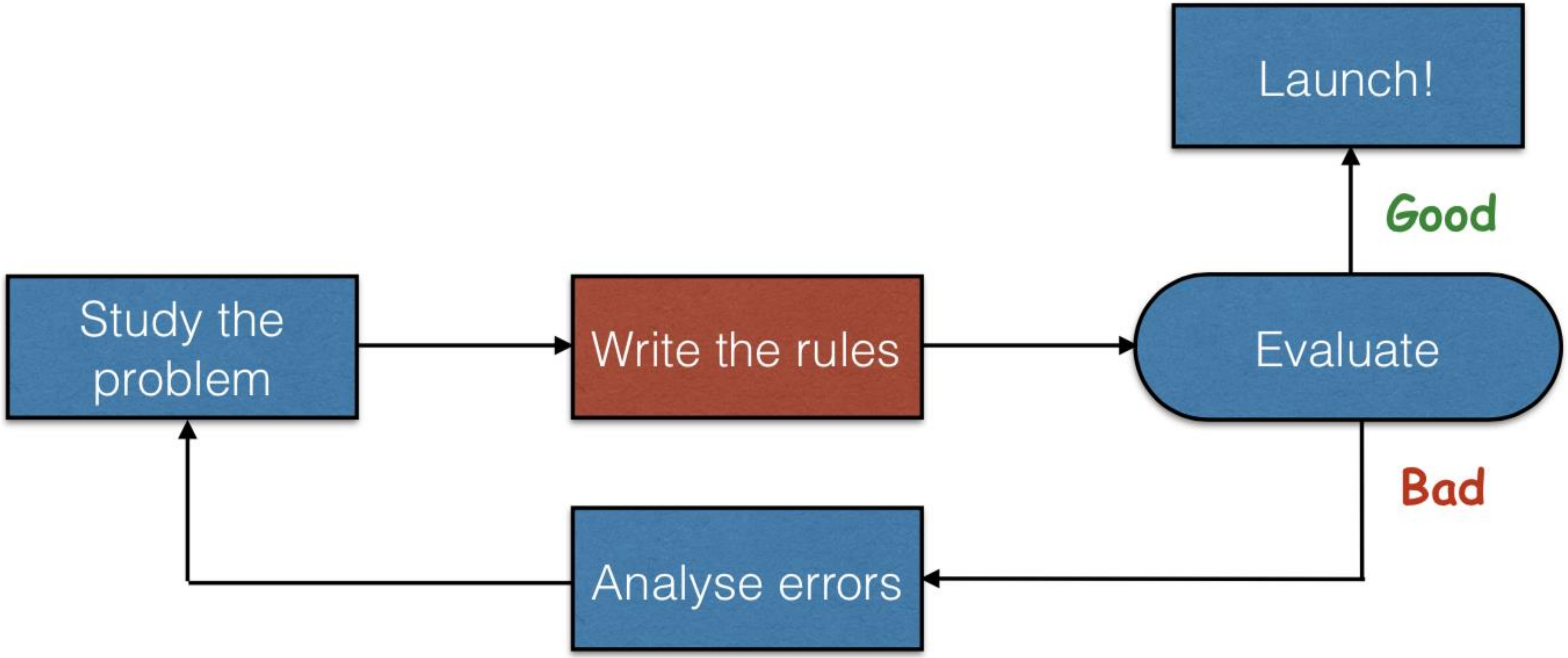
Neural Networks

Géraldine Conti, Matthew Vowels, Mykhailo Vladymyrov

Bern Winter School –
Natural Language Processing,

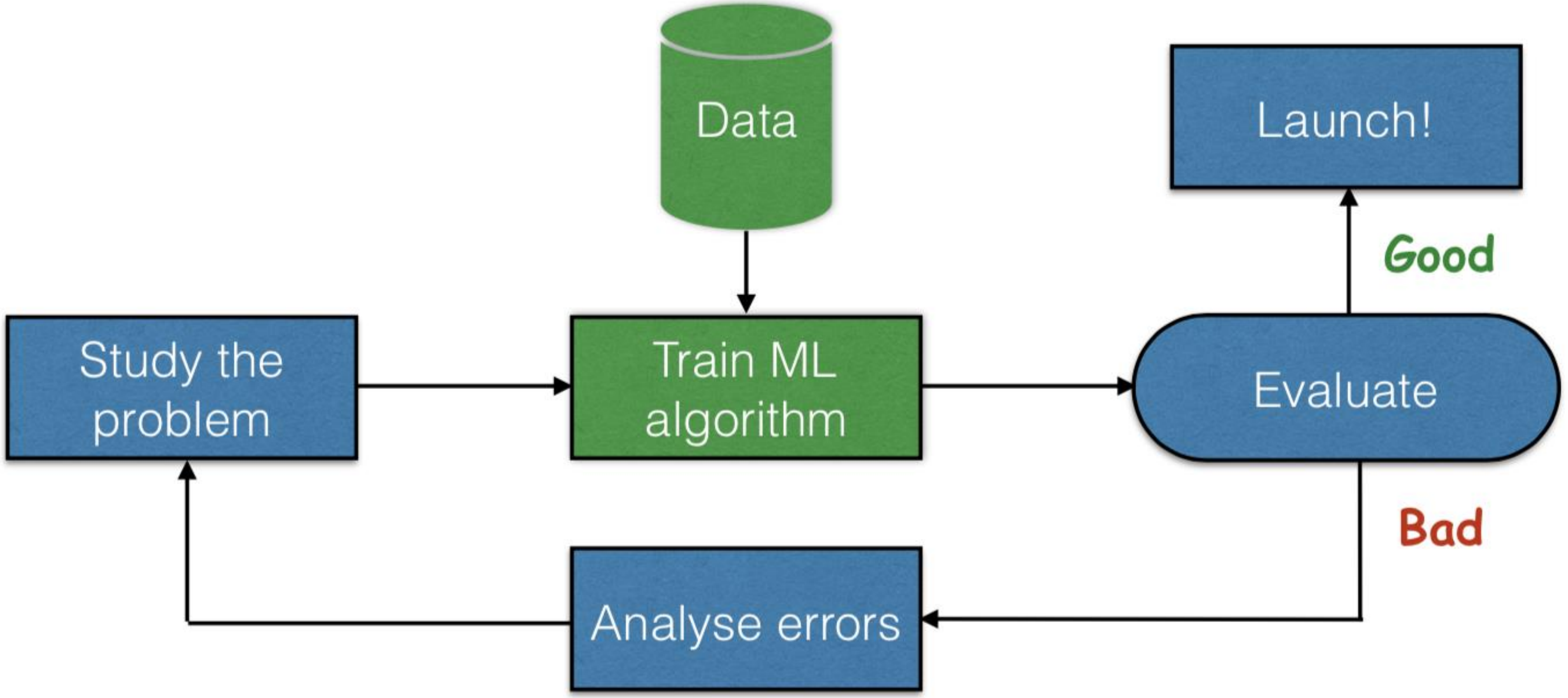
Murren 2024

Traditional approach (Software 1.0)



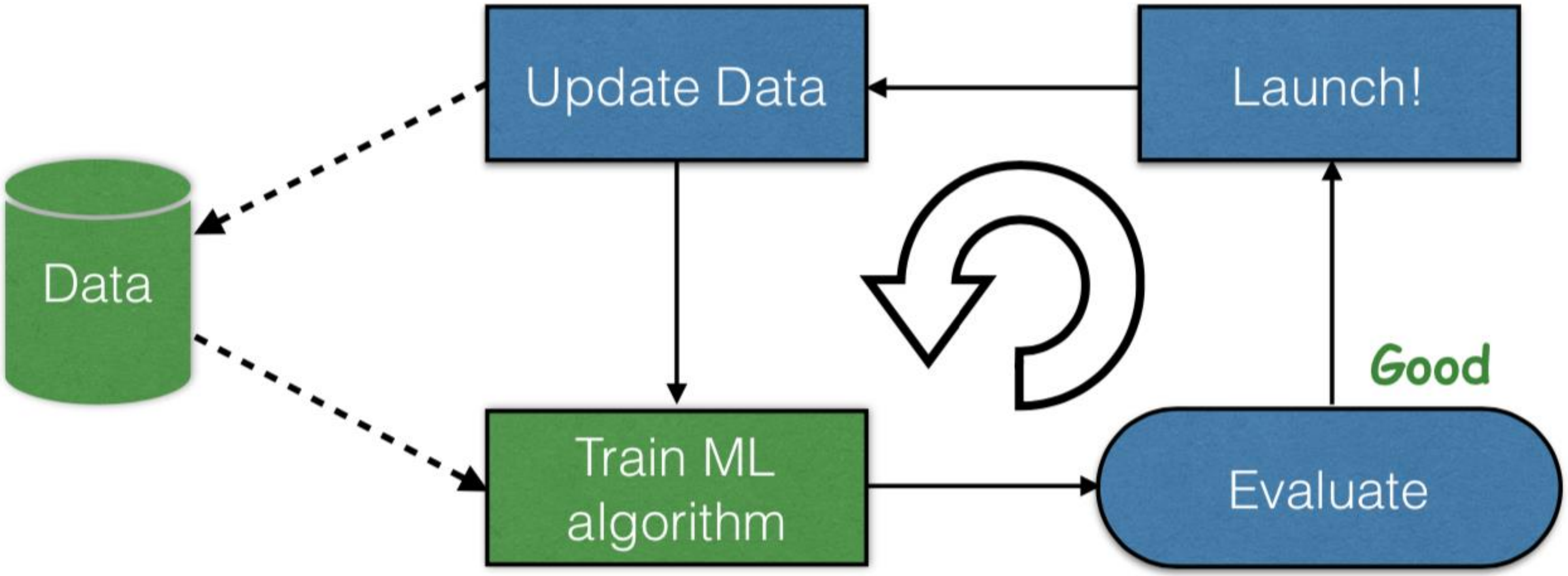
List of all the knowledge and formal rules

