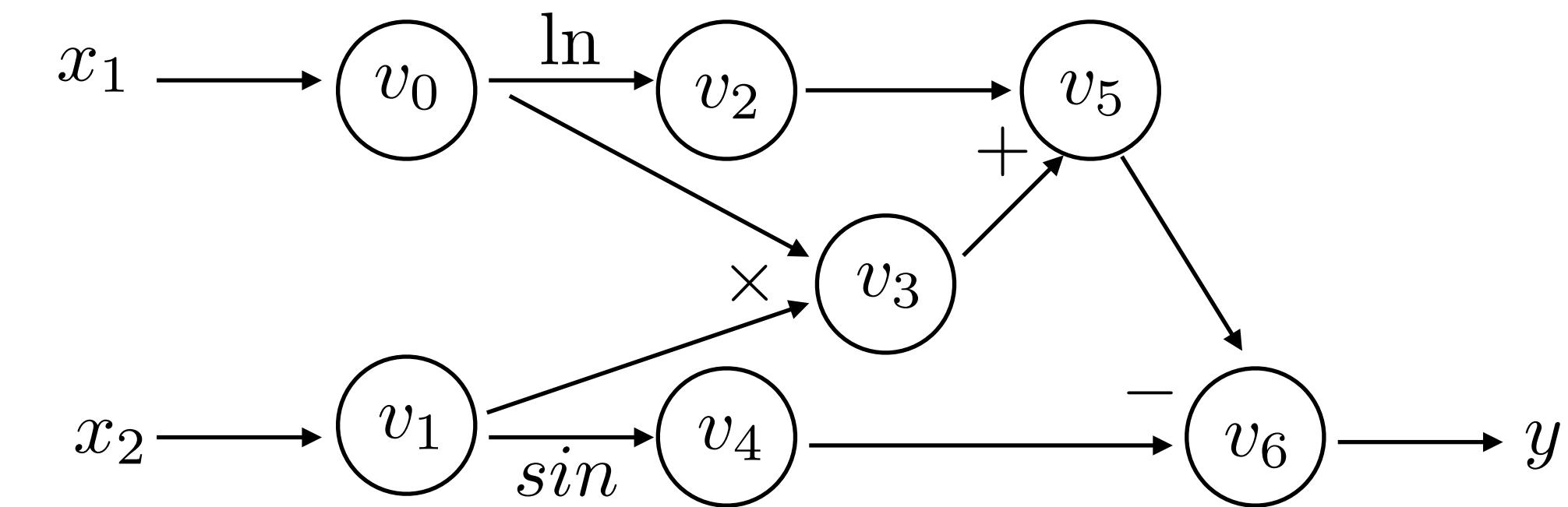
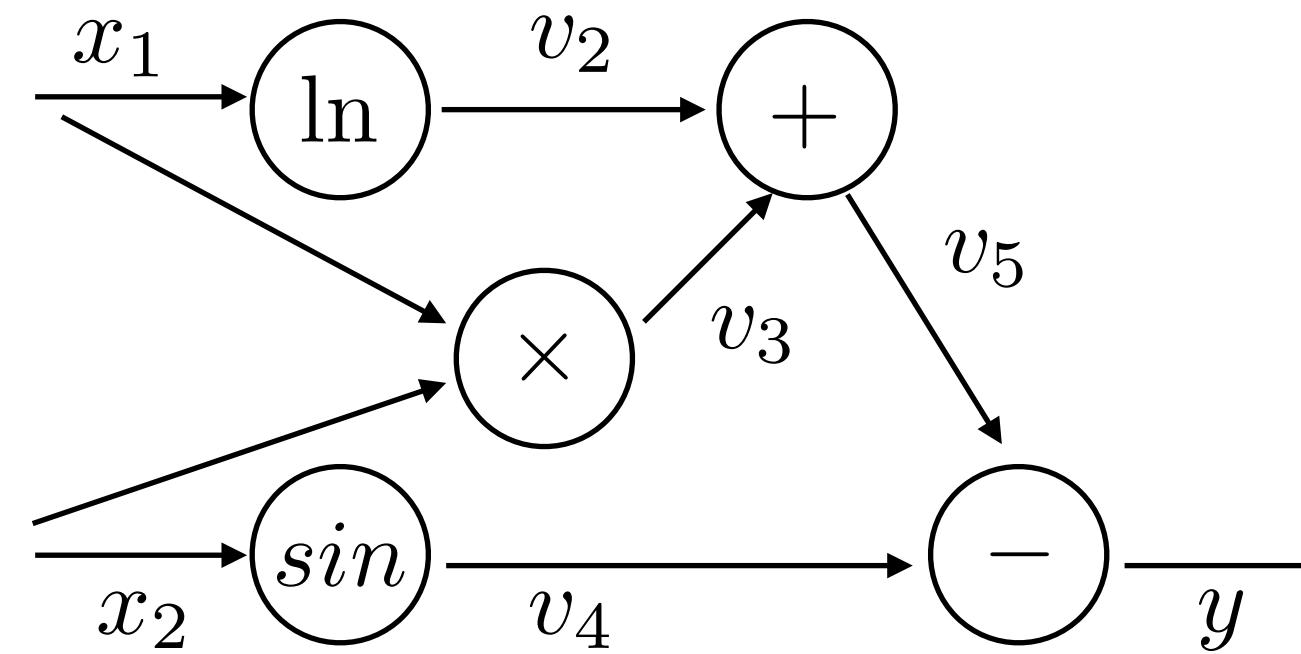
Chain Rule

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

Symbolic Differentiation

$$y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$$

Input function is represented as **computational graph** (a symbolic tree)



Implements differentiation rules for composite functions:

Sum Rule

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

Product Rule

$$\frac{d(f(x) \cdot g(x))}{dx} = \frac{df(x)}{dx}g(x) + f(x)\frac{dg(x)}{dx}$$

Chain Rule

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

Problem: For complex functions, expressions can be exponentially large; also difficult to deal with piece-wise functions (creates many symbolic cases)

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$$

Intuition: Interleave symbolic differentiation and simplification

Key Idea: apply symbolic differentiation at the elementary operation level, evaluate and keep intermediate results

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$$

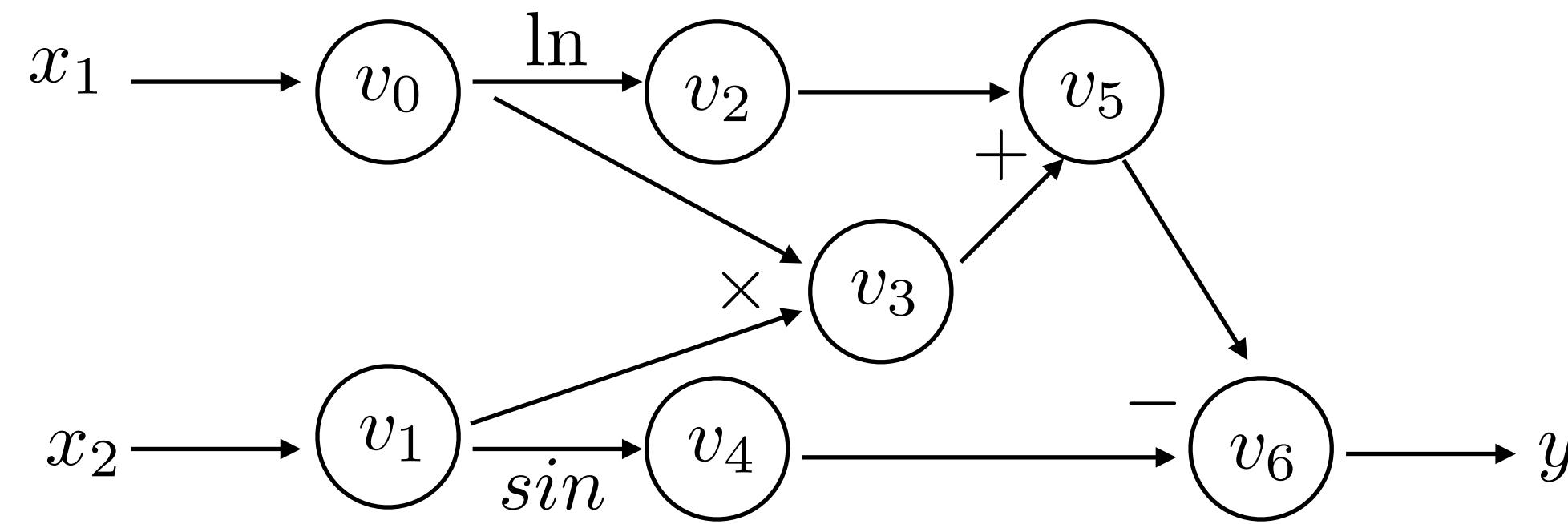
Intuition: Interleave symbolic differentiation and simplification

Key Idea: apply symbolic differentiation at the elementary operation level, evaluate and keep intermediate results

Success of **deep learning** owes A LOT to success of AutoDiff algorithms
(also to advances in parallel architectures, and large datasets, ...)

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

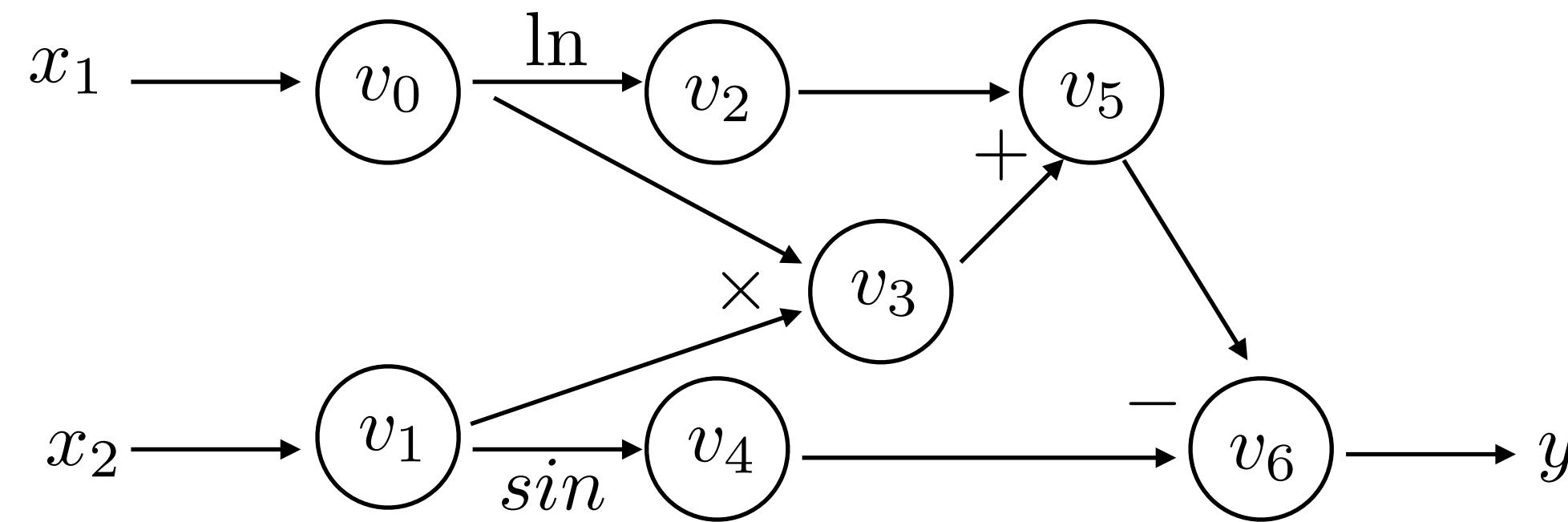


Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Computational graph is governed by these equations

Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

$$v_0 = x_1$$

$$v_1 = x_2$$

$$v_2 = \ln(v_0)$$

$$v_3 = v_0 \cdot v_1$$

$$v_4 = \sin(v_1)$$

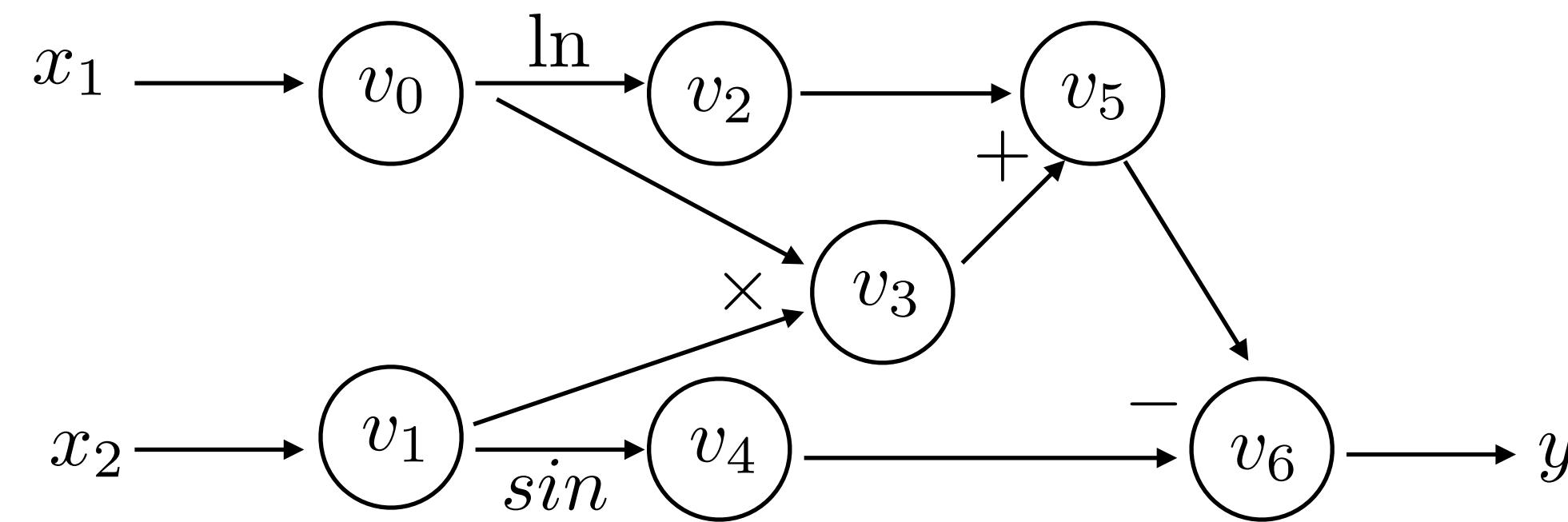
$$v_5 = v_2 + v_3$$

$$v_6 = v_5 - v_4$$

$$y = v_6$$

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Computational graph is governed by these equations

Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

$$v_0 = x_1$$

$$v_1 = x_2$$

$$v_2 = \ln(v_0)$$

$$v_3 = v_0 \cdot v_1$$

$$v_4 = \sin(v_1)$$

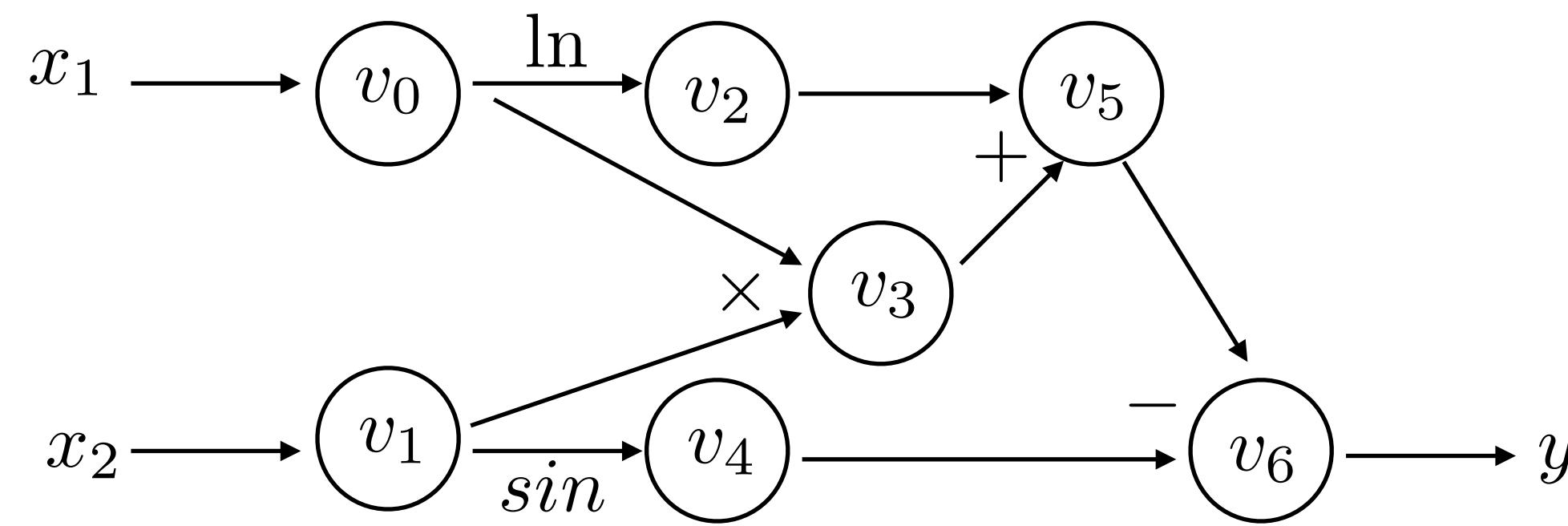
$$v_5 = v_2 + v_3$$

$$v_6 = v_5 - v_4$$

$$y = v_6$$

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

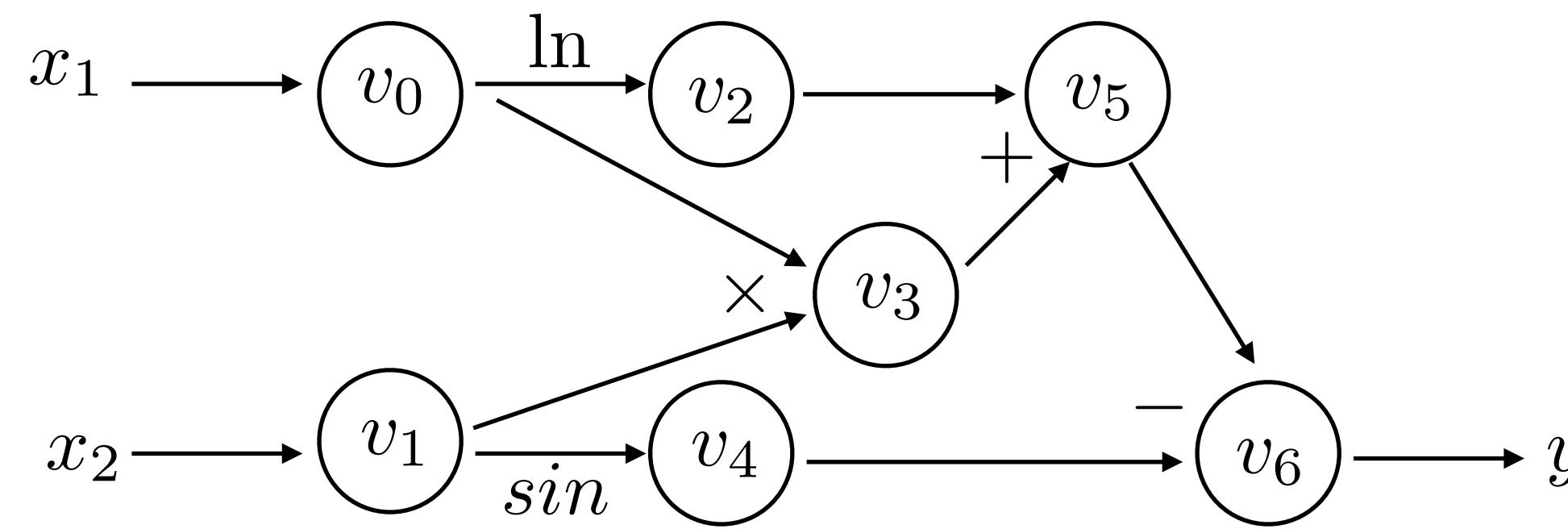
Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	
$v_1 = x_2$	
$v_2 = \ln(v_0)$	
$v_3 = v_0 \cdot v_1$	
$v_4 = \sin(v_1)$	
$v_5 = v_2 + v_3$	
$v_6 = v_5 - v_4$	
$y = v_6$	

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)$$



Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

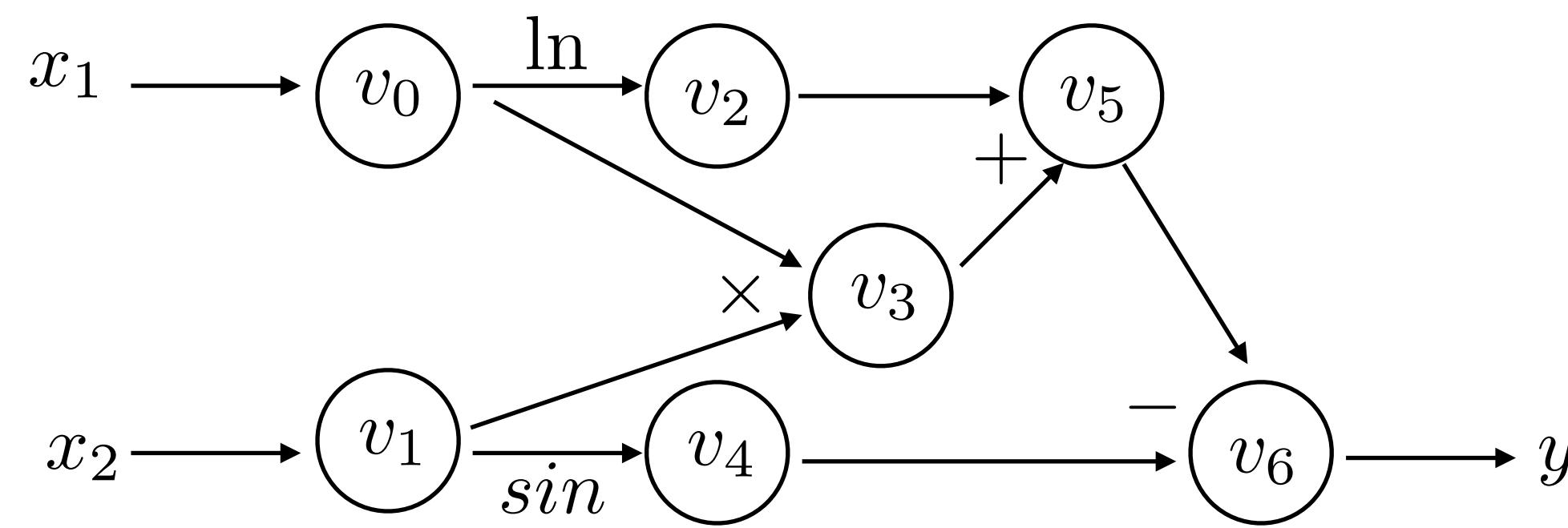
Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	
$v_2 = \ln(v_0)$	
$v_3 = v_0 \cdot v_1$	
$v_4 = \sin(v_1)$	
$v_5 = v_2 + v_3$	
$v_6 = v_5 - v_4$	
$y = v_6$	

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

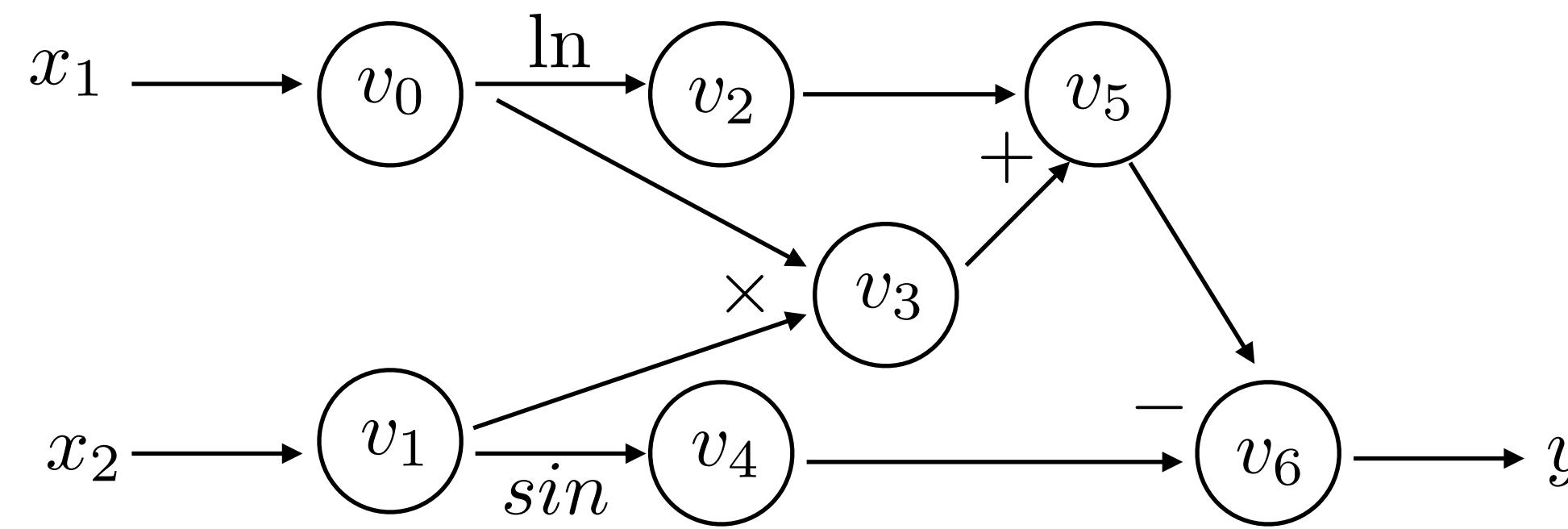
Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	
$v_3 = v_0 \cdot v_1$	
$v_4 = \sin(v_1)$	
$v_5 = v_2 + v_3$	
$v_6 = v_5 - v_4$	
$y = v_6$	

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

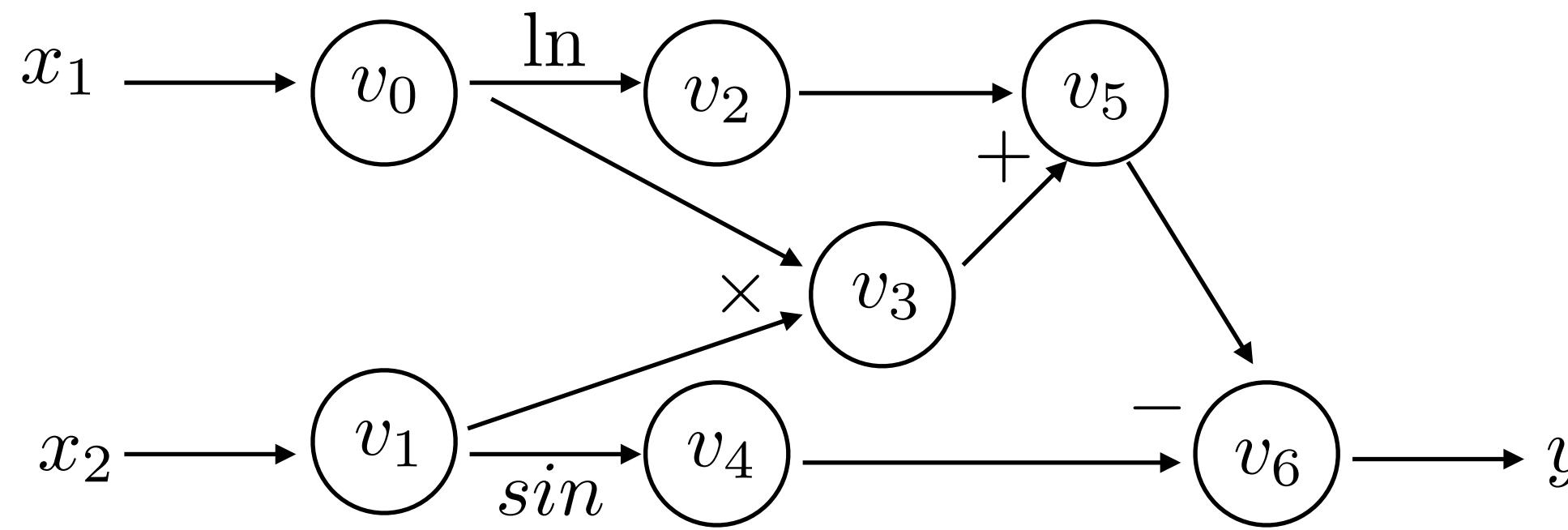
Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	
$v_4 = \sin(v_1)$	
$v_5 = v_2 + v_3$	
$v_6 = v_5 - v_4$	
$y = v_6$	

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

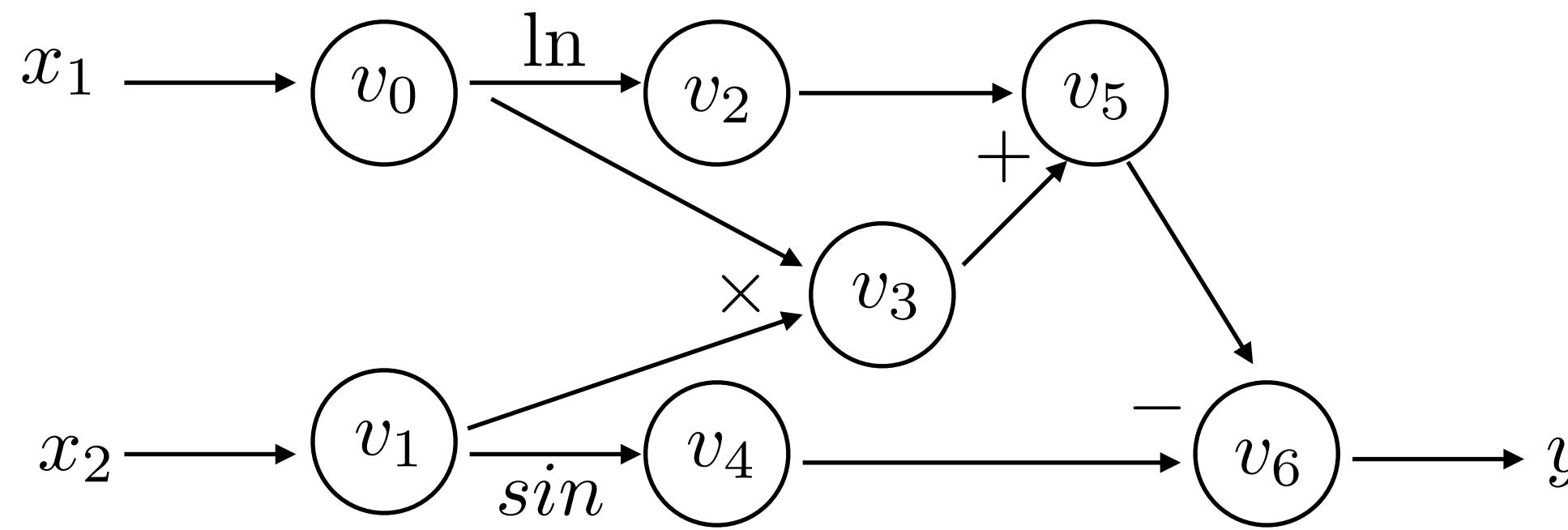
Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Each **node** is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

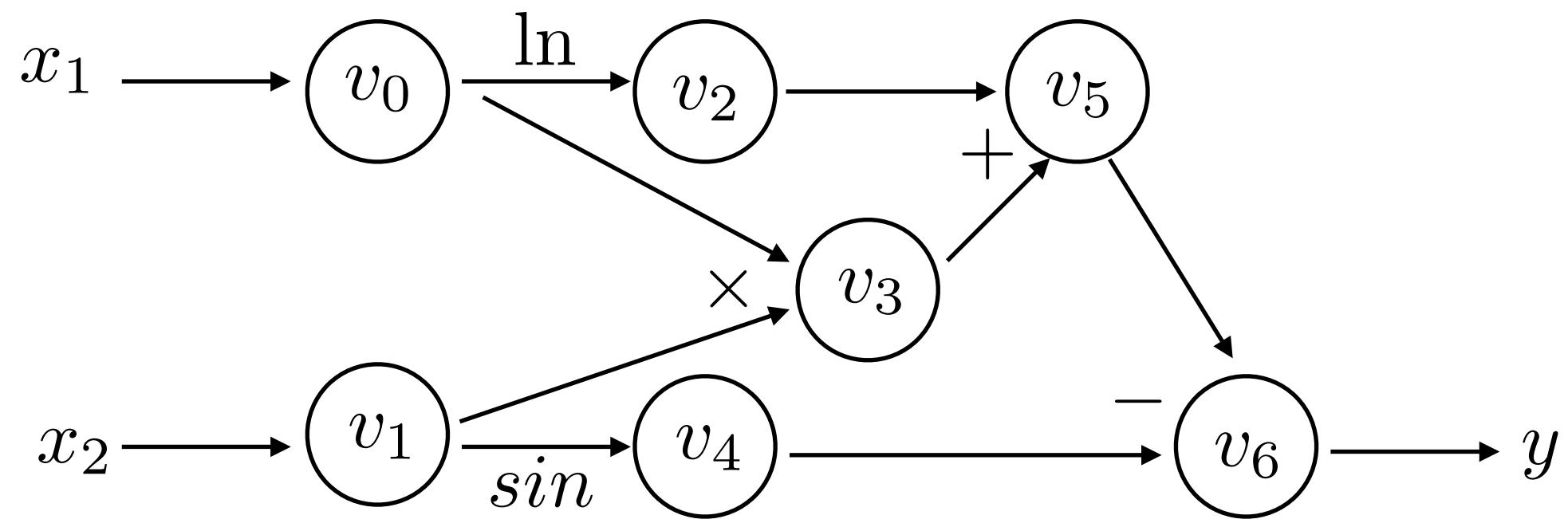
Lets see how we can **evaluate a function** using computational graph (DNN inferences)

Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

Automatic Differentiation (AutoDiff)

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



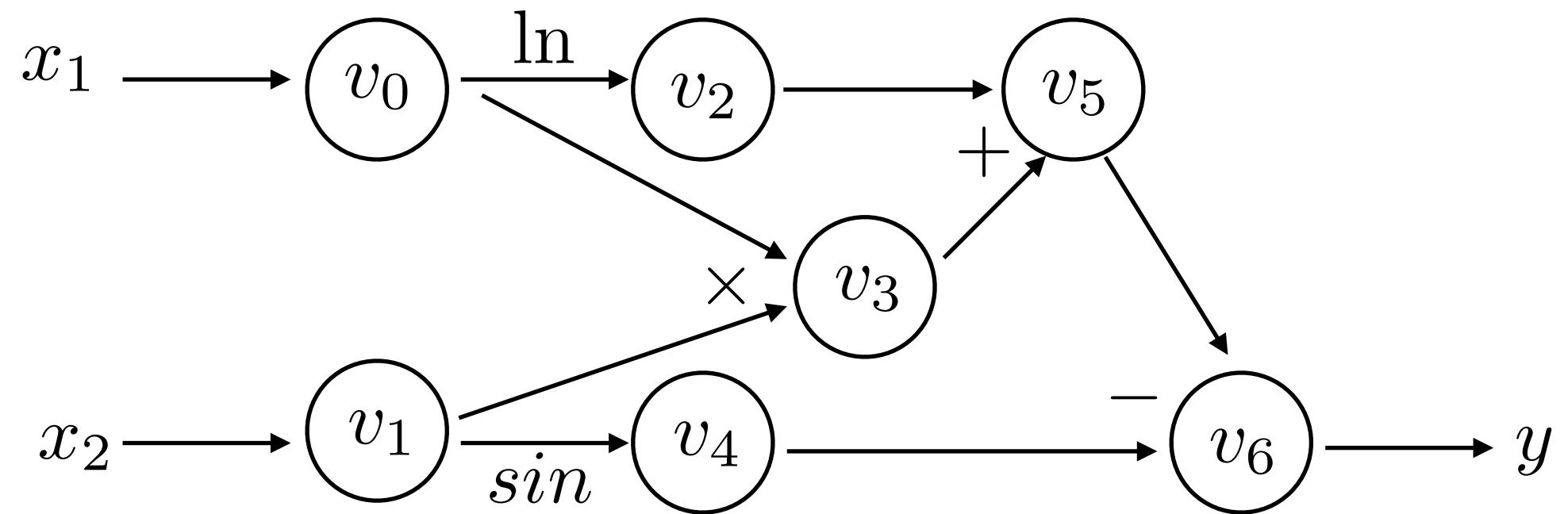
Forward Evaluation Trace:

A vertical black arrow points downwards from the computational graph to the start of the trace table.

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

AutoDiff - Forward Mode

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Forward Evaluation Trace:

A vertical black arrow points downwards from the top of the table to the bottom of the computational graph.

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

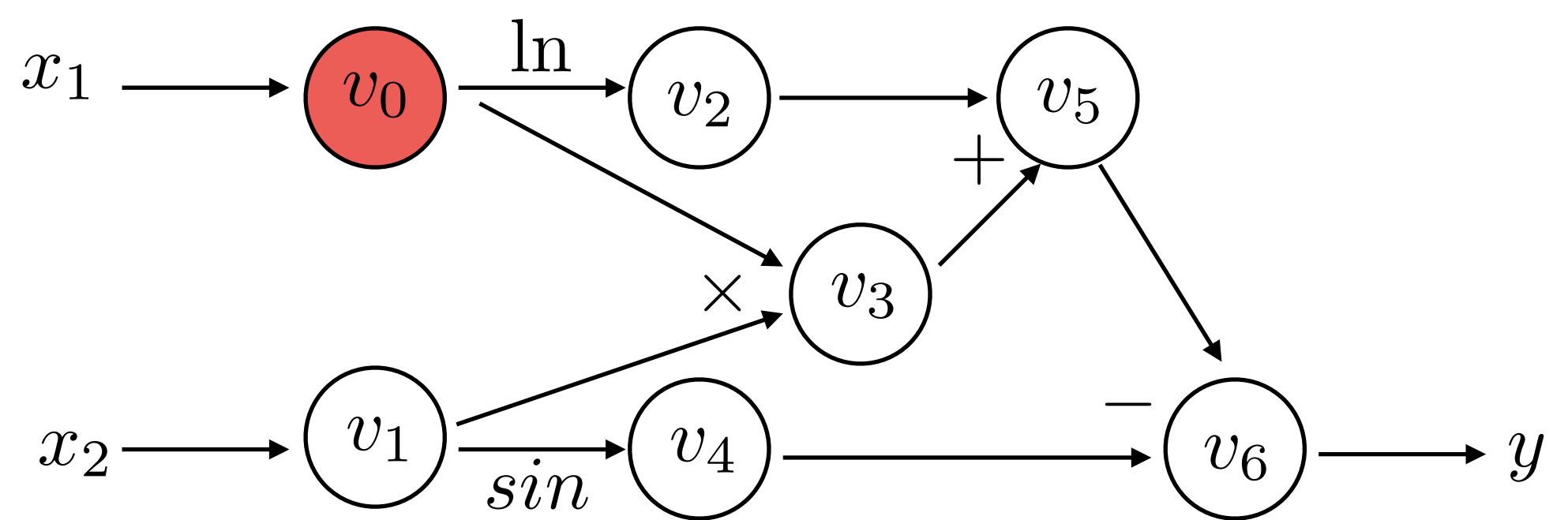
Lets see how we can **evaluate a derivative** using computational graph (DNN learning)

$$\left. \frac{\partial f(x_1, x_2)}{\partial x_1} \right|_{(x_1=2, x_2=5)}$$

We will do this with **forward mode** first, by introducing a derivative of each variable node with respect to the input variable.

