CS 540 Introduction to Artificial Intelligence

Neural Networks (II)

University of Wisconsin-Madison

Spring 2023

Announcements

- **Homeworks:**
    - HW 6 Due Tuesday, March 28
- **Midterms are being graded**
- **Class roadmap:**

Tuesday, Mar 21	ML: Neural Networks II
Thursday, Mar 23	ML: Neural Networks III
Tuesday, Mar 28	Deep Learning I
Thursday, Mar 30	Deep Learning II

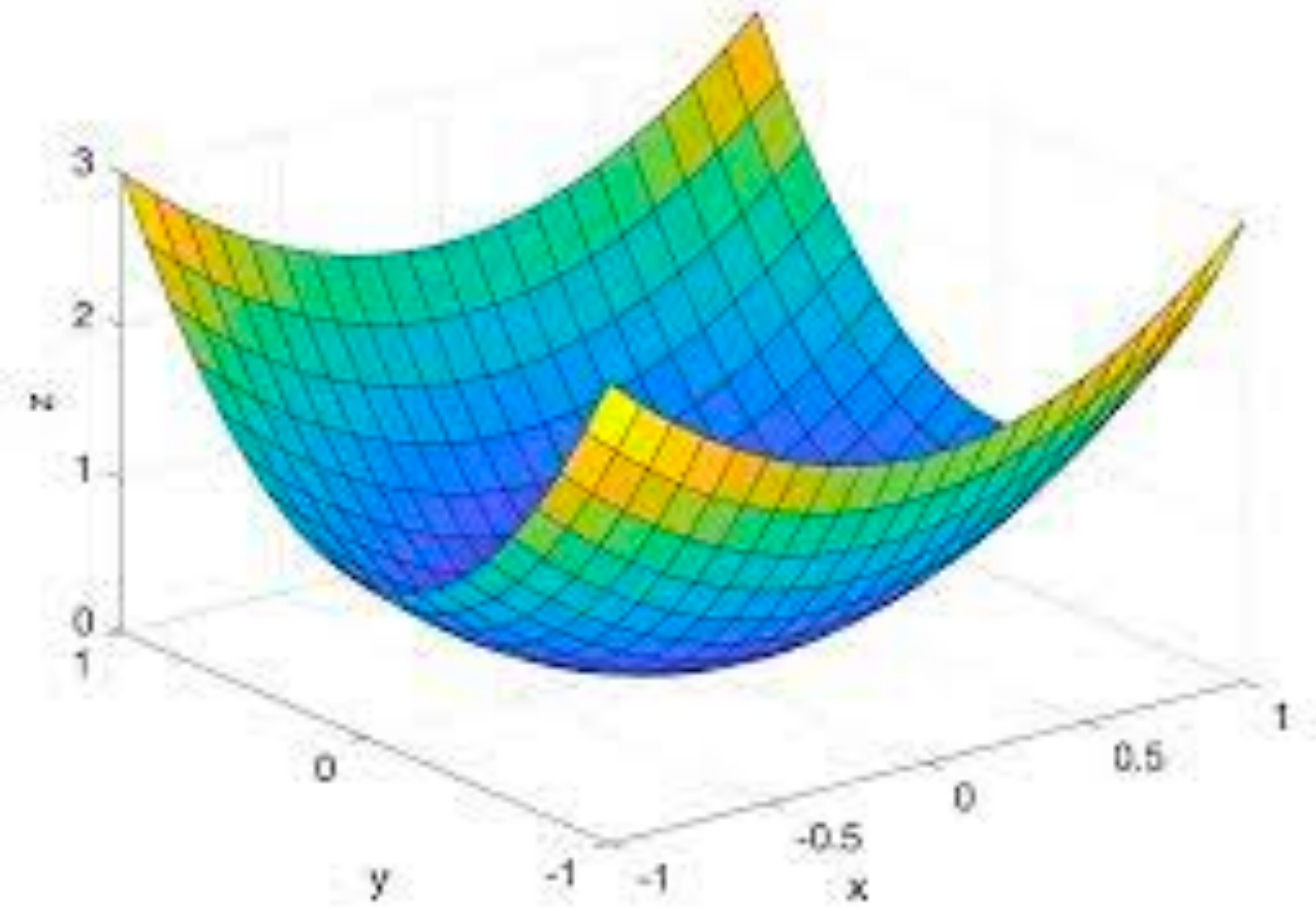
Section 001 Mid-Semester Evaluation

- Lecture
  - Slides not matching website?
  - Amount of material in some lectures.
- Assignments
  - Feedback on what is going wrong.
- Exams
  - Wanting to see more practice problems before the exam.

Today’s outline

- Multivariate Calculus Intro / Review
- Review of Multi-layer Perceptron
    - Single output
    - Multiple output
- How to train neural networks
    - Gradient descent
    - Backpropagation

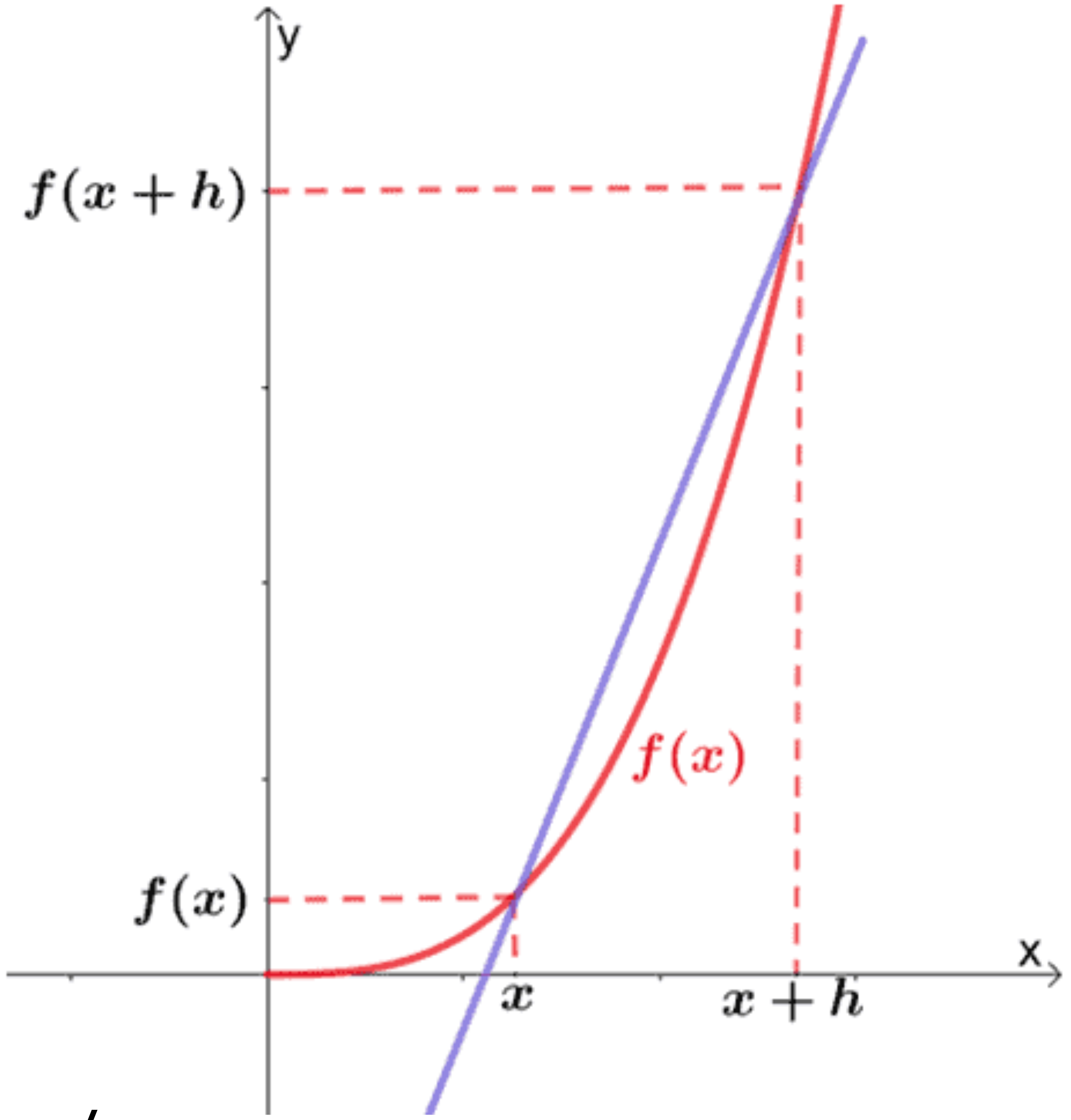
Multivariate Calculus



https://machinelearningmastery.com/a-gentle-introduction-to-multivariate-calculus/

Derivatives of functions of single variables

- Given function $f(x)$, we would like to find the slope of the tangent line at any point x_0.
    - Why? Tells us how fast $f(x)$ is increasing / decreasing at x_0.
- $\dfrac{df}{dx} = \lim_{\Delta x \to 0} \dfrac{f(x + \Delta x) - f(x)}{\Delta x}$



https://www.storyofmathematics.com/derivative-calculus/

Derivatives of functions of single variables

- For many functions $f(x)$ we have simple rules to find the derivative.
- If $f(x) = cx^2$ then $\frac{df}{dx} = 2cx$
    - Or more generally, if $f(x) = cx^p$ then $\frac{df}{dx} = cpx^{p-1}$
- If $f(x) = \log x$ then $\frac{df}{dx} = \frac{1}{x}$
- If $f(x) = c$ then $\frac{df}{dx} = 0$

https://www.storyofmathematics.com/derivative-calculus/

More Complex Functions

- Derivation rules can be applied hierarchically to find derivatives of more complex functions.
- **Sum rule:** If $f(x) = h(x) + g(x)$ then $\dfrac{df}{dx} = \dfrac{dh}{dx} + \dfrac{dg}{dx}$
- **Product rule:** If $f(x) = h(x) \cdot g(x)$ then $\dfrac{df}{dx} = h(x)\dfrac{dg}{dx} + g(x)\dfrac{dh}{dx}$
- **Chain rule:** If $f(x) = h(g(x))$ then $\dfrac{df}{dx} = \dfrac{dh}{dg}\dfrac{dg}{dx}$
- **More complex chain rule:** if $f(x) = f_1(f_2(\cdots f_n(x)\cdots))$ then $\dfrac{df}{dx} = \dfrac{df_1}{df_2}\dfrac{df_2}{df_3}\cdots\dfrac{df_n}{dx}$

https://www.storyofmathematics.com/derivative-calculus/

Derivatives of functions of multiple variables

- Generalize derivative to functions of form $f(x_1, x_2, ..., x_n)$.
- The partial derivative $\frac{\partial f}{\partial x_1}$ tells us how fast f increases / decreases as we increase the value of x_1.
- For each variable x_i we have a partial derivative $\frac{\partial f}{\partial x_i}$.
- The **gradient** is the vector of all of these partial derivatives.
  - $\frac{df}{d\mathbf{x}} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n},]$

https://www.storyofmathematics.com/derivative-calculus/

Computing Partial Derivatives

- Rules from single-variable calculus directly extend to multivariate calculus with one small change.
  - When computing $\frac{\partial f}{\partial x_i}$, treat all other variables as constants.
- Examples:
  - If $f(x_1, x_2) = \log x_1 + \log x_2$ then $\frac{\partial f}{\partial x_1} = \frac{1}{x_1}$
  - If $f(x_1, x_2) = x_1(x_1 + x_2)$ then $\frac{\partial f}{\partial x_2} = x_1$

https://www.storyofmathematics.com/derivative-calculus/

Quiz Break

- What is the partial derivative $\frac{\partial f}{\partial w_1}$ of:

$$f(x_1, x_2, w_1, w_2, y) = y \log \sigma(w_1 x_1 + w_2 x_2) + (1 - y) \log(1 - \sigma(w_1 x_1 + w_2 x_2))$$

when $y = 1$ and $\sigma(z) = \frac{1}{1 + e^{-z}}$. **Hint**: $\frac{\partial \sigma}{\partial z} = \sigma(z)(1 - \sigma(z)$.

Quiz Break

- What is the partial derivative $\dfrac{\partial f}{\partial w_1}$ of:

$$f(x_1, x_2, w_1, w_2, y) = y \log \sigma(w_1 x_1 + w_2 x_2) + (1 - y)\log(1 - \sigma(w_1 x_1 + w_2 x_2))$$

when $y = 1$ and $\sigma(z) = \dfrac{1}{1 + e^{-z}}$. **Hint:** $\dfrac{\partial \sigma}{\partial z} = \sigma(z)(1 - \sigma(z))$.

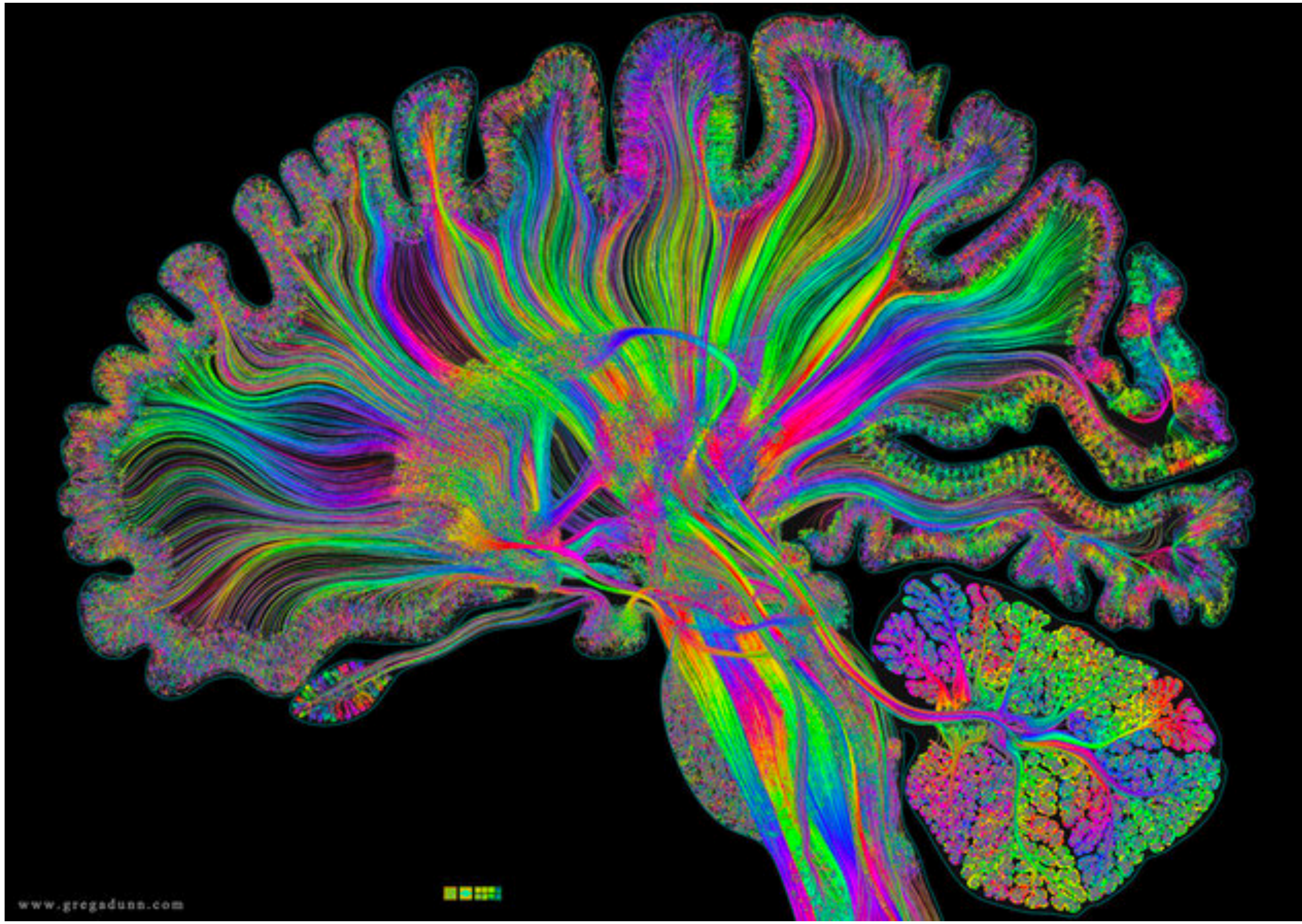
Let $a = \sigma(b)$

Let $b = w_1 x_1 + w_2 x_2$

$$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial b}\frac{\partial b}{w_1}$$

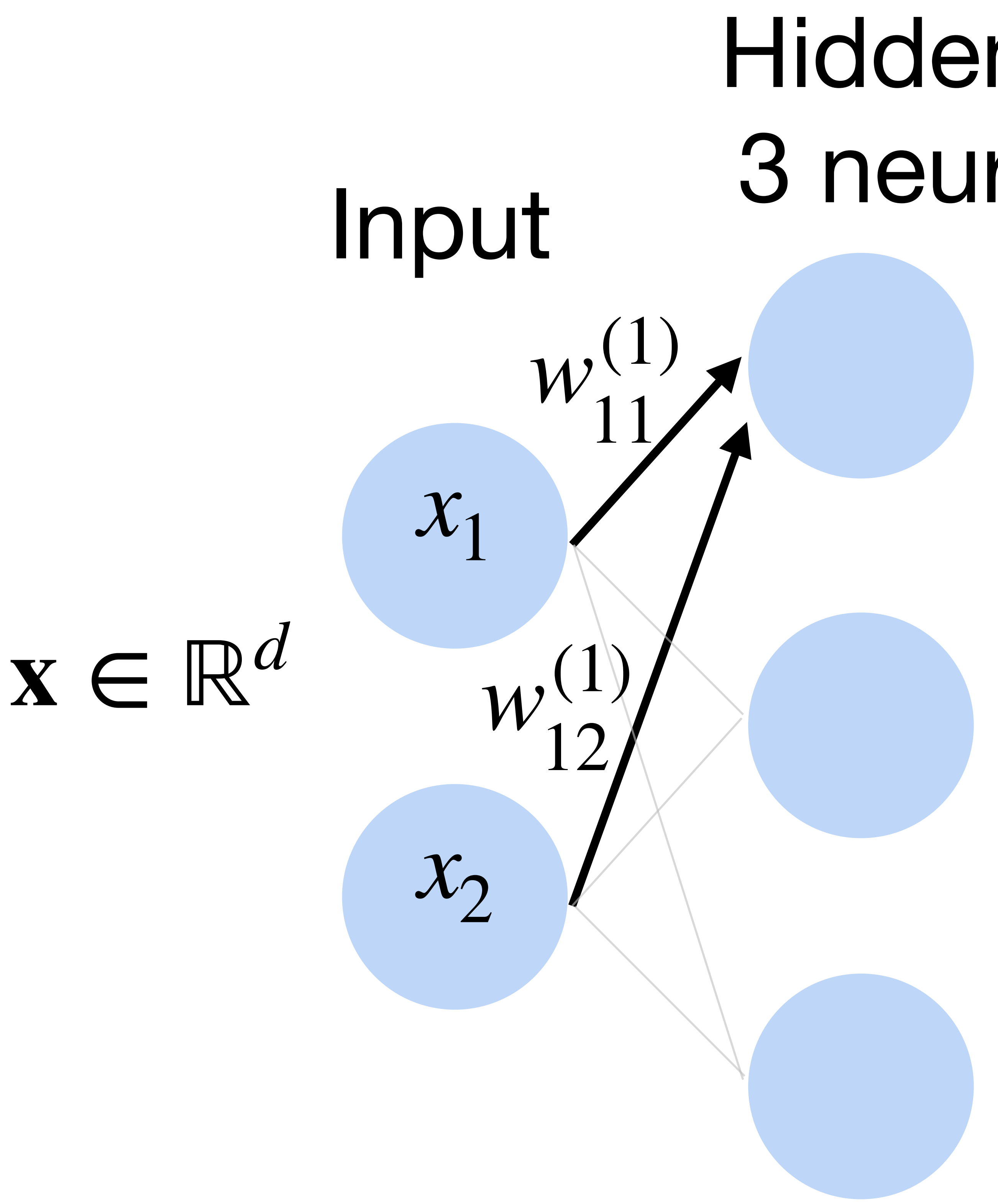
$$\frac{\partial f}{\partial w_1} = \frac{y}{a}\sigma(b)(1 - \sigma(b))x_2 = (1 - \sigma(w_1 x_2 + w_2 x_2))x_2$$

