

# Deep Forward Networks - Introduction

Wednesday  
08h00 – 09h00

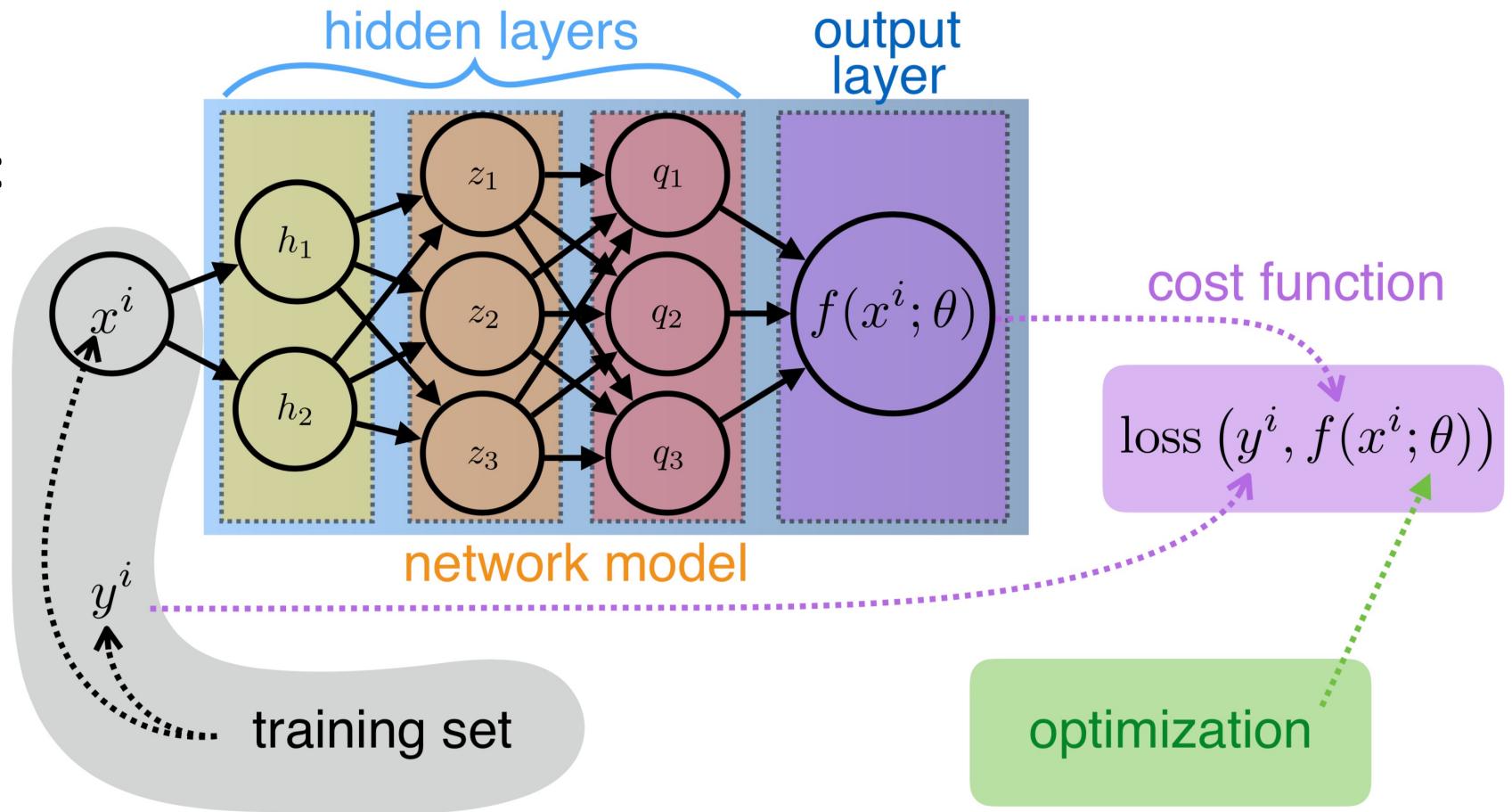
# Deploying a Neural Network

Given a task (in terms of **I/O mappings**), we need :

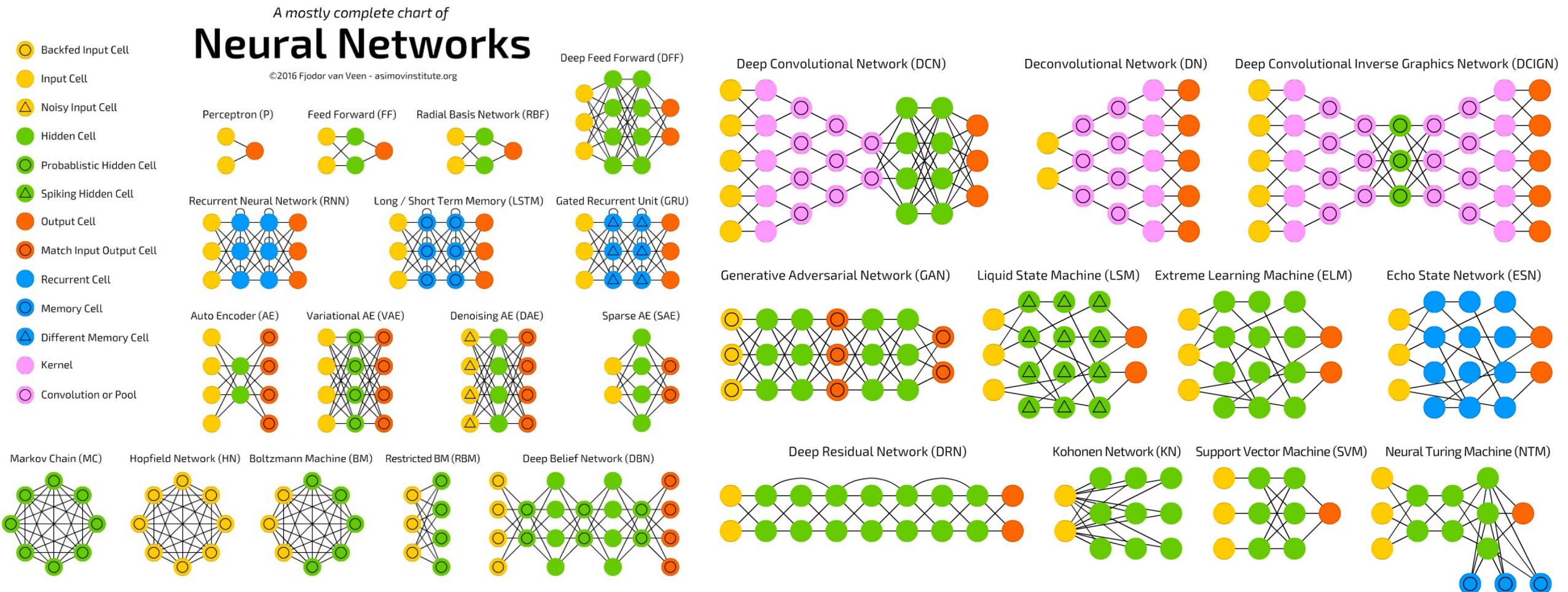
1) **Network model**

2) **Cost function**

3) **Optimization**



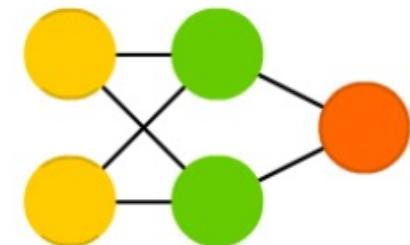
# 1) Network Model



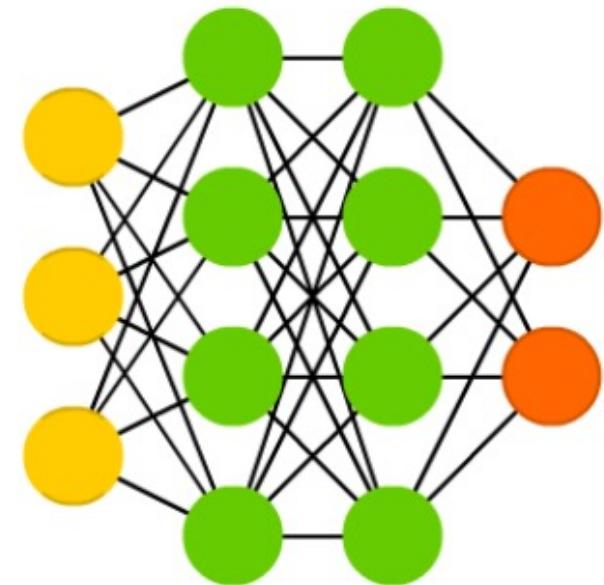
# (Deep) Feedforward NN (DFF)

- the **simplest type** of neural network
- All units are **fully connected** (between layers)
- information flows from **input** to **output** layer  
**without back loops**
- The first single-neuron network was proposed already in 1958 by AI pioneer Frank Rosenblatt
- Deep for “more than 1 **hidden layer**”

Feed Forward (FF)

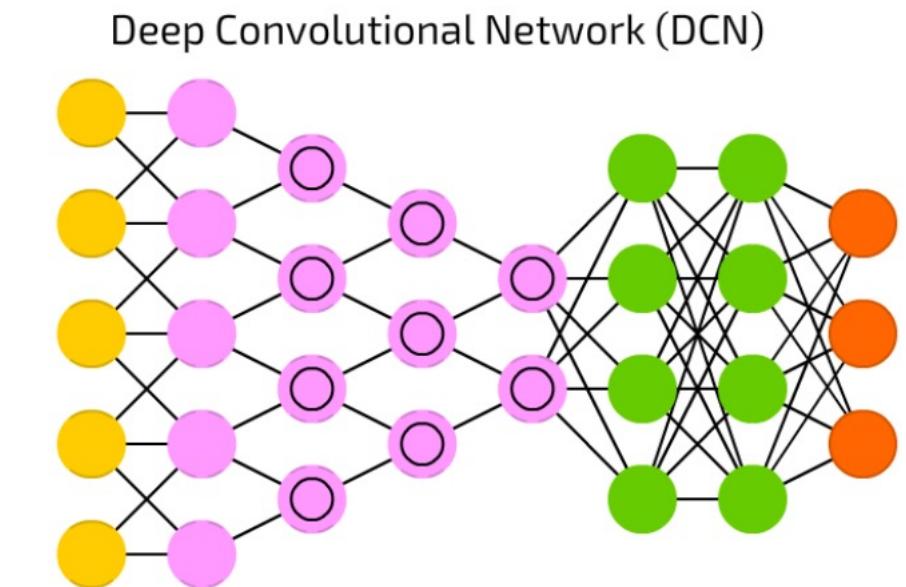


Deep Feed Forward (DFF)



