Deep Forward Networks - Introduction

Wednesday 08h15 – 09h00

Géraldine Conti, Matthew Vowels, Mykhailo Vladymyrov
Bern Winter School on Machine Learning 2024, Mürren

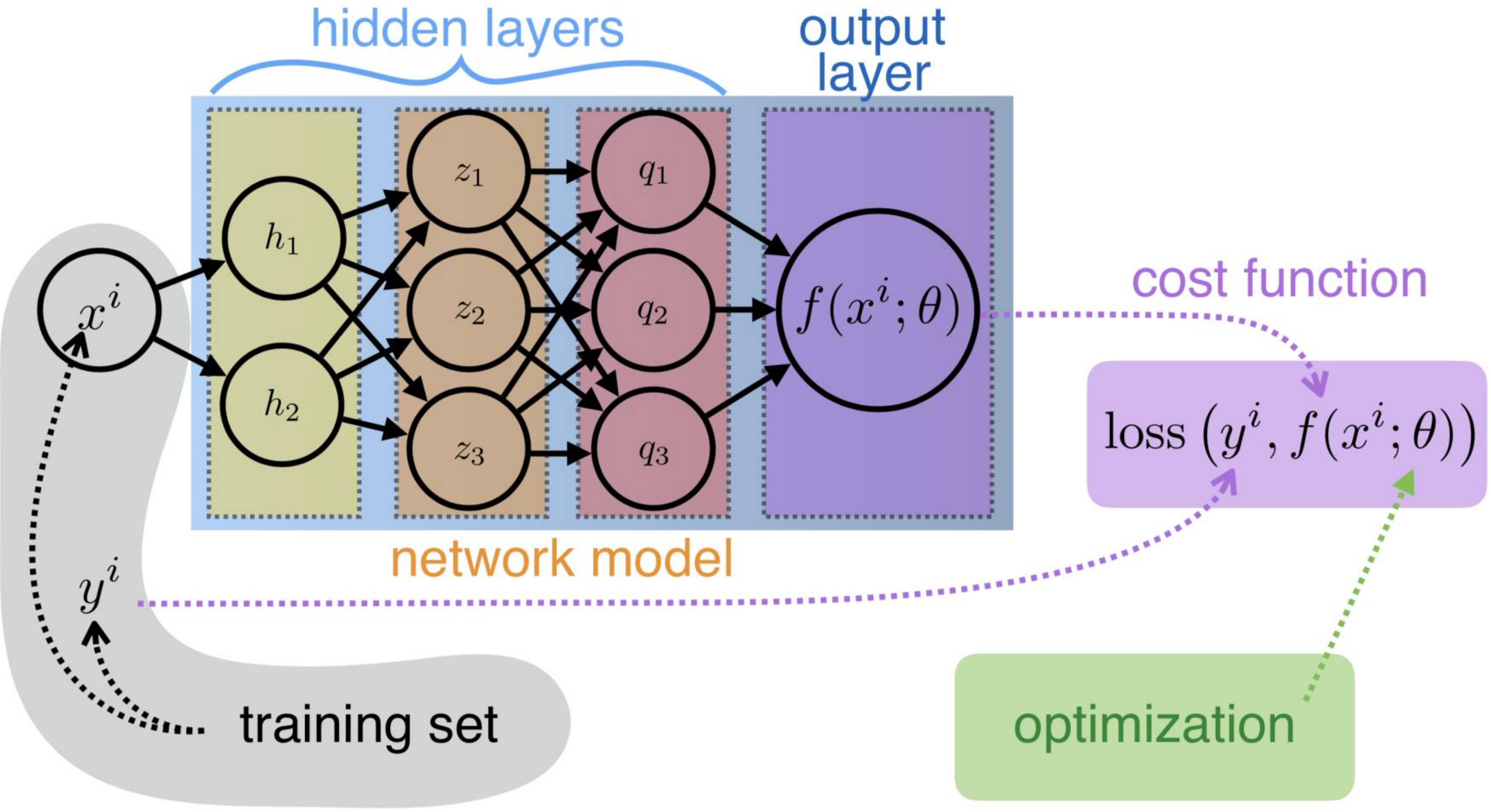
Deploying a Neural Network

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

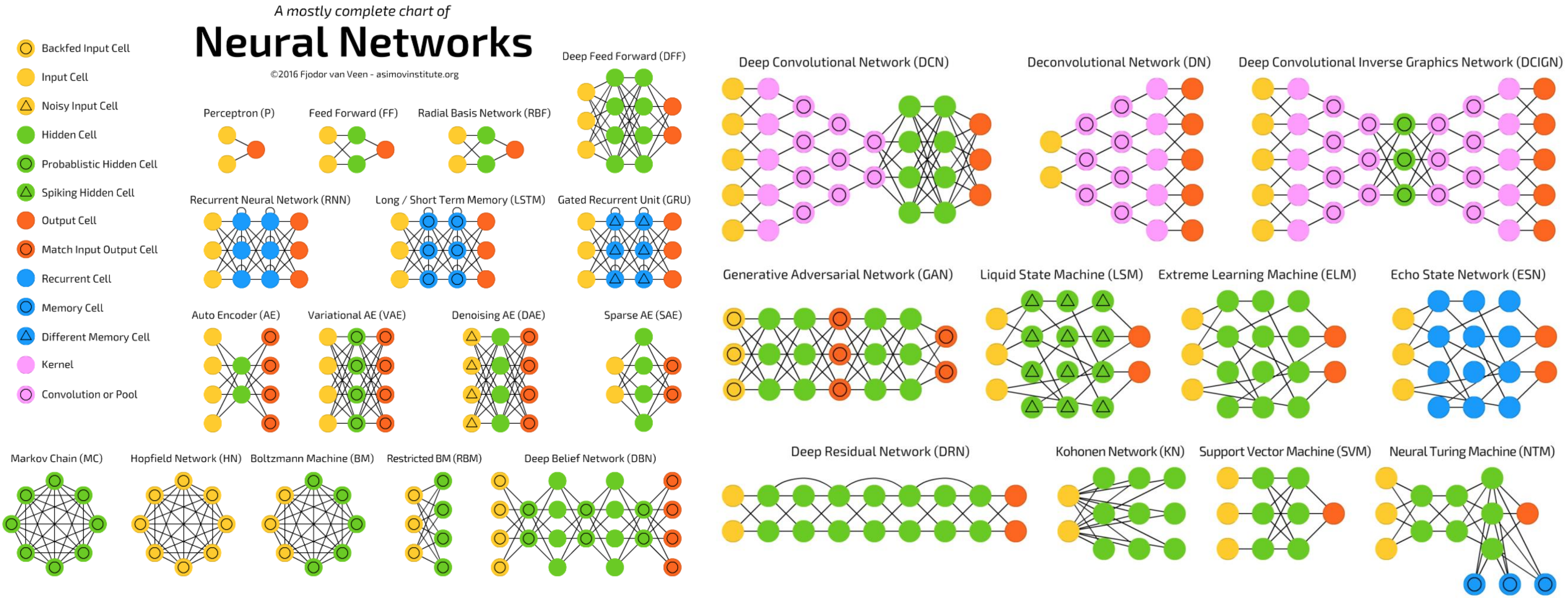
1) Network model

2) Cost function

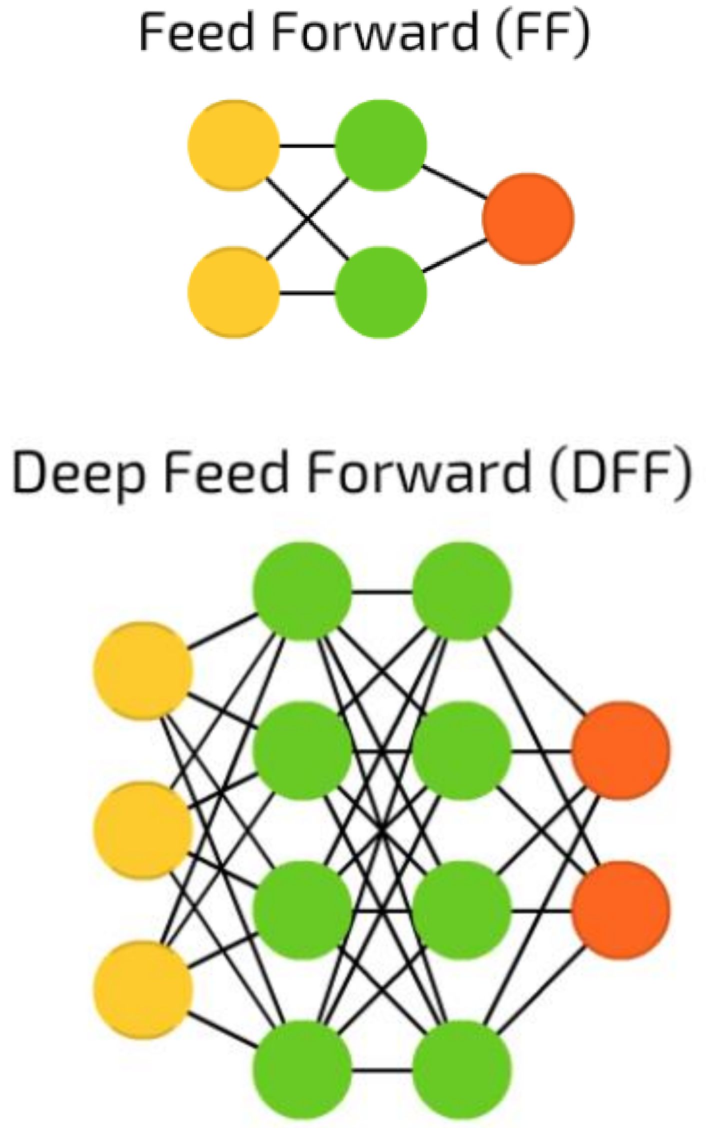
3) Optimization



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**"

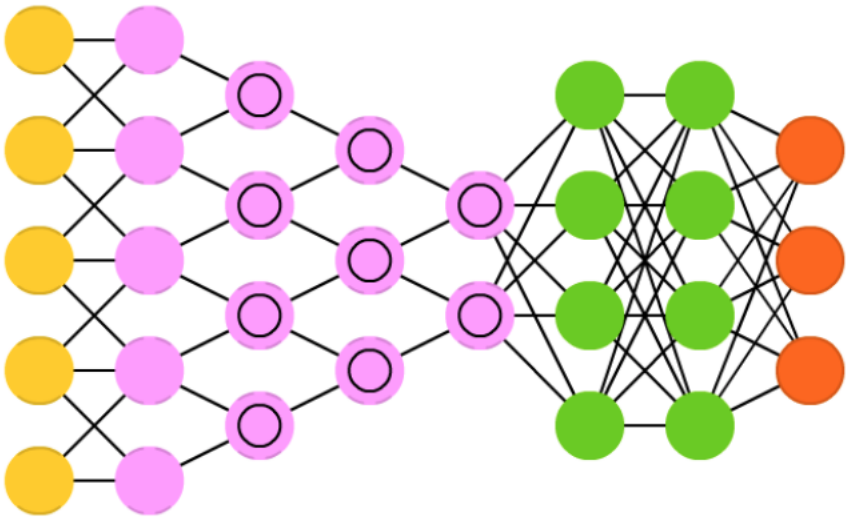
Convolutional Neural Networks (CNN)