Machine Learning approach (Software 2.0)



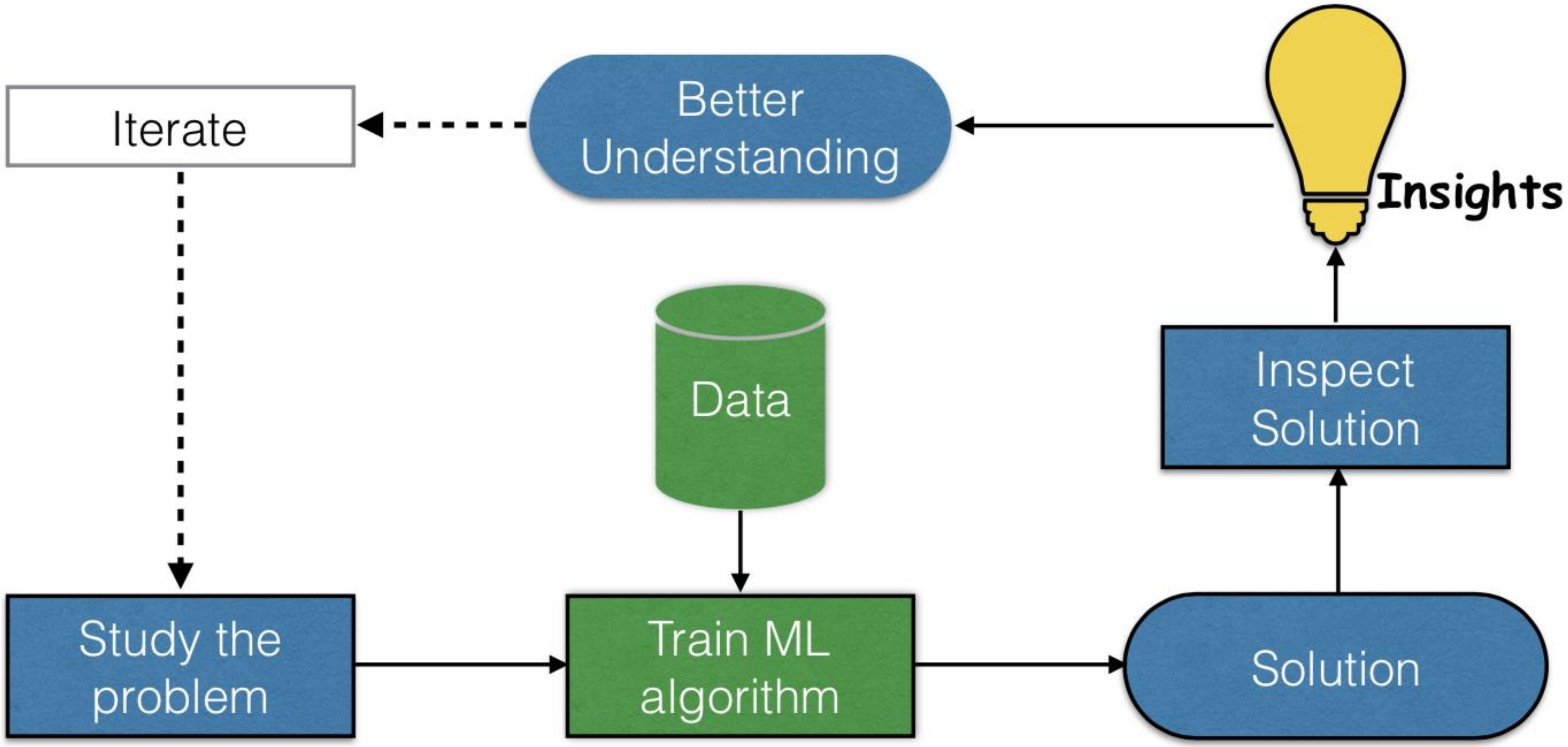
Learning from examples

Machine Learning approach (Software 2.0)



Adapting to change

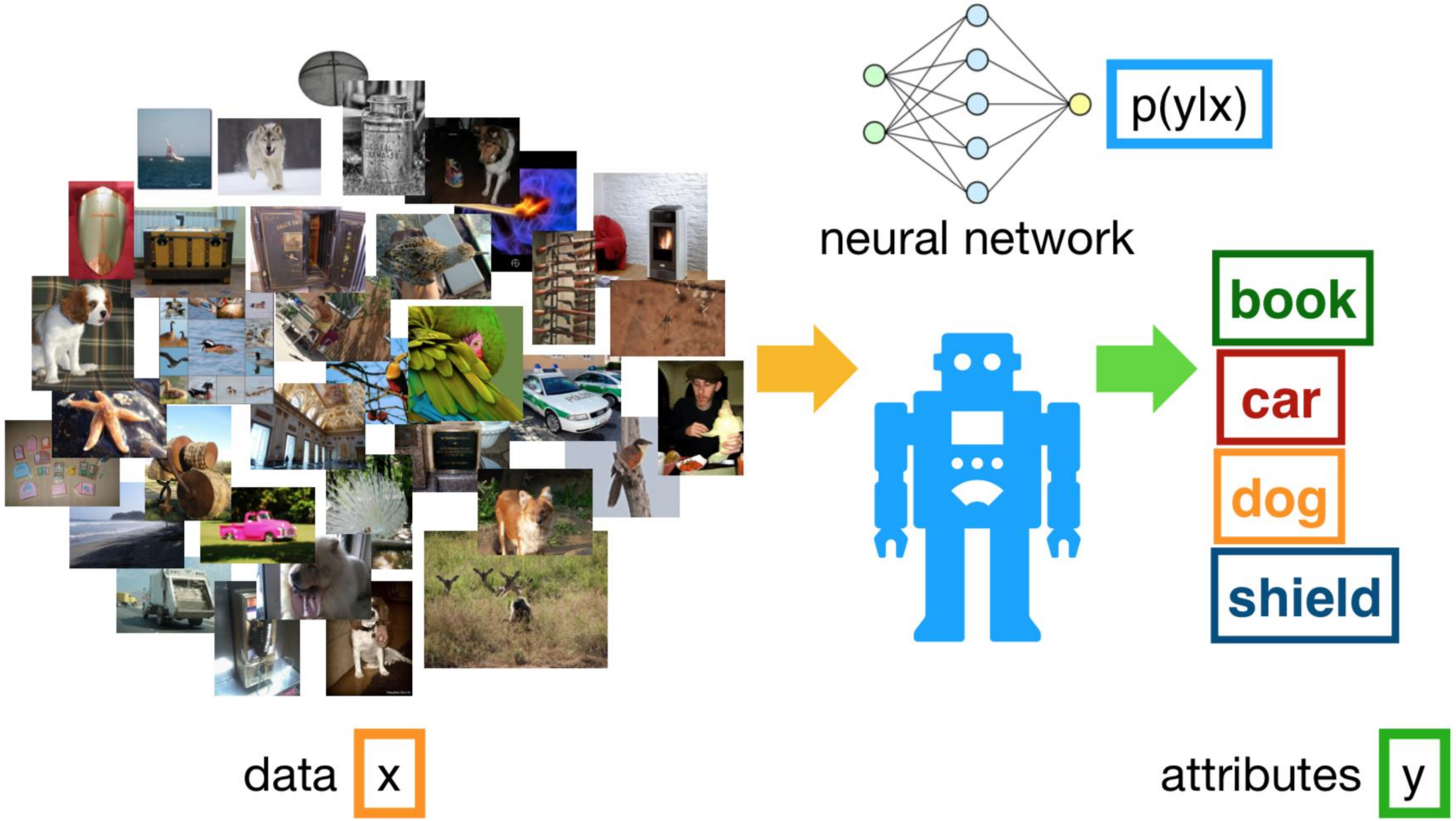
Machine Learning approach (Software 2.0)