# Convolutional Neural Networks (CNN)

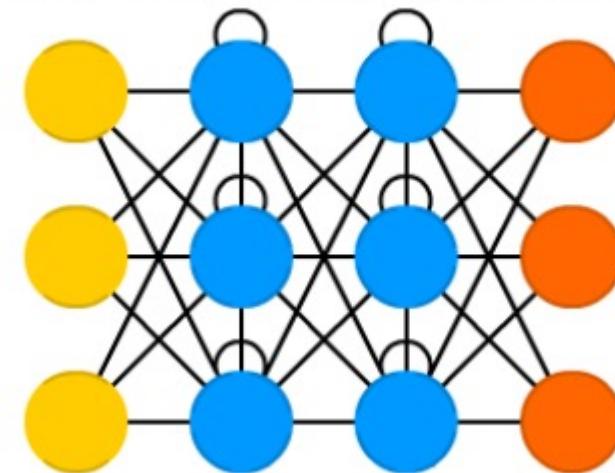
- inspired by the organization of the animal visual cortex
- Kernel and convolution or pool cells used to process and simplify input data
  - Weight sharing between *local regions*  
e.g. adjacent pixels in pictures
- well suited for computer vision tasks
  - Image classification
  - Object detection



# Recurrent Neural Networks (RNN)

- connections between neurons include loops
- **Recurrent cells** (or memory cells) used
  - Weight sharing between *time-steps*
- well-suited for processing sequences of inputs, when **context is important**
  - Text analysis

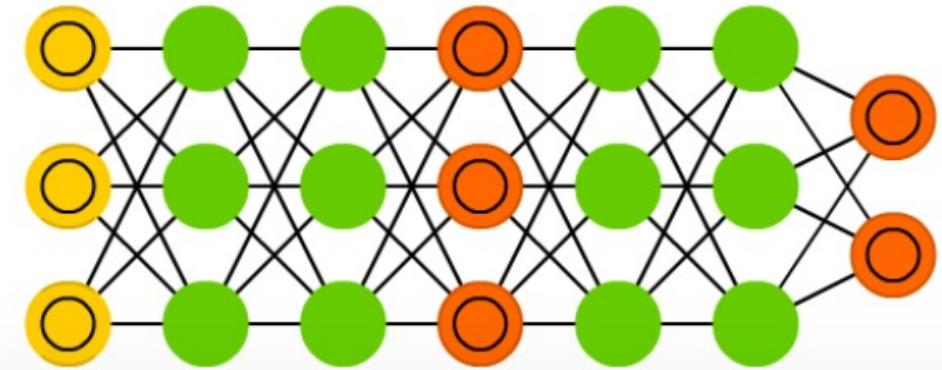
Recurrent Neural Network (RNN)



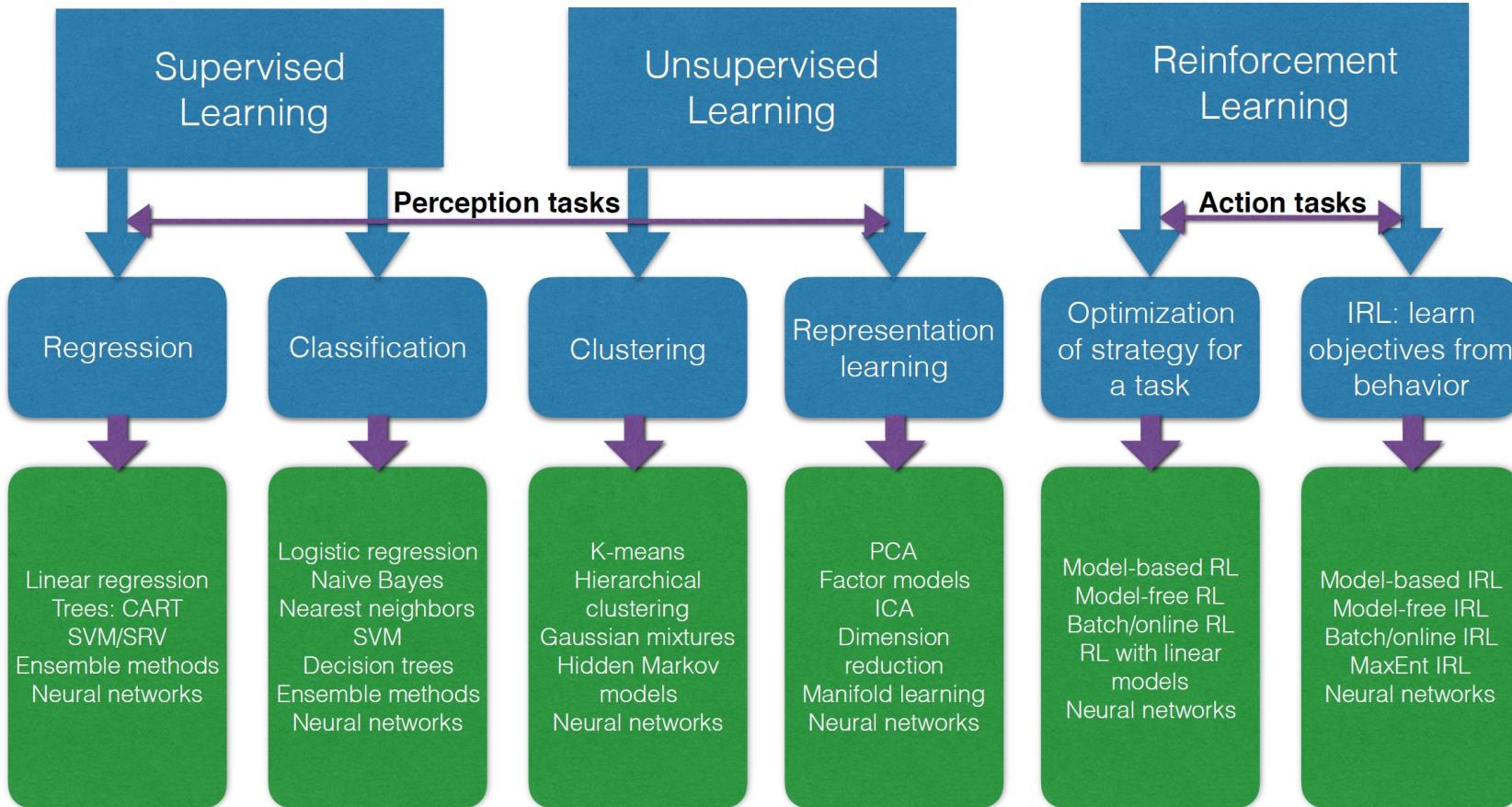
# Generative Adversarial Networks (GAN)

- More of a **Training Paradigm** rather than an architecture
- Double networks composed from generator and discriminator.
- They constantly try to fool each other, hence contain **backfed input cells** and **match input output cells**.
- well-suited for **generating real-life** images, text or speech

Generative Adversarial Network (GAN)



# Use cases



Neural networks are applicable to all problems, but are less specific

## 2) Loss and Cost functions

- Loss function  $L(\hat{y}^{(i)}, y^{(i)})$ , also called error function, measures how different the prediction  $\hat{y} = f(x)$  and the desired output  $y$  are

for each data point

- Cost function  $J(w, b)$  is the average of the loss function on the *entire training set*

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$$

- Goal of the optimization is to find the *parameters*  $\theta = (w, b)$  that minimize the cost function

empirical average --> approximation for our expected value

# 3) Optimization

- Given a task we define

- Training data

$$\{x^i, y^i\}_{i=1,\dots,m}$$

- Network

$$f(x; \theta)$$

*x as input*

- Cost function

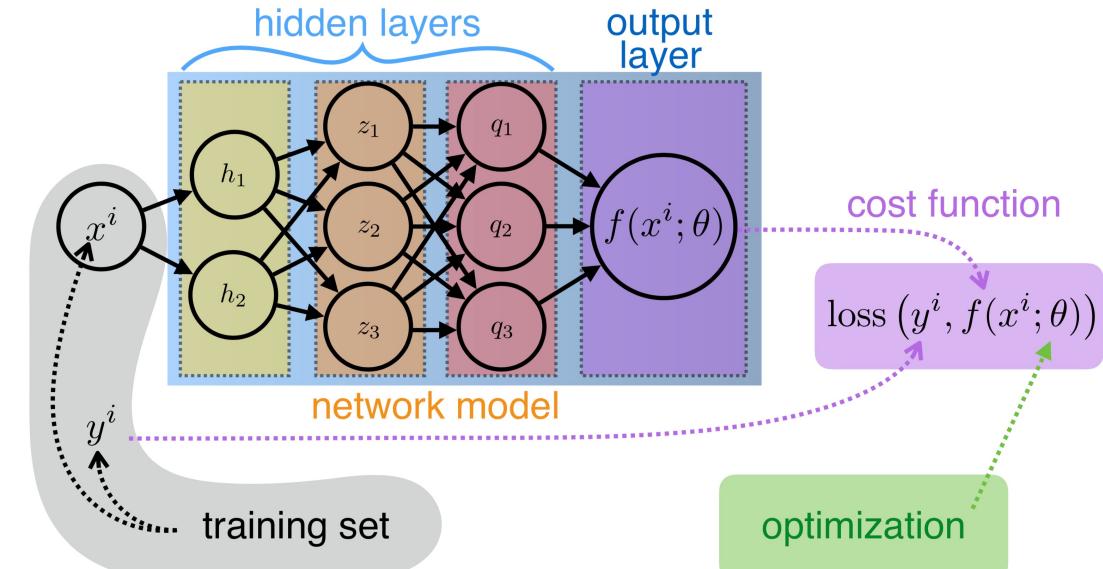
$$J(\theta) = \sum_{i=1}^m \text{loss}(y^i, f(x^i; \theta))$$

- Parameter initialization (weights, biases)

- random weights, biases initialized to small values (0.1)*

- Next, we *optimize the network parameters  $\theta$*  (training)

- In addition, we have to set values for *hyperparameters*



batch size, size and number of layers, choice of optimizer

# Maximum Likelihood

- Given IID input/output samples :  $(x^i, y^i) \sim p_{\text{data}}(x, y)$
- Conditional Maximum Likelihood estimate (between model pdf and data pdf):

$$\begin{aligned}\theta_{\text{ML}} &= \arg \max_{\theta} \prod_{i=1}^m p_{\text{data}}(y^i | x^i; \theta) \\ &= \arg \max_{\theta} \sum_{i=1}^m \log p_{\text{data}}(y^i | x^i; \theta)\end{aligned}$$

product of the probabilities to find all of our datapoints, only works if all the data points are independent

