CS 540 Introduction to Artificial Intelligence
Neural Networks: Gradient Descent

University of Wisconsin–Madison
Fall 2025, Section 3
October 10, 2025

Announcements

- HW4 due Friday 10/6 at 11:59 pm
- HW5 out on Friday (linear regression & gradient descent)
- Midterm exam: Thursday 10/23 from 7:30 pm to 9:00 pm

ML: Unsupervised Learning
ML: Linear Regression
ML: K - Nearest Neighbors & Naive Bayes
ML: Neural Networks I (Perceptron)
ML: Neural Networks II
ML: Neural Networks III

Review: Perceptron

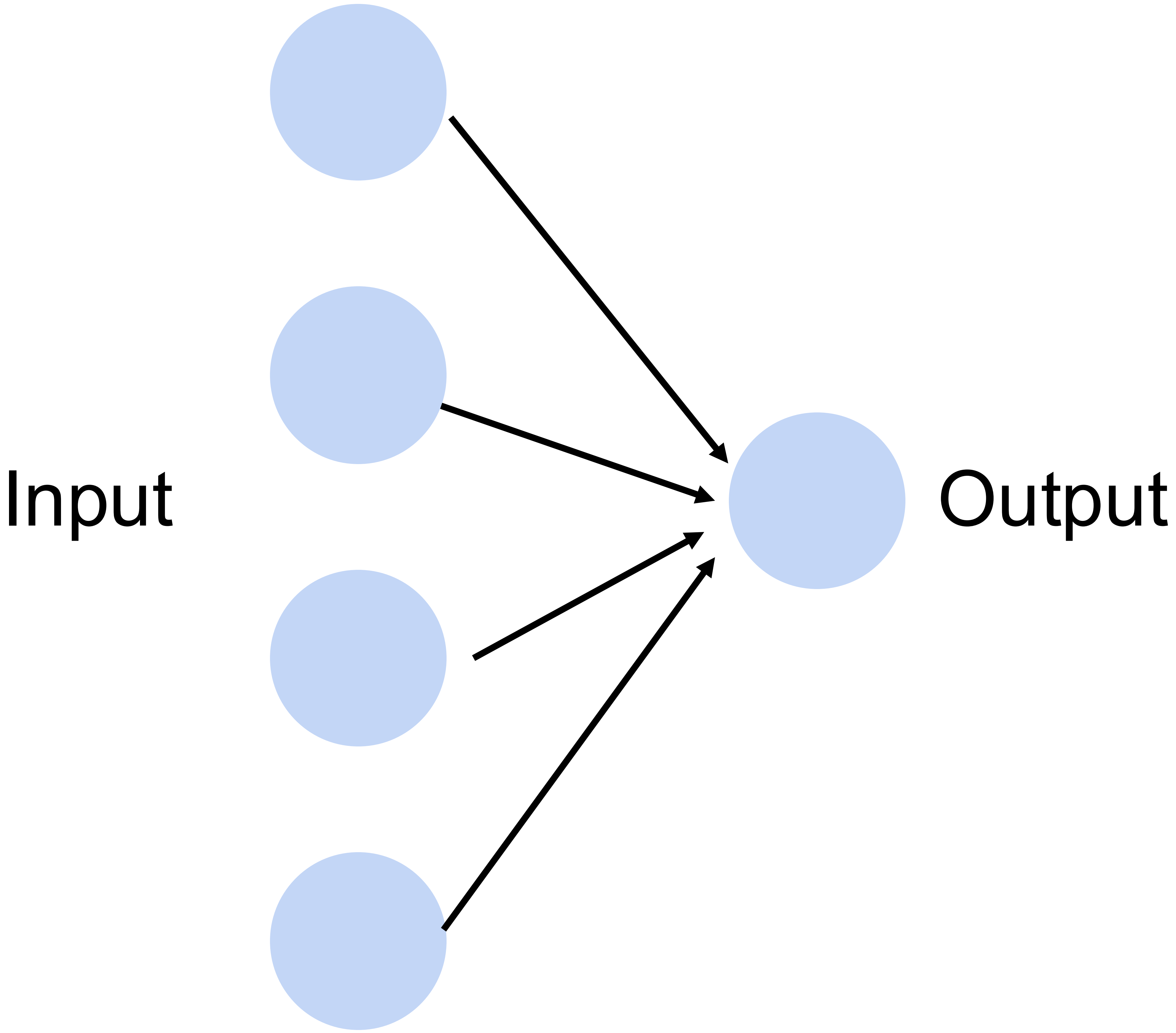
- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b , perceptron outputs:

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

$$\sigma(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & otherwise \end{cases}$$

Activation function

Cats vs. dogs?



Basic Recipe: Perceptron

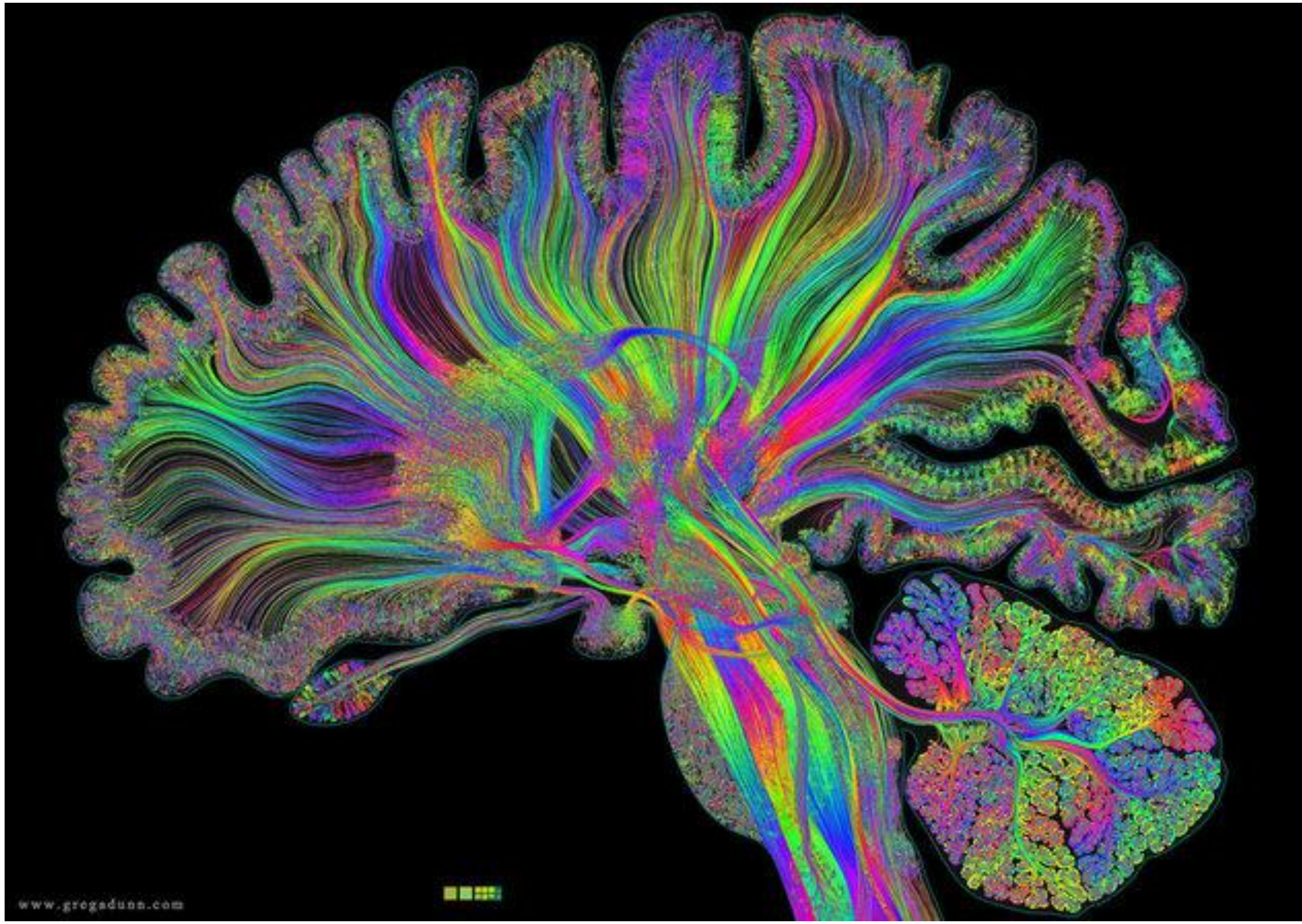
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

1. Linear decision rules

   All functions like: $f_{w,b}(x) = \sigma(\langle w, x \rangle + b)$
2. Misclassification error
3. Perceptron learning rule

   Similar to gradient descent
4. Test/train split

Multilayer Perceptron



Review: Multilayer Perceptron

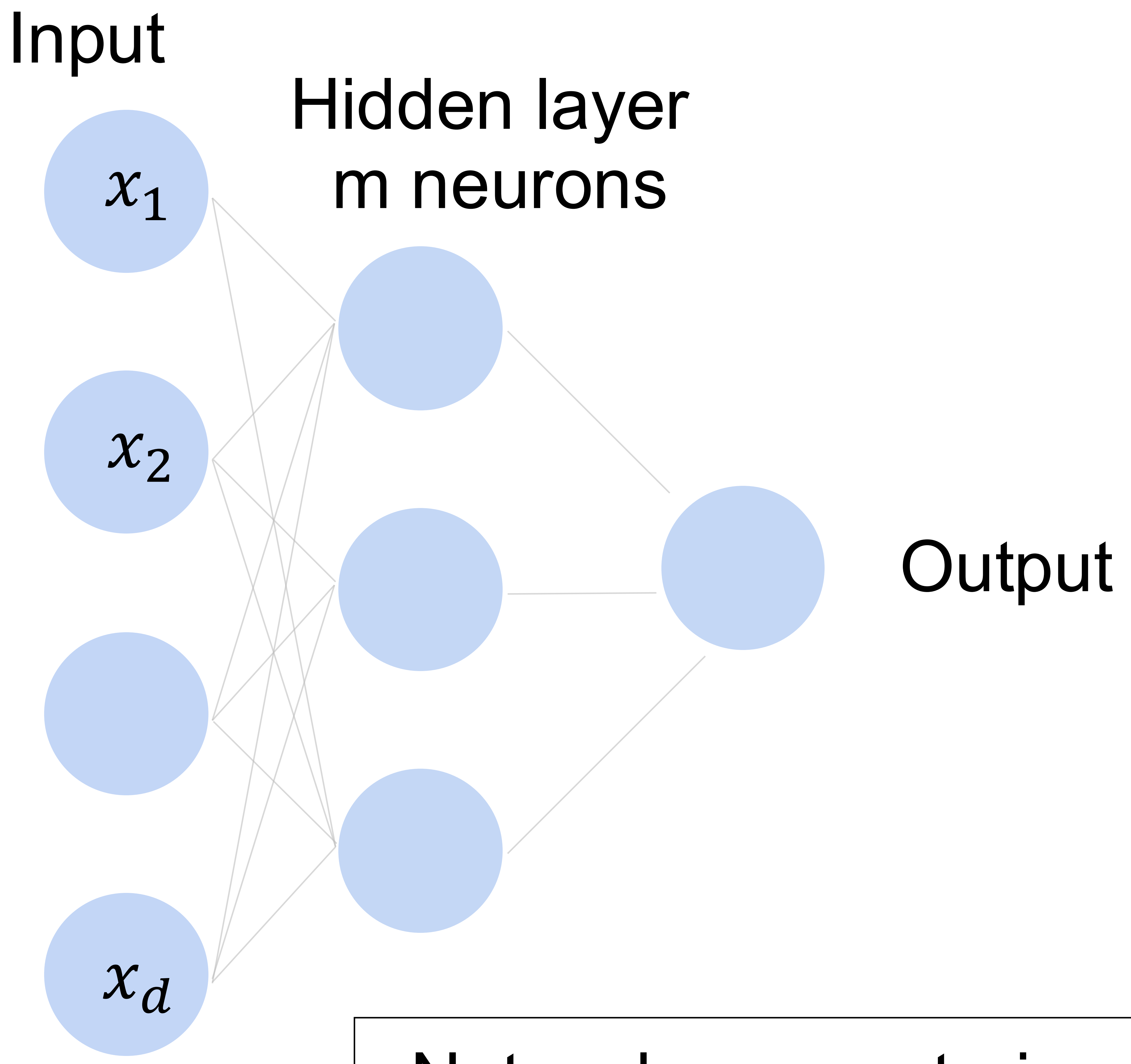
- Input $x \in \mathbb{R}^d$
- Hidden $W_h \in \mathbb{R}^{m \times d}, b_h \in \mathbb{R}^m$
- Intermediate output

$$h = \sigma(W_h x + b_h)$$

σ is an element-wise activation function

- Final output/final perceptron

$$\sigma(\langle w_f, h \rangle + b_f)$$



Network parameterized by W_h, b_h, w_f, b_f

Basic Recipe: Multilayer Perceptron

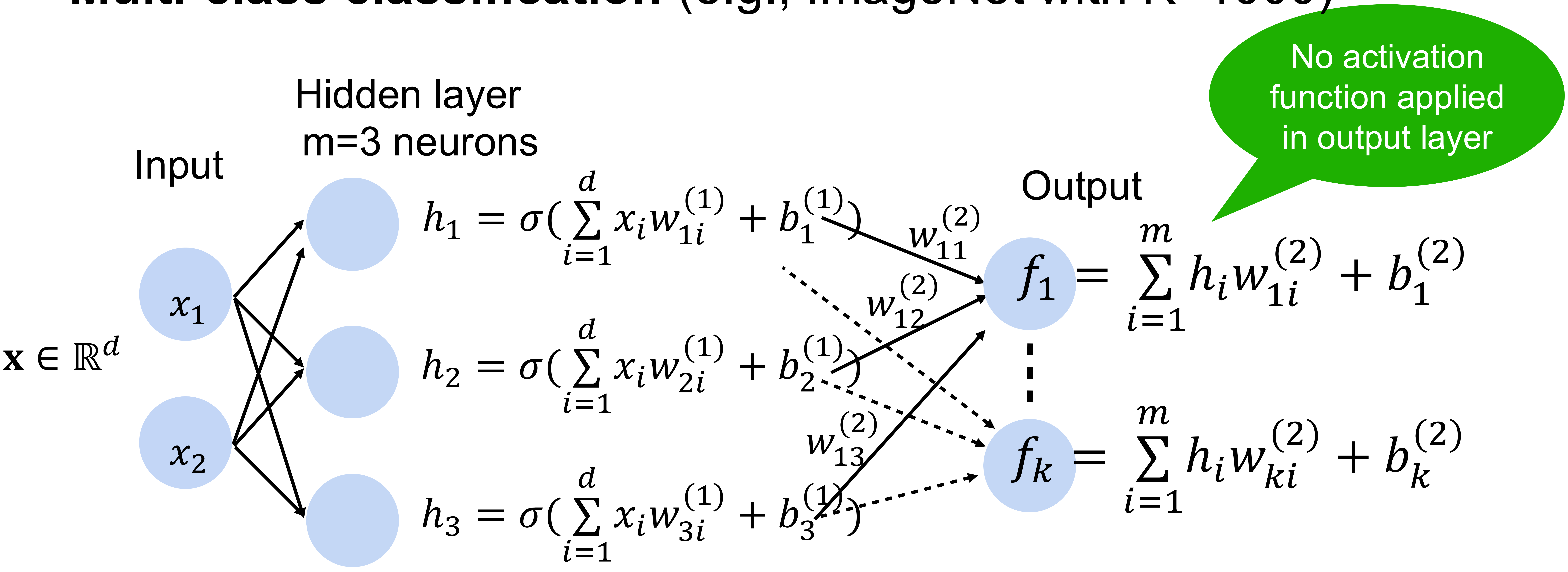
1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