Help humans learn

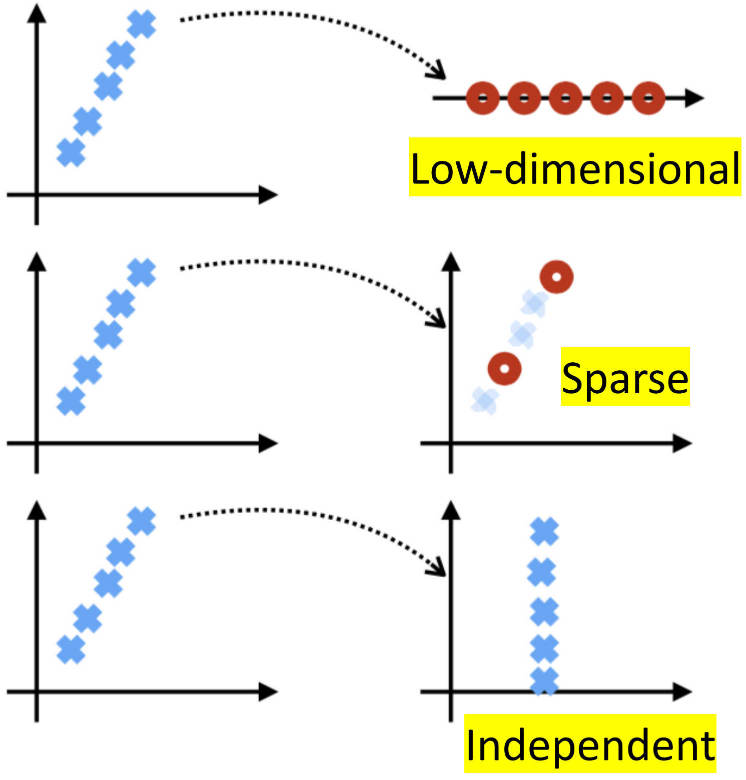
Supervised Learning

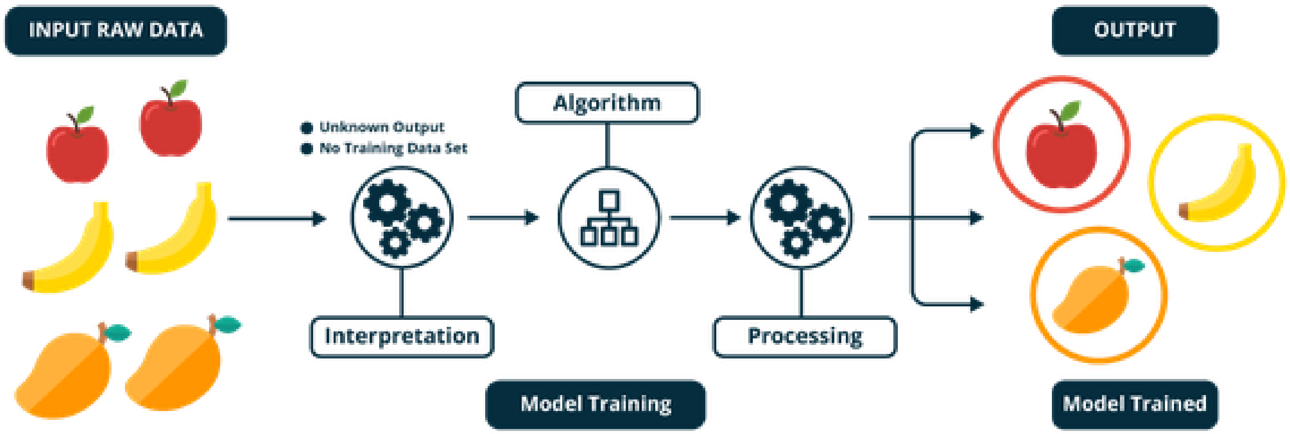
- Prediction of an output **y** given an input **x**



Unsupervised Learning

- Find a *suitable data representation*
  - Preserving all task-relevant information
  - Simpler than the original data and easier to use





Data assumption

- IID (independent and identically distributed)

1) Come from the *same distribution*

$$p_{x^{(i)}}(x) = p_{x^{(j)}}(x)$$

2) Are *independent*

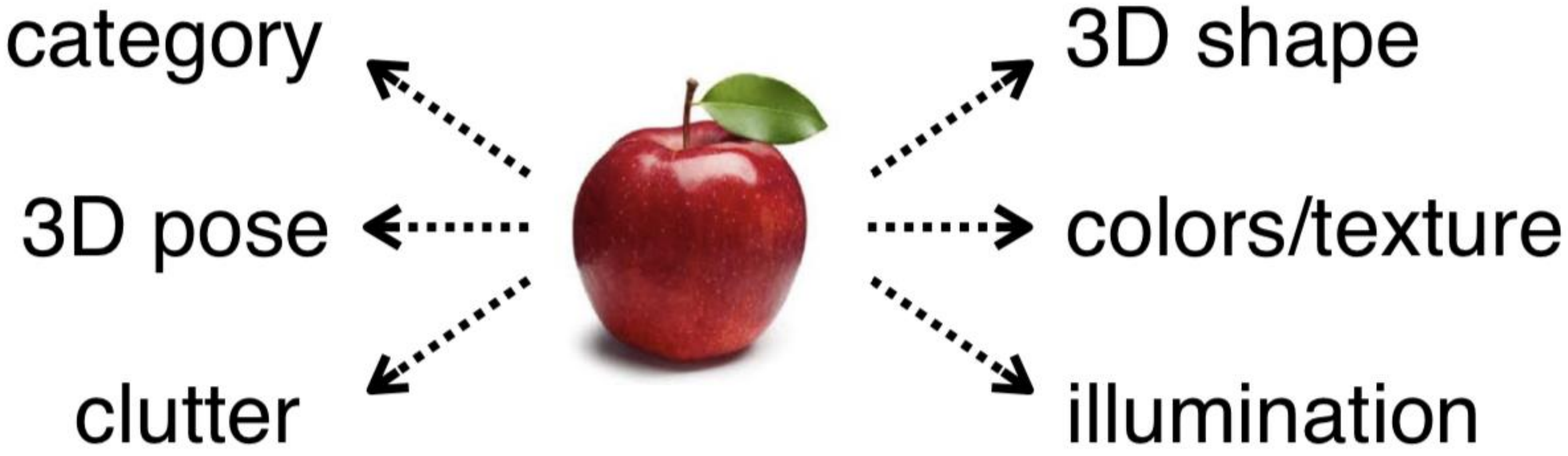
$$p\left(x^{(1)}, \ldots, x^{(m)}\right) = \prod_{i=1}^{m} p\left(x^{(i)}\right)$$

Features

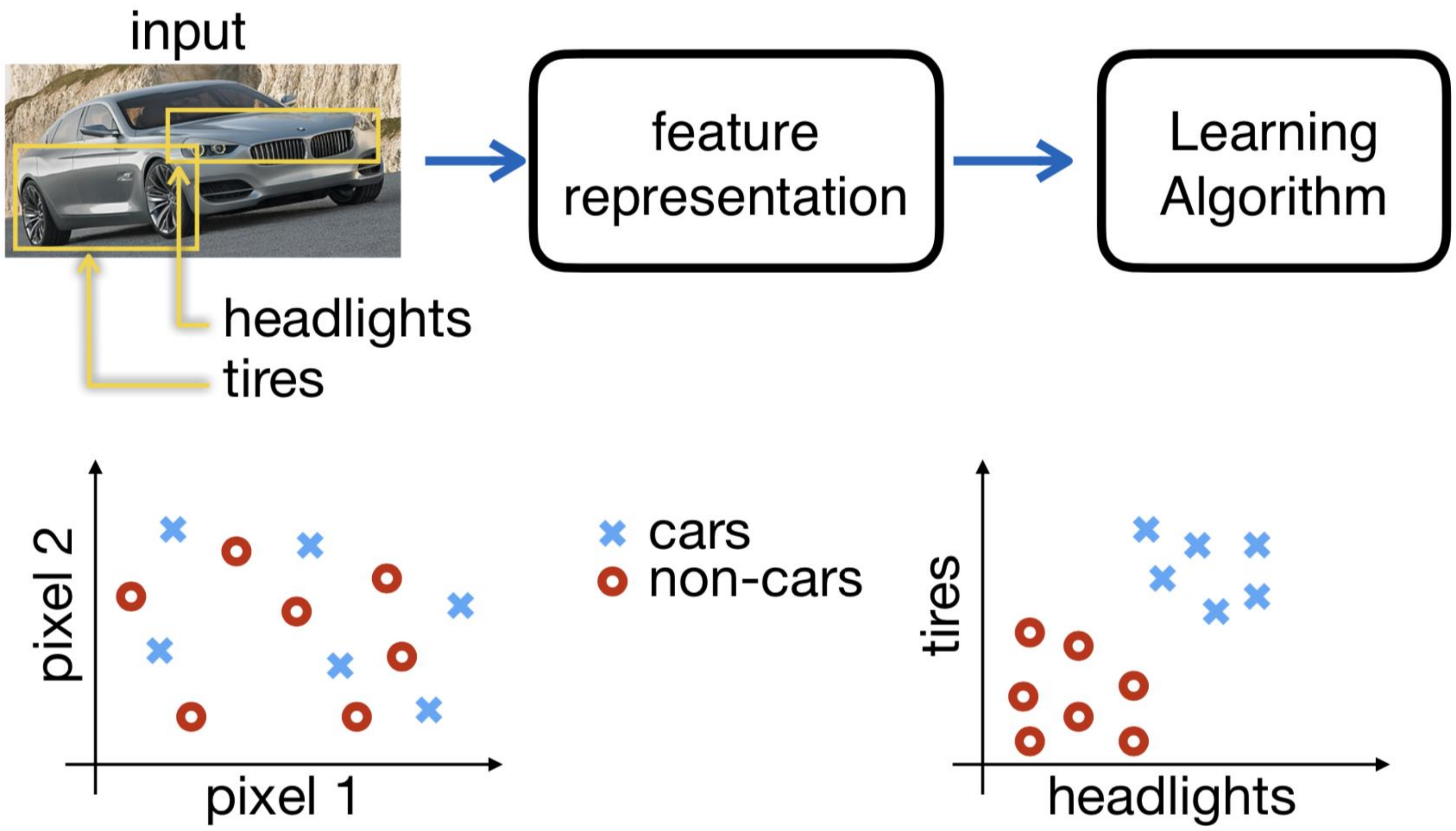
- Data often encoded into more focused relevant information (features or internal representation) to simplify the decision