AutoDiff - Forward Mode

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



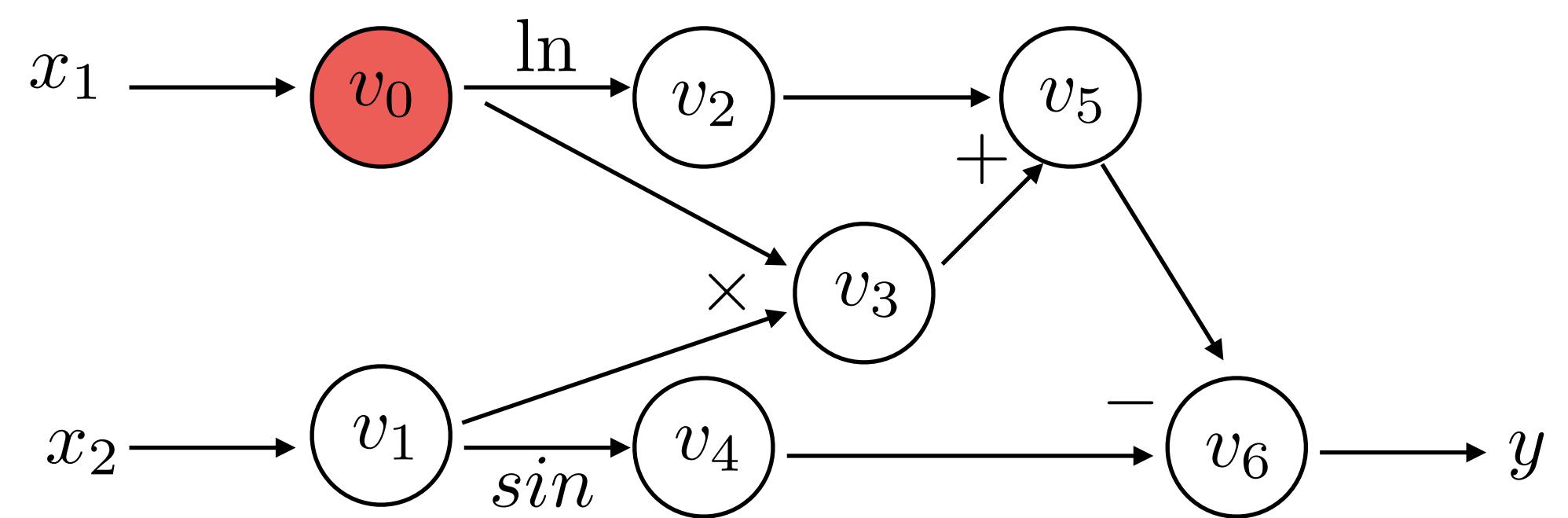
Forward Derivative Trace:

$$\left. \frac{\partial f(x_1, x_2)}{\partial x_1} \right|_{(x_1=2, x_2=5)}$$

Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

AutoDiff - Forward Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 9.734$
$y = v_6$	9.734

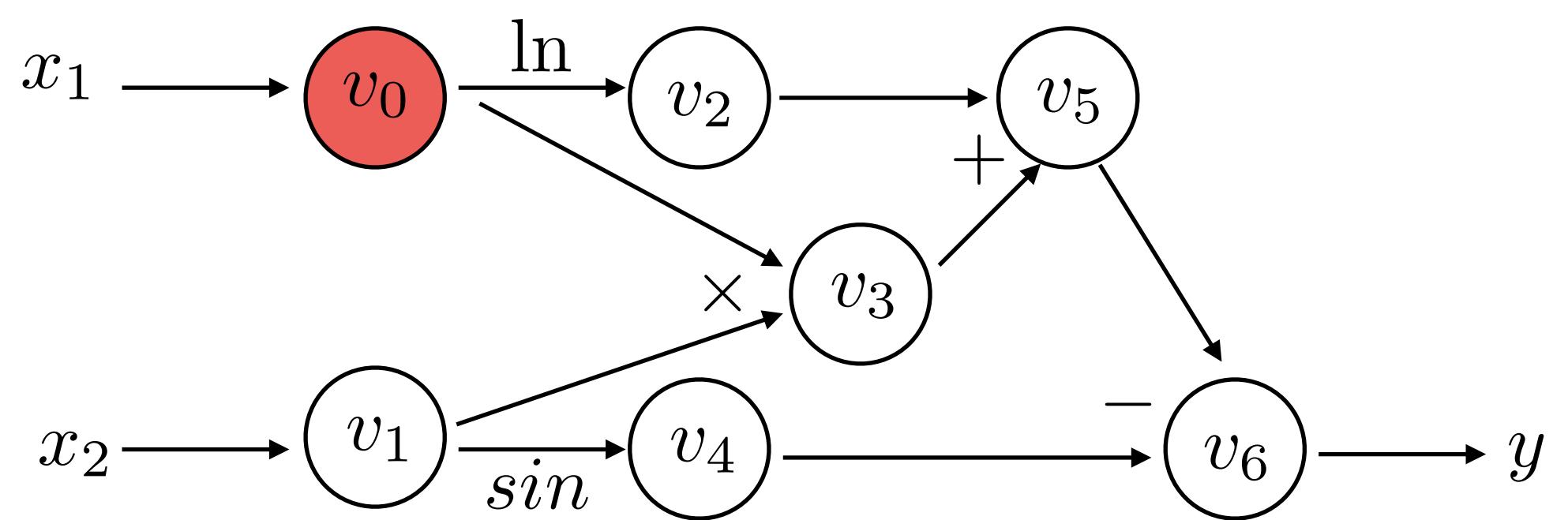
Forward Derivative Trace:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big|_{(x_1=2, x_2=5)}$$

$$\frac{\partial v_0}{\partial x_1}$$

AutoDiff - Forward Mode

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$



Forward Derivative Trace:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big|_{(x_1=2, x_2=5)}$$

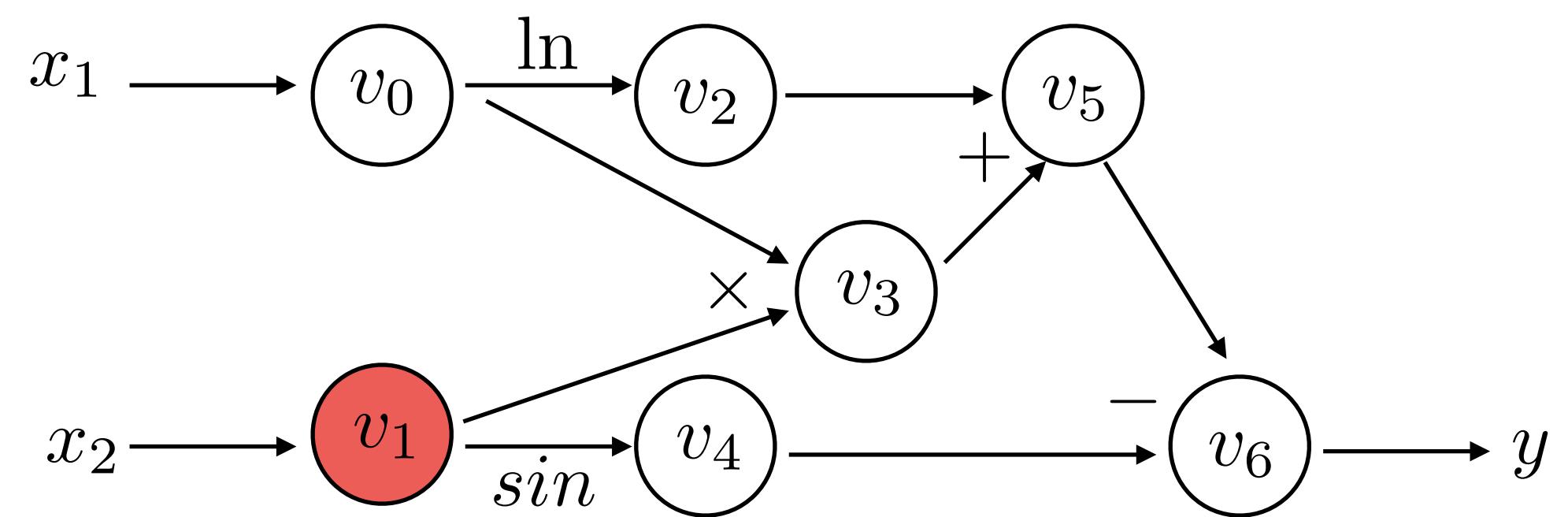
$$\frac{\partial v_0}{\partial x_1}$$

1

Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

AutoDiff - Forward Mode



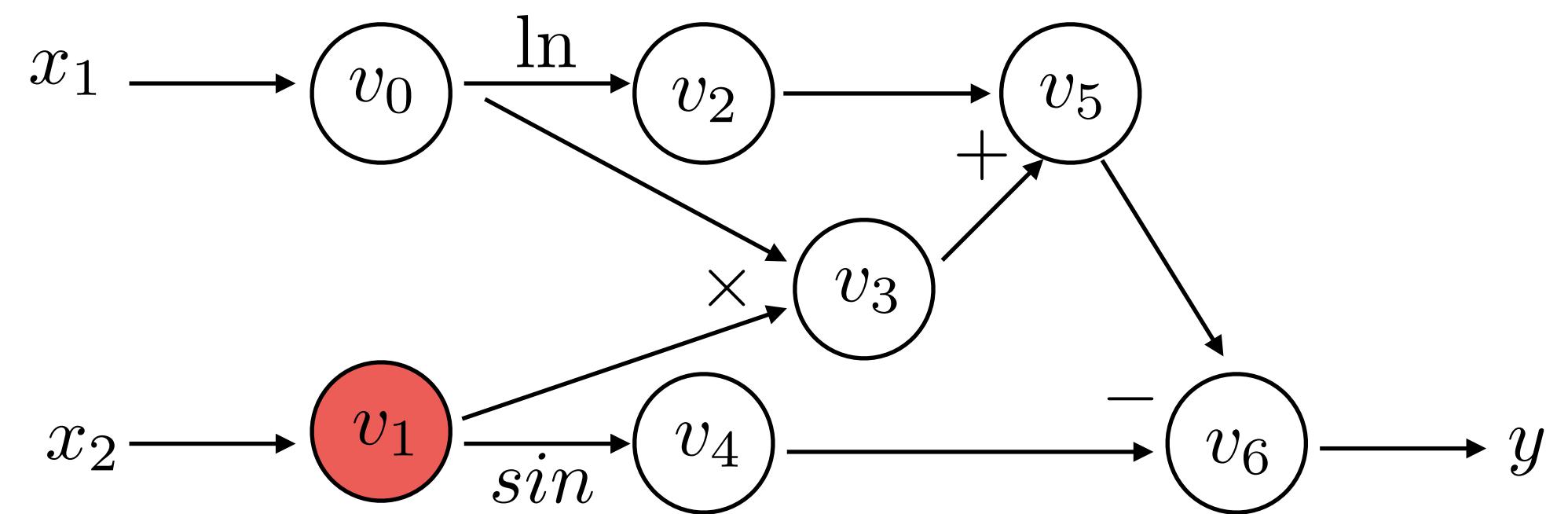
Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
<u>$v_1 = x_2$</u>	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	
$\frac{\partial v_1}{\partial x_1}$	

AutoDiff - Forward Mode



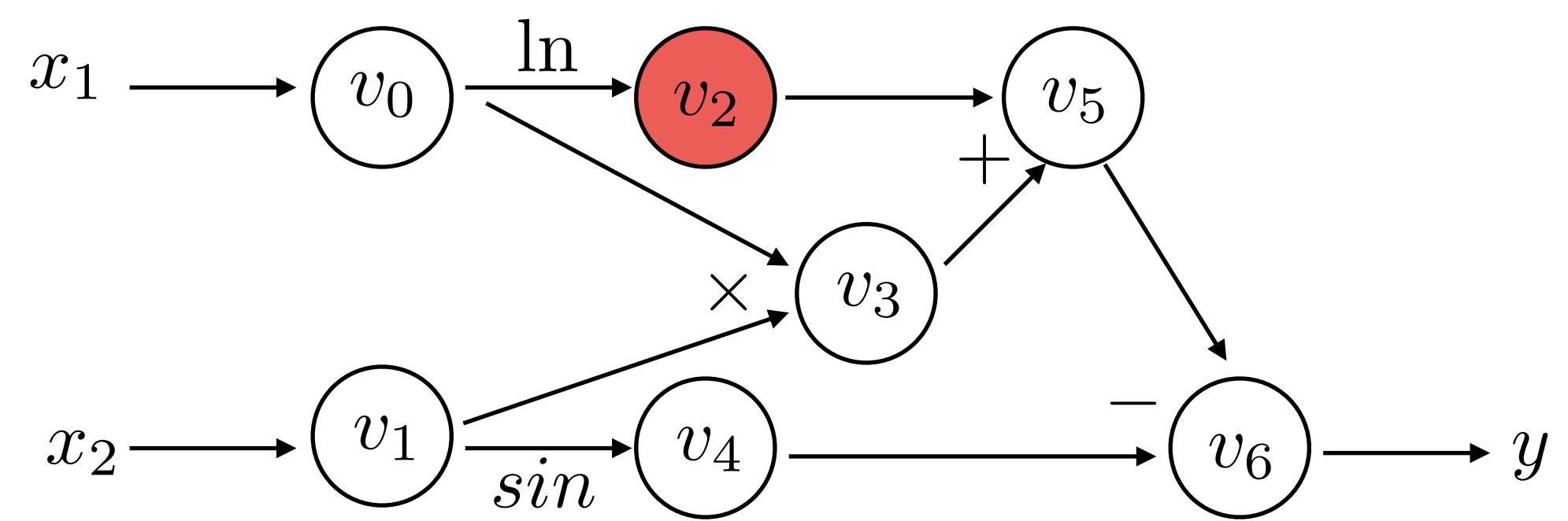
Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
<u>$v_1 = x_2$</u>	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1}$ $(x_1=2, x_2=5)$	1	0
$\frac{\partial v_0}{\partial x_1}$		
$\frac{\partial v_1}{\partial x_1}$		

AutoDiff - Forward Mode



Forward Evaluation Trace:

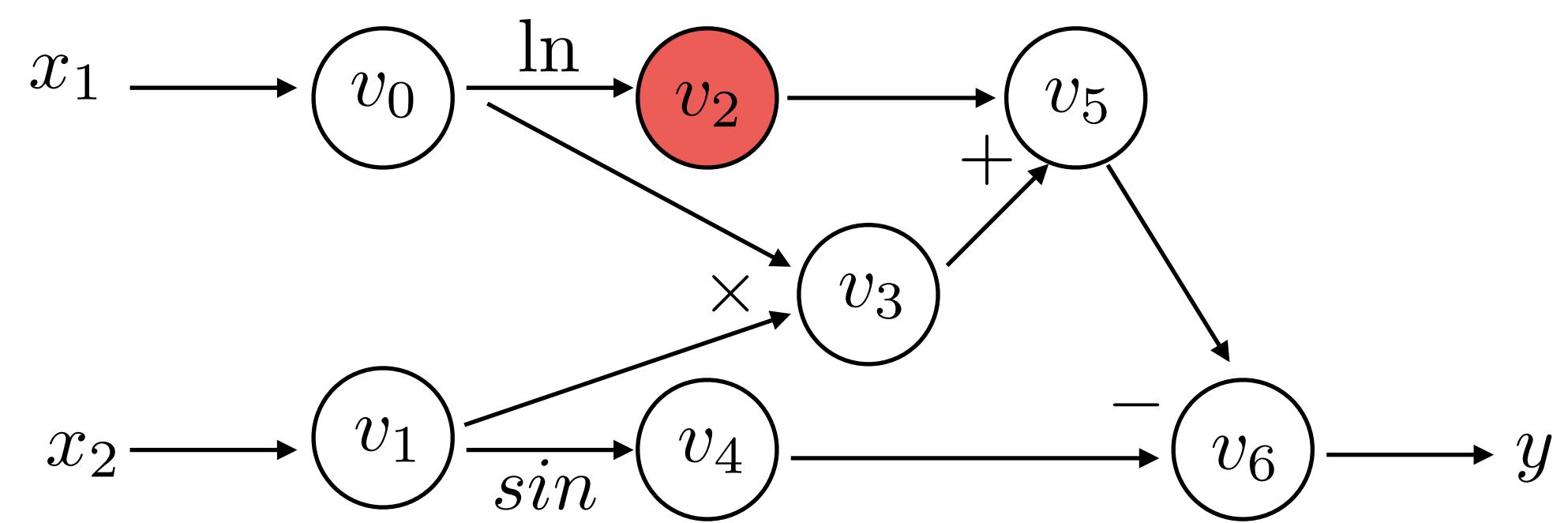
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1}$	1

AutoDiff - Forward Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

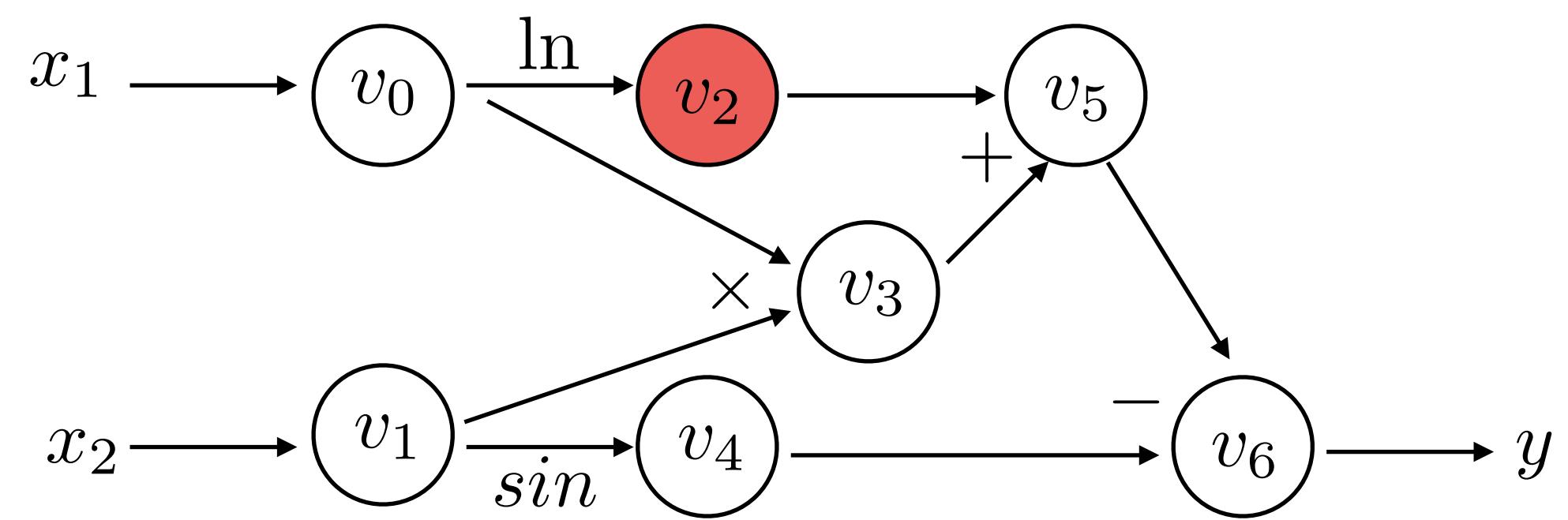
$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1}$	0

Chain Rule

AutoDiff - Forward Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big|_{(x_1=2, x_2=5)}$$

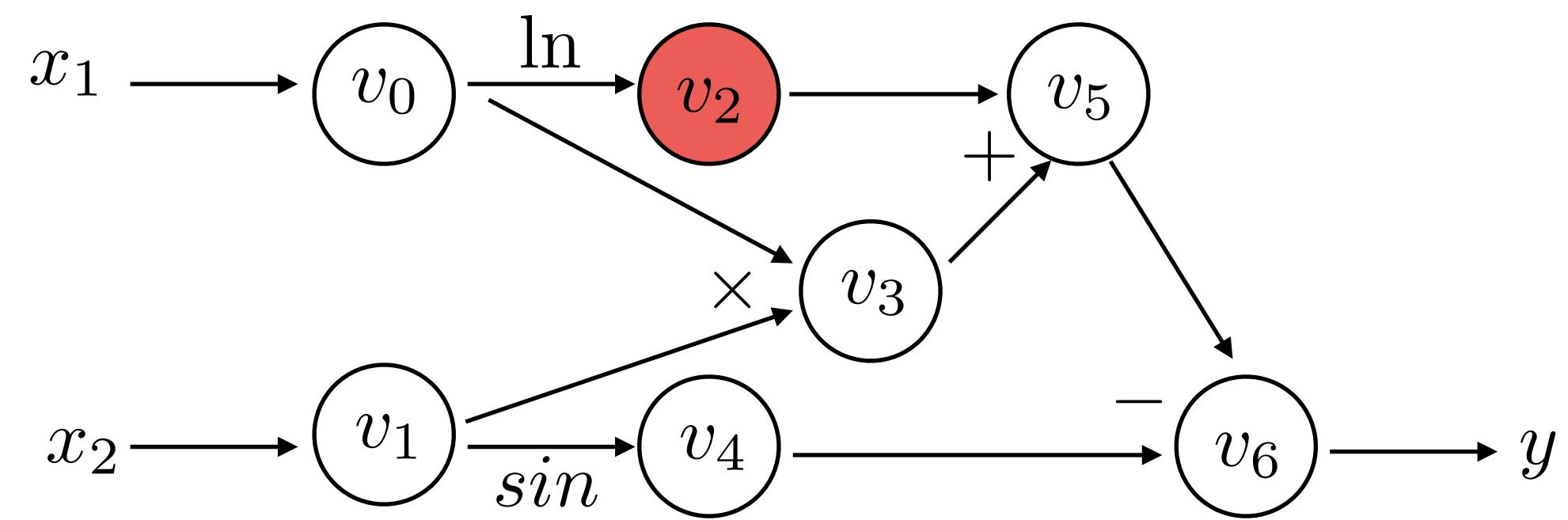
$$\frac{\partial v_0}{\partial x_1}$$

$$\frac{\partial v_1}{\partial x_1}$$

$$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$$

Chain Rule

AutoDiff - Forward Mode



Forward Evaluation Trace:

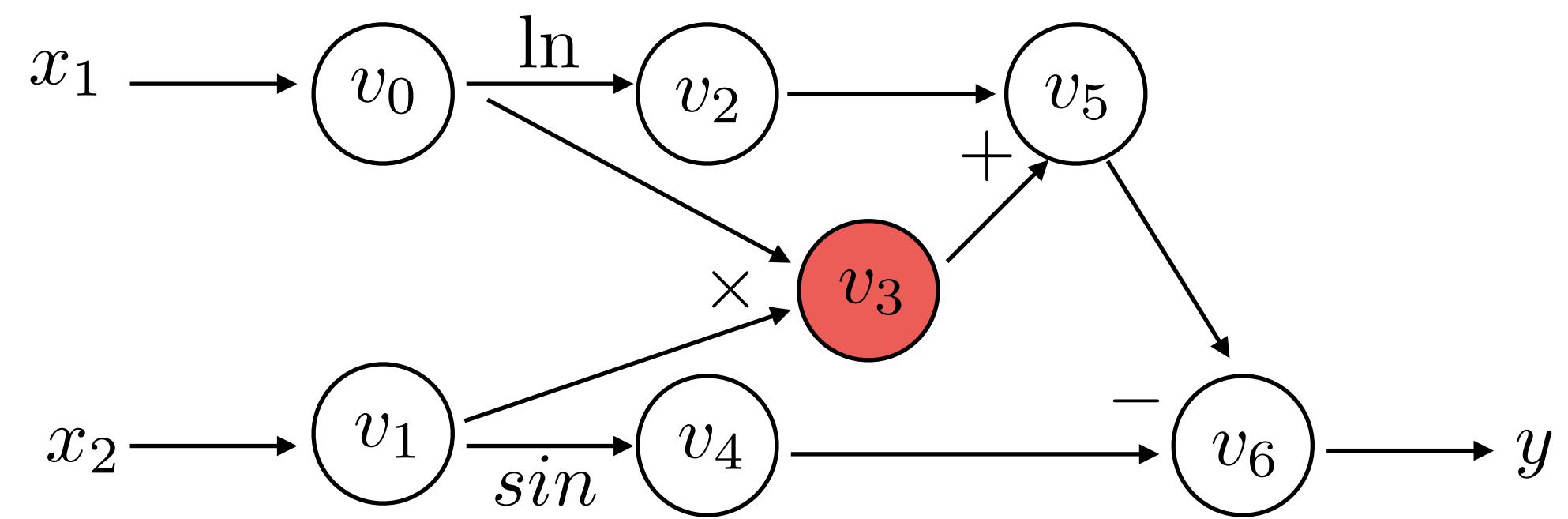
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 9.734$
$y = v_6$	9.734

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
Chain Rule	

AutoDiff - Forward Mode



Forward Evaluation Trace:

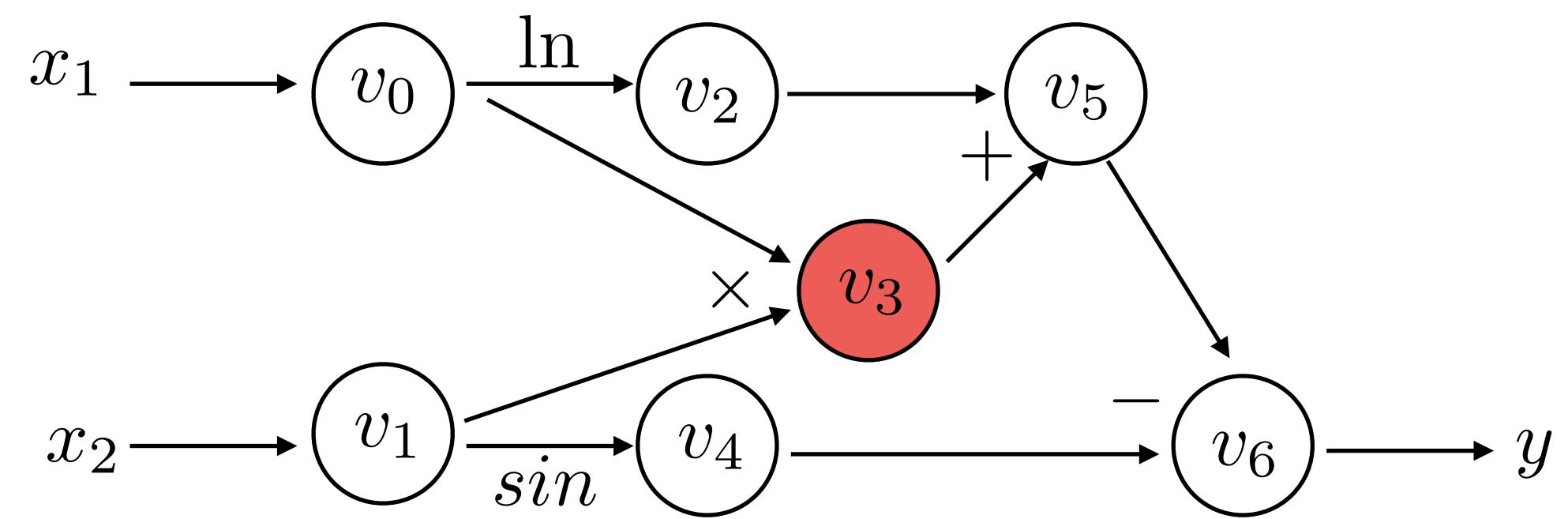
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 9.734$
$y = v_6$	9.734

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1}$	0

AutoDiff - Forward Mode



Forward Evaluation Trace:

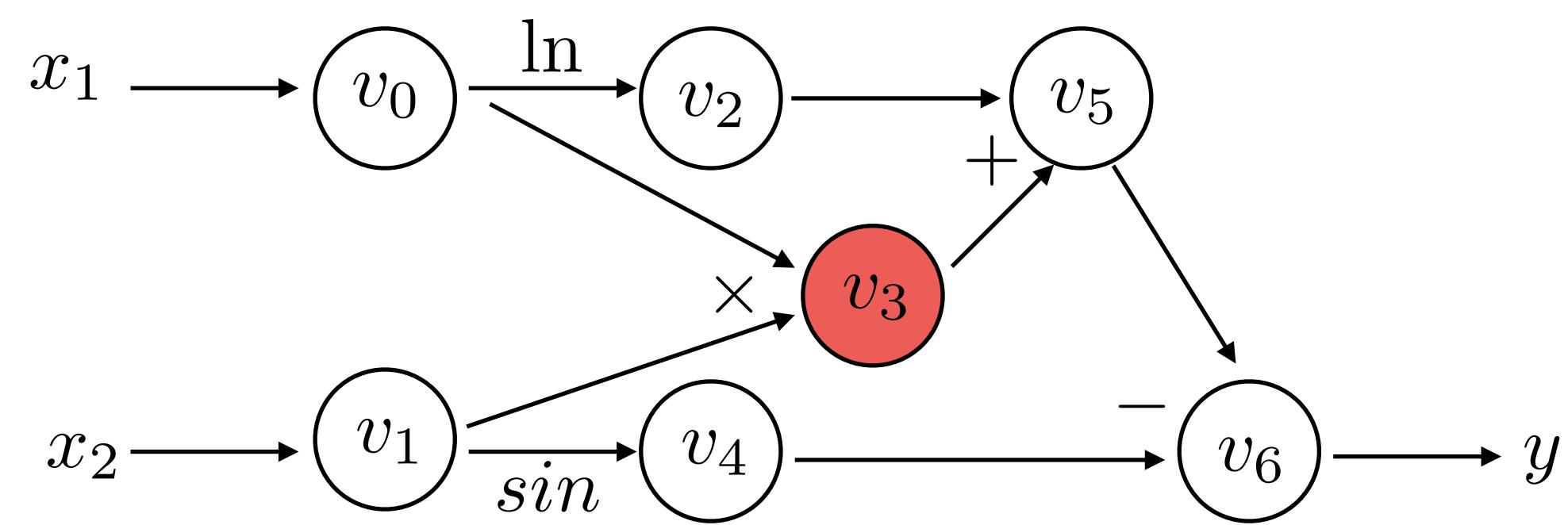
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 9.734$
$y = v_6$	9.734

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1}$	0
Product Rule	

AutoDiff - Forward Mode



Forward Evaluation Trace:

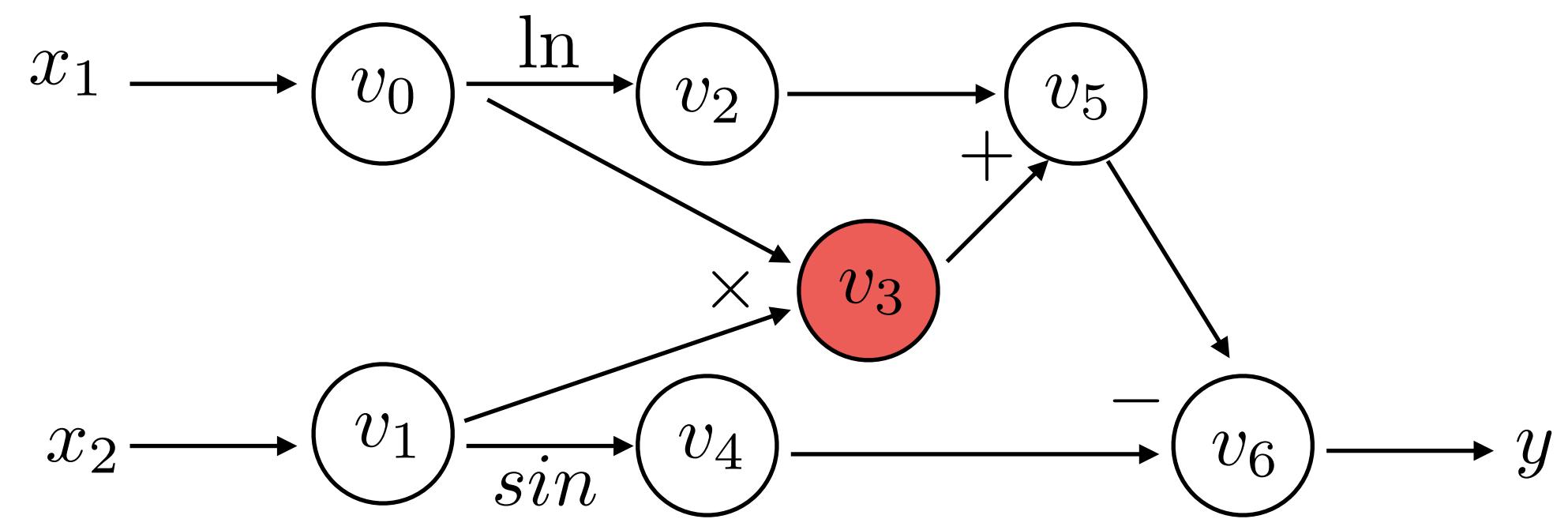
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1} = \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1}$	
Product Rule	

AutoDiff - Forward Mode



Forward Evaluation Trace:

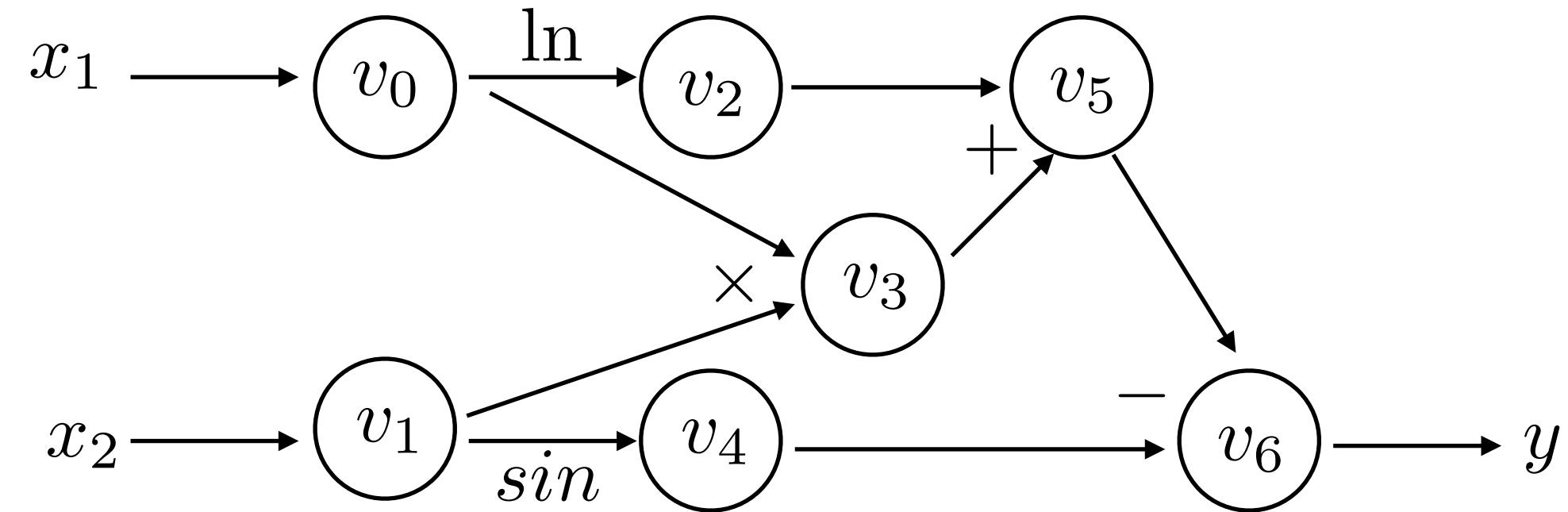
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	1
$\frac{\partial v_0}{\partial x_1}$	0
$\frac{\partial v_1}{\partial x_1}$	
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1} = \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1}$	$1*5 + 2*0 = 5$
Product Rule	

AutoDiff - Forward Mode



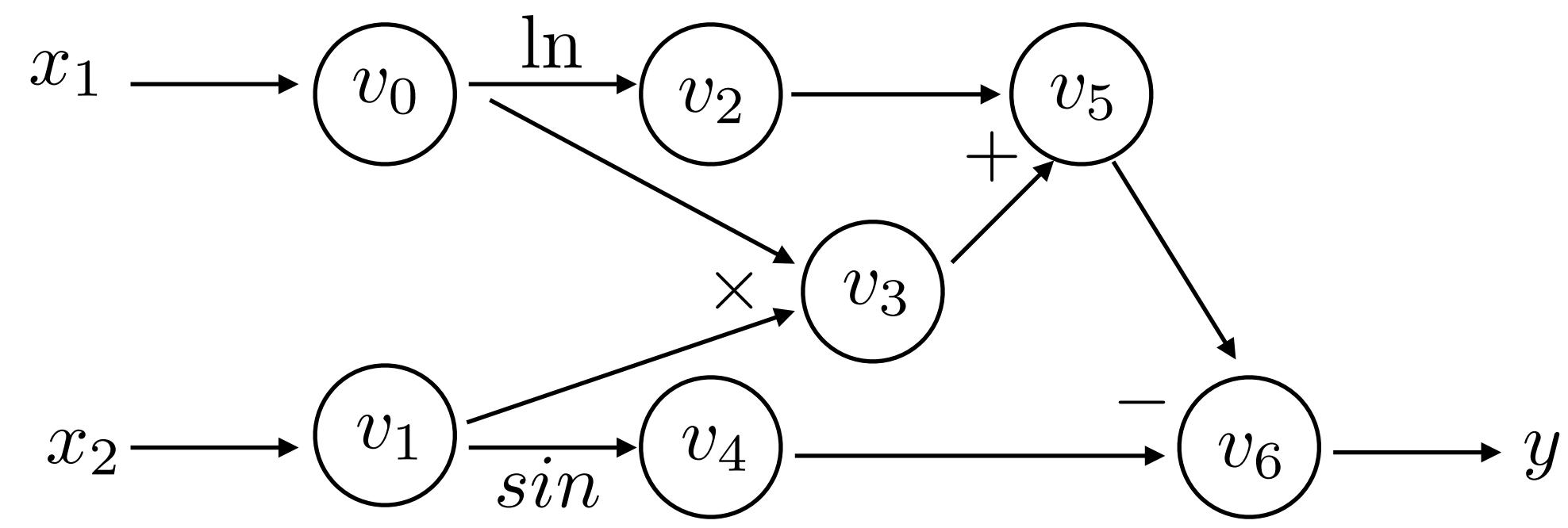
Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	
$\frac{\partial v_0}{\partial x_1}$	1
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1} = \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1}$	$1 * 5 + 2 * 0 = 5$
$\frac{\partial v_4}{\partial x_1} = \frac{\partial v_1}{\partial x_1} \cos(v_1)$	$0 * \cos(5) = 0$
$\frac{\partial v_5}{\partial x_1} = \frac{\partial v_2}{\partial x_1} + \frac{\partial v_3}{\partial x_1}$	$0.5 + 5 = 5.5$
$\frac{\partial v_6}{\partial x_1} = \frac{\partial v_5}{\partial x_1} - \frac{\partial v_4}{\partial x_1}$	$5.5 - 0 = 5.5$
$\frac{\partial y}{\partial x_1} = \frac{\partial v_6}{\partial x_1}$	5.5

AutoDiff - Forward Mode



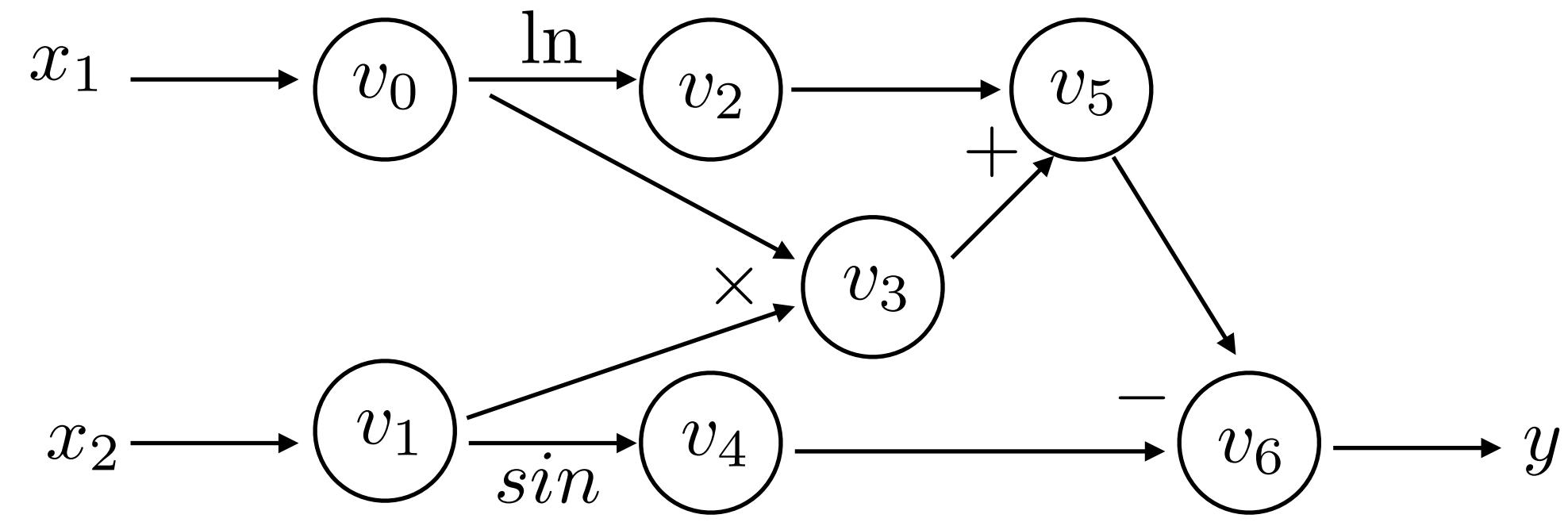
We now have:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big|_{(x_1=2, x_2=5)} = 5.5$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	
$\frac{\partial v_0}{\partial x_1}$	1
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1} = \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1}$	$1*5 + 2*0 = 5$
$\frac{\partial v_4}{\partial x_1} = \frac{\partial v_1}{\partial x_1} \cos(v_1)$	$0 * \cos(5) = 0$
$\frac{\partial v_5}{\partial x_1} = \frac{\partial v_2}{\partial x_1} + \frac{\partial v_3}{\partial x_1}$	$0.5 + 5 = 5.5$
$\frac{\partial v_6}{\partial x_1} = \frac{\partial v_5}{\partial x_1} - \frac{\partial v_4}{\partial x_1}$	$5.5 - 0 = 5.5$
$\frac{\partial y}{\partial x_1} = \frac{\partial v_6}{\partial x_1}$	5.5

AutoDiff - Forward Mode



We now have:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big|_{(x_1=2, x_2=5)} = 5.5$$

Still need:

$$\frac{\partial f(x_1, x_2)}{\partial x_2} \Big|_{(x_1=2, x_2=5)}$$

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Derivative Trace:

$\frac{\partial f(x_1, x_2)}{\partial x_1} \Big _{(x_1=2, x_2=5)}$	
$\frac{\partial v_0}{\partial x_1}$	1
$\frac{\partial v_1}{\partial x_1}$	0
$\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1}$	$1/2 * 1 = 0.5$
$\frac{\partial v_3}{\partial x_1} = \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1}$	$1*5 + 2*0 = 5$
$\frac{\partial v_4}{\partial x_1} = \frac{\partial v_1}{\partial x_1} \cos(v_1)$	$0 * \cos(5) = 0$
$\frac{\partial v_5}{\partial x_1} = \frac{\partial v_2}{\partial x_1} + \frac{\partial v_3}{\partial x_1}$	$0.5 + 5 = 5.5$
$\frac{\partial v_6}{\partial x_1} = \frac{\partial v_5}{\partial x_1} - \frac{\partial v_4}{\partial x_1}$	$5.5 - 0 = 5.5$
$\frac{\partial y}{\partial x_1} = \frac{\partial v_6}{\partial x_1}$	5.5

AutoDiff - Forward Mode

Forward mode needs m forward passes to get a full Jacobian (all gradients of output with respect to each input), where m is the number of inputs

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

AutoDiff - Forward Mode

Forward mode needs m forward passes to get a full Jacobian (all gradients of output with respect to each input), where m is the number of inputs

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

Problem: DNN typically has large number of inputs:

image as an input, plus all the weights and biases of layers = millions of inputs!

and very few outputs (many DNNs have $n = 1$)

AutoDiff - Forward Mode

Forward mode needs m forward passes to get a full Jacobian (all gradients of output with respect to each input), where m is the number of inputs

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

Problem: DNN typically has large number of inputs:

image as an input, plus all the weights and biases of layers = millions of inputs!

and very few outputs (many DNNs have $n = 1$)

Why?