- 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*

- well suited for computer vision tasks
  - Image classification
  - Object detection

Deep Convolutional Network (DCN)

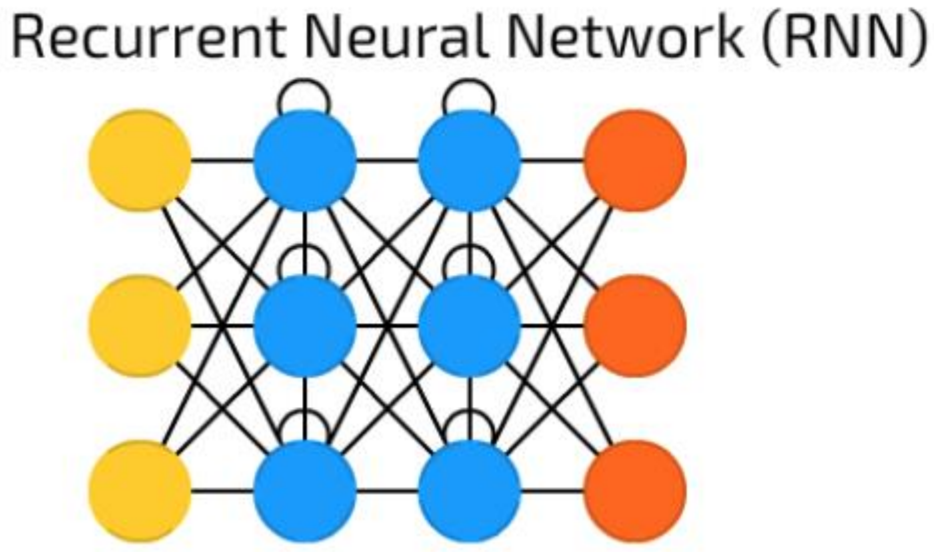


Recurrent Neural Networks (RNN)

- connections between neurons include loops

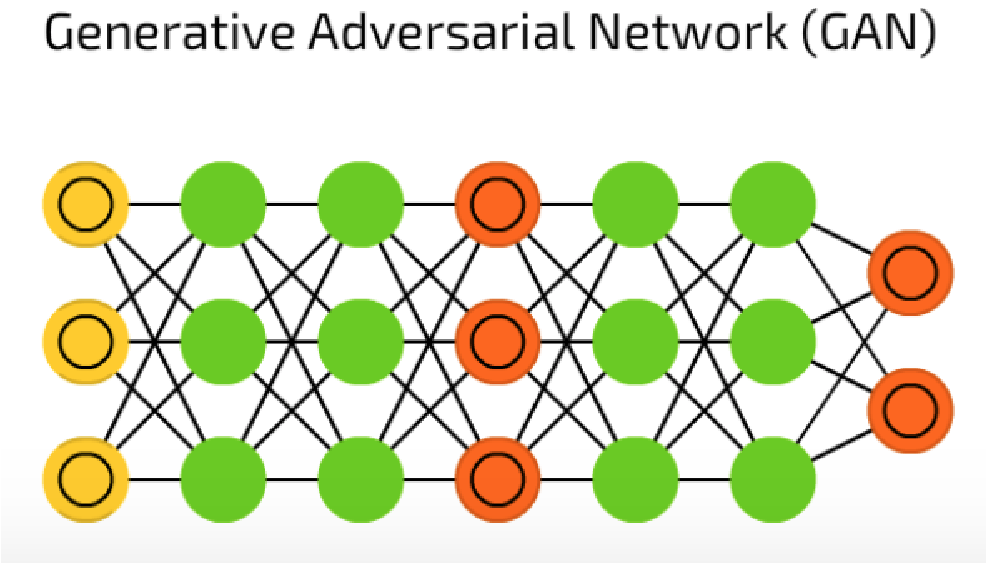
- **Recurrent cells** (or memory cells) used
  - Weight sharing between *time-steps*

- well-suited for processing sequences of inputs, when context is important
  - Text analysis



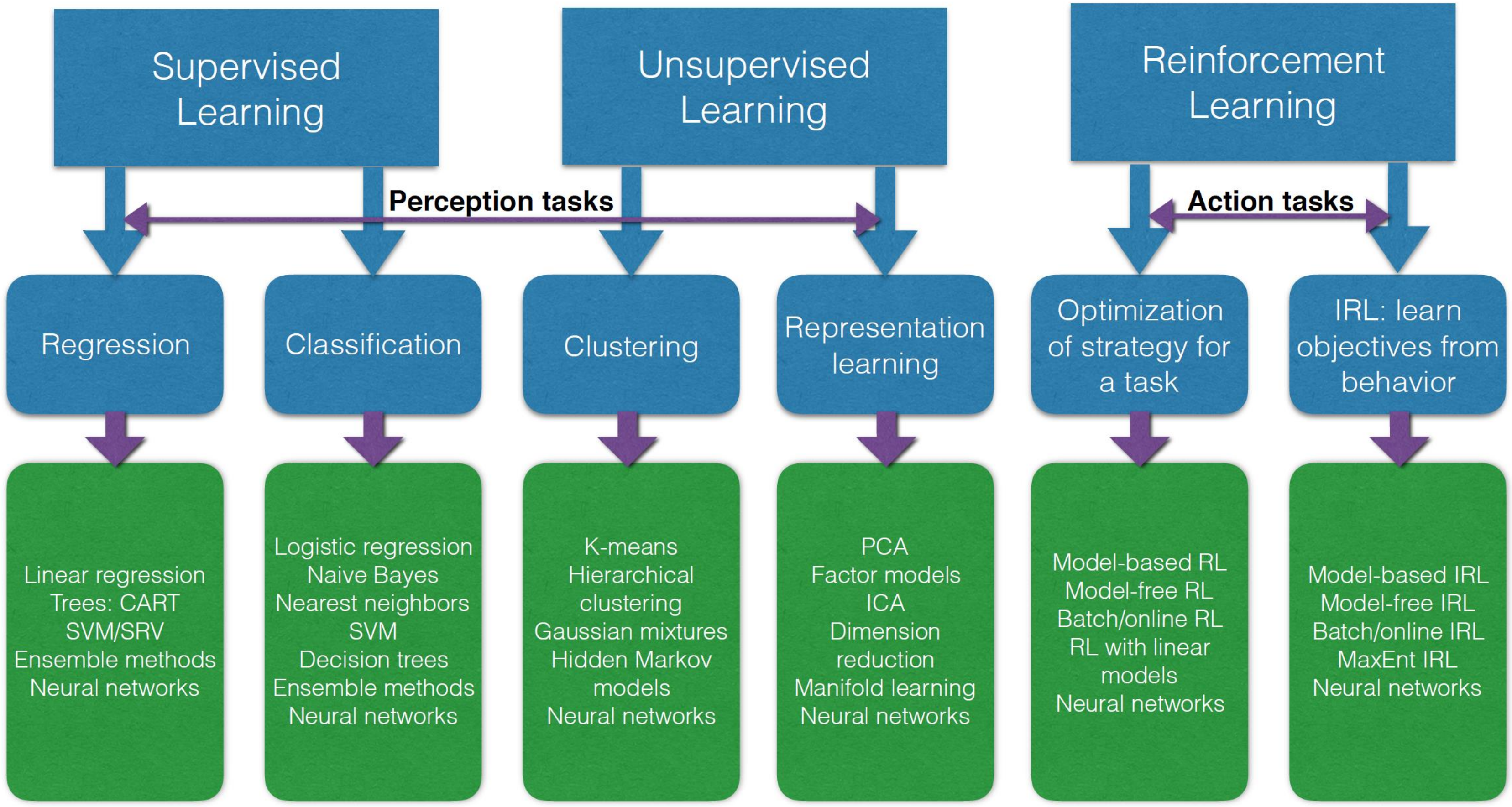
Generative Adversarial Networks (GAN)

- 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



Can be hard to train

Use cases



Supervised Learning
Unsupervised Learning
Reinforcement Learning
Perception tasks
Action tasks
Regression
Classification
Clustering
Representation learning
Optimization of strategy for a task
IRL: learn objectives from behavior
Linear regression
Trees: CART
SVM/SRV
Ensemble methods
Neural networks
Logistic regression
Naive Bayes
Nearest neighbors
SVM
Decision trees
Ensemble methods
Neural networks
K-means
Hierarchical clustering
Gaussian mixtures
Hidden Markov models
Neural networks
PCA
Factor models
ICA
Dimension reduction
Manifold learning
Neural networks
Model-based RL
Model-free RL
Batch/online RL
RL with linear models
Neural networks
Model-based IRL
Model-free IRL
Batch/online IRL
MaxEnt IRL
Neural networks

2) Loss and Cost functions

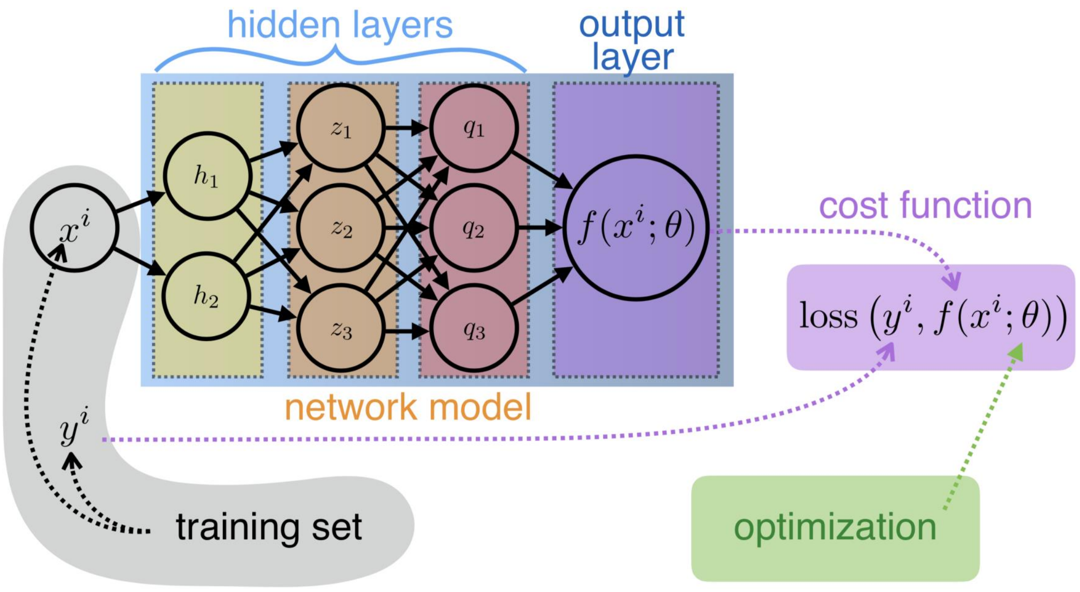
- Loss function $L\left(\hat{y}^{(i)}, y^{(i)}\right)$, also called error function, measures how different the prediction $\hat{y} = f(x)$ and the desired output y are