$$x \to \phi(x)$$

data feature



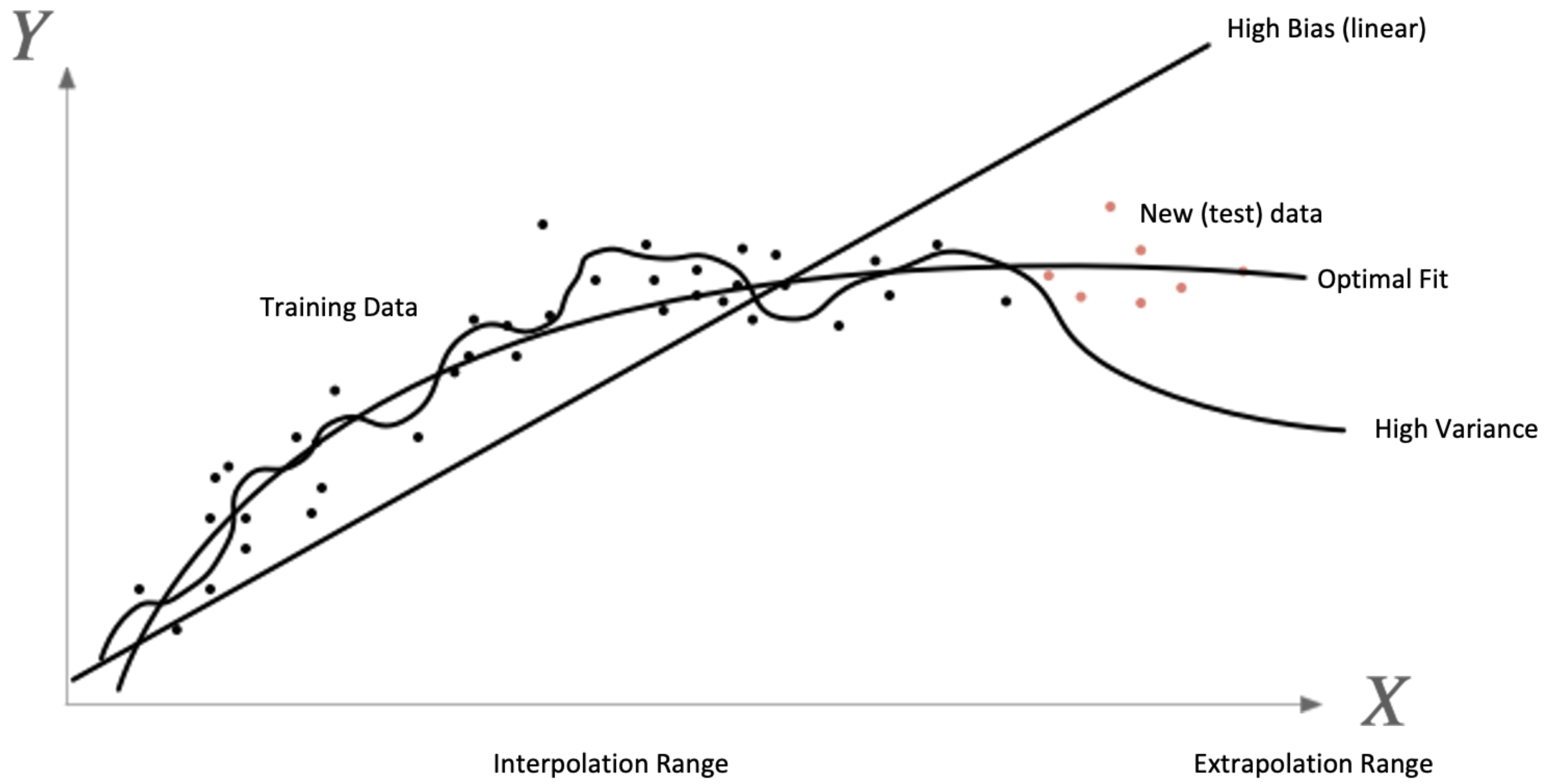
Features Example :Image classification

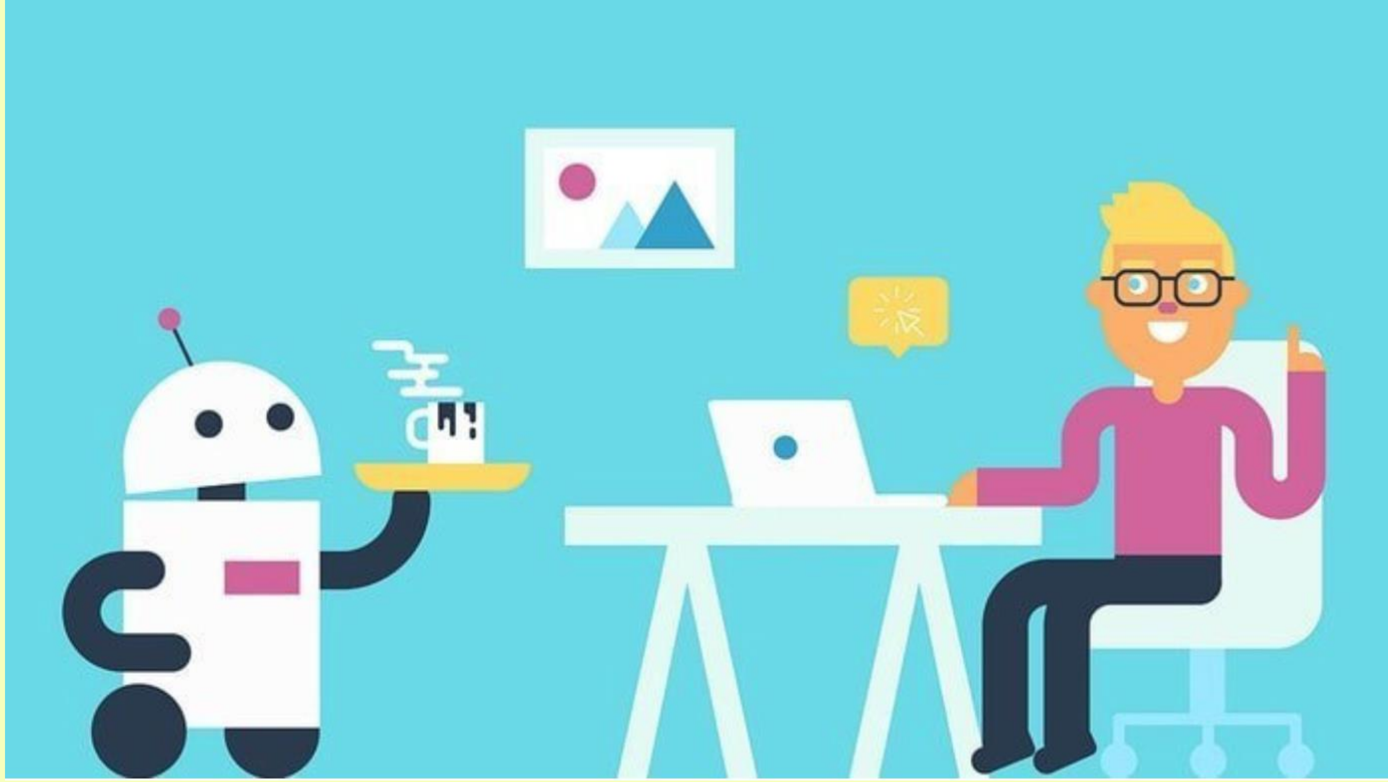


Training/Validation/Test sets

- Separate the data into 2(3) sets
  - Training set for training
  - Development / Validation set to find the best parameters
  - (Test set to estimate the performance)
- Separation depends on size of the dataset
- Make sure no algorithmic decisions are being made using data which are also being used to test the algorithm