AutoDiff - Forward Mode

Forward mode needs m forward passes to get a full Jacobian (all gradients of output with respect to each input), where m is the number of inputs

$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

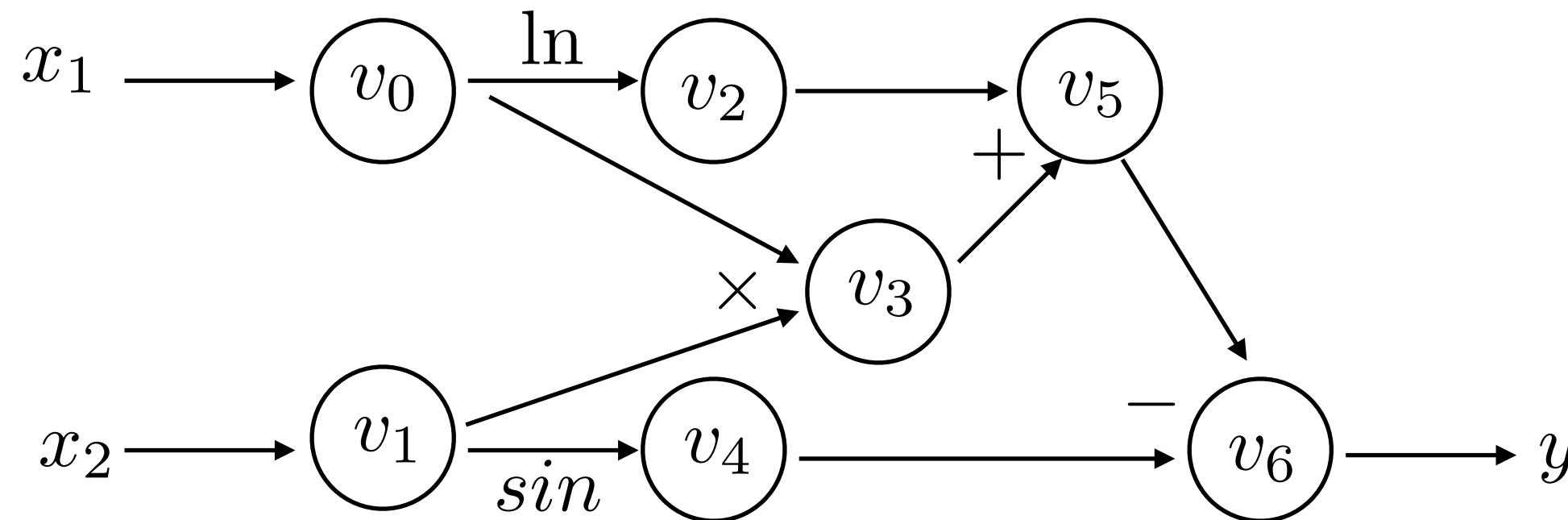
Problem: DNN typically has large number of inputs:

image as an input, plus all the weights and biases of layers = millions of inputs!

and very few outputs (many DNNs have $n = 1$)

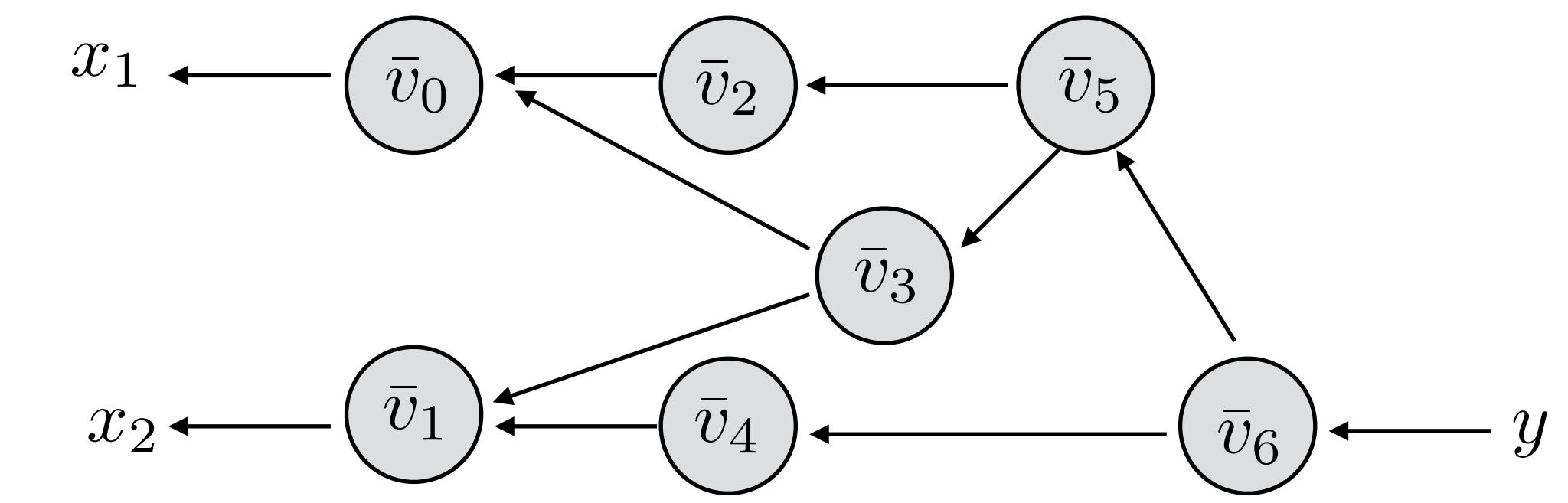
Automatic differentiation in **reverse mode** computes all gradients in n backwards passes (so for most DNNs in a single back pass — **back propagation**)

AutoDiff - Reverse Mode



Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

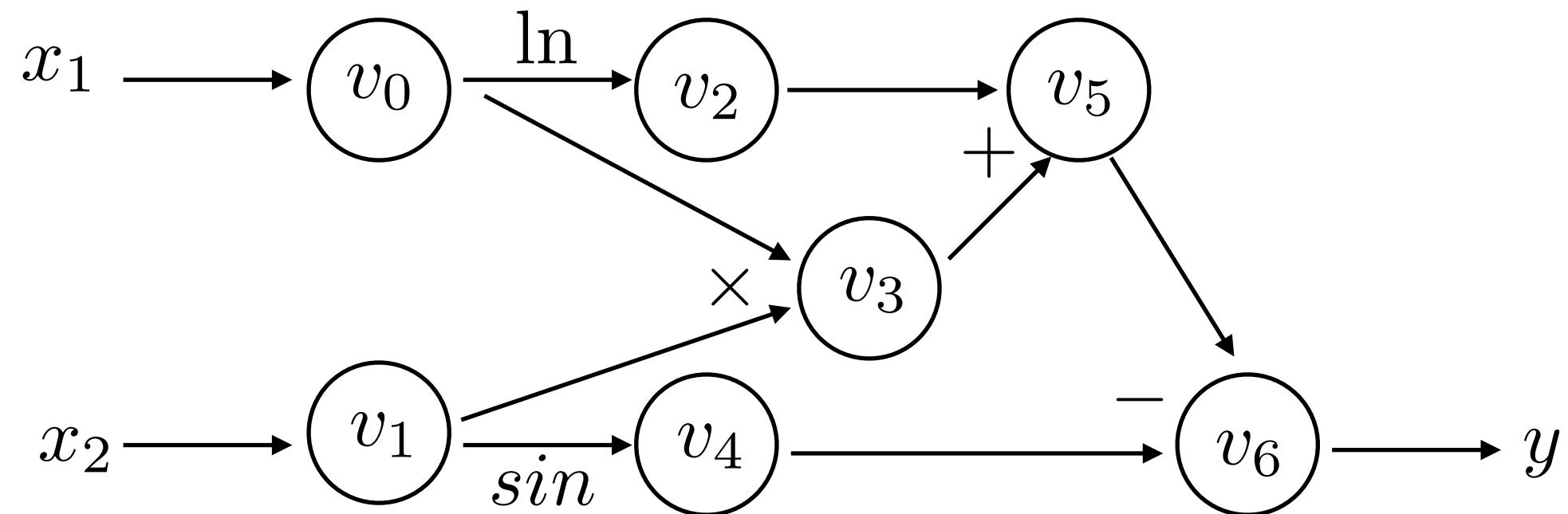


Traverse the original graph in the *reverse* topological order and for each node in the original graph introduce an **adjoint node**, which computes derivative of the output with respect to the local node (using Chain rule):

$$\bar{v}_i = \frac{\partial y_j}{\partial v_i} = \sum_{k \in \text{pa}(i)} \frac{\partial v_k}{\partial v_i} \frac{\partial y_j}{\partial v_k} = \sum_{k \in \text{pa}(i)} \frac{\partial v_k}{\partial v_i} \bar{v}_k$$

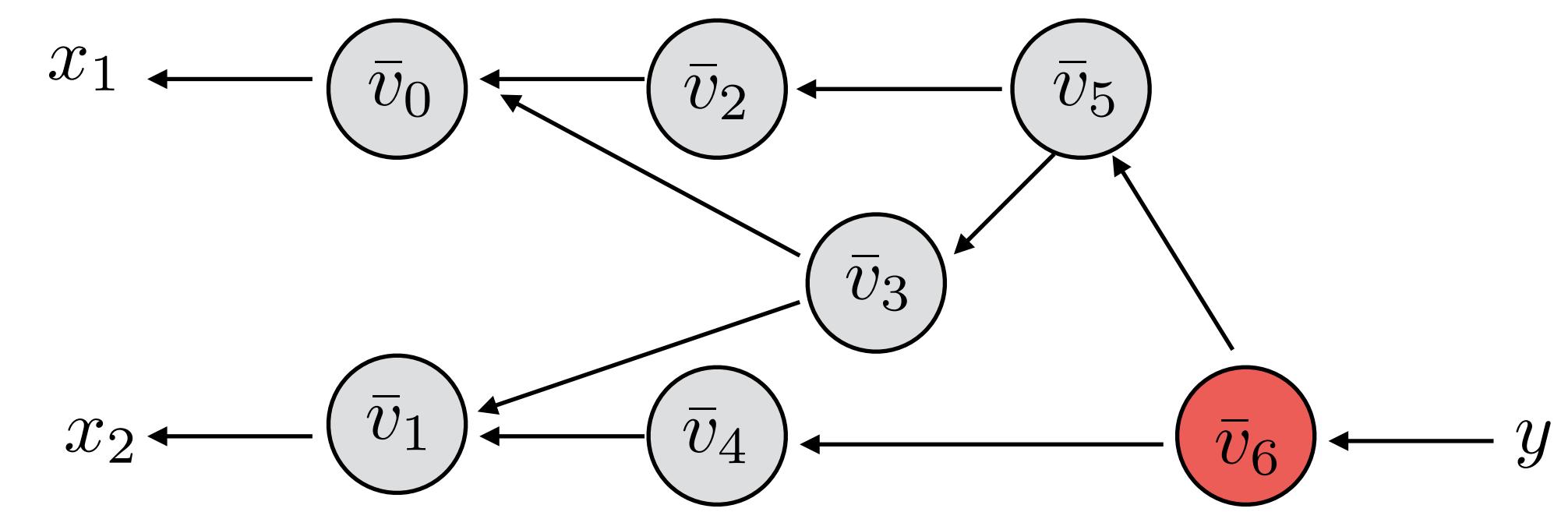
“local” derivative

AutoDiff - Reverse Mode



Forward Evaluation Trace:

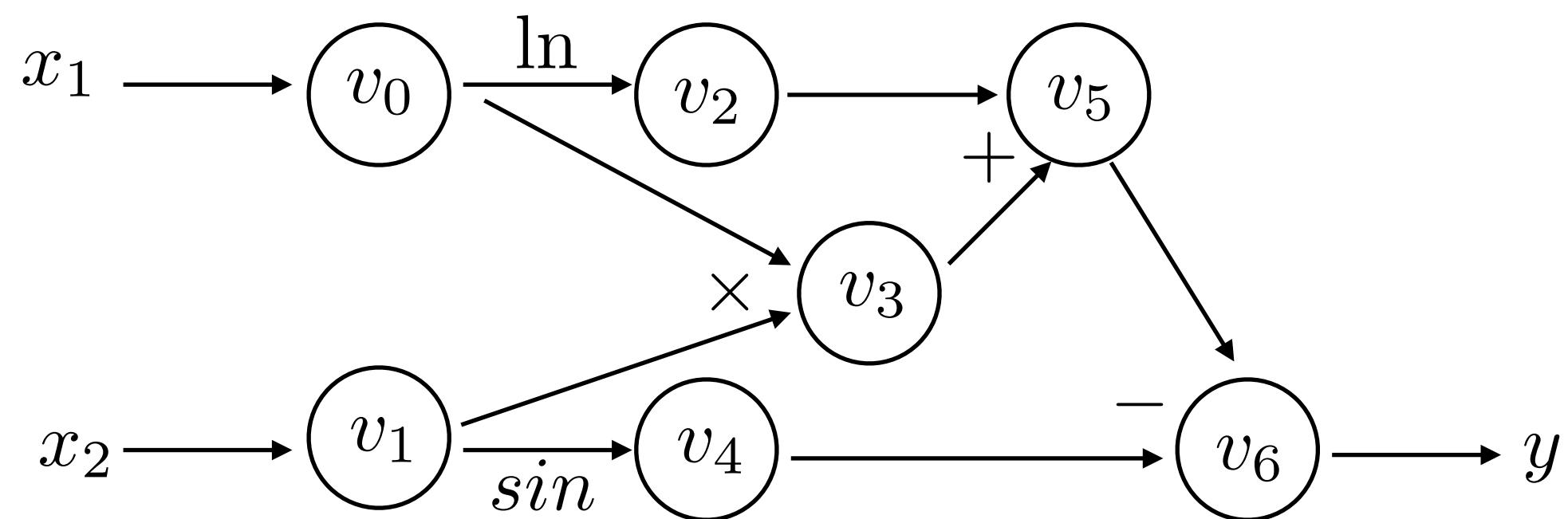
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

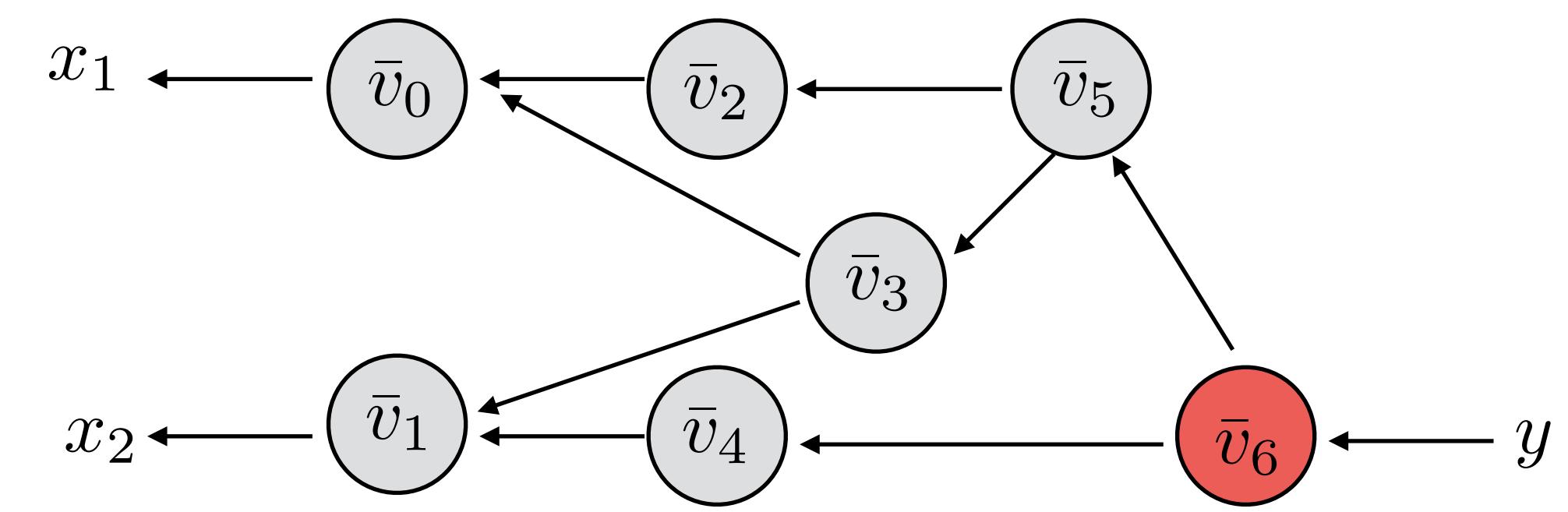
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

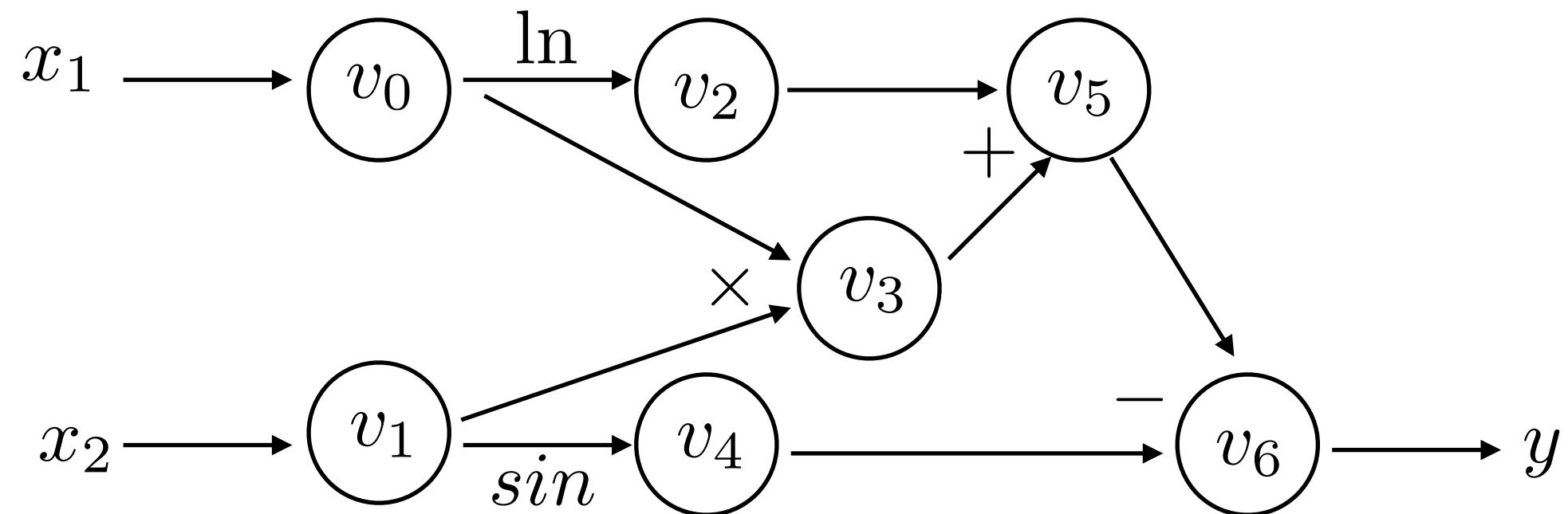
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

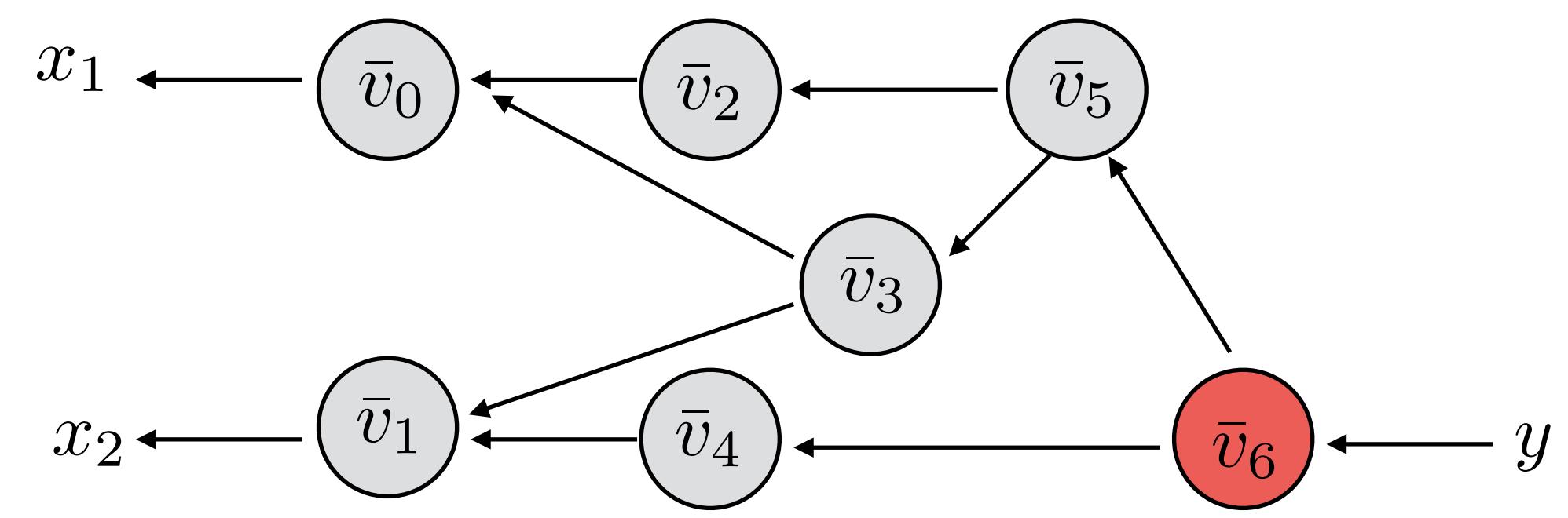
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

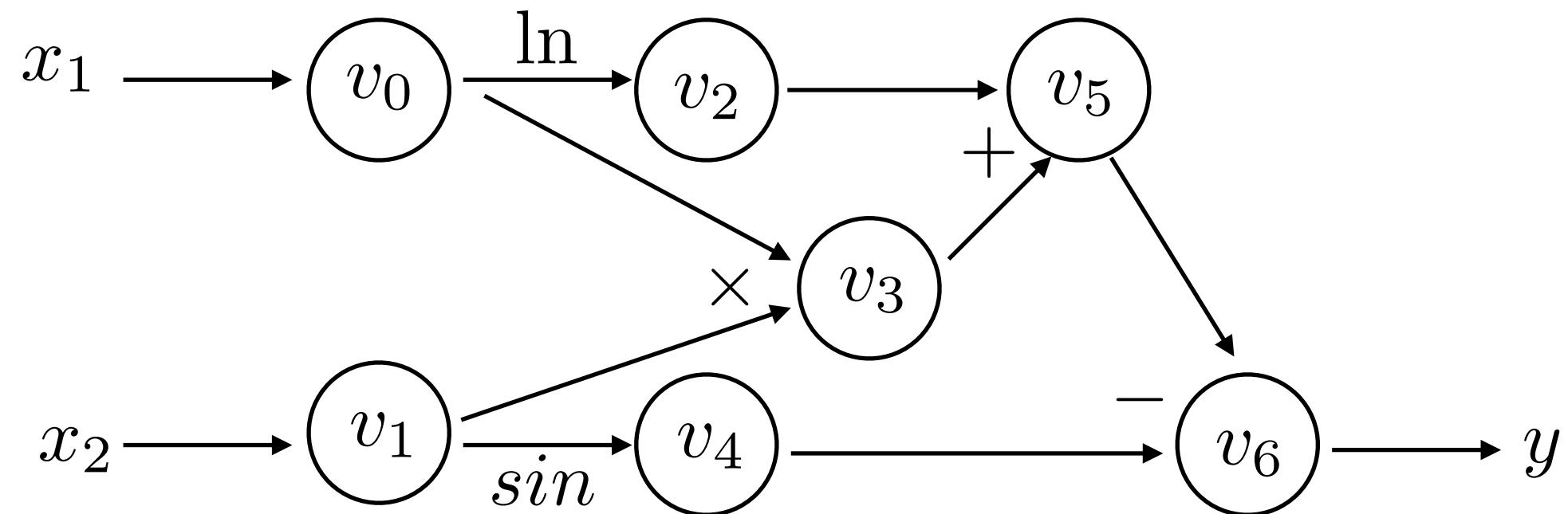
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

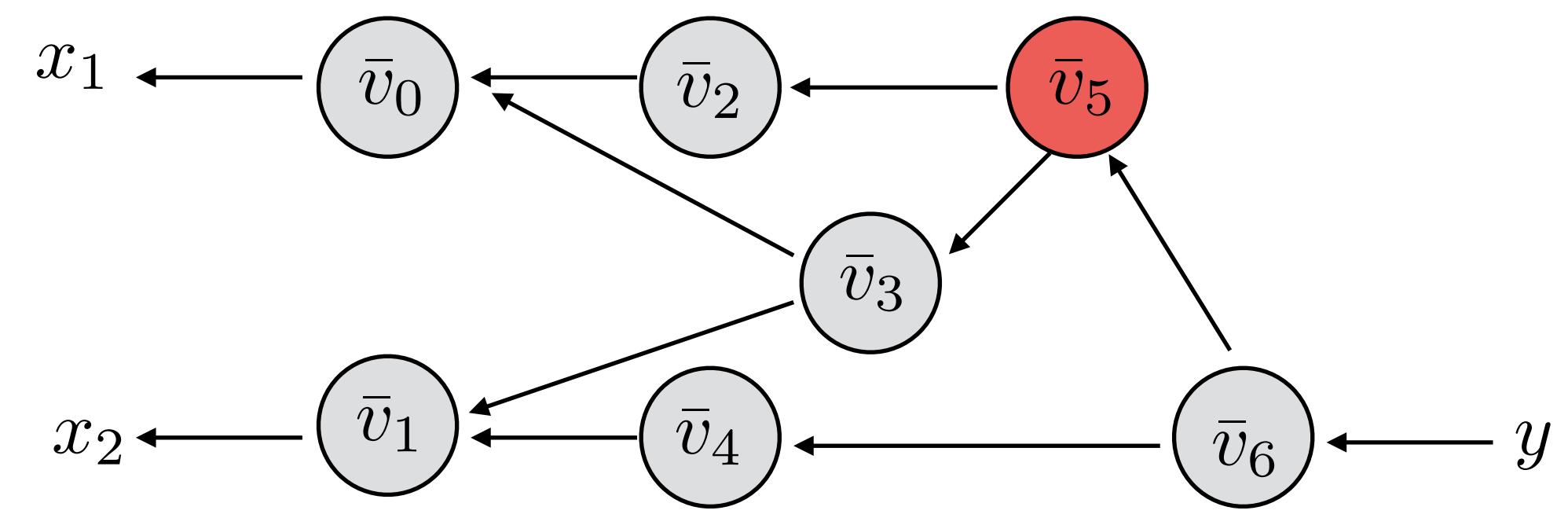
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

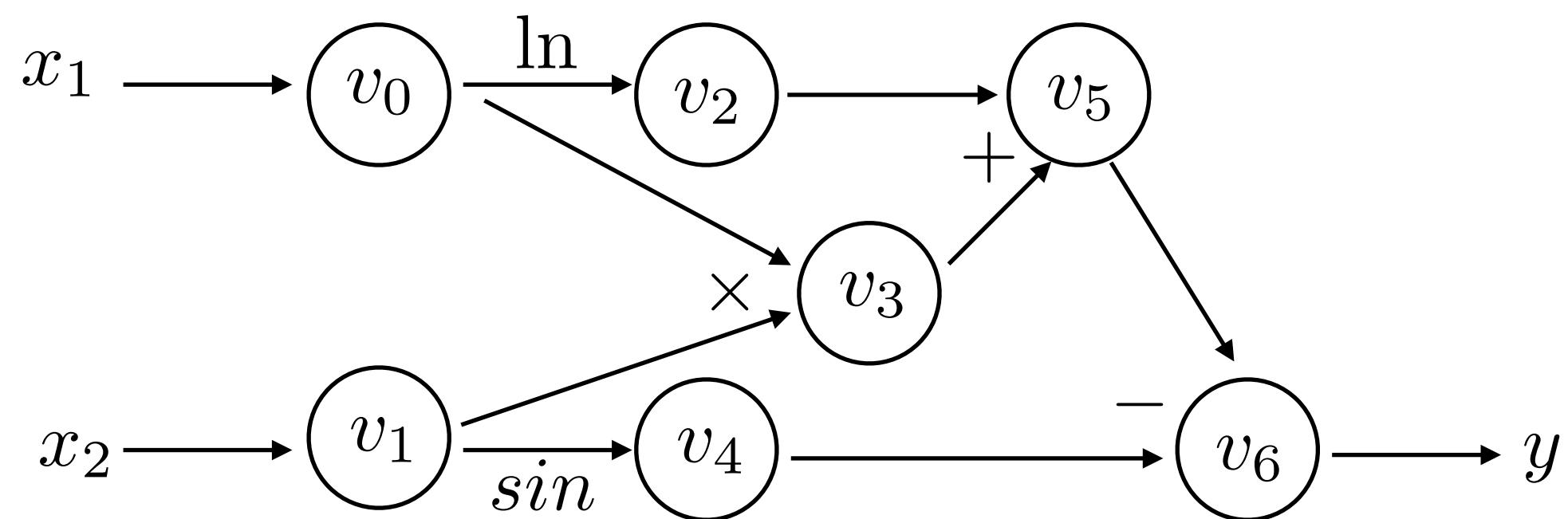


Backwards Derivative Trace:

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5}$$

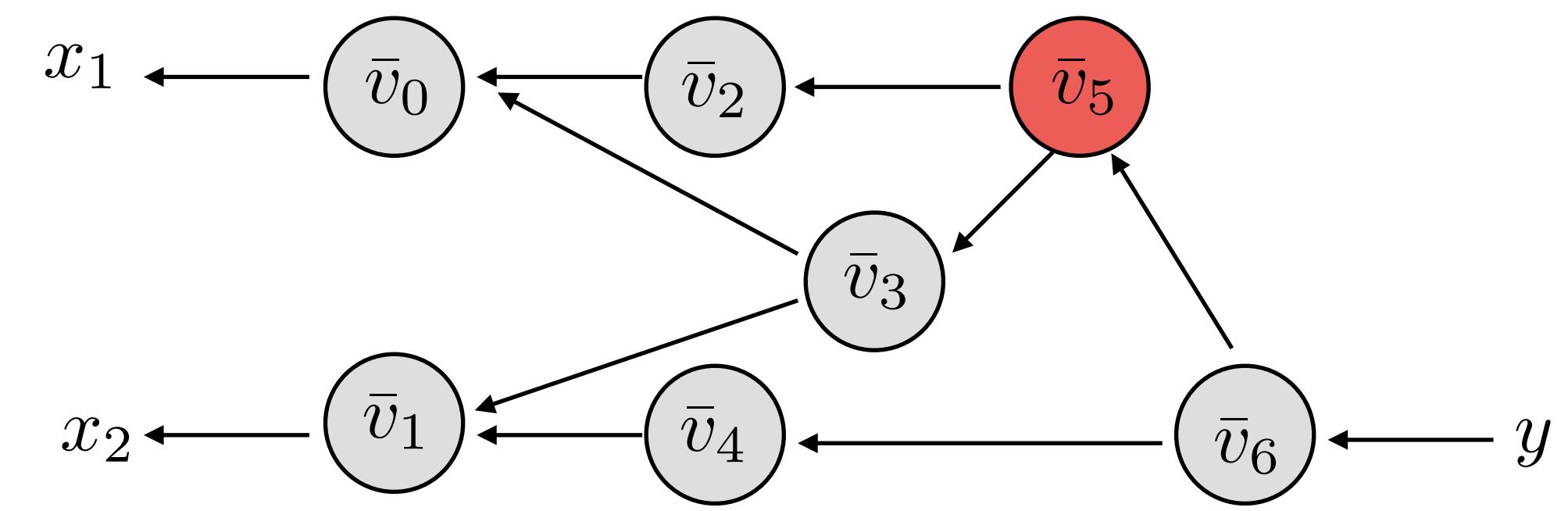
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
<u>$y = v_6$</u>	11.652

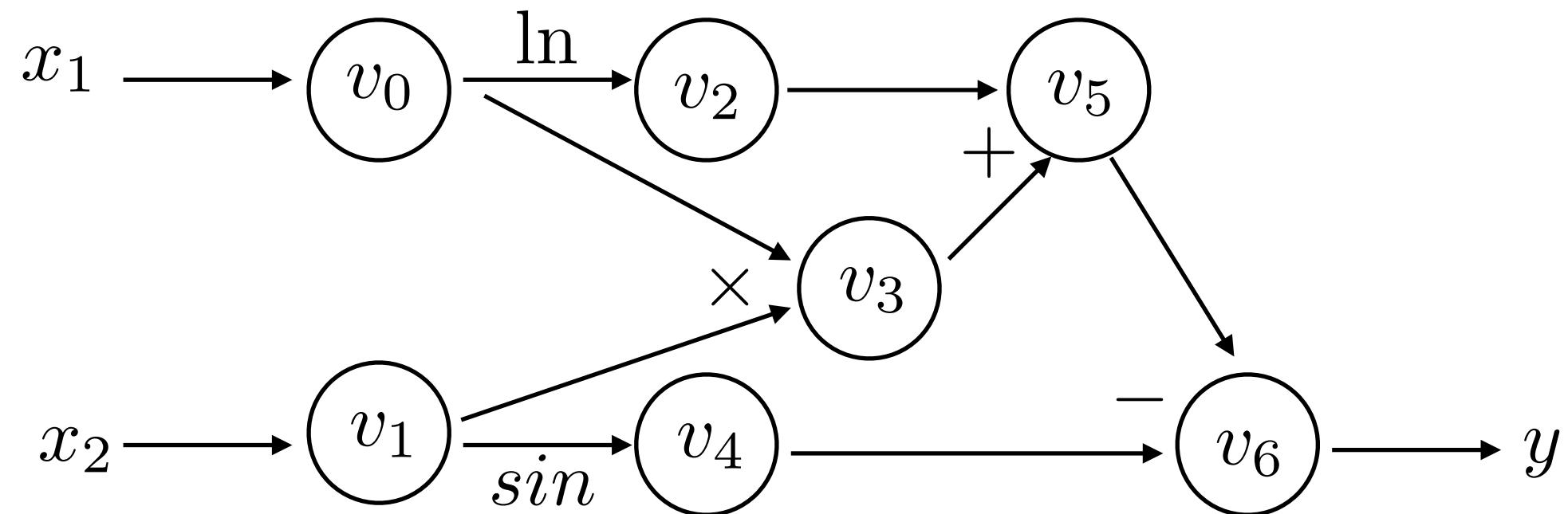


Backwards Derivative Trace:

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5}$$

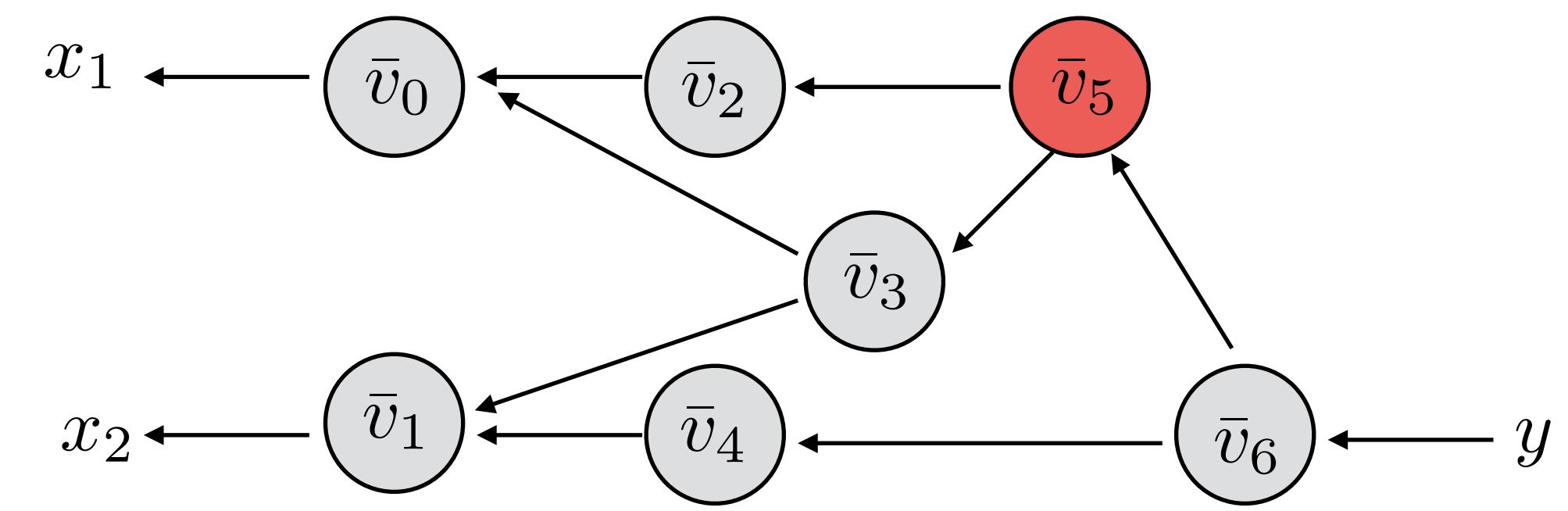
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
<u>$y = v_6$</u>	11.652

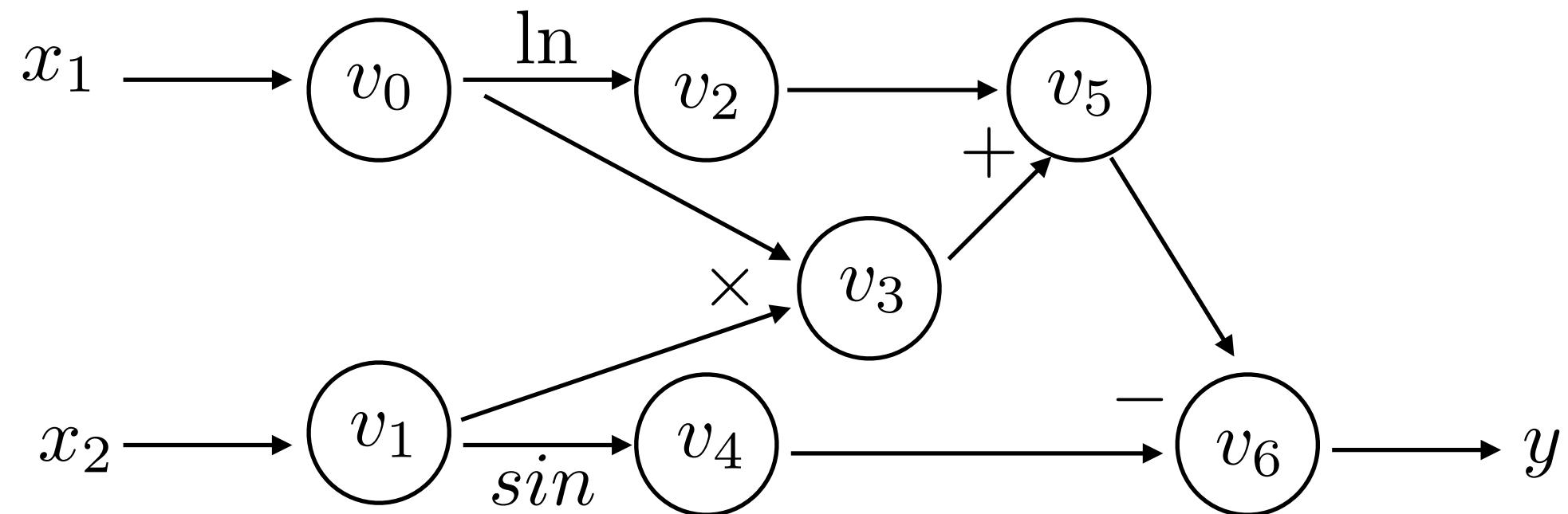


Backwards Derivative Trace:

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

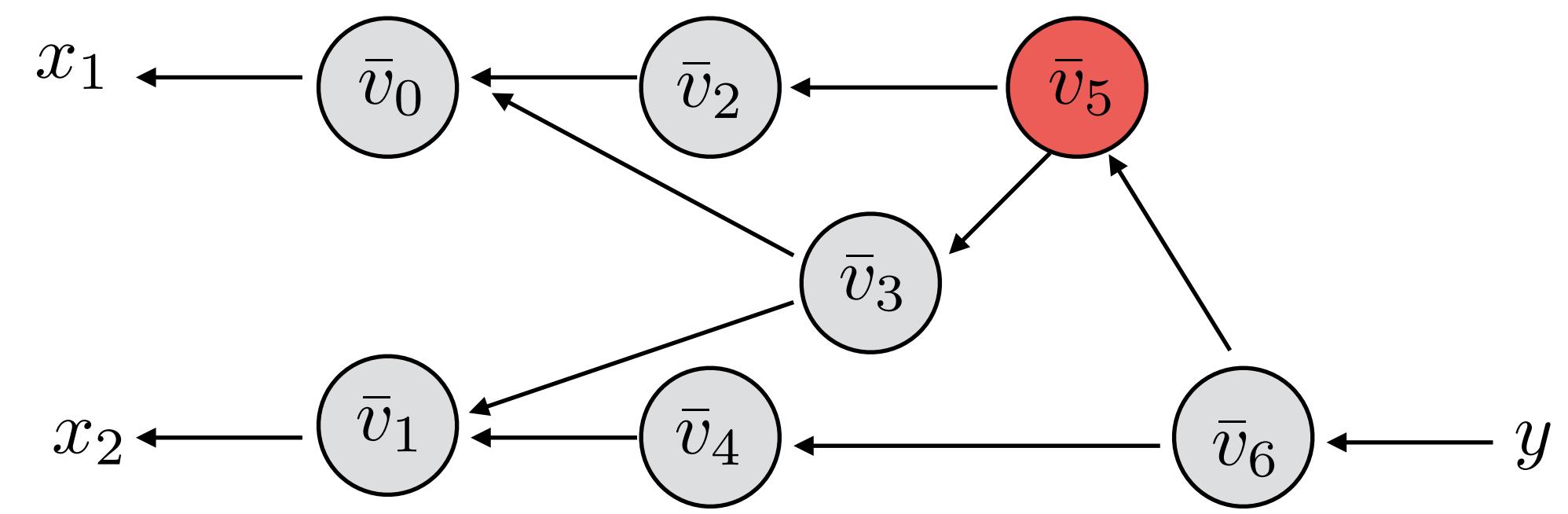
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

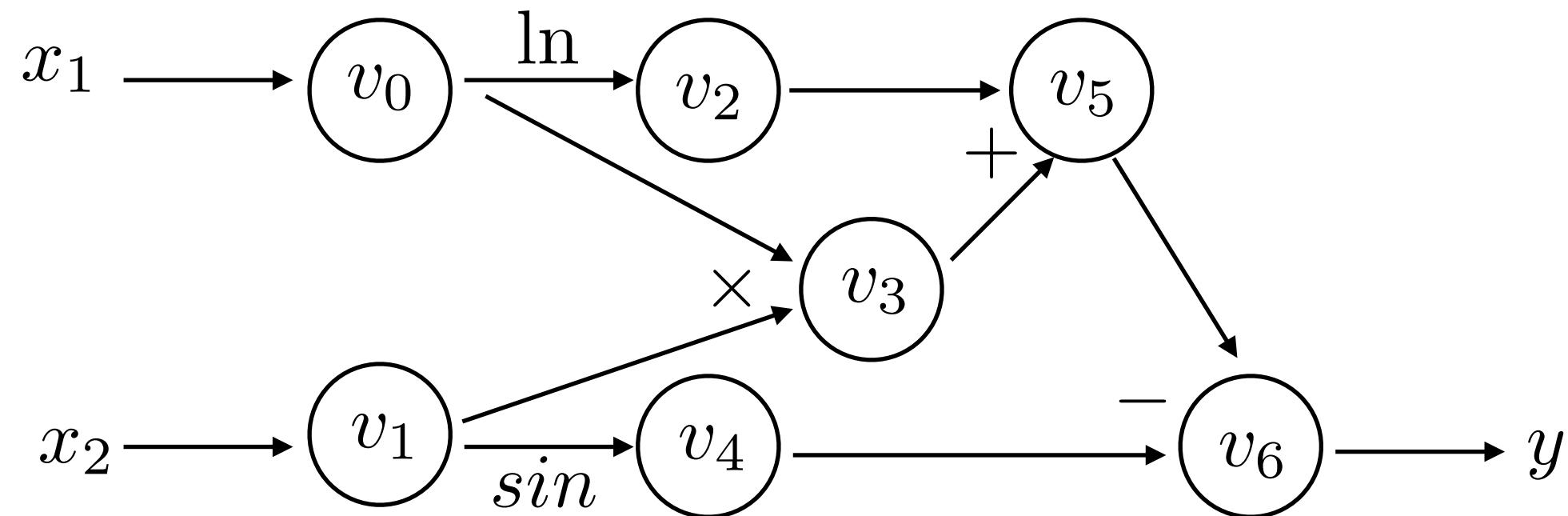
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
<u>$y = v_6$</u>	11.652



Backwards Derivative Trace:

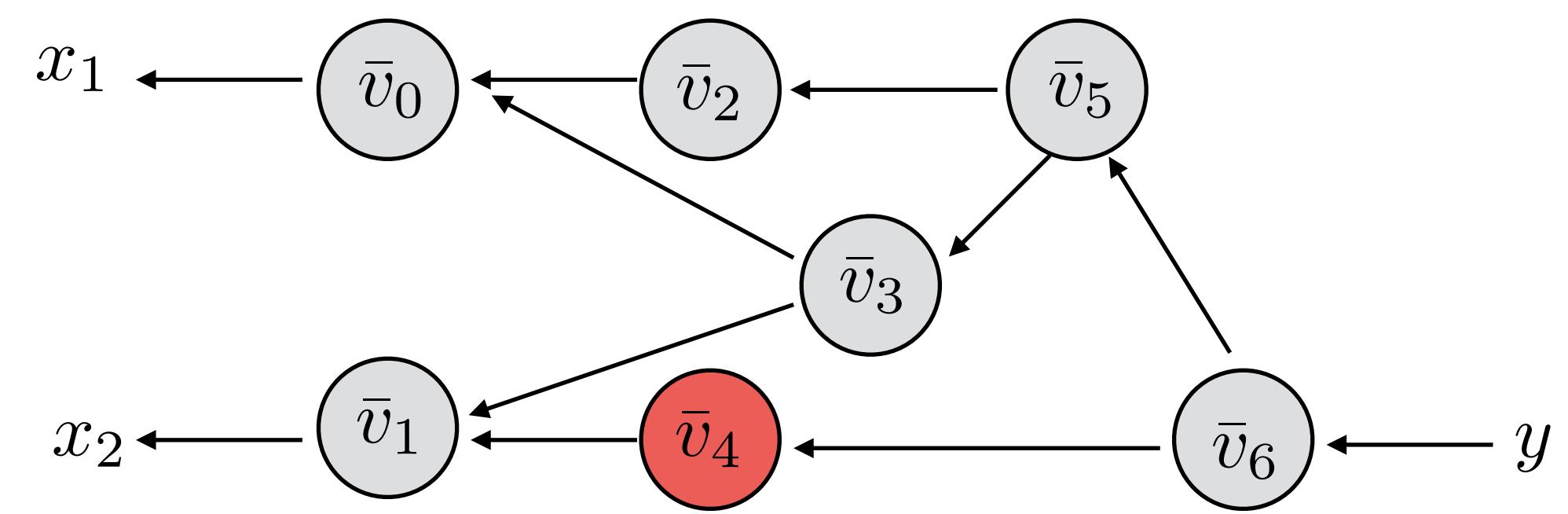
$$\begin{aligned}
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 &= 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1
 \end{aligned}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



$$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4}$$

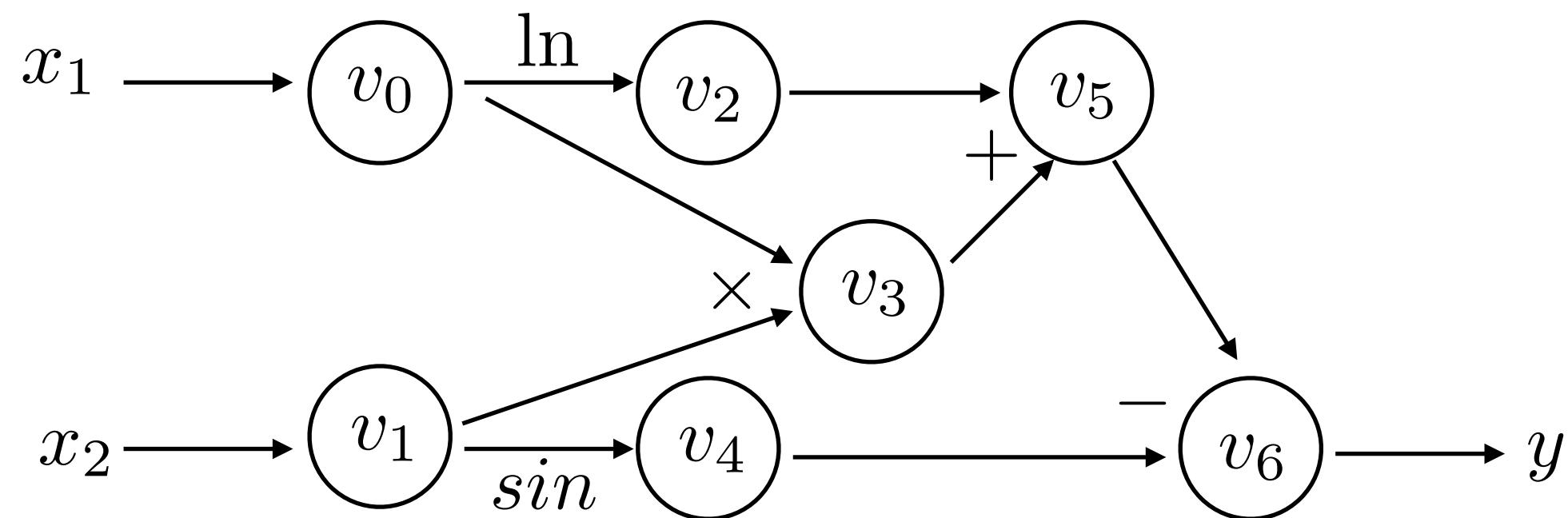
$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

$$1 \times 1 = 1$$

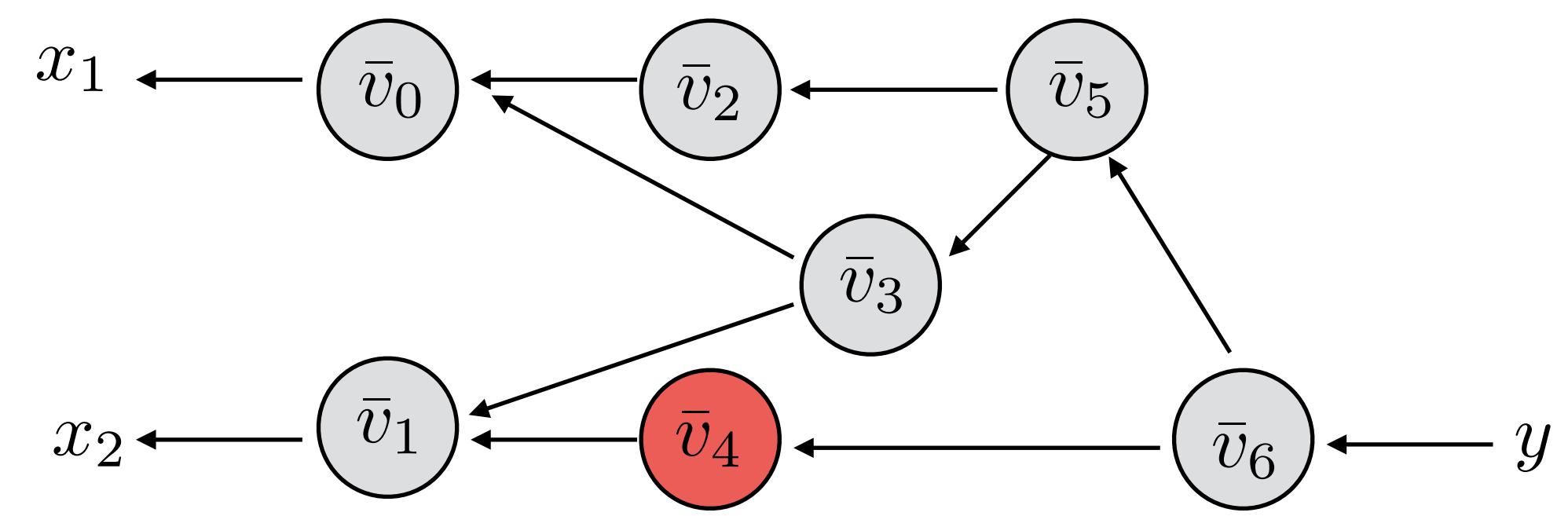
$$1$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
<u>$y = v_6$</u>	11.652



$$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4}$$

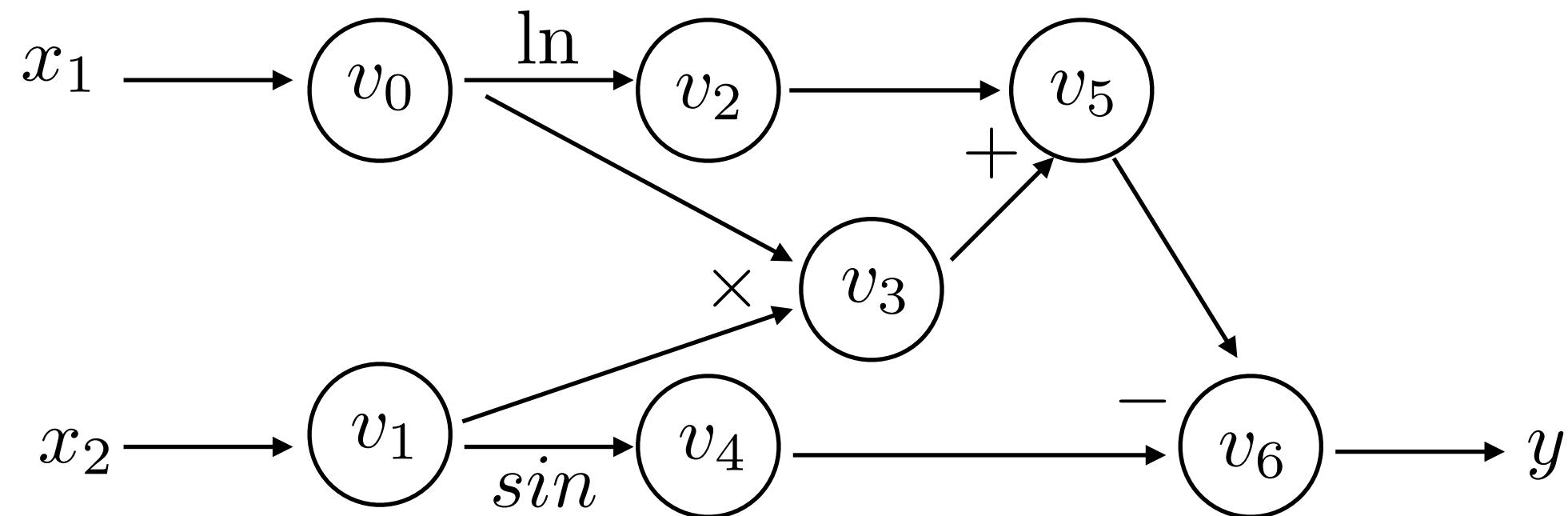
$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

$$1 \times 1 = 1$$

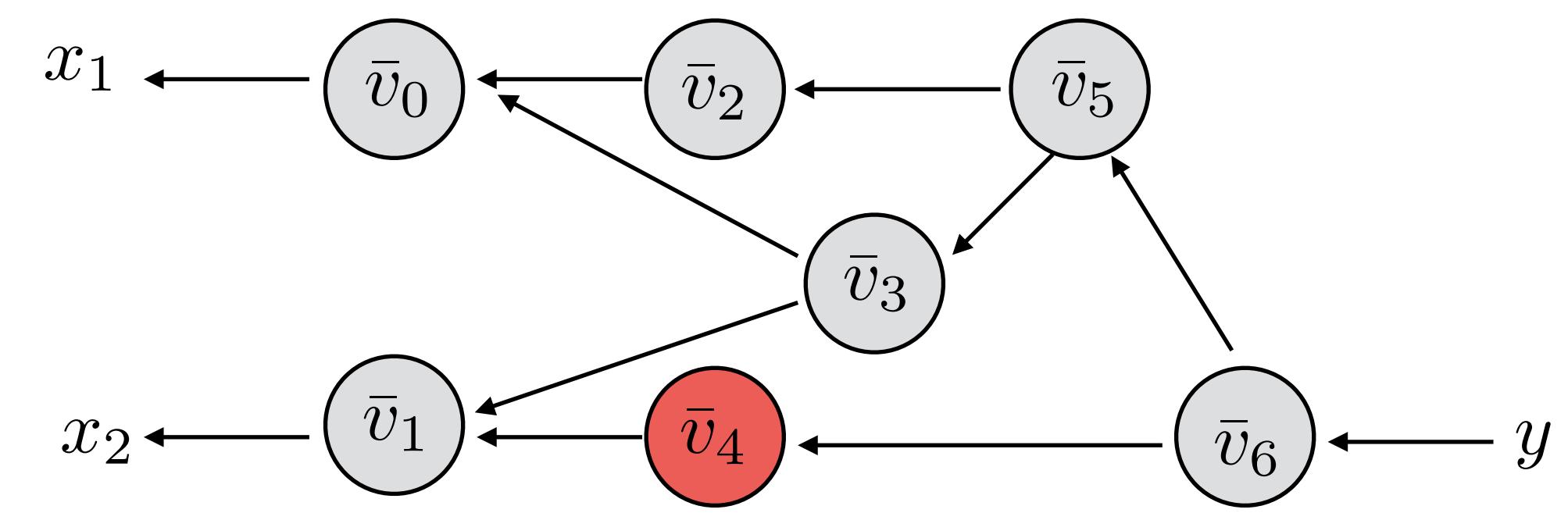
$$1$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