1. Stacked Perceptrons
   (various activation functions)
2. Cross-entropy loss
   (usually)
3. Gradient Descent
   (usually, a variant of gradient decent)
4. Test/train split

Neural network for K-way classification

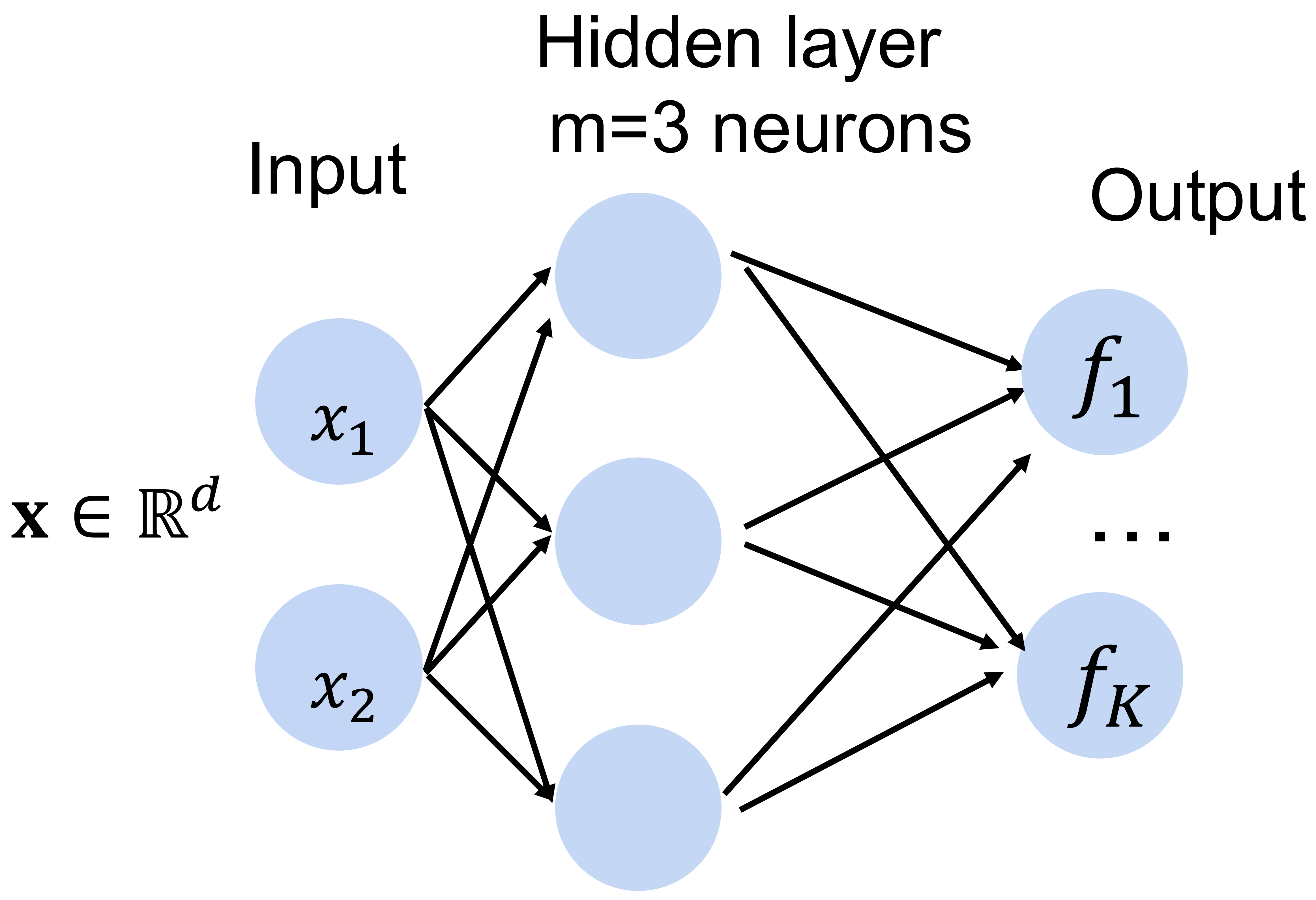
- K outputs in the final layer

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



Softmax

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

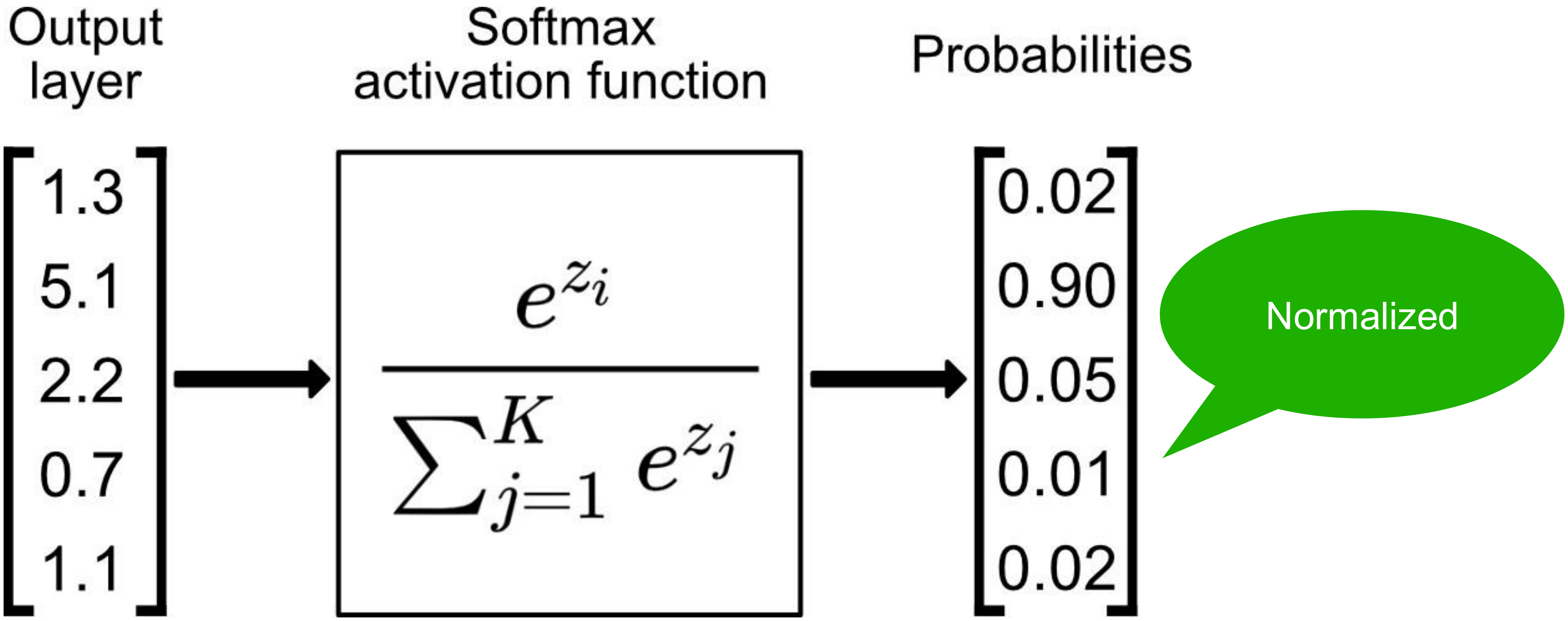


$$p(y|\mathbf{x}) = softmax(f)$$

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

Softmax

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



More complicated neural networks: multiple hidden layers

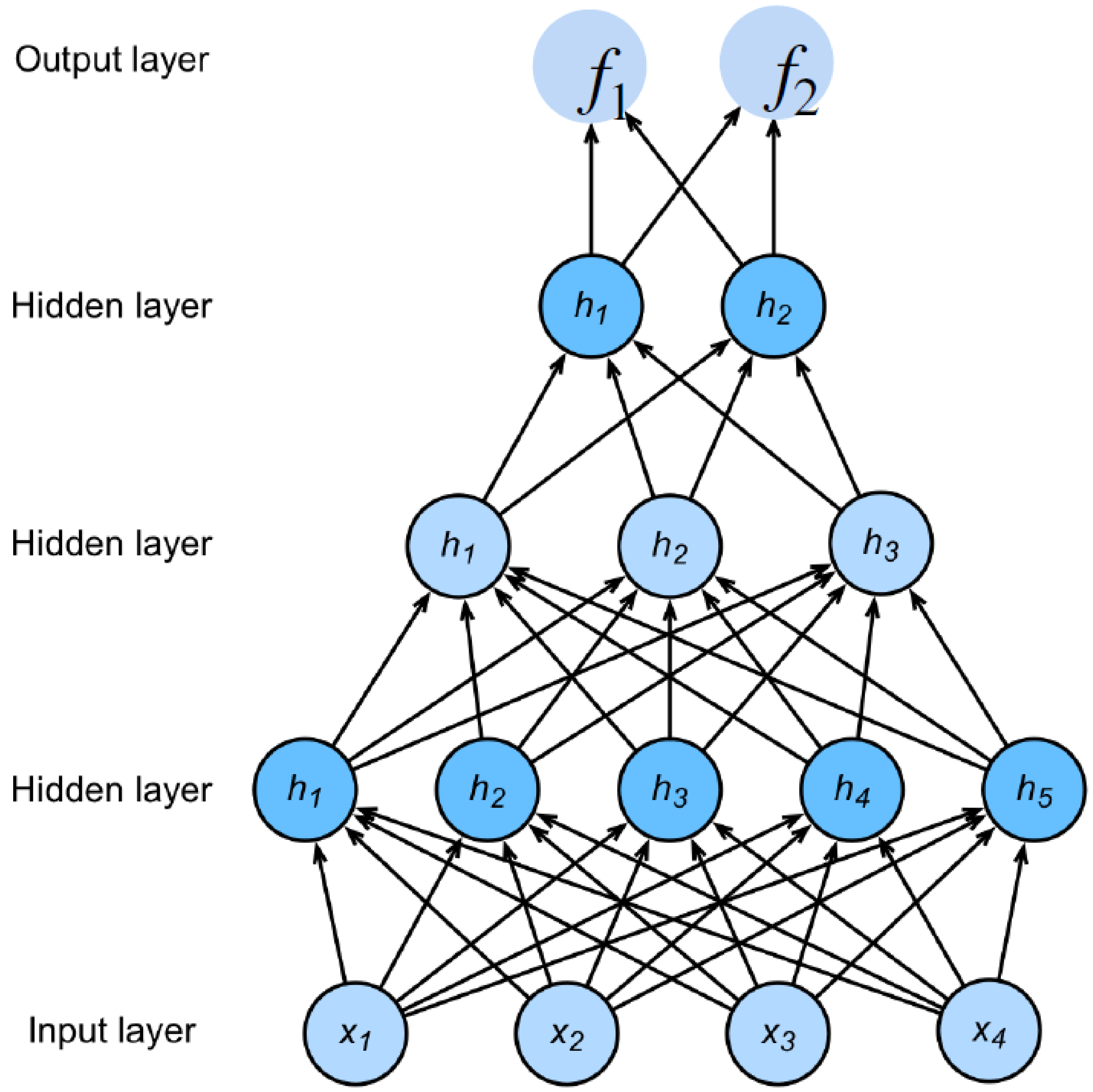
$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

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

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

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

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



How do we train these networks?