Training/Validation/Test sets

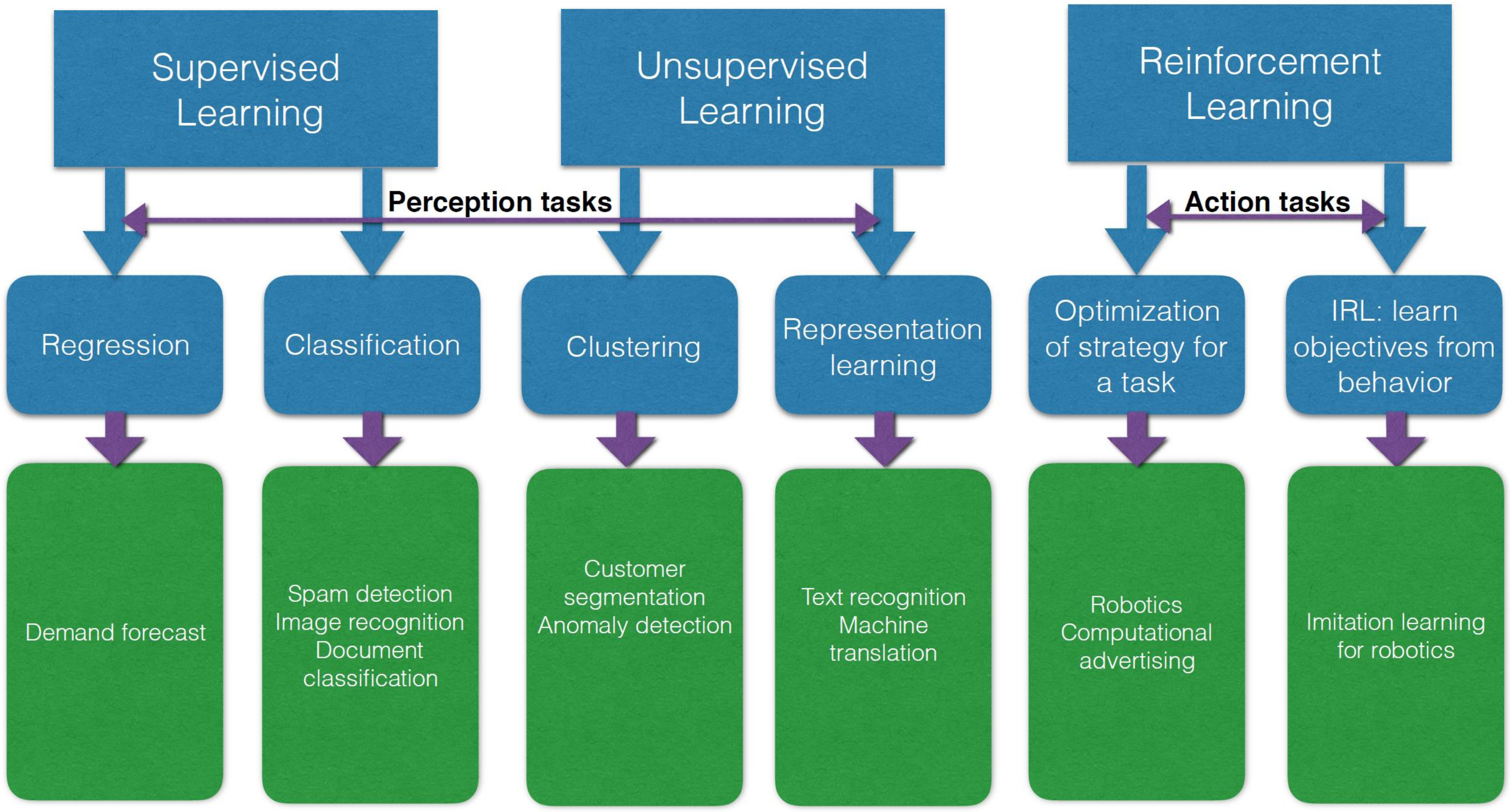




Task T

Find the function f that satisfies f(x) = y using the *training set*

Problem Types



Regression

Predict results within *a continuous output*

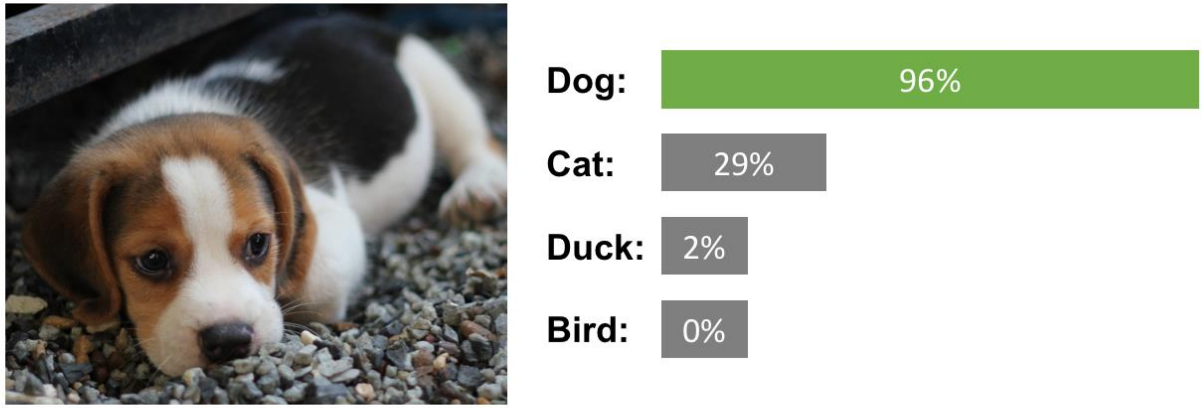
$82000

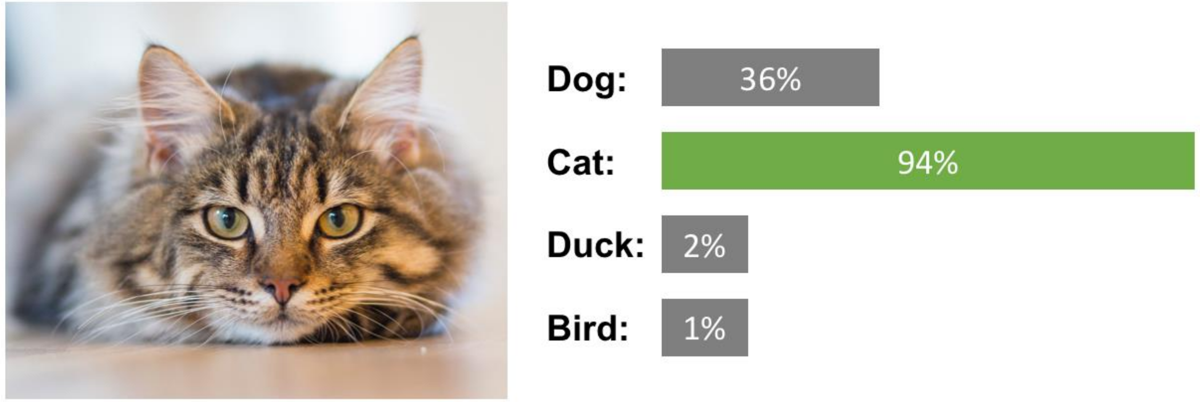
$55500

???

Classification

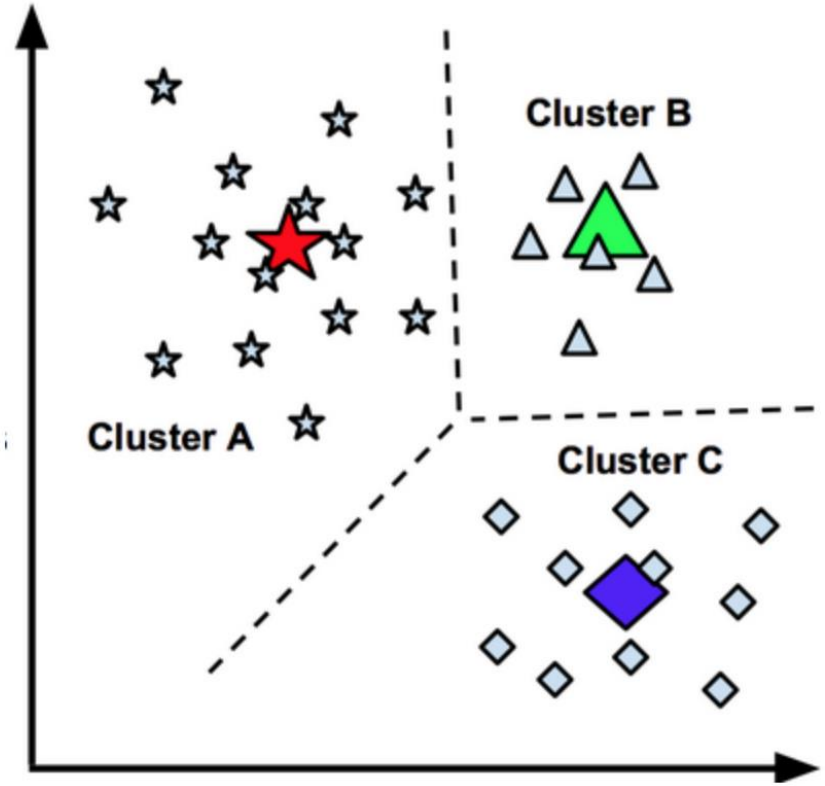
- categorize new inputs as belonging to one of a set of categories → Predict results within *a discrete output (categories)*