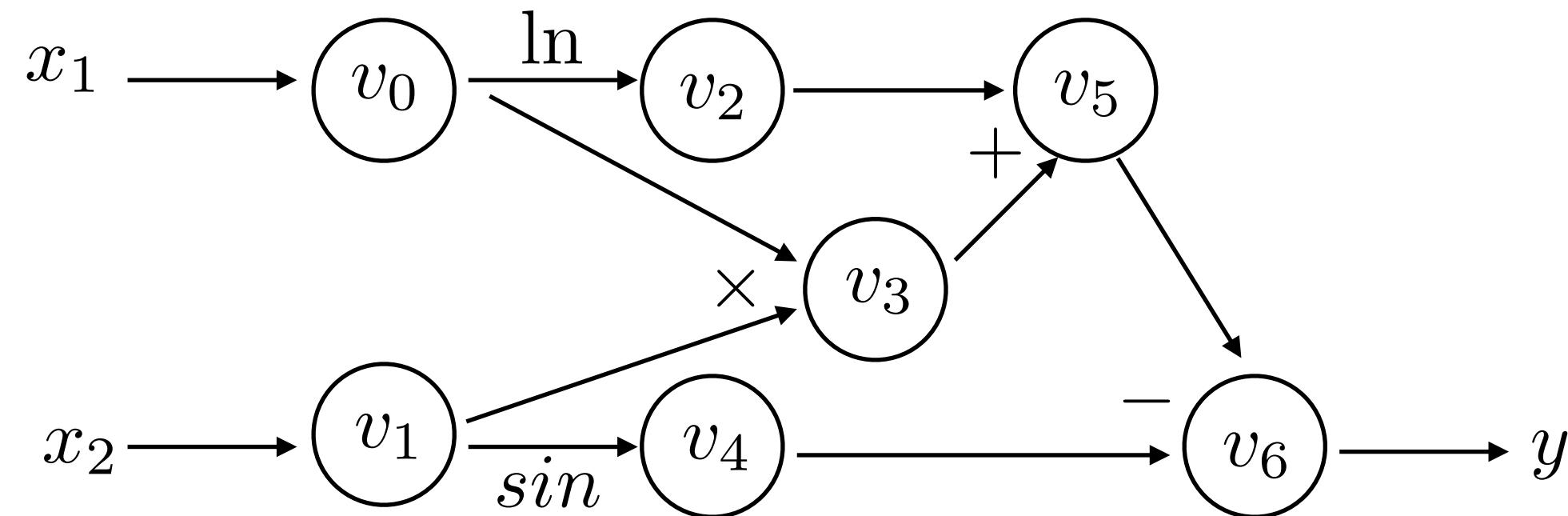
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
<u>$y = v_6$</u>	11.652



Backwards Derivative Trace:

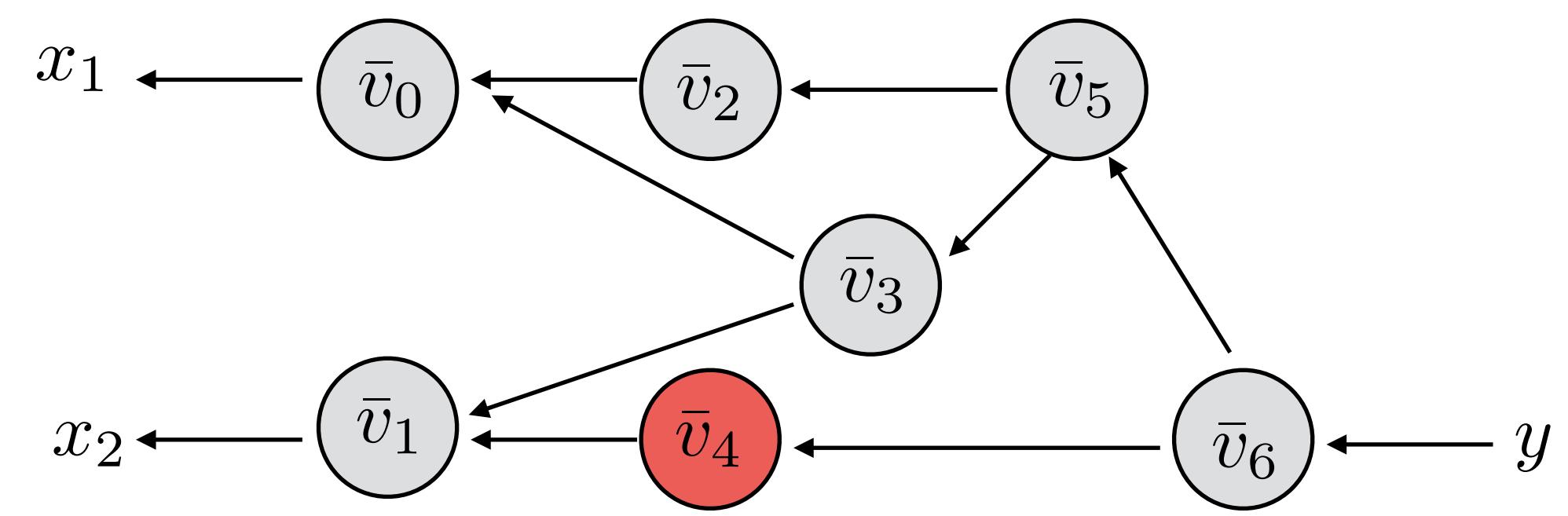
$$\begin{aligned}
 \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \\
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} = 1
 \end{aligned}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
<u>$y = v_6$</u>	11.652



$$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$$

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

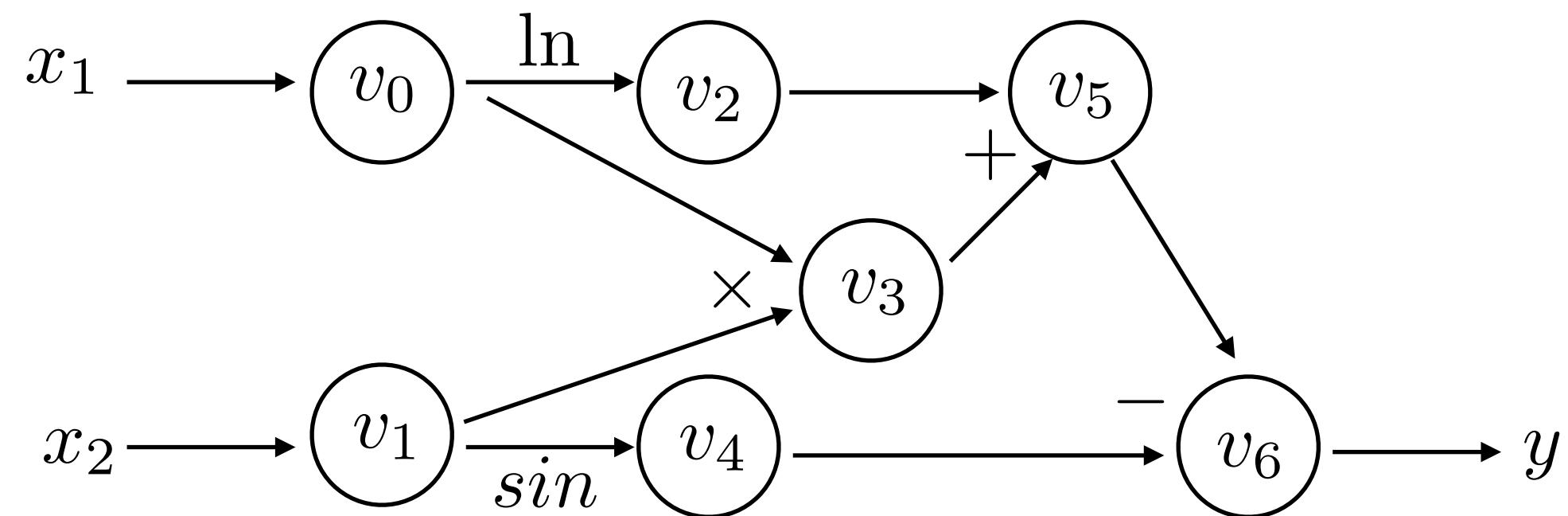
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

$$1 \times -1 = -1$$

$$1 \times 1 = 1$$

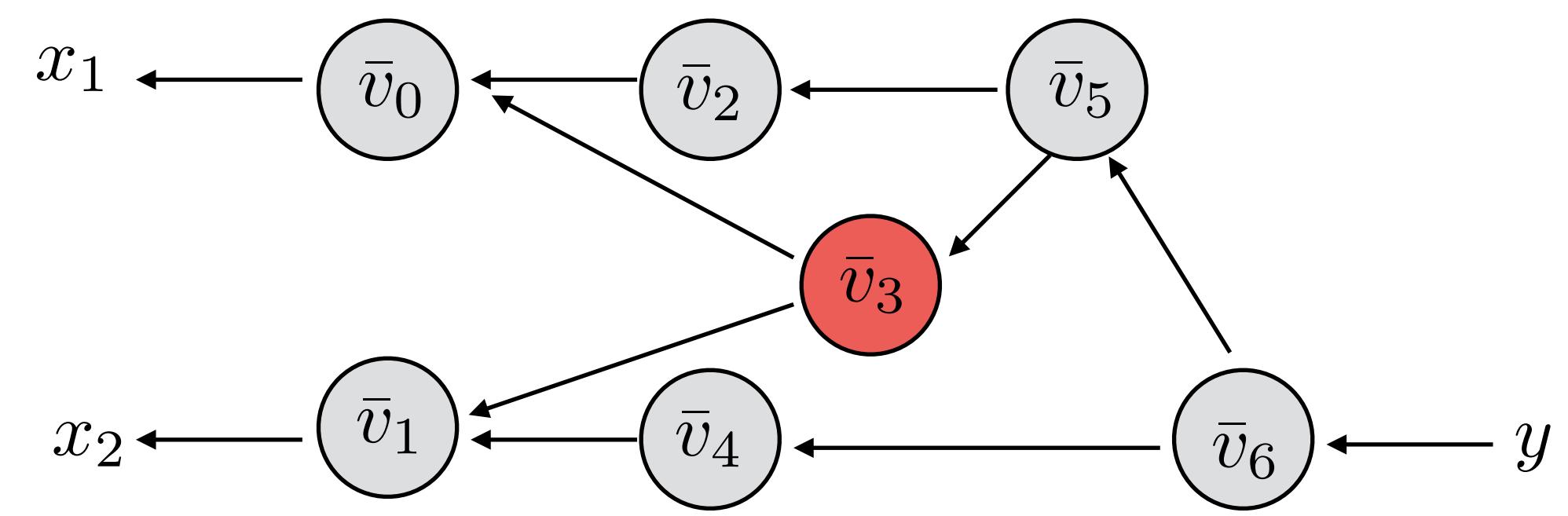
$$1$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

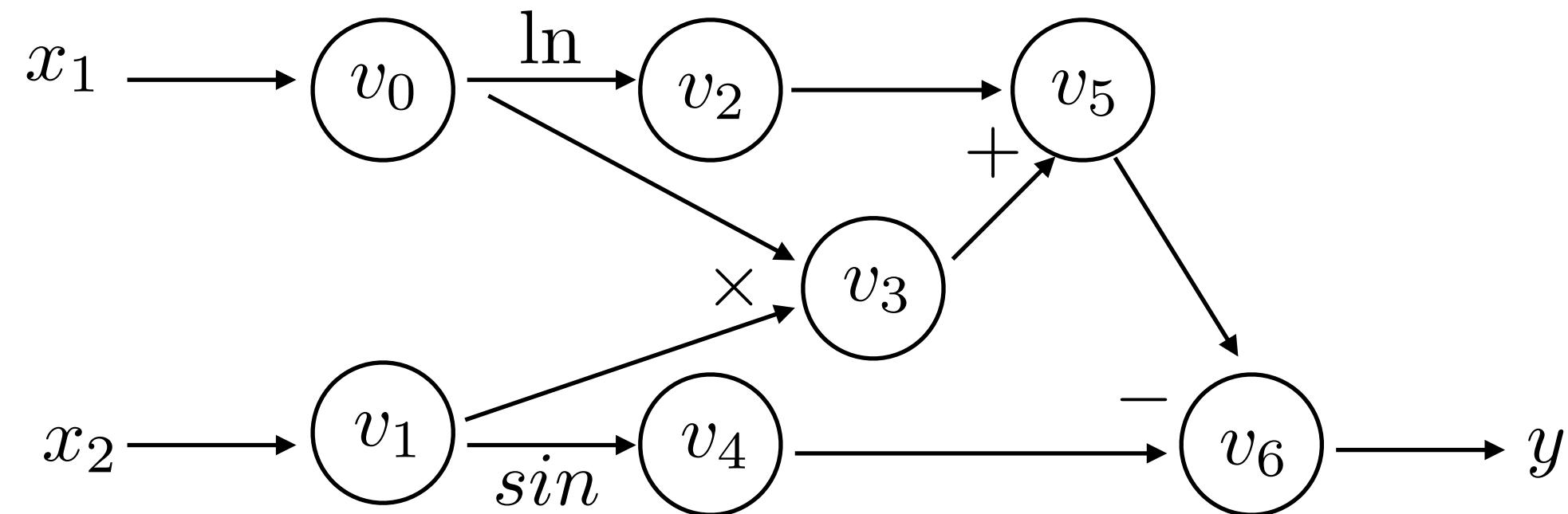
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

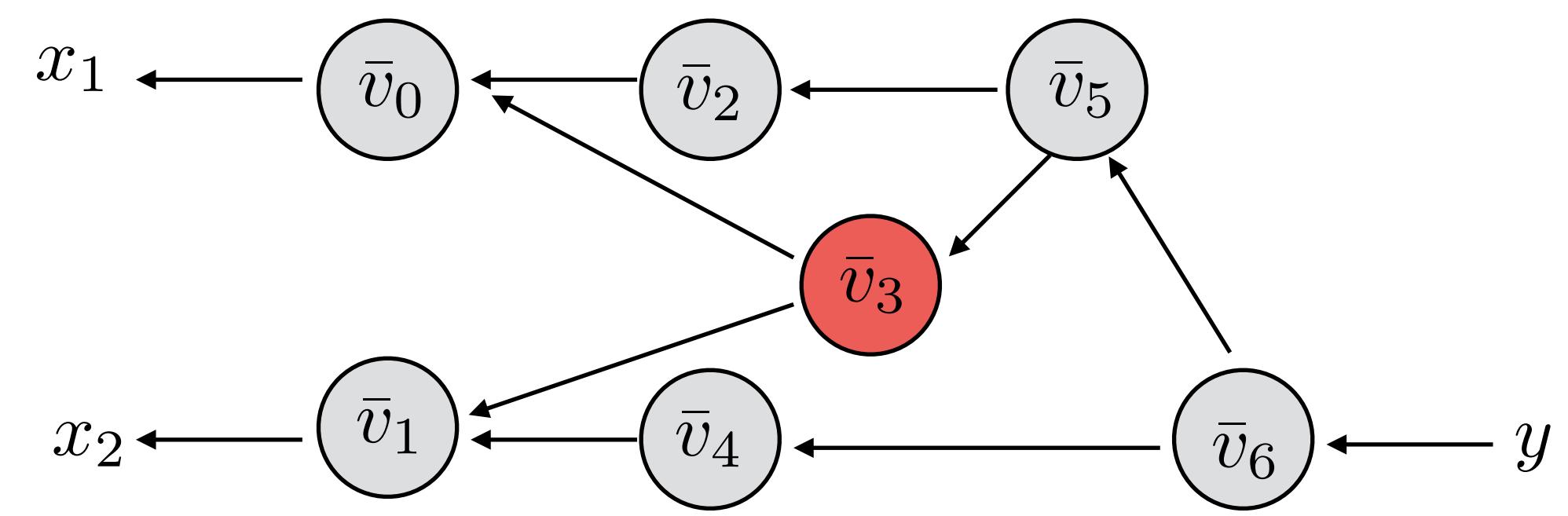
$$\begin{aligned}
 \bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} \\
 \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 = -1 \\
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 = 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1
 \end{aligned}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

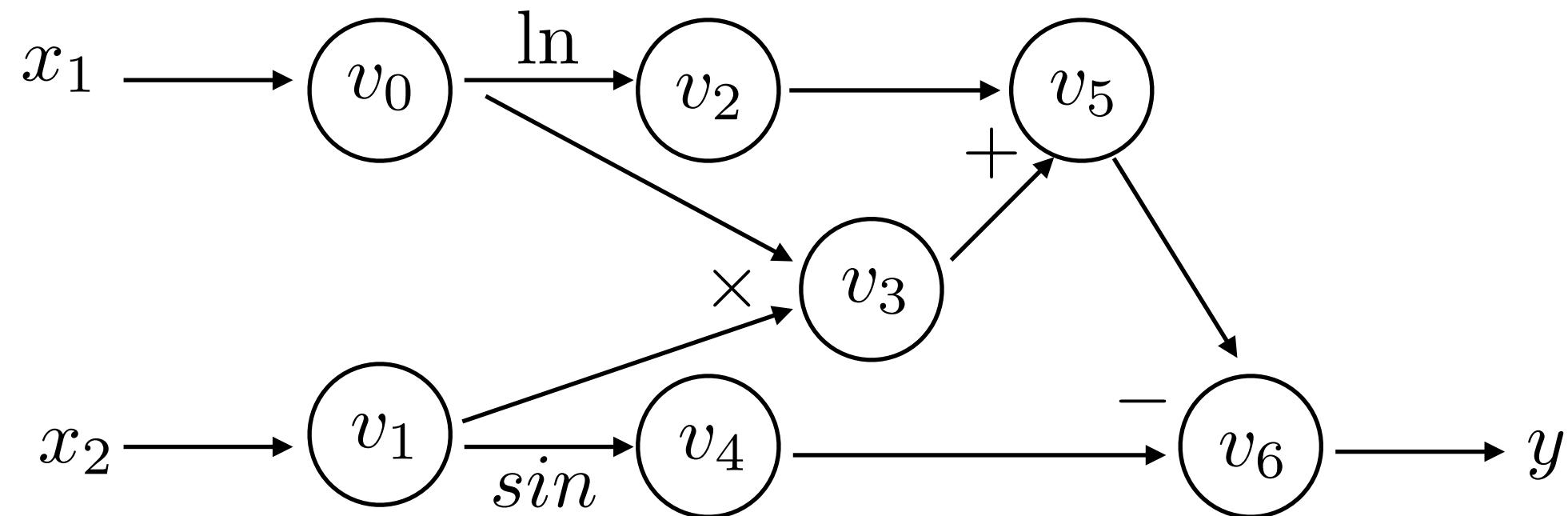
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
<u>$v_6 = v_5 - v_4$</u>	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

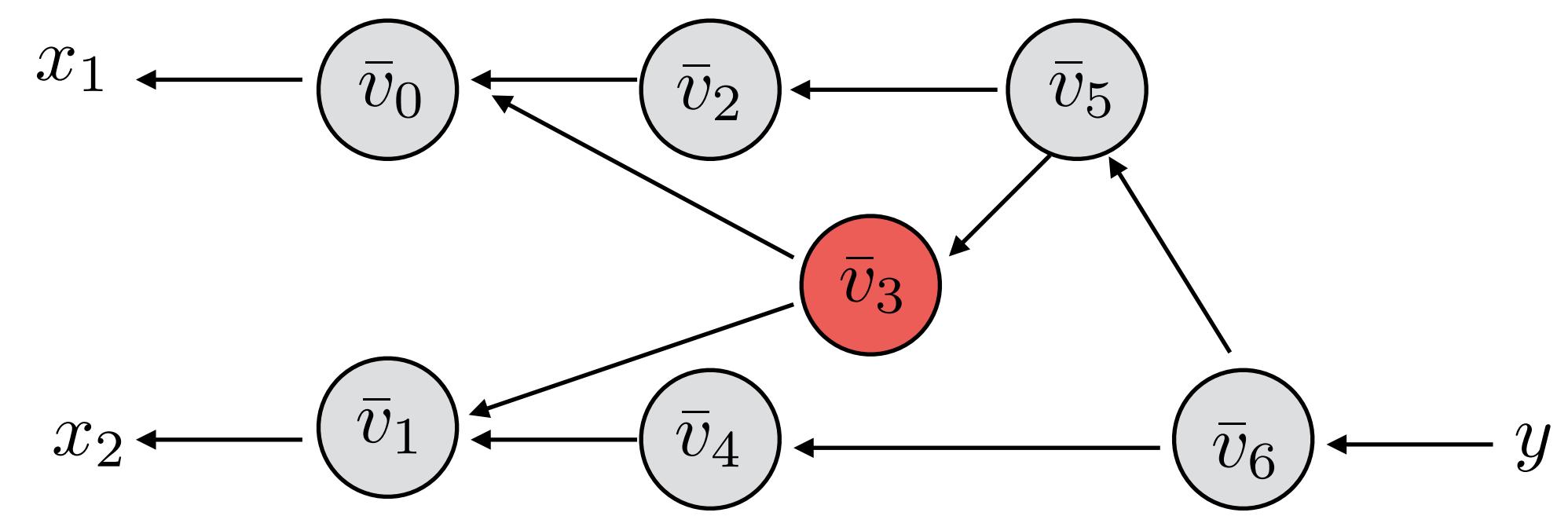
$$\begin{aligned}
 \bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} \\
 \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 = -1 \\
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 = 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1
 \end{aligned}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
<u>$v_5 = v_2 + v_3$</u>	<u>$0.693 + 10 = 10.693$</u>
<u>$v_6 = v_5 - v_4$</u>	<u>$10.693 + 0.959 = 11.652$</u>
$y = v_6$	11.652



$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$$

$$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$$

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

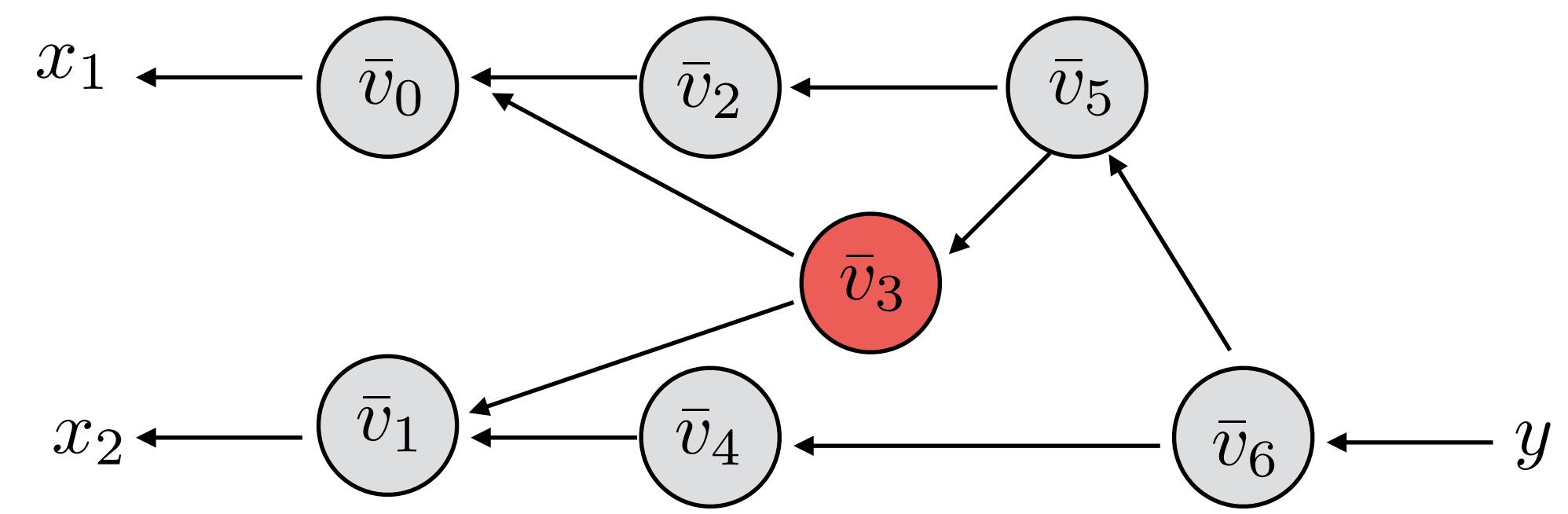
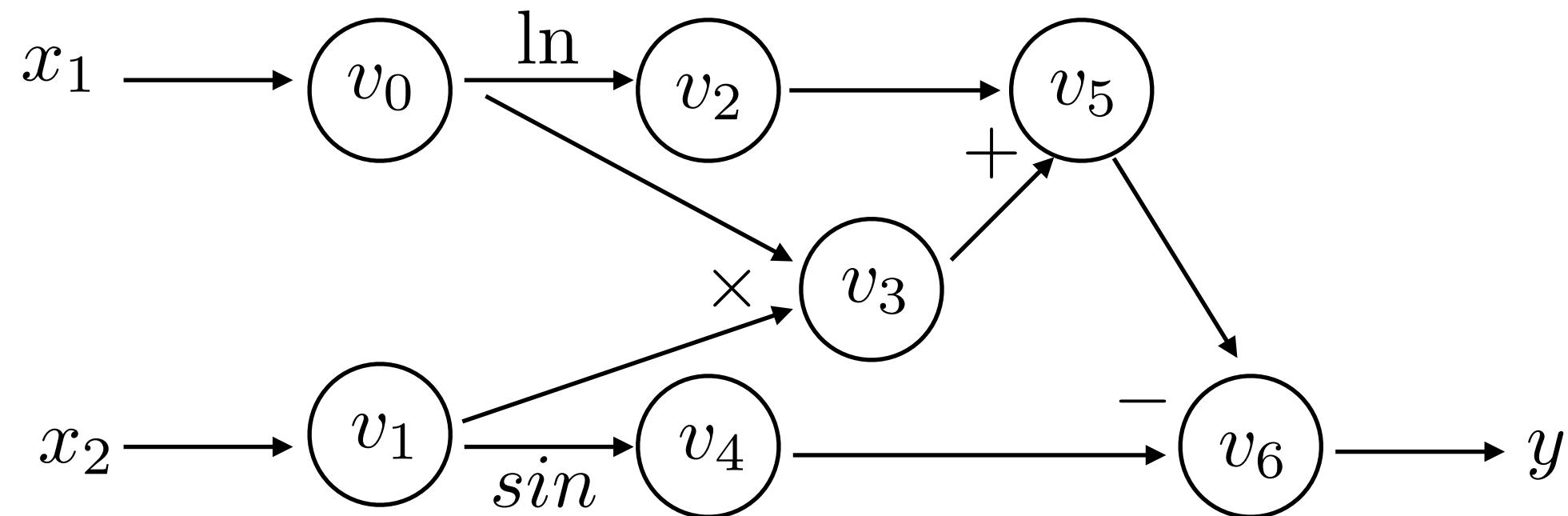
$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

$1 \times -1 = -1$

$1 \times 1 = 1$

1

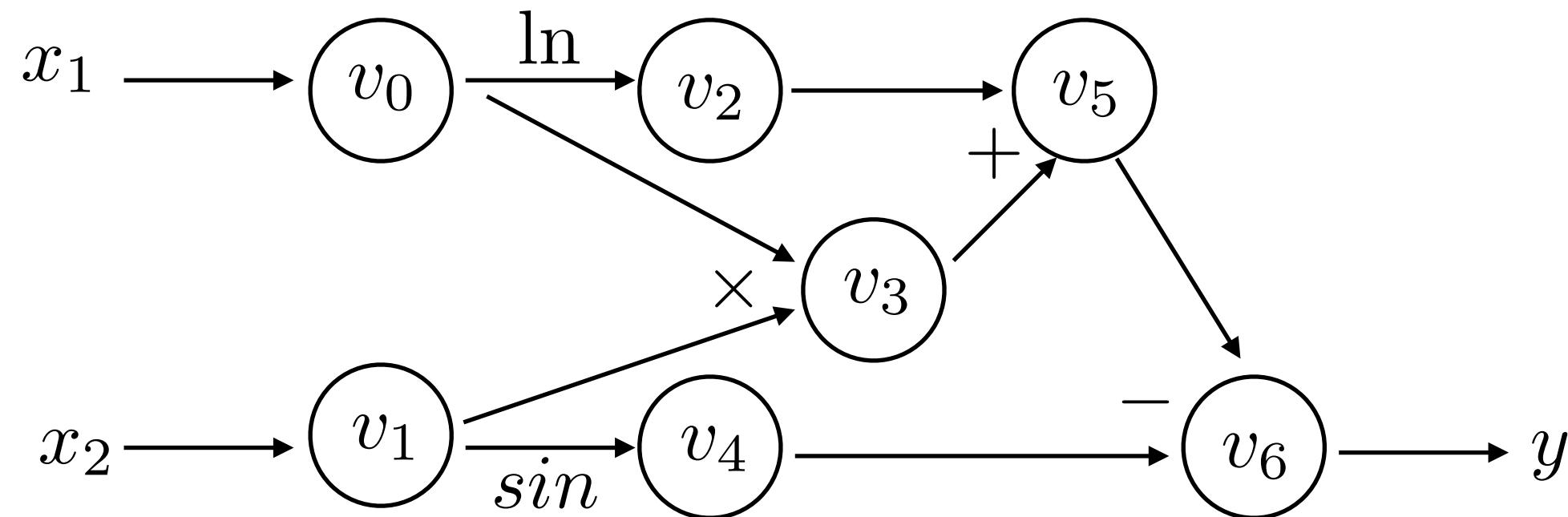
AutoDiff - Reverse Mode



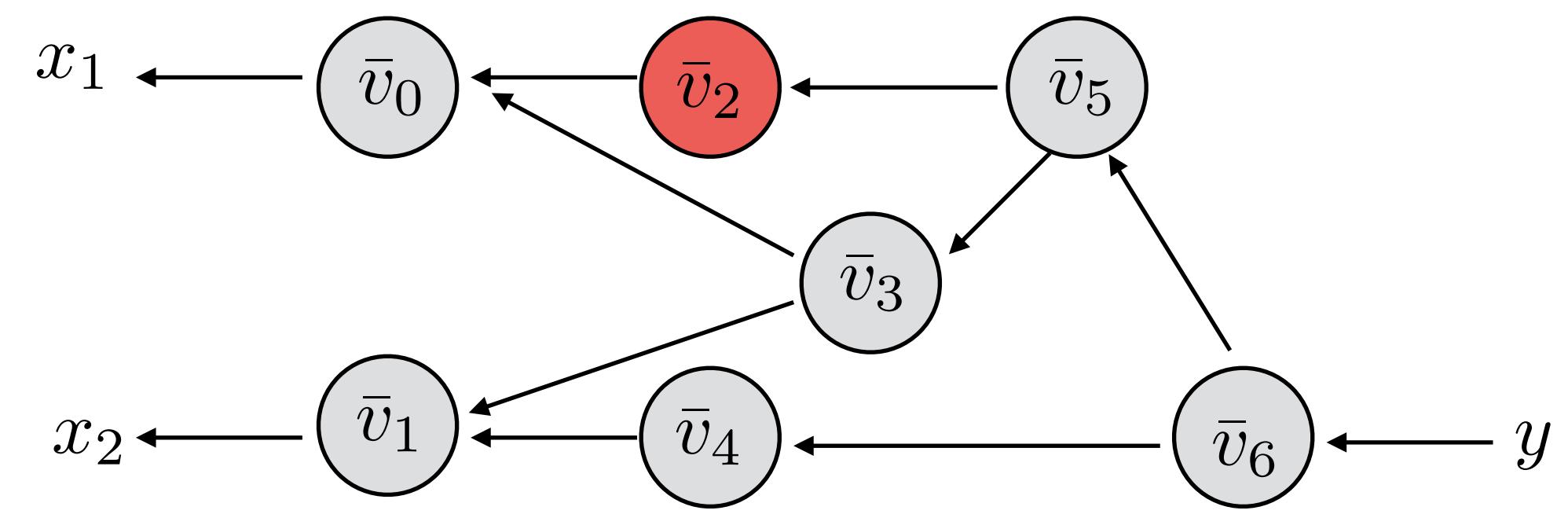
Backwards Derivative Trace:

$$\begin{aligned} \bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) & 1 \times 1 &= 1 \\ \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 &= -1 \\ \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 &= 1 \\ \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1 & \end{aligned}$$

AutoDiff - Reverse Mode



$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



$$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2}$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$$

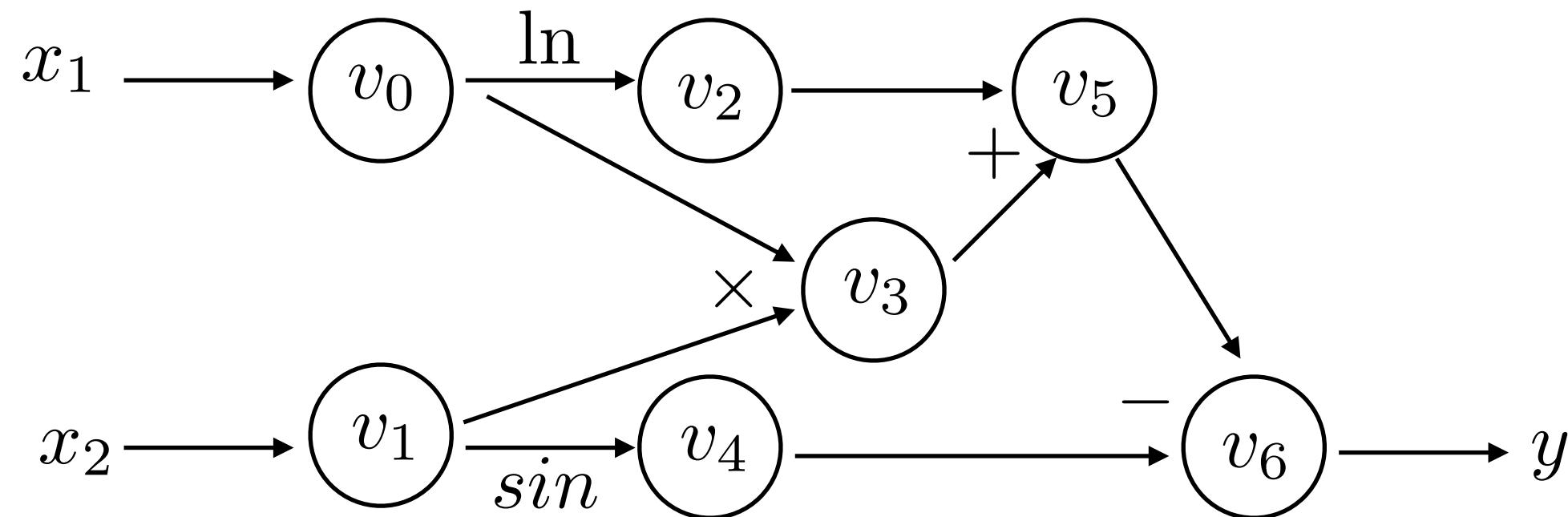
$$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$$

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

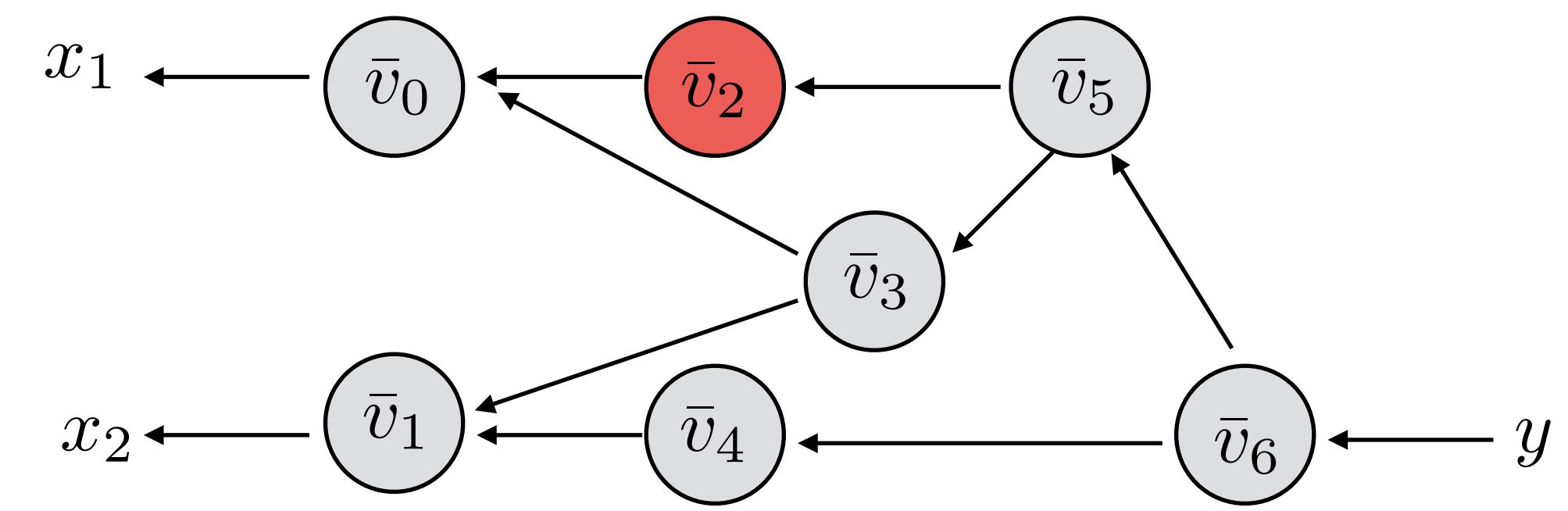
1x1 = 1
1x-1 = -1
1x1 = 1
1

AutoDiff - Reverse Mode



Forward Evaluation Trace:

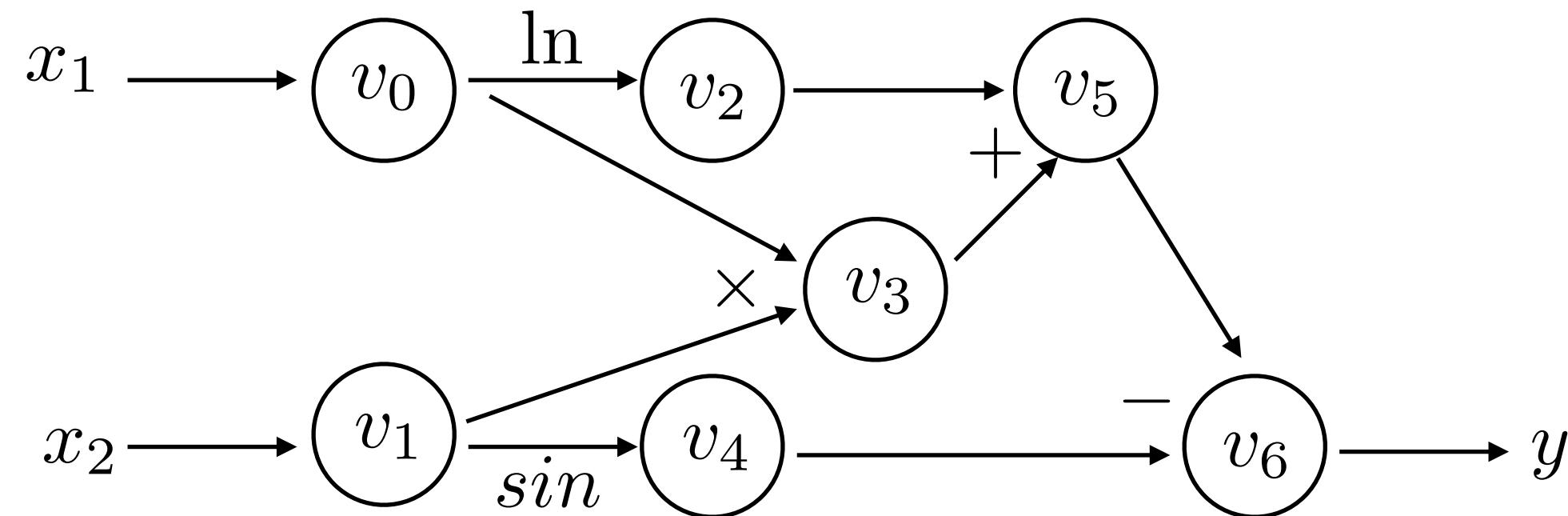
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
<u>$v_5 = v_2 + v_3$</u>	<u>$0.693 + 10 = 10.693$</u>
$v_6 = v_5 - v_4$	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