Multilayer Perceptron



Multi-layer perceptron: Example

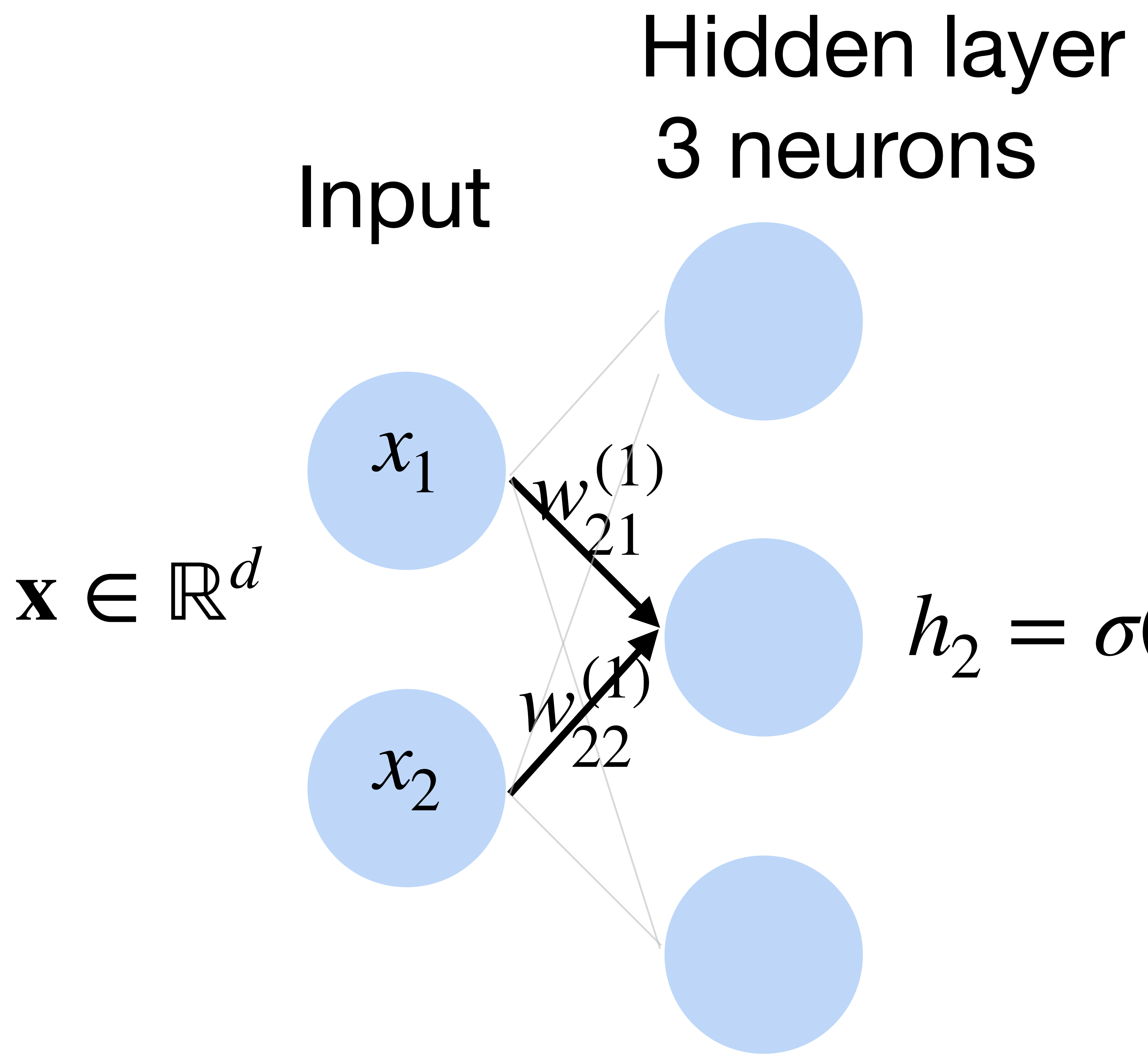
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_1 = \sigma\left(\sum_{i=1}^{d} x_i w_{1i}^{(1)} + b_1\right)$$

Multi-layer perceptron: Example

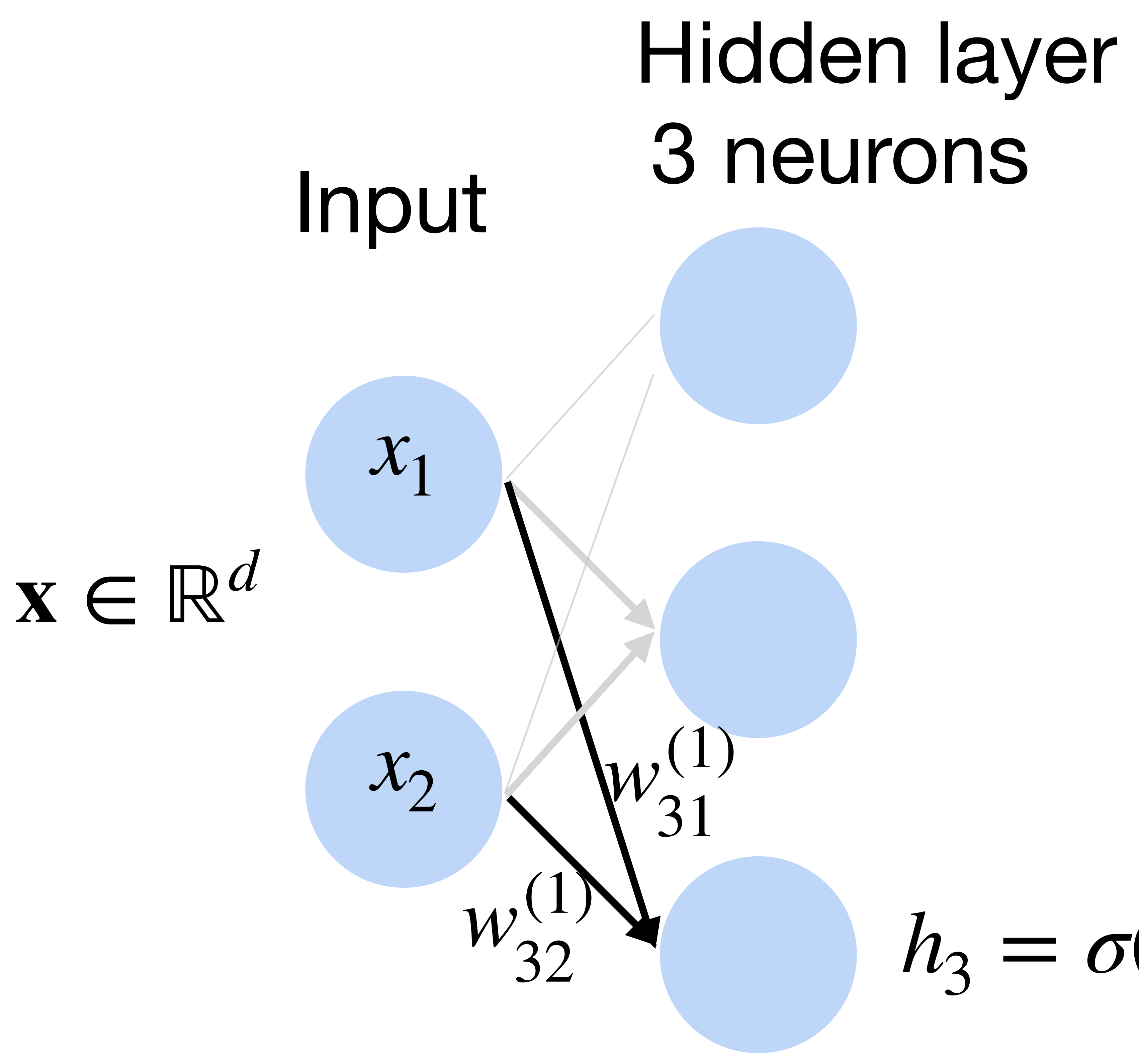
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_2 = \sigma(\sum_{i=1}^{d} x_i w_{2i}^{(1)} + b_2)$$

Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_3 = \sigma(\sum_{i=1}^{d} x_i w_{3i}^{(1)} + b_3)$$

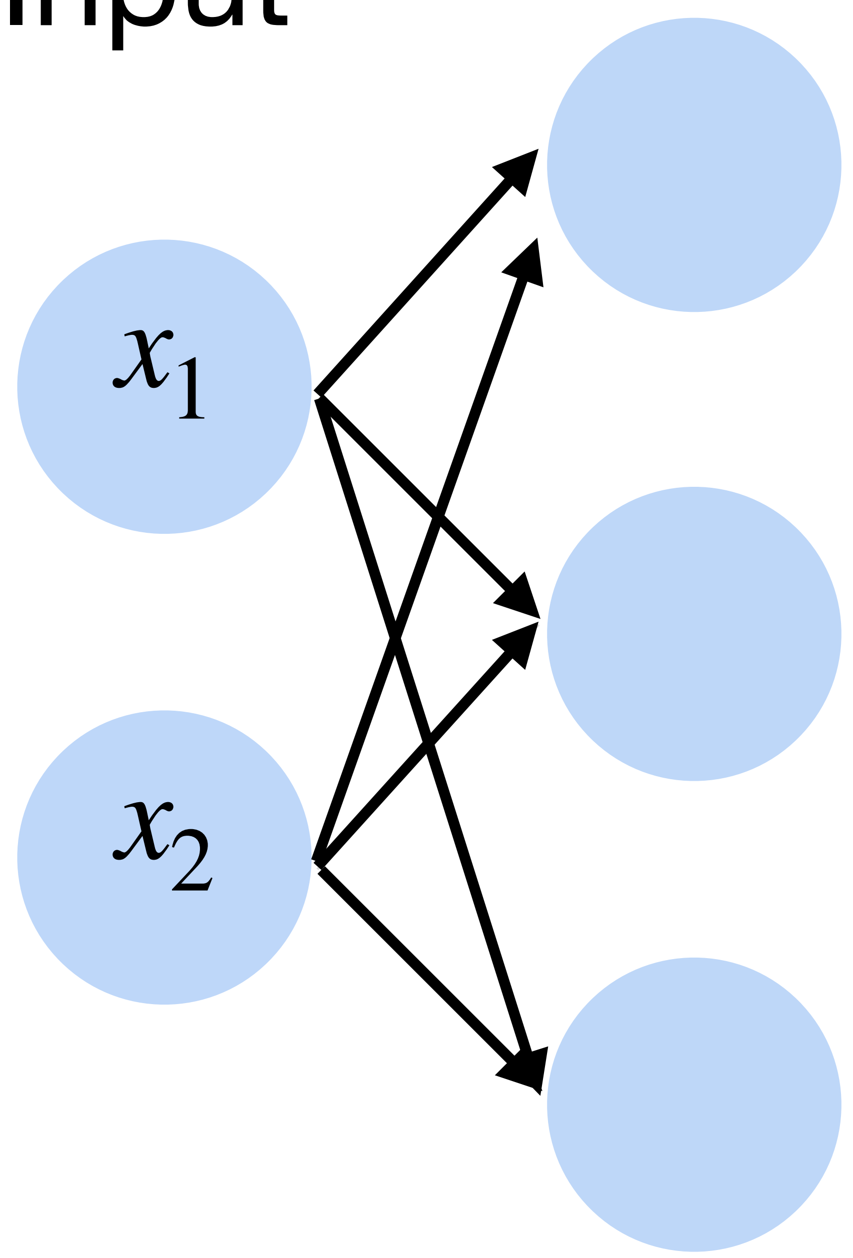
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2

Hidden layer
m=3 neurons

Input

$\mathbf{x} \in \mathbb{R}^d$



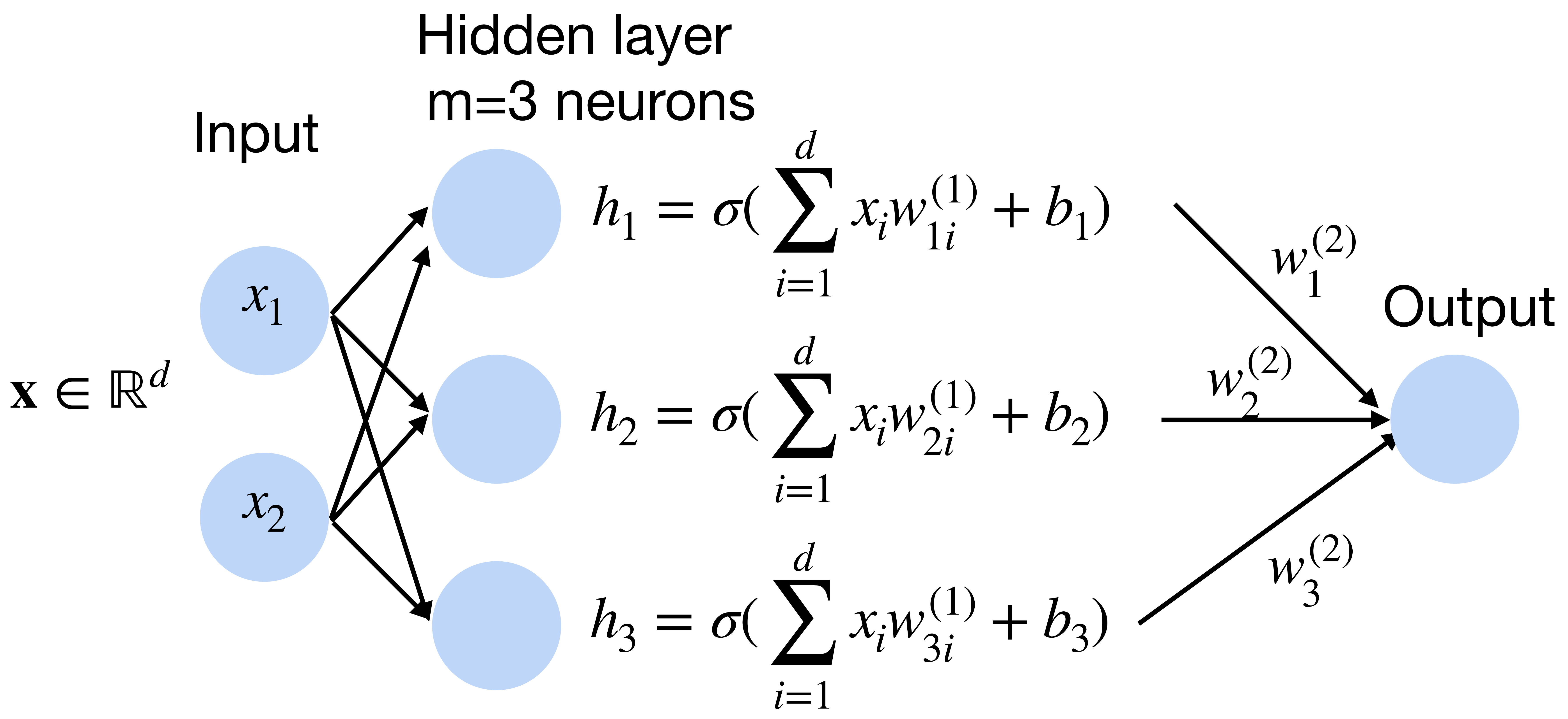
$$h_1 = \sigma(\sum_{i=1}^{d} x_i w_{1i}^{(1)} + b_1)$$

$$h_2 = \sigma(\sum_{i=1}^{d} x_i w_{2i}^{(1)} + b_2)$$

$$h_3 = \sigma(\sum_{i=1}^{d} x_i w_{3i}^{(1)} + b_3)$$

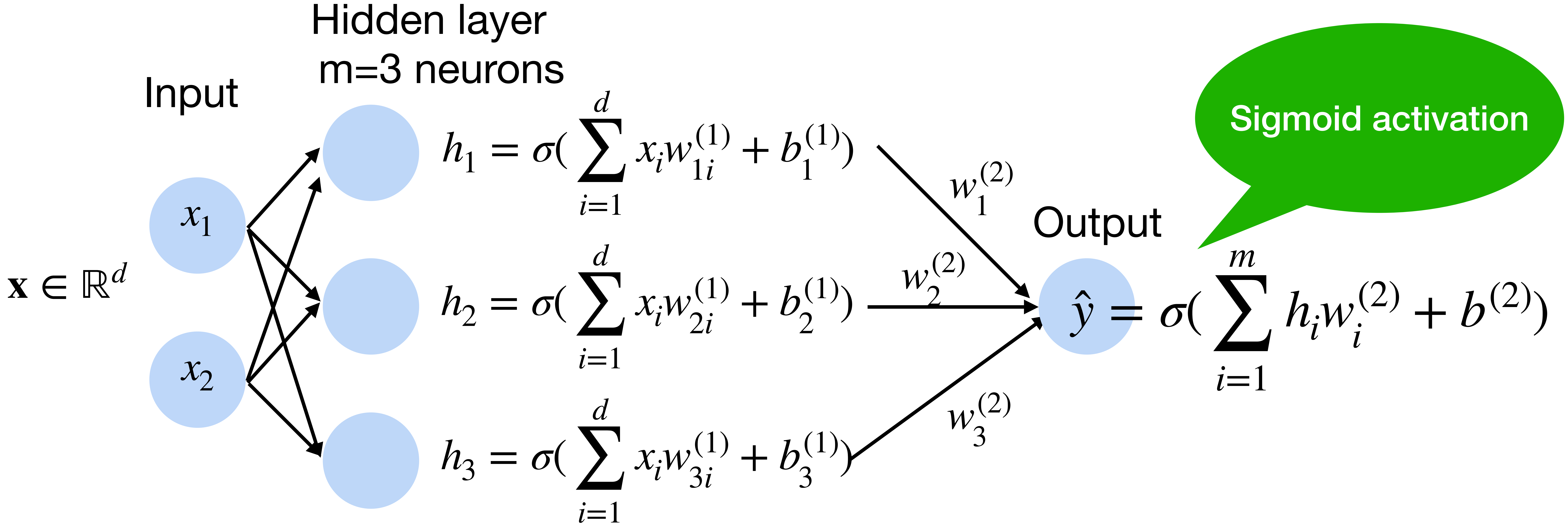
Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