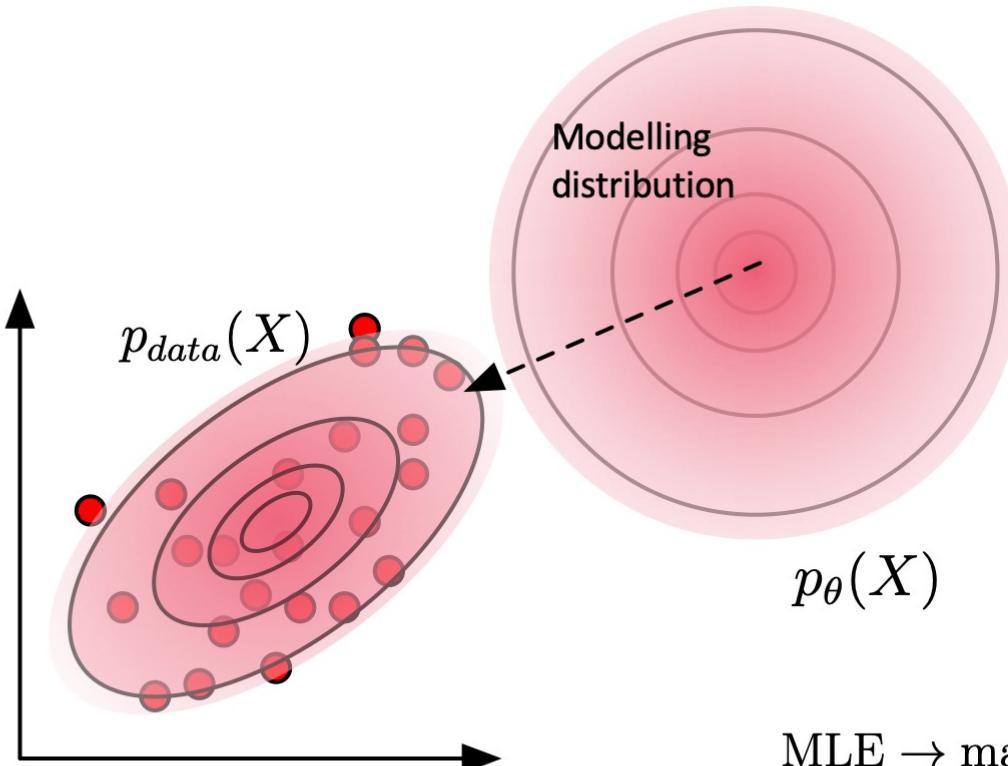
we want to find the theta that maximizes this

$$\min_{\theta} -E_{x,y \sim \hat{p}_{\text{data}}} [\log p_{\text{model}}(y|x; \theta)]$$

minimize negative log-likelihood = maximize log-likelihood

Maximize the likelihood == **Minimize the negative log-likelihood**

# Maximum Likelihood



$$\text{MLE} \rightarrow \max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \log p_\theta(x_i)$$

Fisher 1922

$$\min_{\theta \in \mathcal{M}} KL(P_{\text{data}}, P_\theta) = \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} \left[ \log \frac{p_{\text{data}}(\mathbf{x})}{p_\theta(\mathbf{x})} \right]$$

# Loss function choice

- Choice determined by the **output representation**

- Probability vector (**classification**) : Cross-entropy

similar to logistic regression

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

$$p(y|\hat{y}) = \hat{y}^y(1 - \hat{y})^{(1-y)}$$

$$L(\hat{y}, y) = -\log p(y|\hat{y}) = -(y \log(\hat{y}) + (1 - y)\log(1 - \hat{y}))$$

(**binary classification**)

- Mean estimate (**regression**) : Mean Squared Error, L2 loss

$$\hat{y} = W^\top h + b$$

$$p(y|\hat{y}) = N(y; \hat{y})$$

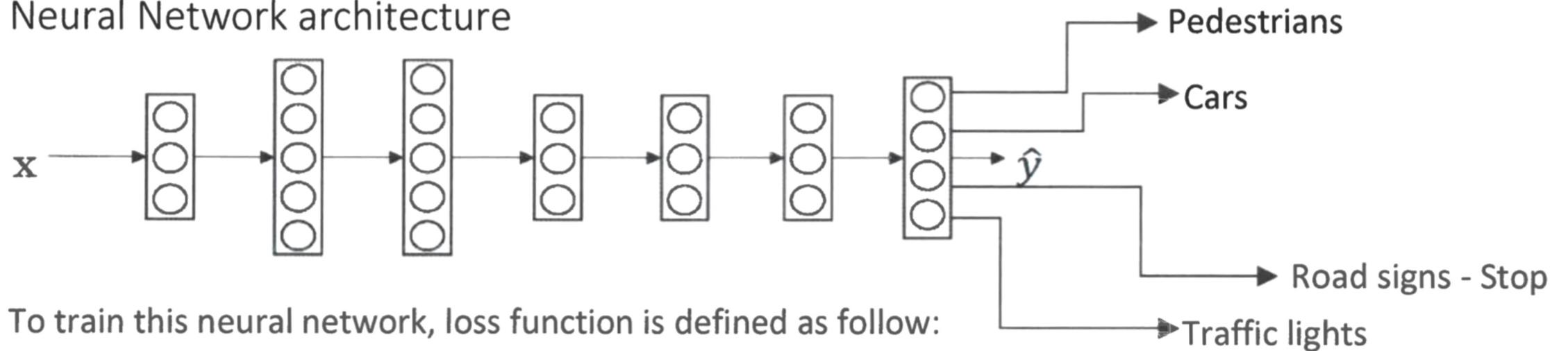
$$L_2(\hat{y}, y) = -\log p(y|\hat{y}) = \sum_{i=0}^m (y^i - \hat{y}^i)^2$$

# Loss function example

- NN does **simultaneously several tasks (multi-task)**



Neural Network architecture

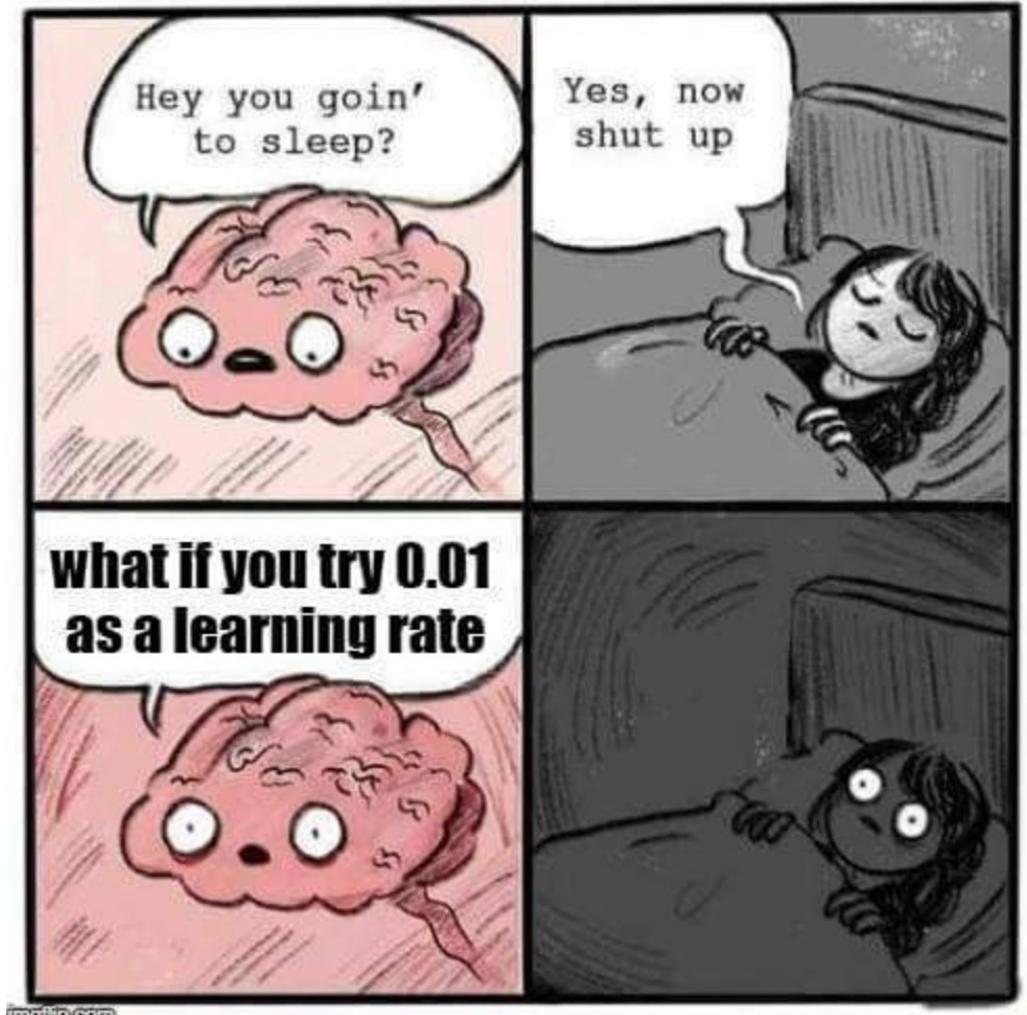


To train this neural network, loss function is defined as follow:

$$-\frac{1}{m} \sum_{i=1}^m \left[ \sum_{j=1}^4 \left( y_j^{(i)} \log(\hat{y}_j^{(i)}) + (1 - y_j^{(i)}) \log(1 - \hat{y}_j^{(i)}) \right) \right]$$