Basic Recipe: Multilayer Perceptron

1. Select model class
2. Select loss
3. Optimize parameters
4. Evaluate output

1. Stacked Perceptrons
   (various activation functions)
2. Cross-entropy loss
   (usually)
3. Gradient Descent
   (usually, a variant of gradient decent)
4. Test/train split

How to train a neural network? Binary classification

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

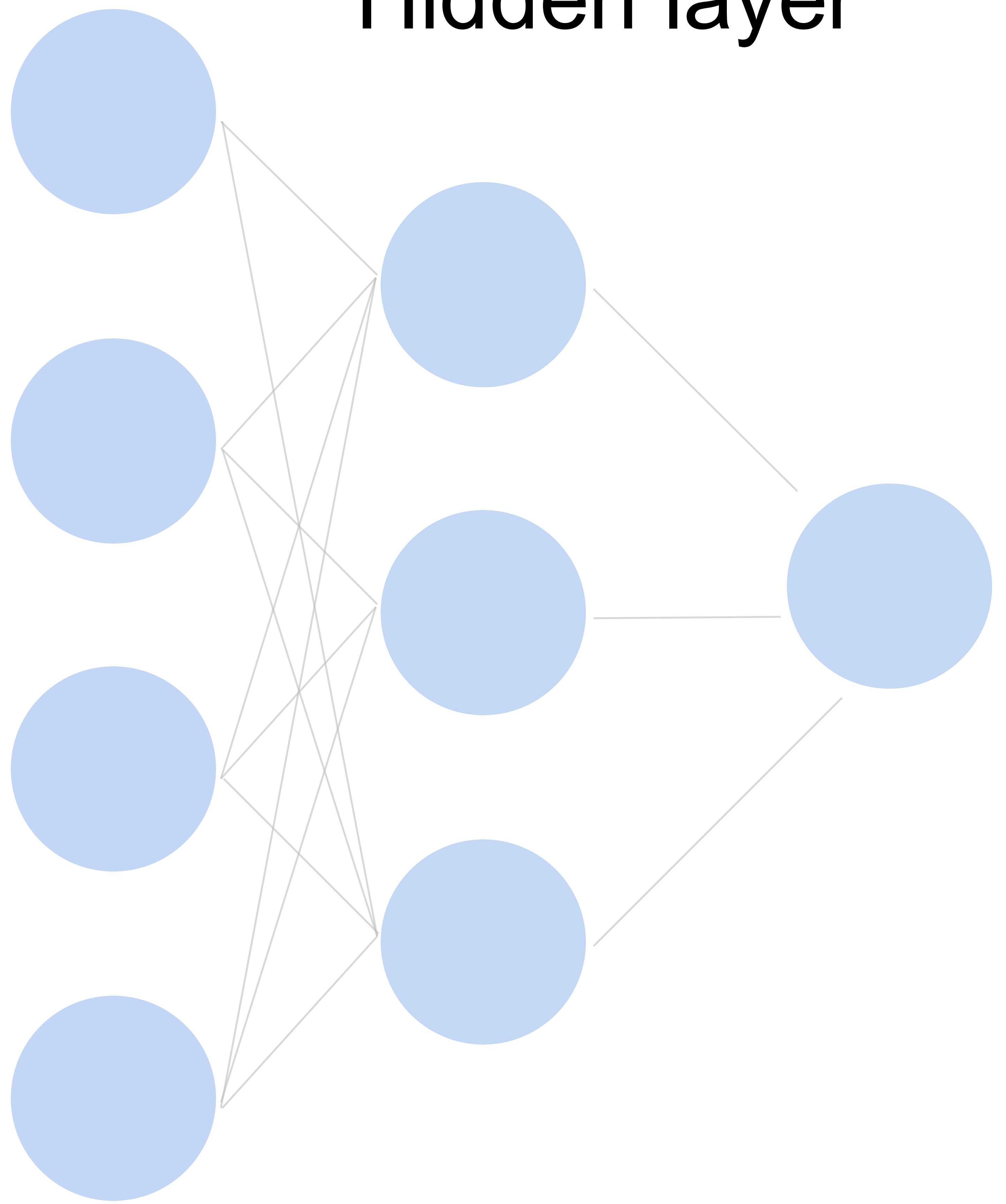
$\hat{y} \in [0,1]$ Model output for example **x**

(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 $\hat{y} \rightarrow 1$ when $y = 1$,
and $\hat{y} \rightarrow 0$ when $y = 0$



How to train a neural network? Binary classification

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

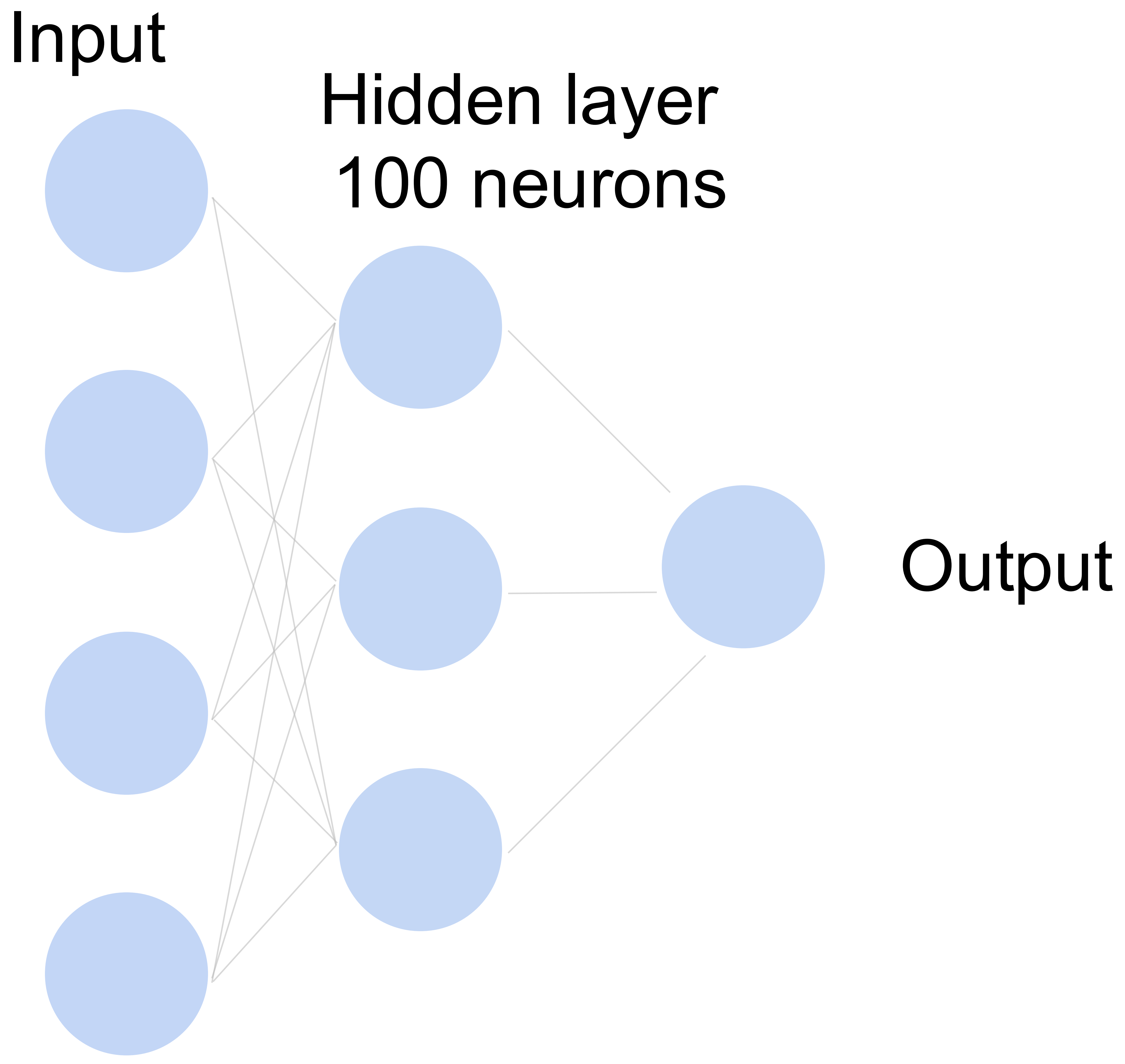
Per-sample loss:

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

Negative log likelihood

Minimizing NLL is equivalent to Max Likelihood Learning (MLE)

Also known as binary cross-entropy loss



How to train a neural network? Multiclass

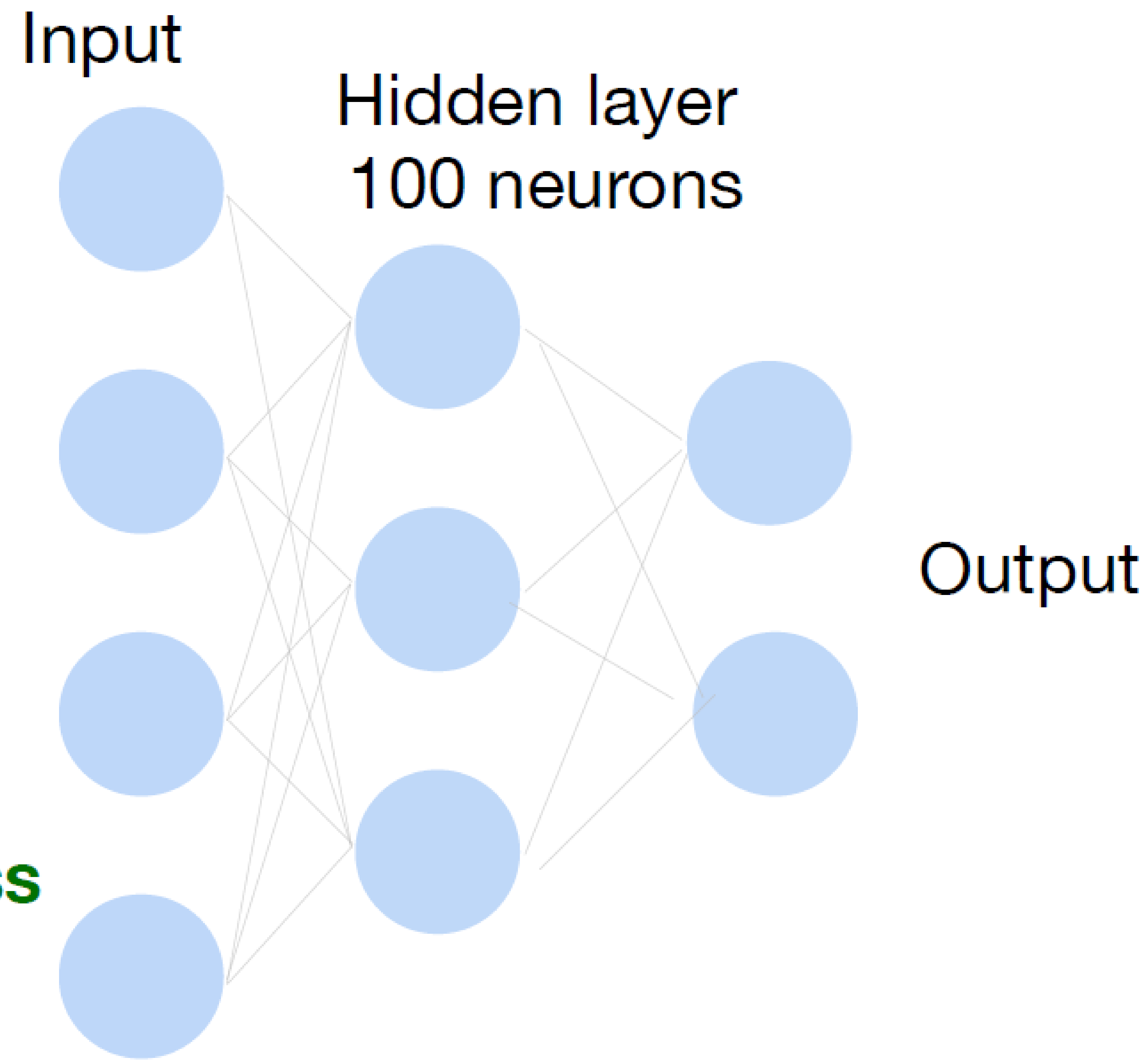
Loss function: $$\frac{1}{|D|}\sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$$

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

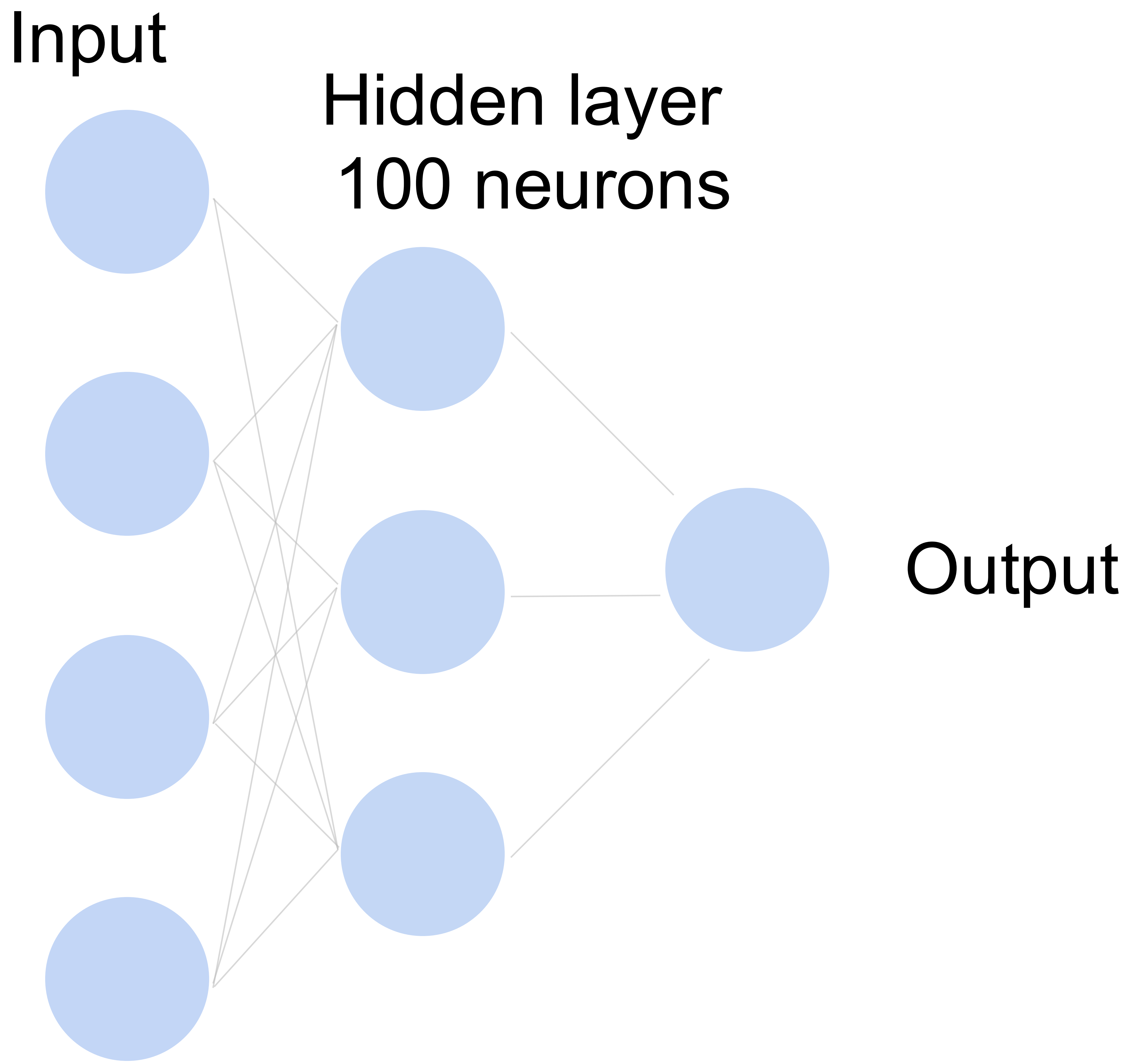


How to train a neural network?