Multi-layer perceptron: Example

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



$$h_1 = \sigma\left(\sum_{i=1}^{d} x_i w_{1i}^{(1)} + b_1^{(1)}\right)$$

$$h_2 = \sigma\left(\sum_{i=1}^{d} x_i w_{2i}^{(1)} + b_2^{(1)}\right)$$

$$h_3 = \sigma\left(\sum_{i=1}^{d} x_i w_{3i}^{(1)} + b_3^{(1)}\right)$$

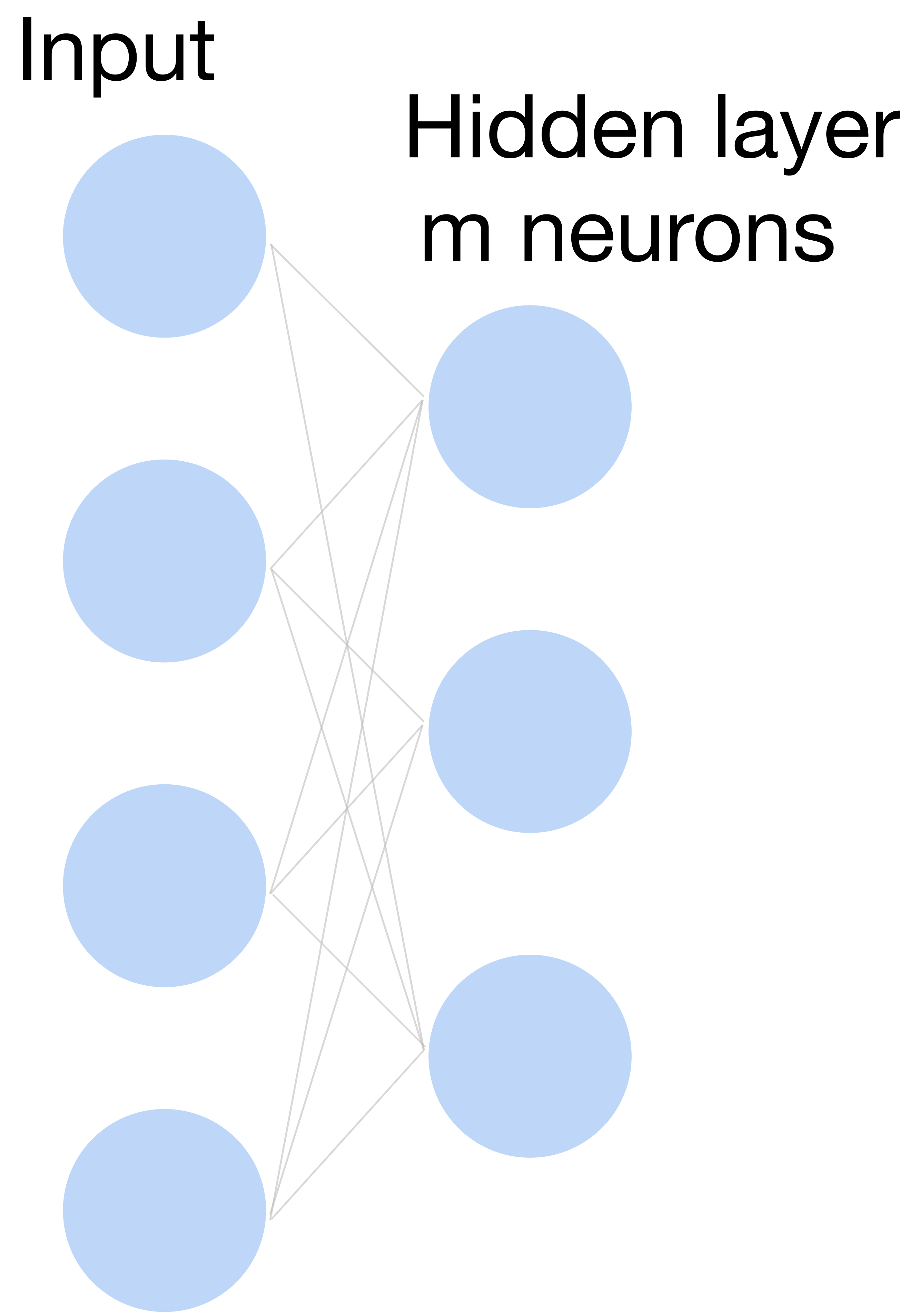
$$\hat{y} = \sigma\left(\sum_{i=1}^{m} h_i w_i^{(2)} + b^{(2)}\right)$$

Multi-layer perceptron: Matrix Notation

- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h} \in \mathbb{R}^m$$

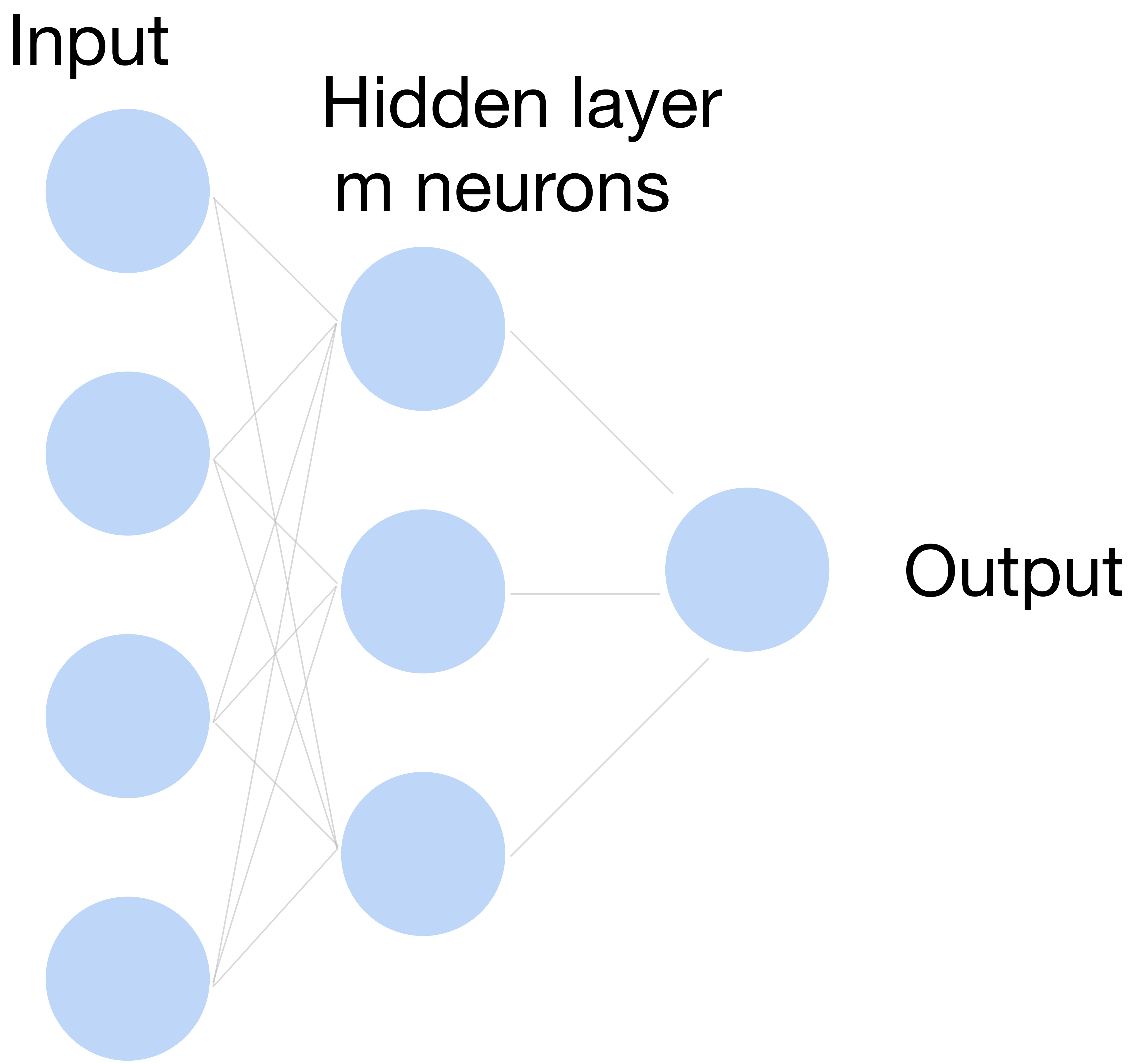


Multi-layer perceptron: Matrix Notation

- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}, \mathbf{b}^{(1)} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

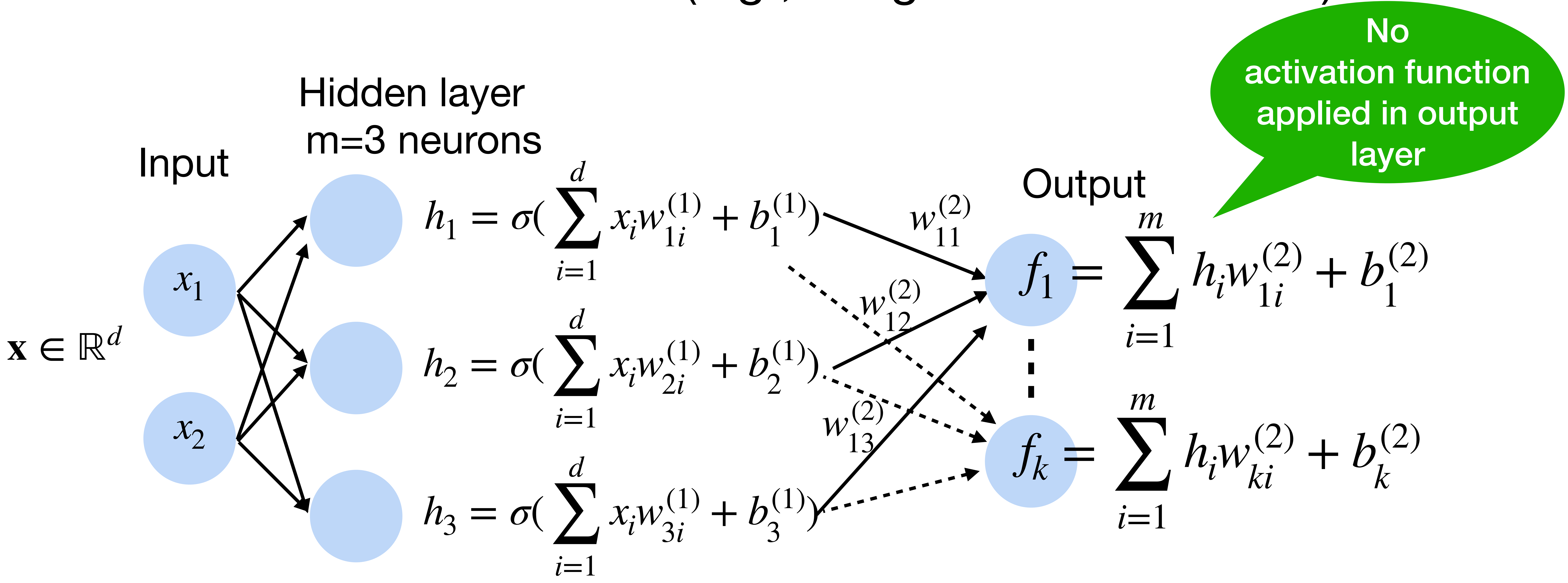
$$\hat{y} = \sigma(\mathbf{w}^{(2)}\mathbf{h} + \mathbf{b}^{(2)})$$



Neural network for K-way classification

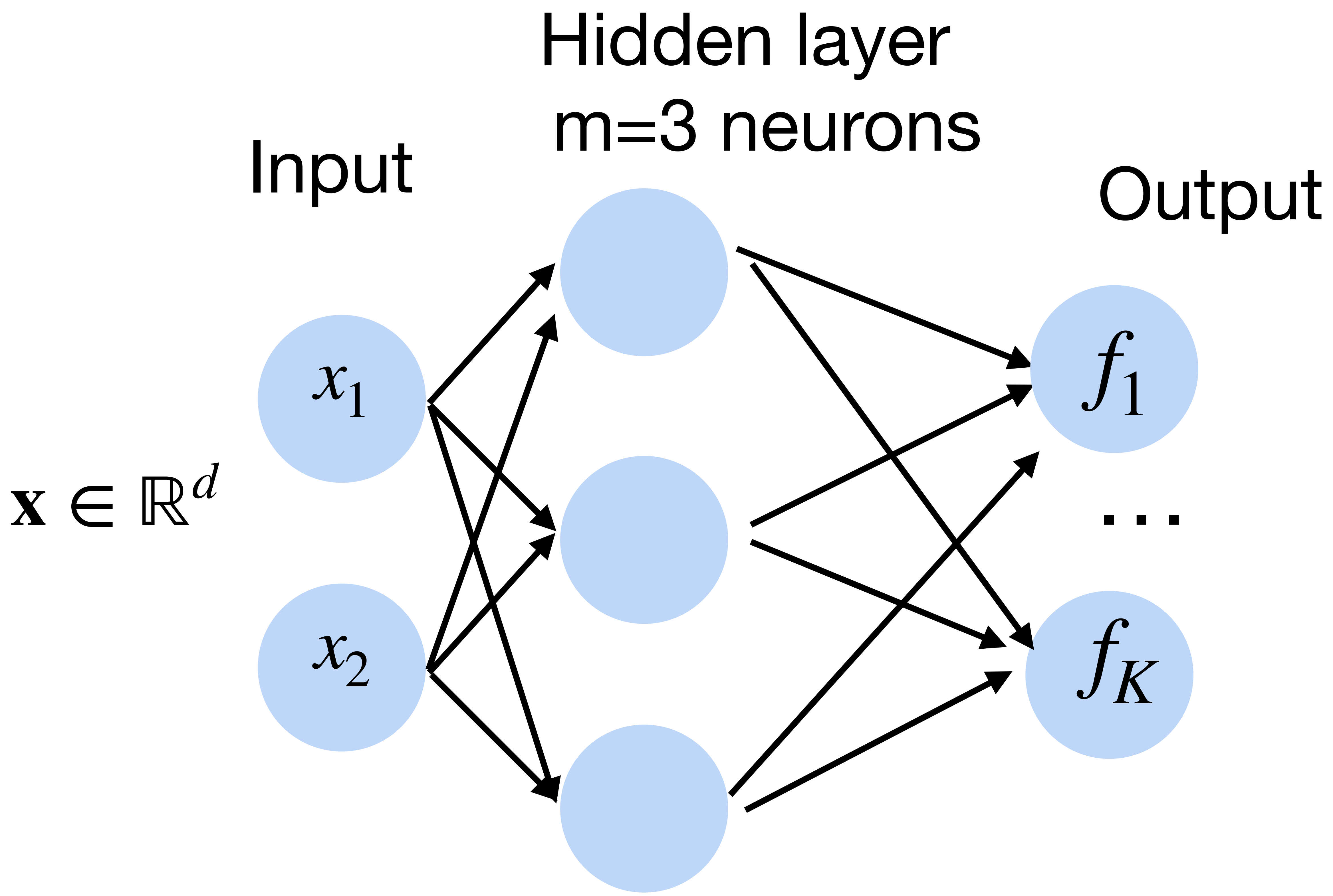
- K outputs in the final layer

Multi-class classification (e.g., ImageNet with K=1000)



Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)

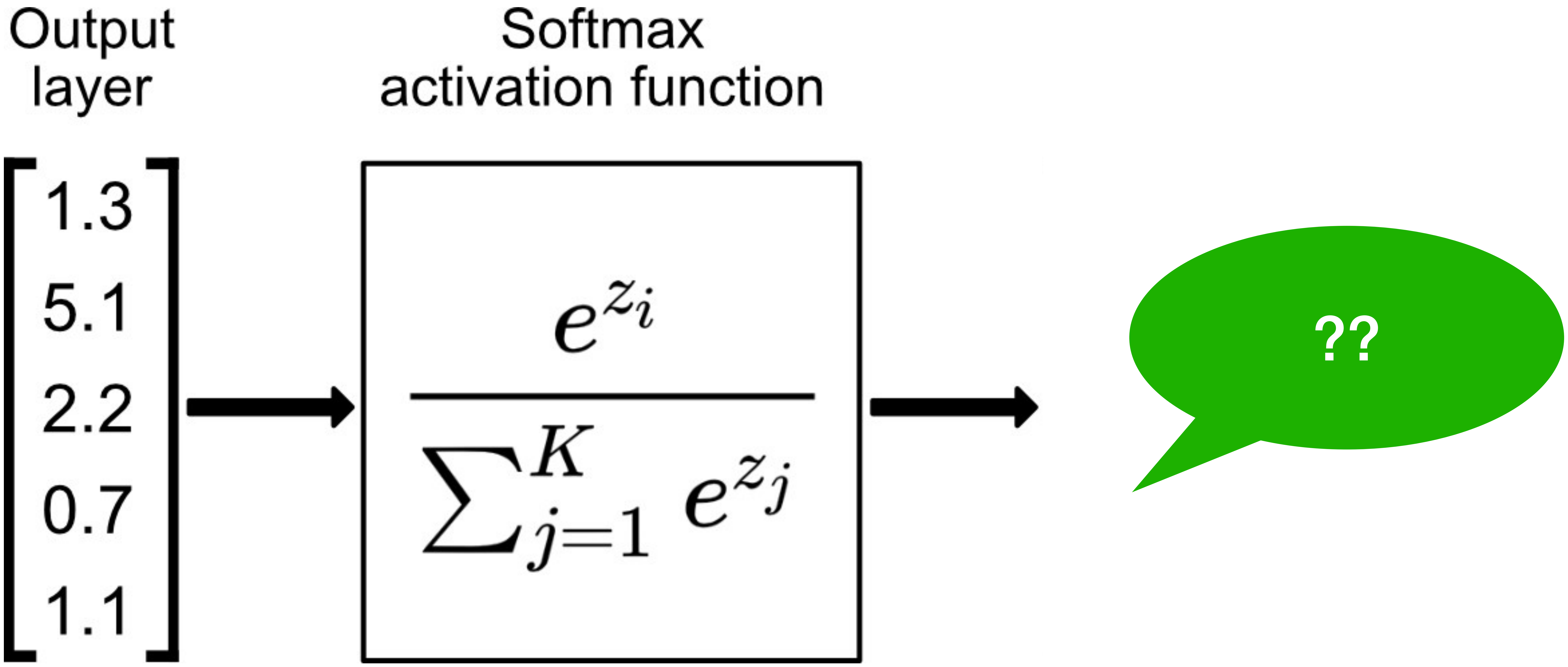


$$p(y \mid \mathbf{x}) = \text{softmax}(f)$$

$$= \frac{\exp(f_y(x))}{\sum_{k=1}^{K} \exp(f_k(x))}$$

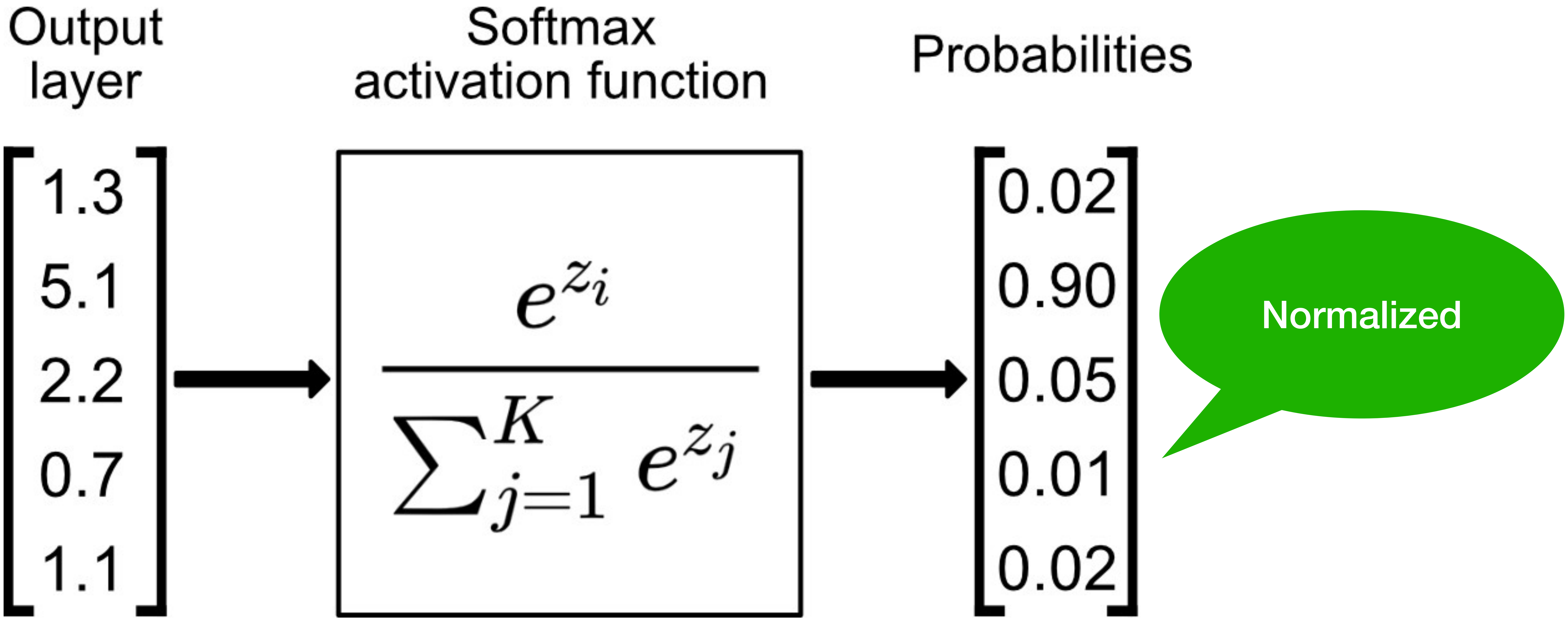
Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)



Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)



More complicated neural networks: multiple hidden layers

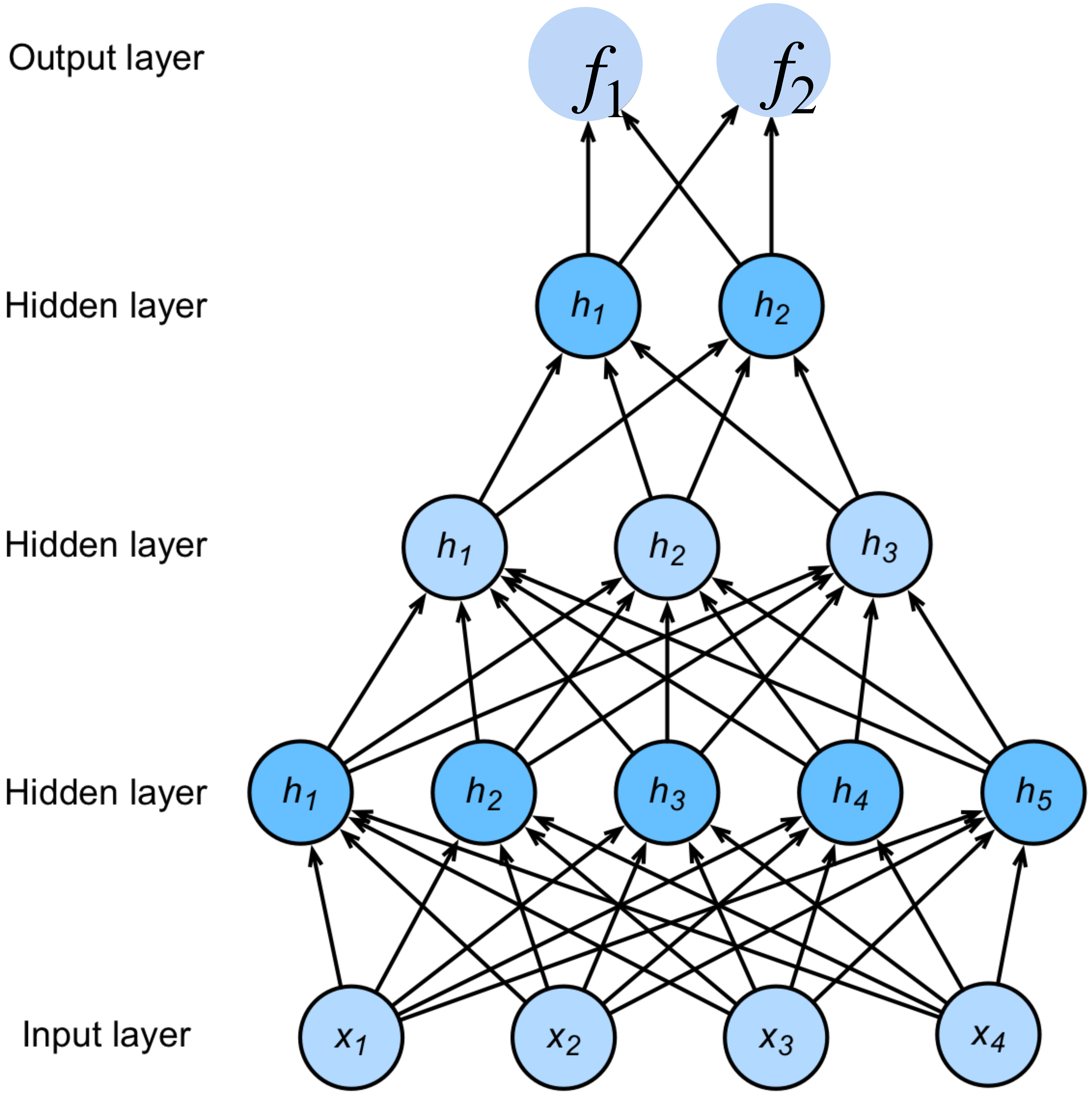
$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}^{(2)}\mathbf{h}_1 + \mathbf{b}^{(2)})$$

$$\mathbf{h}_3 = \sigma(\mathbf{W}^{(3)}\mathbf{h}_2 + \mathbf{b}^{(3)})$$

$$\mathbf{f} = \mathbf{W}^{(4)}\mathbf{h}_3 + \mathbf{b}^{(4)}$$

$$\mathbf{p} = \text{softmax}(\mathbf{f})$$



Quiz Break

Which output function is often used for multi-class classification tasks?

A Sigmoid function

B Rectified Linear Unit (ReLU)

C Softmax function

D Max function

Quiz Break

Which output function is often used for multi-class classification tasks?

A Sigmoid function

B Rectified Linear Unit (ReLU)

C Softmax function

D Max function

Quiz Break

Suppose you are given a 3-layer multilayer perceptron (2 hidden layers h1 and h2 and 1 output layer). All activation functions are sigmoids, and the output layer uses a softmax function. Suppose h1 has 1024 units and h2 has 512 units. Given a dataset with 2 input features and 3 unique class labels, how many learnable parameters does the perceptron have in total?

Quiz Break

Suppose you are given a 3-layer multilayer perceptron (2 hidden layers h1 and h2 and 1 output layer). All activation functions are sigmoids, and the output layer uses a softmax function. Suppose h1 has 1024 units and h2 has 512 units. Given a dataset with 2 input features and 3 unique class labels, how many learnable parameters does the perceptron have in total?

$1024 * 2 + 1024 + 512 * 1024 + 512 + 512 * 3 + 3 = 529411$

Quiz Break

Consider a three-layer network with **linear Perceptrons** for binary classification. The hidden layer has 3 neurons. Can the network represent a XOR problem?

a) Yes

b) No

Quiz Break

Consider a three-layer network with **linear Perceptrons** for binary classification. The hidden layer has 3 neurons. Can the network represent a XOR problem?

a) Yes

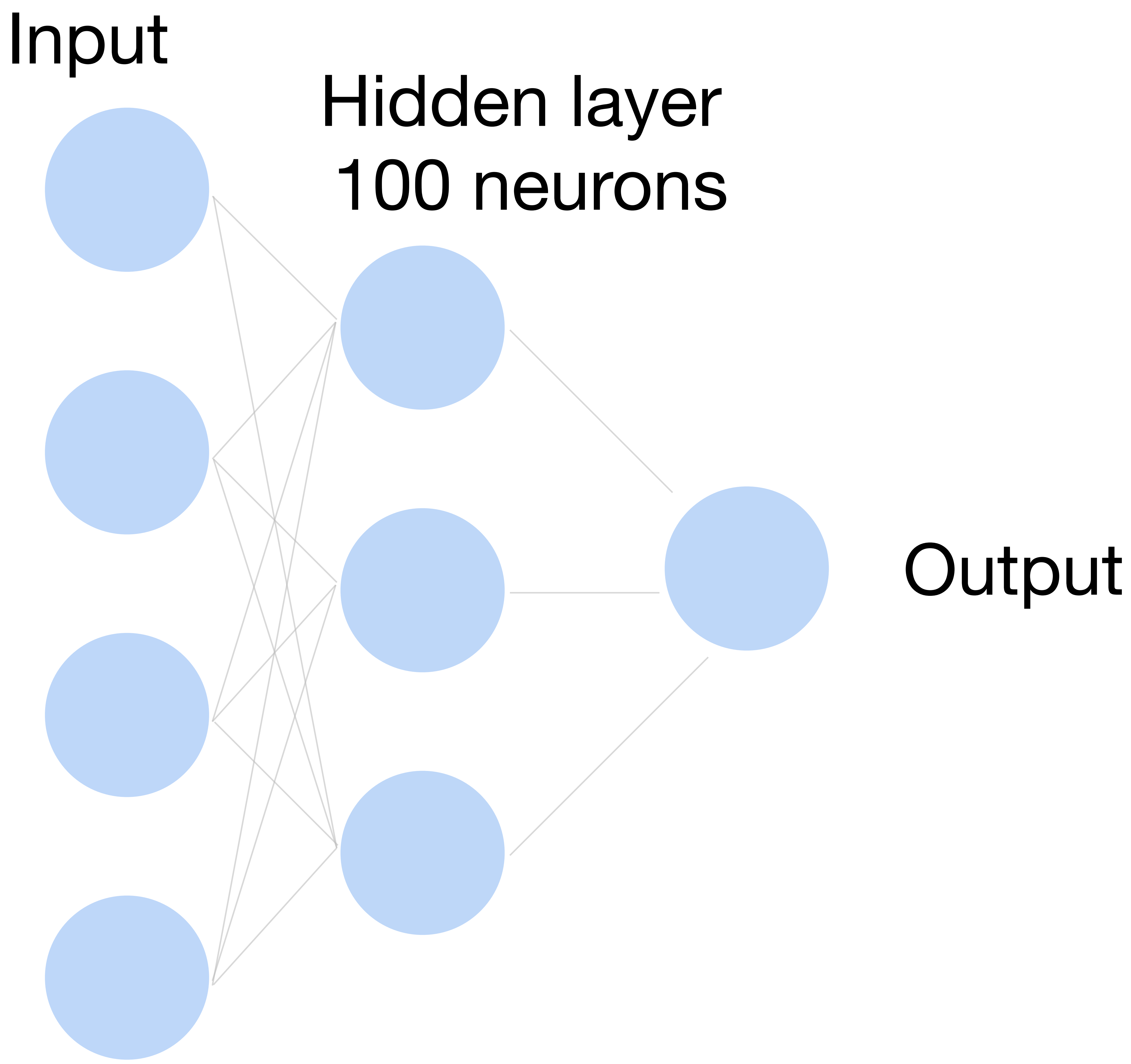
b) No

Solution:

A combination of linear Perceptrons is still a linear function.

How to train a neural network?

Classify cats vs. dogs



How to train a neural network? Binary classification

$\mathbf{x} \in \mathbb{R}^d$ One training data point in the training set D

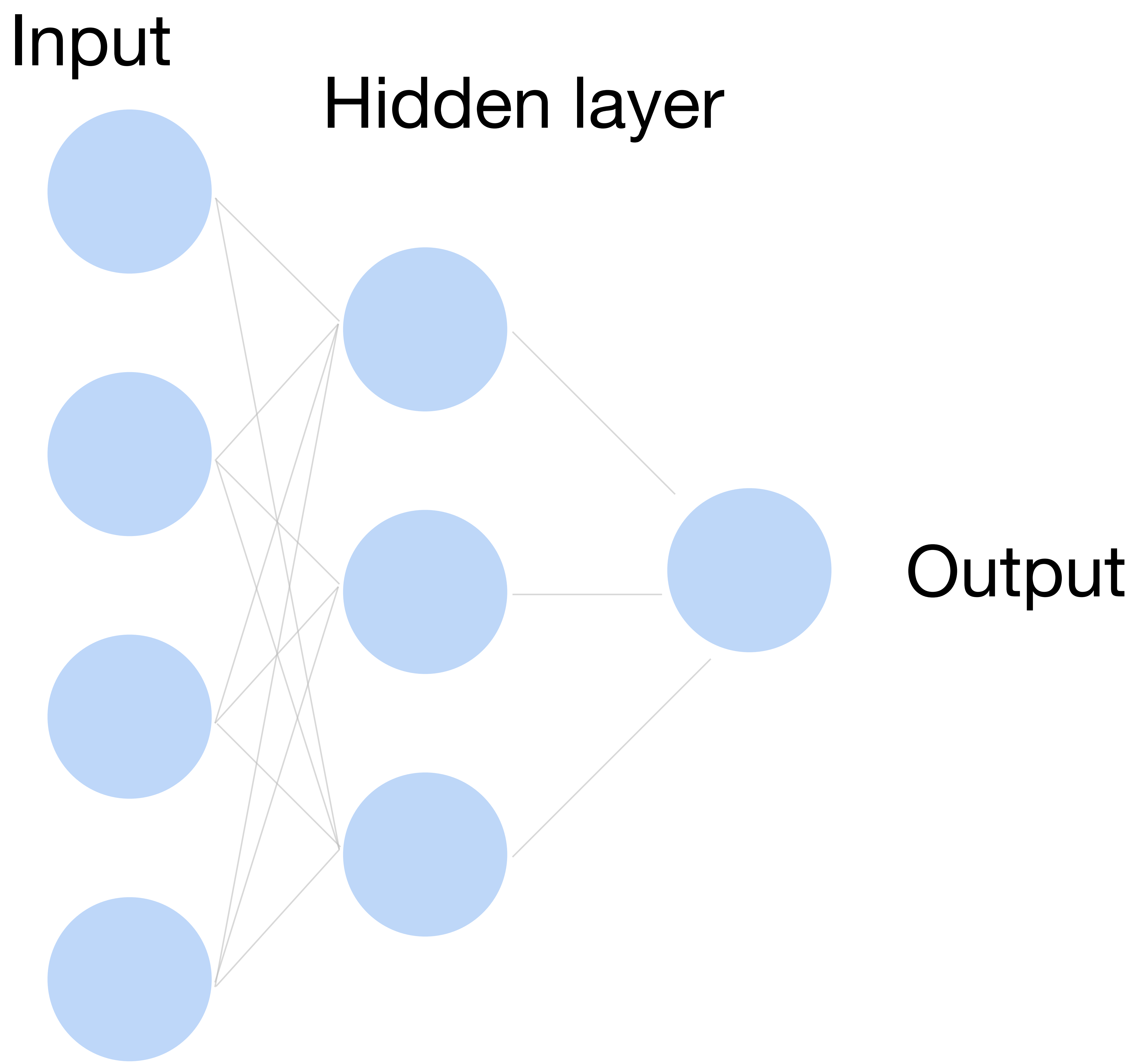
$\hat{y} \in [0,1]$ Model output for example **x**

(This is a function of all weights W: $\hat{y} = g(W)$)

y Ground truth label for example **x**

Learning by matching the output to the label

We want $\hat{y} \to 1$ **when** $y = 1$**,**
and $\hat{y} \to 0$ **when** $y = 0$



How to train a neural network? Binary classification

Loss function: $\frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$

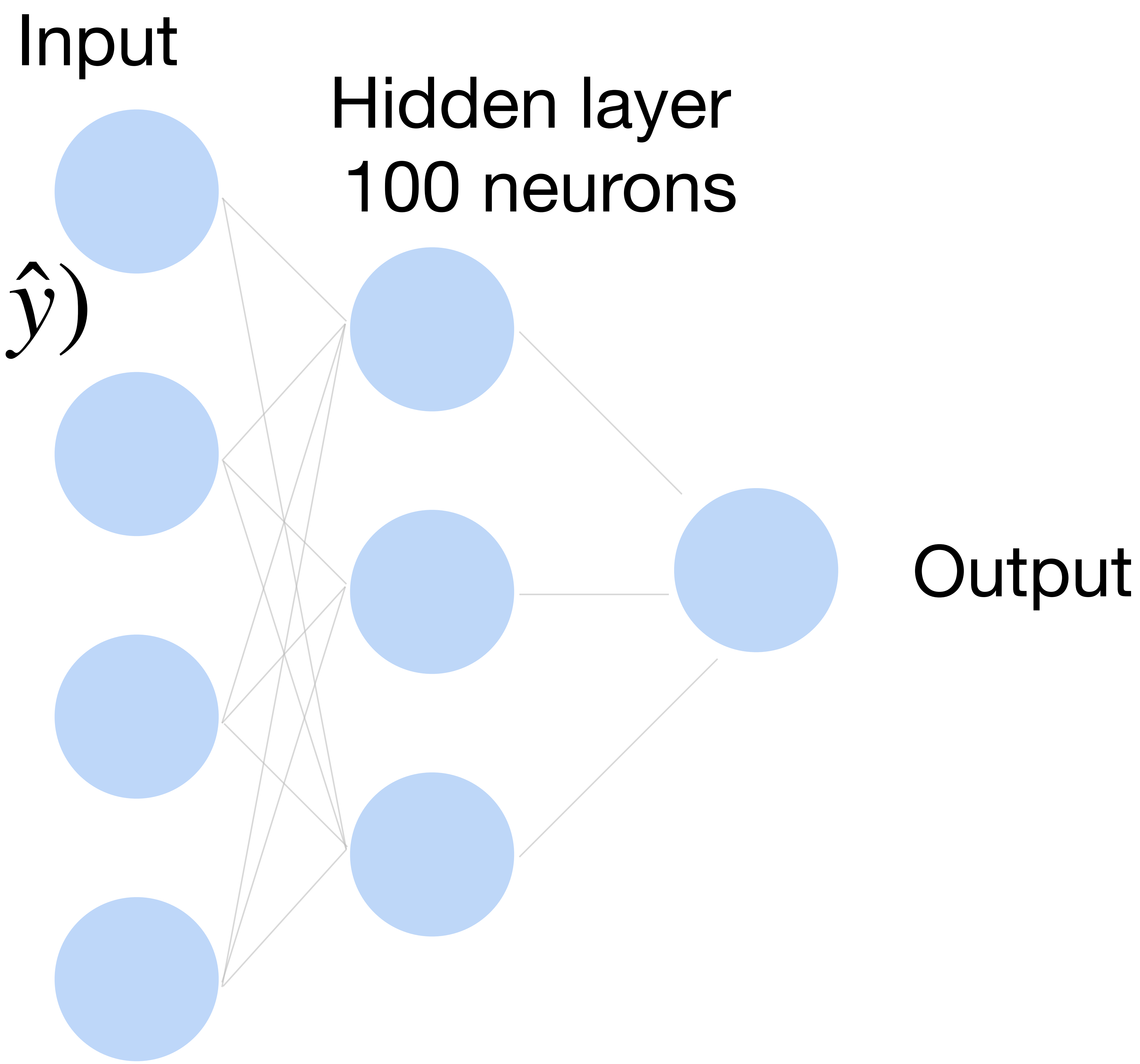
Per-sample loss:

$$\ell(\mathbf{x}, y) = -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$$

Negative log likelihood

Minimizing NLL is equivalent to Max Likelihood Learning (MLE)

Also known as binary cross-entropy



How to train a neural network? Multiclass

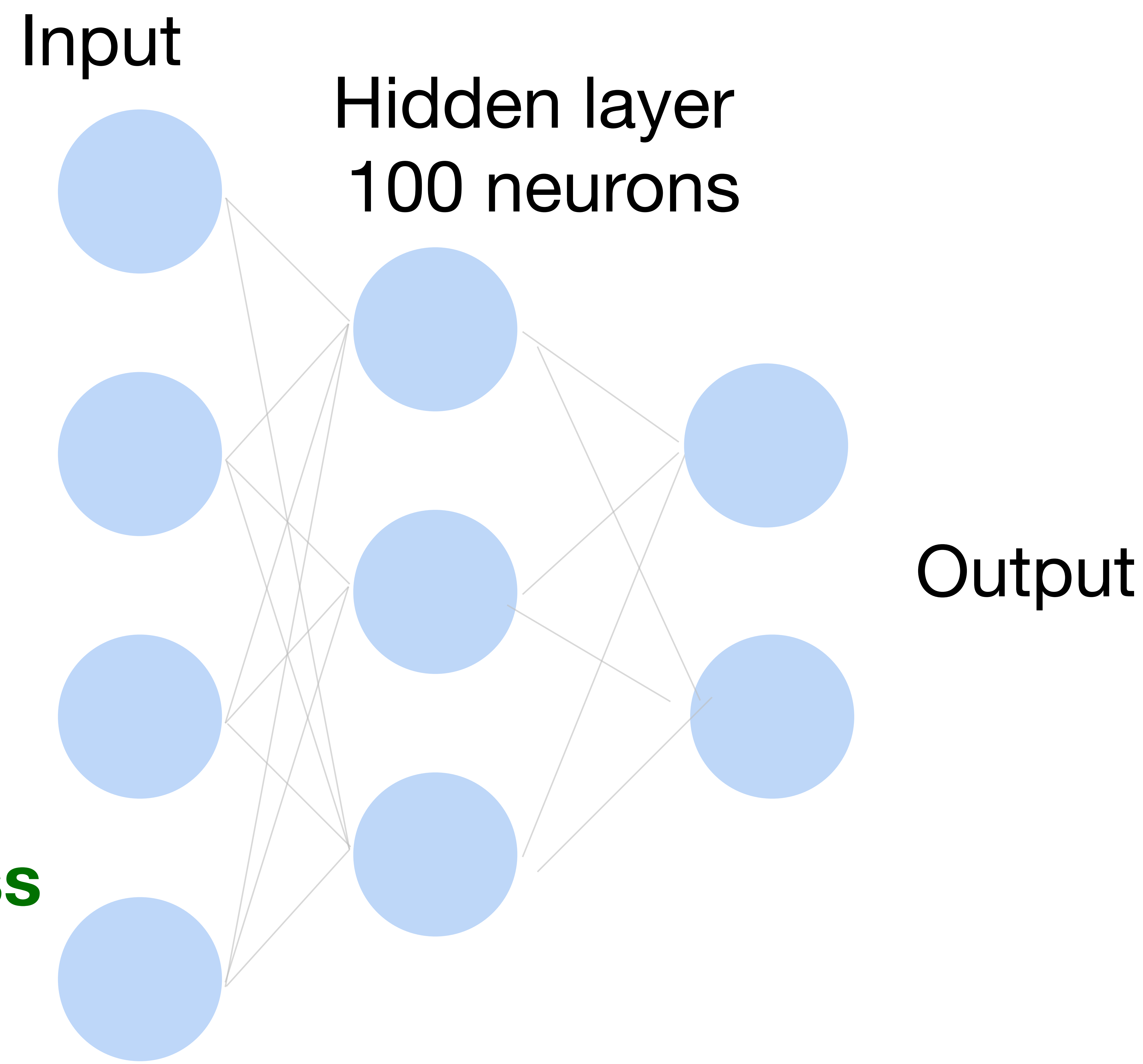
Loss function: $$\frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$$