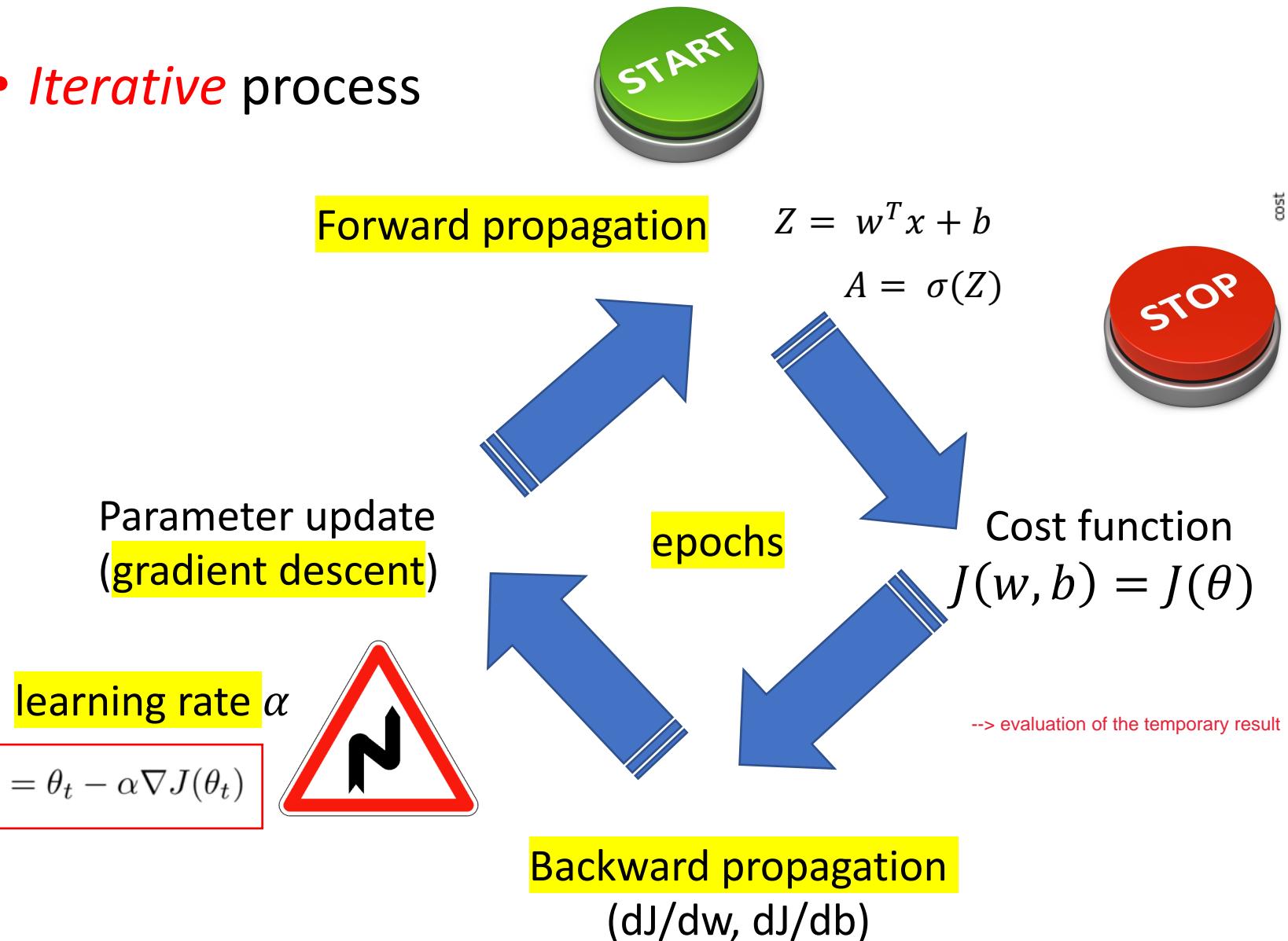
# Hyperparameters

- Parameters that **cannot** be learnt directly from training data
- A long list...
  - Learning rate  $\alpha$
  - Number of iterations (**epochs**)
  - Number of hidden layers
  - Number of hidden units
  - Choice of activation function
  - *More to come !*

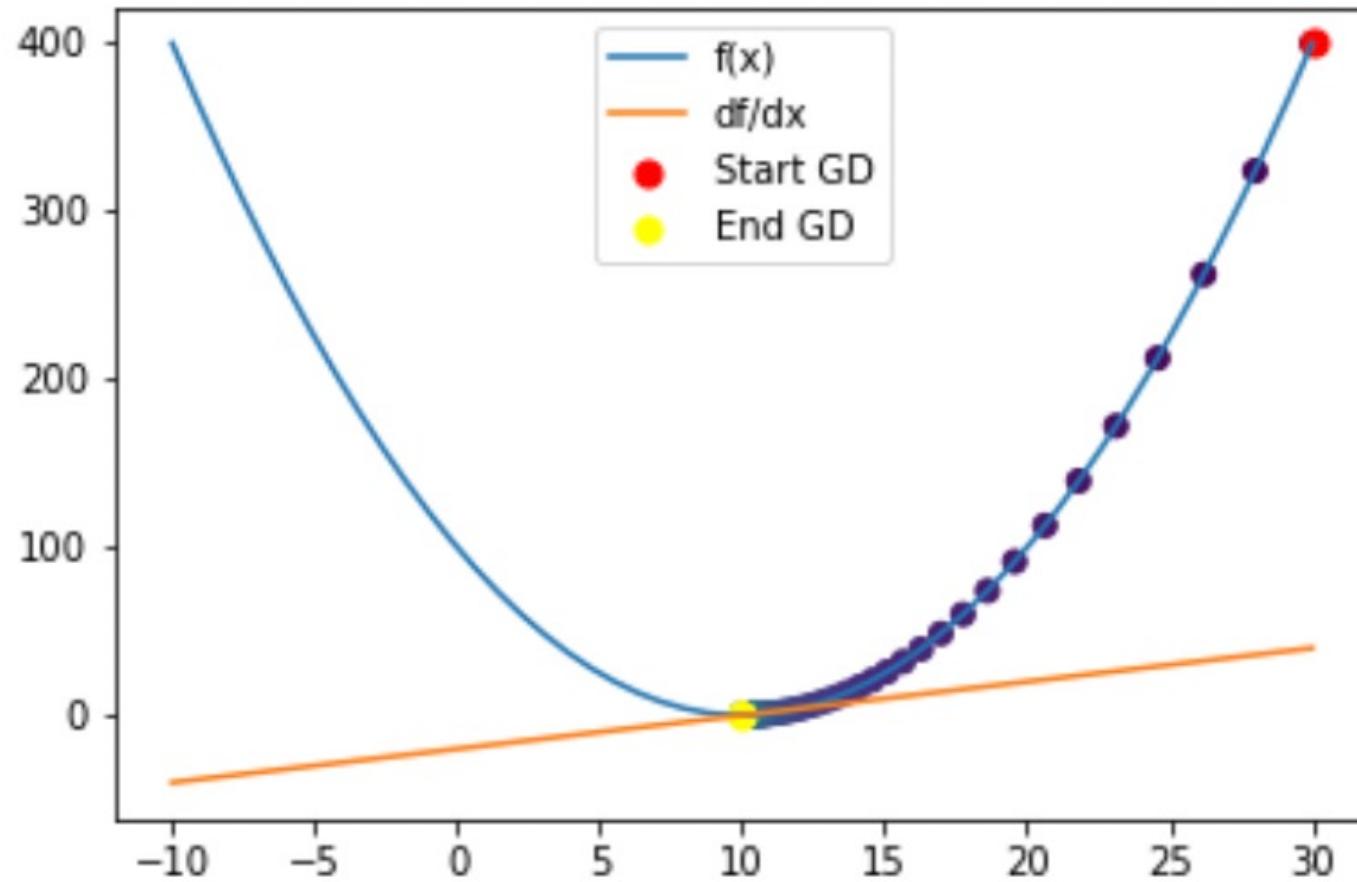


# Training

- *Iterative* process



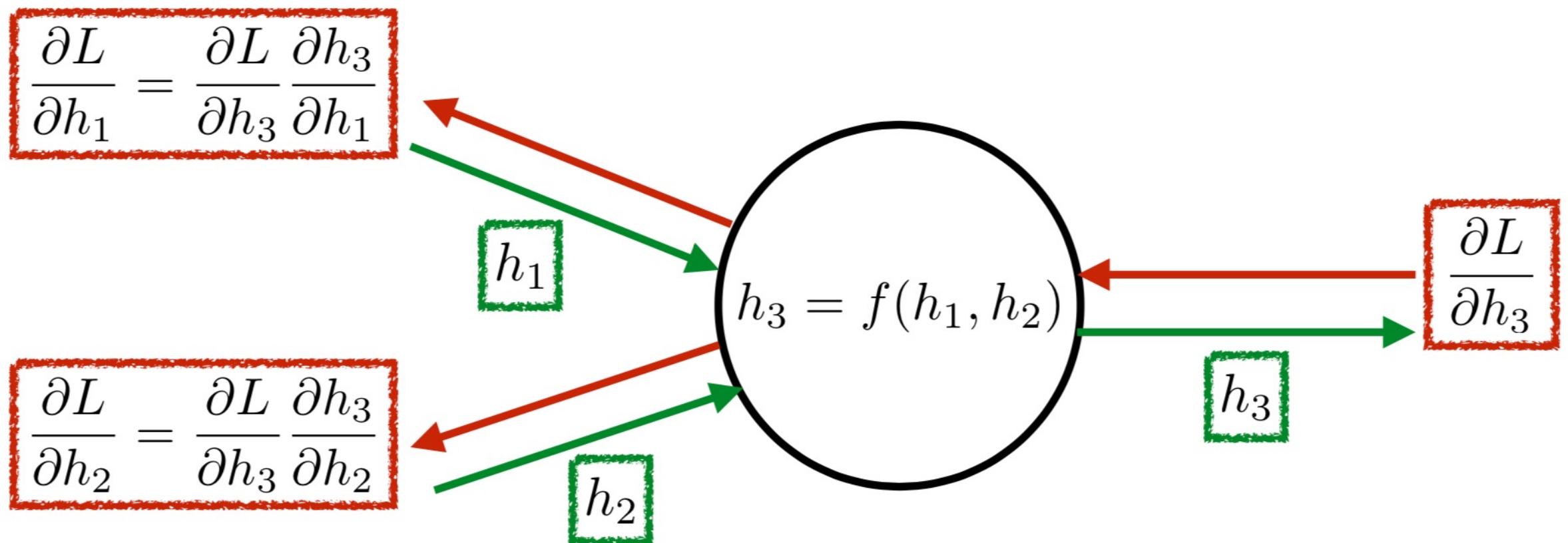
## Gradient descent quadratic function



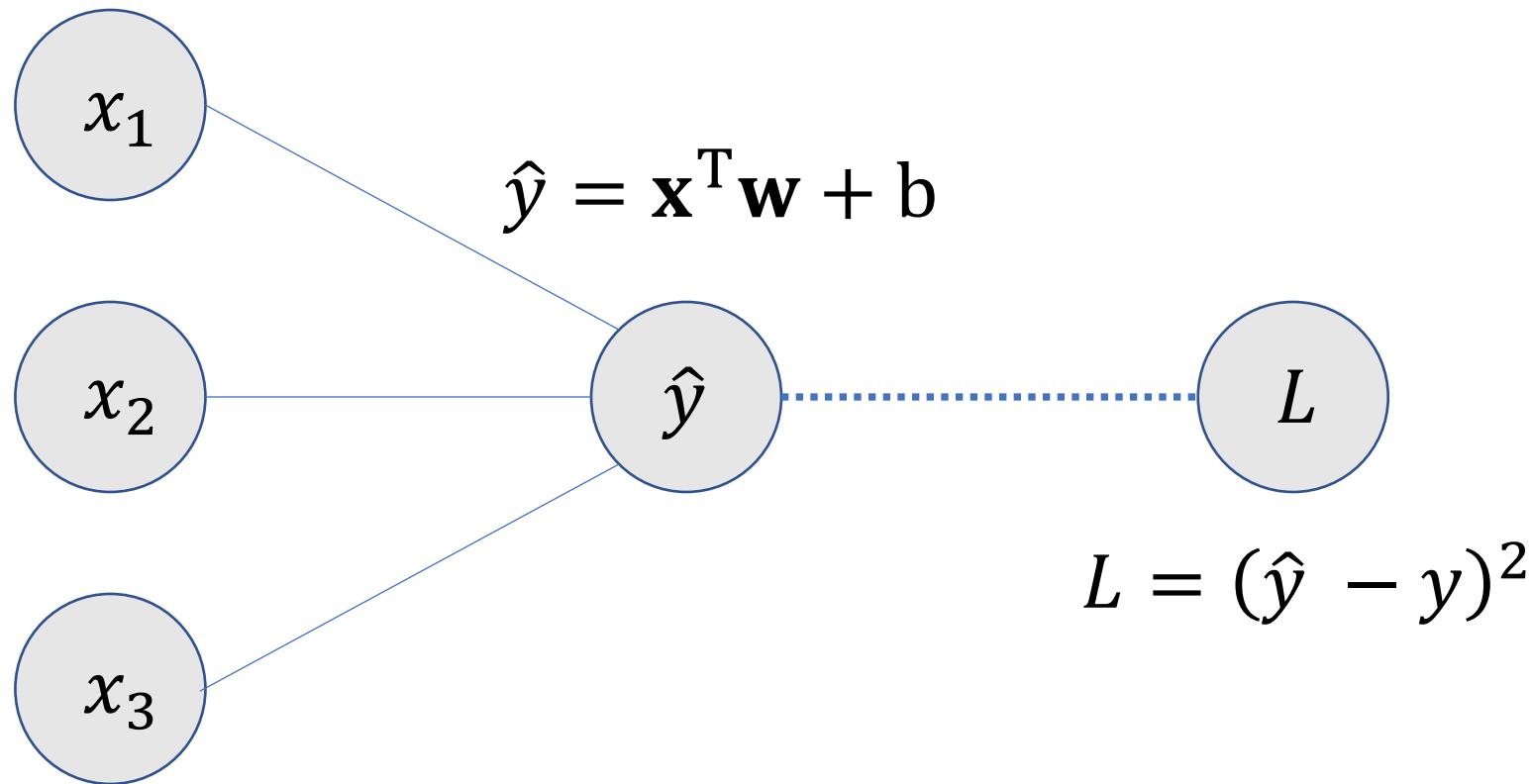
we want to find the  $x$  that minimizes  $f(x)$