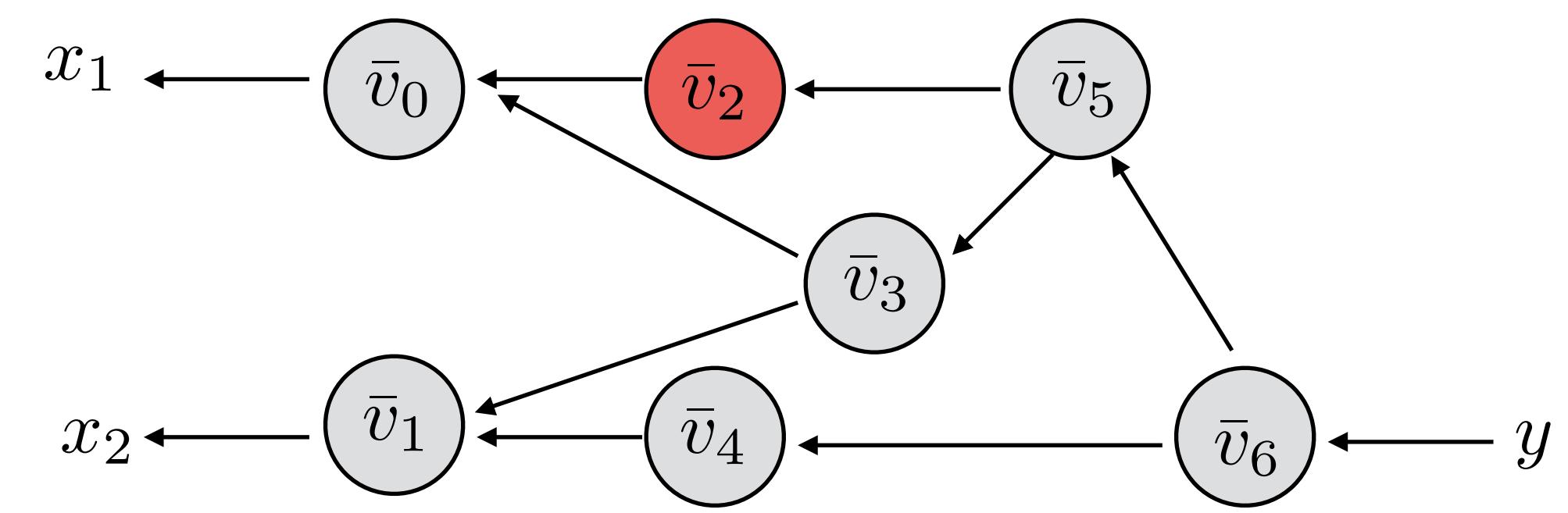
$$\begin{aligned}
 \bar{v}_2 &= \bar{v}_5 \frac{\partial v_5}{\partial v_2} \\
 \bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) & 1 \times 1 = 1 \\
 \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 = -1 \\
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 = 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1
 \end{aligned}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

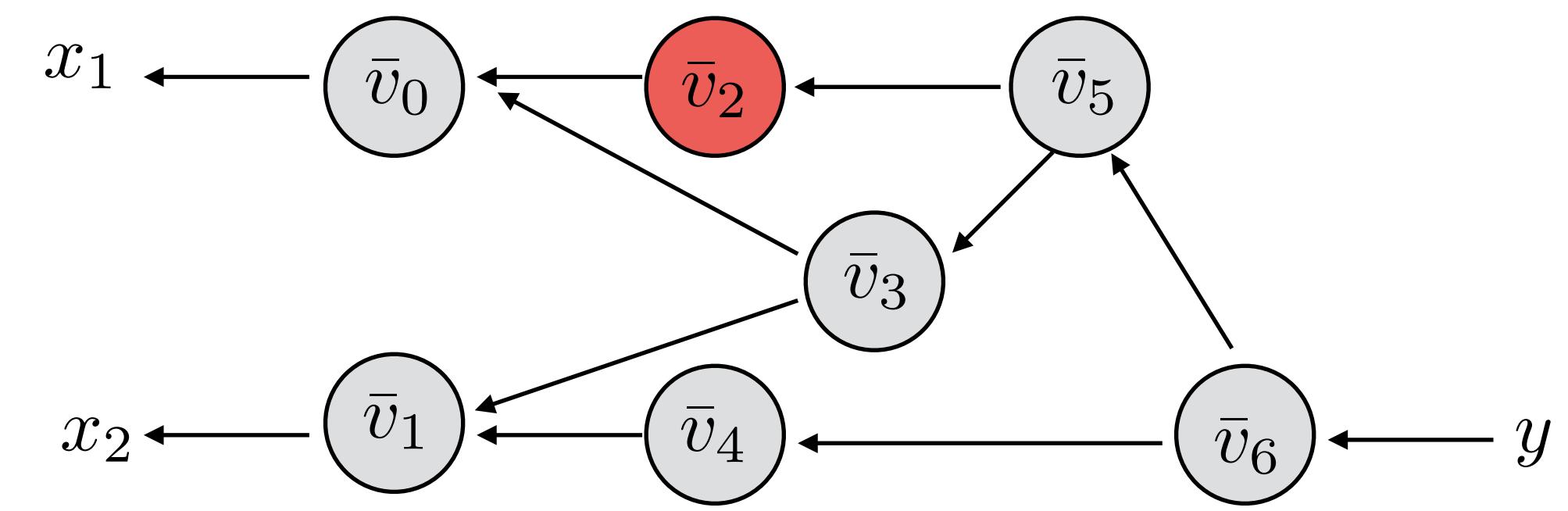
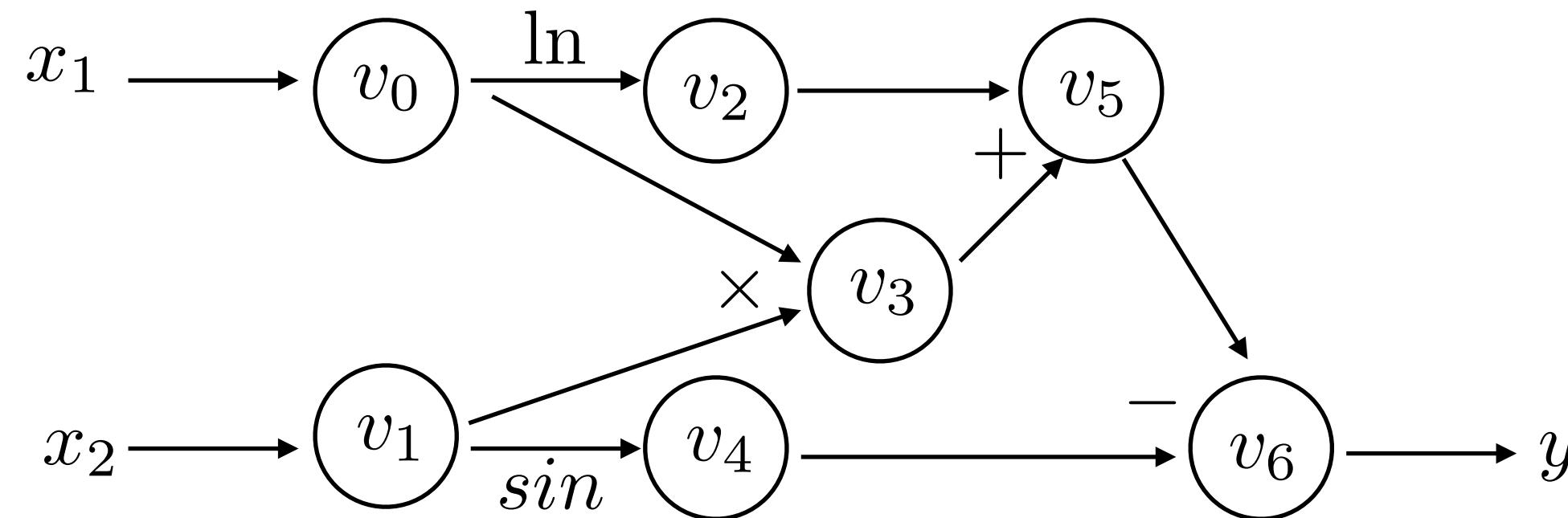
$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
<u>$v_6 = v_5 - v_4$</u>	$10.693 - 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

$$\begin{aligned}
 \bar{v}_2 &= \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1) & 1 \times 1 &= 1 \\
 \bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) & 1 \times 1 &= 1 \\
 \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 &= -1 \\
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 &= 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1 &
 \end{aligned}$$

AutoDiff - Reverse Mode



$$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$$

$$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$$

$$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$$

$$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$$

$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$

1x1 = 1

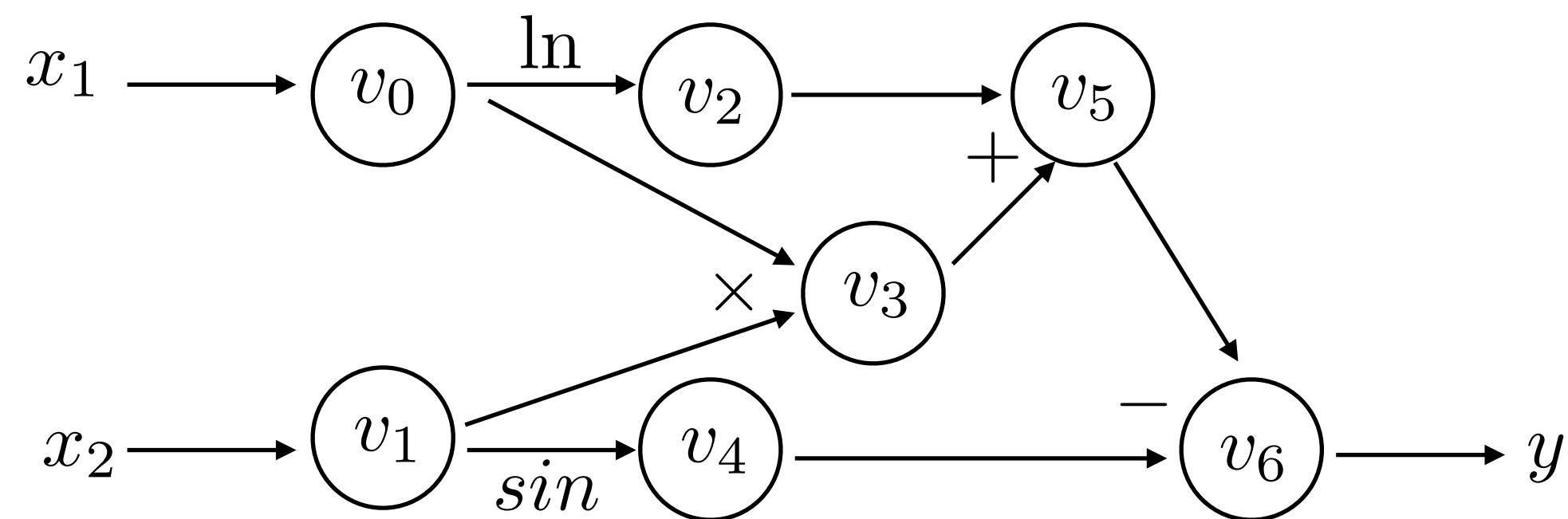
1x1 = 1

1x-1 = -1

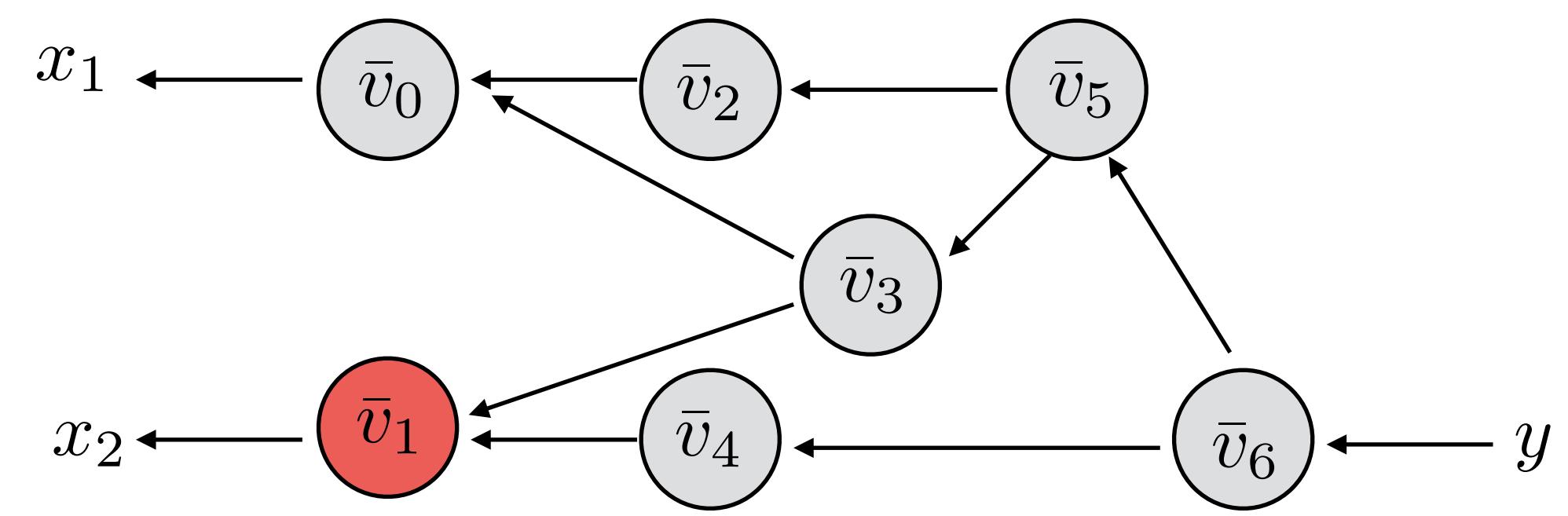
1x1 = 1

1

AutoDiff - Reverse Mode

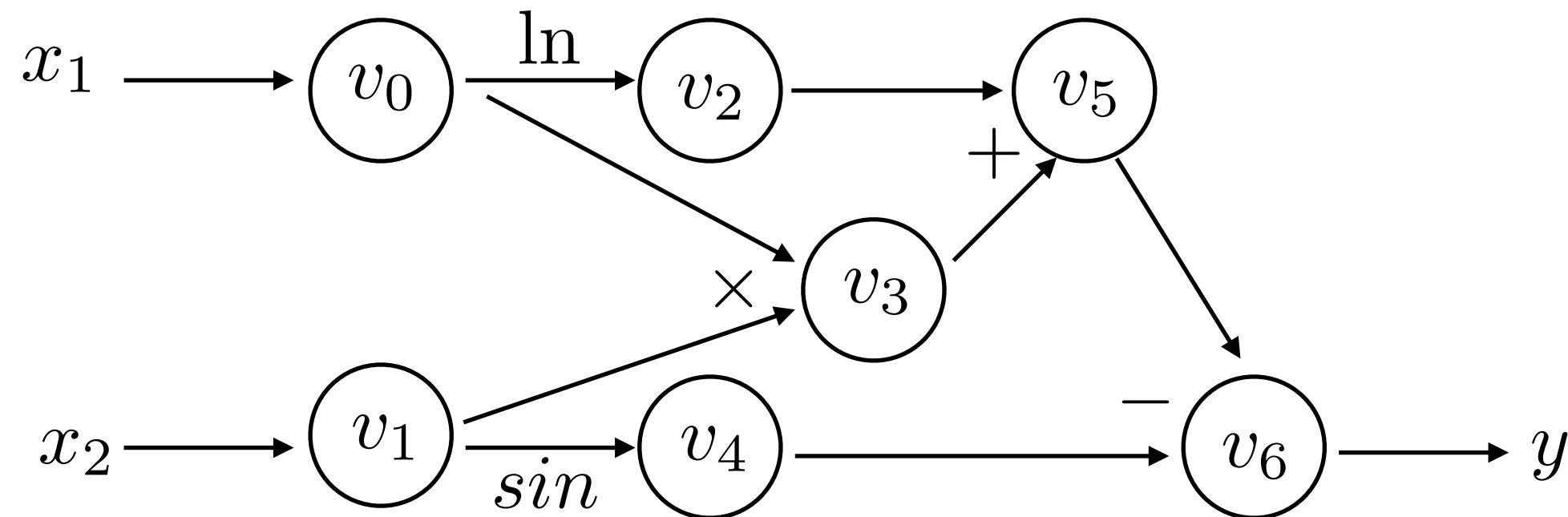


$f(2, 5)$	
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



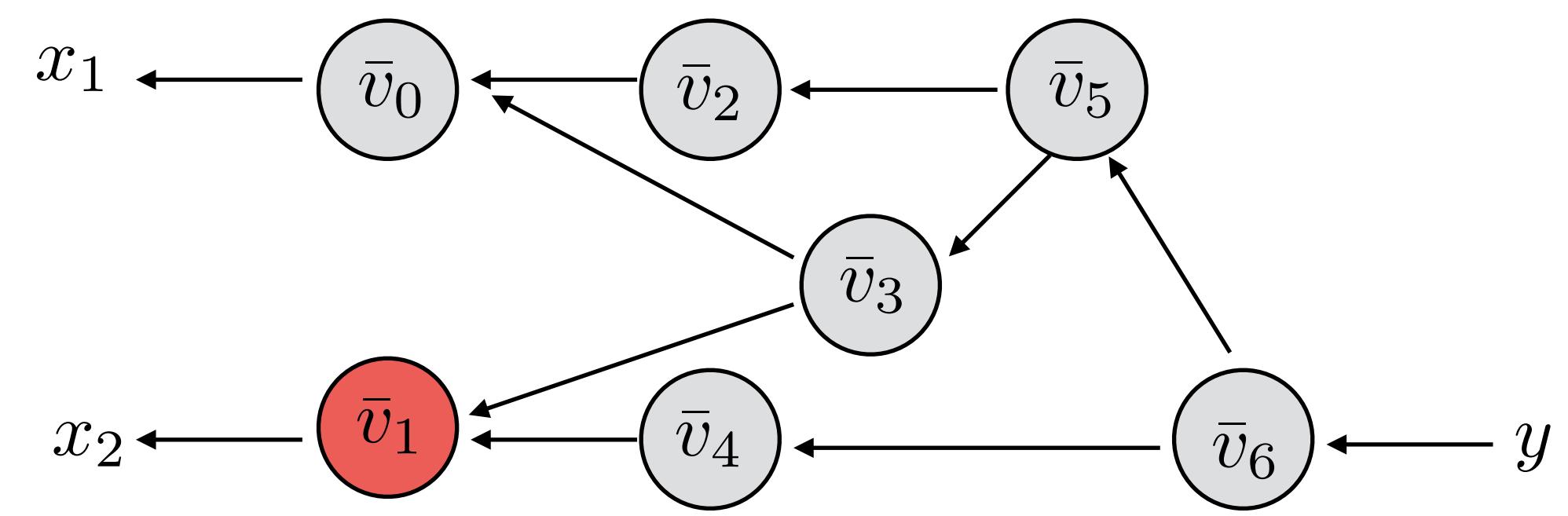
$$\begin{aligned}
 \bar{v}_1 &: \\
 \bar{v}_2 &= \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1) & 1 \times 1 = 1 \\
 \bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) & 1 \times 1 = 1 \\
 \bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 = -1 \\
 \bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 = 1 \\
 \bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1
 \end{aligned}$$

AutoDiff - Reverse Mode



Forward Evaluation Trace:

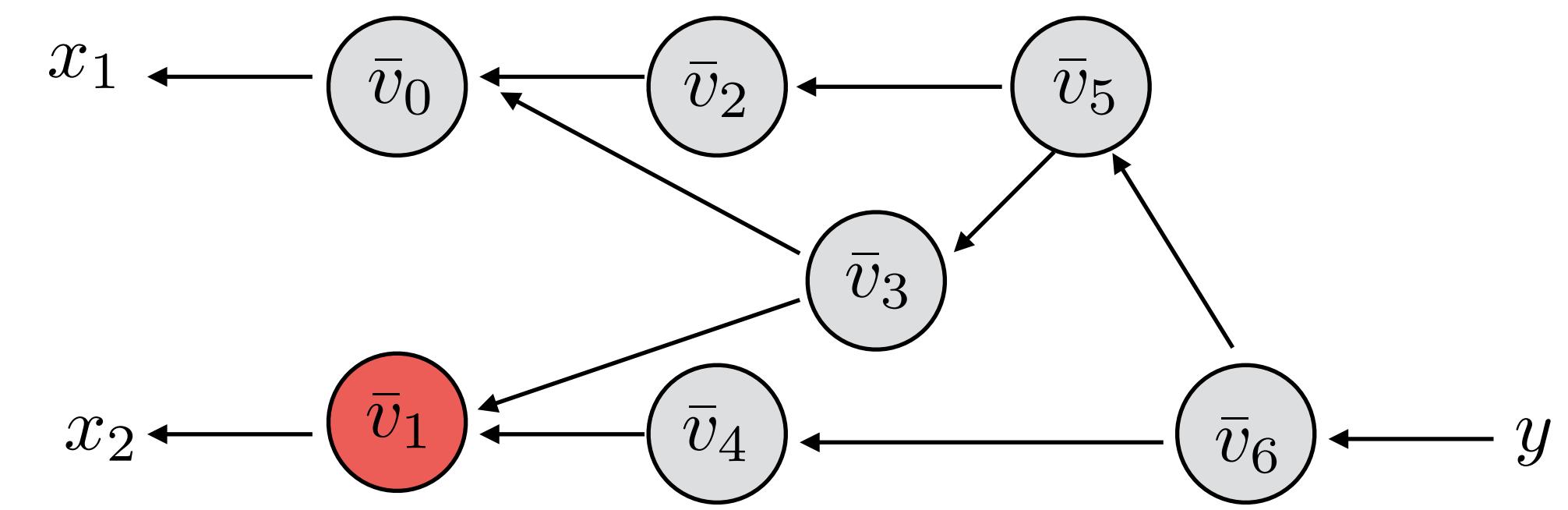
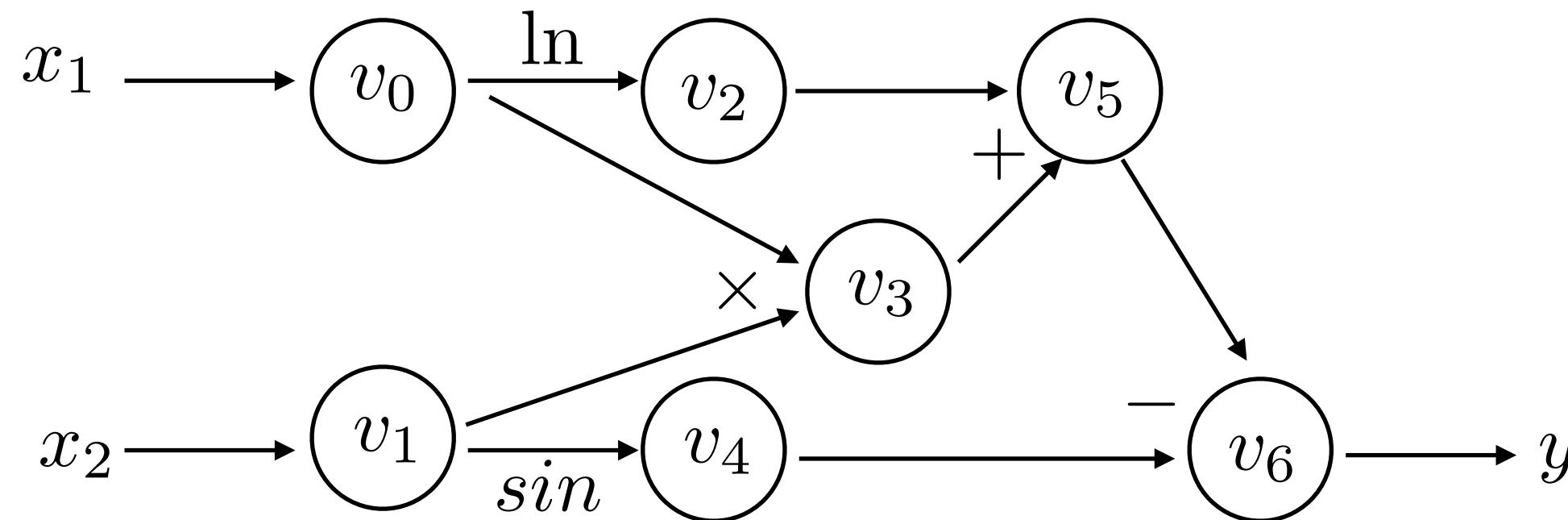
	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$1 \times 1 = 1$
$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$	$1 \times -1 = -1$
$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$	$1 \times 1 = 1$
$\bar{v}_6 = \frac{\partial y}{\partial v_6}$	1

AutoDiff - Reverse Mode

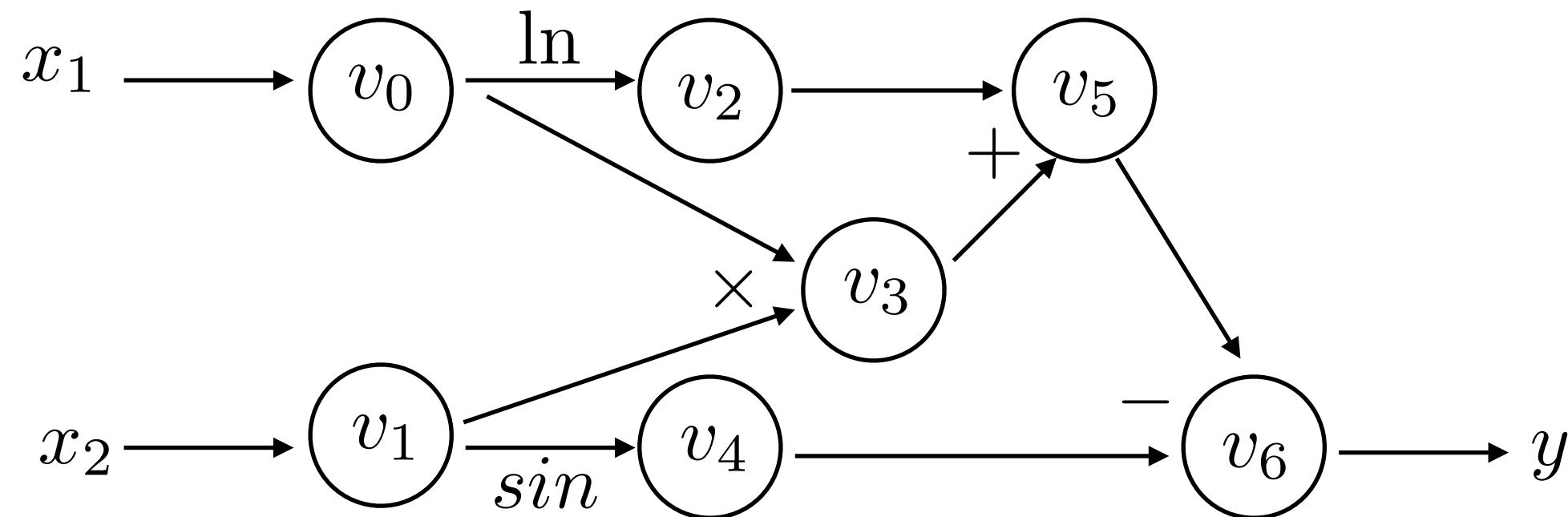


Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
<u>$v_5 = v_2 + v_3$</u>	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652

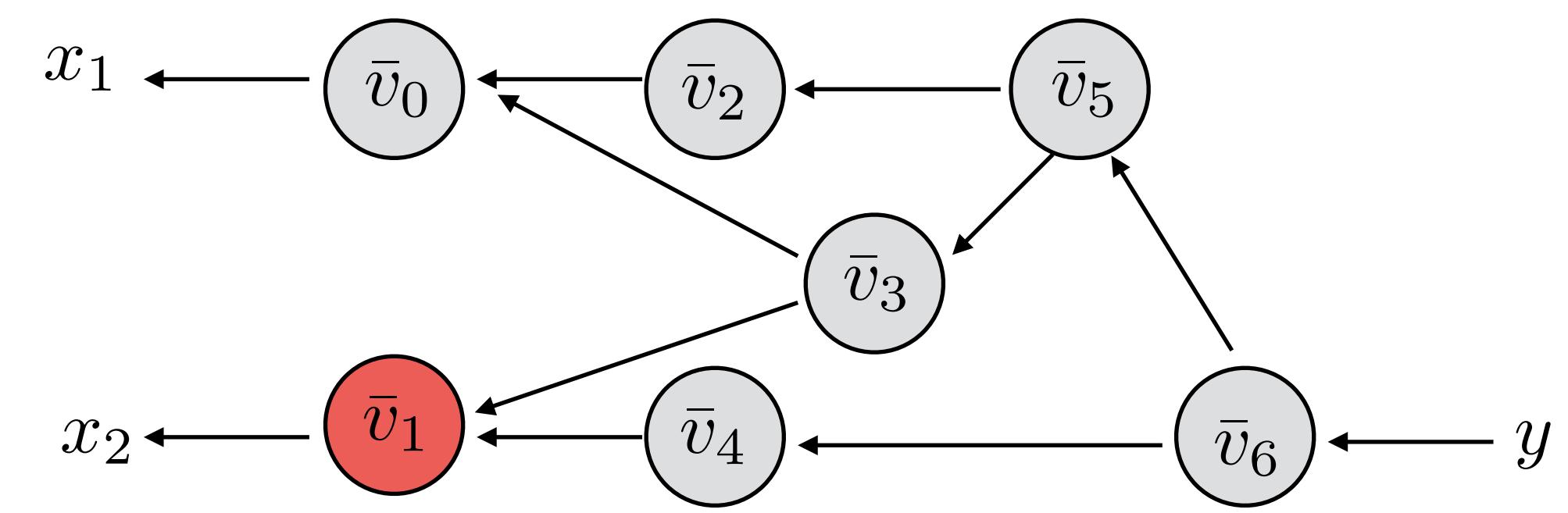
$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$1 \times 1 = 1$
$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$	$1 \times -1 = -1$
$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$	$1 \times 1 = 1$
$\bar{v}_6 = \frac{\partial y}{\partial v_6}$	1

AutoDiff - Reverse Mode



Forward Evaluation Trace:

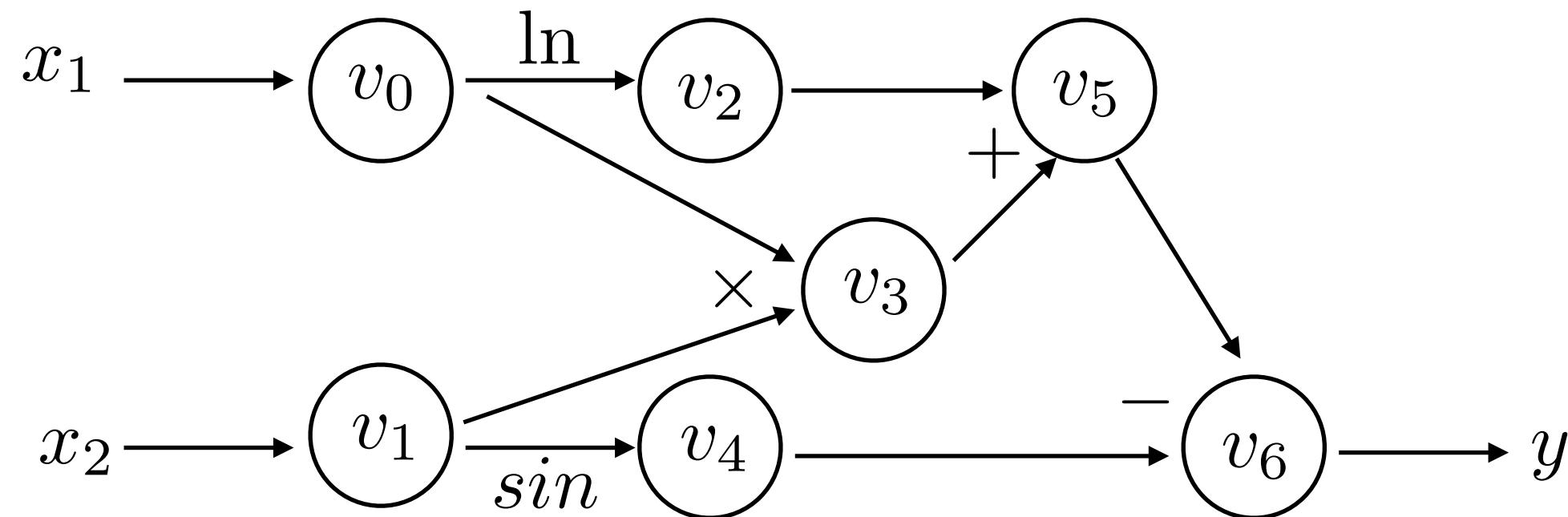
	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
<u>$v_5 = v_2 + v_3$</u>	$0.693 + 10 = 10.693$
<u>$v_6 = v_5 - v_4$</u>	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

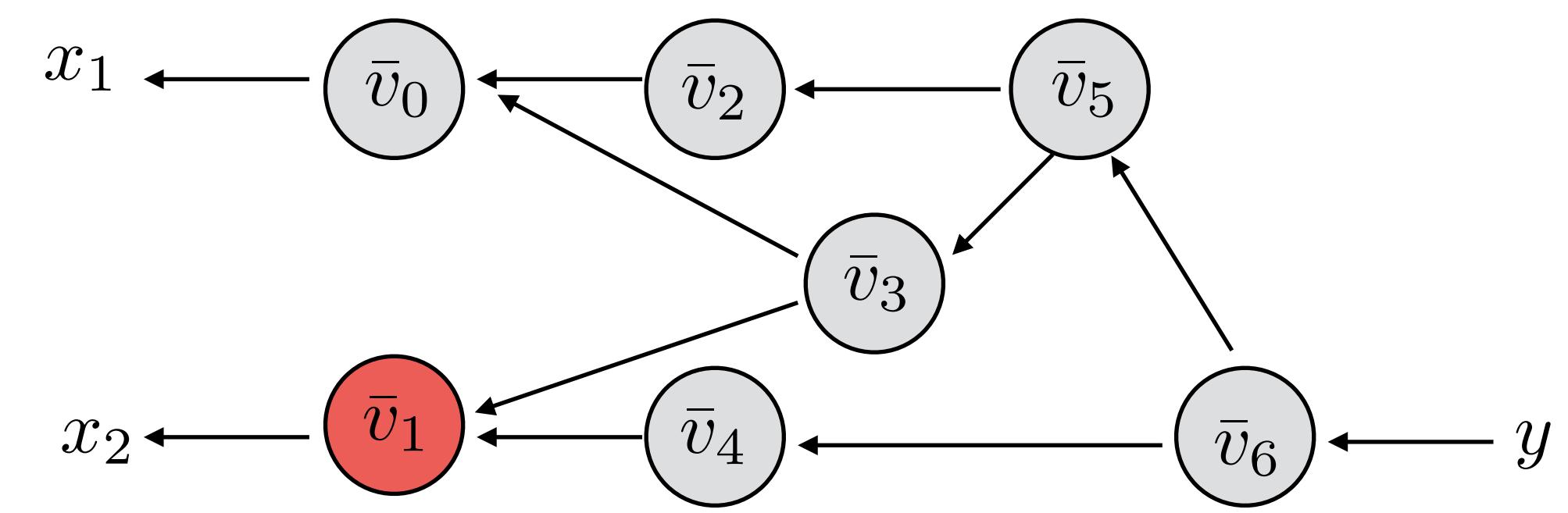
$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1)$	$1 \times 1 = 1$
$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$	$1 \times -1 = -1$
$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$	$1 \times 1 = 1$
$\bar{v}_6 = \frac{\partial y}{\partial v_6}$	1

AutoDiff - Reverse Mode



Forward Evaluation Trace:

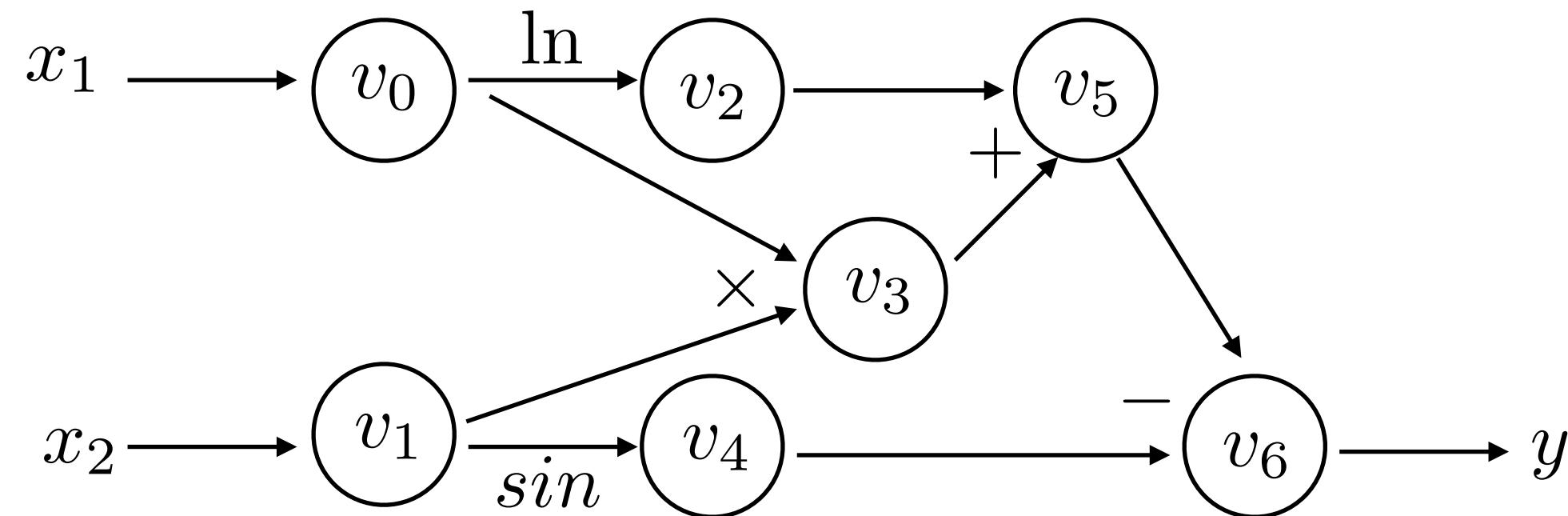
	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
<u>$v_4 = \sin(v_1)$</u>	$\sin(5) = 0.959$
<u>$v_5 = v_2 + v_3$</u>	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

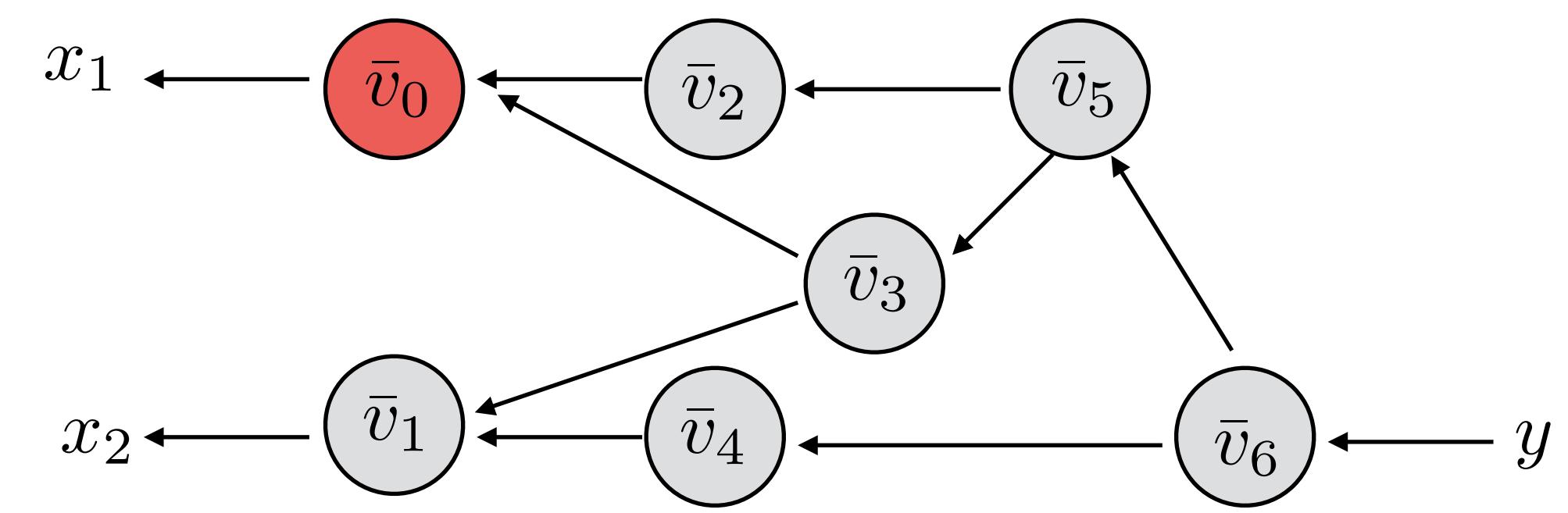
$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1)$	1.716
$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$	$1 \times -1 = -1$
$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$	$1 \times 1 = 1$
$\bar{v}_6 = \frac{\partial y}{\partial v_6}$	1

AutoDiff - Reverse Mode



Forward Evaluation Trace:

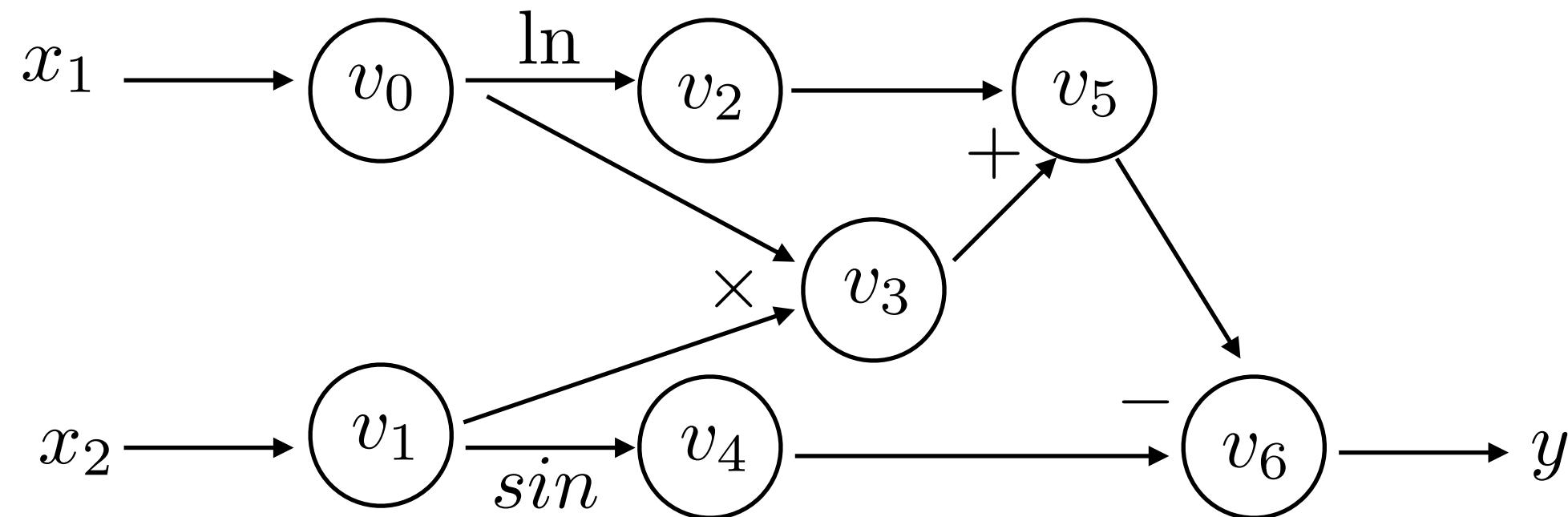
	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



Backwards Derivative Trace:

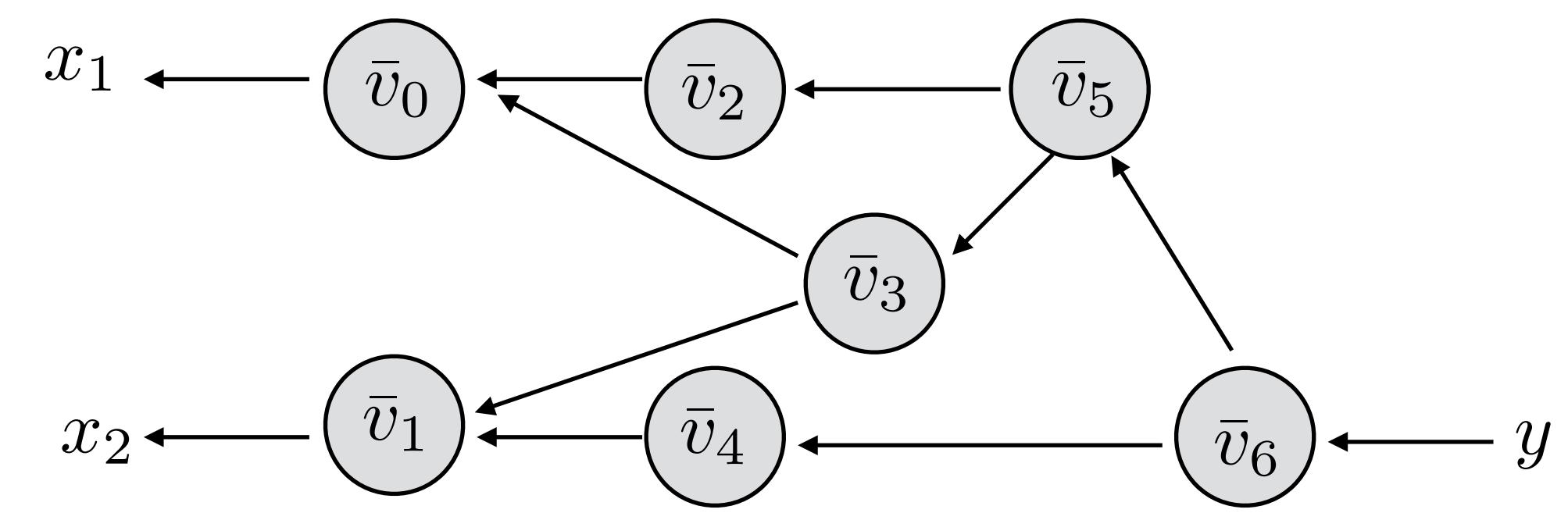
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0} + \bar{v}_2 \frac{\partial v_2}{\partial v_0} = \bar{v}_3 v_1 + \bar{v}_2 \frac{1}{v_0}$	5.5
$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1)$	1.716
$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$	$1 \times -1 = -1$
$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$	$1 \times 1 = 1$
$\bar{v}_6 = \frac{\partial y}{\partial v_6}$	1

AutoDiff - Reverse Mode



Forward Evaluation Trace:

	$f(2, 5)$
$v_0 = x_1$	2
$v_1 = x_2$	5
$v_2 = \ln(v_0)$	$\ln(2) = 0.693$
$v_3 = v_0 \cdot v_1$	$2 \times 5 = 10$
$v_4 = \sin(v_1)$	$\sin(5) = 0.959$
$v_5 = v_2 + v_3$	$0.693 + 10 = 10.693$
$v_6 = v_5 - v_4$	$10.693 + 0.959 = 11.652$
$y = v_6$	11.652



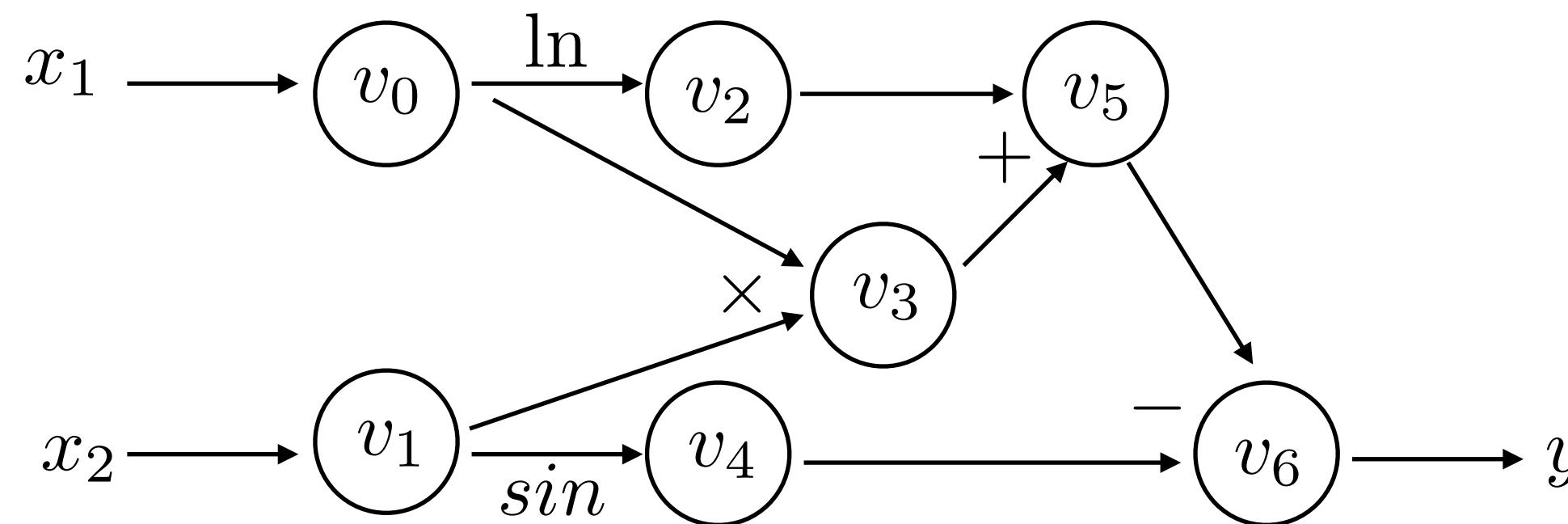
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0} + \bar{v}_2 \frac{\partial v_2}{\partial v_0} = \bar{v}_3 v_1 + \bar{v}_2 \frac{1}{v_0}$	5.5
$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1)$	1.716
$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$	$1 \times 1 = 1$
$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$	$1 \times -1 = -1$
$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$	$1 \times 1 = 1$
$\bar{v}_6 = \frac{\partial y}{\partial v_6}$	1

Automatic Differentiation (AutoDiff)

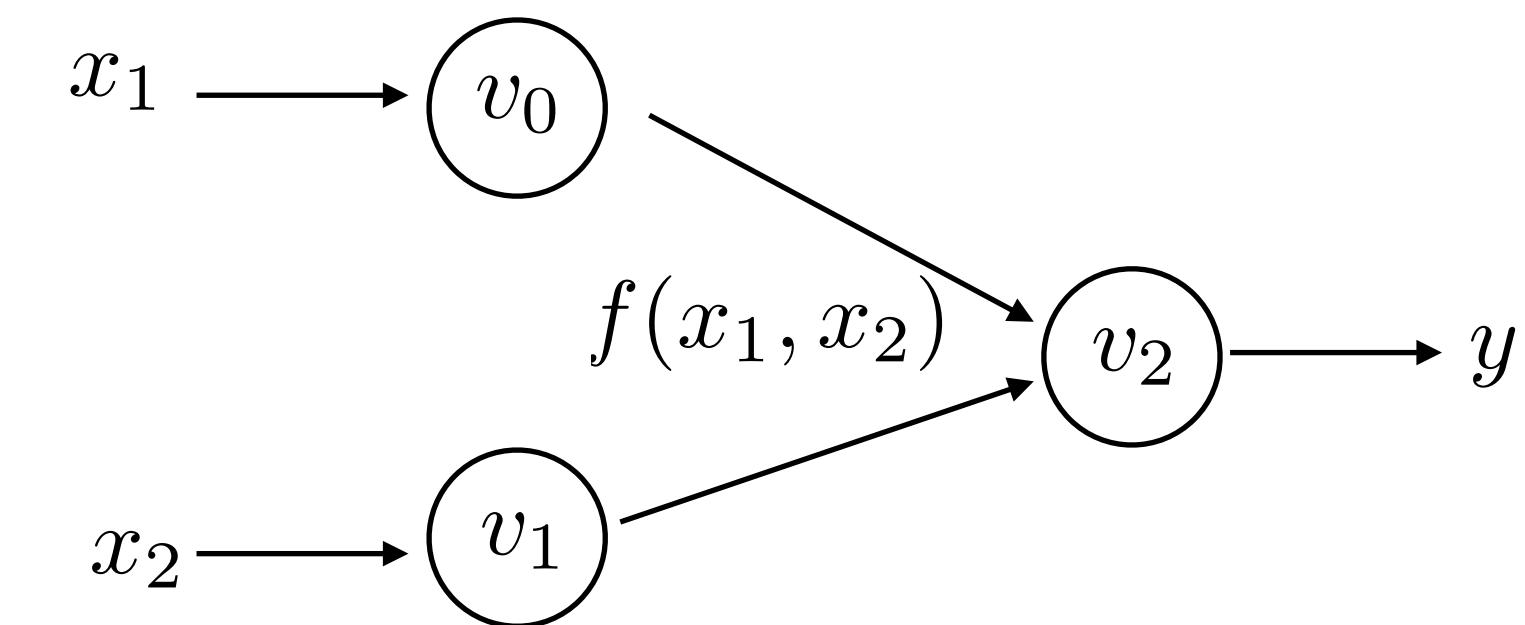
$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

AutoDiff can be done at various **granularities**

Elementary function granularity:



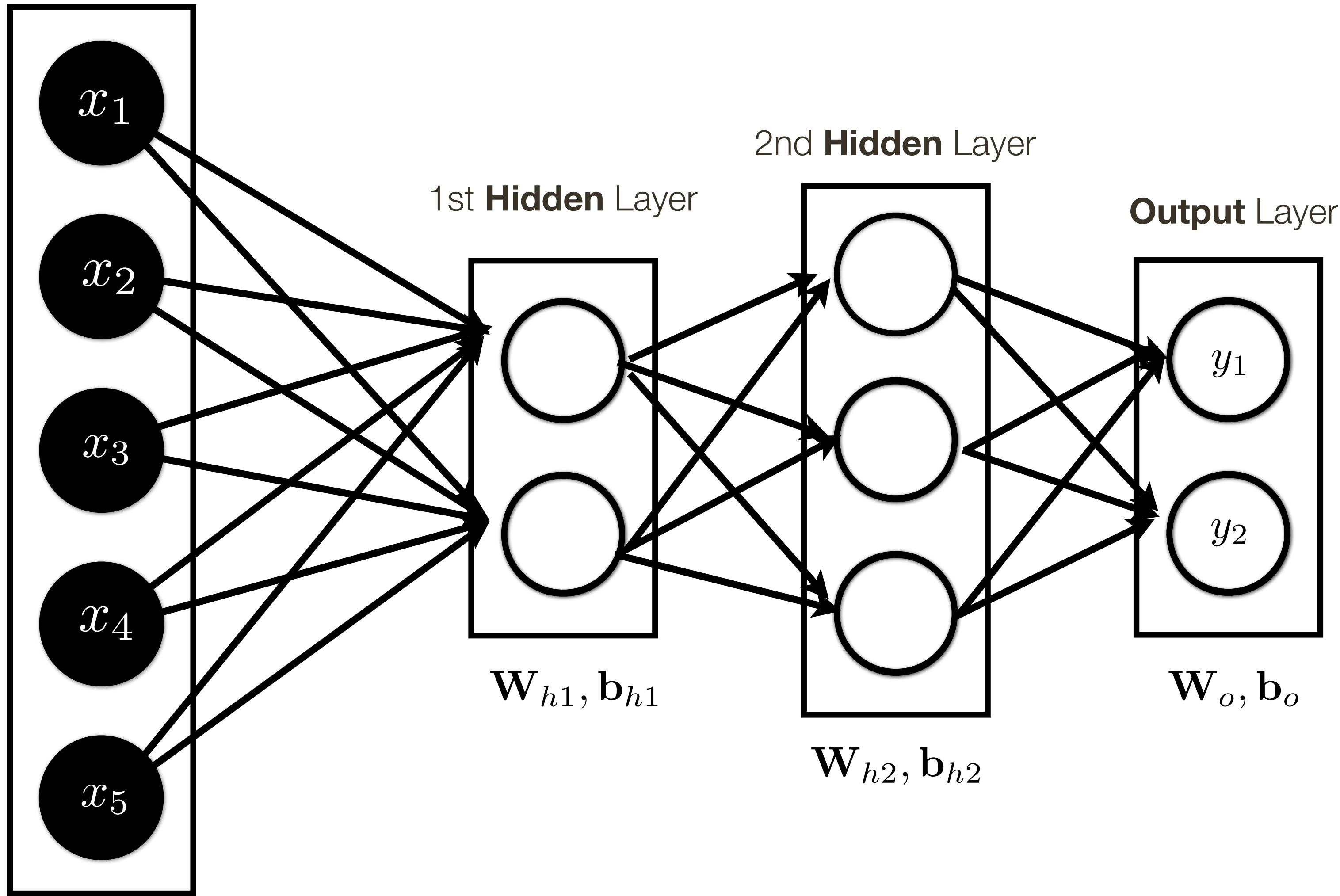
Complex function granularity:



Backpropagation Practical Issues

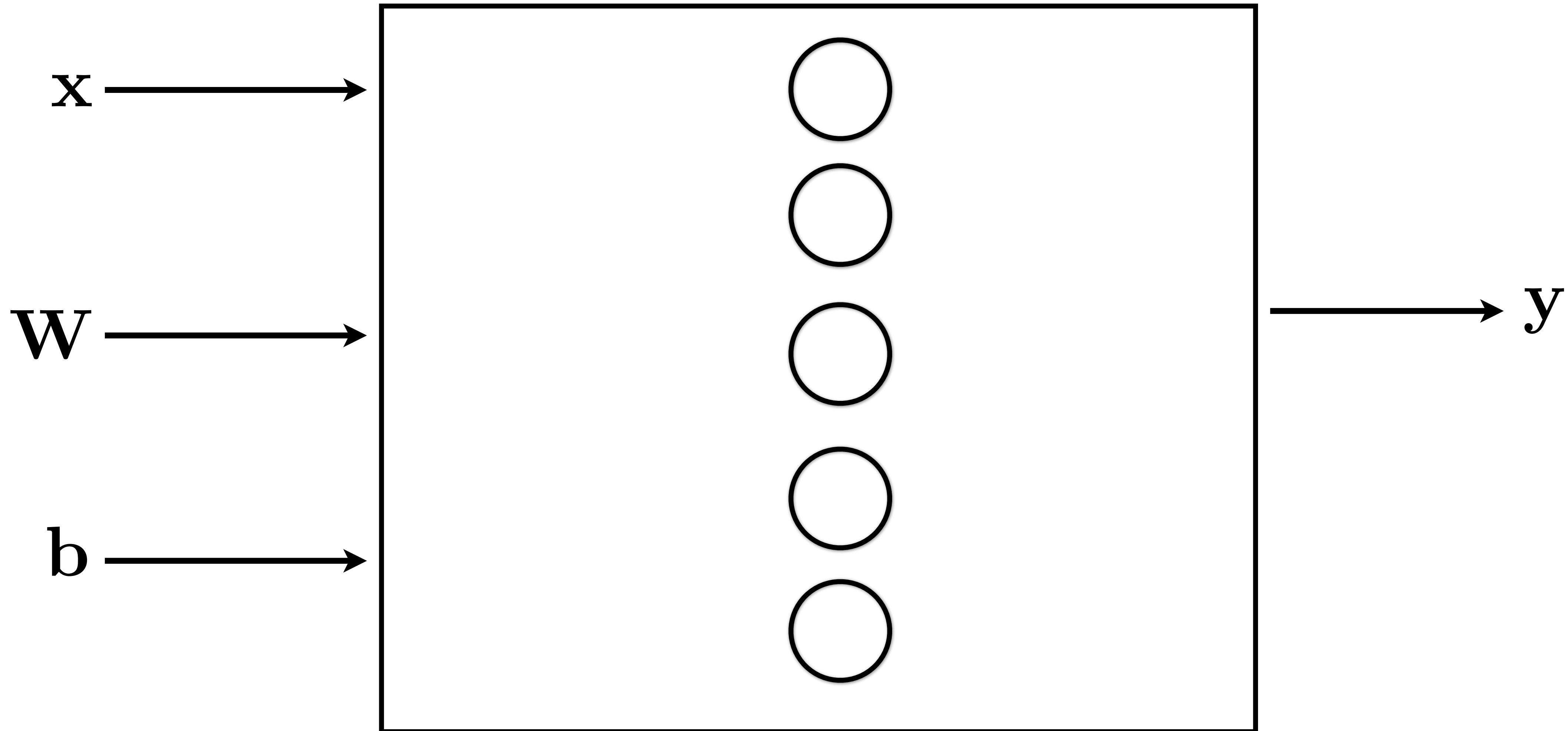
Input Layer

Easier to deal with in **vector form**



Backpropagation Practical Issues

$$y = f(\mathbf{W}, \mathbf{b}, \mathbf{x}) = \text{sigmoid}(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

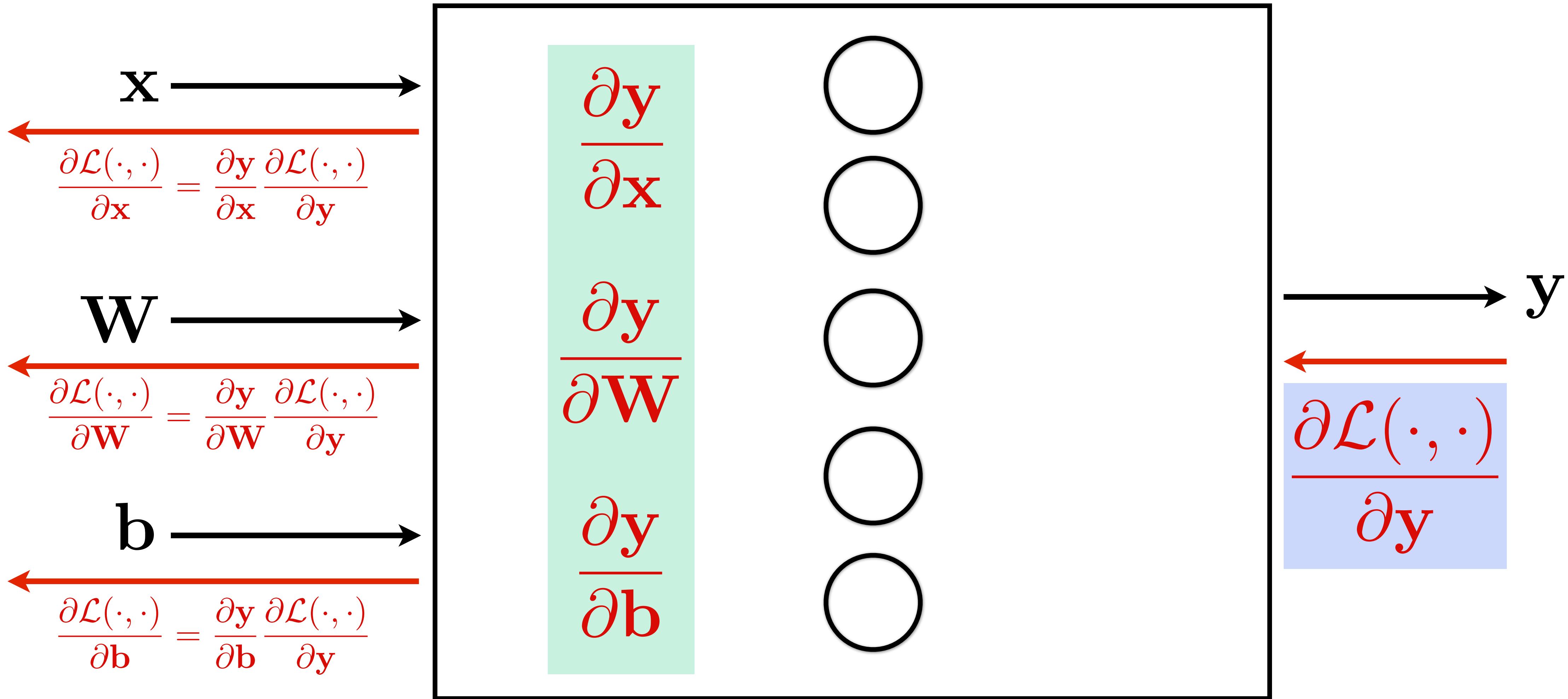


Backpropagation Practical Issues

“local” Jacobians
(matrix of partial derivatives, e.g. size $|x| \times |y|$)

$$y = f(\mathbf{W}, \mathbf{b}, \mathbf{x}) = \text{sigmoid}(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

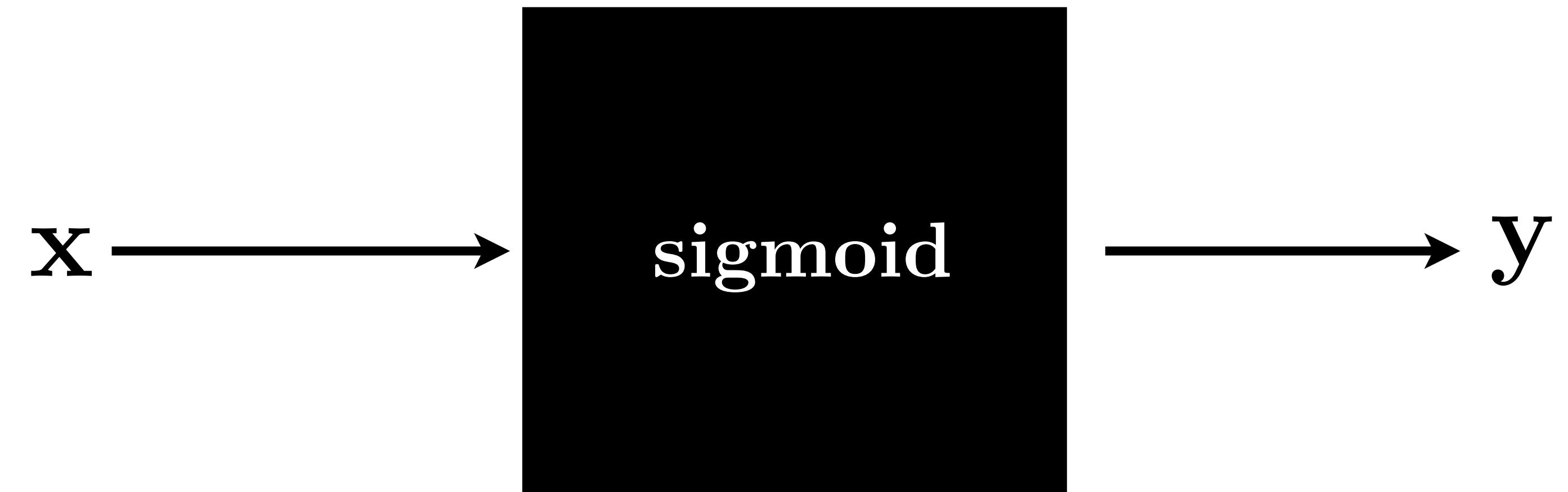
“backprop” Gradient



Jacobian of Sigmoid layer

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^{2048}$$

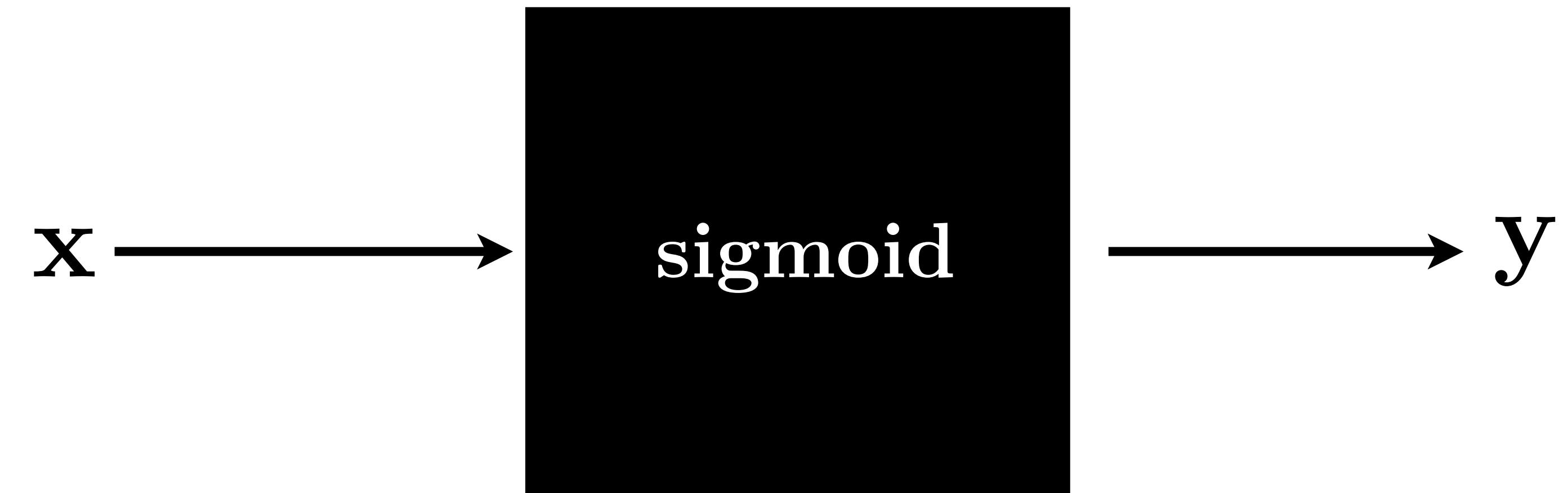
Element-wise sigmoid layer:



Jacobian of Sigmoid layer

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^{2048}$$

Element-wise sigmoid layer:

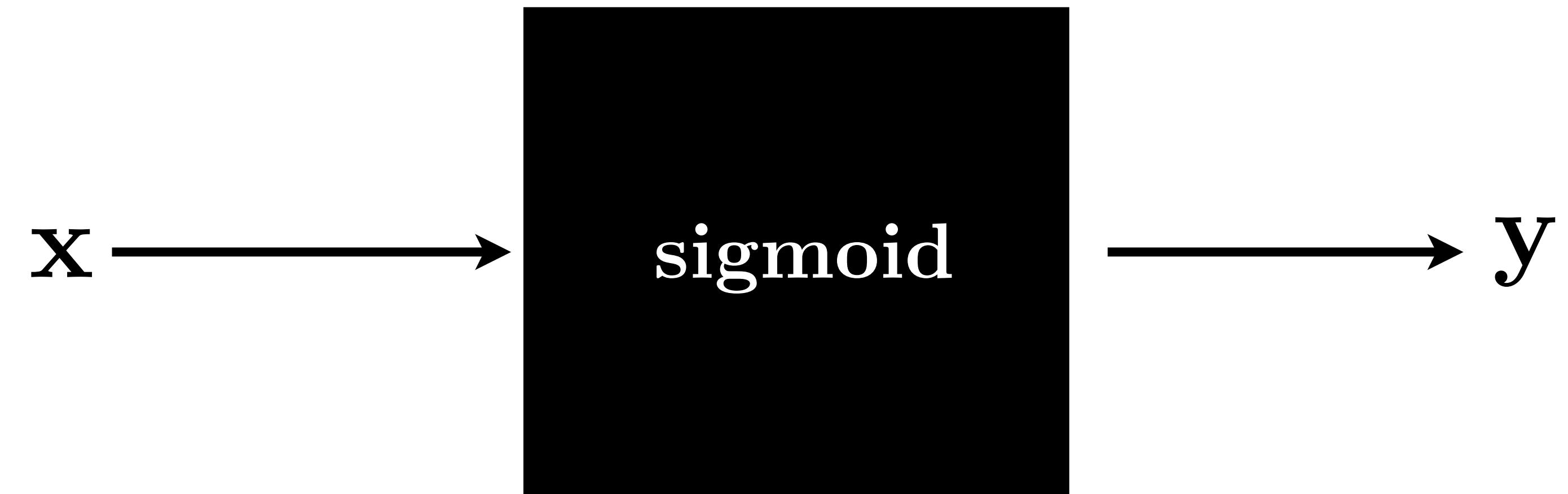


What is the dimension of **Jacobian**?

Jacobian of Sigmoid layer

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^{2048}$$

Element-wise sigmoid layer:



What is the dimension of **Jacobian**?

What does it look like?

Jacobian of Sigmoid layer

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^{2048}$$

Element-wise sigmoid layer:



What is the dimension of **Jacobian**?

What does it look like?

If we are working with a mini batch of 100 inputs-output pairs, technically Jacobian is a matrix $204,800 \times 204,800$

Backpropagation: Common questions

Question: Does BackProp only work for certain layers?

Answer: No, for any differentiable functions

Question: What is computational cost of BackProp?

Answer: On average about twice the forward pass

Question: Is BackProp a dual of forward propagation?

Answer: Yes

Backpropagation: Common questions

Question: Does BackProp only work for certain layers?

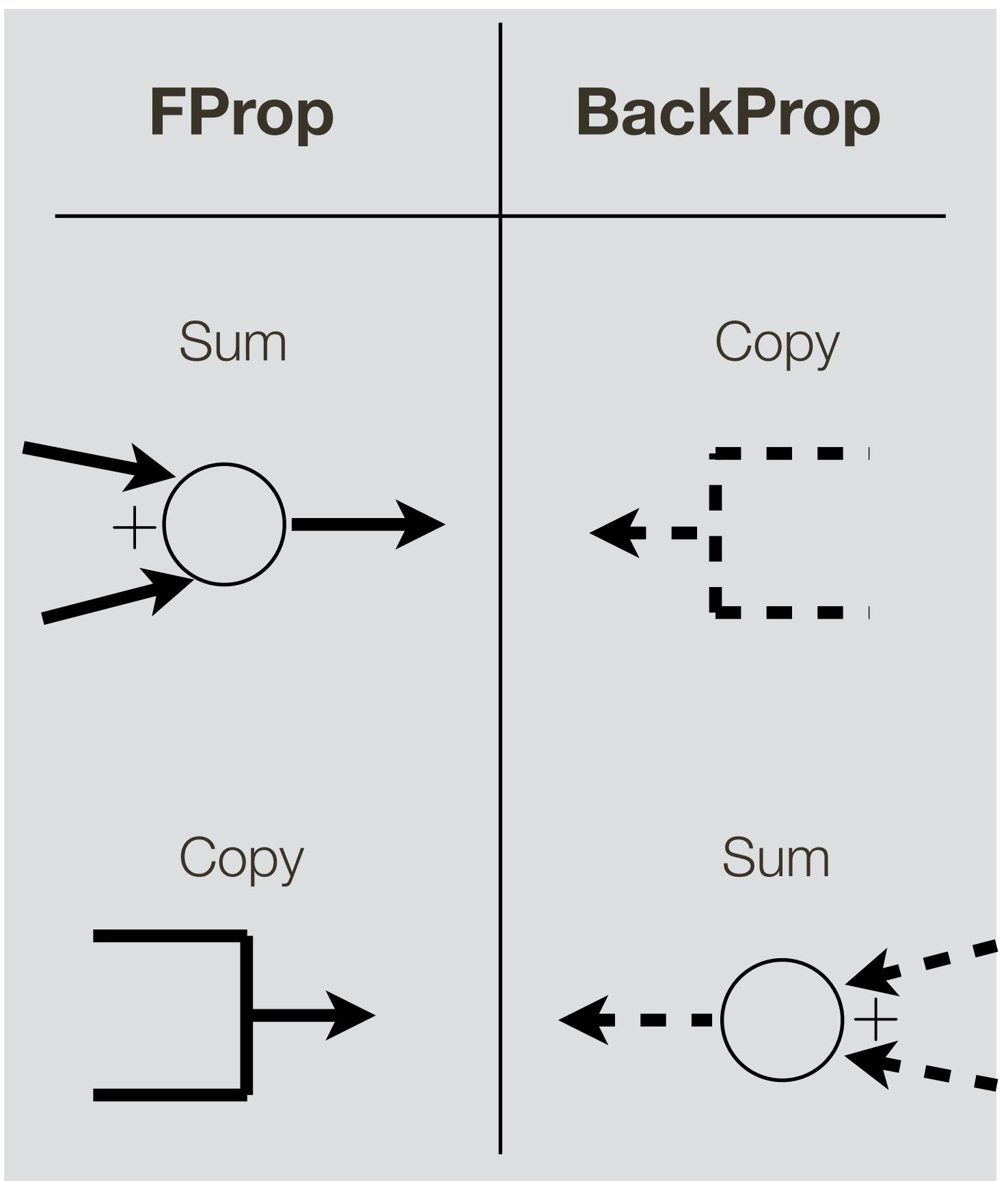
Answer: No, for any differentiable functions

Question: What is computational cost of BackProp?

Answer: On average about twice the forward pass

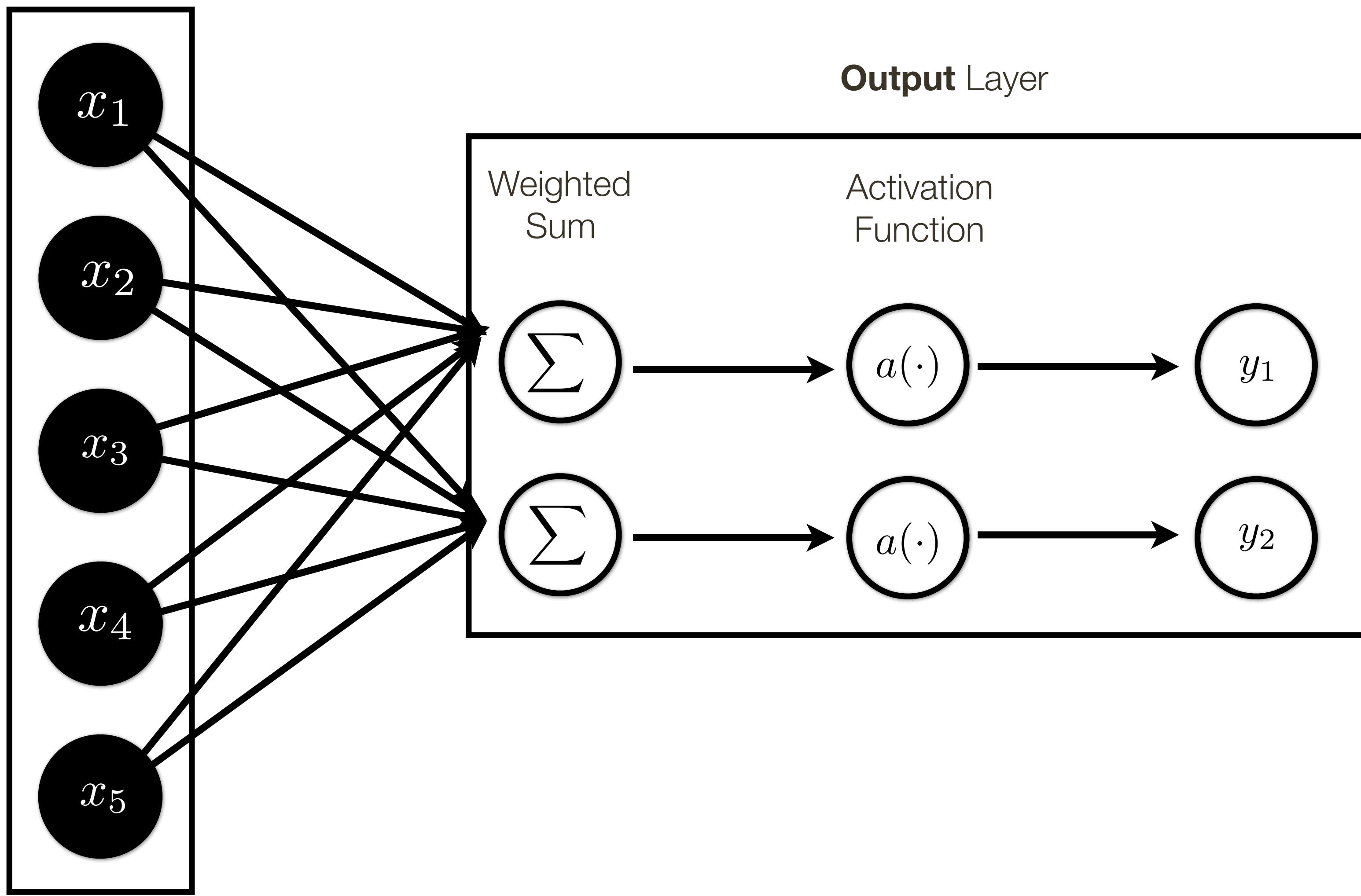
Question: Is BackProp a dual of forward propagation?