Per-sample loss:

$$\ell(\mathbf{x}, y) = \sum_{k=1}^{K} -Y_k \log p_k = -\log p_y$$

where Y is one-hot encoding of y

Also known as cross-entropy loss or softmax loss

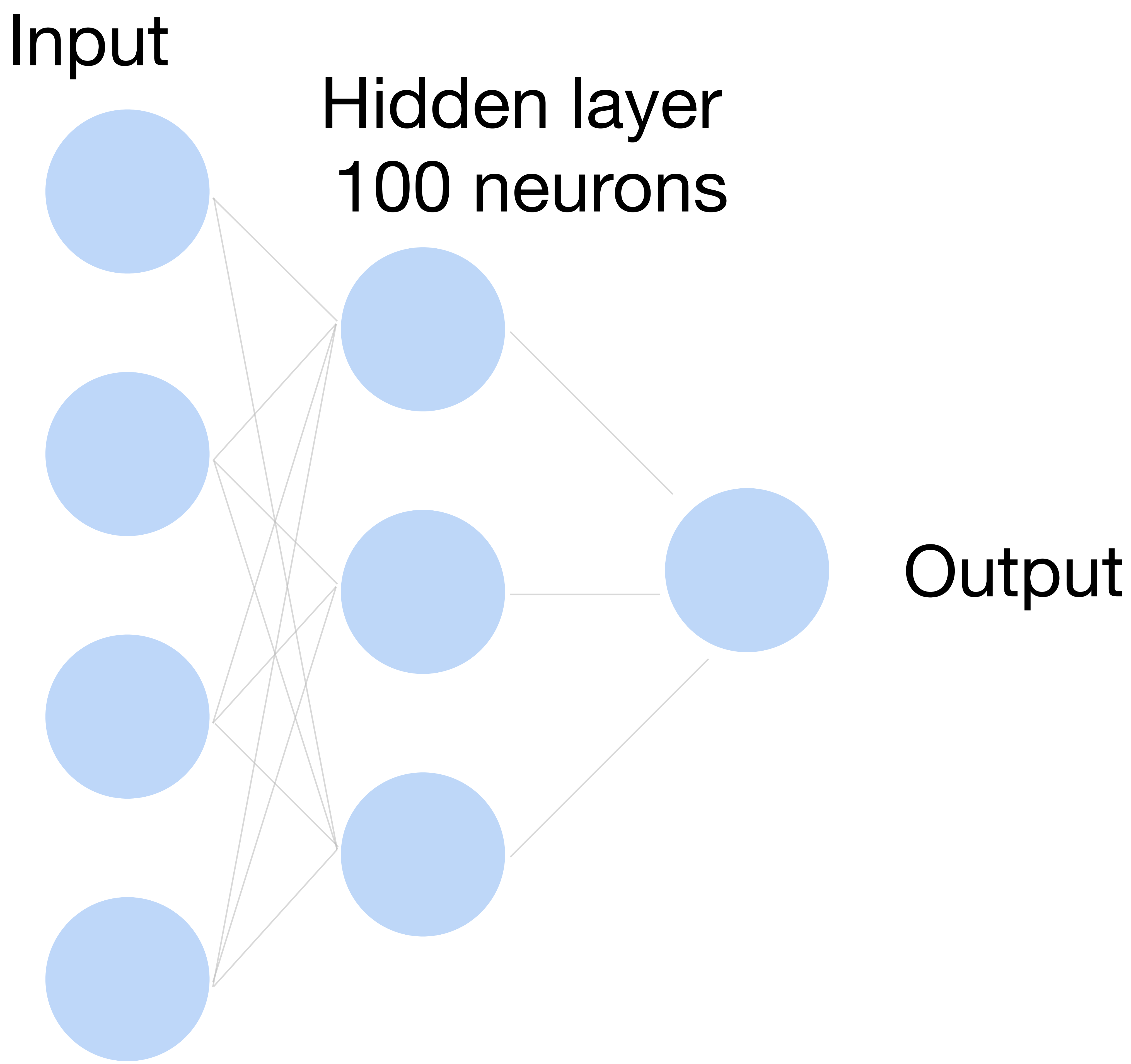


How to train a neural network?

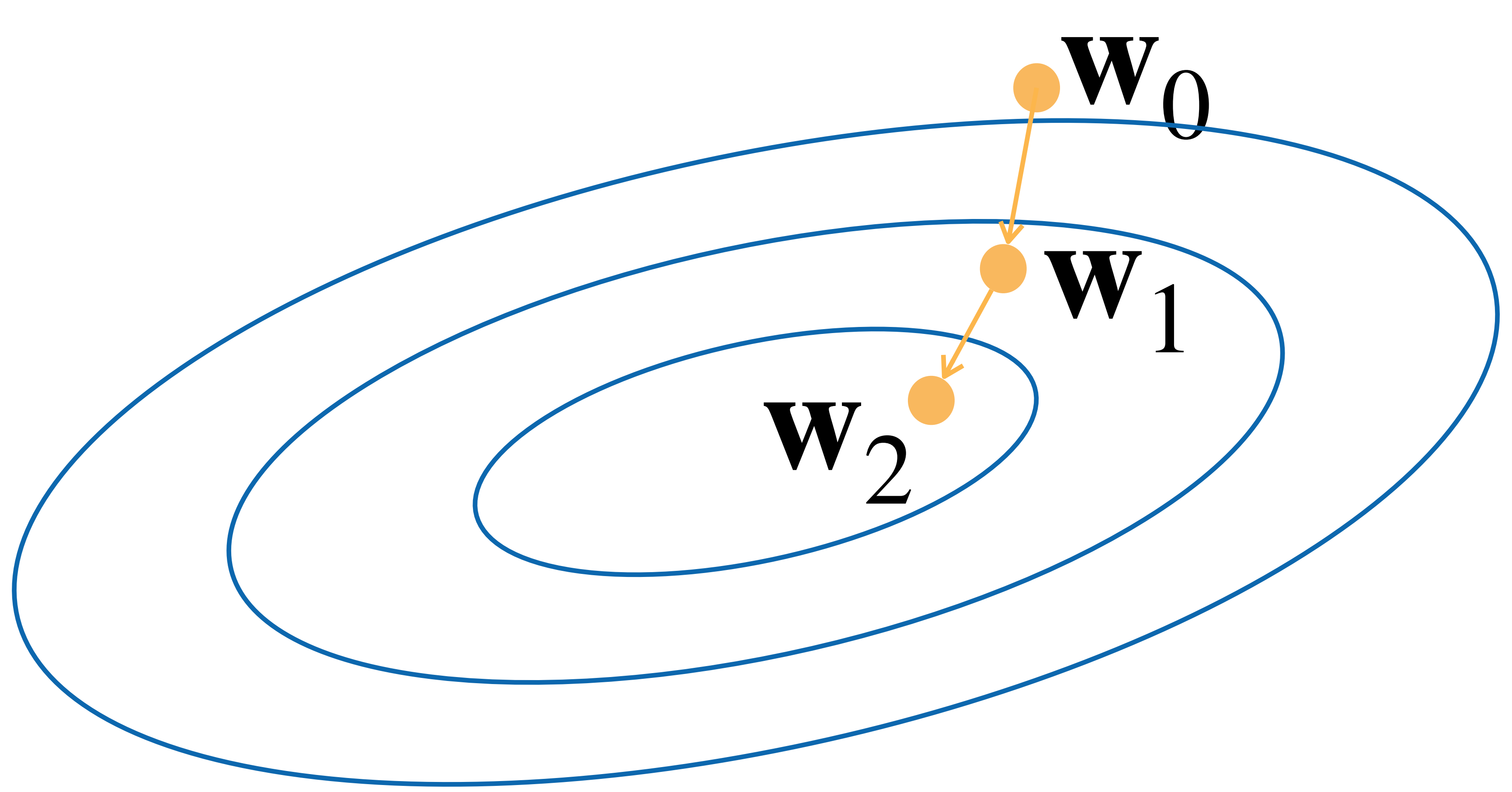
Update the weights W to minimize the loss function

$$L = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$$

Use gradient descent!



Gradient Descent



- Choose a learning rate $\alpha > 0$
- Initialize the model parameters w_0
- For t =1,2,…
  - Update parameters:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \frac{\partial L}{\partial \mathbf{w}_{t-1}}$$

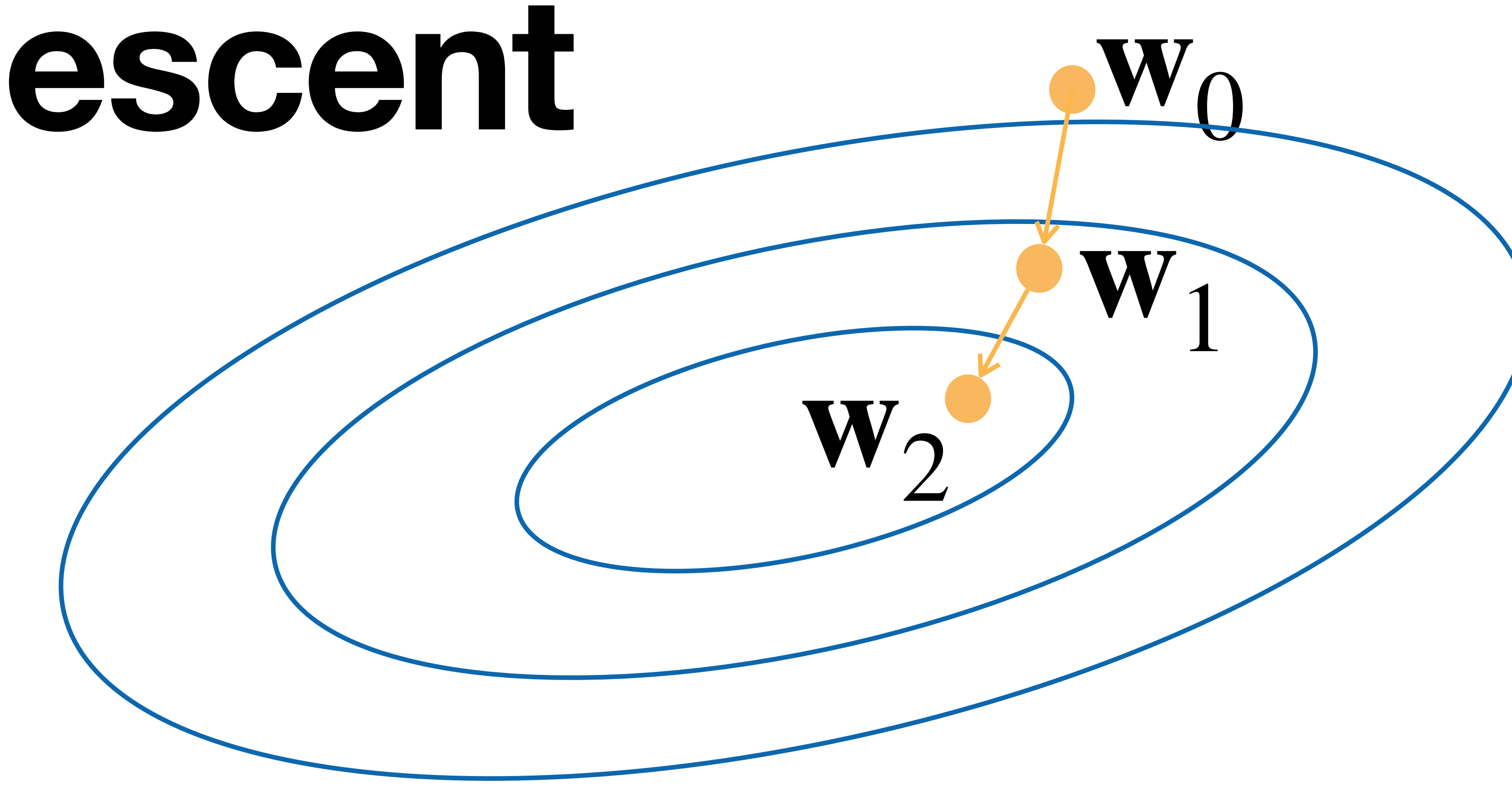
$$= \mathbf{w}_{t-1} - \alpha \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \frac{\partial \ell(\mathbf{x}, y)}{\partial \mathbf{w}_{t-1}}$$

D can be very large. Expensive

The gradient w.r.t. all parameters is obtained by concatenating the partial derivatives w.r.t. each parameter

- Repeat until converges

Minibatch Stochastic Gradient Descent

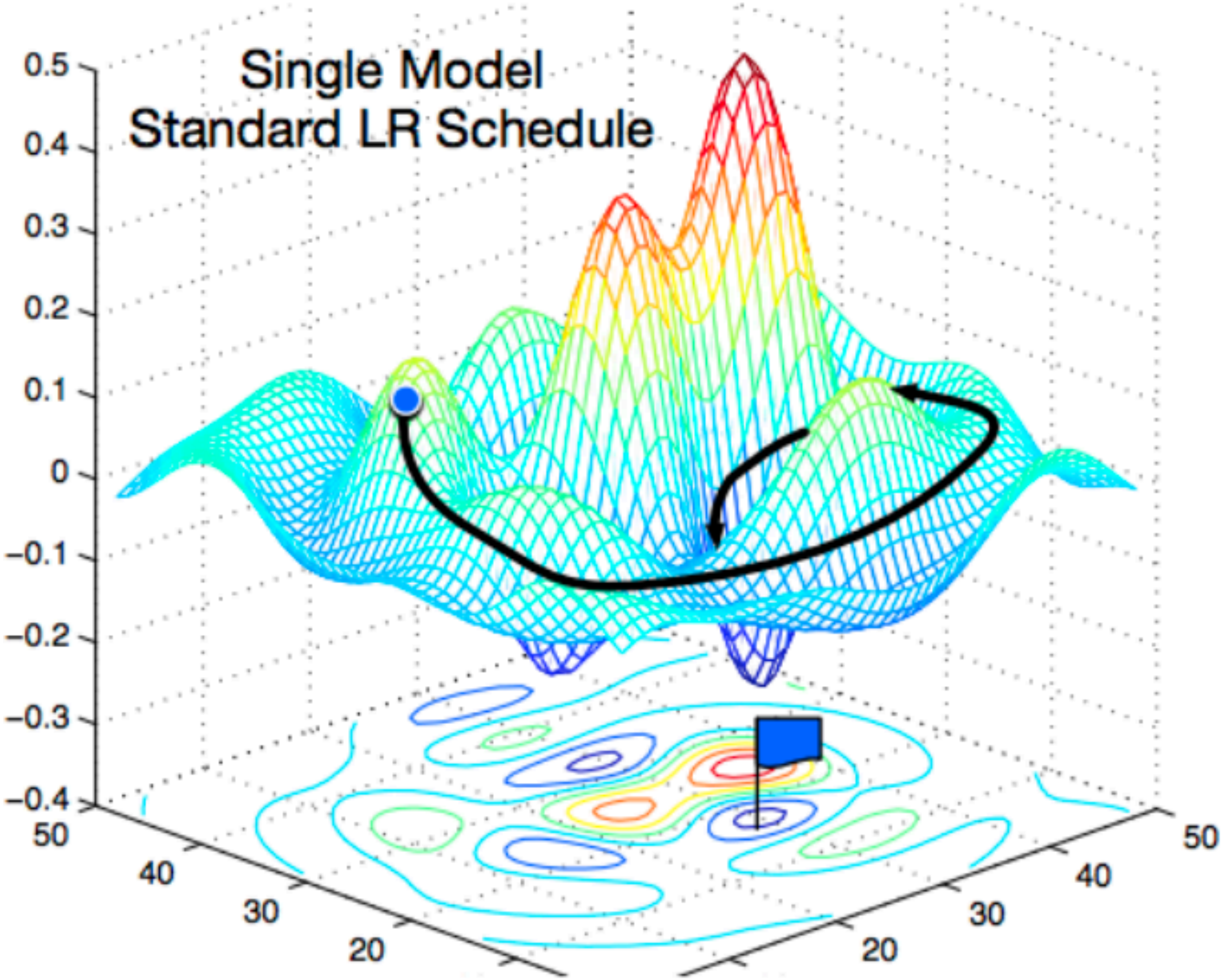


- Choose a learning rate $\alpha > 0$
- Initialize the model parameters w_0
- For t =1,2,…
    - **Randomly sample a subset (mini-batch)** $B \subset D$
      Update parameters:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \frac{1}{|B|} \sum_{(\mathbf{x},y)\in B} \frac{\partial \ell(\mathbf{x}, y)}{\partial \mathbf{w}_{t-1}}$$