Update the weights W to minimize the loss function

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

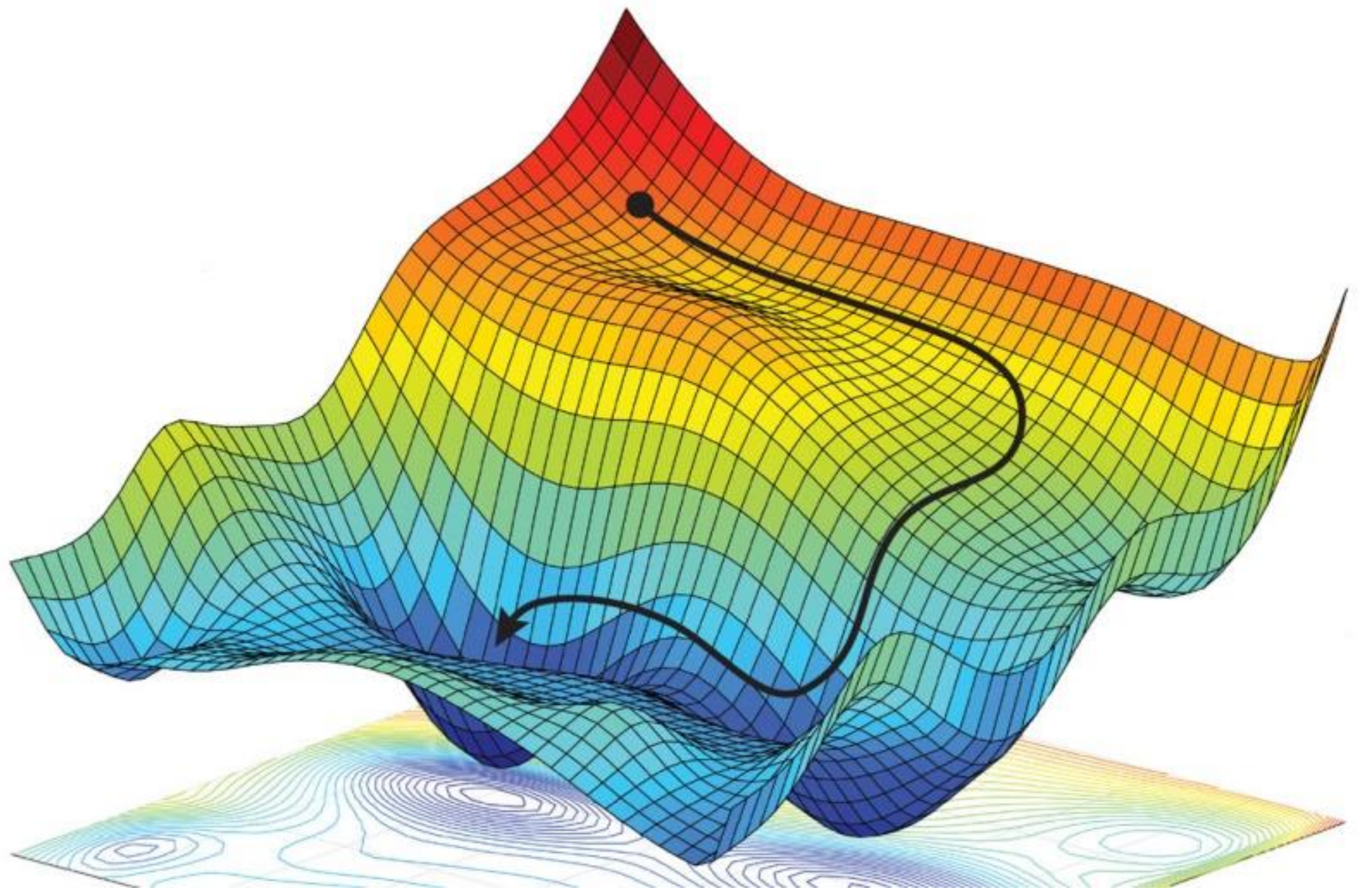
Use gradient descent!



Gradients & Gradient Descent

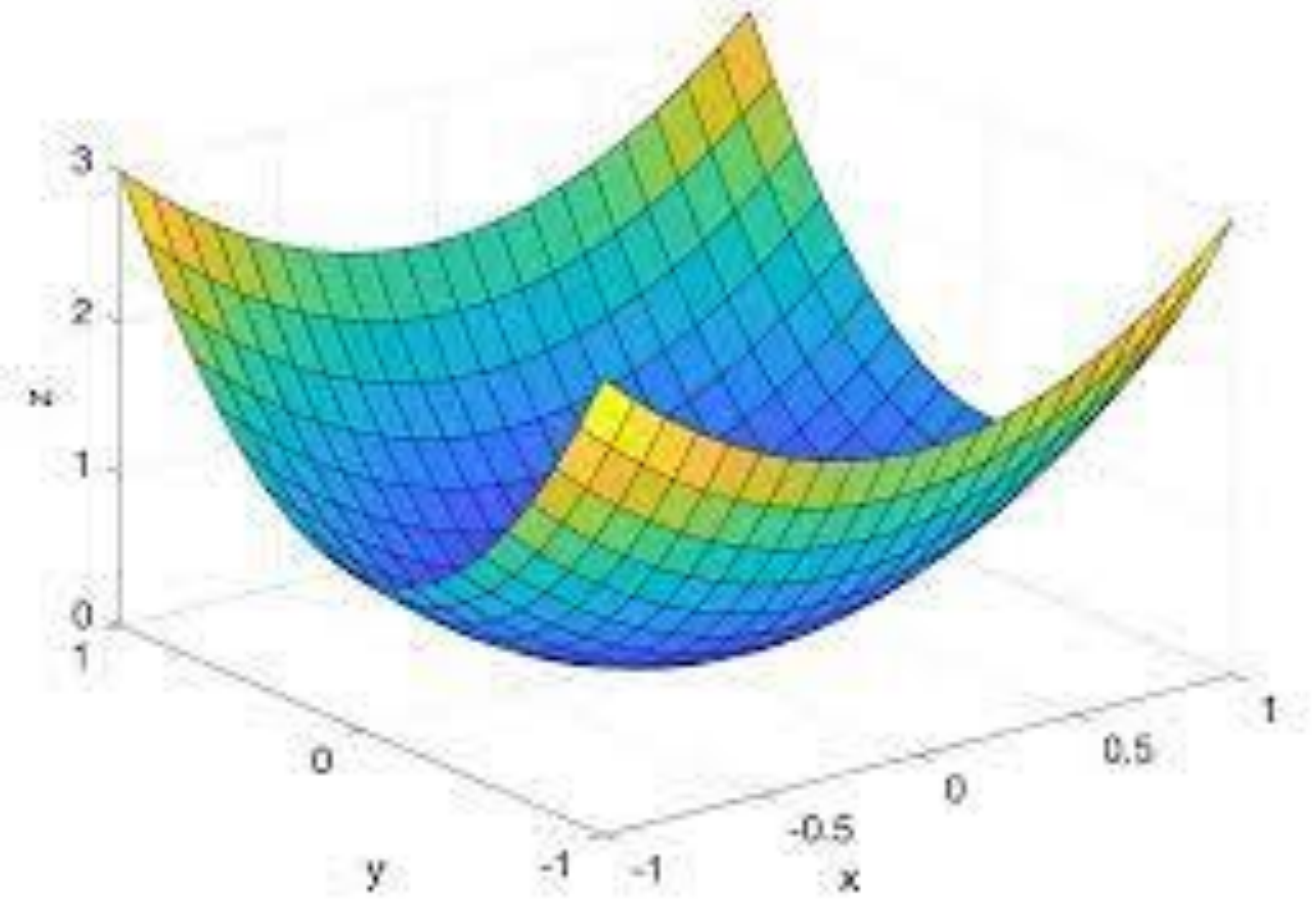
- Gradient descent takes iterative steps to make loss function smaller

Gradient Descent
<u>Input:</u> dataset (X, y), loss function L, number of steps T, step size η
1. Initialize θ_0 2. For $t = 1,2, \dots, T$ 3. Calculate $g_t = \nabla L(\theta_{t-1}; X, y)$ 4. Update $\theta_t \leftarrow \theta_{t-1} - \eta g_t$ 5. Return θ_T



M. Hutson

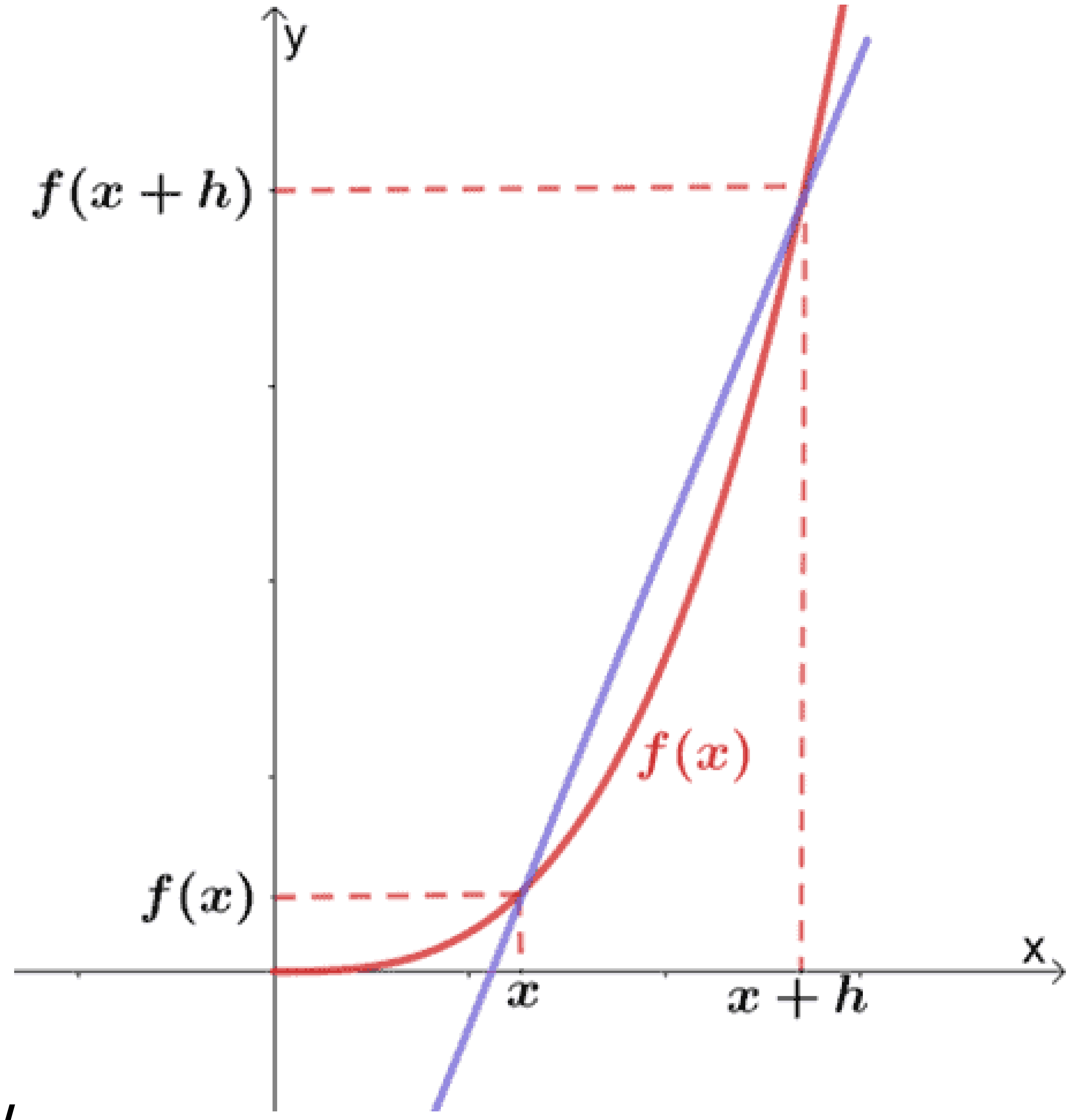
Detour: Multivariate Calculus



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.
- $\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x)-f(x)}{\Delta x}$



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 $\frac{df}{dx} = \frac{dh}{dx} + \frac{dg}{dx}$
- **Product rule:** If $f(x) = h(x) \cdot g(x)$ then $\frac{df}{dx} = h(x)\frac{dg}{dx} + g(x)\frac{dh}{dx}$
- **Chain rule:** If $f(x) = h(g(x))$ then $\frac{df}{dx} = \frac{dh}{dg}\frac{dg}{dx}$
- **More complex chain rule:** if $f(x) = f_1(f_2(\cdots f_n(x)\cdots))$ then $\frac{df}{dx}$ $= \frac{df_1}{df_2}\frac{df_2}{df_3}\cdots\frac{df_n}{dx}$

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/

Derivatives of functions of multiple variables

- Generalize derivative to functions of form $f(x_1, x_2, \ldots, x_n)$.
- 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 these partial derivatives.
  - $\nabla f = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n},]$

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

Dissecting our Gradient Notation

- When f maps vector $x \in \mathbb{R}^d$ to scalar $f(x) \in \mathbb{R}$

$$\nabla f(x) \in \mathbb{R}^d$$

- We will work with functions of many vectors (and matrices!)

$$L(\theta; X, y)$$

- Write gradient with respect to θ

$$\nabla_\theta L(\theta_{t-1}; X, y)$$

Gradient Descent
Input: dataset (X, y), loss function L, number of steps T, step size η
1. Initialize θ_0 2. For $t = 1,2, \ldots, T$ 3. Calculate $g_t = \nabla_\theta L(\theta_{t-1}; X, y)$ 4. Update $\theta_t \leftarrow \theta_{t-1} - \eta g_t$ 5. Return θ_T

(Minibatch) Stochastic Gradient Descent

- Gradient descent uses loss on entire dataset every step:

$$\nabla L(\theta; X, y) = \sum_{i=1}^{n} \nabla \ell(\theta; x_i, y_i)$$

- This is extremely inefficient!