# Backpropagation

- Efficient implementation of the **chain-rule** to compute derivatives with respect to network weights



# Example



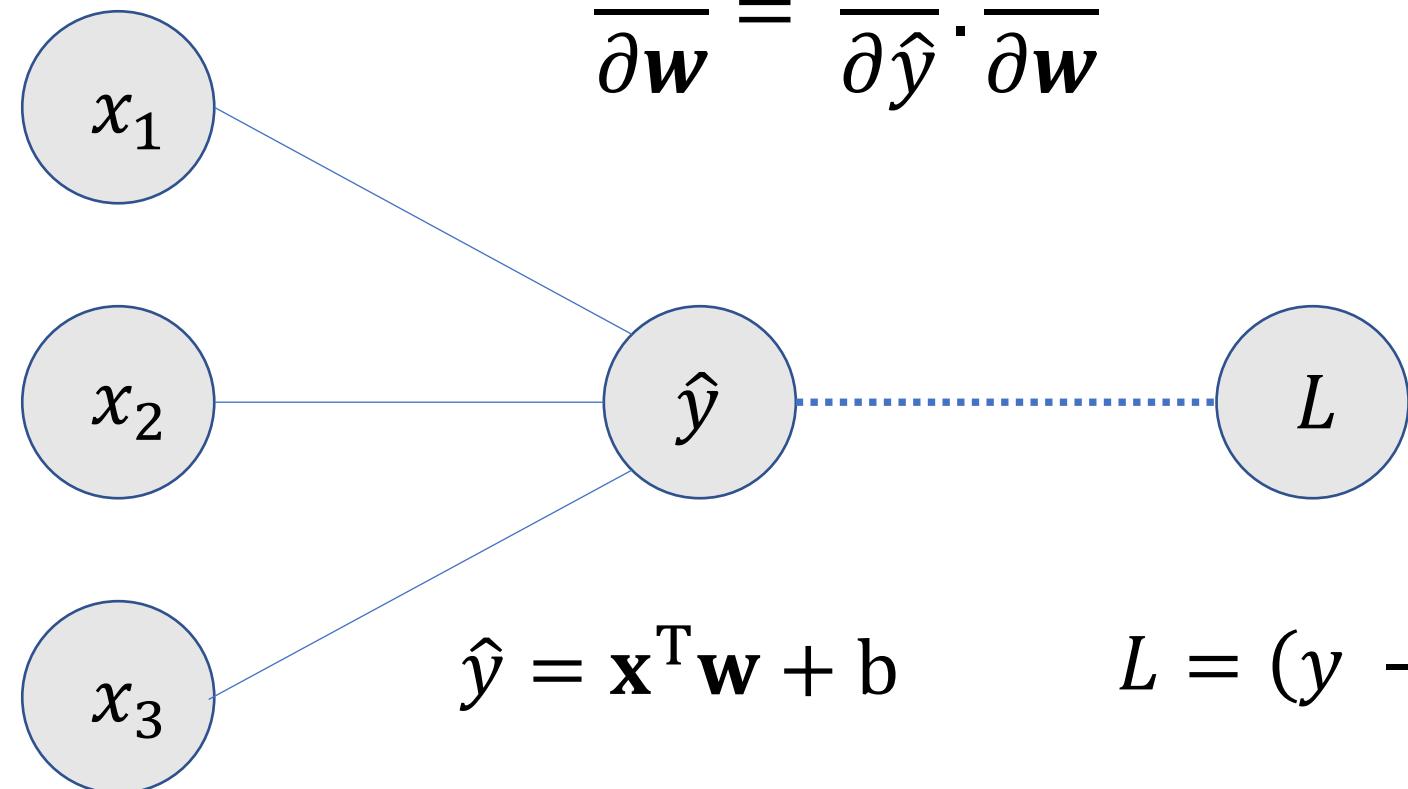
# Example

We need to calculate the gradients:

Let's start with this part!

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b}$$

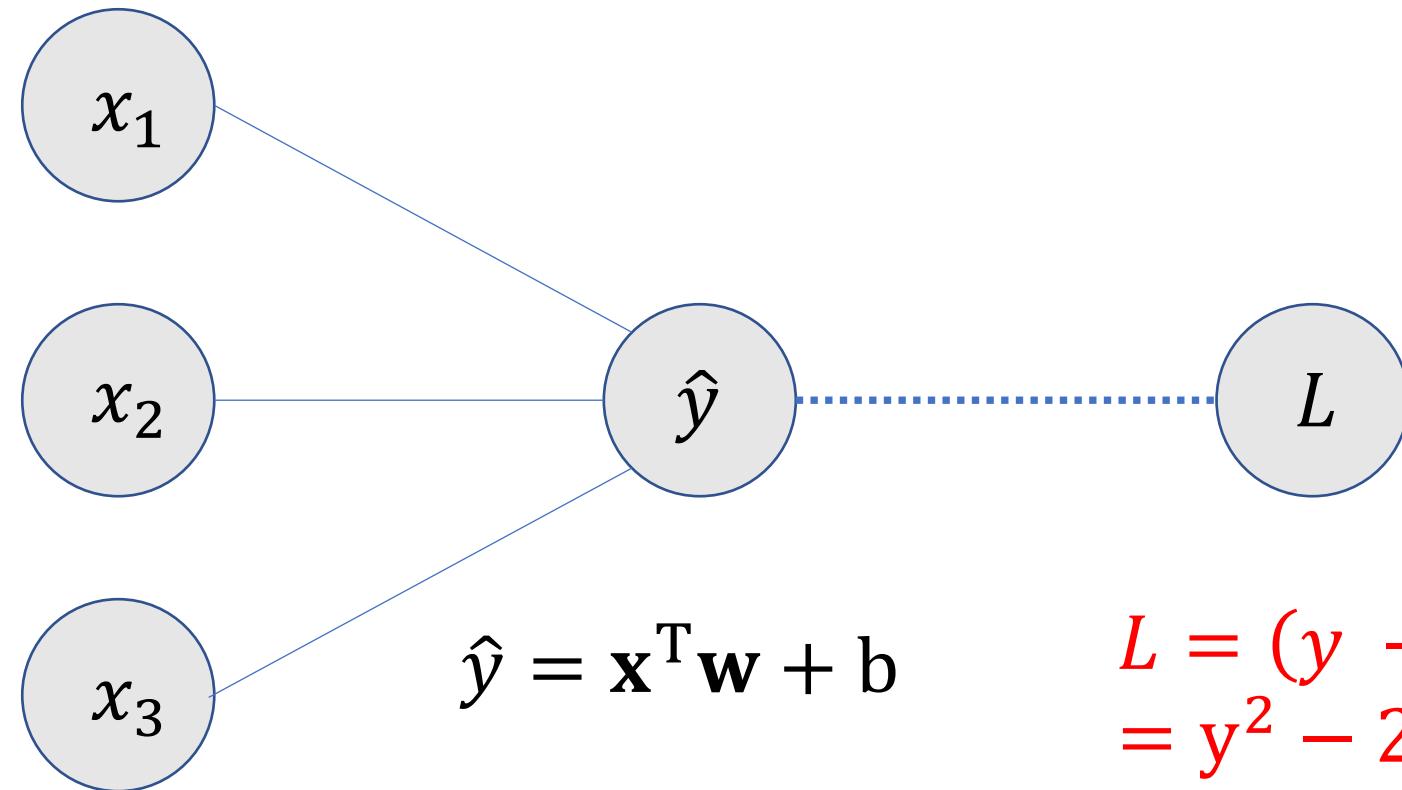


$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

# Example

First:

$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - y)$$



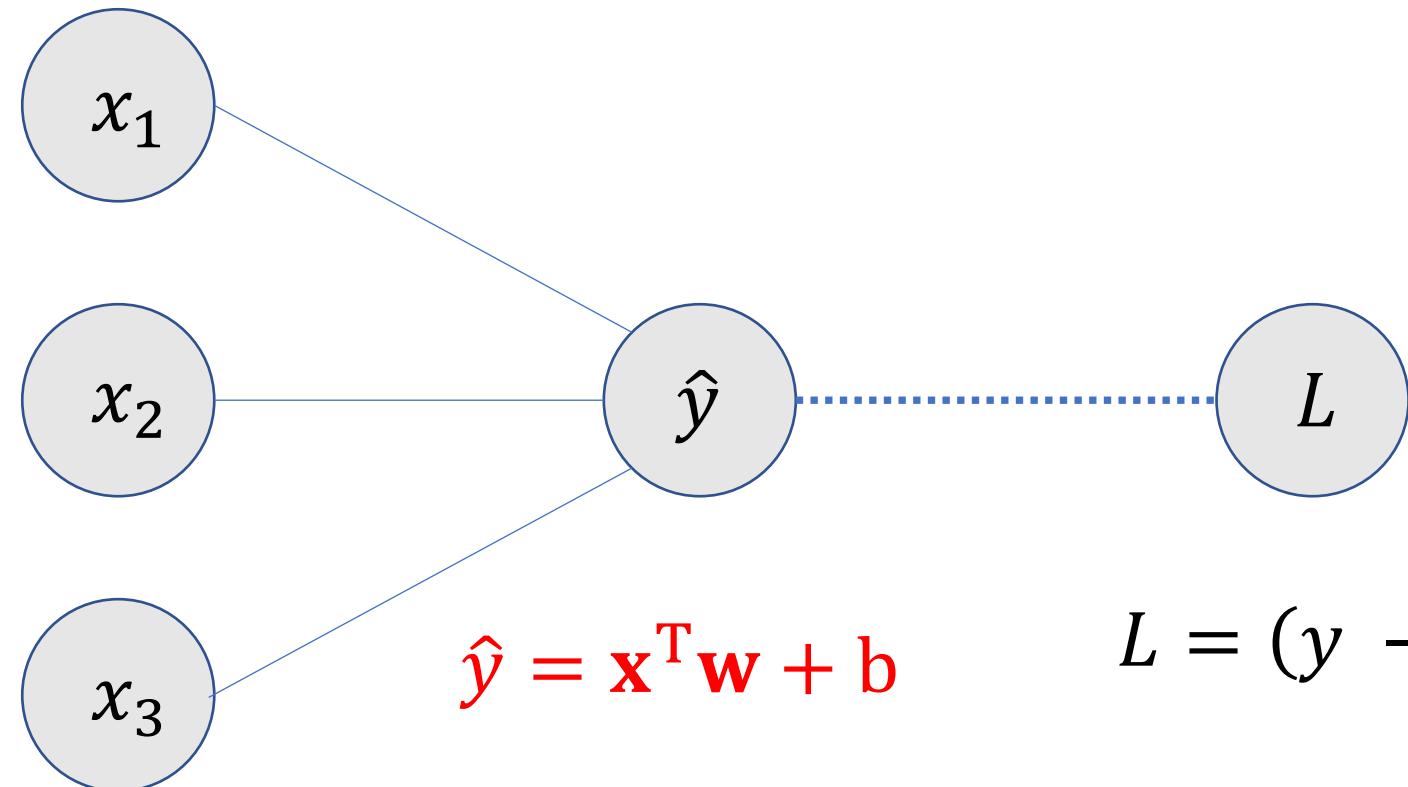
$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - y)$$

# Example

Second:

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{x}^T \mathbf{w} + b) = \mathbf{x}^T \cdot \frac{\partial}{\partial \mathbf{w}} (\mathbf{w}) = \mathbf{x}^T$$



$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

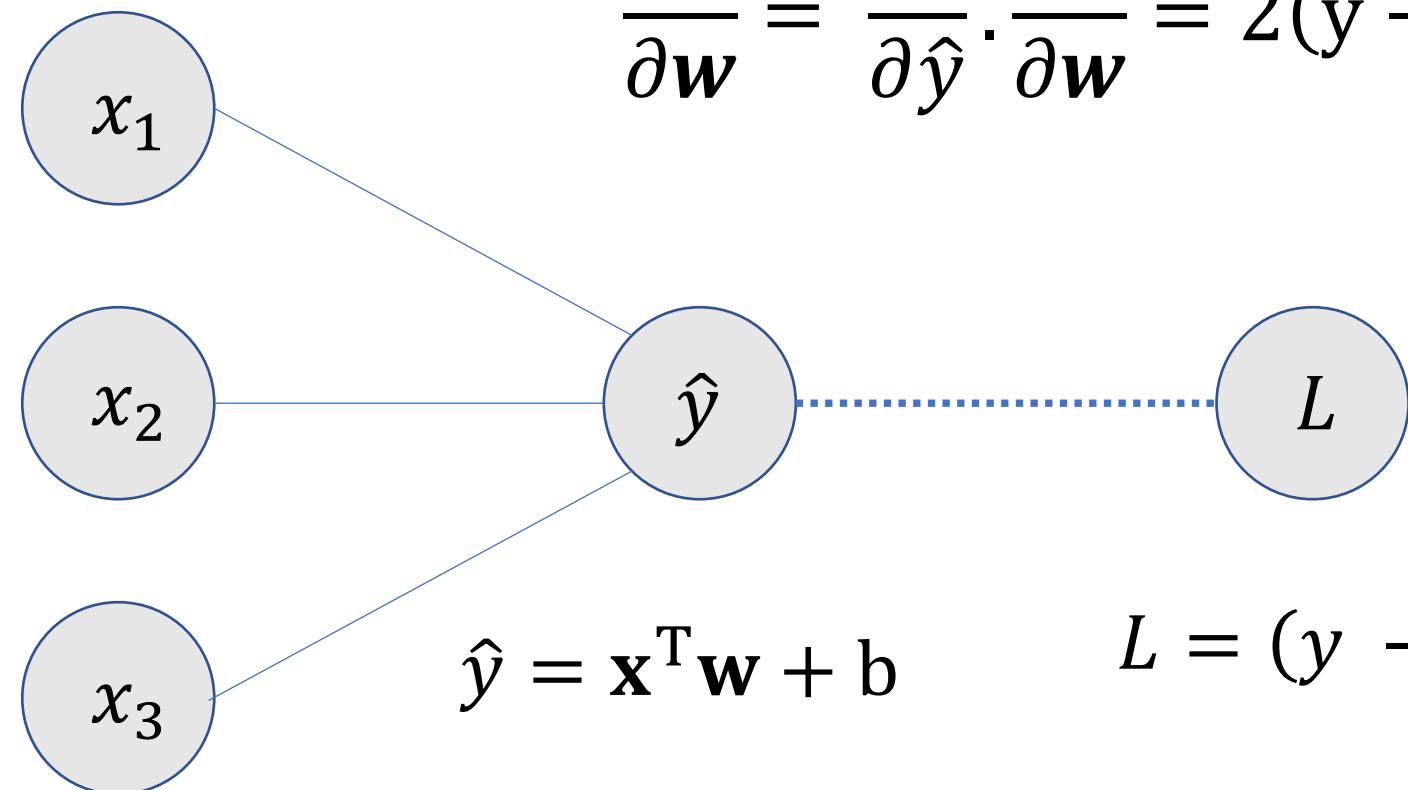
$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - y)$$

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{x}^T \mathbf{w} + b) = \mathbf{x}^T$$

# Example

Putting these together:

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}} = 2(y - \hat{y}) \cdot \mathbf{x}^T$$



$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

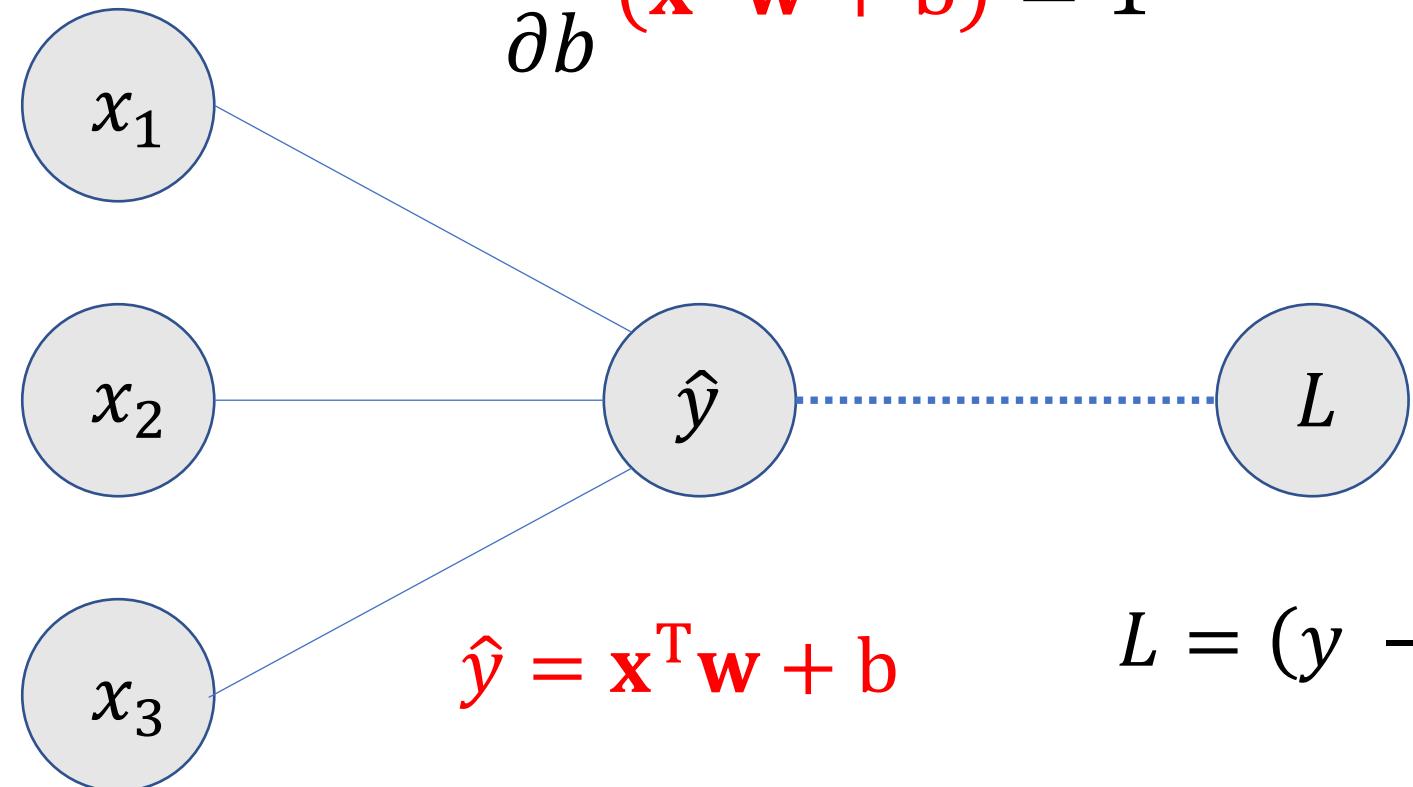
$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - y)$$

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{x}^T \mathbf{w} + b) = \mathbf{x}^T$$

# Example

Now for the bias...