- 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

3) Optimization



- Given a task we define
  - Training data $\{x^i, y^i\}_{i=1,\dots,m}$
  - Network $f(x;\theta)$
  - Cost function $J(\theta) = \sum_{i=1}^{m} \text{loss}\left(y^i, f(x^i;\theta)\right)$
  - Parameter initialization (weights, biases)
    - *random weights*, *biases initialized to small values (0.1)*

      they are initialized at random but from a specific distribution
- Next, we *optimize the network parameters θ* (training)
- In addition, we have to set values for hyperparameters

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):

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

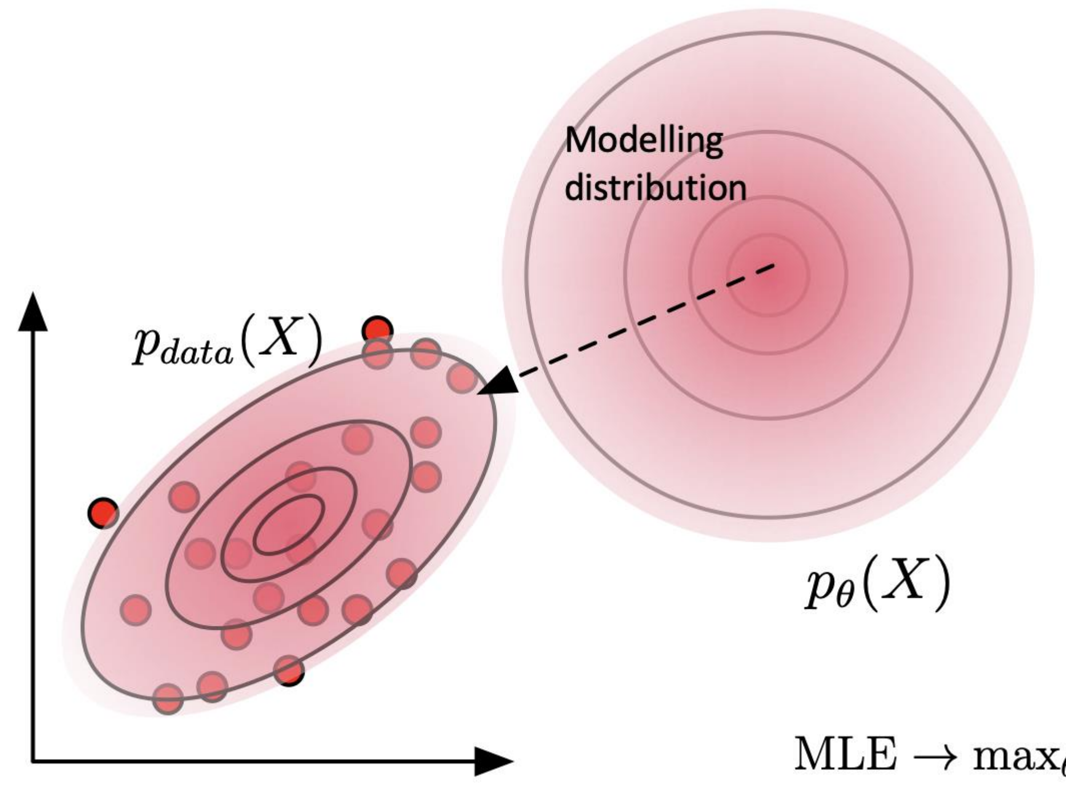
- Mathematical tricks :

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log p_{\text{data}}(y^i | x^i; \theta)$$

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

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

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

Loss function choice

- Choice determined by the output representation
  - Probability vector (**classification**) : Cross-entropy

$$\hat{y} = \sigma(w^\top h + b) \qquad\qquad p(y|\hat{y}) = \hat{y}^y(1-\hat{y})^{(1-y)}$$

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

(binary classification)

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

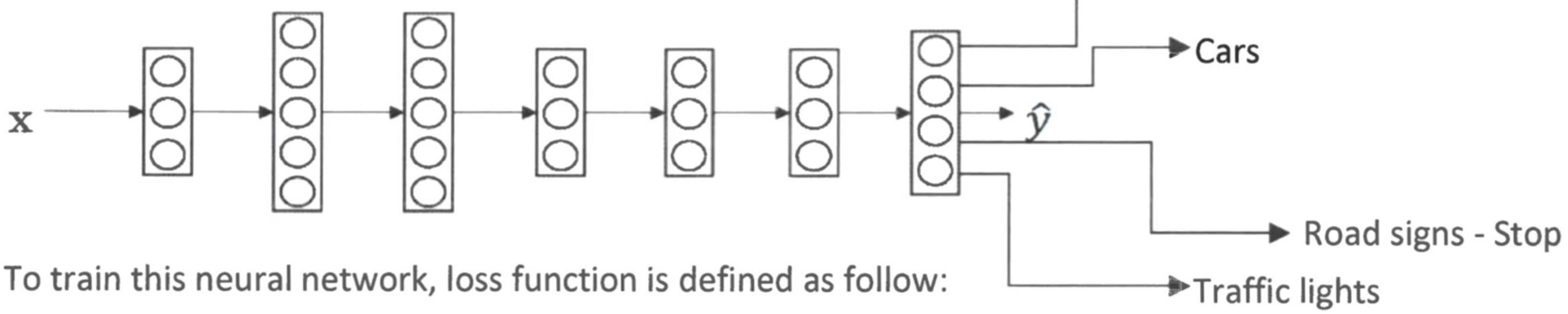
$$\hat{y} = W^\top h + b \qquad\qquad p(y|\hat{y}) = \mathrm{N}(y; \hat{y})$$

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

Loss function example

- NN does simultaneously several tasks (multi-task)

Neural Network architecture

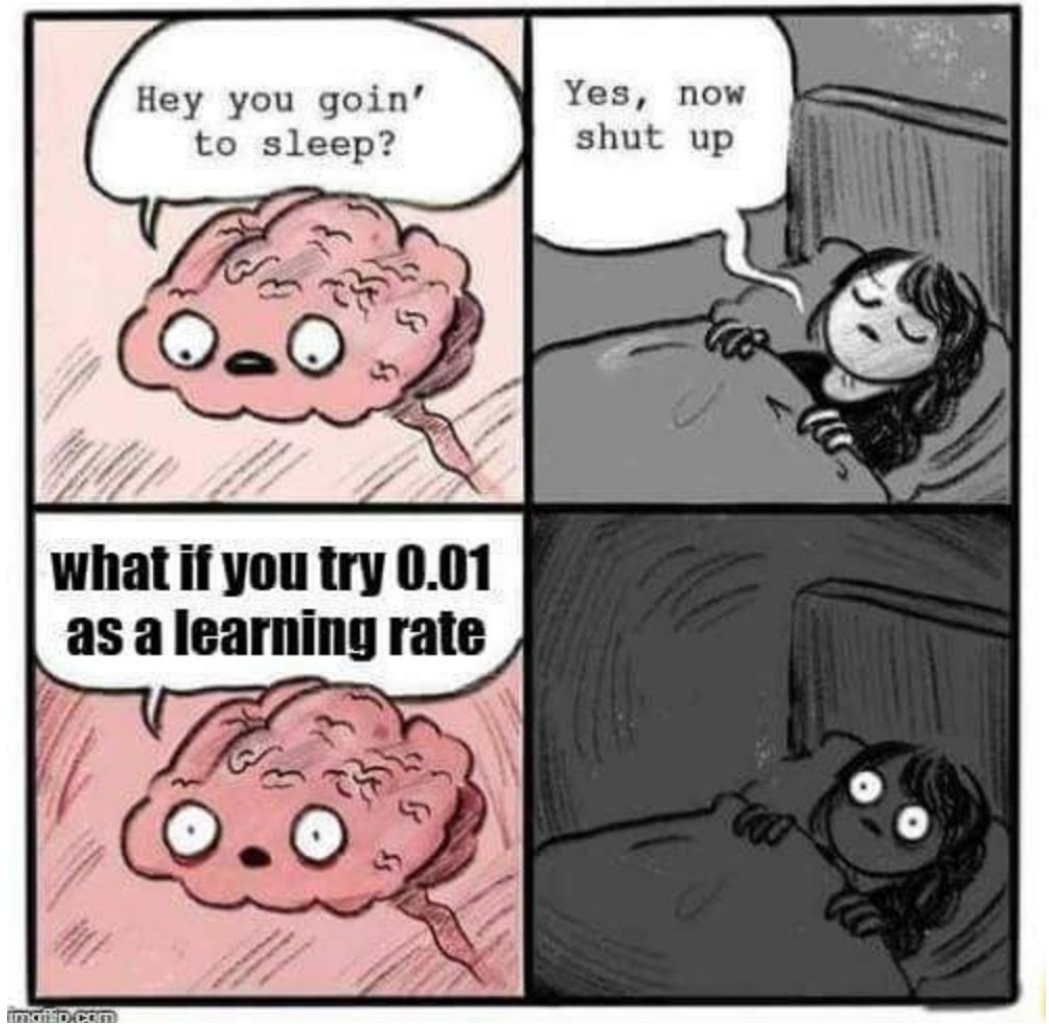


To train this neural network, loss function is defined as follow:

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

Hyperparameters

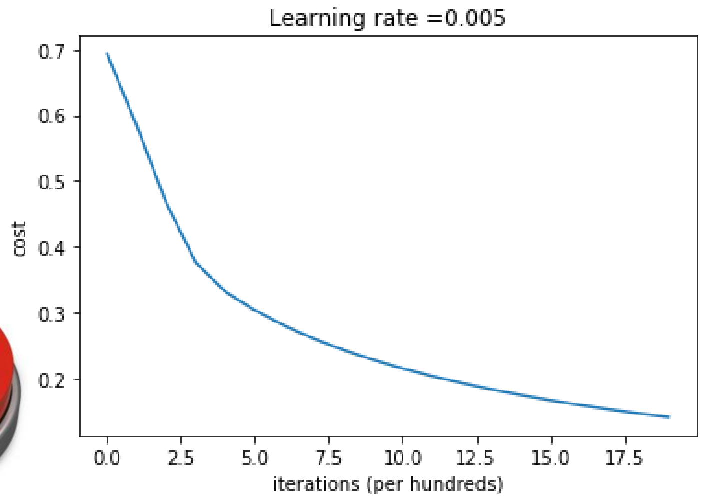
- Parameters that cannot be learnt directly from training data
- A long list…
  - Learning rate α
  - Number of iterations (epochs)
  - Number of hidden layers
  - Number of hidden units
  - Choice of activation function
  - *More to come !*