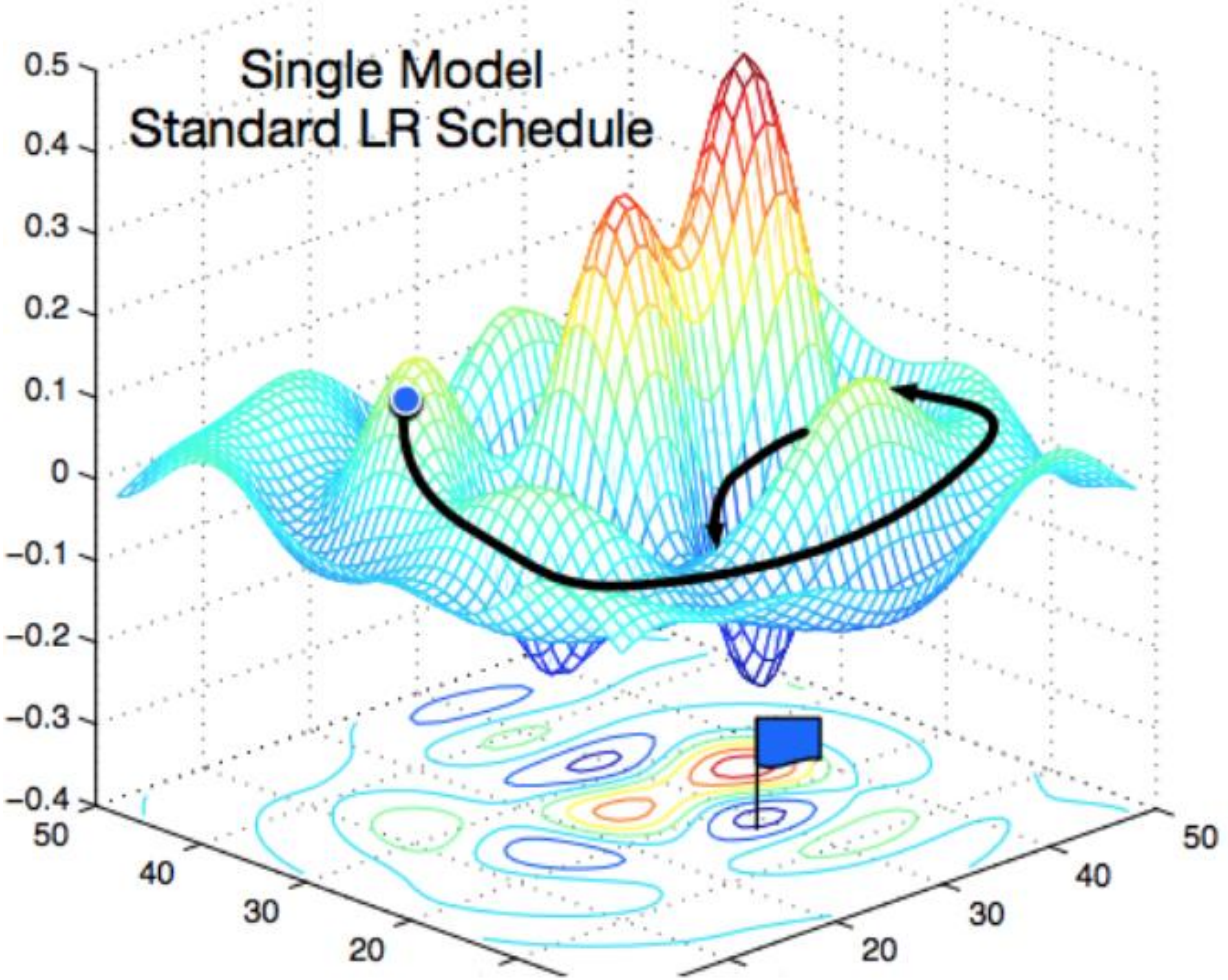
- On big datasets, better to update based on a few examples

Stochastic Gradient Descent

Input: dataset (X, y), loss function L, number of steps T, step size η, batch size m

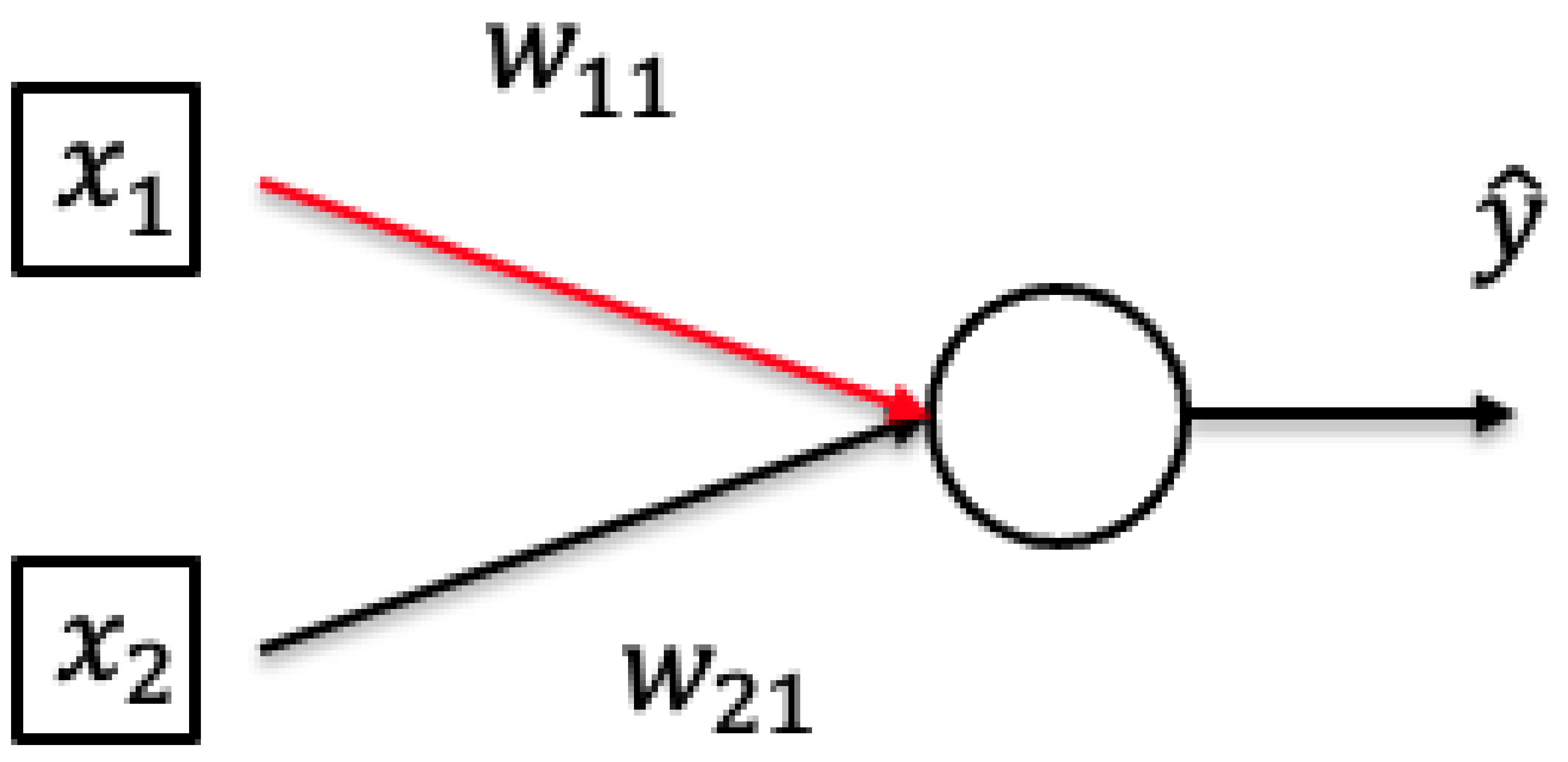
1. Initialize θ_0
2. For $t = 1,2, \ldots, T$
3. Select random $(x_1, y_1), \ldots, (x_m, y_m)$
4. Calculate $g_t = \sum_{i=1}^{m} \nabla_\theta \ell(\theta_{t-1}; x_i, y_i)$
5. Update $\theta_t \leftarrow \theta_{t-1} - \eta g_t$
6. Return θ_T

How do we get the gradient?



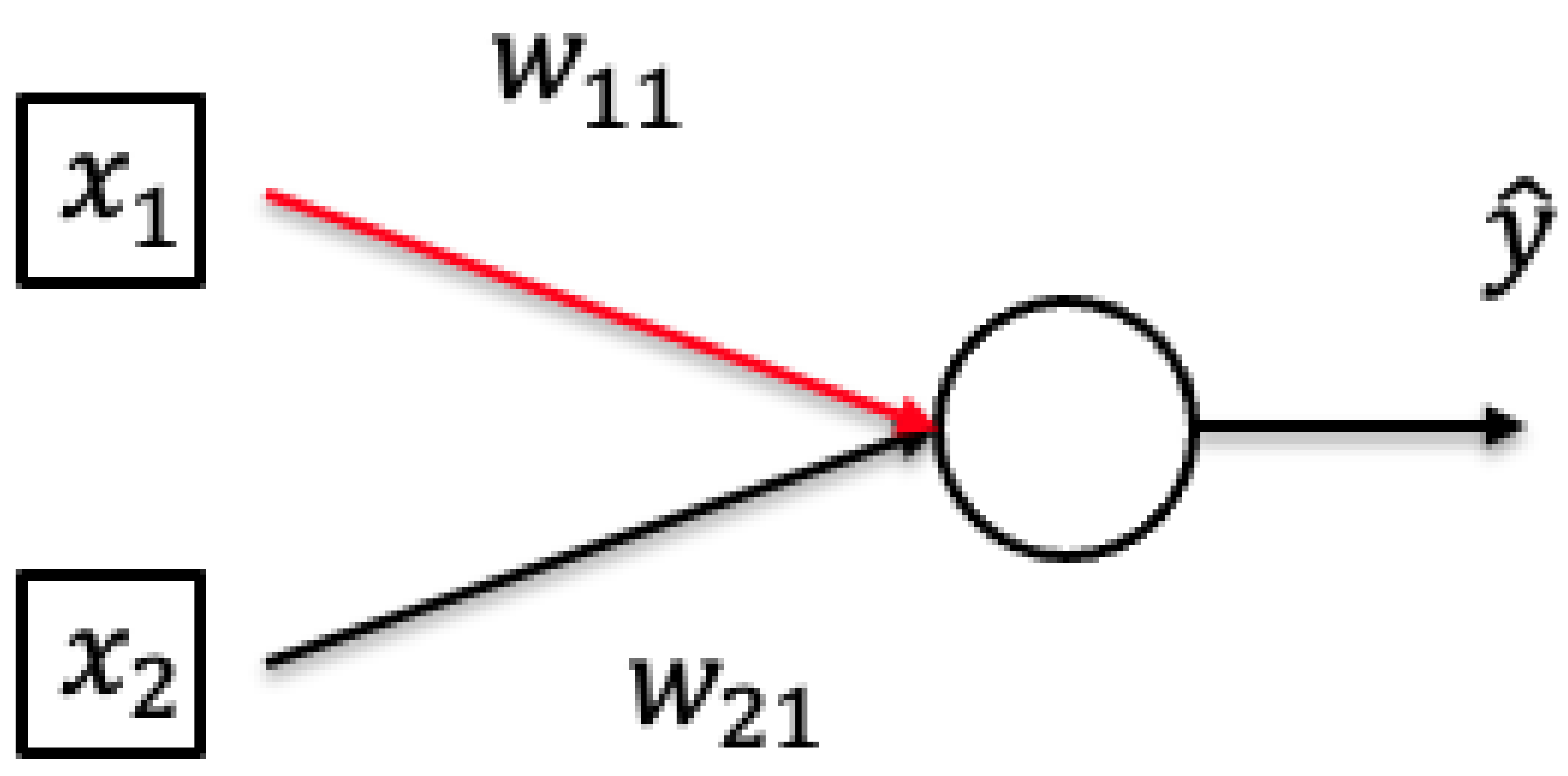
[Gao and Li et al., 2018]

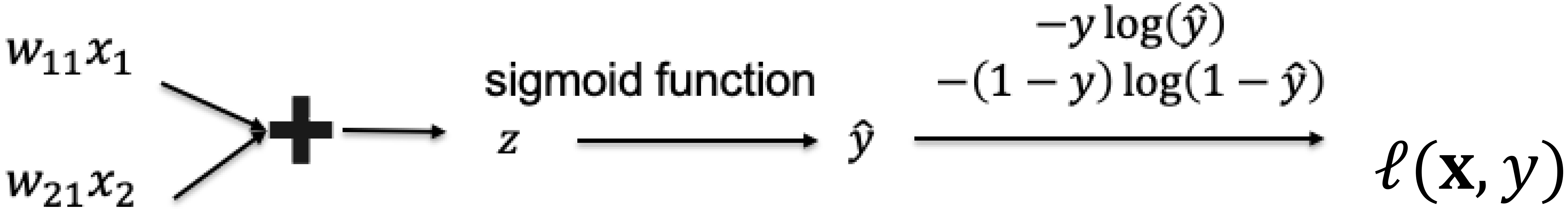
Calculate Gradient (on one data point)