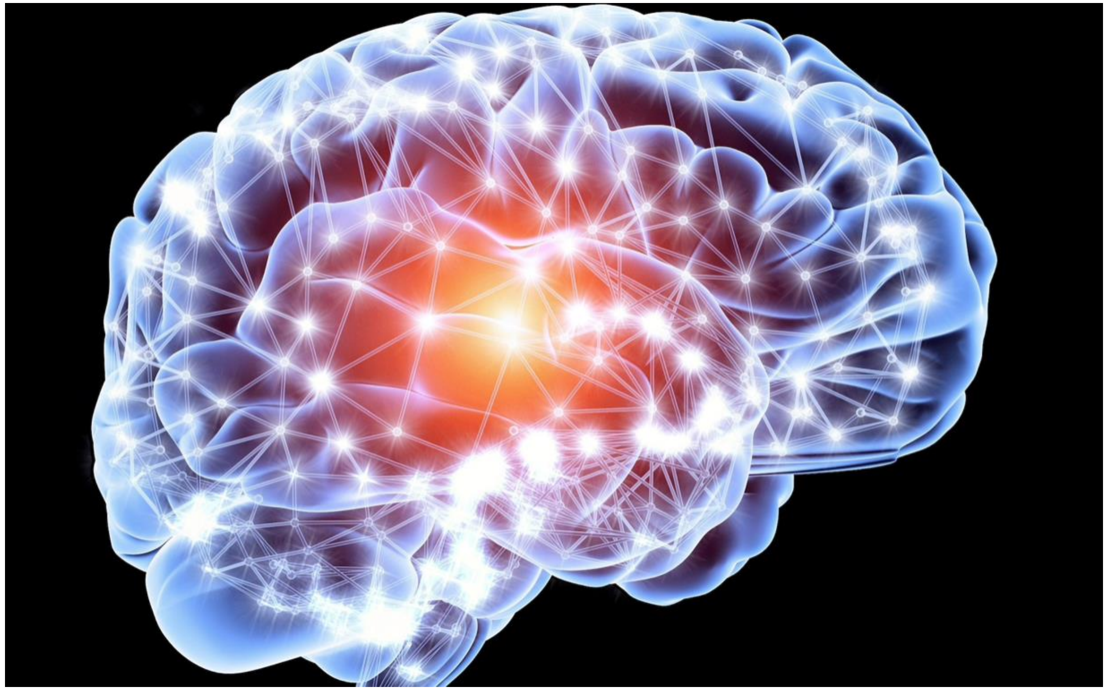
Clustering

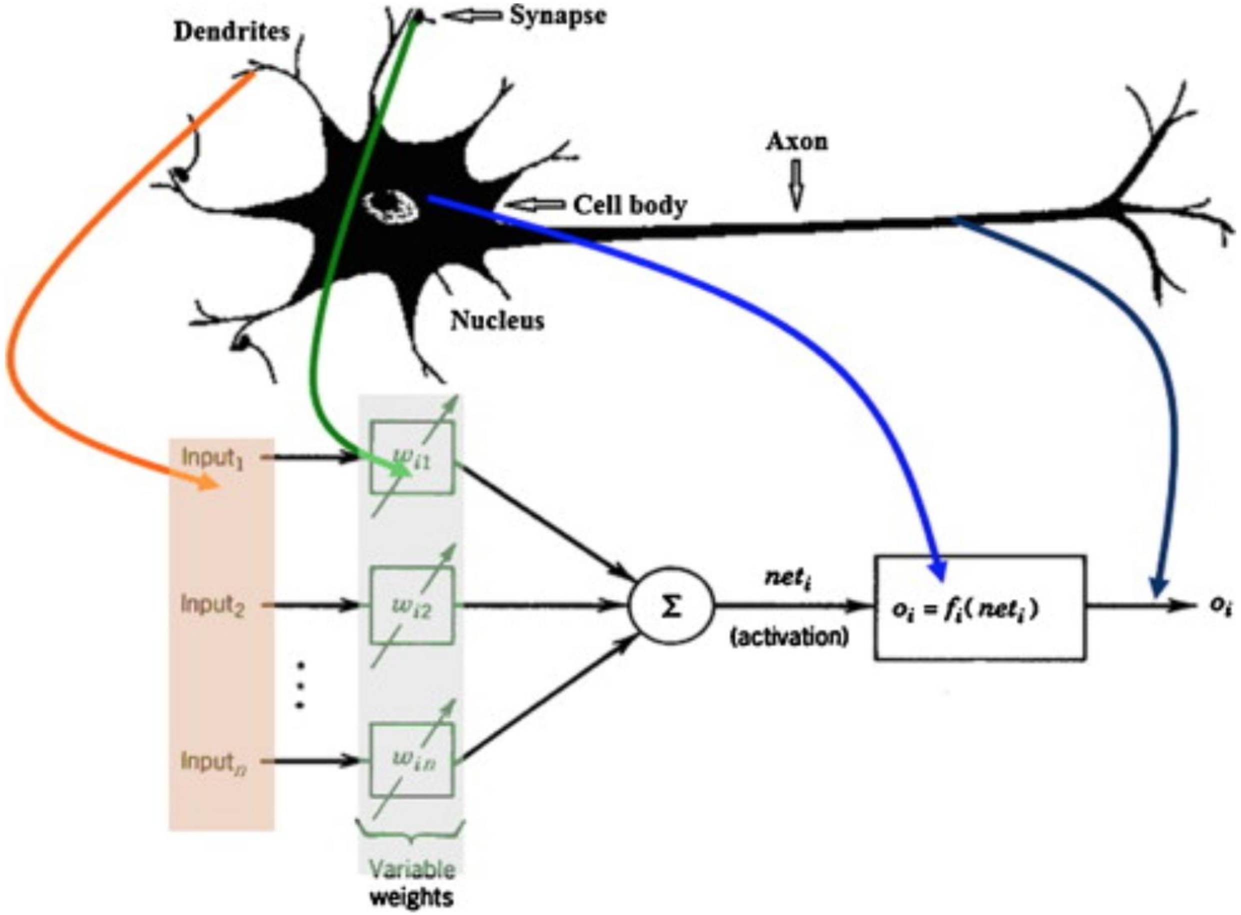
- create a set of categories, for which individual data instances have a set of common or similar characteristics.



Neural Network (NN)

- Learning algorithm inspired by *how the brain works*

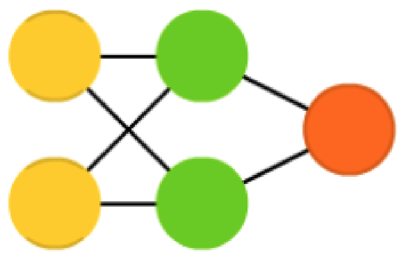




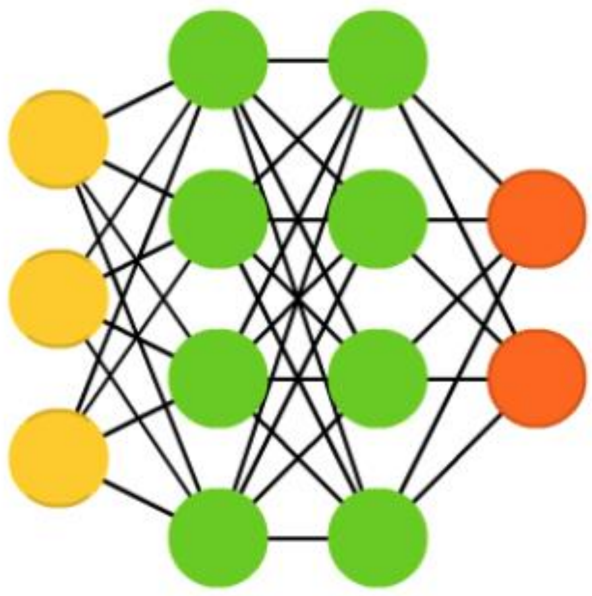
(Deep) Feedforward NN (DFF)

- 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)