Training

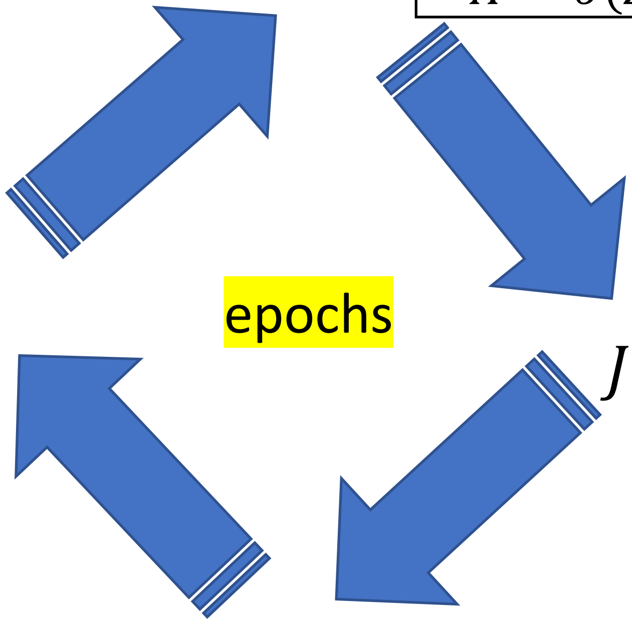
- *Iterative* process

Learning curve



Forward propagation $Z = w^T x + b$

$A = \sigma(Z)$



epochs

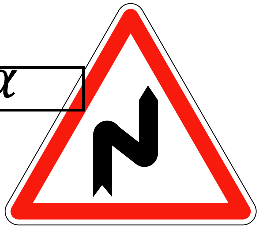
Cost function
$J(w, b) = J(\theta)$

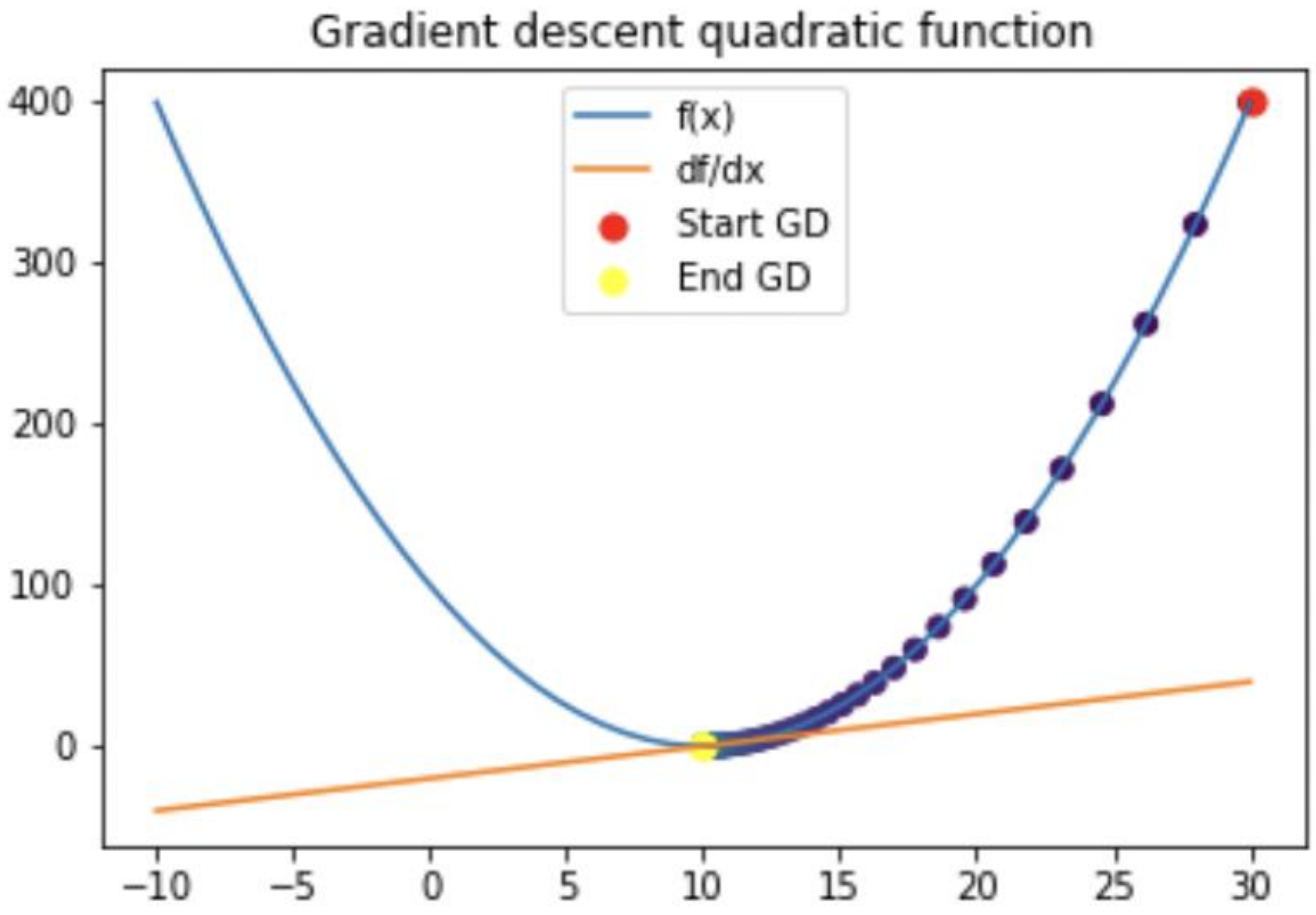
Backward propagation
(dJ/dw, dJ/db)

Parameter update
(gradient descent)

learning rate α

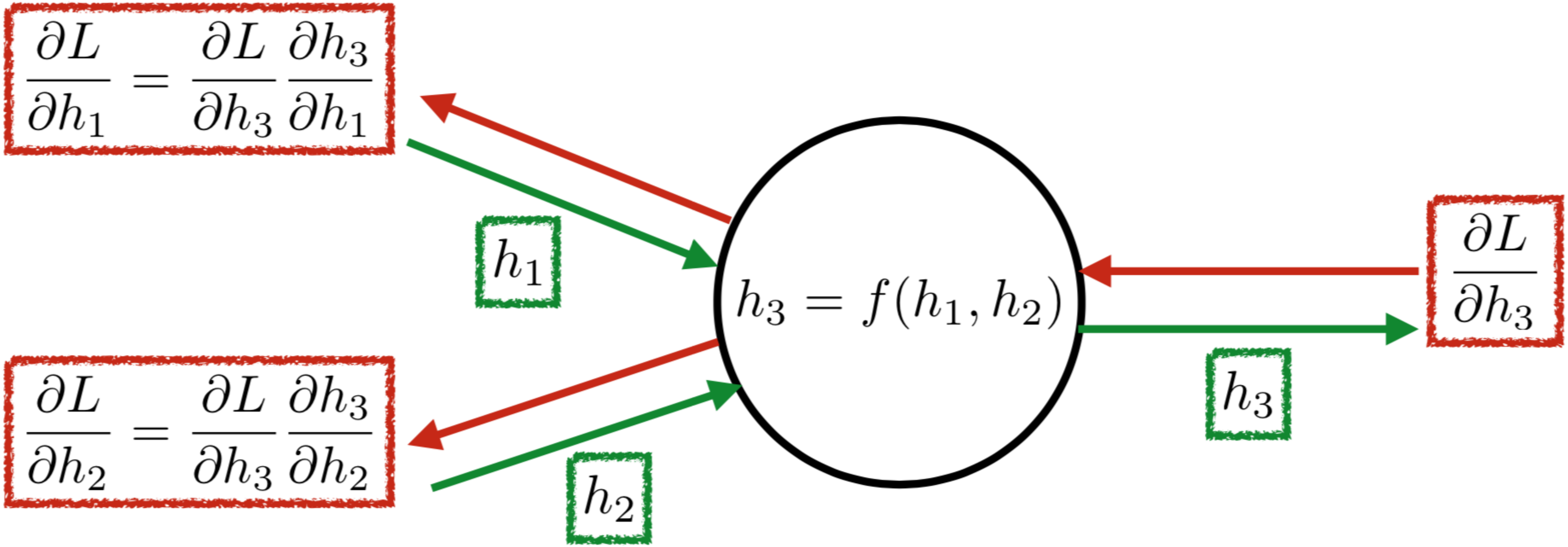
$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$$