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

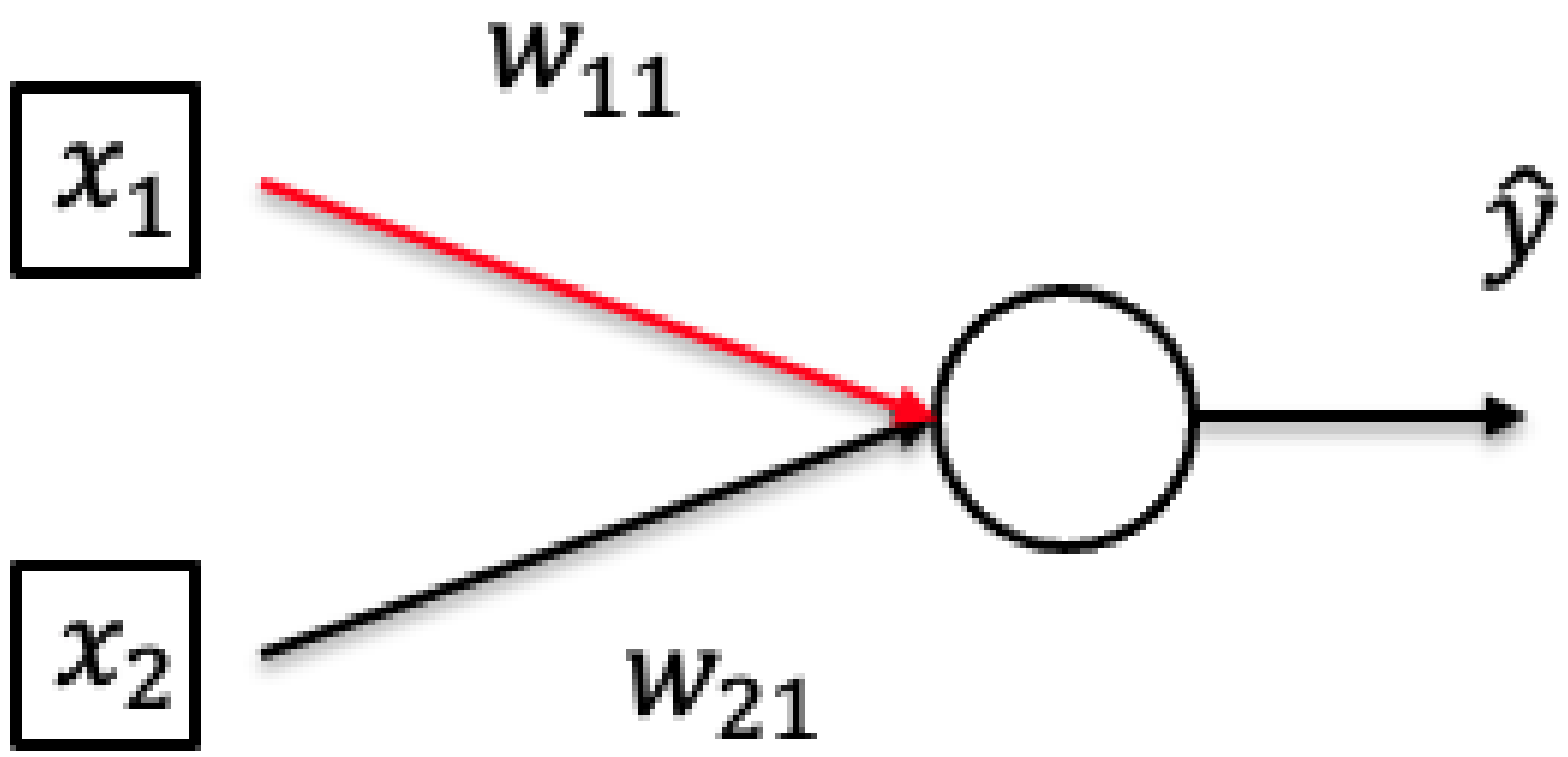
Calculate Gradient (on one data point)

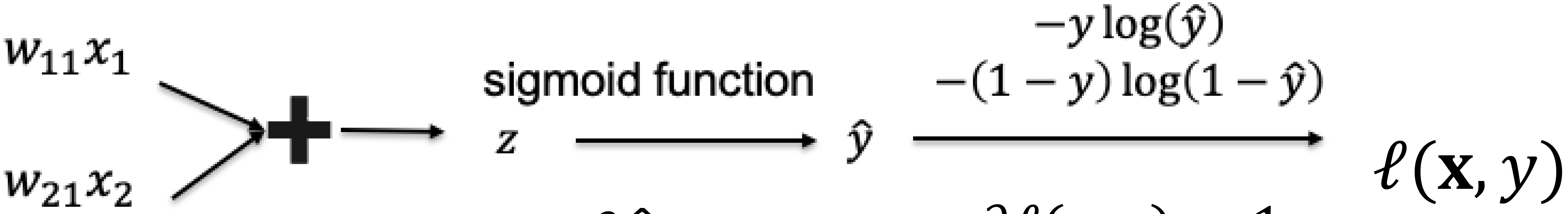




Use chain rule!

Calculate Gradient (on one data point)



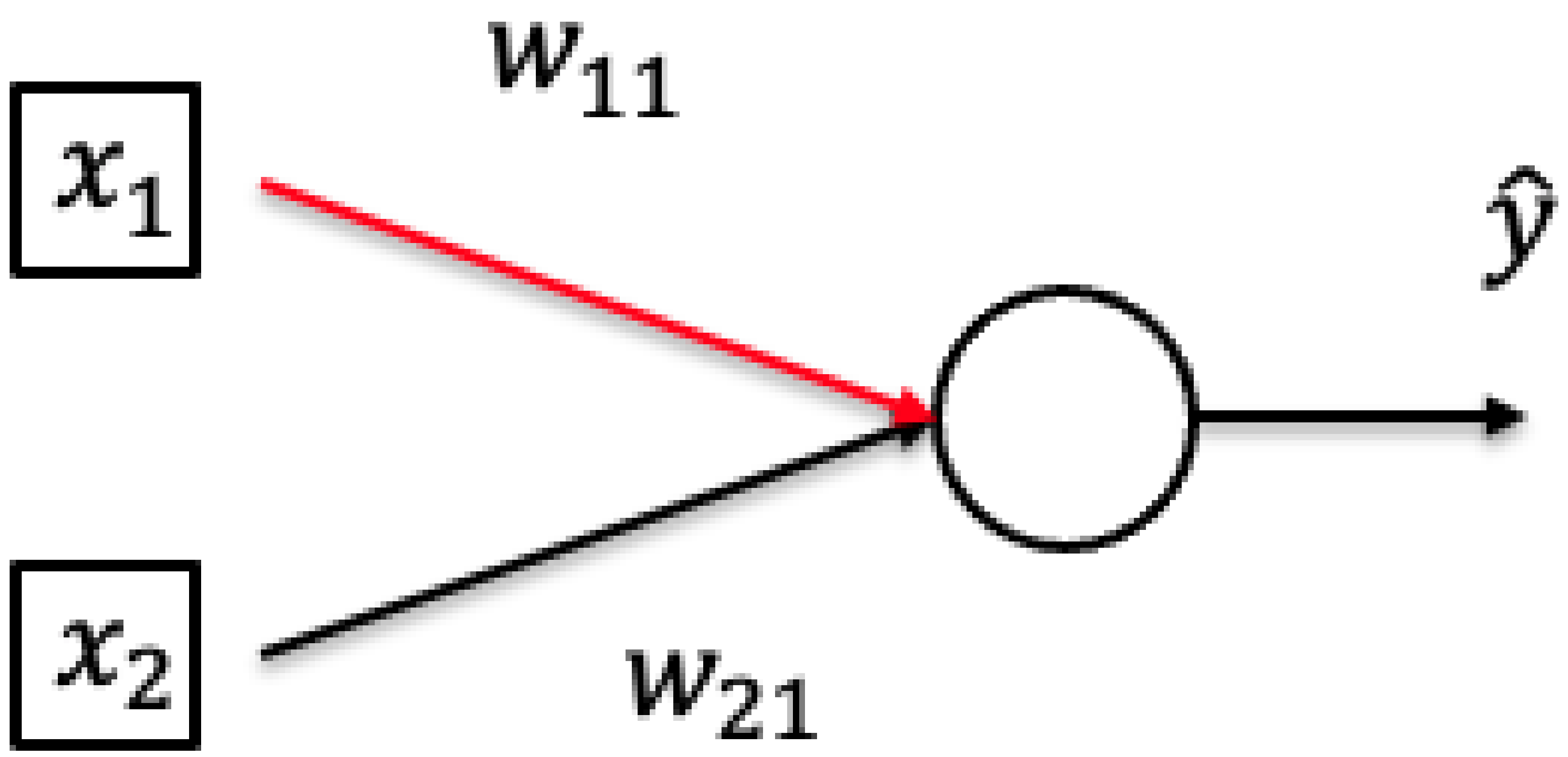


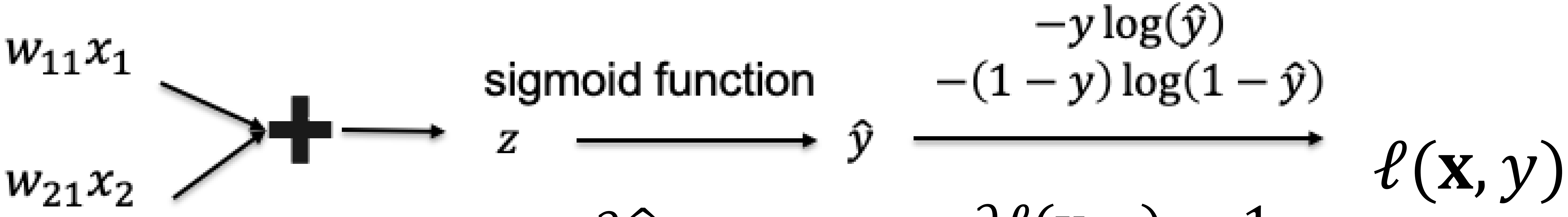
$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z)$$

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

- 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)

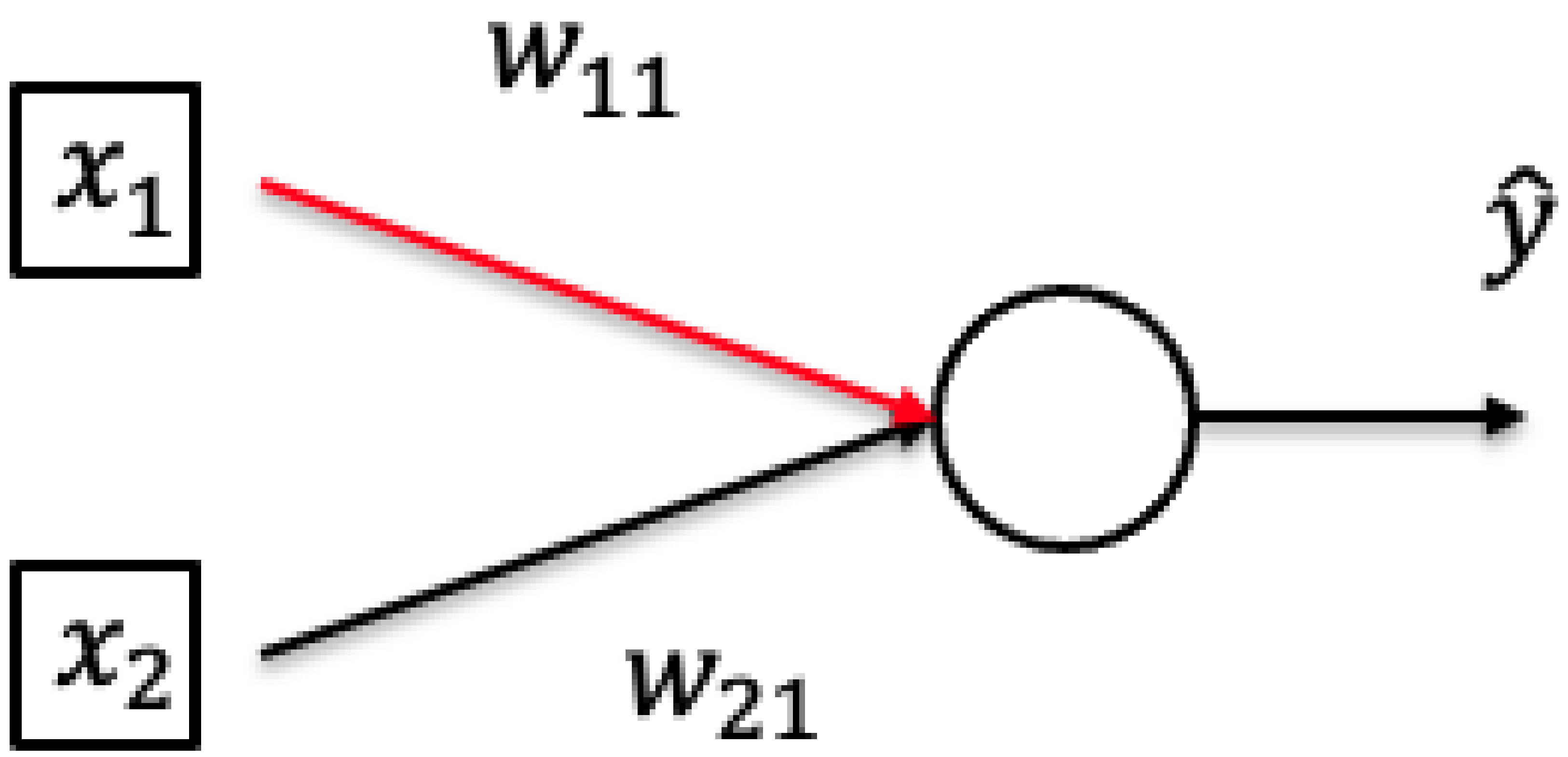


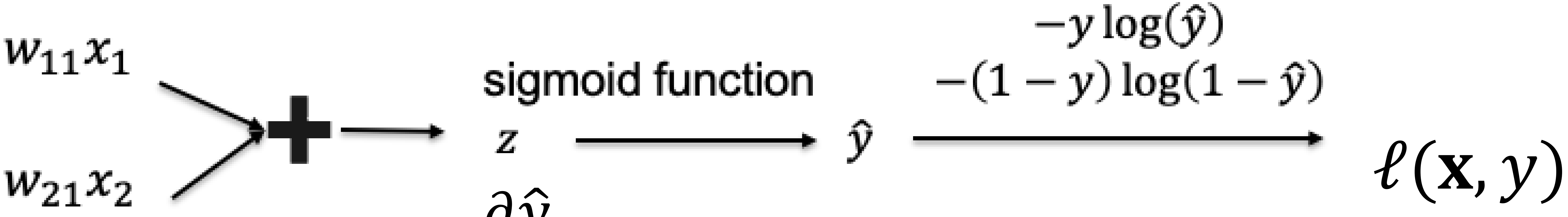


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) \qquad \frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- 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)