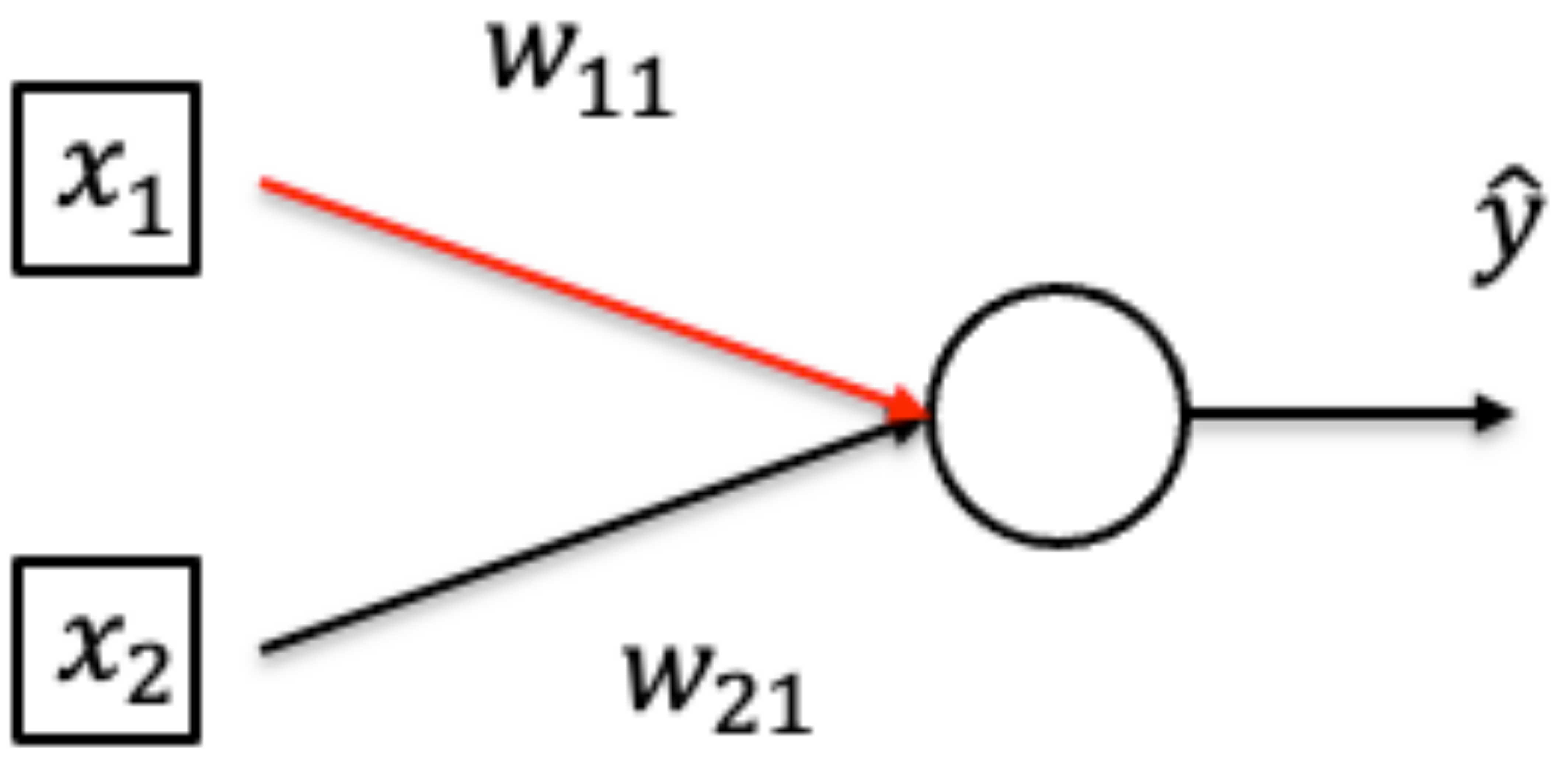
- Repeat until converges

Non-convex Optimization



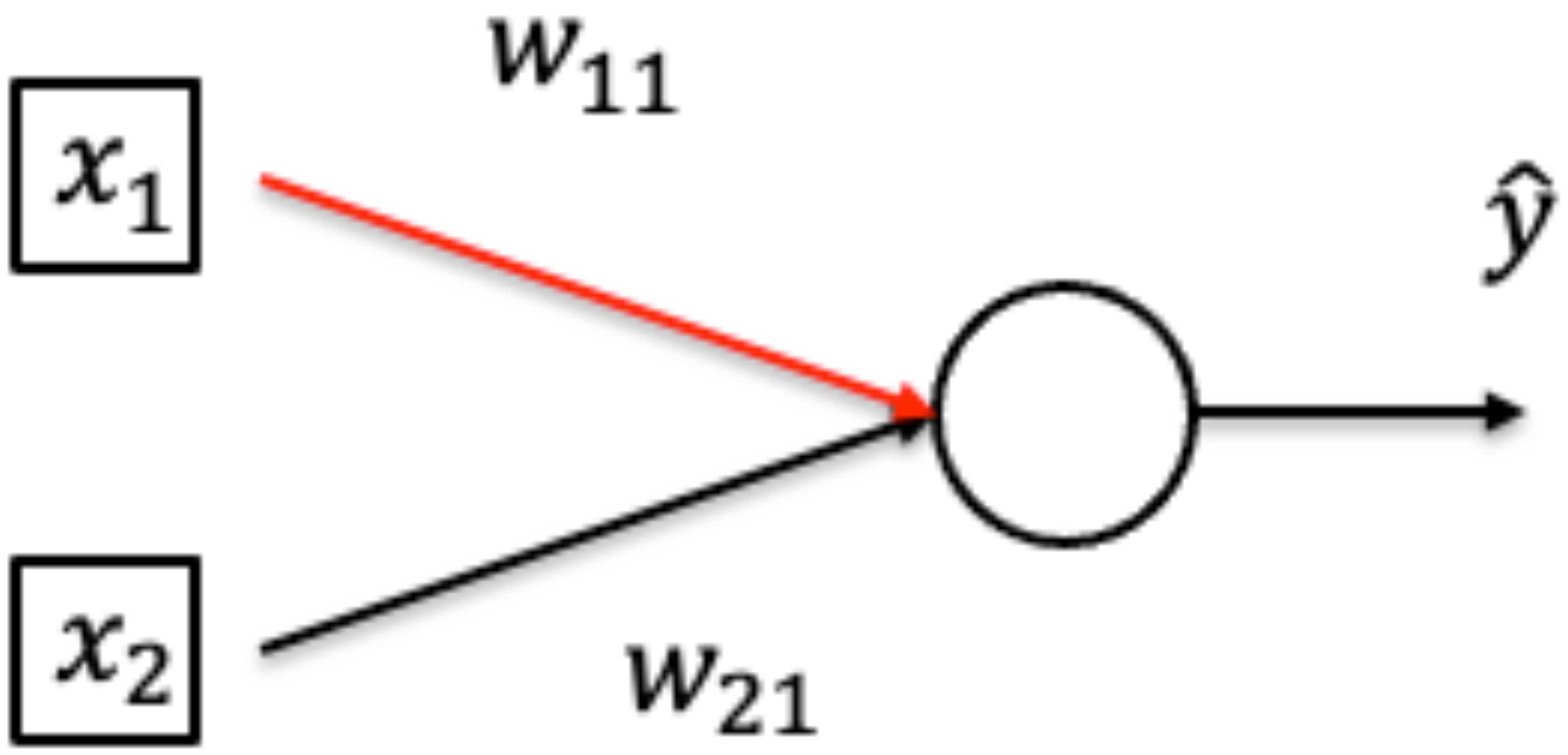
[Gao and Li et al., 2018]

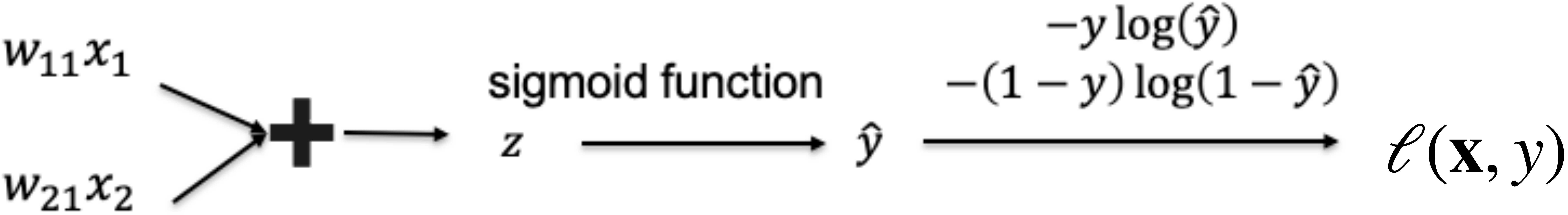
Calculate Gradient (on one data point)



- Want to compute $\dfrac{\partial \ell(\mathbf{x}, y)}{\partial w_{11}}$
- Data point: $((x_1, x_2), y)$

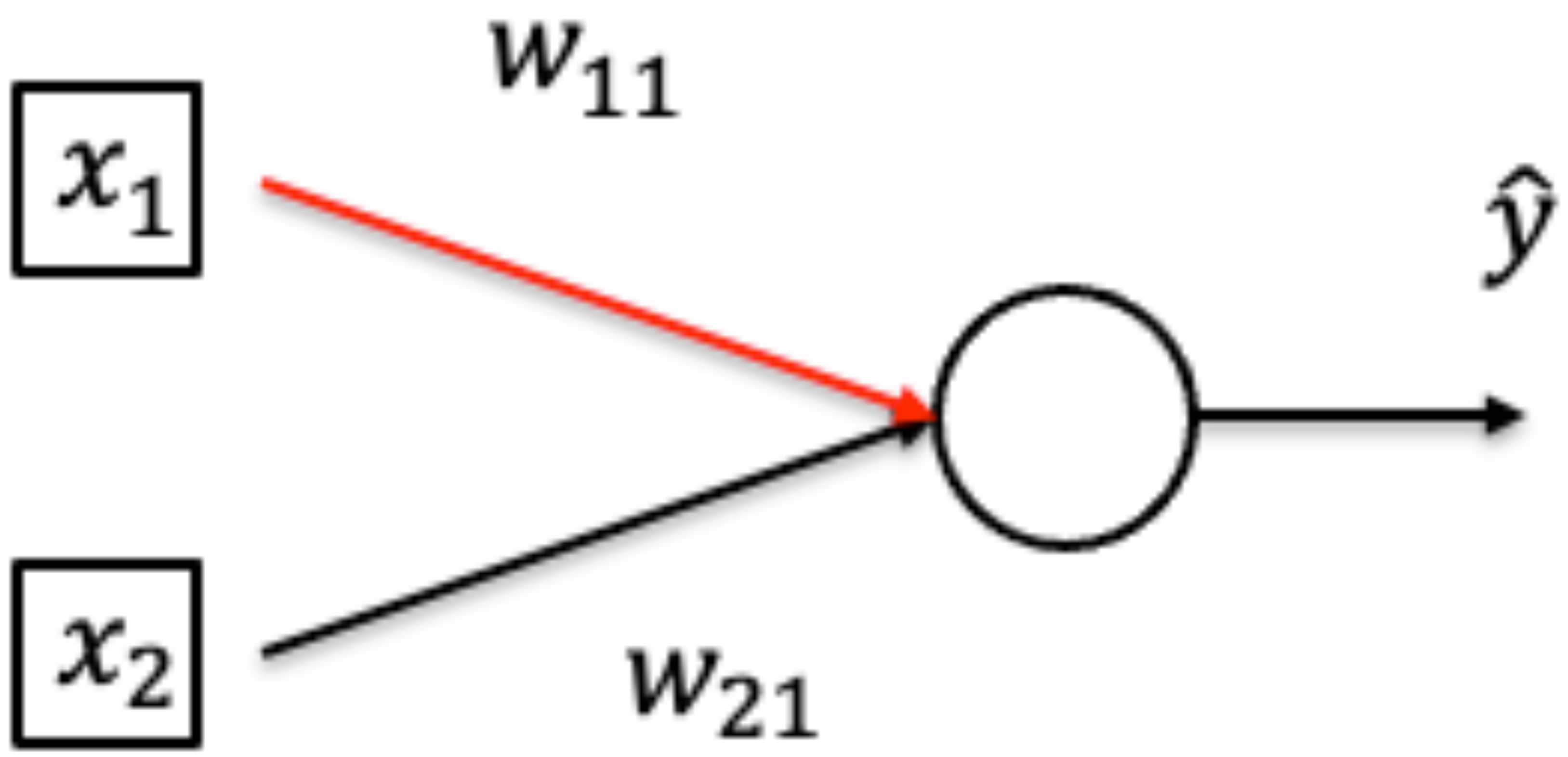
Calculate Gradient (on one data point)

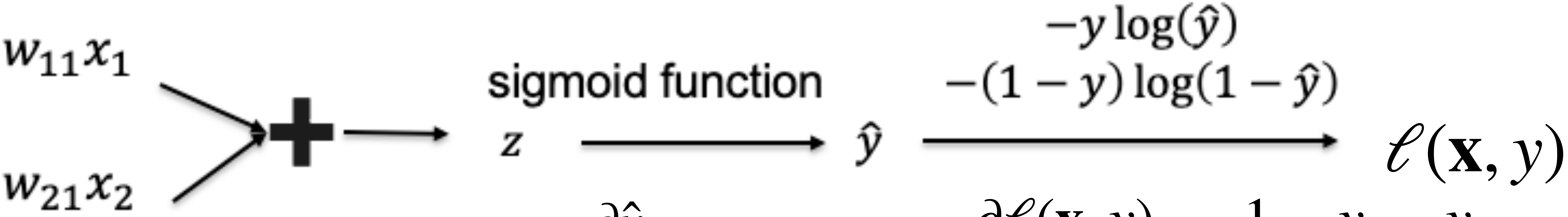




Use chain rule!

Calculate Gradient (on one data point)





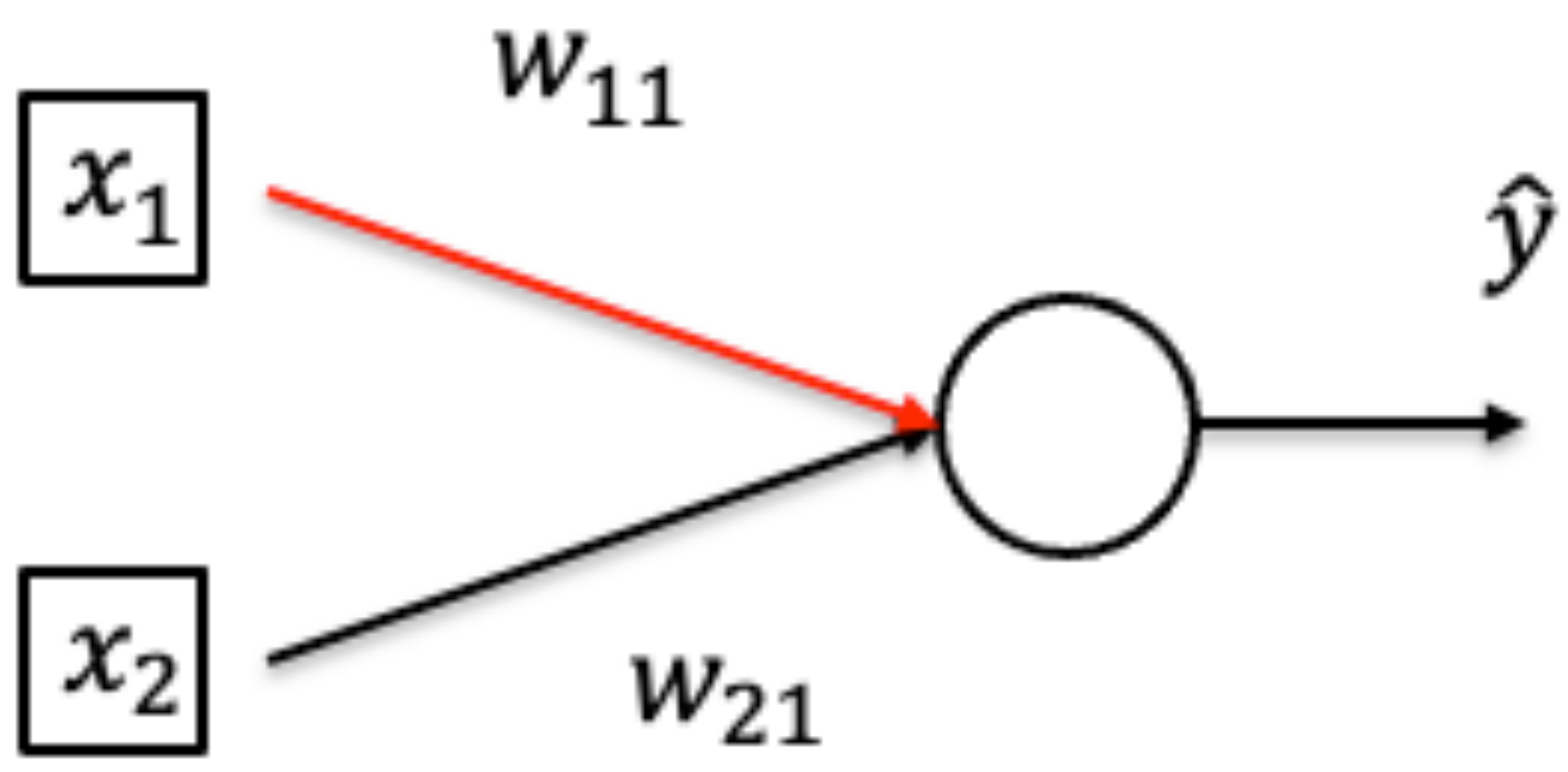
$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z)$$

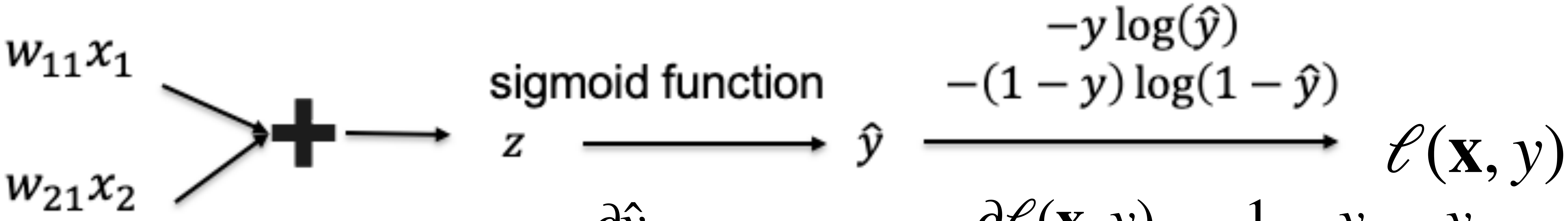
$$\frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- By chain rule:

$$\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{11}}$$

Calculate Gradient (on one data point)





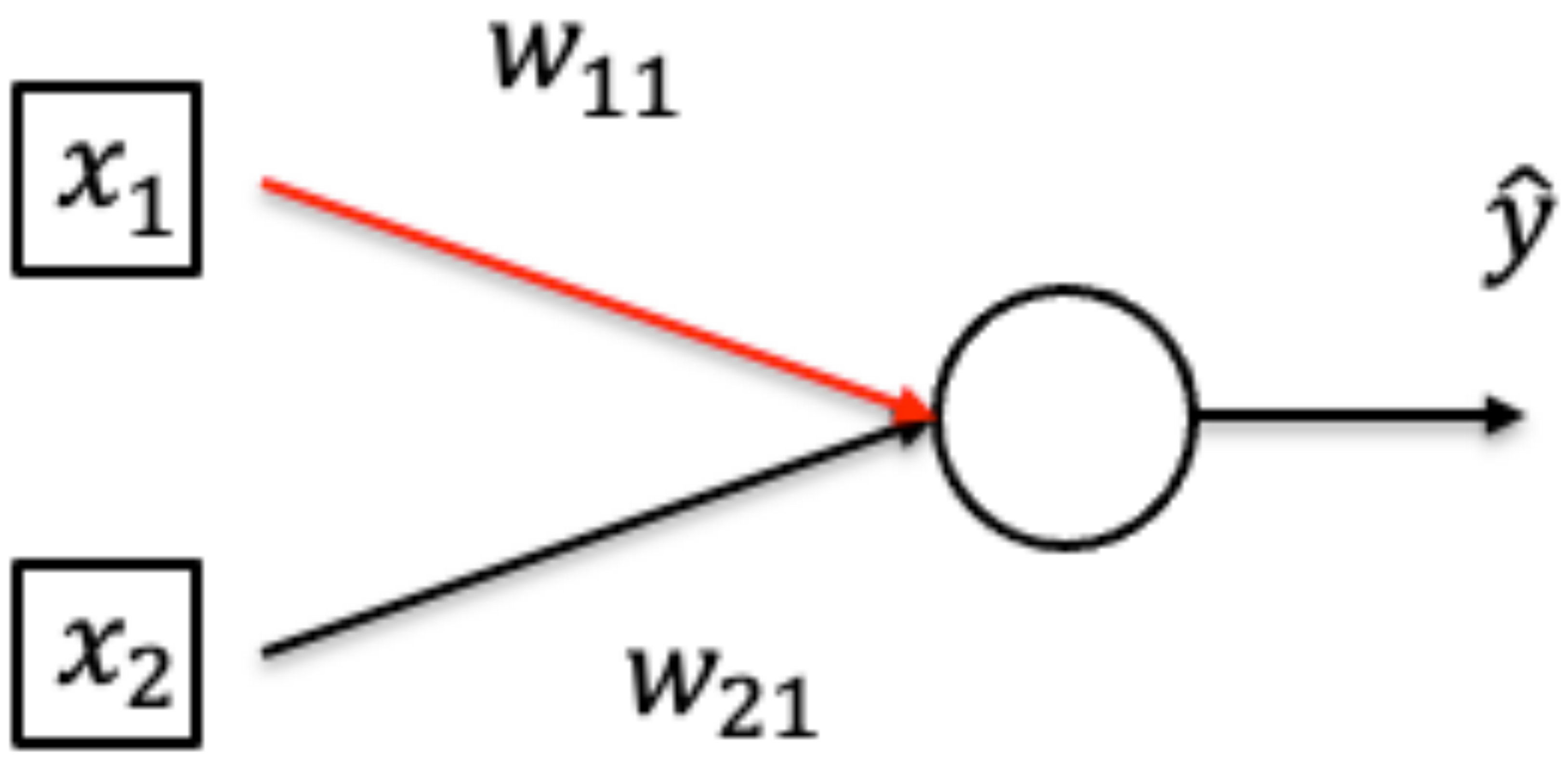
$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z)$$