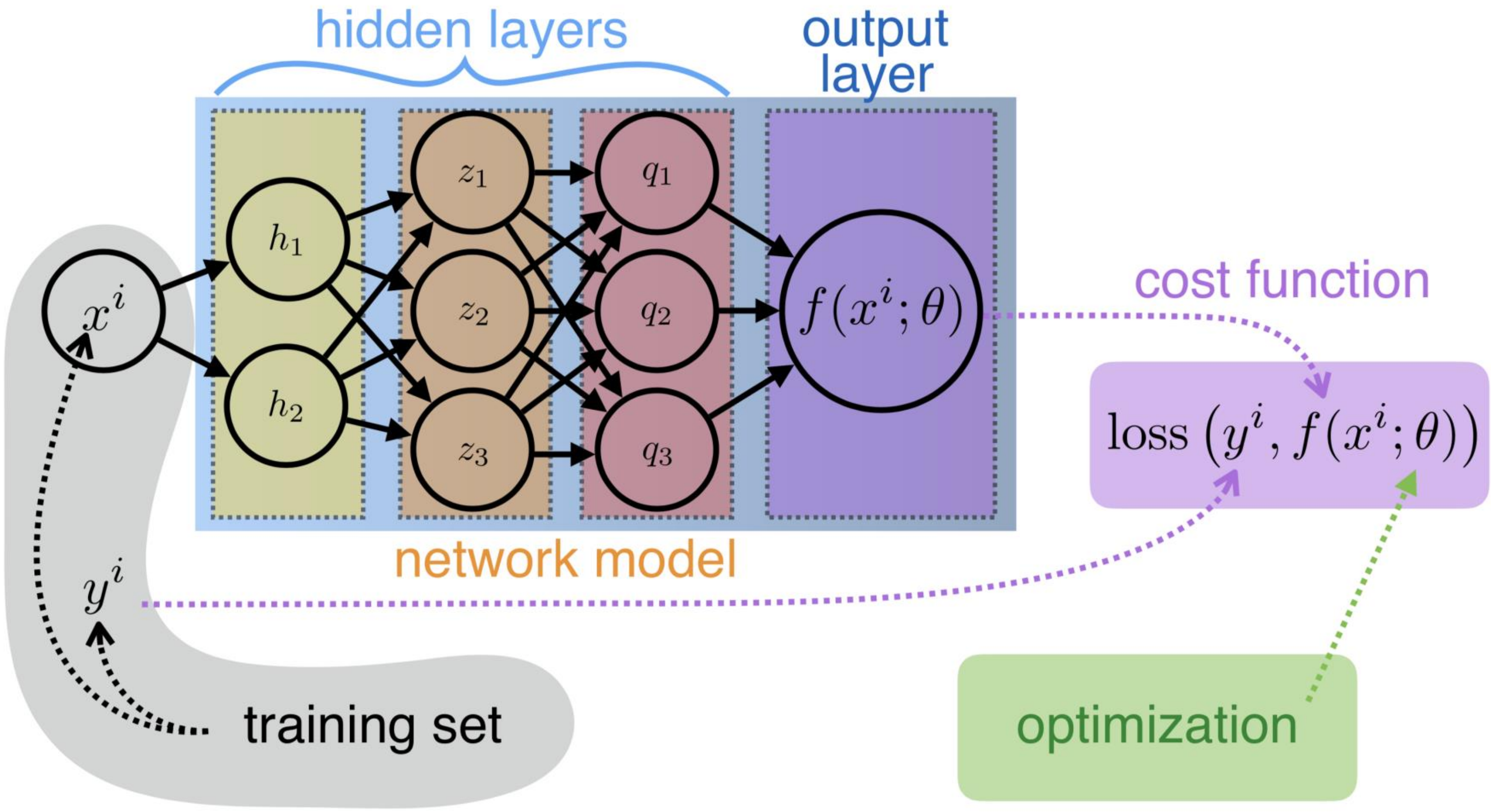
Deploying a Neural Network

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

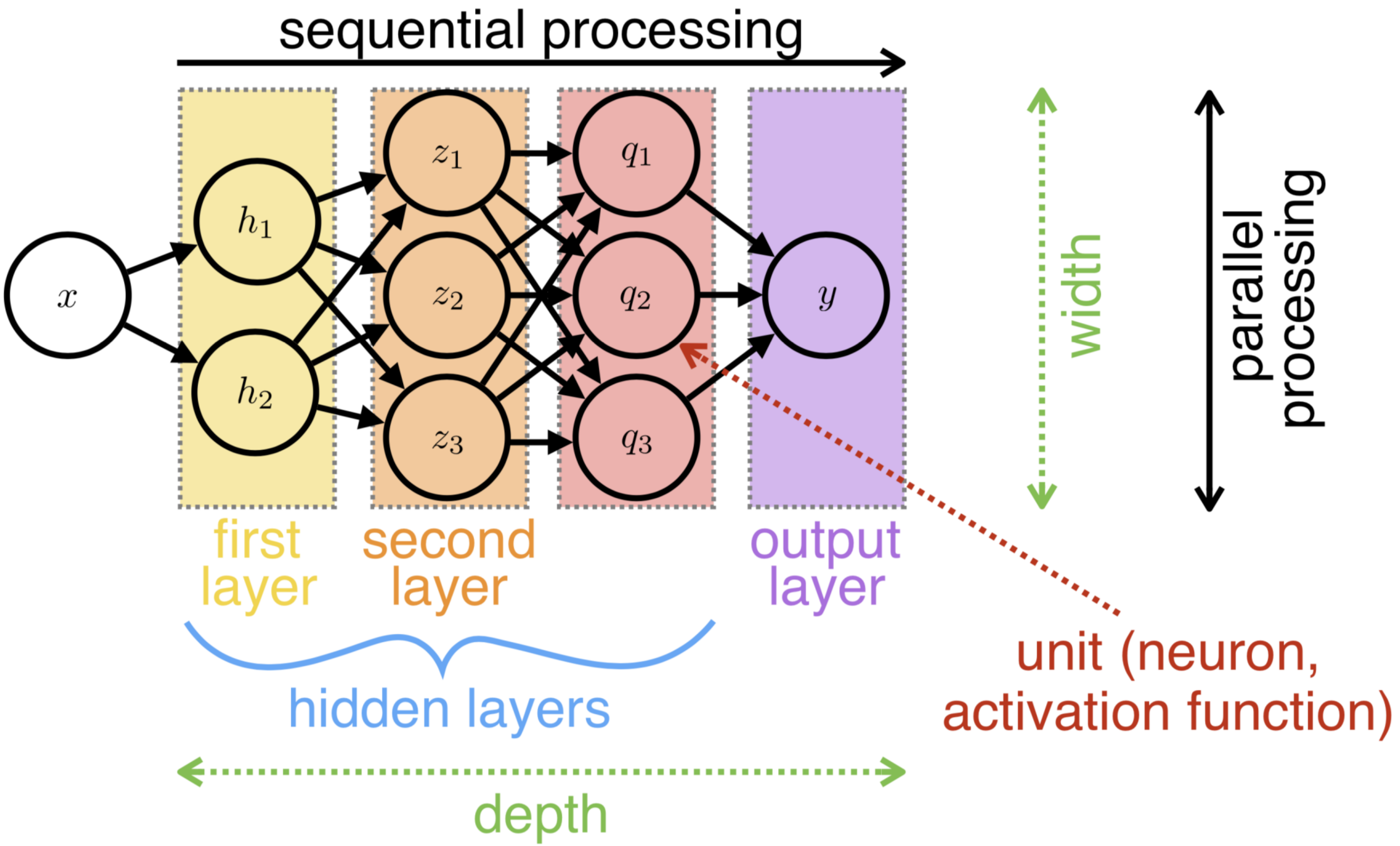
1) **Network model**

2) **Cost function (/objective/loss function)**

3) **Optimization**

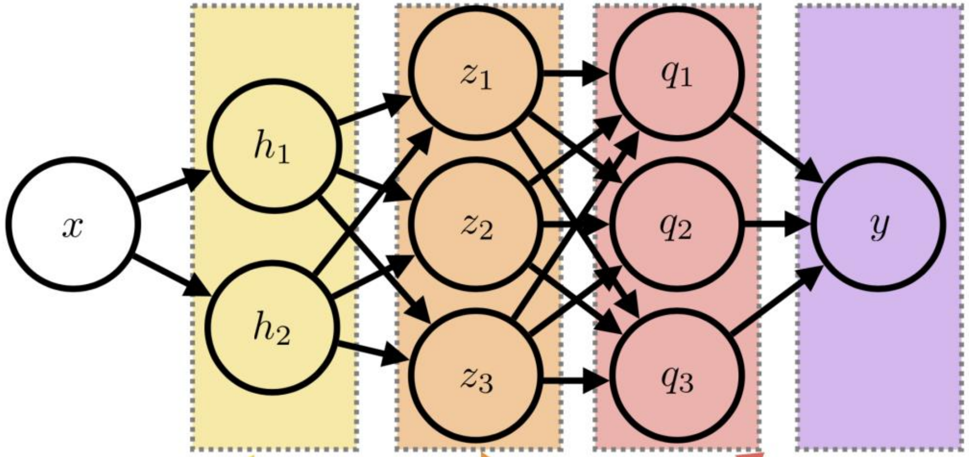


Network model



1) Activation functions

Different types of activation functions for the hidden layers and the output layer



$$y = f_4(q_1, q_2, q_3)$$

$$\boxed{y = f_4(f_{3,1}(f_{2,1}(f_{1,1}(x), f_{1,2}(x)), \dots), \dots)}$$

Hierarchical representation

$$h_1 = f_{1,1}(x)$$
$$h_2 = f_{1,2}(x)$$

$$z_1 = f_{2,1}(h_1, h_2)$$
$$z_2 = f_{2,2}(h_1, h_2)$$
$$z_3 = f_{2,3}(h_1, h_2)$$

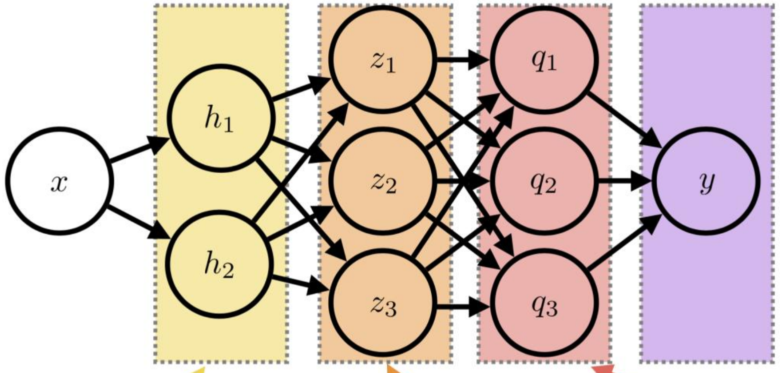
$$q_1 = f_{3,1}(z_1, z_2, z_3)$$
$$q_2 = f_{3,2}(z_1, z_2, z_3)$$
$$q_3 = f_{3,3}(z_1, z_2, z_3)$$

Fully connected $\boxed{f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}}$

Weights w and bias b parameters to optimize

Activation functions

Different types of activation functions for the hidden layers and the output layer



$$y = f_4(q_1, q_2, q_3)$$

$$h_1 = f_{1,1}(x)$$
$$h_2 = f_{1,2}(x)$$

$$z_1 = f_{2,1}(h_1, h_2)$$
$$z_2 = f_{2,2}(h_1, h_2)$$
$$z_3 = f_{2,3}(h_1, h_2)$$