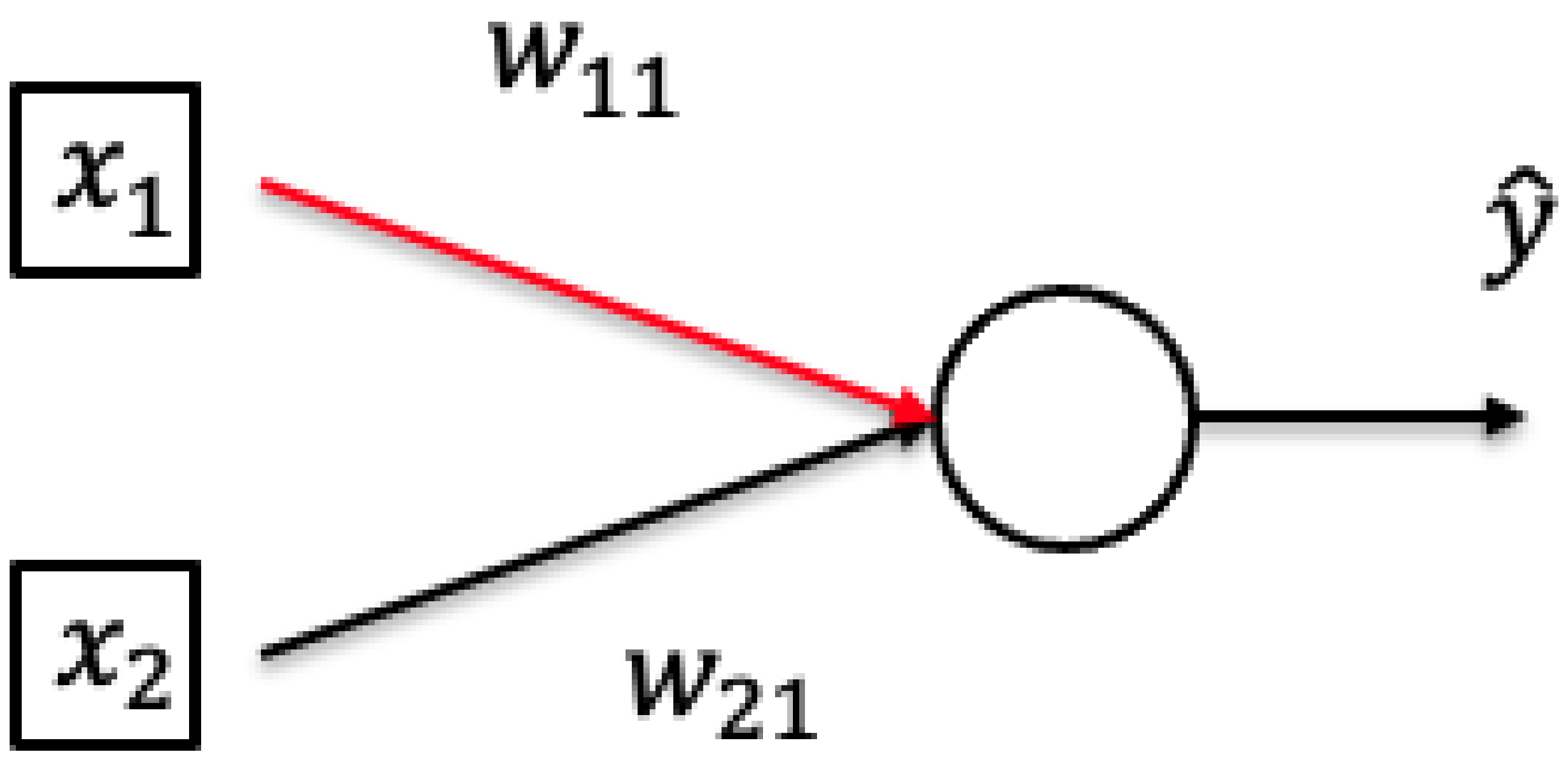


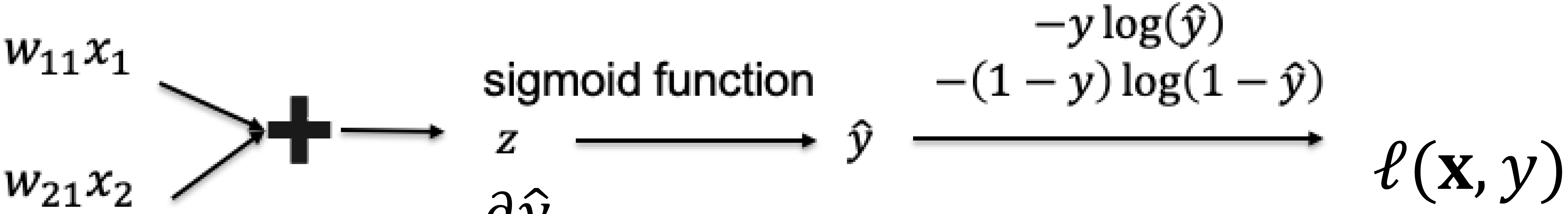


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1 - \sigma(z))$$

- 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)

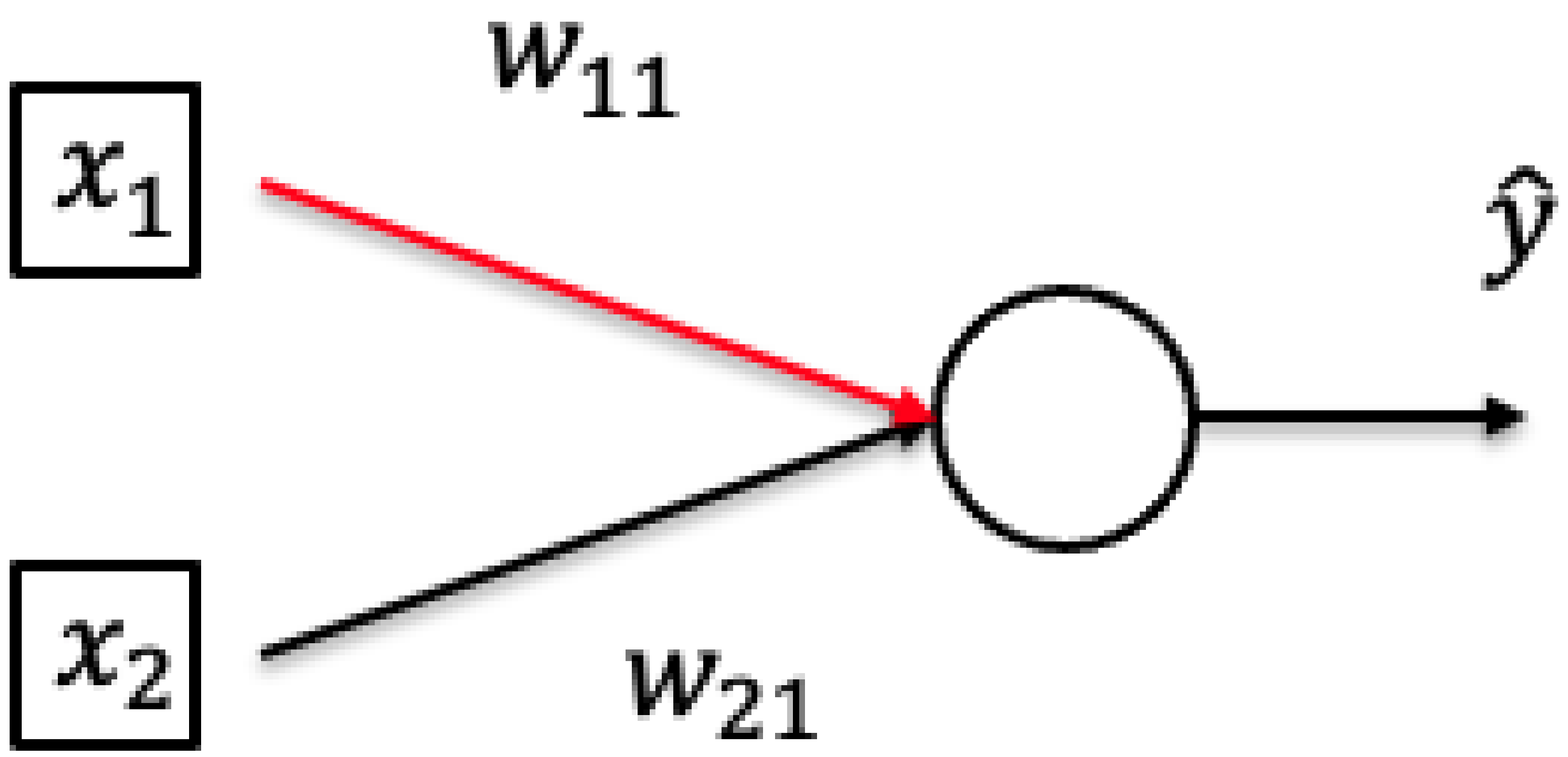


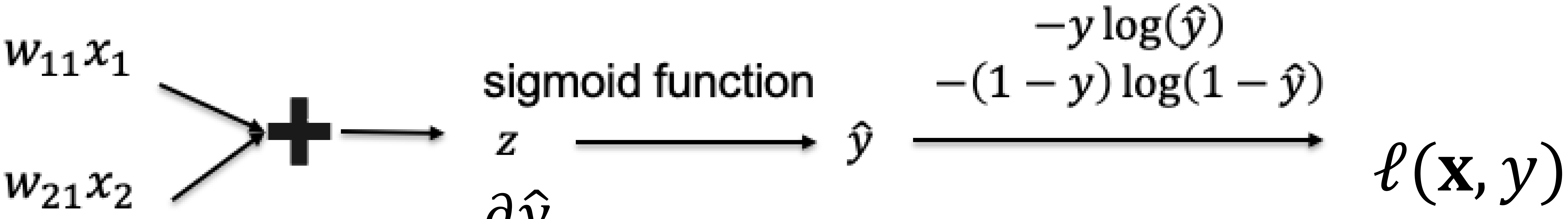


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- 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)

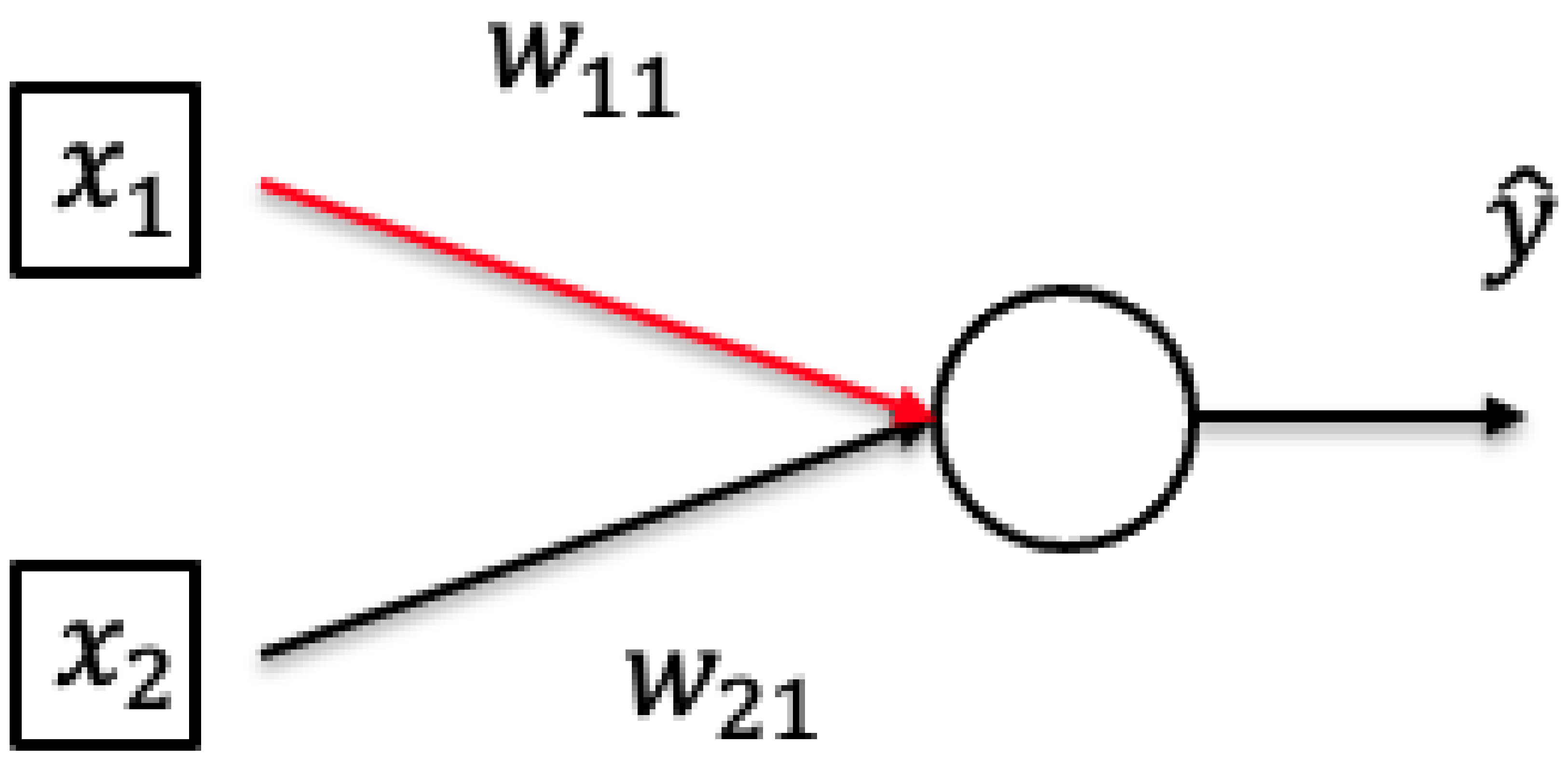


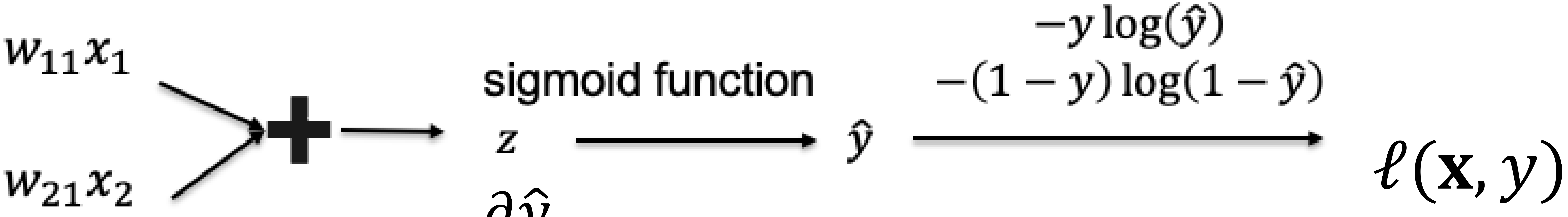


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

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

Calculate Gradient (on one data point)