$$\frac{\partial}{\partial b} (\mathbf{x}^T \mathbf{w} + b) = 1$$



$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - y)$$

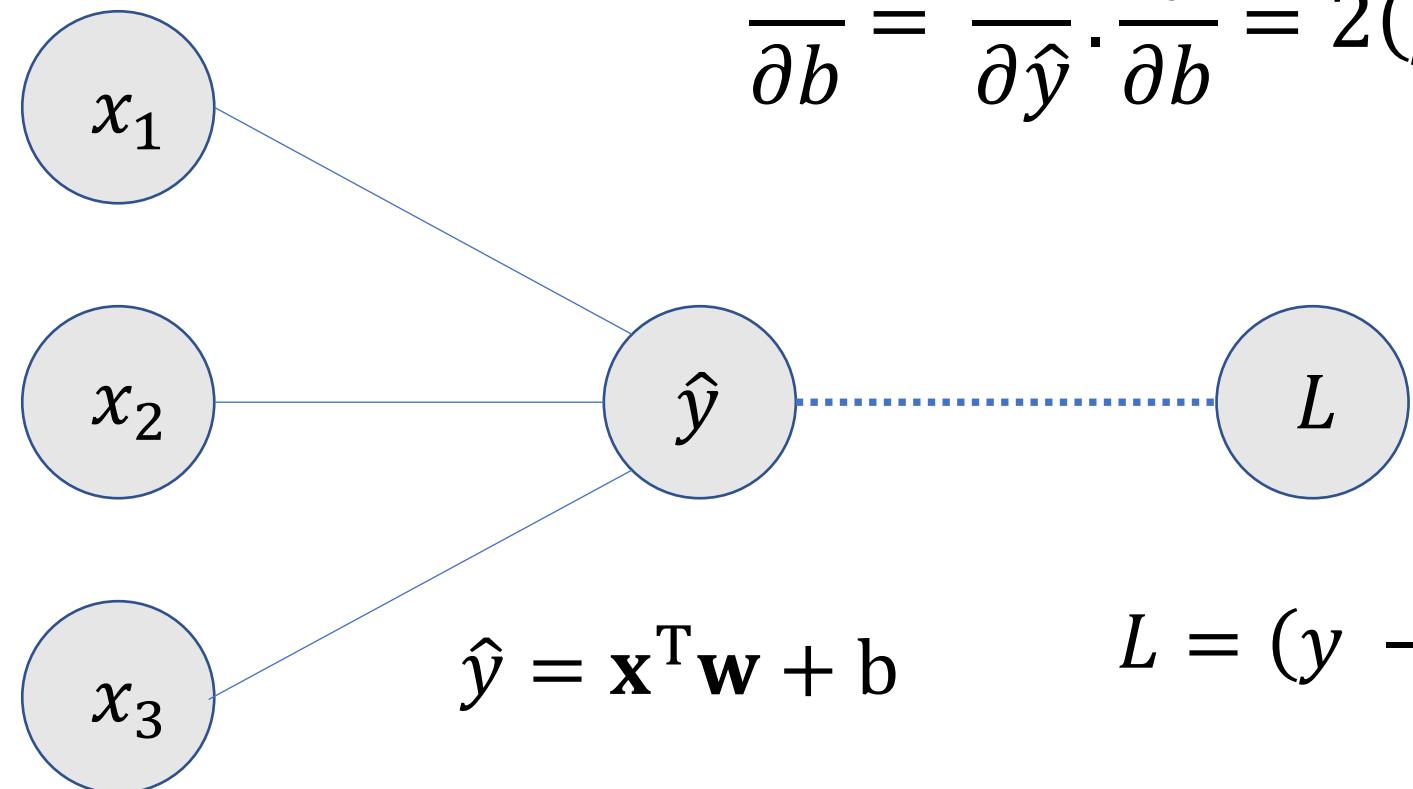
$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{x}^T \mathbf{w} + b) = \mathbf{x}^T$$

$$\frac{\partial}{\partial b} (\mathbf{x}^T \mathbf{w} + b) = 1$$

# Example

Putting these together:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b} = 2(\hat{y} - y) \cdot 1$$



$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - y)$$

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{x}^T \mathbf{w} + b) = \mathbf{x}^T$$

$$\frac{\partial}{\partial b} (\mathbf{x}^T \mathbf{w} + b) = 1$$

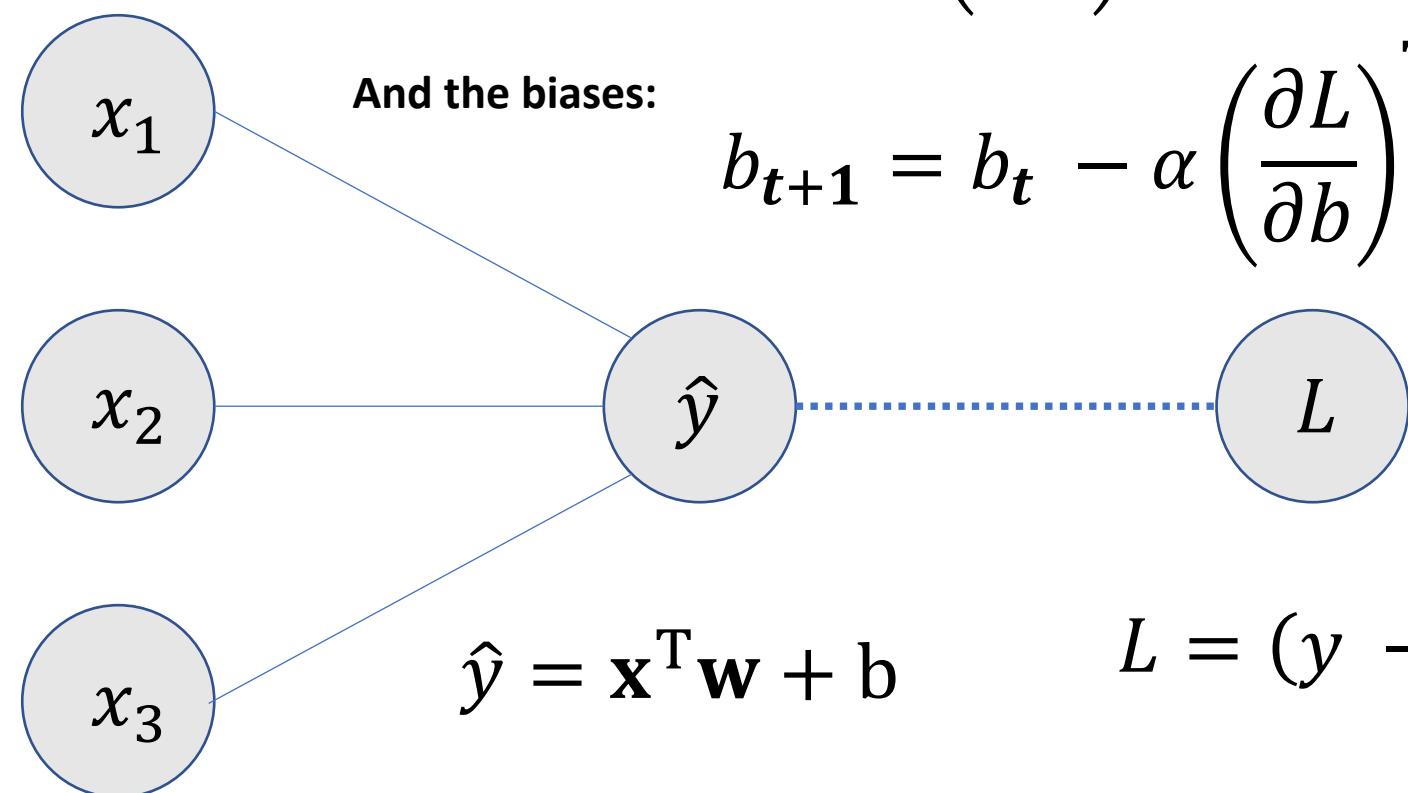
# Example

Finally the updates for the weights:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \left( \frac{\partial L}{\partial \mathbf{w}} \right)^T = \mathbf{w}_t - 2\alpha(\hat{y} - y)\mathbf{x}$$

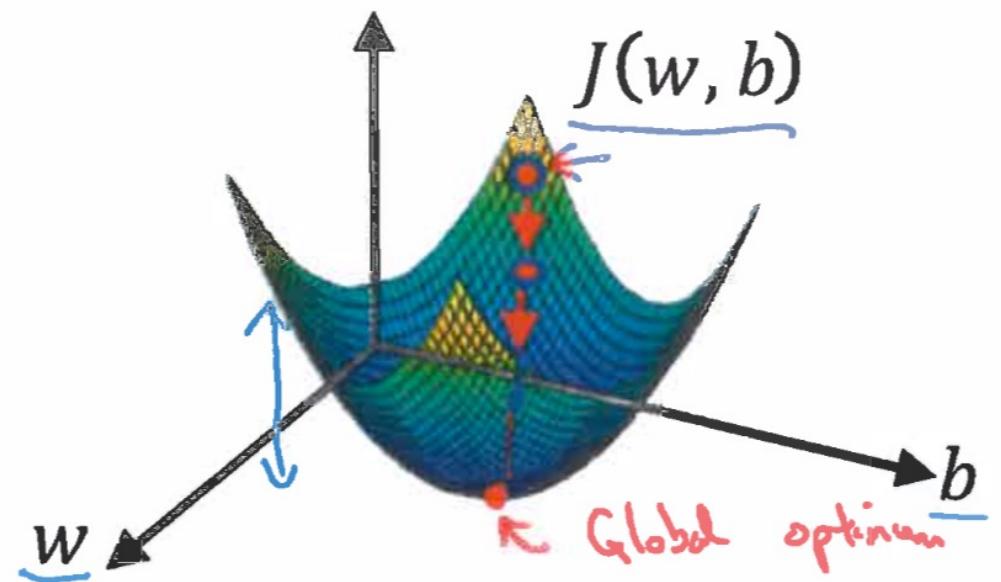
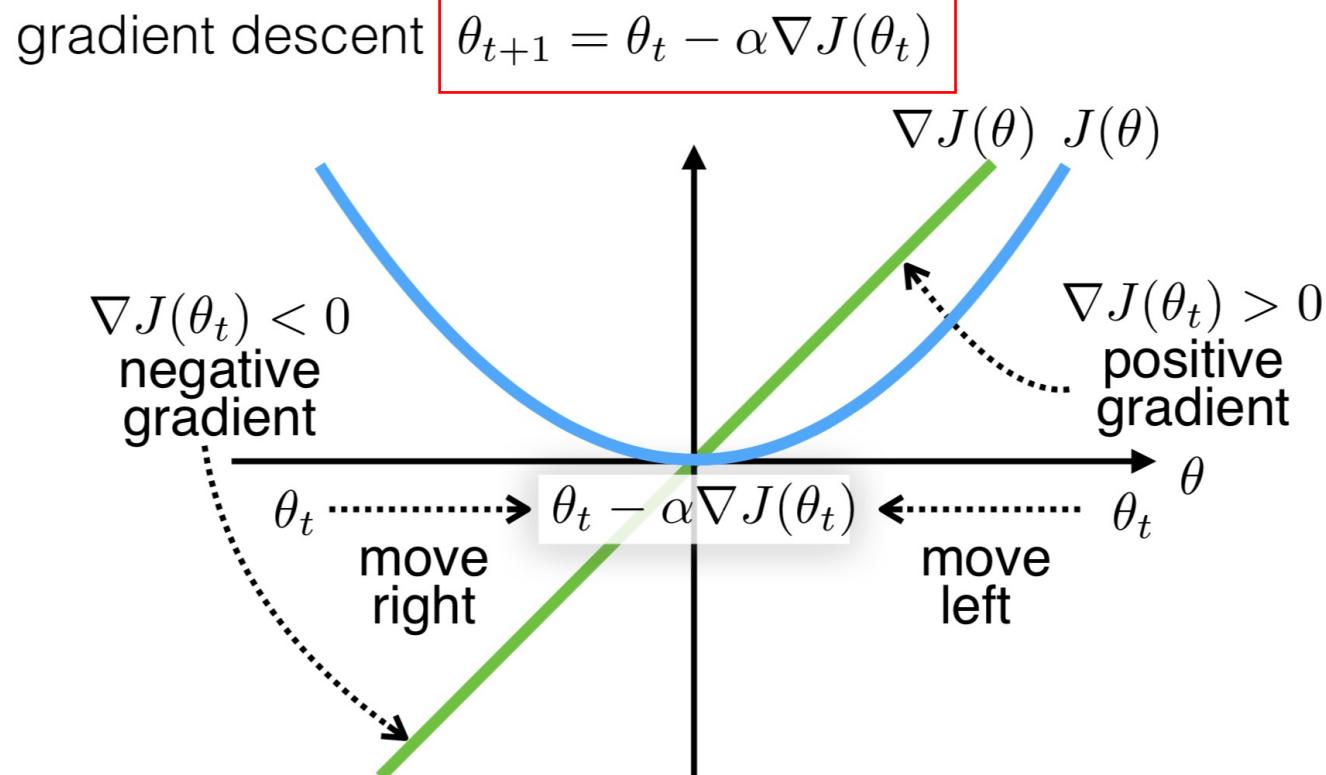
And the biases:

$$b_{t+1} = b_t - \alpha \left( \frac{\partial L}{\partial b} \right)^T = b_t - 2\alpha(\hat{y} - y)$$