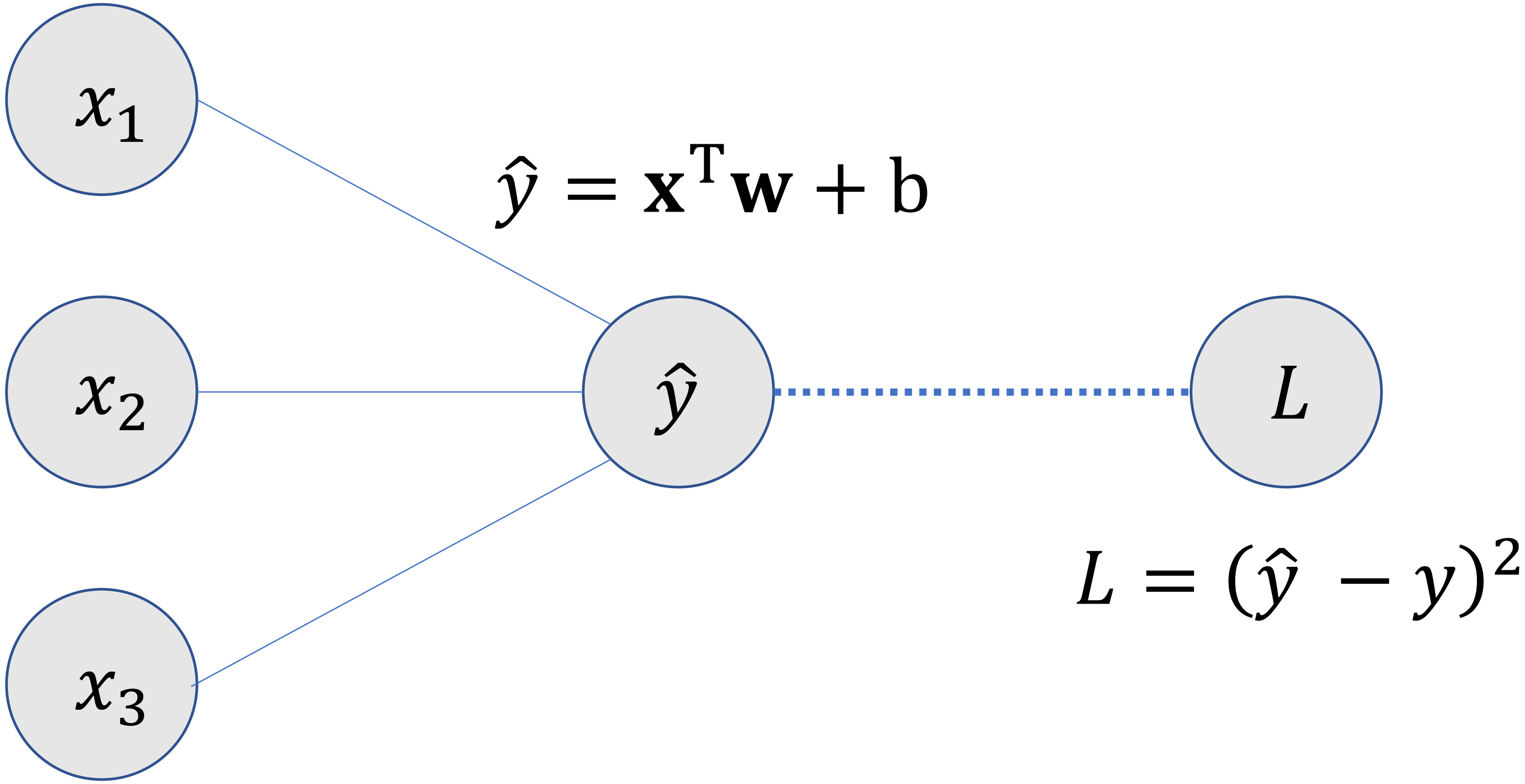

Gradient descent quadratic function
f(x)
df/dx
Start GD
End GD
400
300
200
100
0
−10
−5
0
5
10
15
20
25
30

Backpropagation

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



Example



x_1
x_2
x_3
$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$
$\hat{y}$
L
$L = (\hat{y} - y)^2$

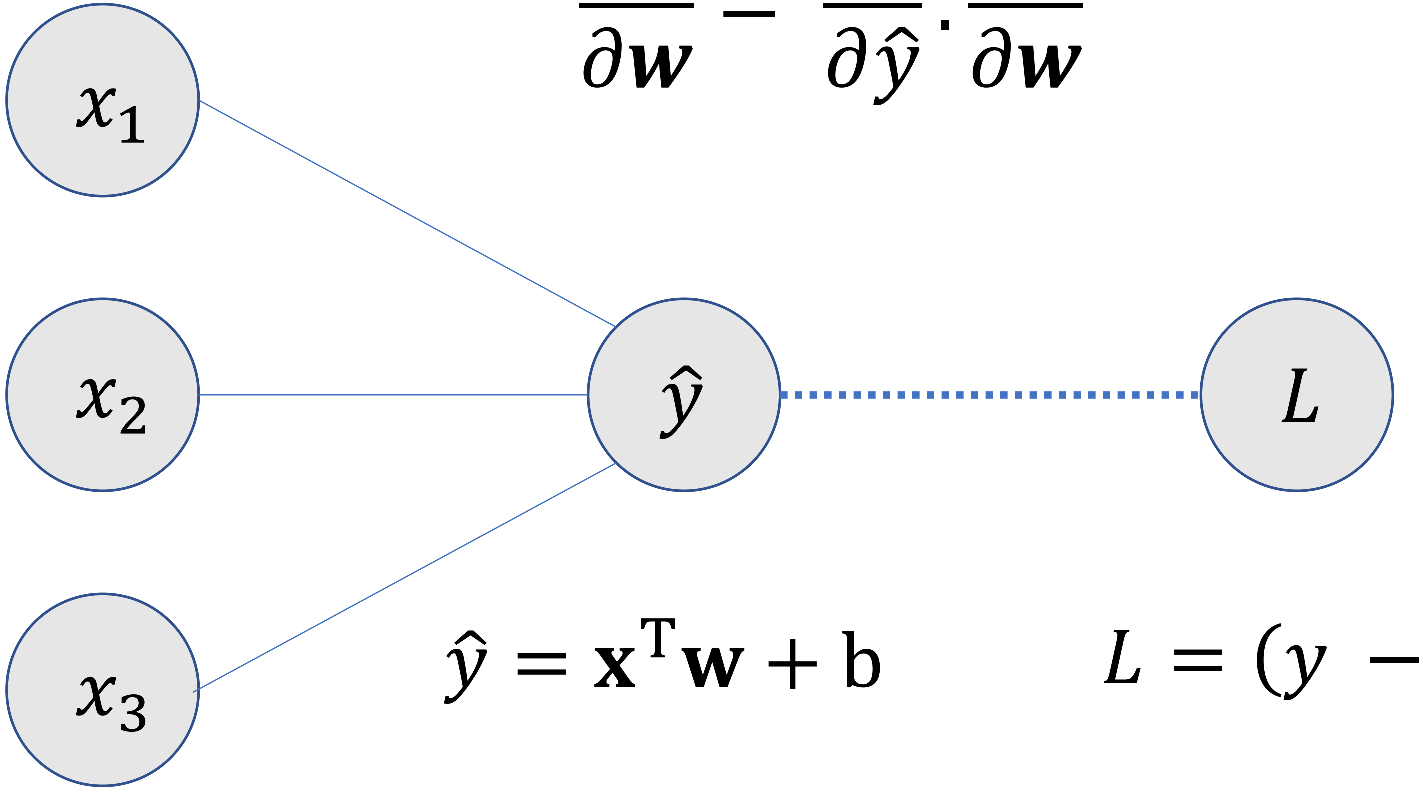
Example

We need to calculate the gradients:

Let's start with this part!

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

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



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

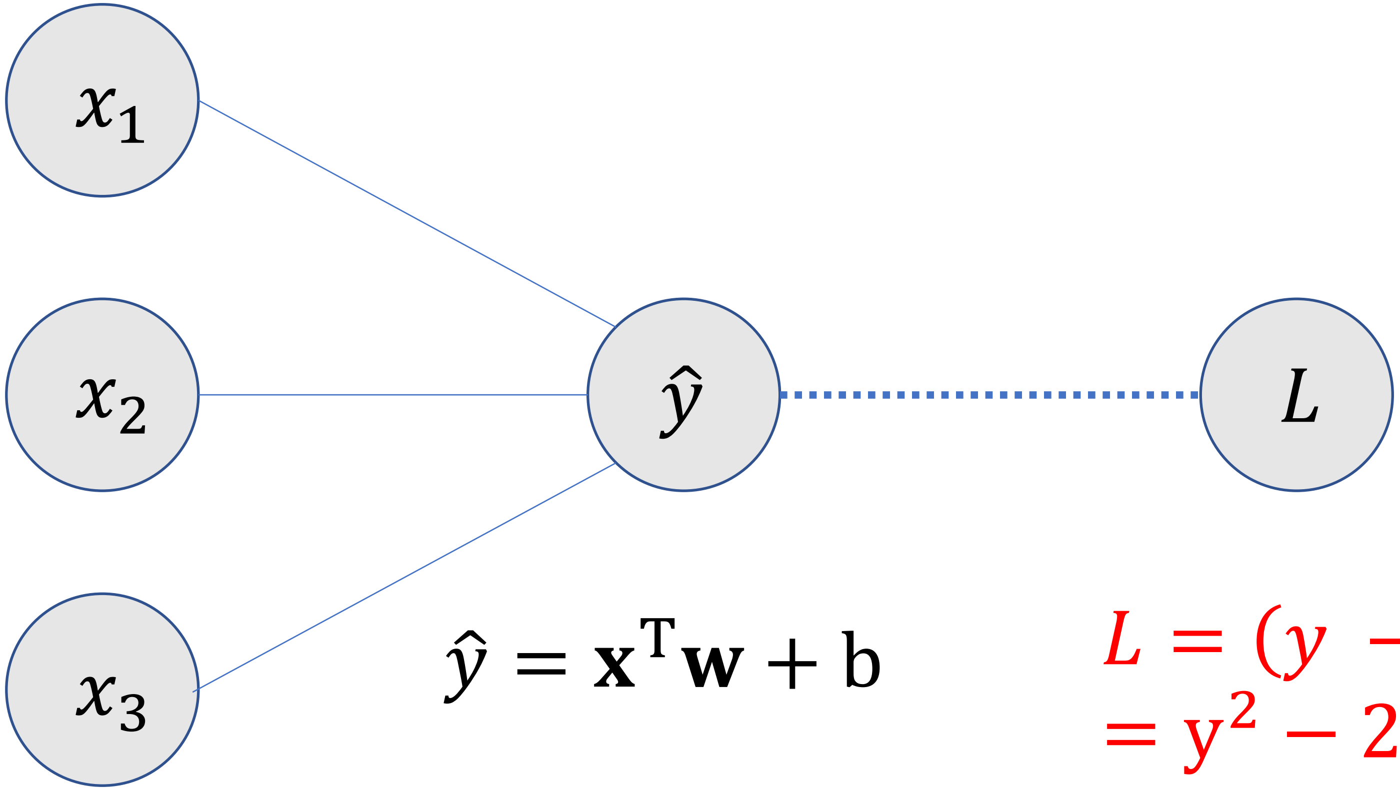
$$L = (y - \hat{y})^2$$

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

Example

First:

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



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

$$L = (y - \hat{y})^2$$
$$= \mathrm{y}^2 - 2\mathrm{y}\hat{y} + \hat{y}^2$$

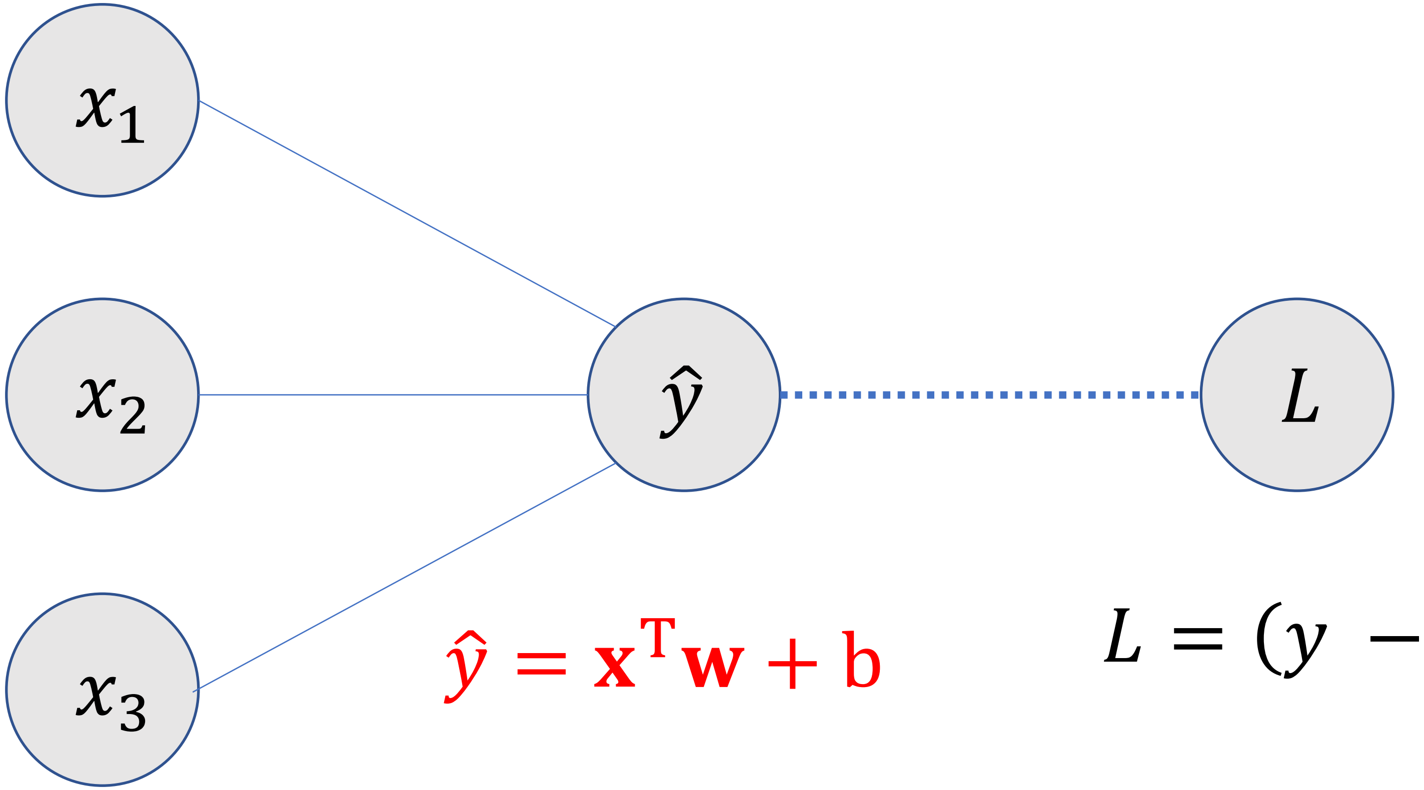
$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

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

Example

Second:

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



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

$$L = (y \; - \; \hat{y})^2$$

Example

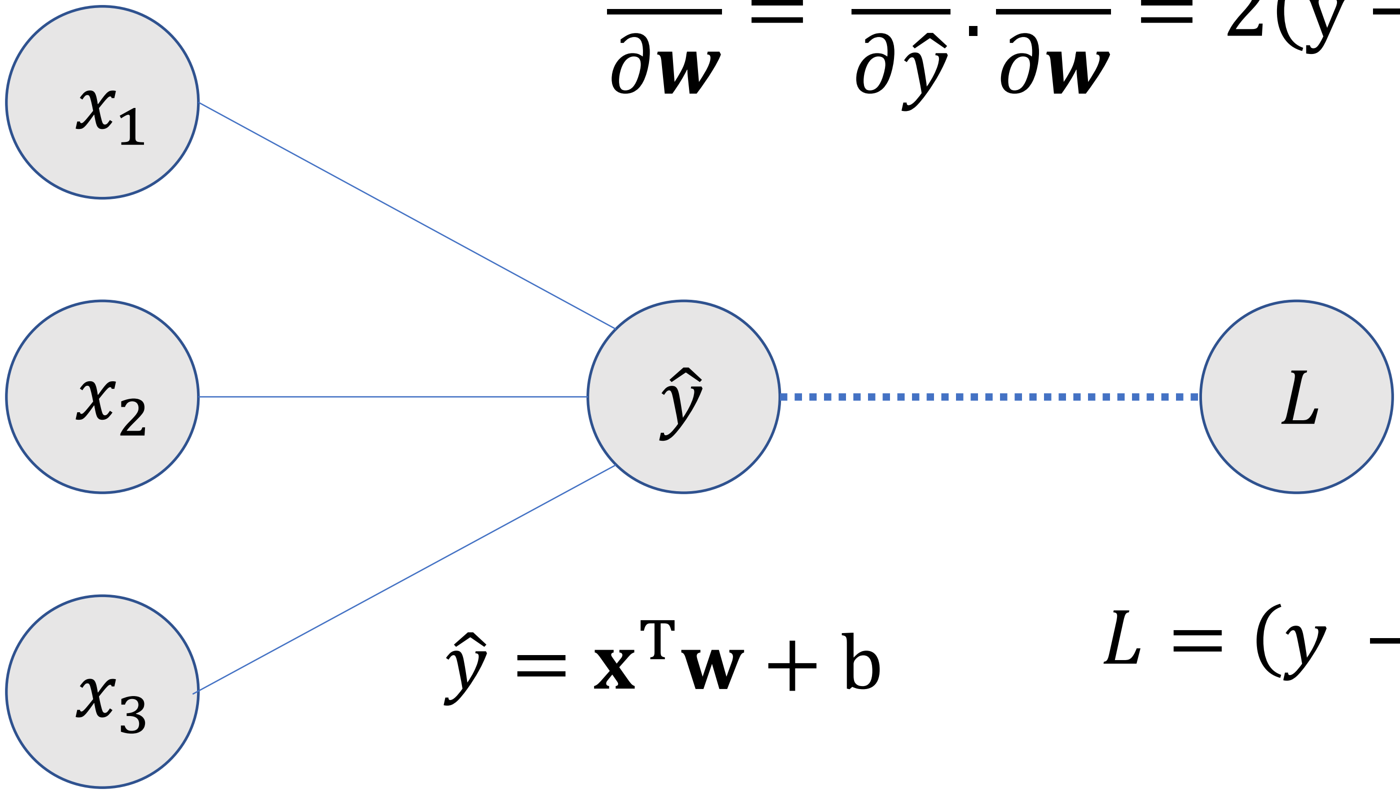
$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

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

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

Putting these together:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}} = 2(\mathrm{y} - \hat{y}). \boldsymbol{x}^{T}$$



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

$$L = (y \; - \; \hat{y})^2$$

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

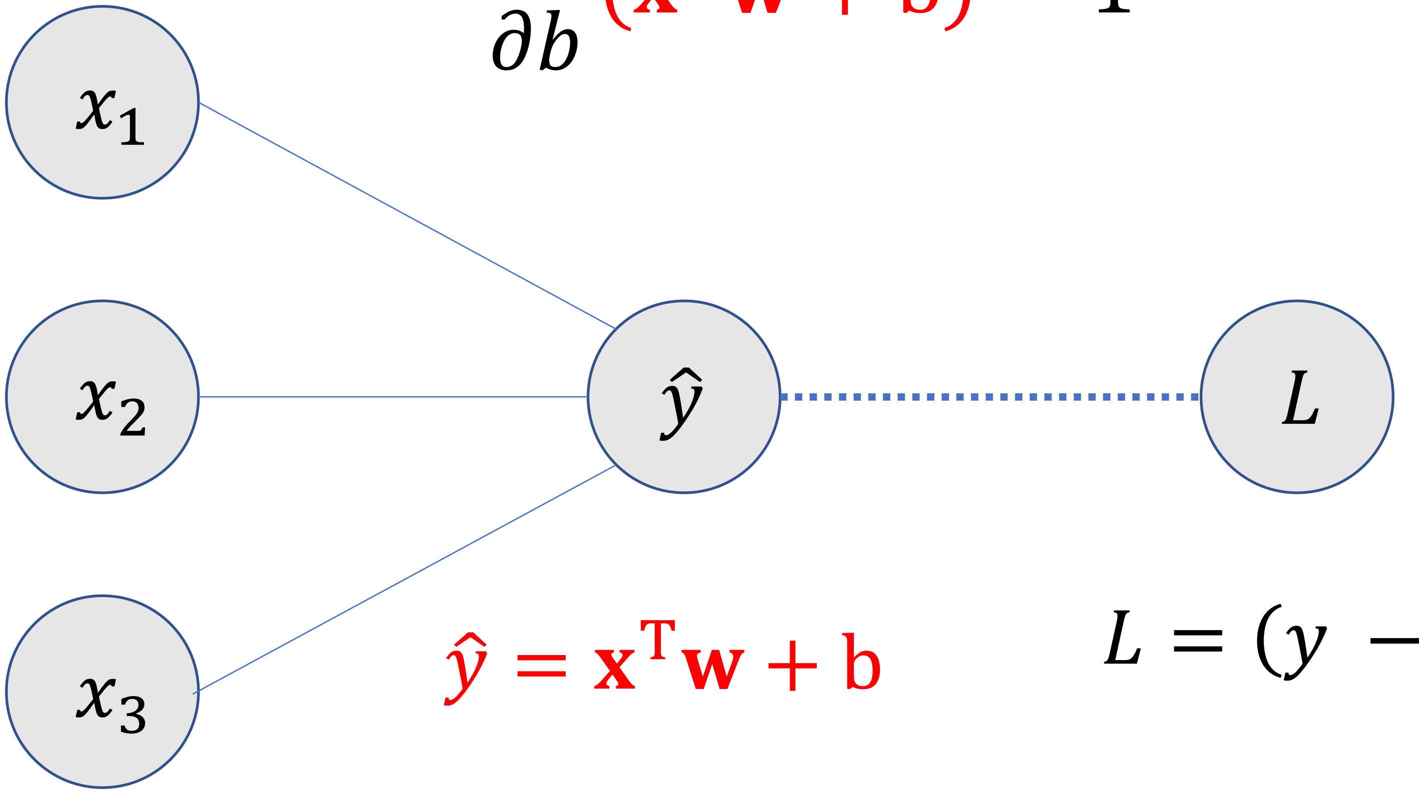
$$\frac{\partial L}{\partial \hat{y}} = -2y \; - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$

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

Example

Now for the bias…

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



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

$$L = (y \; - \; \hat{y})^2$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