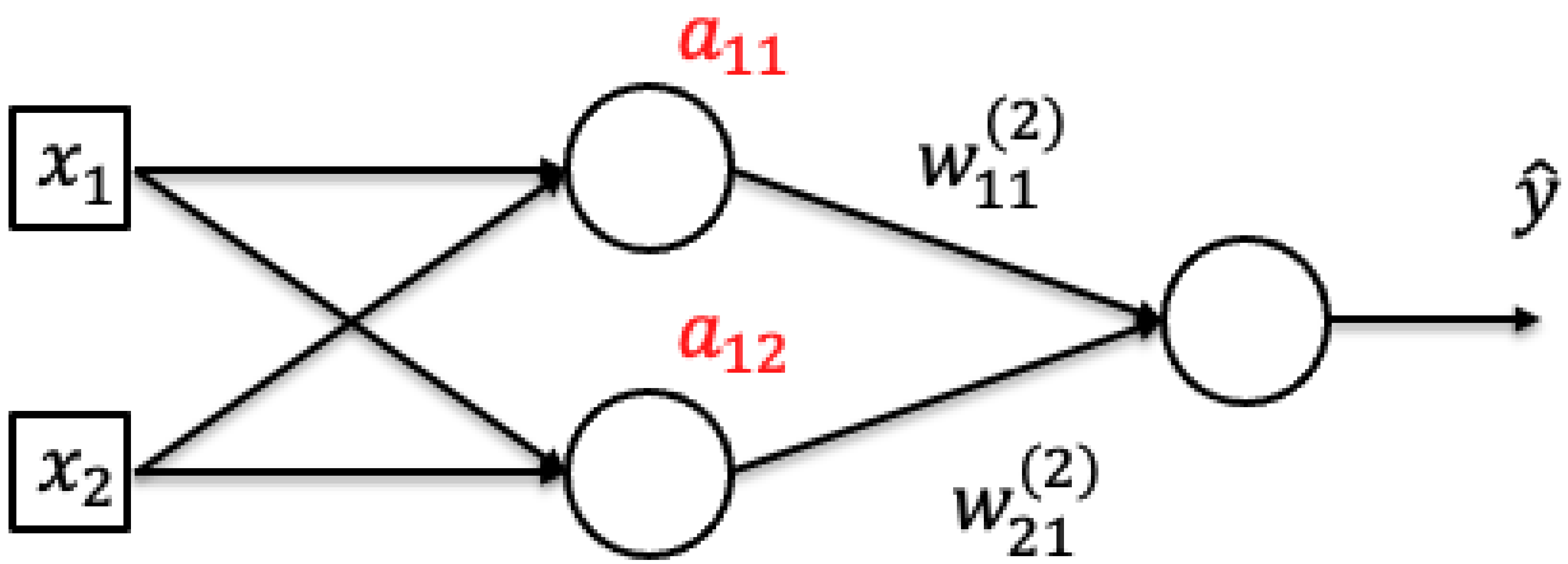




$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- 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)



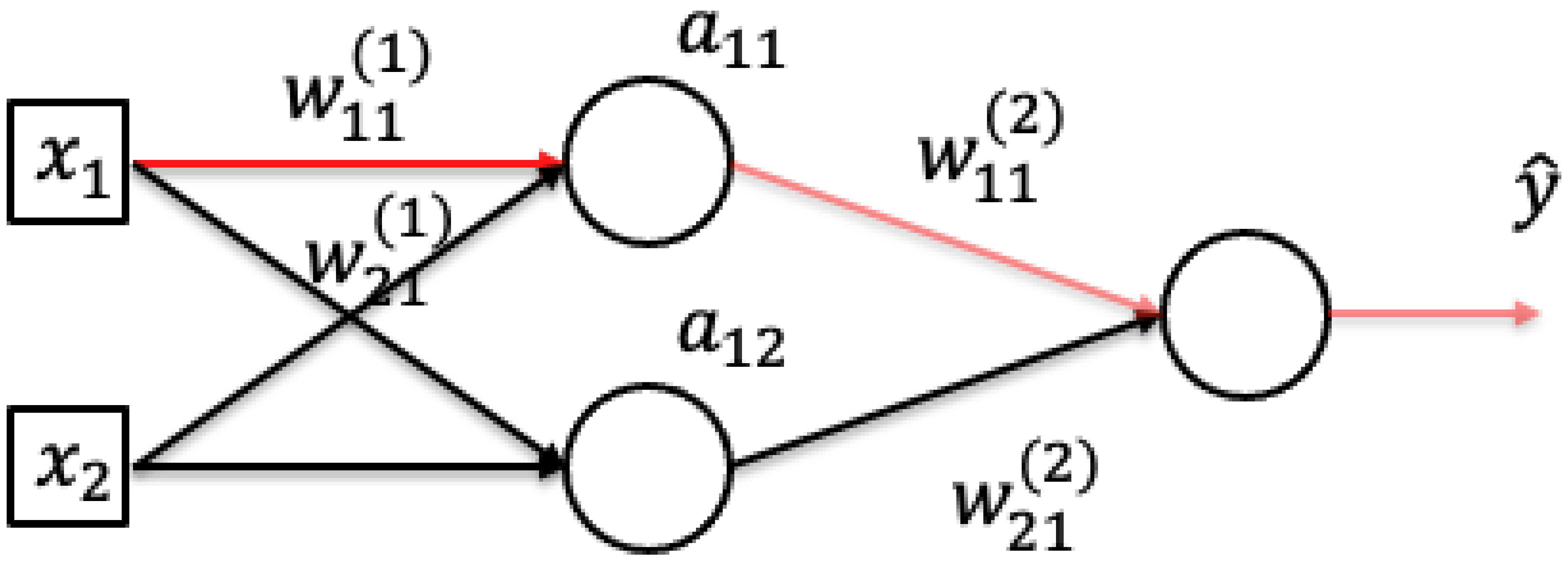
Make it deeper

$$w_{11}^{(2)}a_{11} + w_{21}^{(2)}a_{12} \longrightarrow z \xrightarrow{\text{sigmoid function}} \hat{y} \xrightarrow{-y\log(\hat{y}) \;-(1-y)\log(1-\hat{y})} \ell(\mathbf{x}, y)$$

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

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

Calculate Gradient (on one data point)

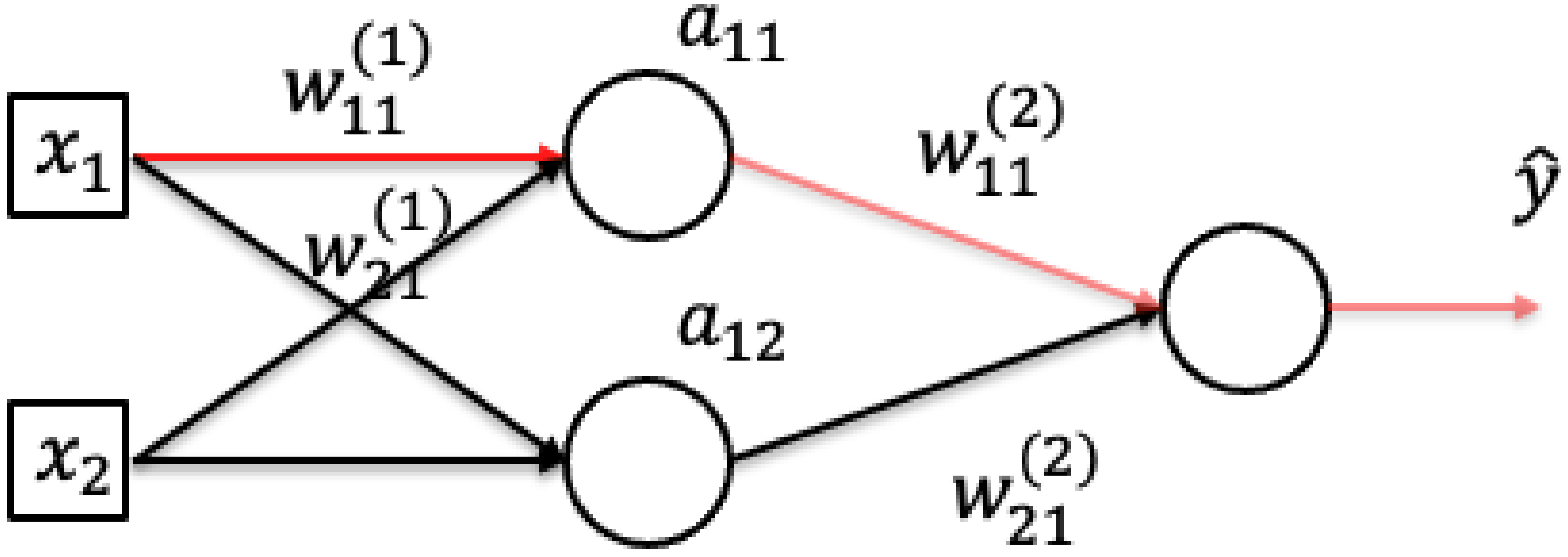


$$w_{11}^{(1)} x_1, \; w_{21}^{(1)} x_2 \longrightarrow + \longrightarrow z_{11} \xrightarrow{\sigma(z_{11})} a_{11} \longrightarrow l(x, y)$$

$$\frac{\partial a_{11}}{\partial z_{11}} = \sigma'(z_{11}) \qquad \frac{\partial l}{\partial a_{11}} = (\hat{y} - y) w_{11}^{(2)}$$

- 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)

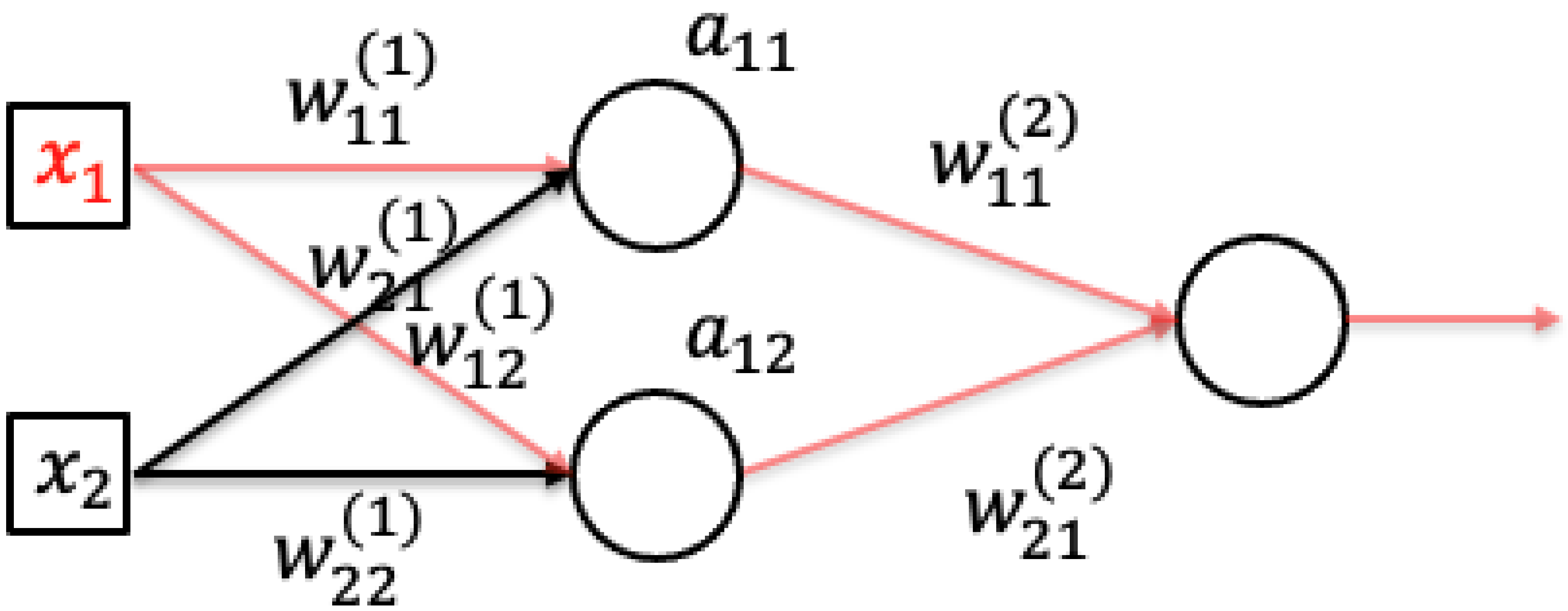


$$w_{11}^{(1)} x_1, \; w_{21}^{(1)} x_2 \longrightarrow + \longrightarrow z_{11} \xrightarrow{\sigma(z_{11})} a_{11} \longrightarrow l(\boldsymbol{x}, y)$$

$$\frac{\partial a_{11}}{\partial z_{11}} = \sigma'(z_{11}) \qquad \frac{\partial l}{\partial a_{11}} = (\hat{y} - y) w_{11}^{(2)}$$

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

Calculate Gradient (on one data point)



$$w_{11}^{(1)}x_1, \; w_{21}^{(1)}x_2 \longrightarrow + \longrightarrow z_{11} \xrightarrow{\sigma(z_{11})} a_{11} \longrightarrow l(x, y)$$

$$\frac{\partial a_{11}}{\partial z_{11}} = \sigma'(z_{11}) \qquad \frac{\partial l}{\partial a_{11}} = (\hat{y} - y)w_{11}^{(2)}$$

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