# Gradient Descent

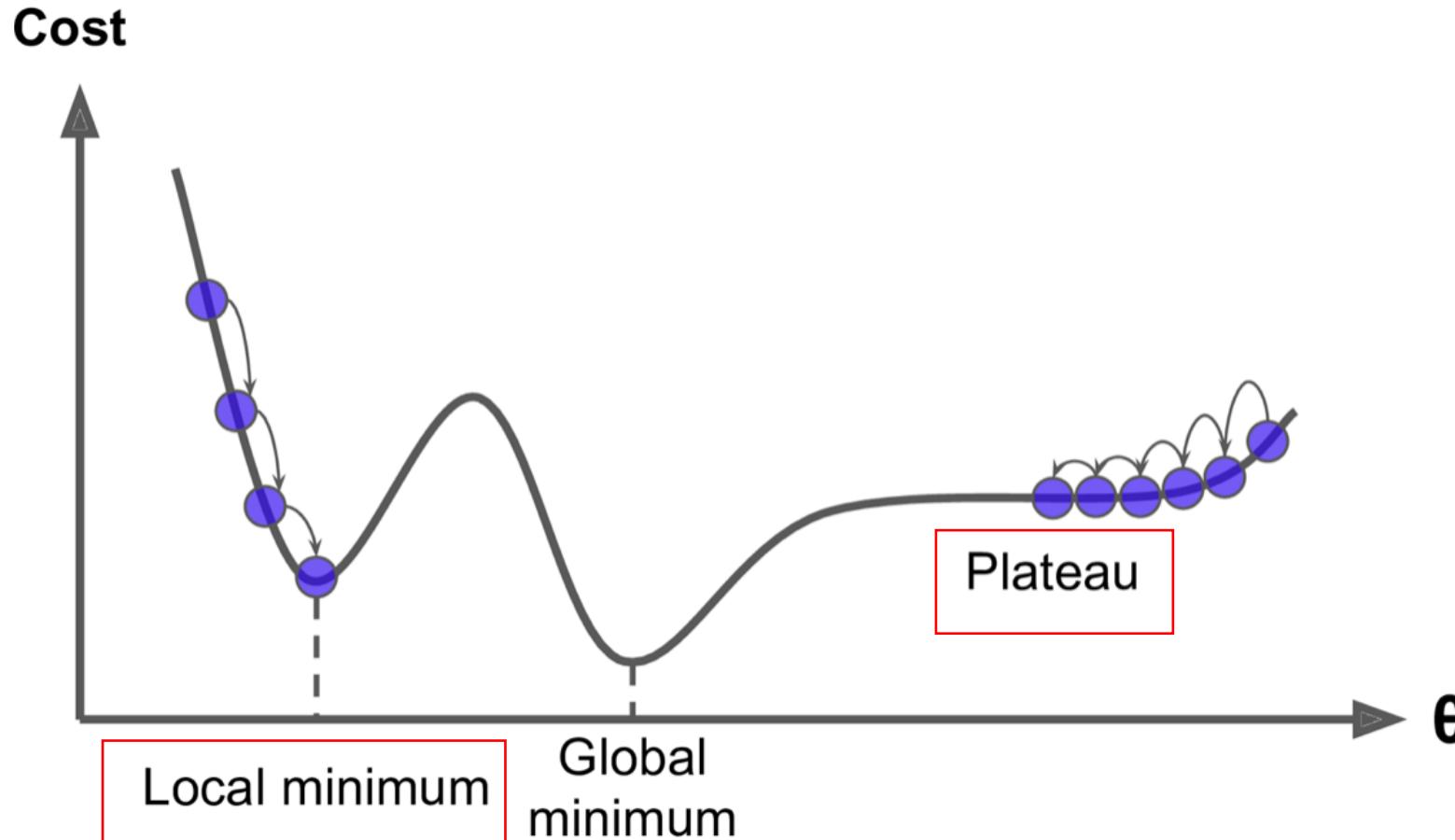
- Iterative method to find the parameters  $\theta = (w, b)$  that minimize  $J(\theta)$



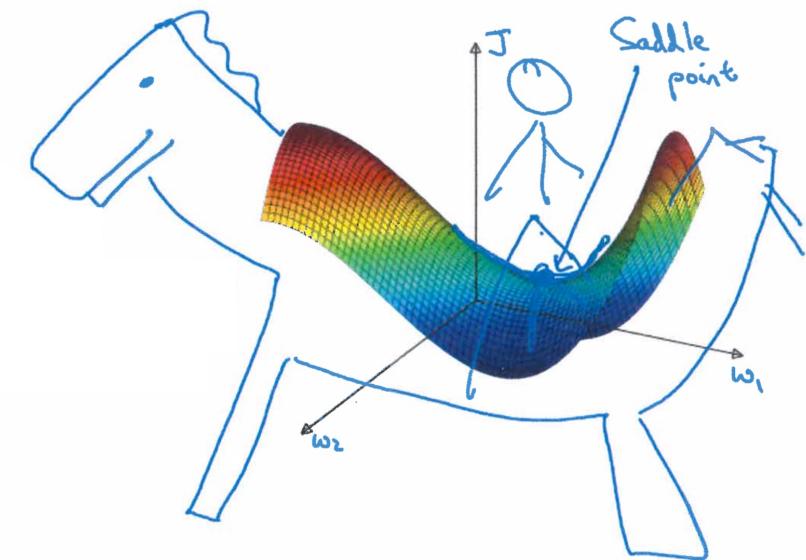
$$\nabla J(w) = \frac{dJ(w, b)}{dw}$$

$$\nabla J(b) = \frac{dJ(w, b)}{db}$$

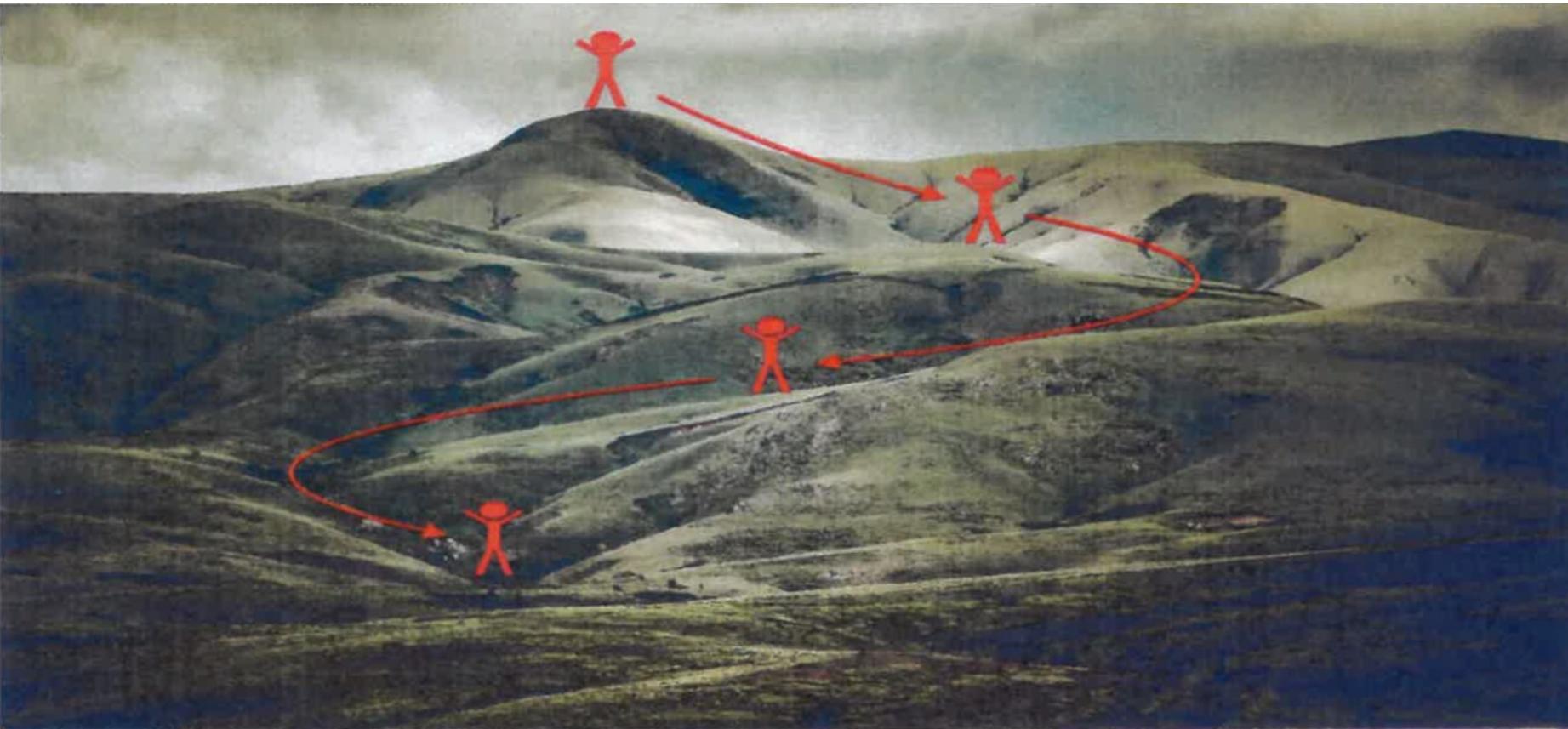
# Optimization pitfalls



Saddle point



# Gradient Descent Illustration



# Tutorial / Practical