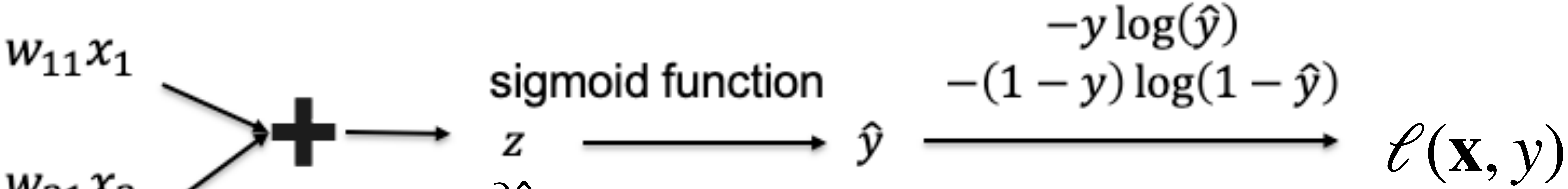
$$\frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- By chain rule:

$$\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} x_1$$

Calculate Gradient (on one data point)

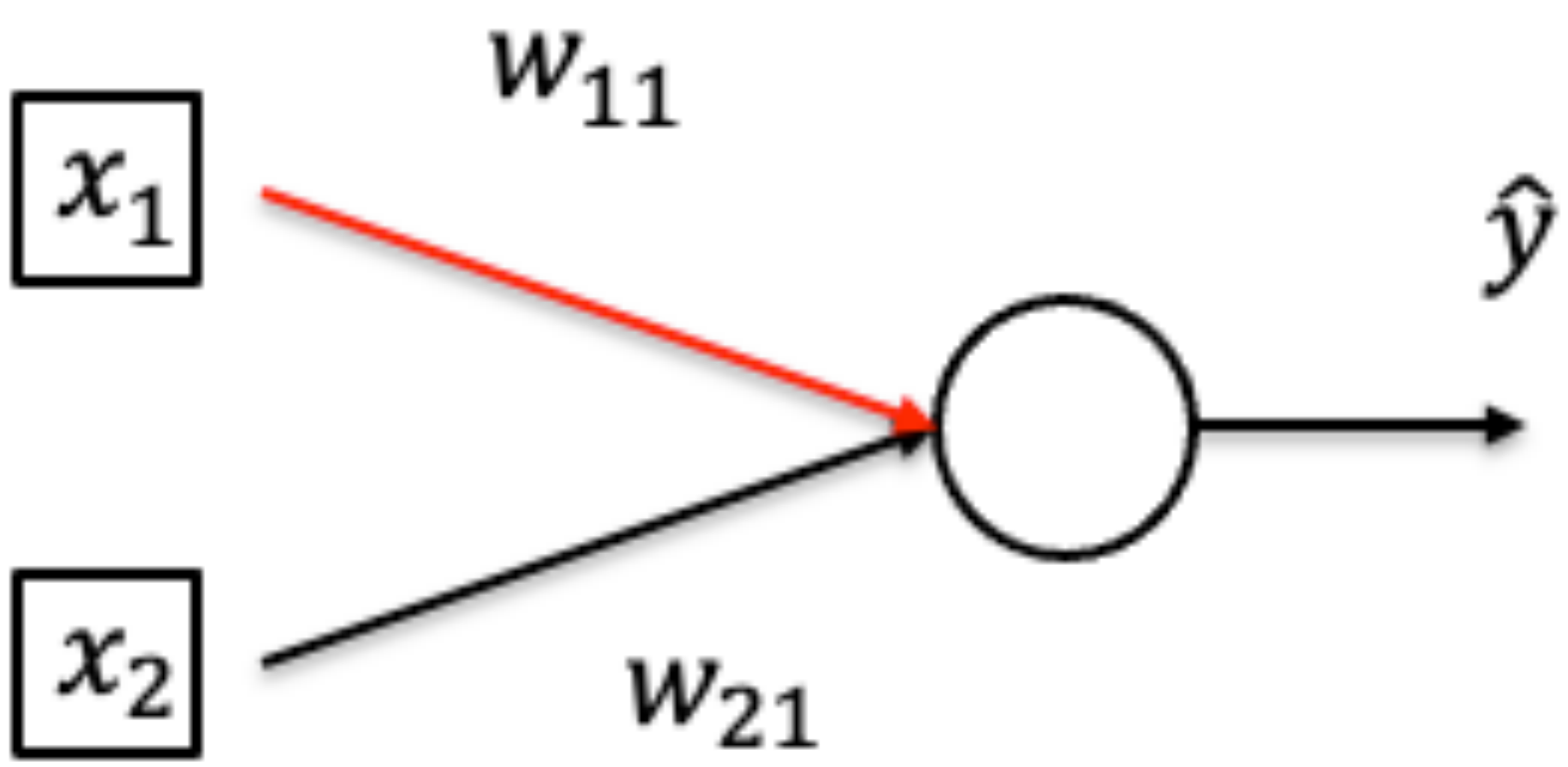


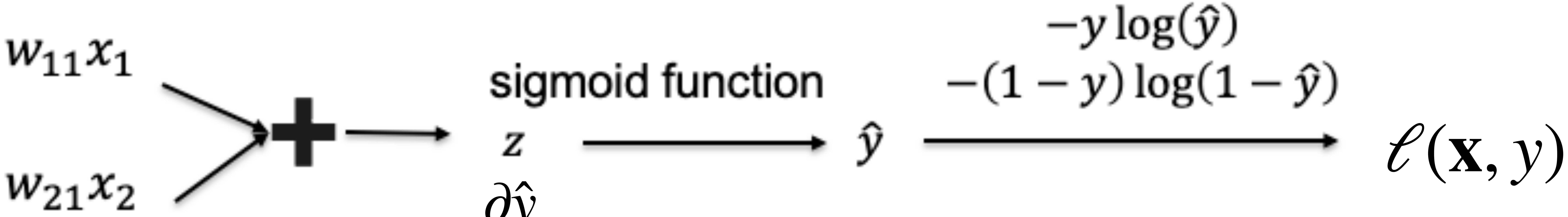


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $$\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \; \hat{y}(1-\hat{y})x_1$$

Calculate Gradient (on one data point)

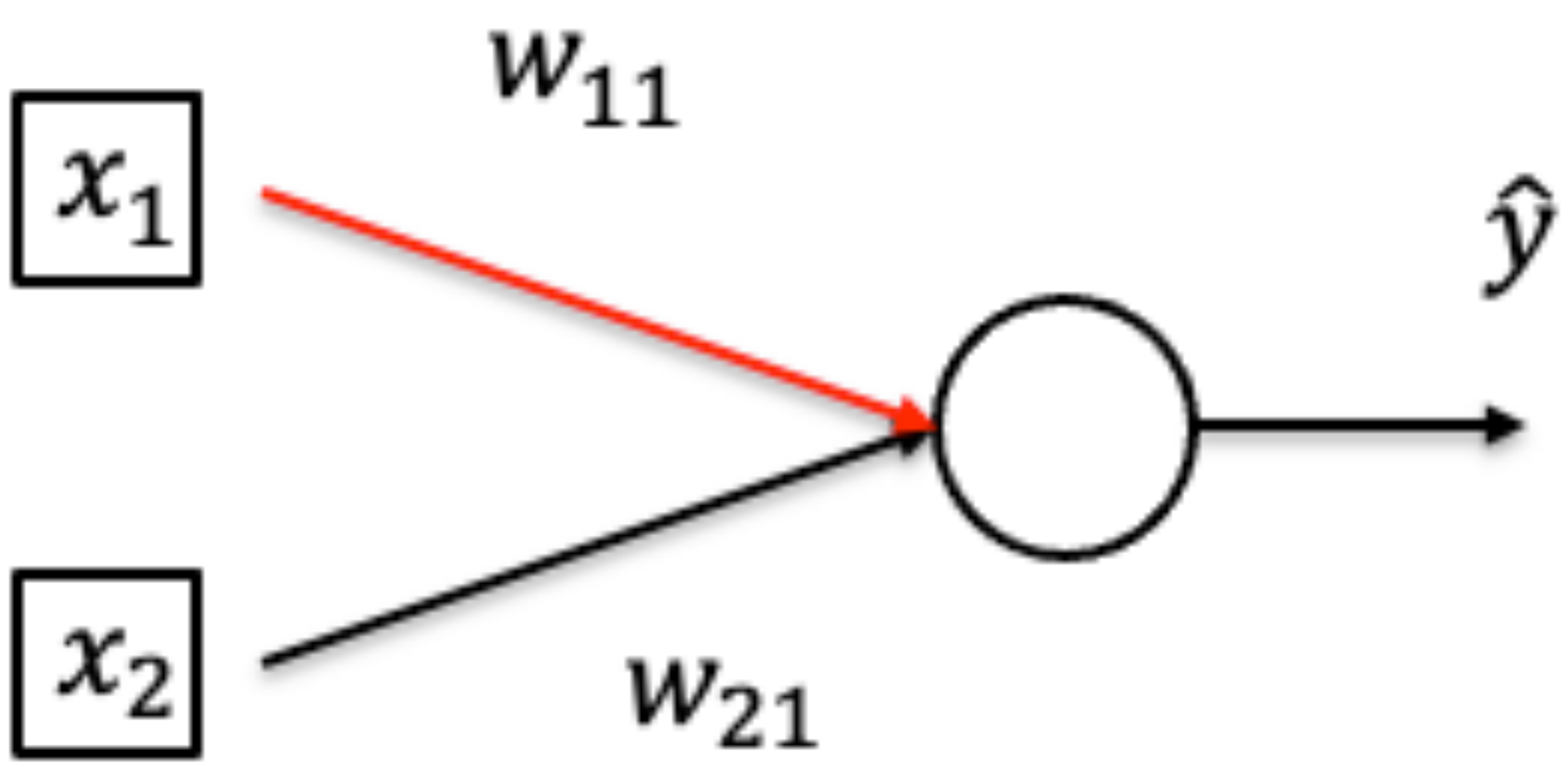


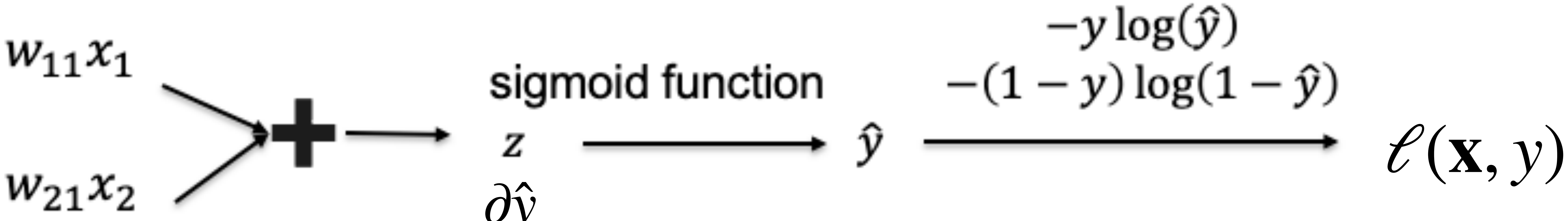


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $$\frac{\partial l}{\partial w_{11}} = \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\hat{y}(1-\hat{y})x_1$$

Calculate Gradient (on one data point)

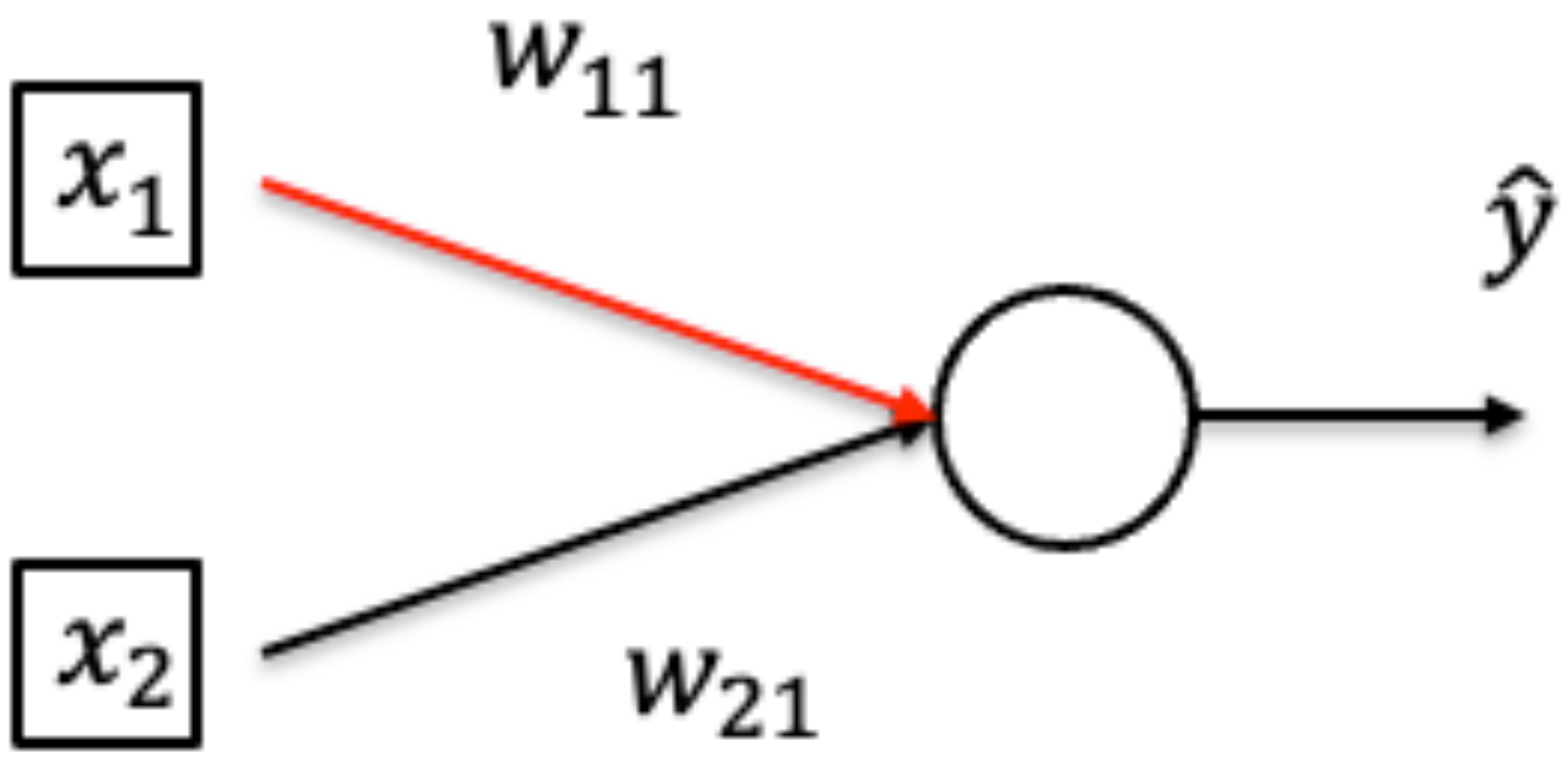


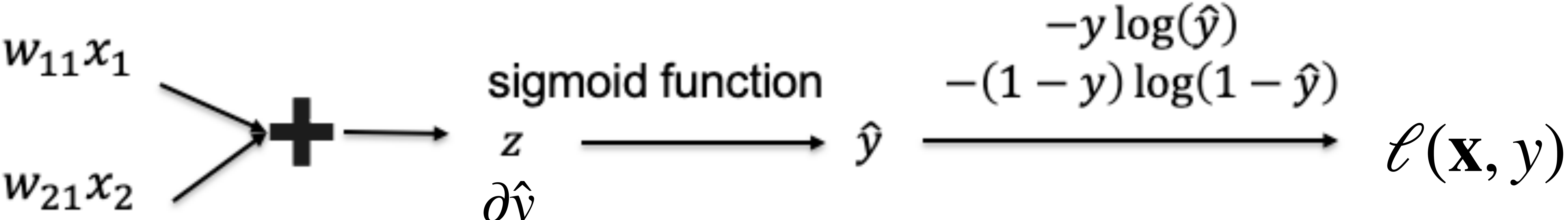


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $$\frac{\partial l}{\partial w_{11}} = (\hat{y} - y)x_1$$

Calculate Gradient (on one data point)

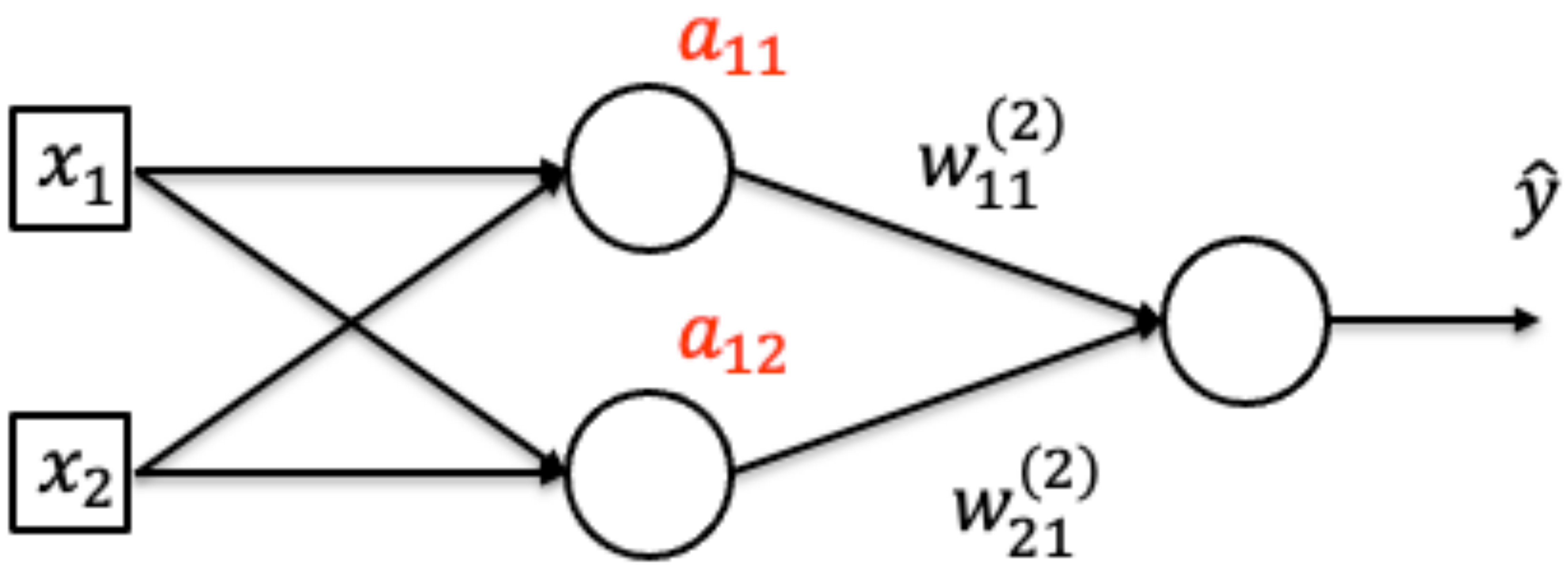




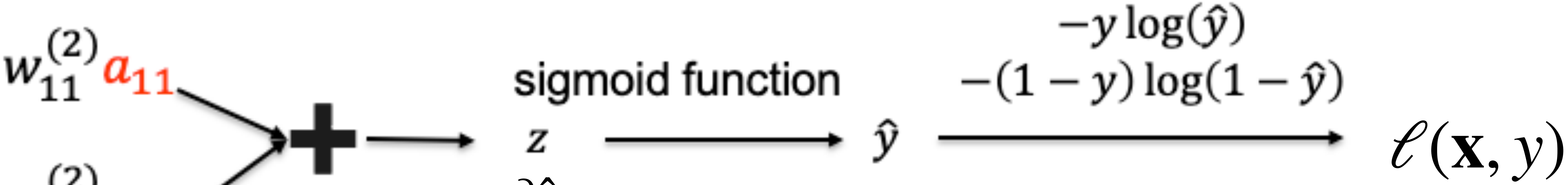
$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1 - \sigma(z))$$