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

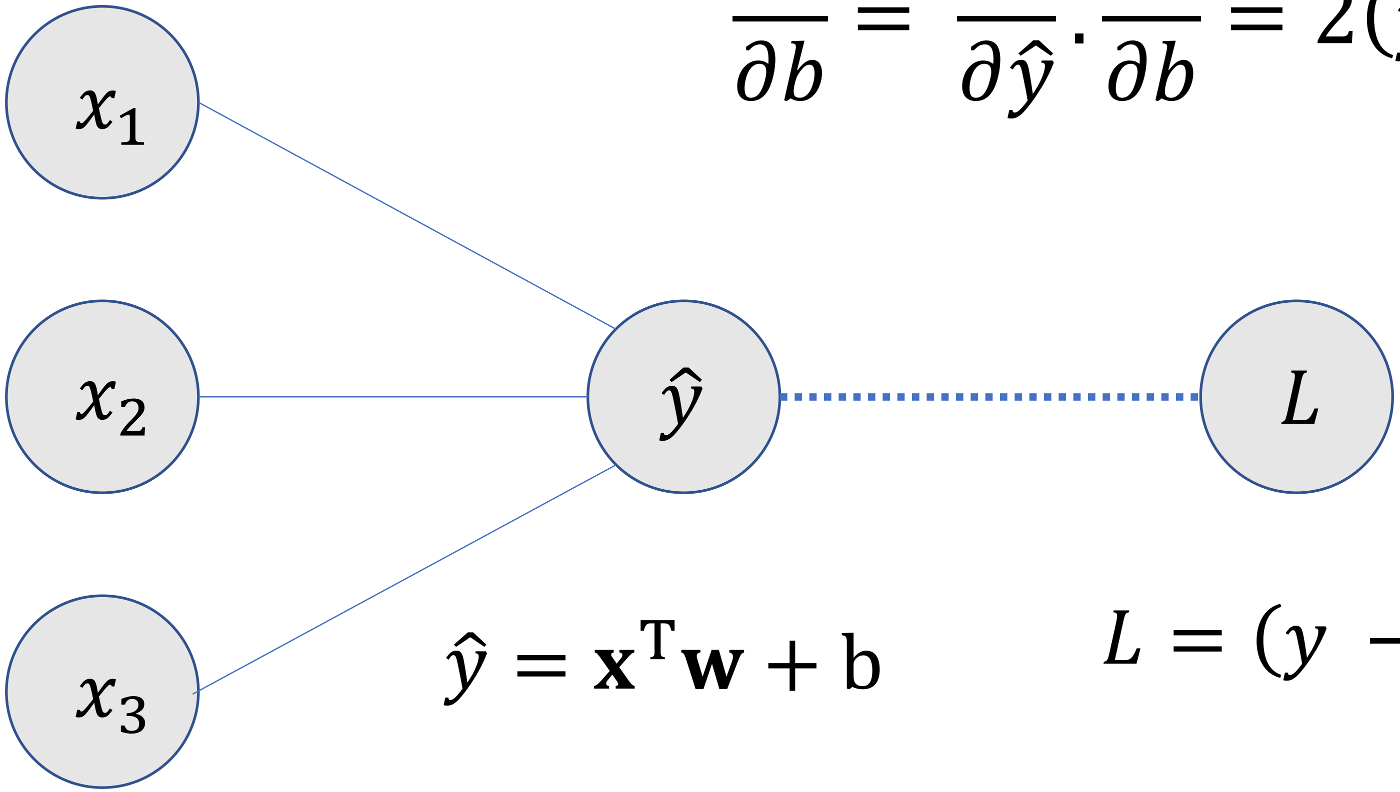
$$\frac{\partial}{\partial \boldsymbol{w}}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = \boldsymbol{x}^{T}$$

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

Example

Putting these together:

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



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

$$L = (y \; - \; \hat{y})^2$$

Example

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

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

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

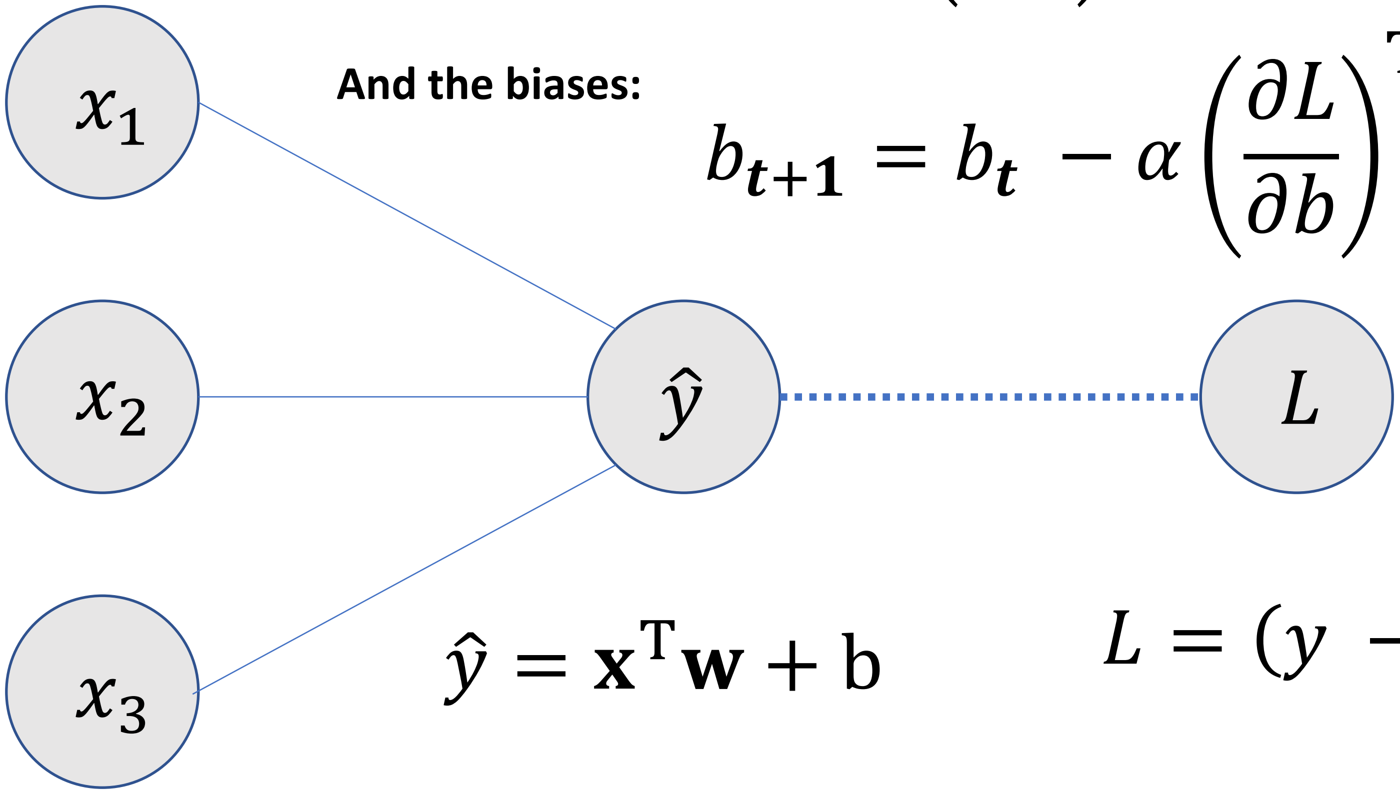
$$\frac{\partial}{\partial b}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = 1$$

Finally the updates for the weights:

$$\boldsymbol{w_{t+1}} = \boldsymbol{w_t} \; - \alpha\left(\frac{\partial L}{\partial \boldsymbol{w}}\right)^{\mathrm{T}} = \boldsymbol{w_t} - 2\alpha(\hat{y} - y)\boldsymbol{x}$$

And the biases:

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



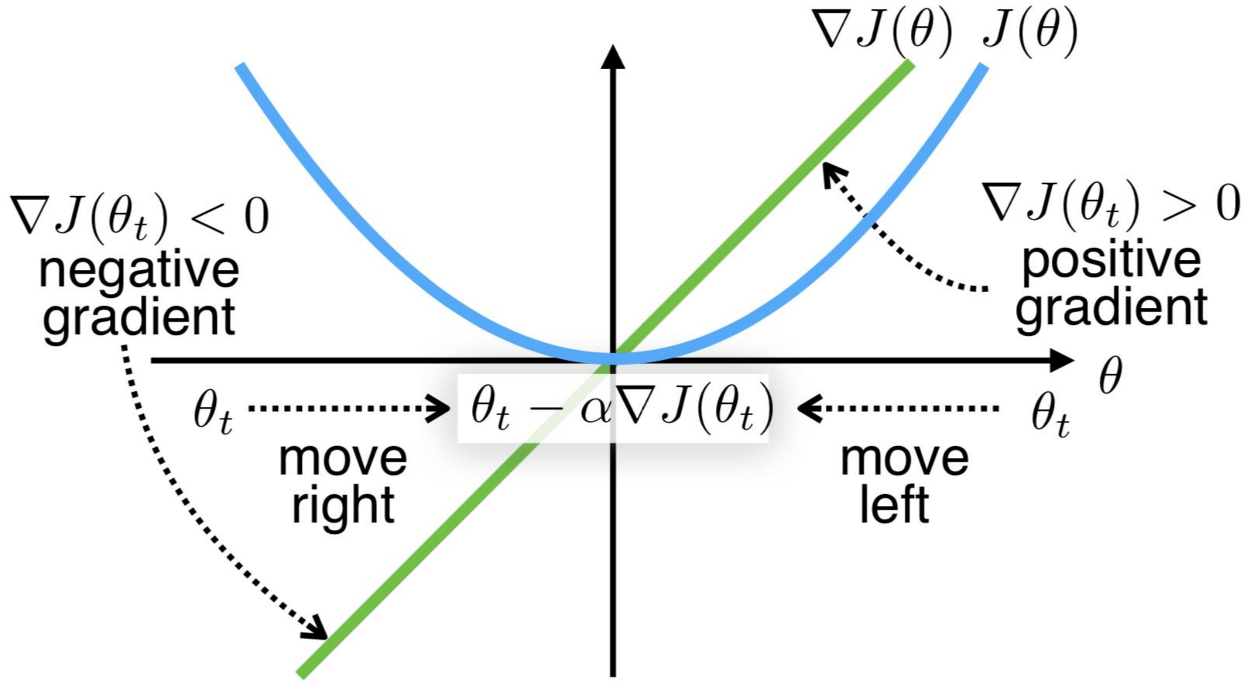
$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