Answer: Yes

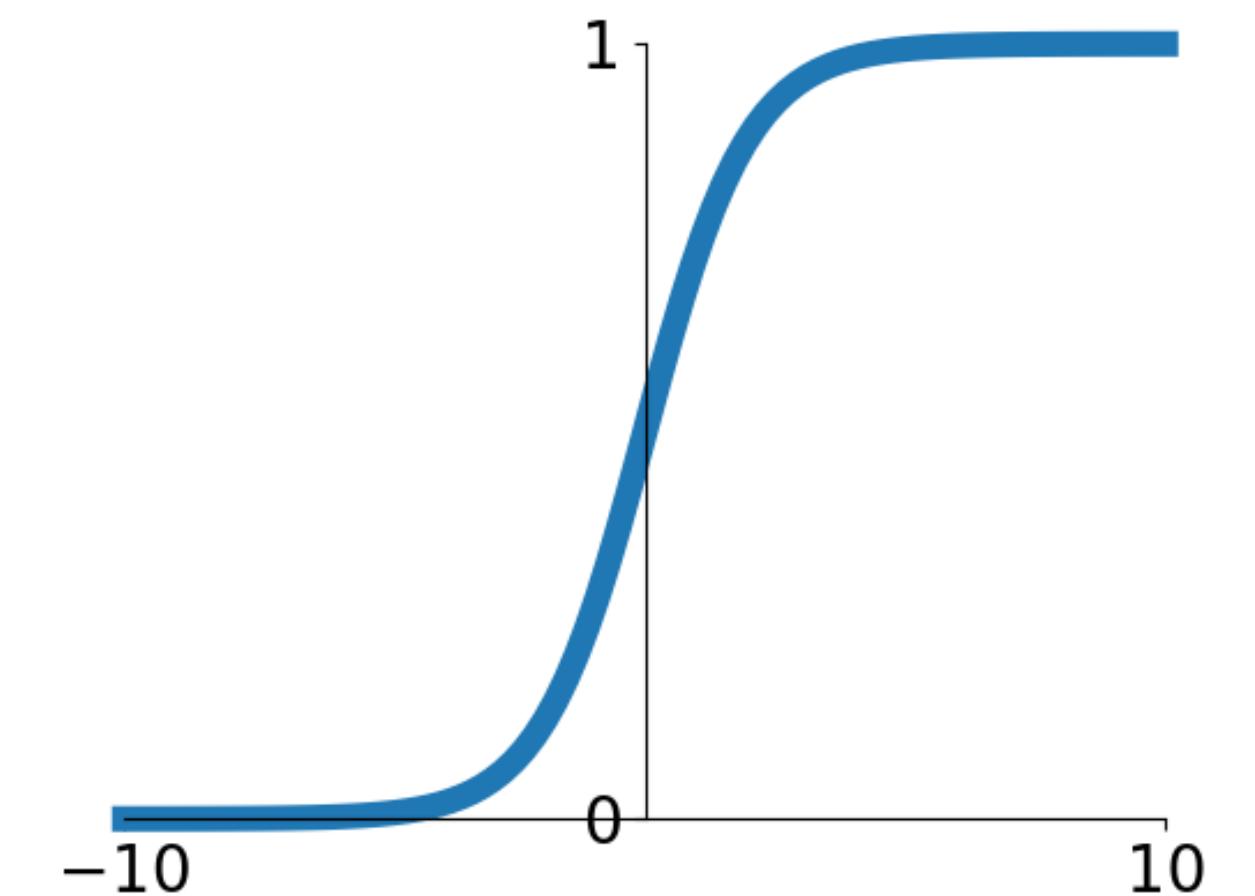


Activation Function: Sigmoid

Input Layer

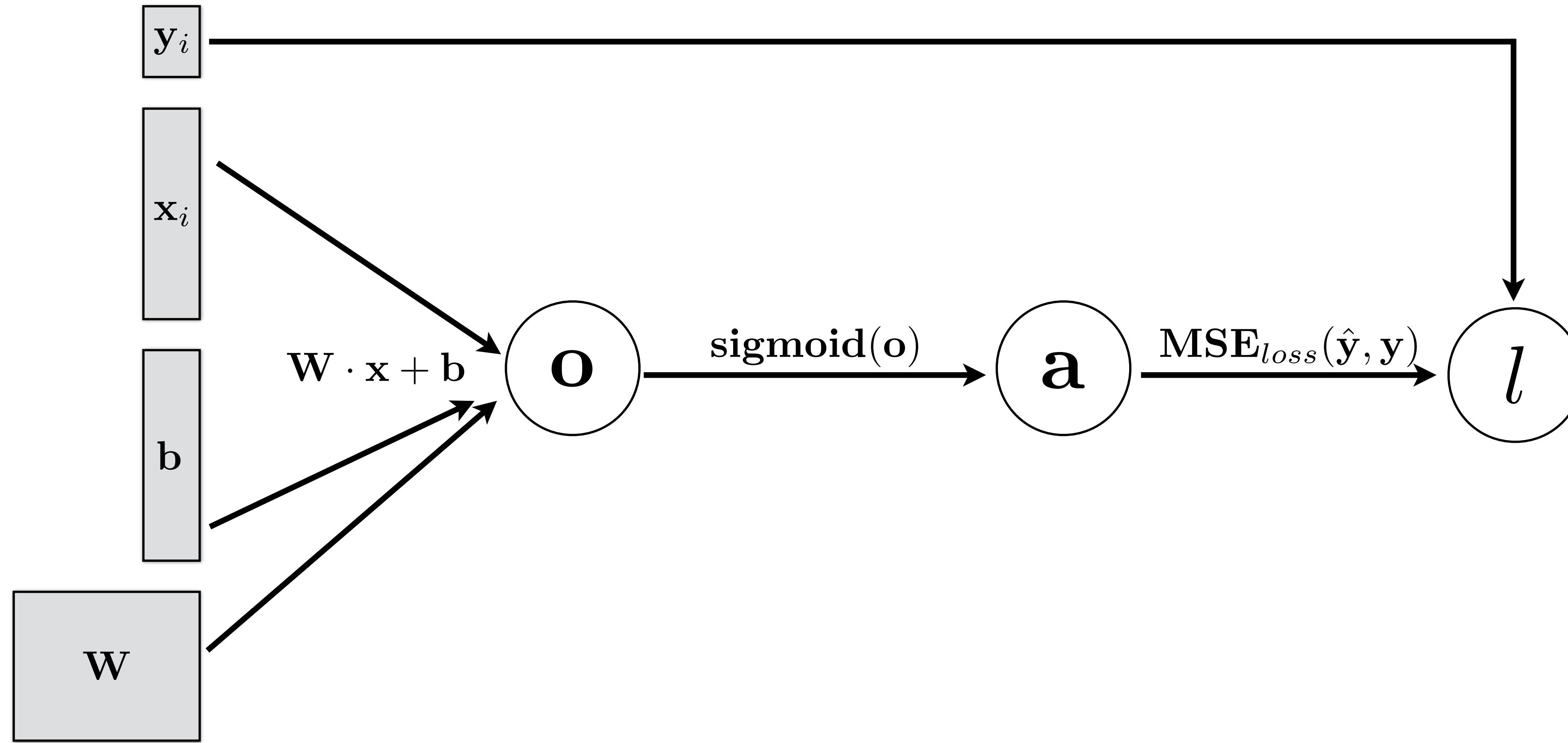


$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



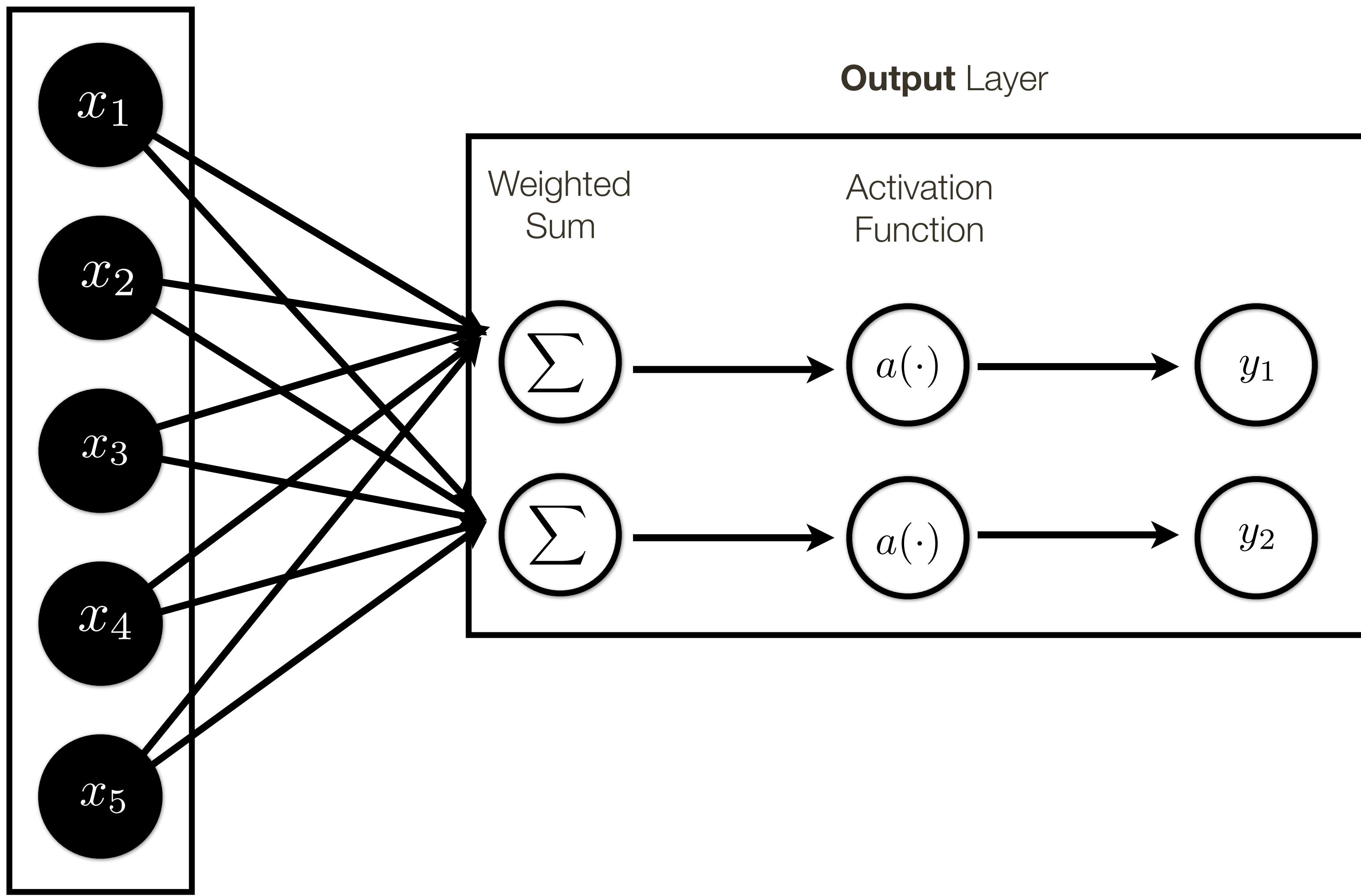
Sigmoid Activation

Computational Graph: 1-layer network

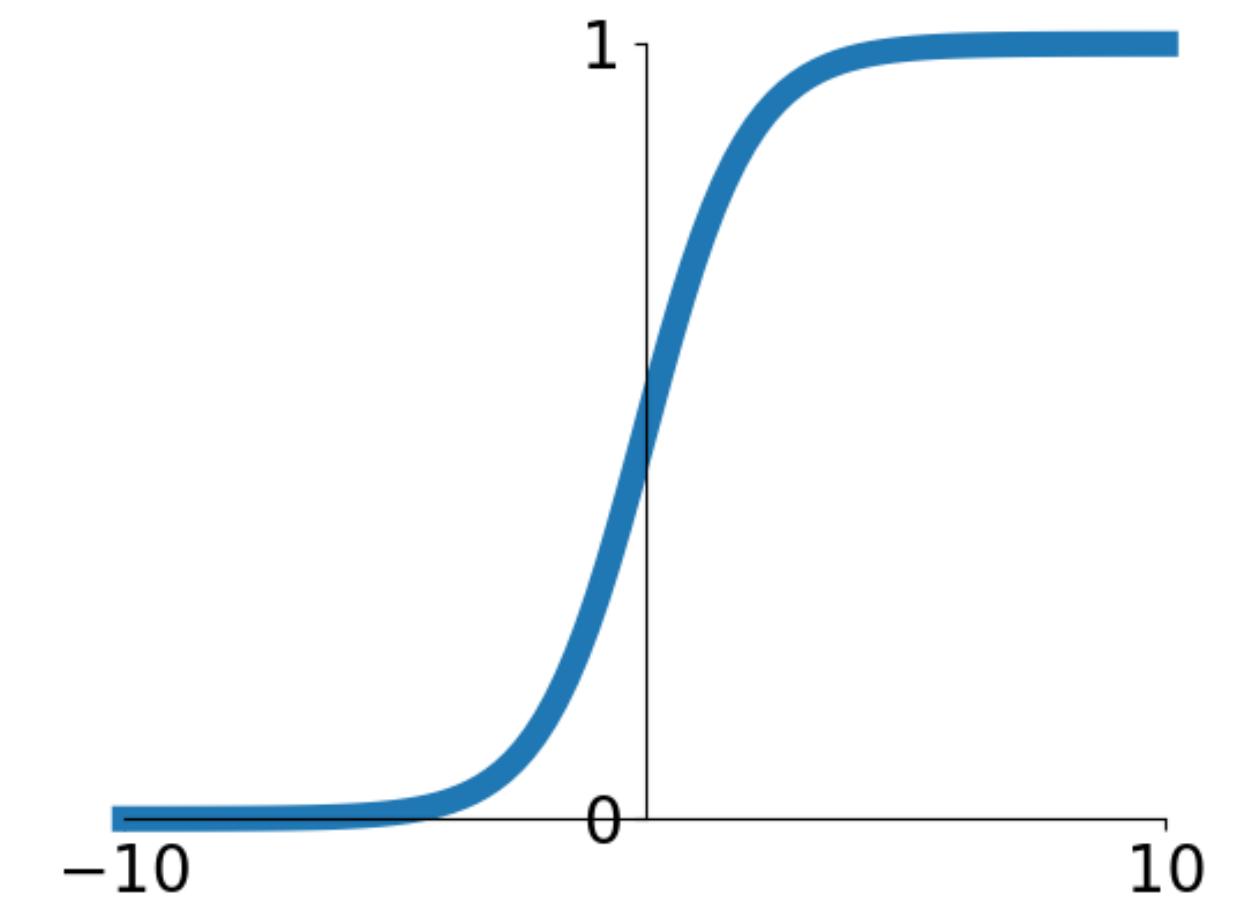


Activation Function: Sigmoid

Input Layer



$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



Sigmoid Activation

Activation Function: Sigmoid

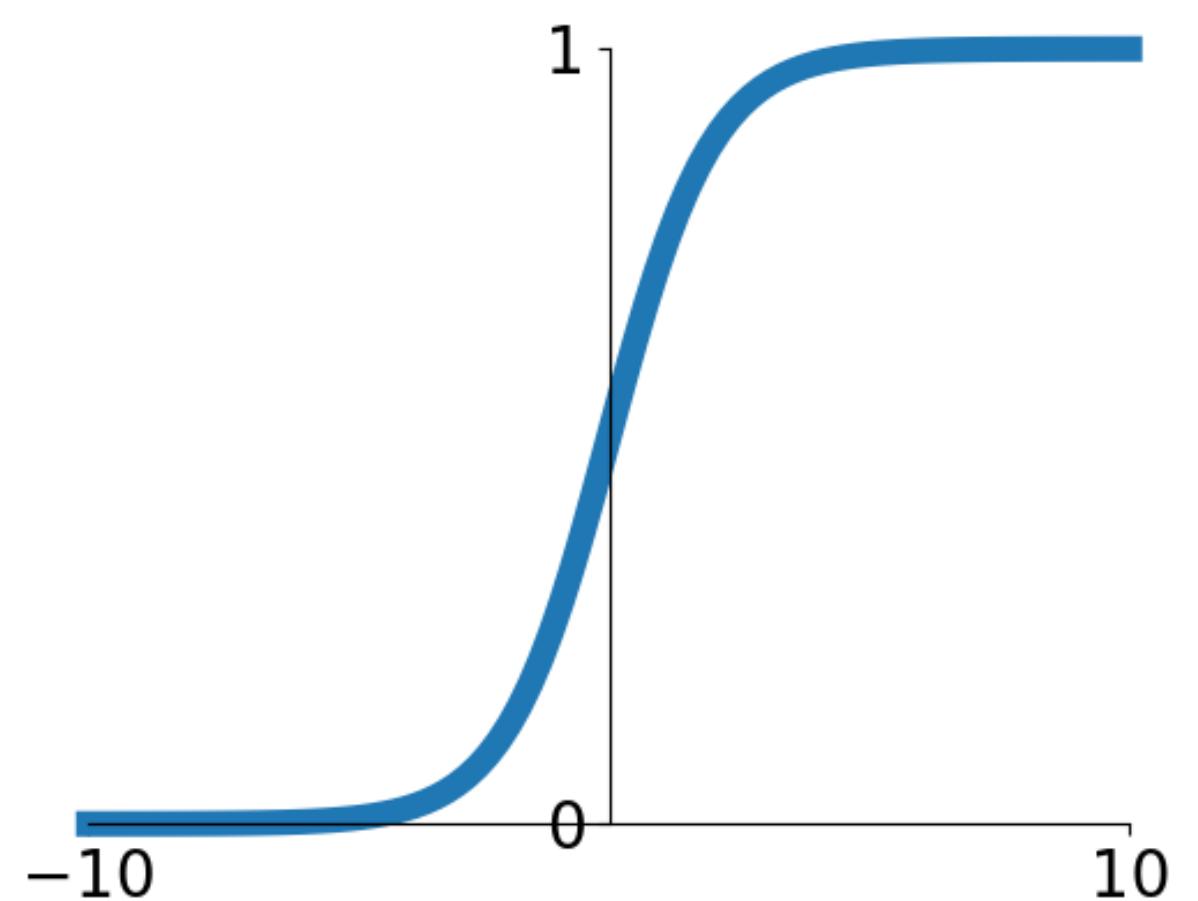
$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

Pros:

- Squishes everything in the range $[0, 1]$
- Can be interpreted as “probability”
- Has well defined gradient everywhere

Cons:

- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute

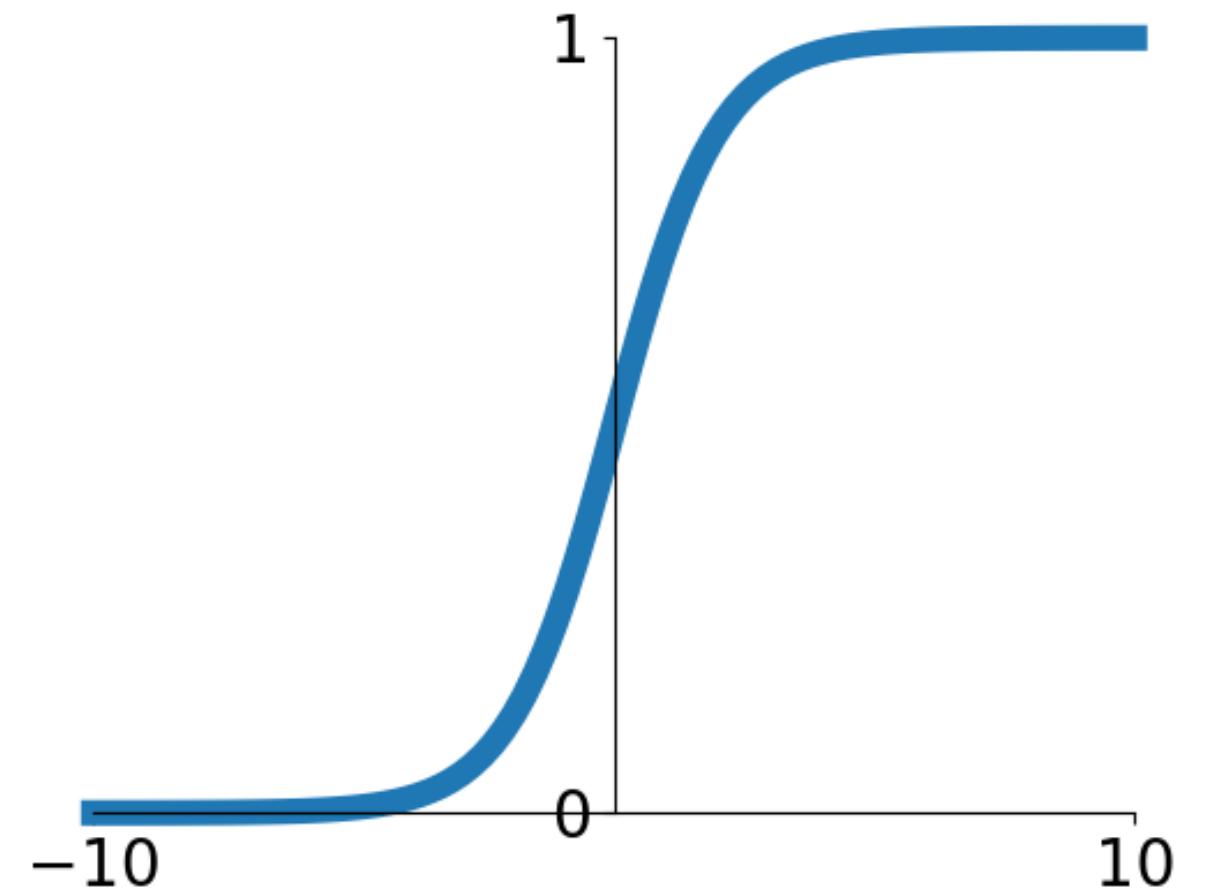


Sigmoid Activation

Activation Function: Sigmoid

Sigmoid
Gate

$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

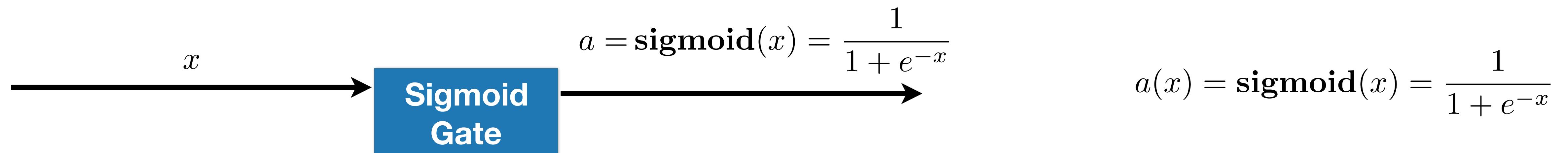


Cons:

- Saturated neurons “**kill**” the gradients
- Non-zero centered
- Could be expensive to compute

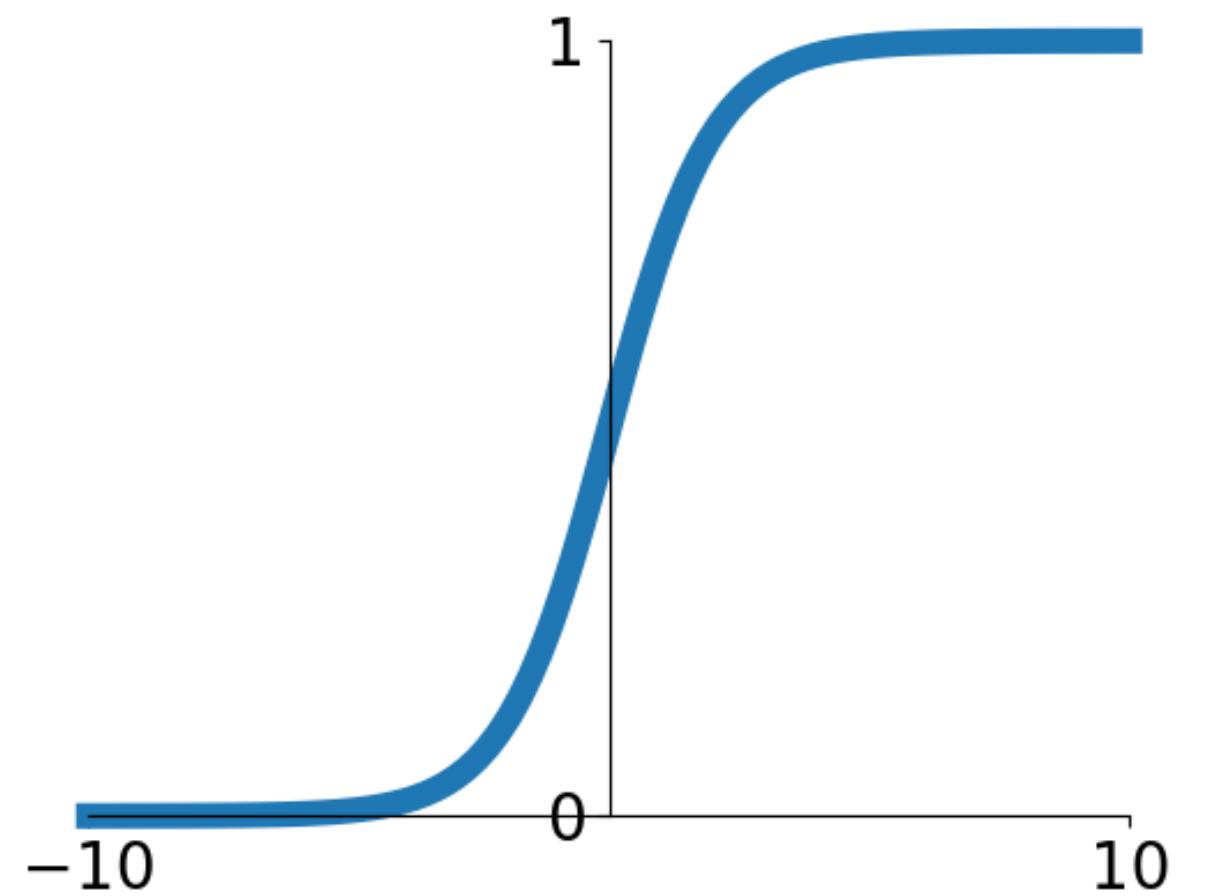
Sigmoid Activation

Activation Function: Sigmoid



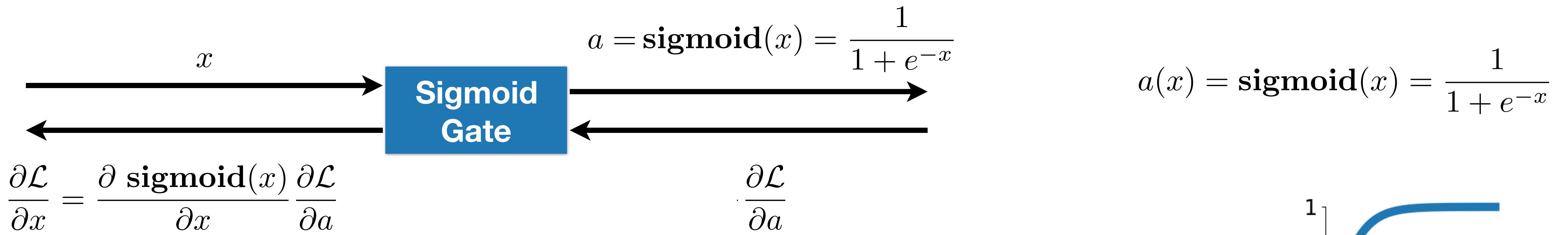
Cons:

- Saturated neurons “**kill**” the gradients
- Non-zero centered
- Could be expensive to compute



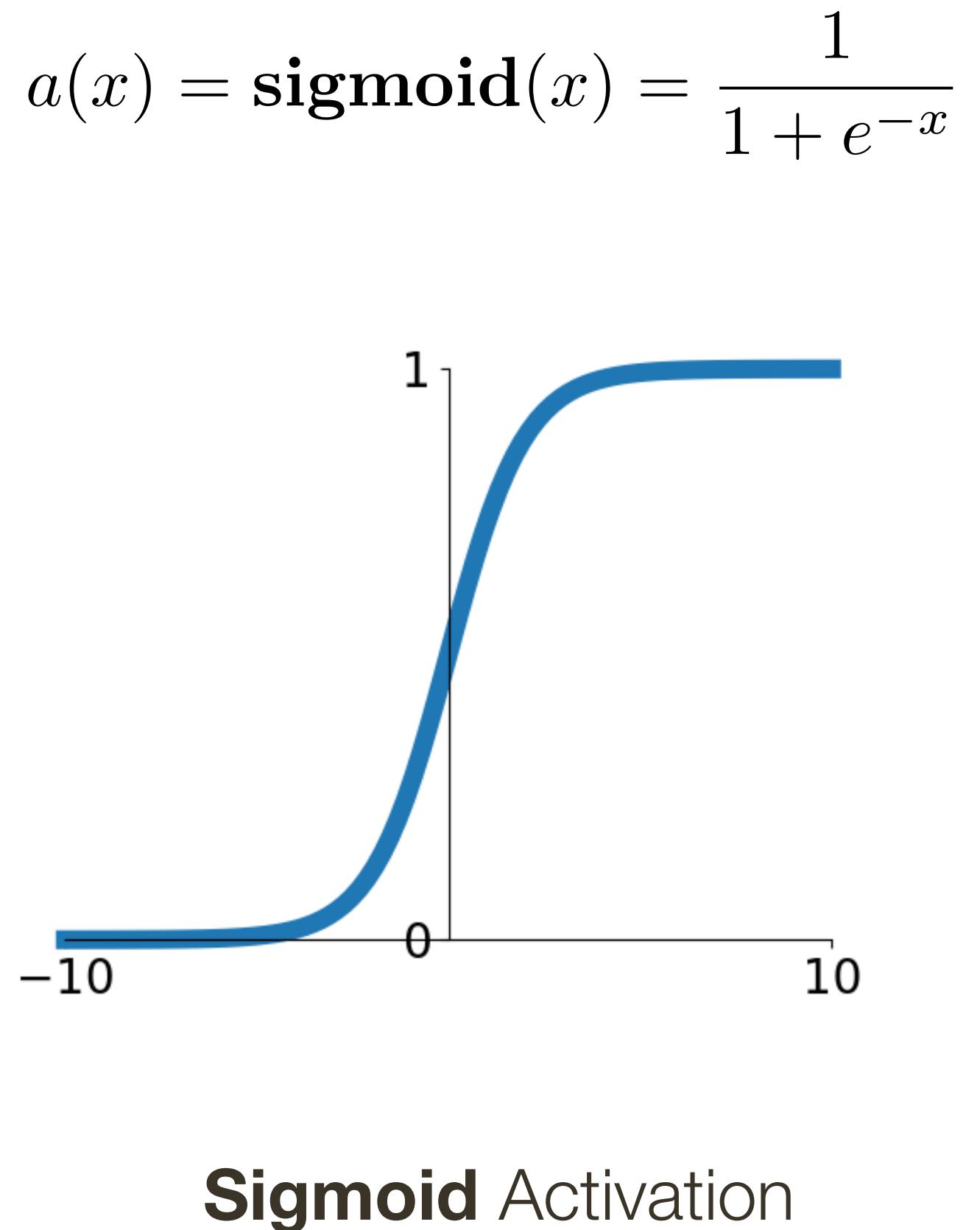
Sigmoid Activation

Activation Function: Sigmoid

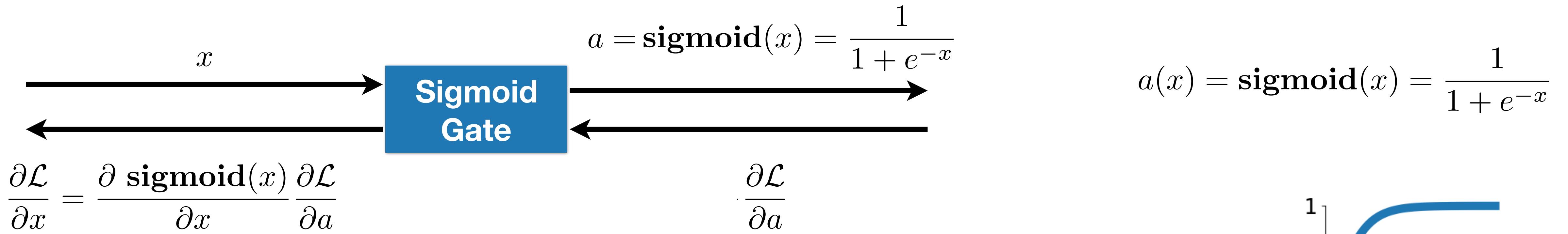


Cons:

- Saturated neurons “**kill**” the gradients
- Non-zero centered
- Could be expensive to compute

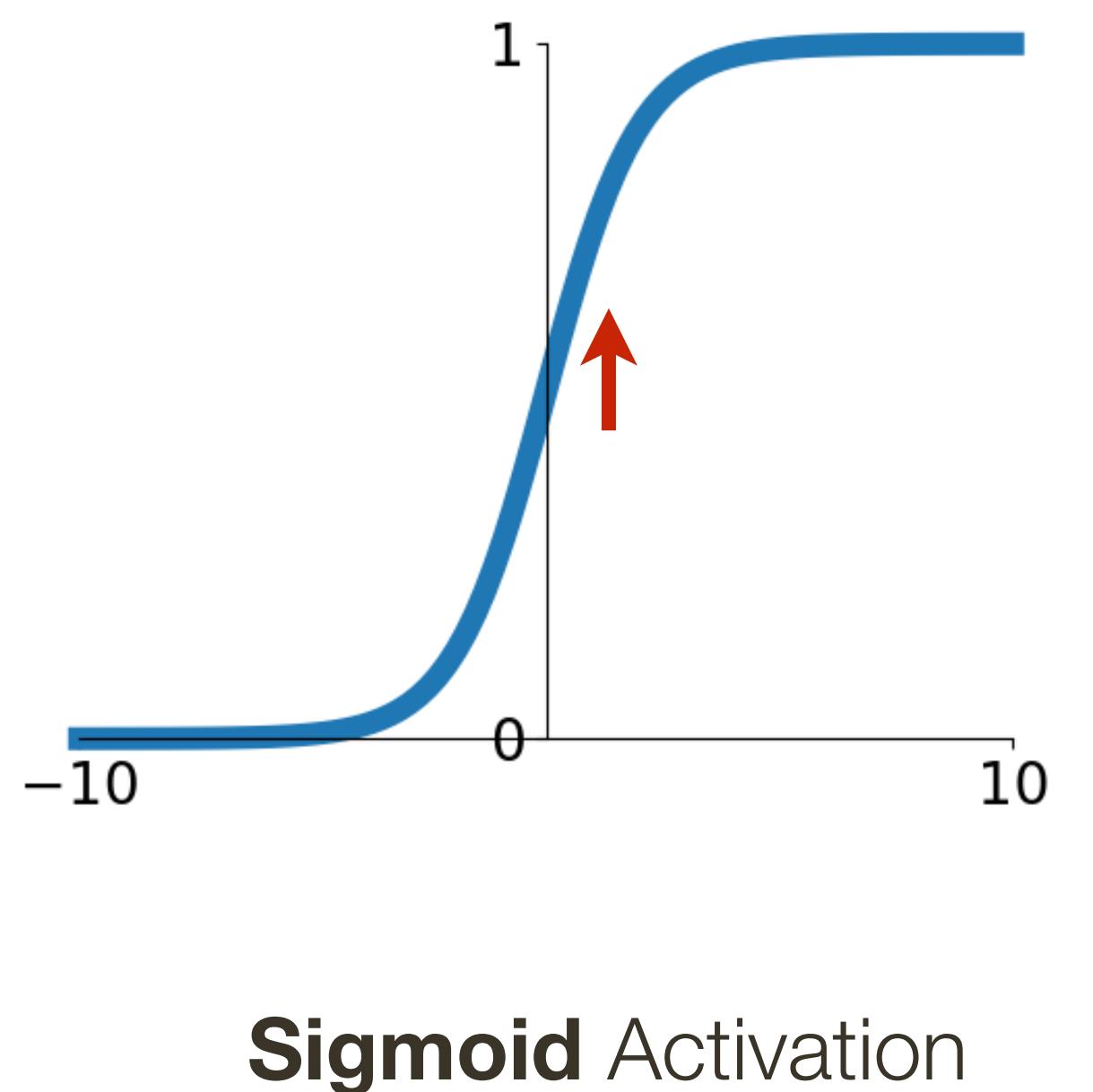


Activation Function: Sigmoid



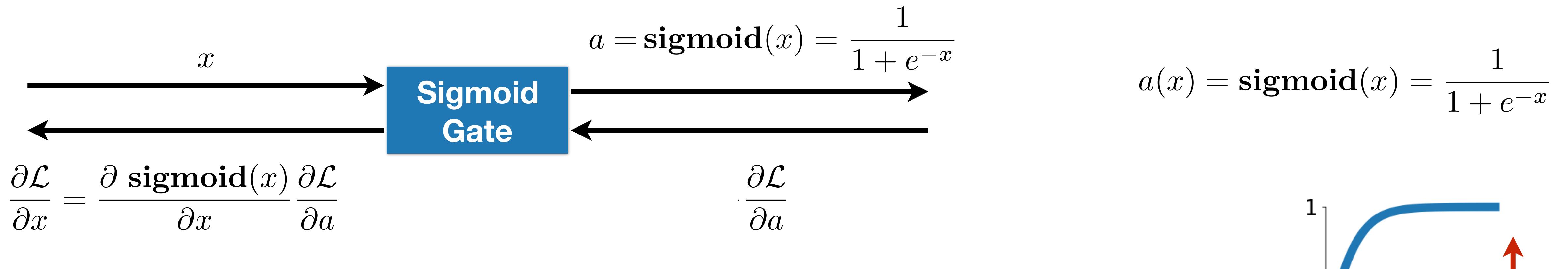
Cons:

- Saturated neurons “**kill**” the gradients
- Non-zero centered
- Could be expensive to compute



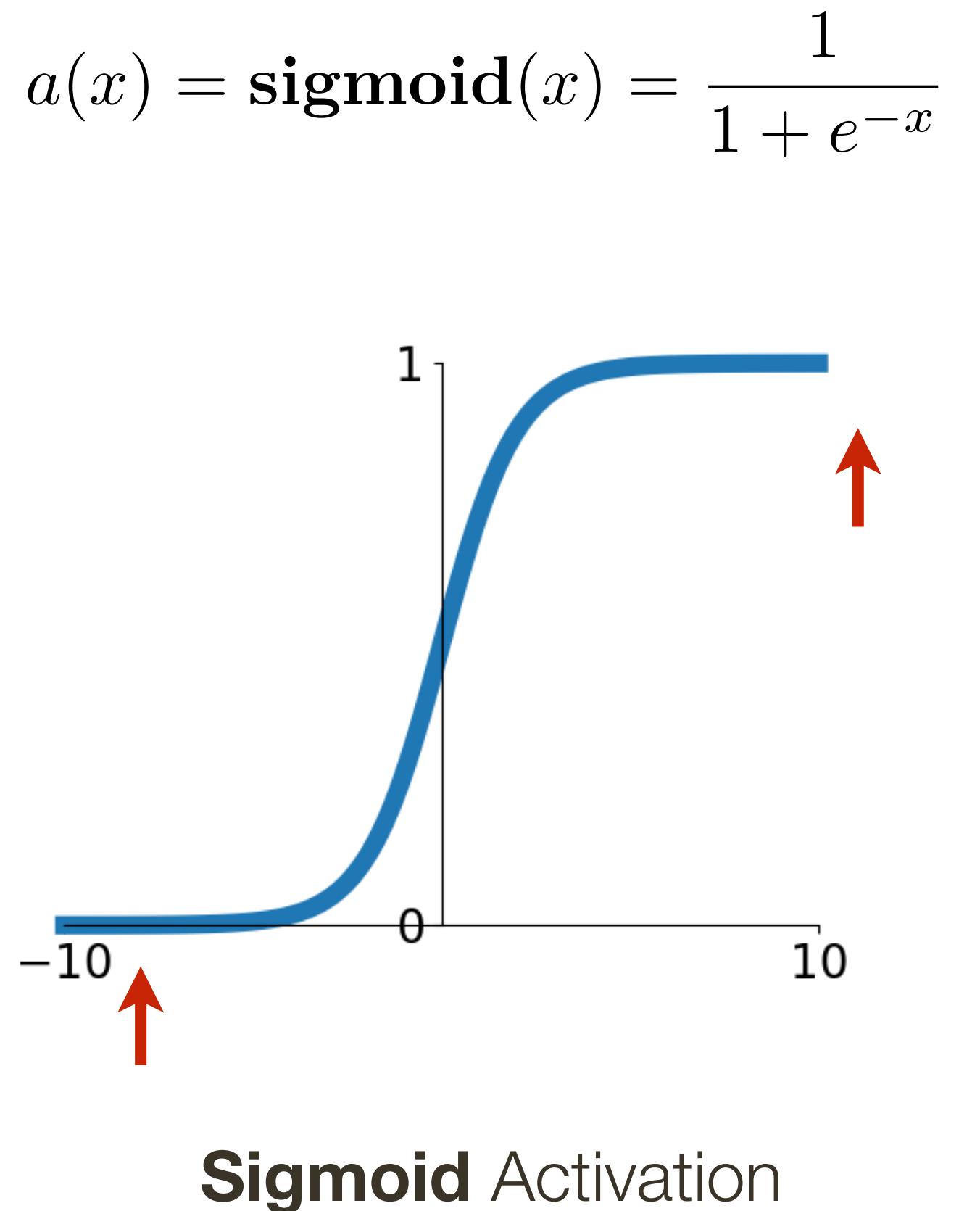
Sigmoid Activation

Activation Function: Sigmoid



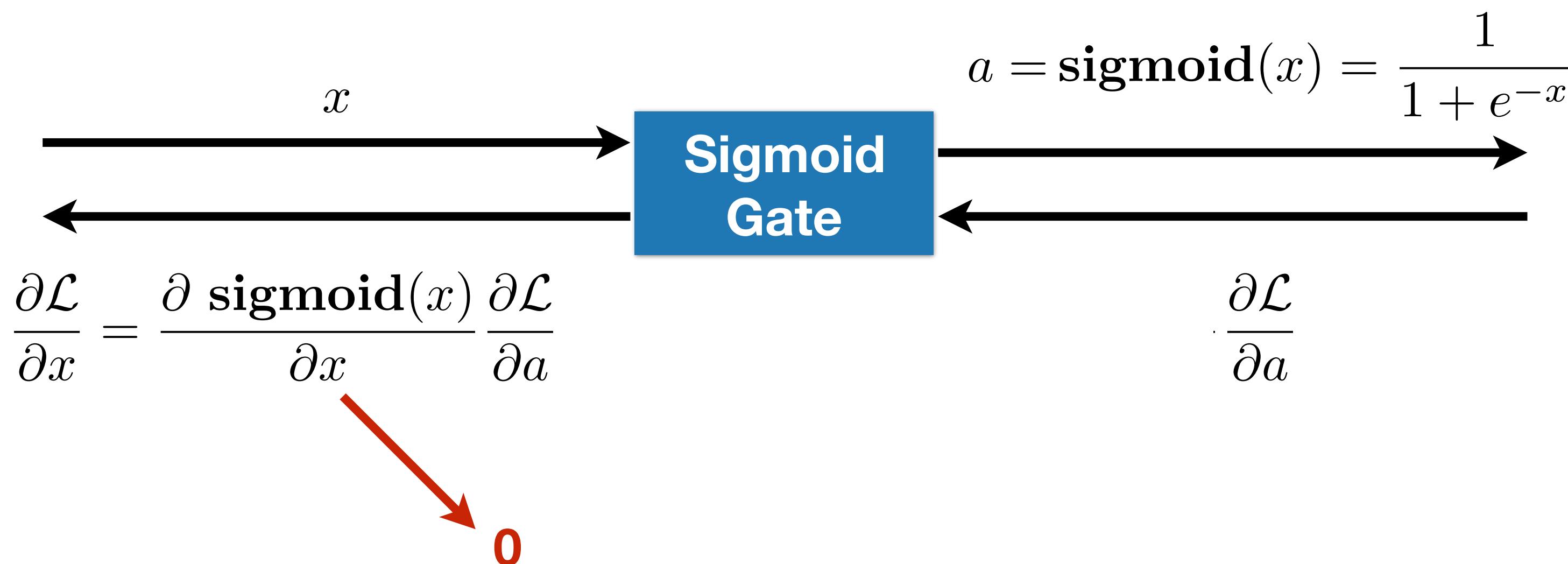
Cons:

- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute



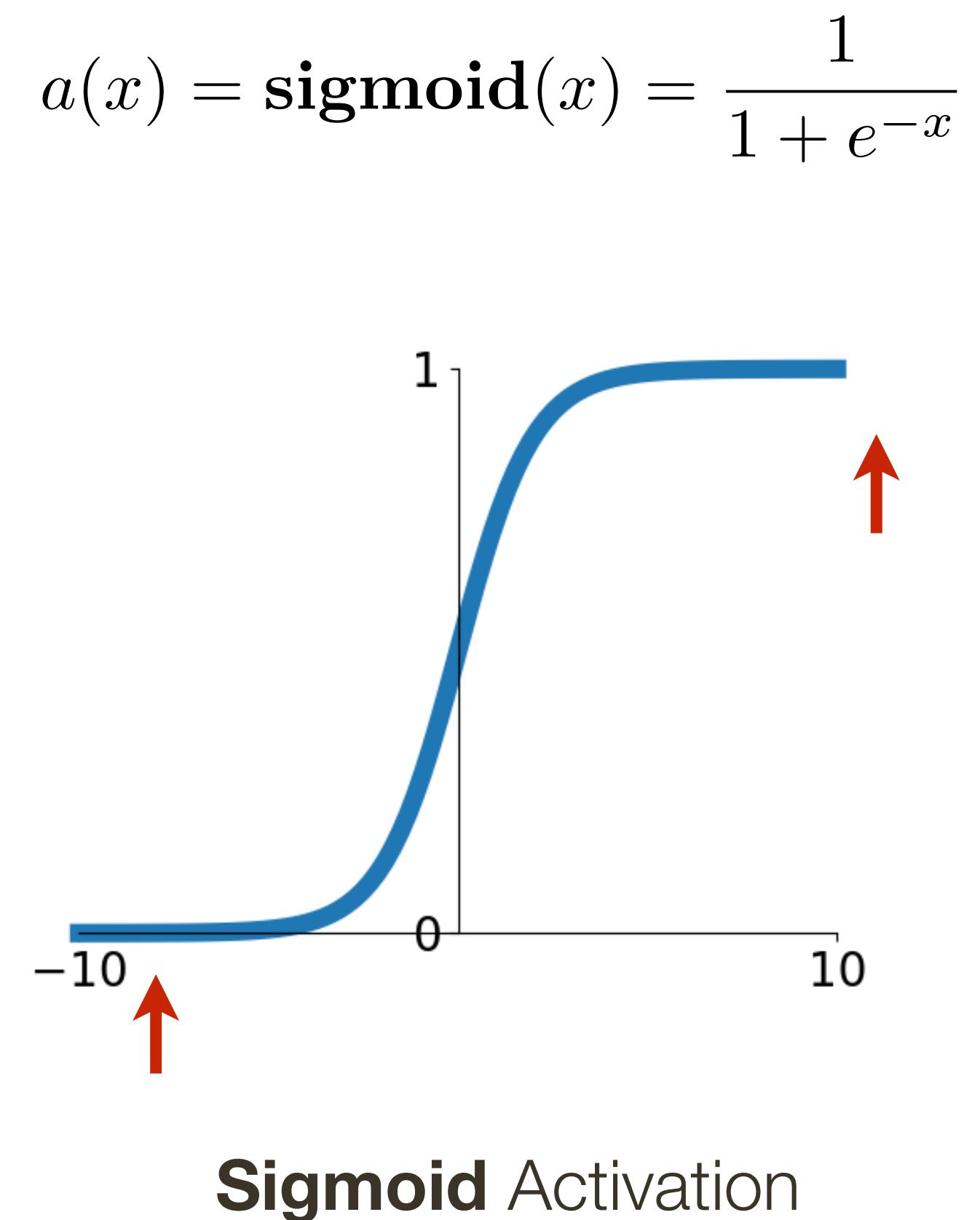
Sigmoid Activation

Activation Function: Sigmoid



Cons:

- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute



Sigmoid Activation

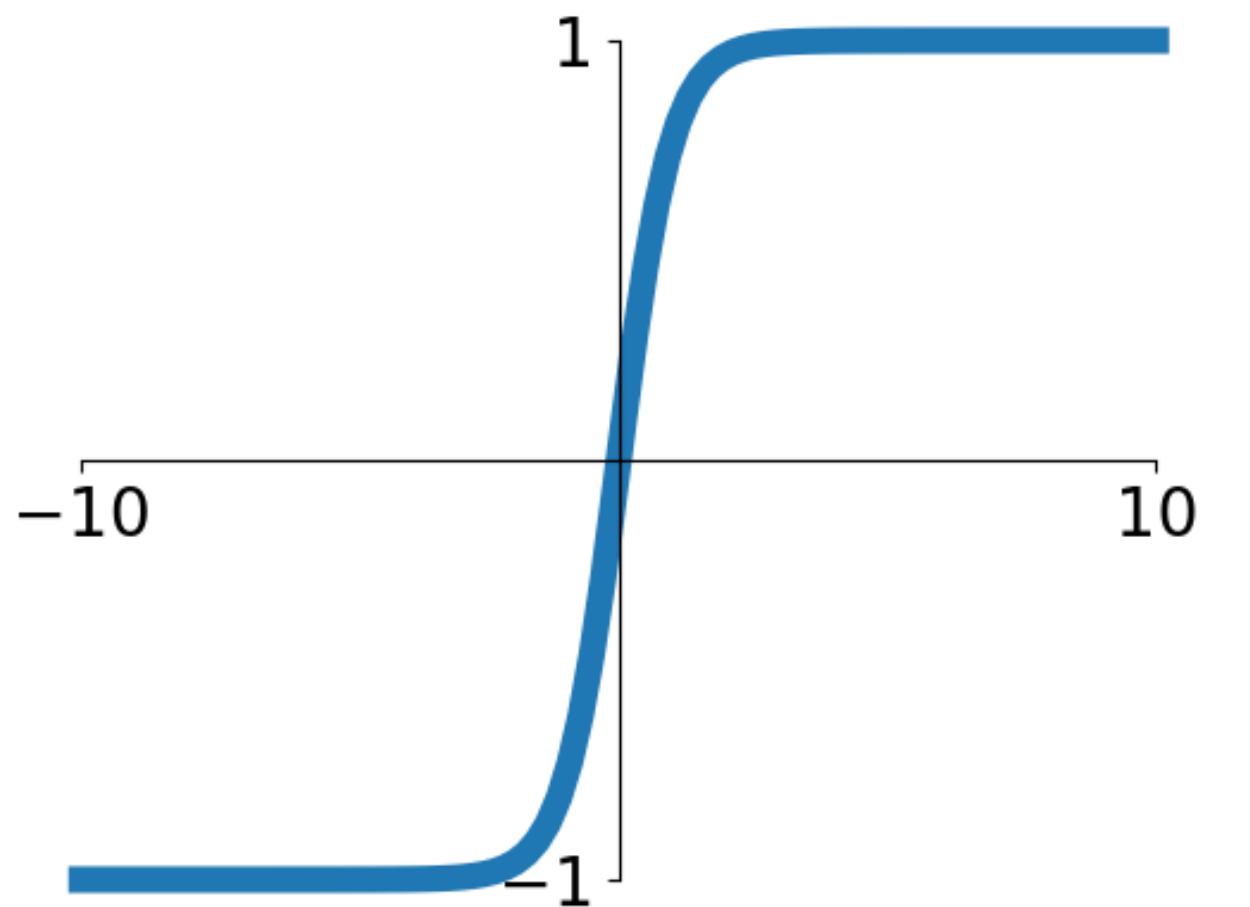
Activation Function: Tanh

$$a(x) = \tanh(x) = 2 \cdot \text{sigmoid}(2x) - 1$$

$$a(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Pros:

- Squishes everything in the range $[-1, 1]$
- Centered around zero
- Has well defined gradient everywhere



Cons:

- Saturated neurons “kill” the gradients

Tanh Activation

Activation Function: Rectified Linear Unit (ReLU)

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

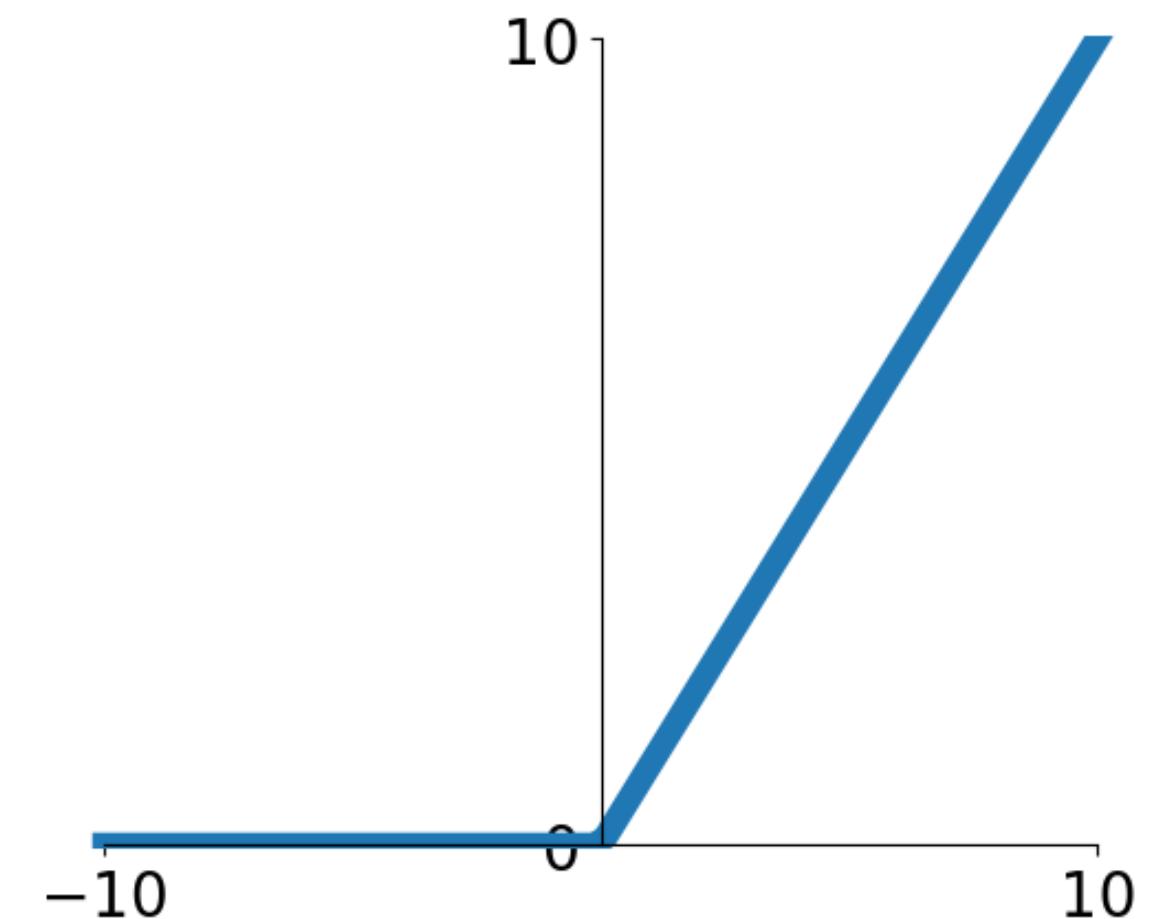
$$a'(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Pros:

- Does not saturate (for $x > 0$)
- Computationally very efficient
- Converges faster in practice (e.g. 6 times faster)

Cons:

- Not zero centered



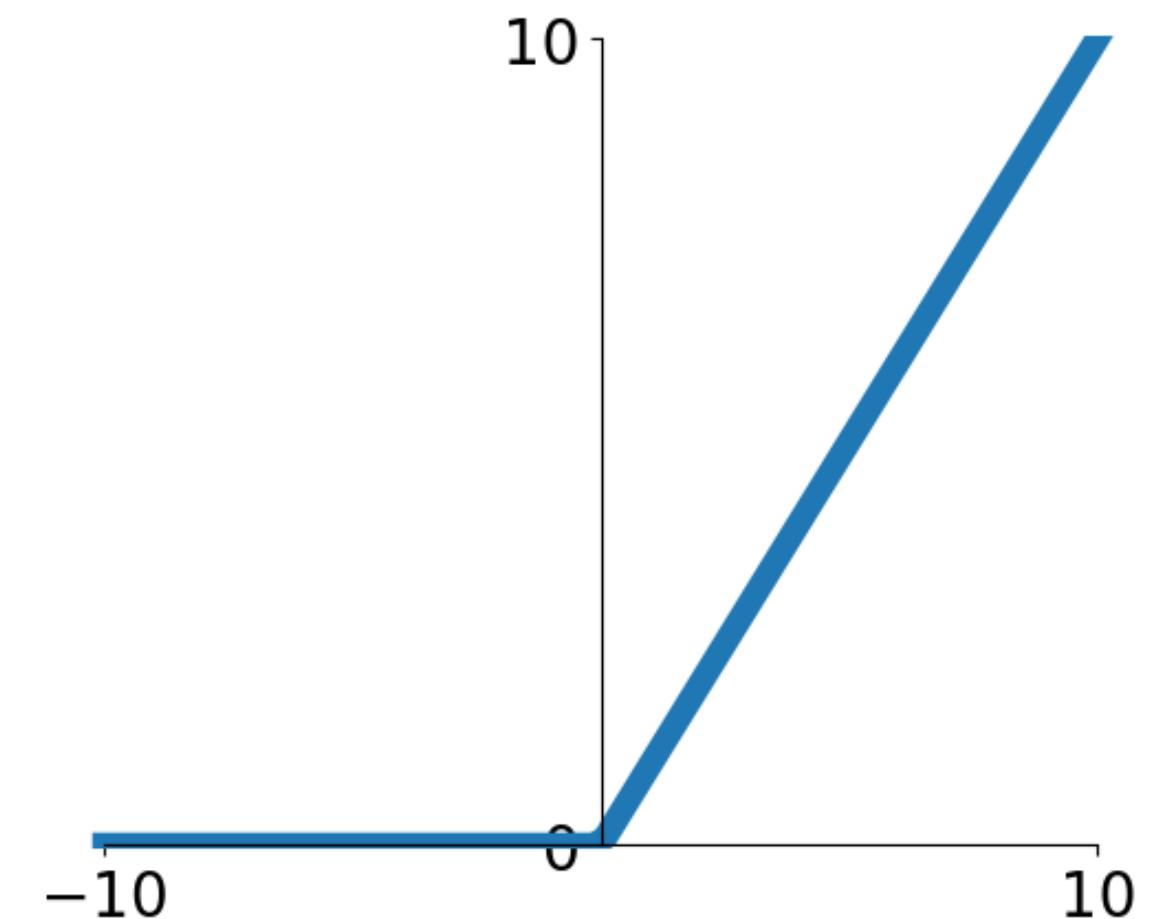
ReLU Activation

Activation Function: Rectified Linear Unit (ReLU)

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

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

Question: What do ReLU layers accomplish?