- By chain rule: $$\frac{\partial l}{\partial x_1} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} w_{11} = (\hat{y} - y) w_{11}$$

Calculate Gradient (on one data point)



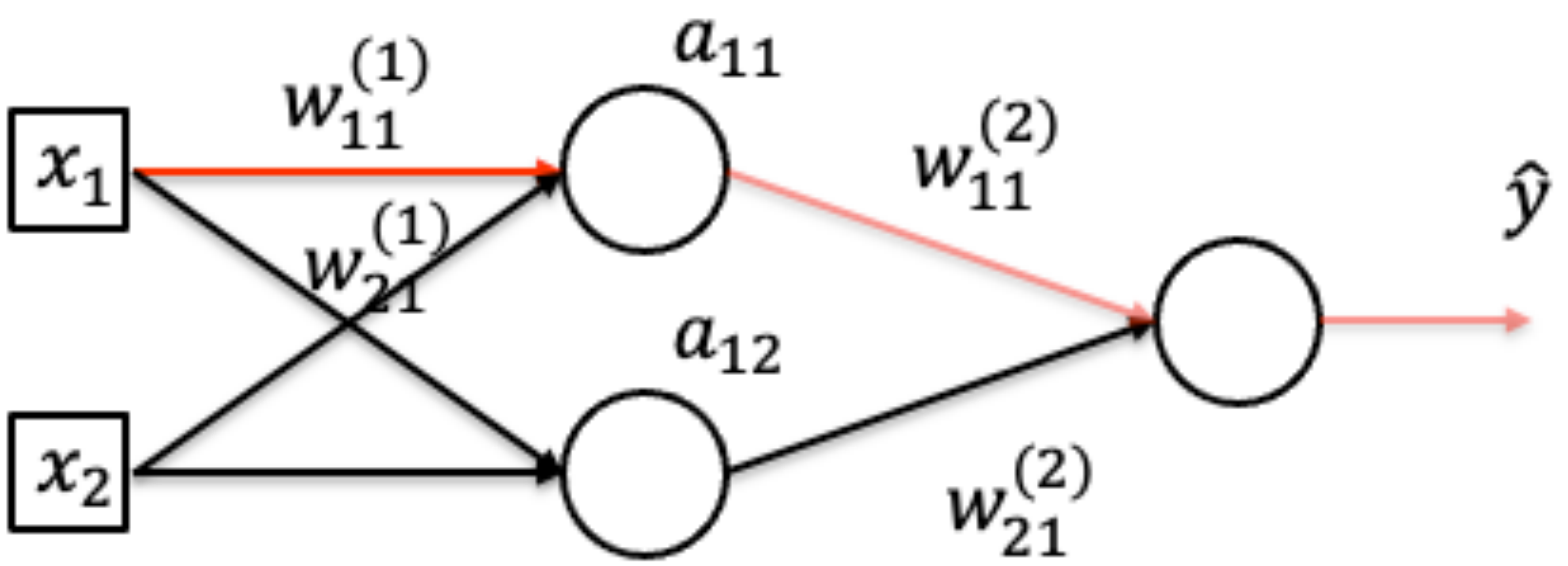
Make it deeper

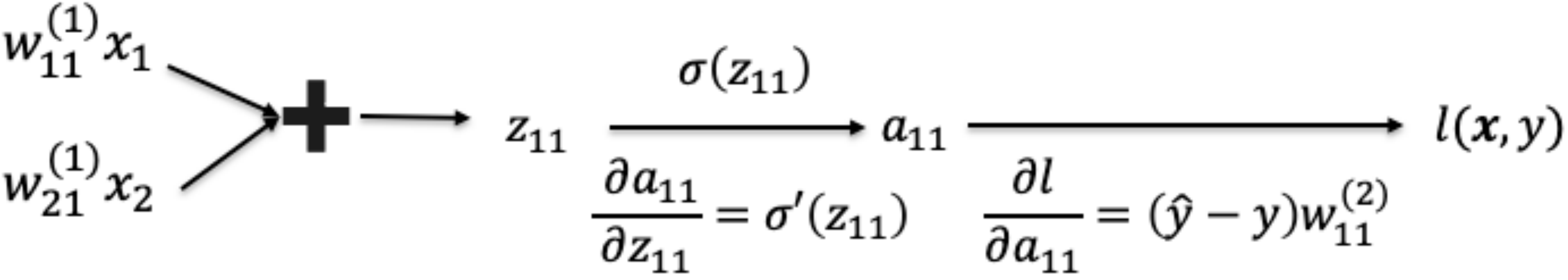


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1 - \sigma(z))$$

- By chain rule: $\frac{\partial l}{\partial a_{11}} = (\hat{y} - y) w_{11}^{(2)}, \ \frac{\partial l}{\partial a_{12}} = (\hat{y} - y) w_{21}^{(2)}$

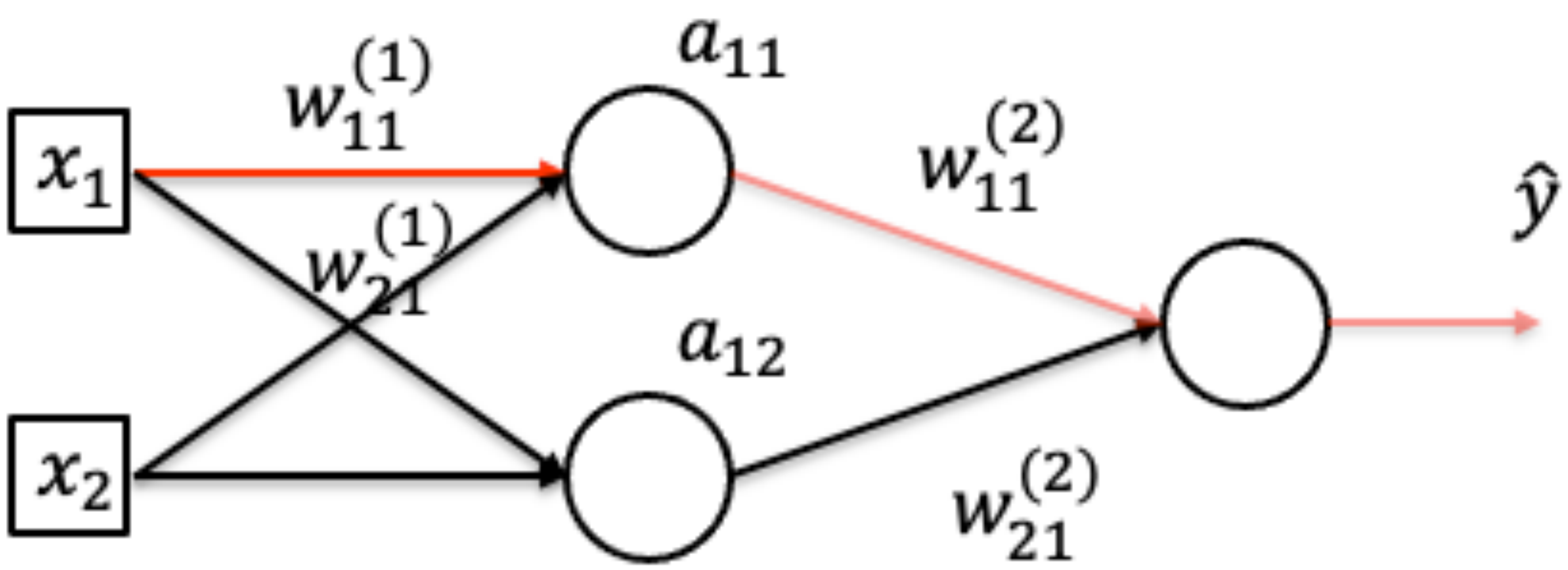
Calculate Gradient (on one data point)

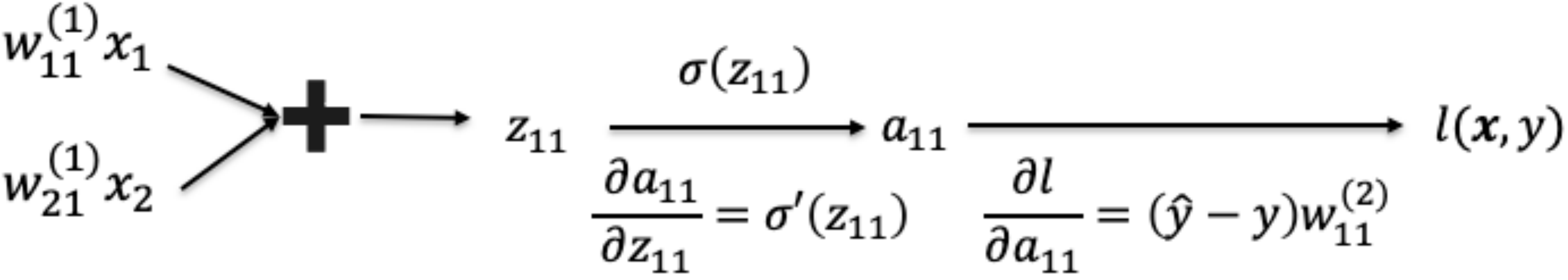




- By chain rule: $\frac{\partial l}{\partial w_{11}^{(1)}} = \frac{\partial l}{\partial a_{11}} \frac{\partial a_{11}}{\partial w_{11}^{(1)}} = (\hat{y} - y) w_{11}^{(2)} \frac{\partial a_{11}}{\partial w_{11}^{(1)}}$

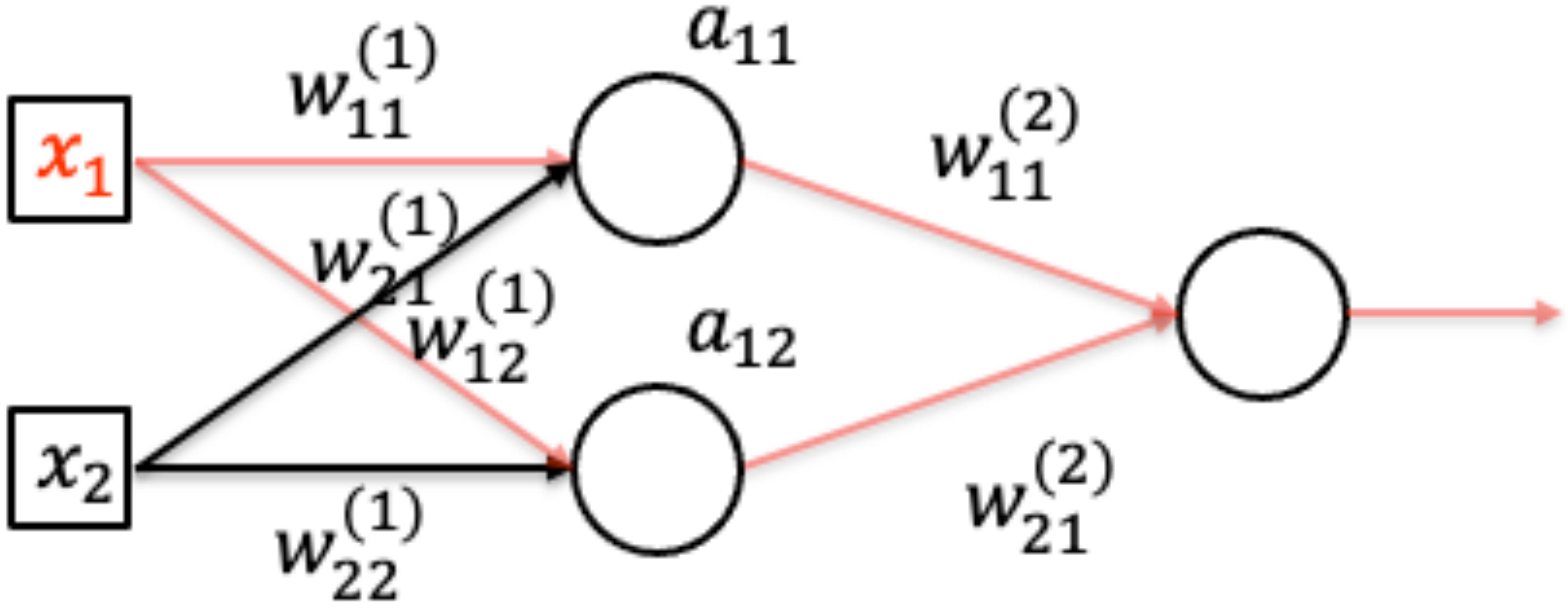
Calculate Gradient (on one data point)

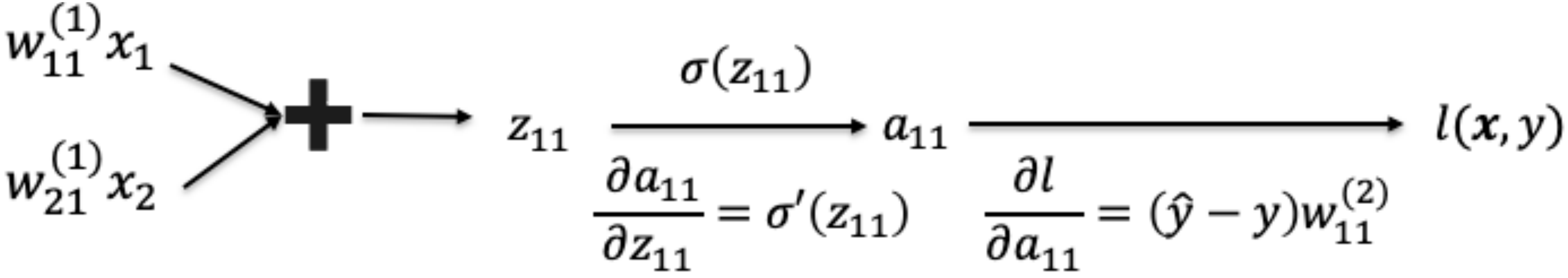




- By chain rule: $\frac{\partial l}{\partial w_{11}^{(1)}} = \frac{\partial l}{\partial a_{11}} \frac{\partial a_{11}}{\partial w_{11}^{(1)}} = (\hat{y} - y) w_{11}^{(2)} a_{11}(1 - a_{11}) x_1$

Calculate Gradient (on one data point)





- By chain rule: $$\frac{\partial l}{\partial x_1} = \frac{\partial l}{\partial a_{11}}\frac{\partial a_{11}}{\partial x_1} + \frac{\partial l}{\partial a_{12}}\frac{\partial a_{12}}{\partial x_1}$$

Quiz Break

Gradient Descent in neural network training computes the ______ of a loss function with respect to the model ______ until convergence.

A gradients, parameters

B parameters, gradients

C loss, parameters

D parameters, loss

Quiz Break

Gradient Descent in neural network training computes the ______ of a loss function with respect to the model ______ until convergence.

A gradients, parameters

B parameters, gradients

C loss, parameters

D parameters, loss

Quiz Break

Suppose you are given a dataset with 1,000,000 images to train with. Which of the following methods is more desirable if training resources are limit but enough accuracy is needed?

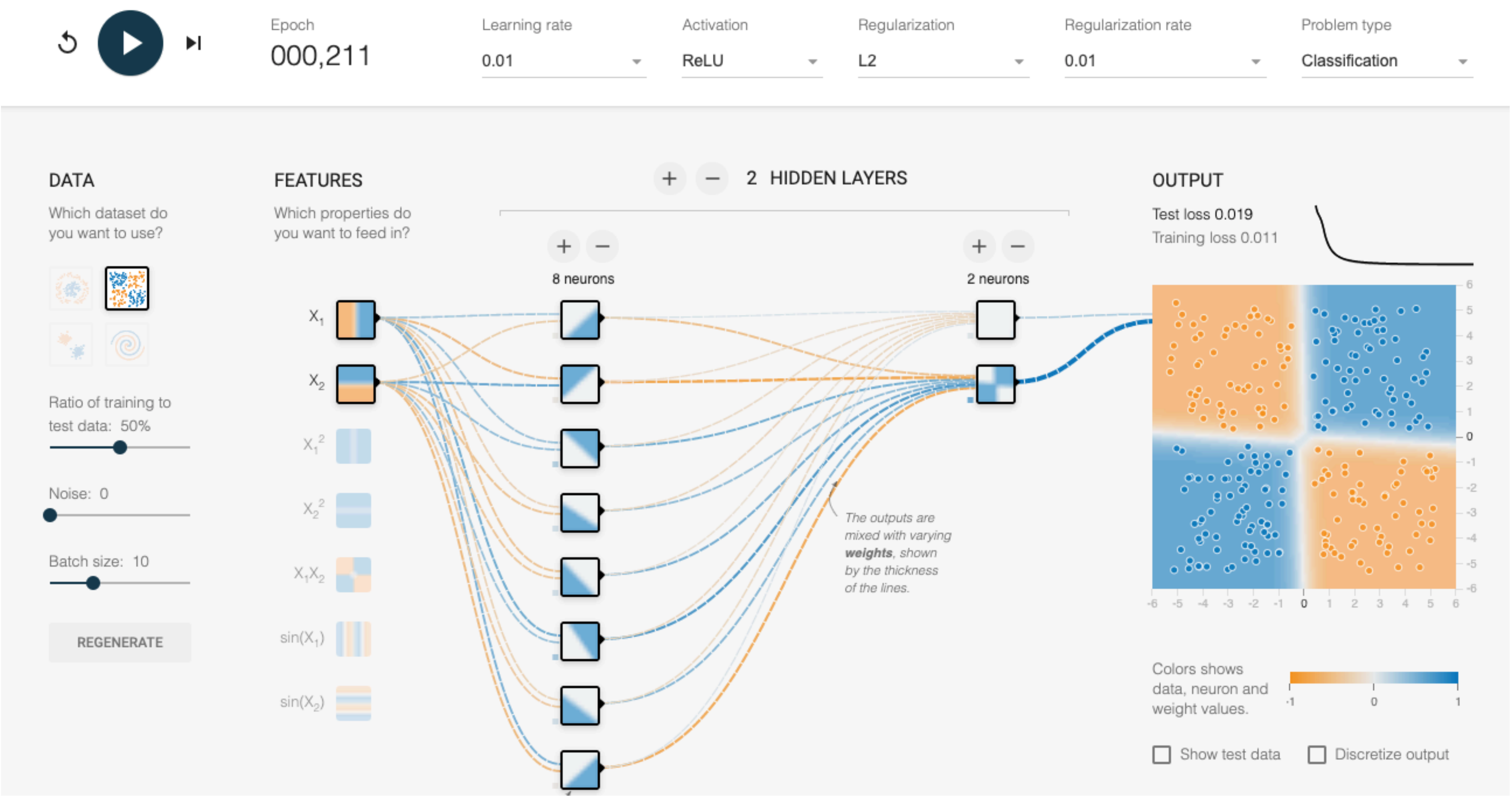
A Gradient Descent

B Stochastic Gradient Descent

C Minibatch Stochastic Gradient Descent

D Computation Graph

Demo: Learning XOR using neural net



• https://playground.tensorflow.org/

What we've learned today…

- Calculus Review
- Multi-layer Perceptron
  - Single output
  - Multiple output
- How to train neural networks
  - Gradient descent