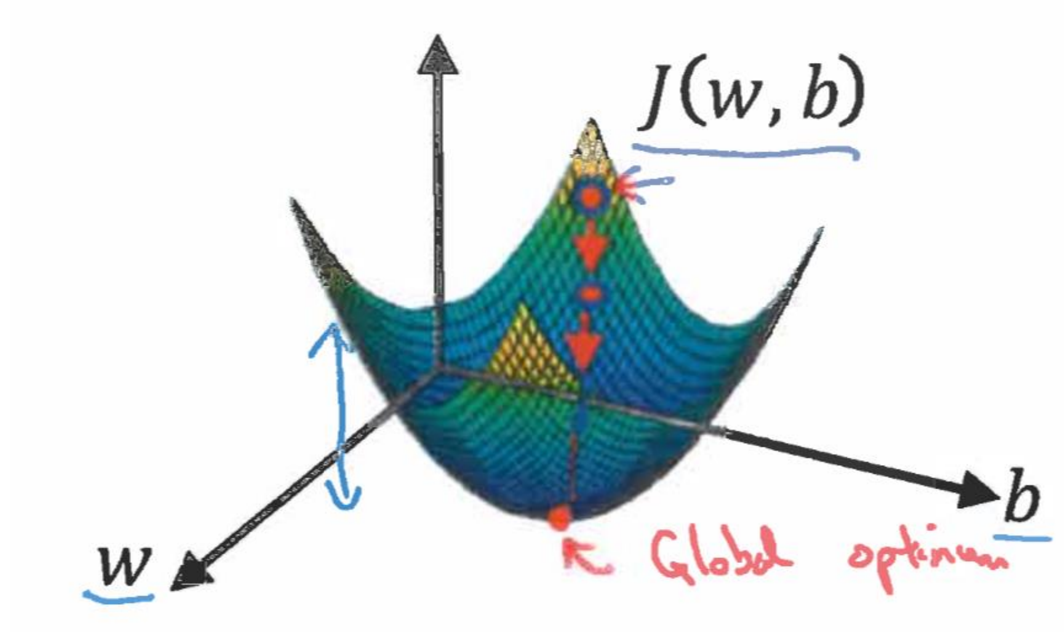
$$L = (y \; - \; \hat{y})^2$$

Gradient Descent

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

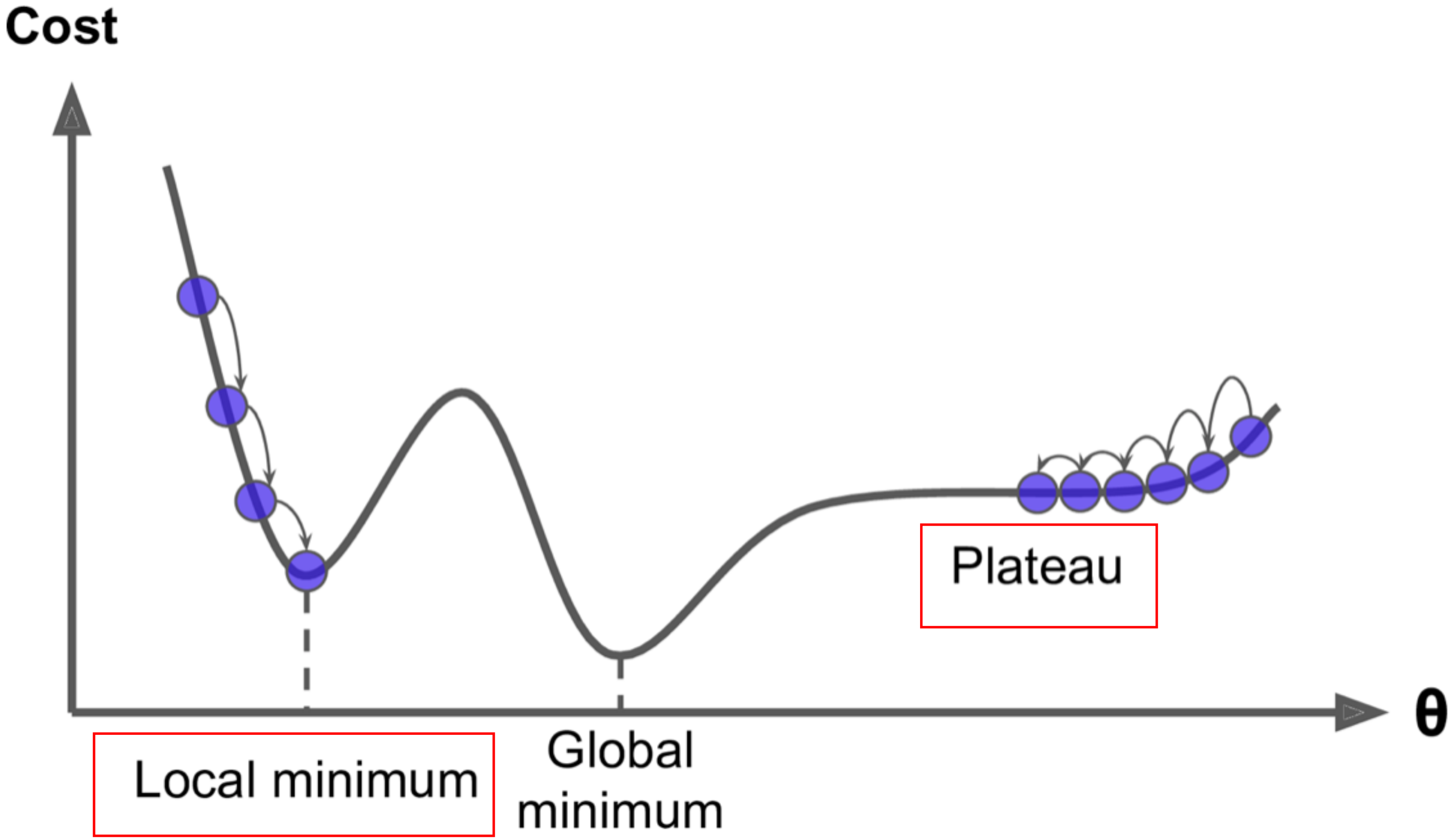
gradient descent $\boxed{\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)}$



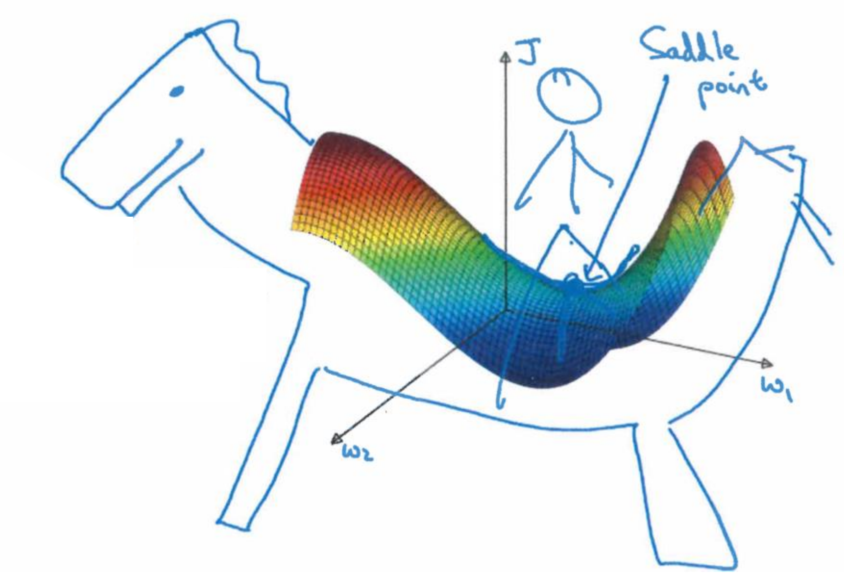


$$\nabla J(w) = \frac{dJ(w,b)}{dw} \qquad \nabla J(b) = \frac{dJ(w,b)}{db}$$

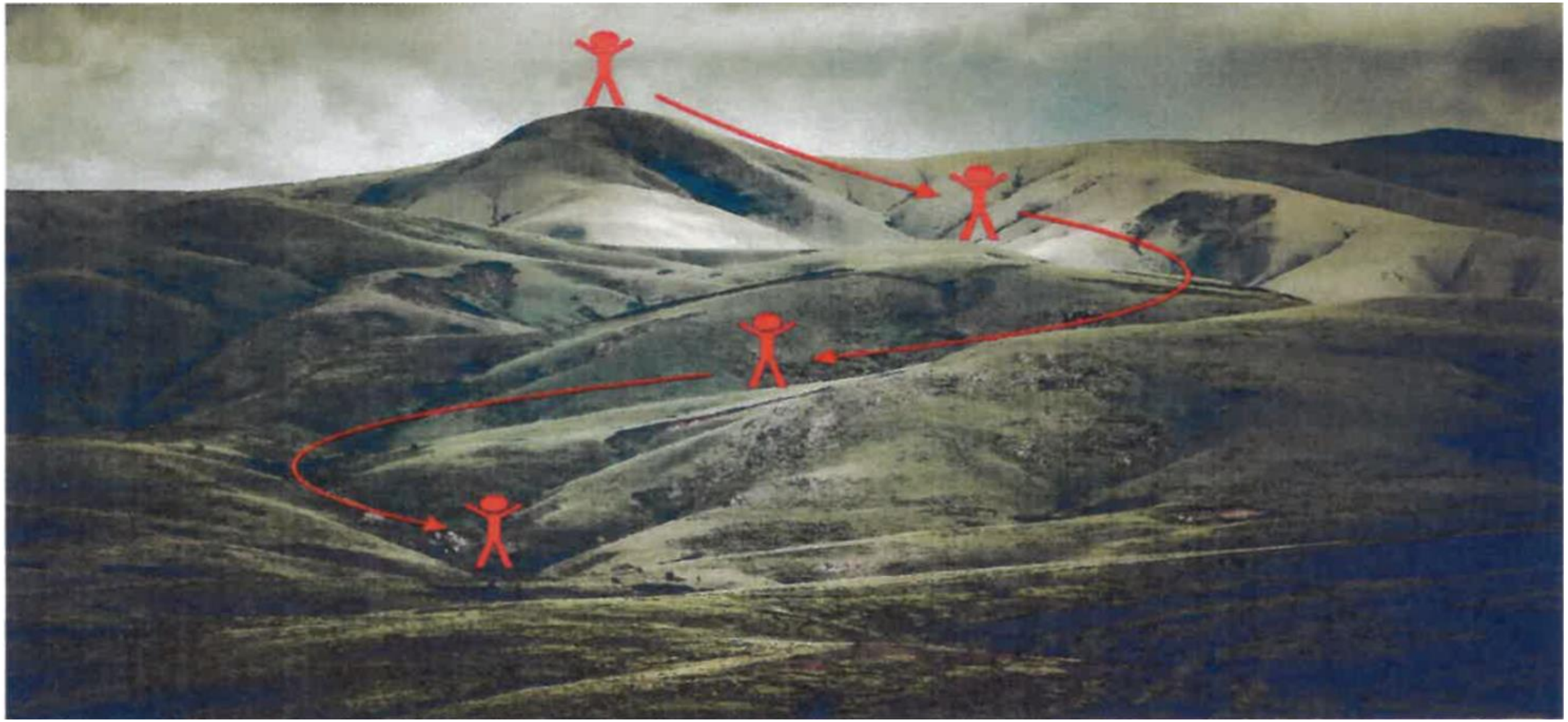
Optimization pitfalls



Saddle point



Gradient Descent Illustration



Tutorial / Practical