ReLU Activation

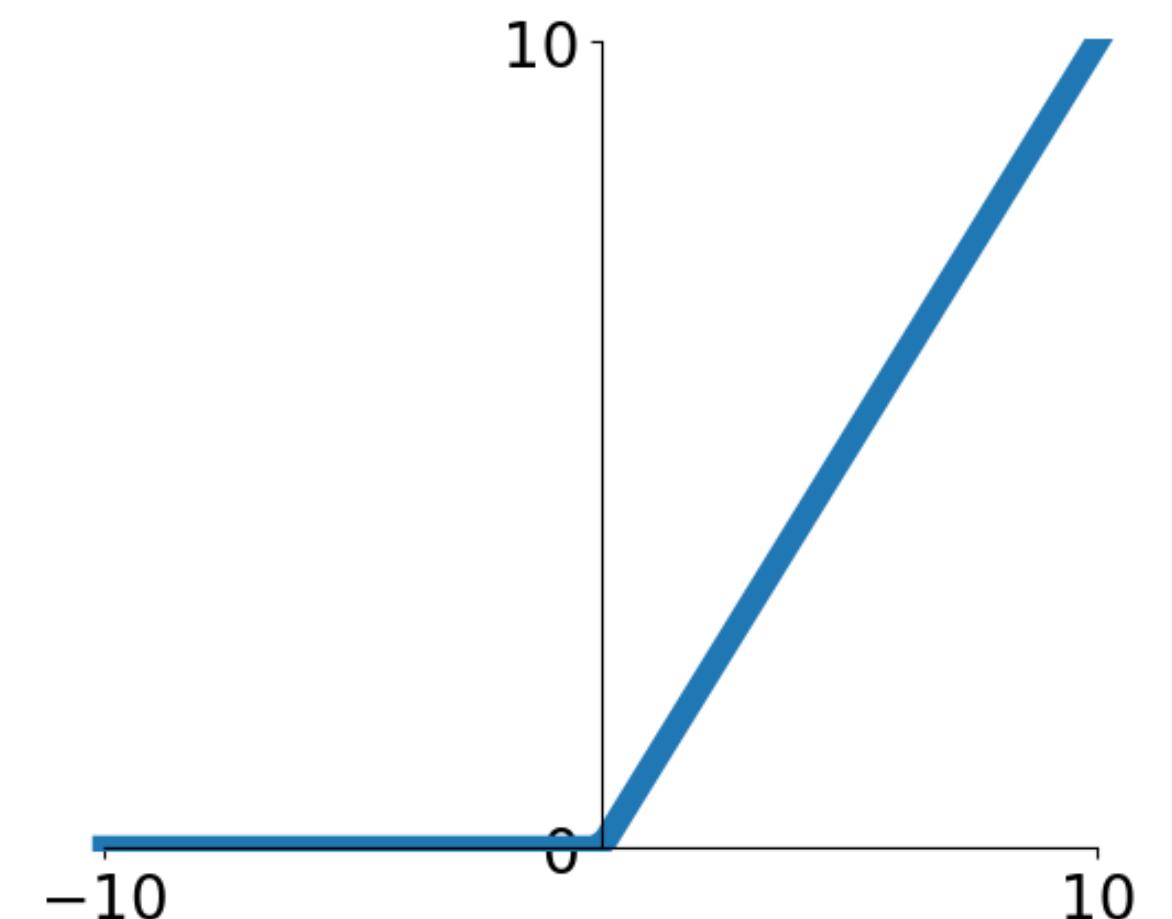
Activation Function: Rectified Linear Unit (ReLU)

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

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

Question: What do ReLU layers accomplish?

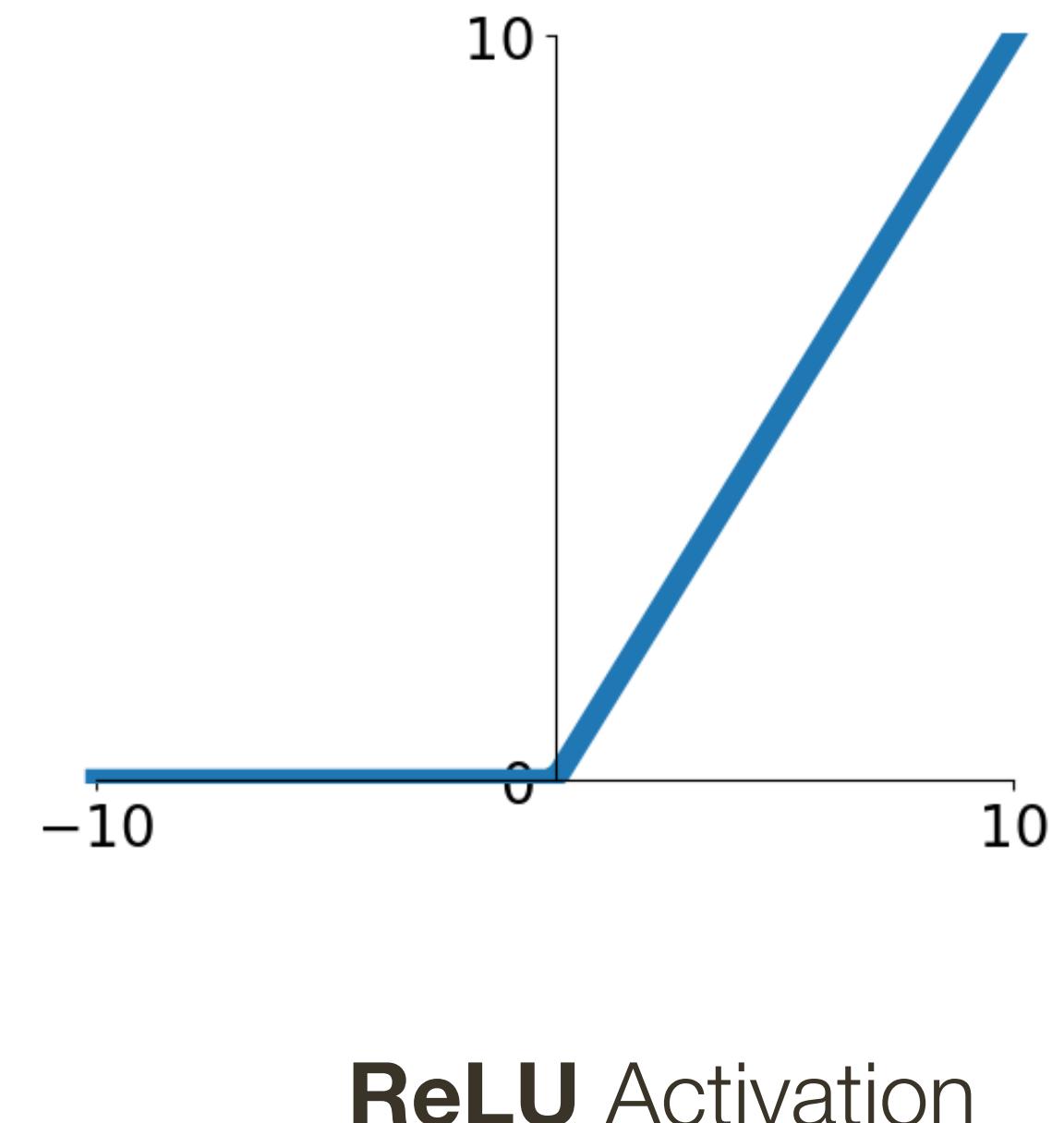
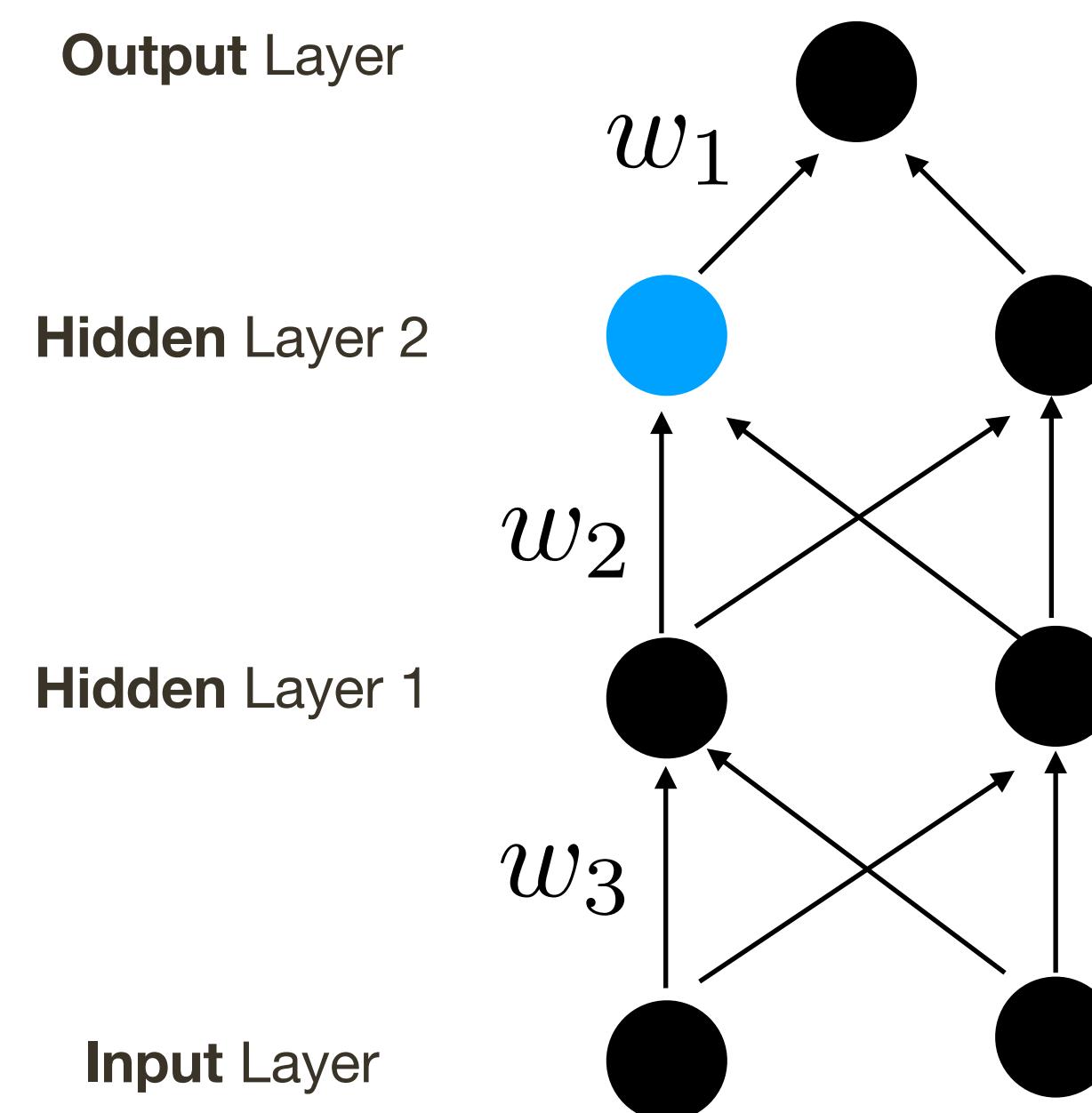
Answer: Locally linear tiling, function is locally linear



Activation Function: Rectified Linear Unit (ReLU)

ReLU sparcifies activations and derivatives

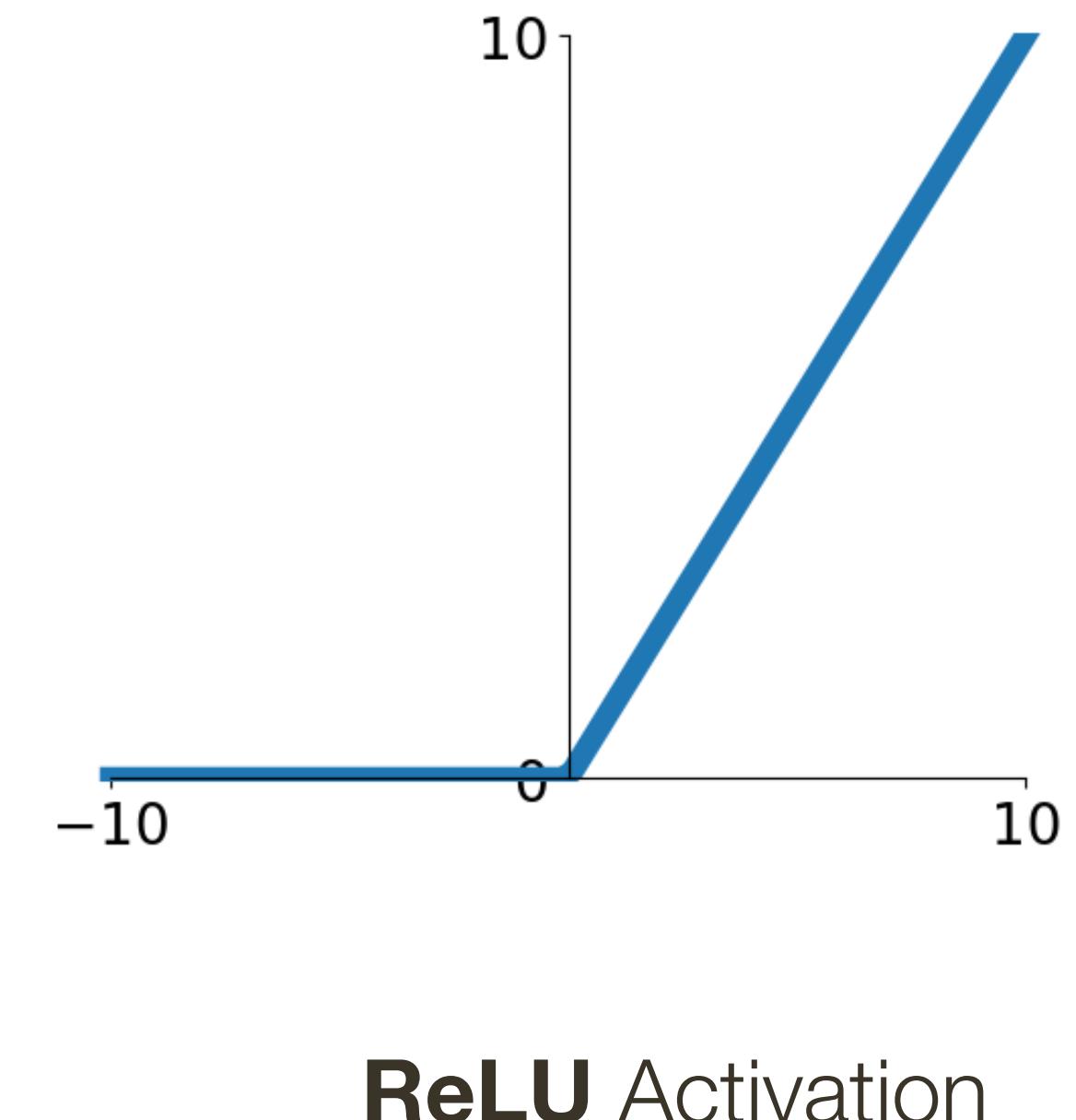
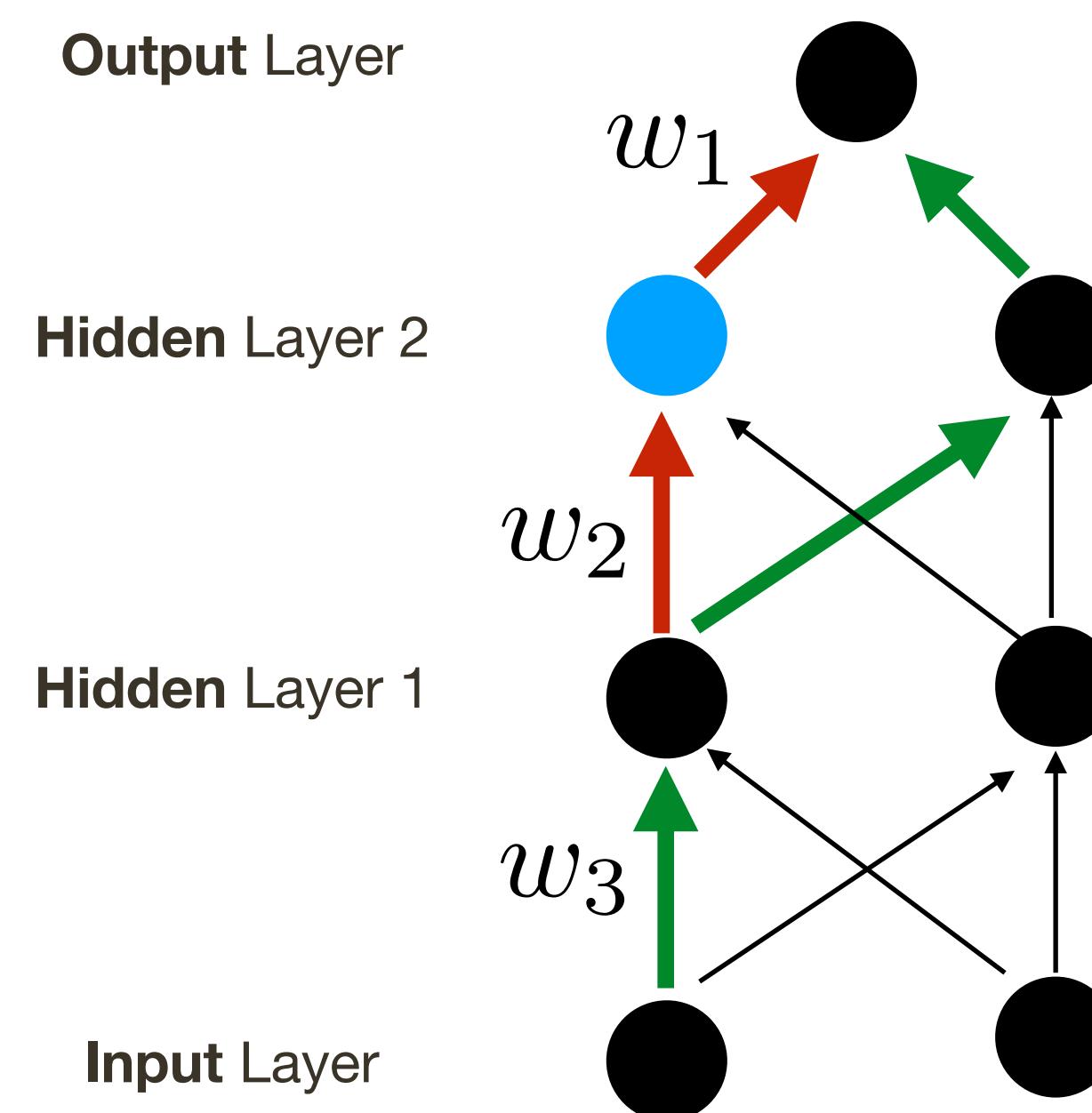
$$a(x) = \max(0, x)$$
$$a'(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



Activation Function: Rectified Linear Unit (ReLU)

ReLU sparcifies activations and derivatives

$$a(x) = \max(0, x)$$
$$a'(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



Recall:

Conditions needed to prove NN is a universal approximator: Activation function needs to be well defined

$$\lim_{x \rightarrow \infty} a(x) = A$$

$$\lim_{x \rightarrow -\infty} a(x) = B$$

$$A \neq B$$

Recall:

Conditions needed to prove NN is a universal approximator: Activation function needs to be well defined

$$\lim_{x \rightarrow \infty} a(x) = A$$

$$\lim_{x \rightarrow -\infty} a(x) = B$$

$$A \neq B$$

Fun **Exercise:** Try to prove that network with ReLU is still a universal approximator (not too difficult if you think about it visually)

Activation Function: Leaky / Parametrized ReLU

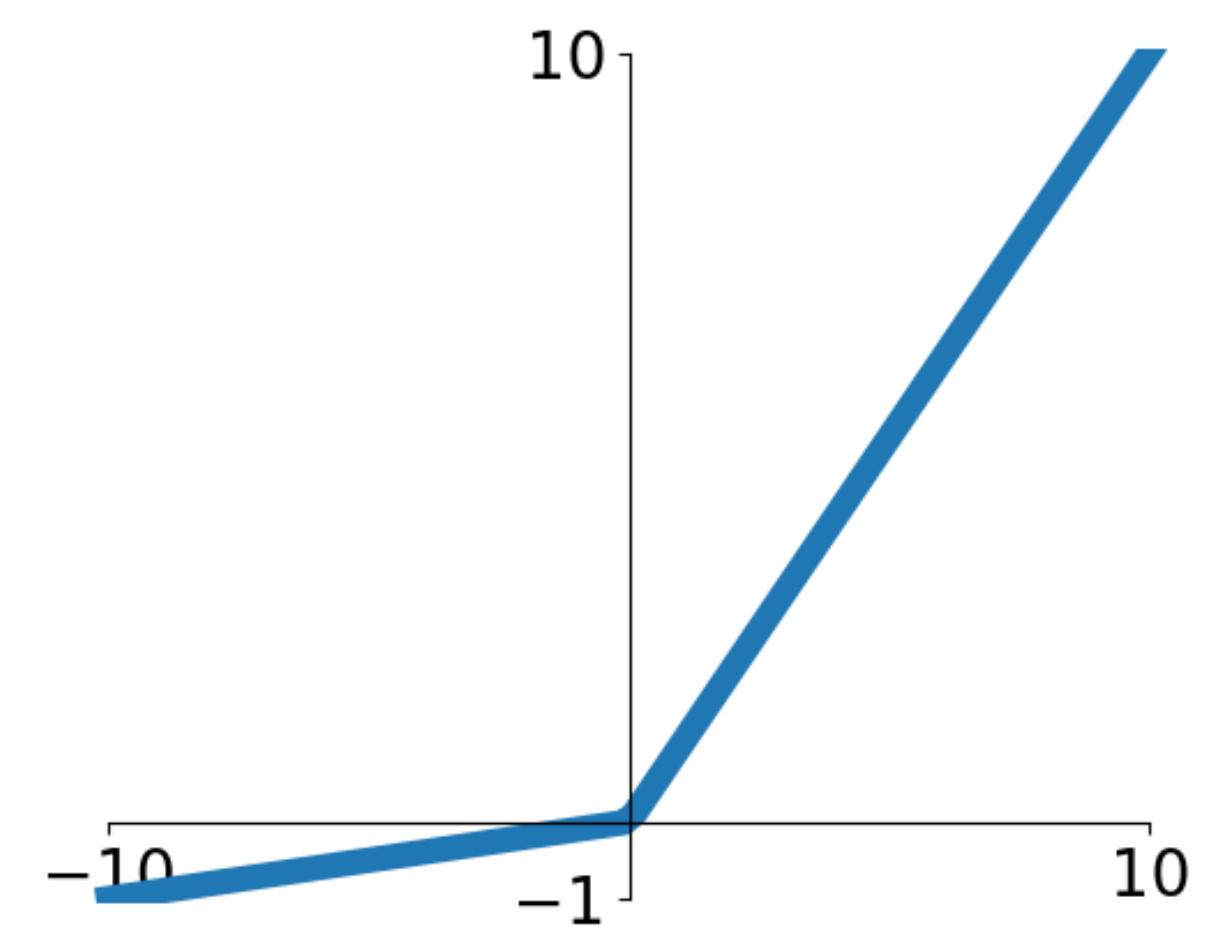
Leaky: alpha is fixed to a small value (e.g., 0.01)

$$a(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$

Parametrized: alpha is optimized as part of the network (BackProp through)

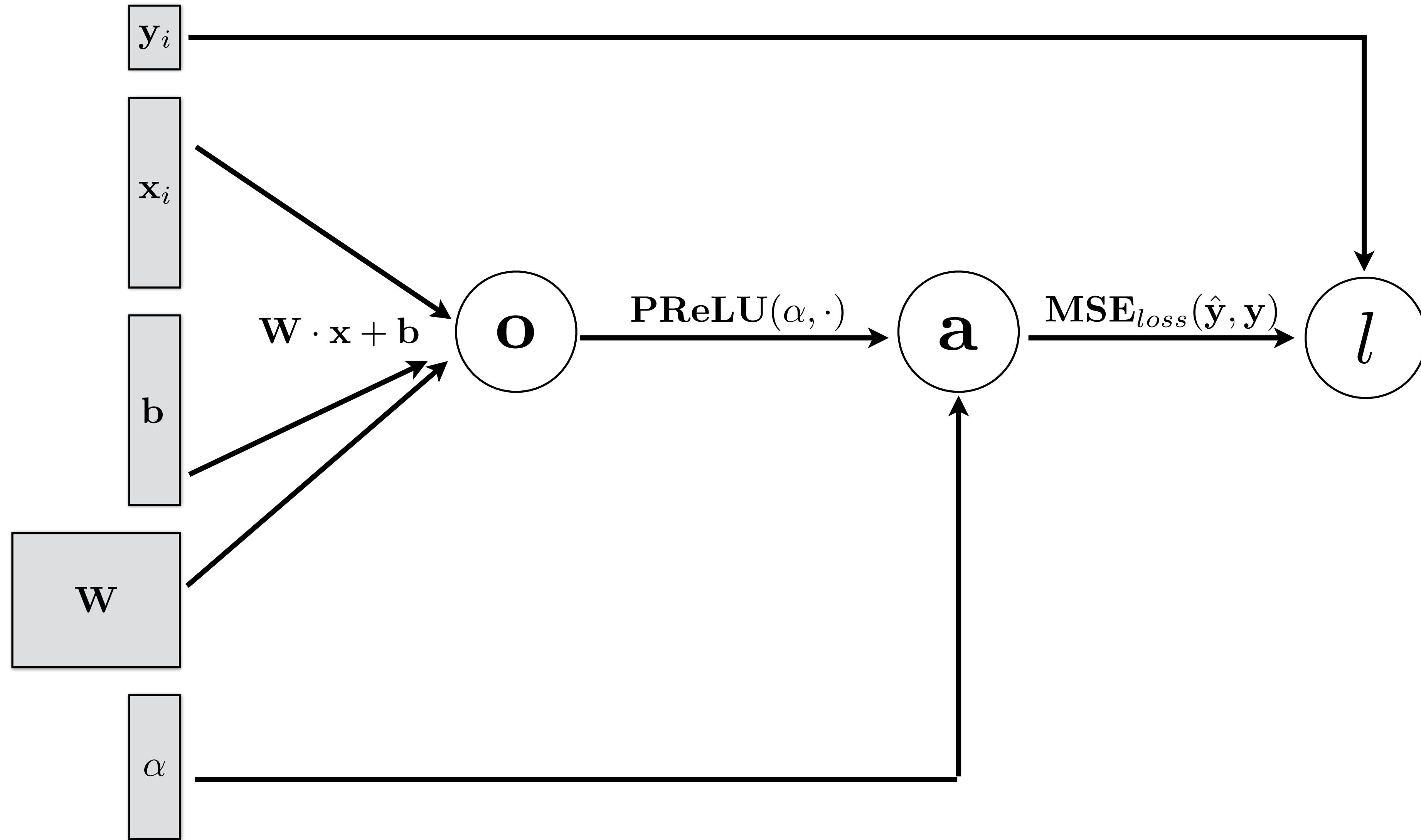
Pros:

- Does not saturate
- Computationally very efficient
- Converges faster in practice (e.g. 6x)



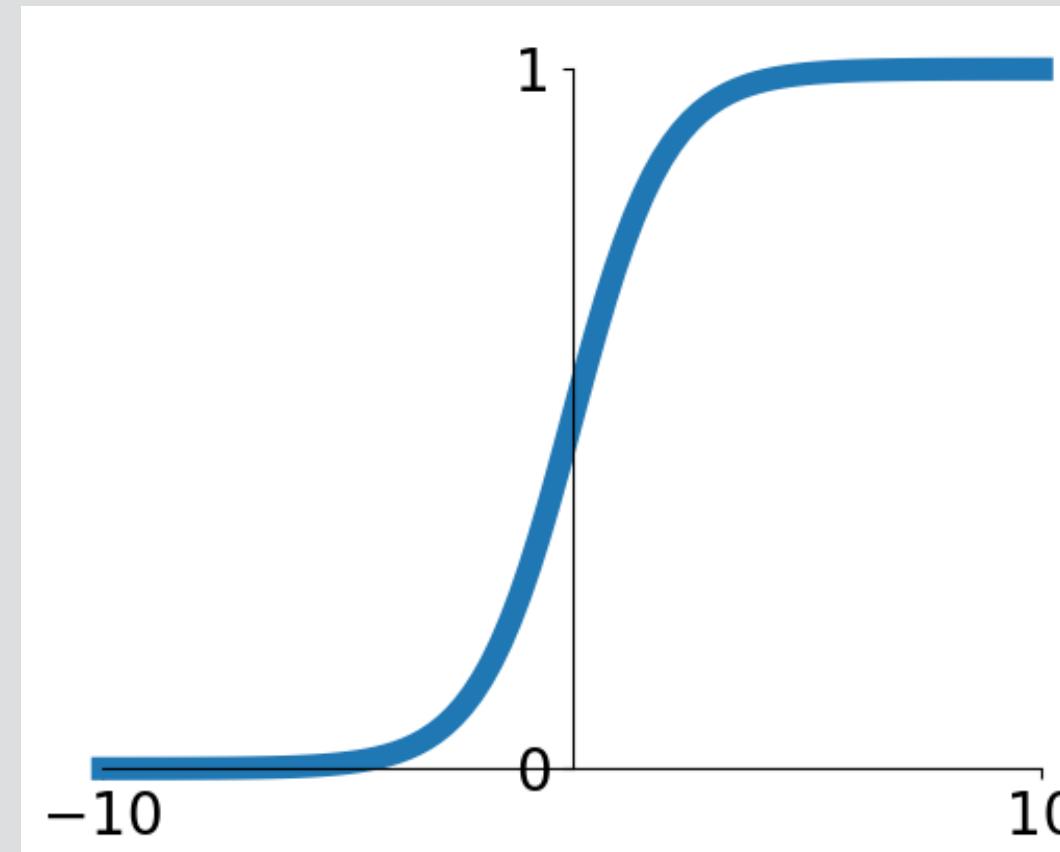
Leaky / Parametrized ReLU Activation

Computational Graph: 1-layer with PReLU



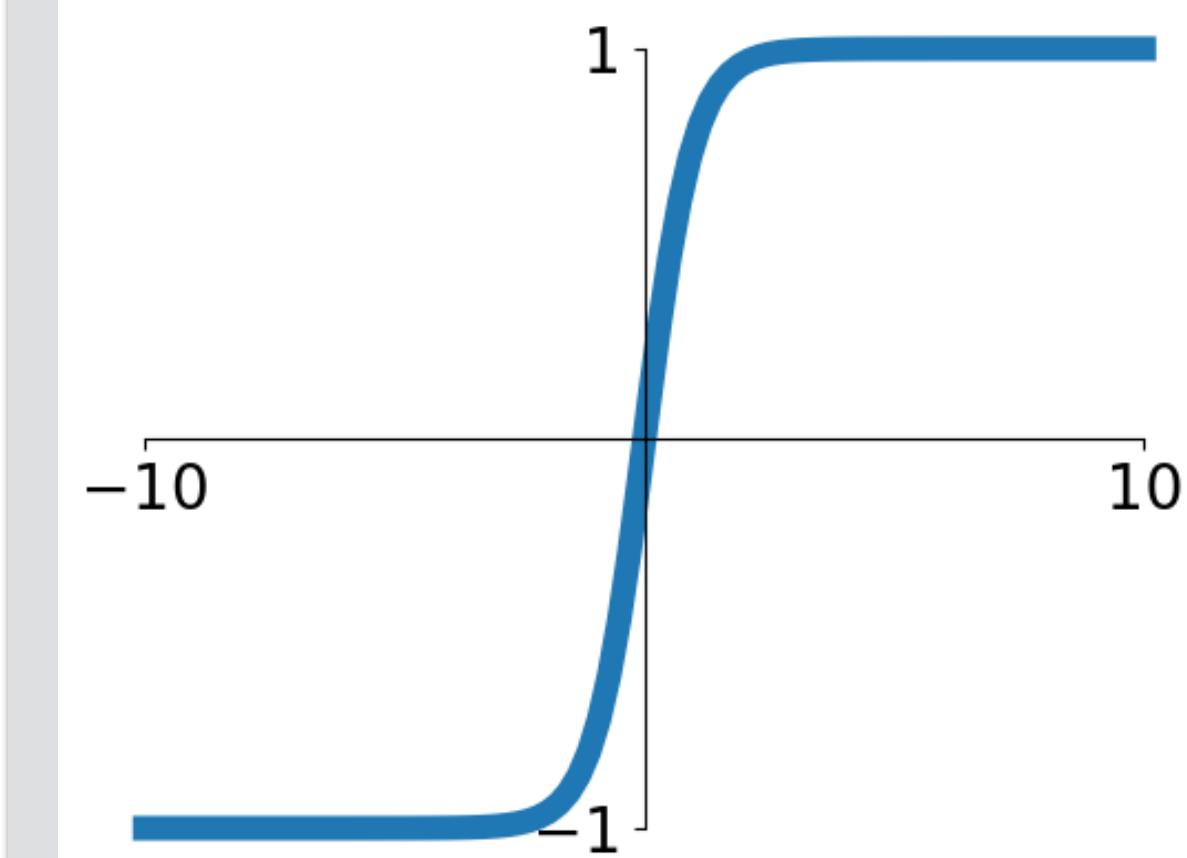
Activation Functions: Review

$$a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



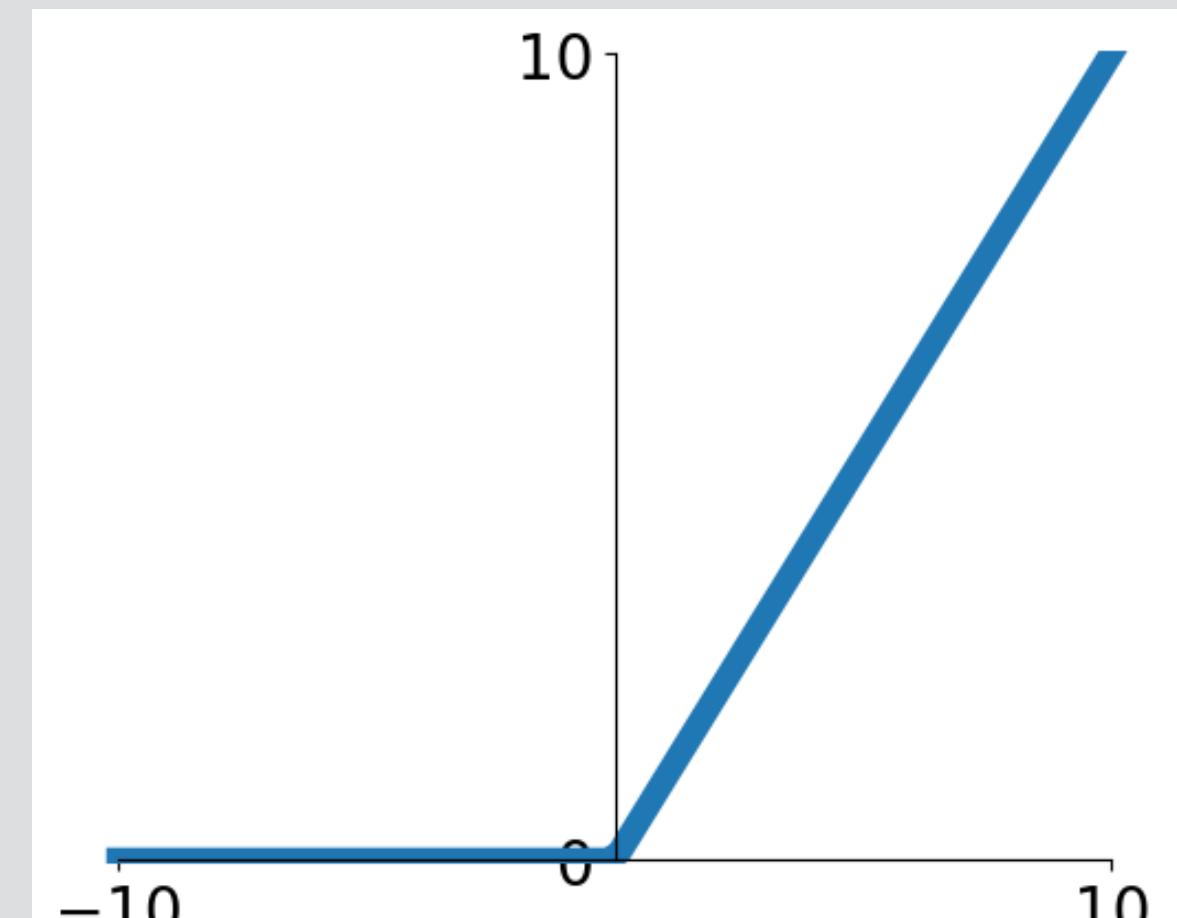
Sigmoid

$$a(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



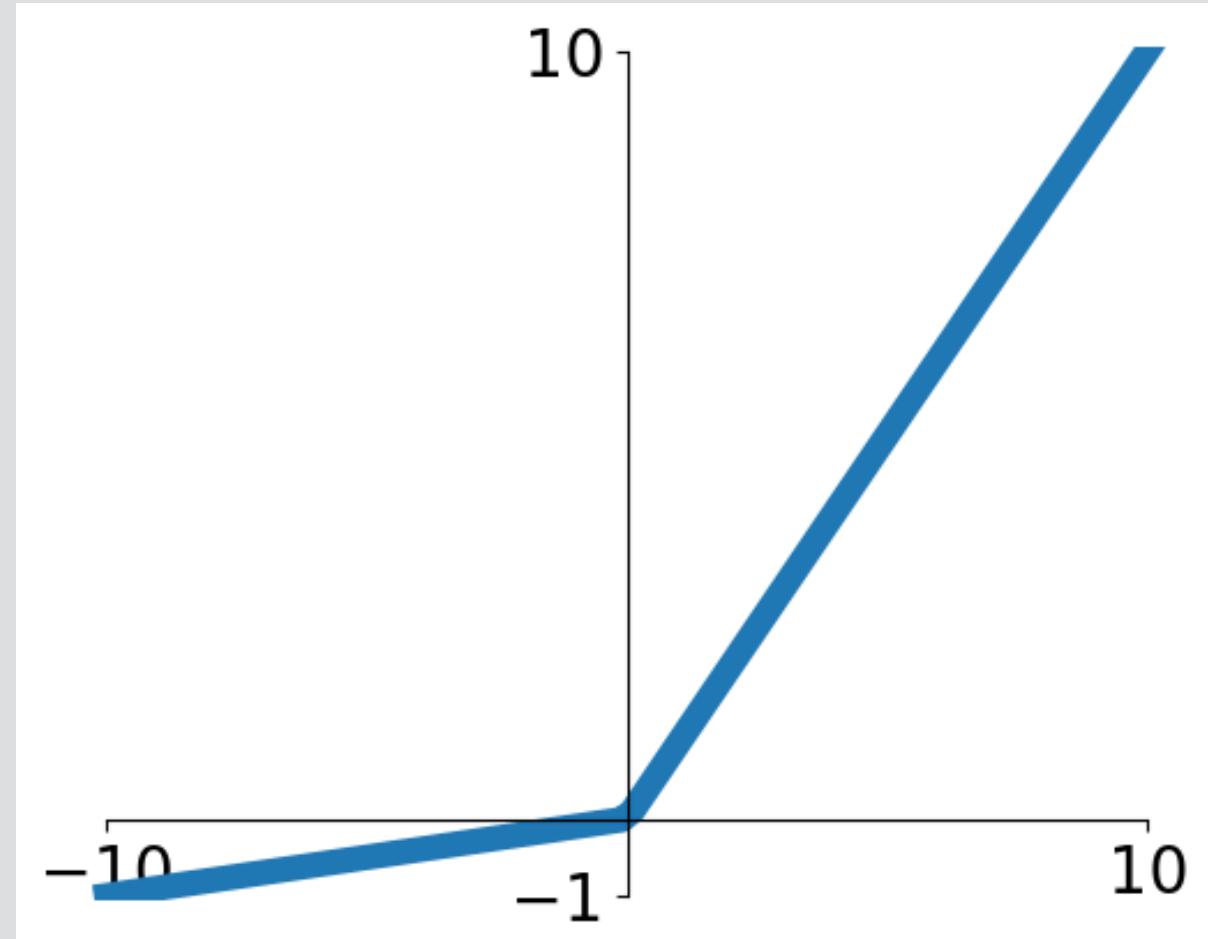
Tanh

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



ReLU

$$a(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$



Leaky / Parametrized ReLU