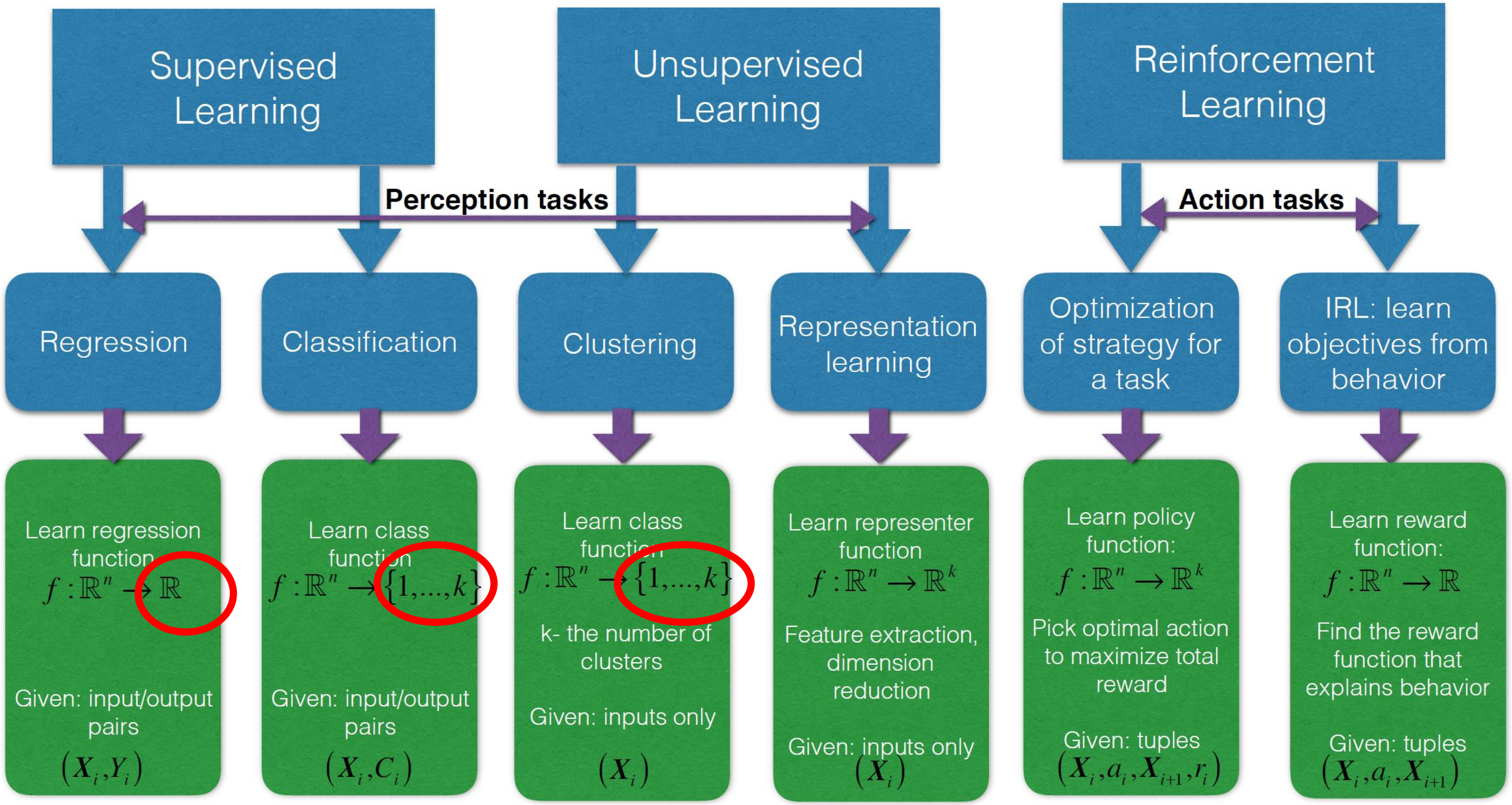
$$q_1 = f_{3,1}(z_1, z_2, z_3)$$
$$q_2 = f_{3,2}(z_1, z_2, z_3)$$
$$q_3 = f_{3,3}(z_1, z_2, z_3)$$

(3,1) **(3,2)** **(2,1)**

$$Z = W_z H$$

Fully connected $\boxed{f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}}$

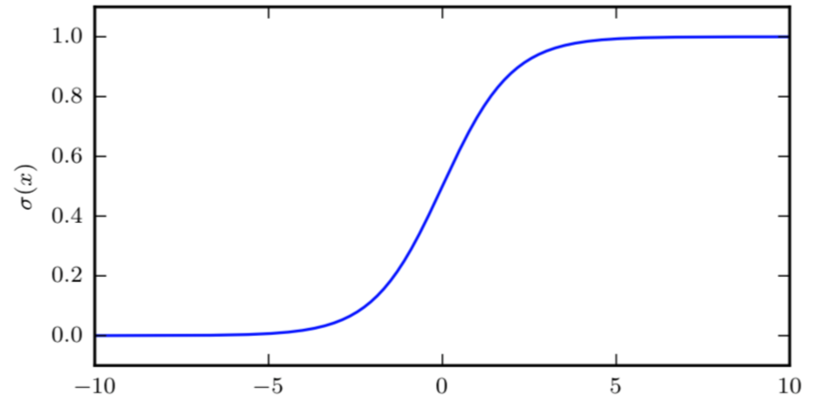
Neural Network Outputs



Output layer : activation functions

1) Classification: probability vector

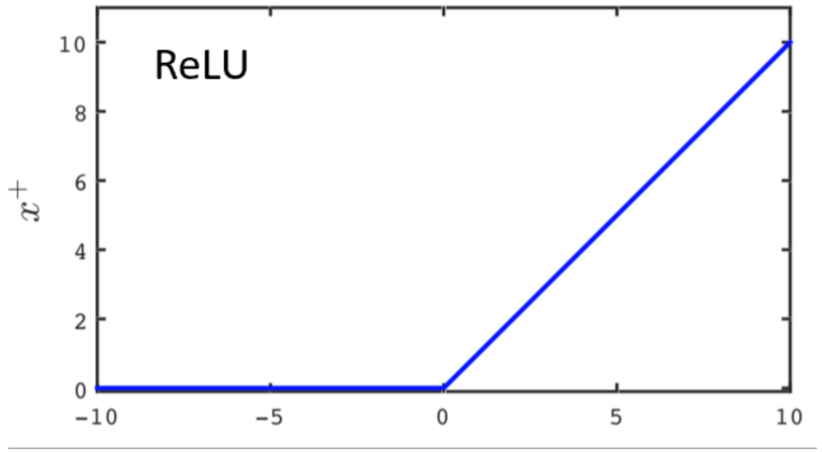
- Sigmoid (binary class)
- Softmax (multiple class)



$$Z = \sigma(W_z H)$$

2) Regression: mean estimate

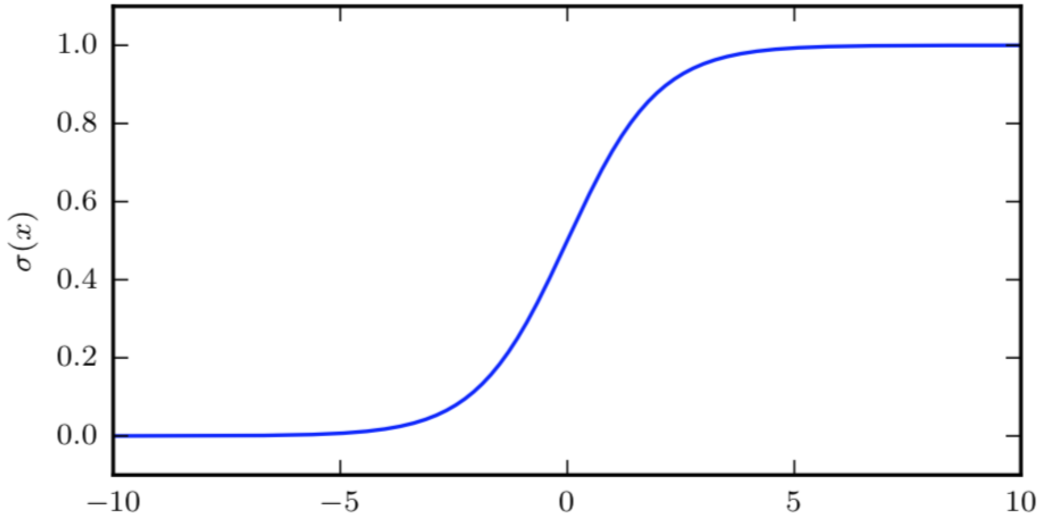
- No activation
- ReLU
- Softplus
- Smoothed max
- Generalization of ReLU (leaky ReLU,...)



Sigmoid and softmax

Sigmoid (*two-class* classifier) :

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$



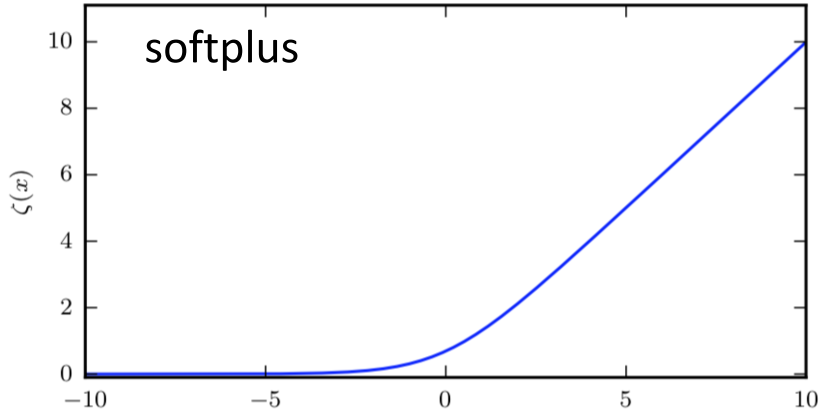
Softmax (*multi-class* classifier) :

$$\boxed{\mathrm{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}}$$

ReLU, softplus and smoothed max

Softplus (smooth approx. of ReLU) :

$$\zeta(x) = \log(1 + \exp(x))$$

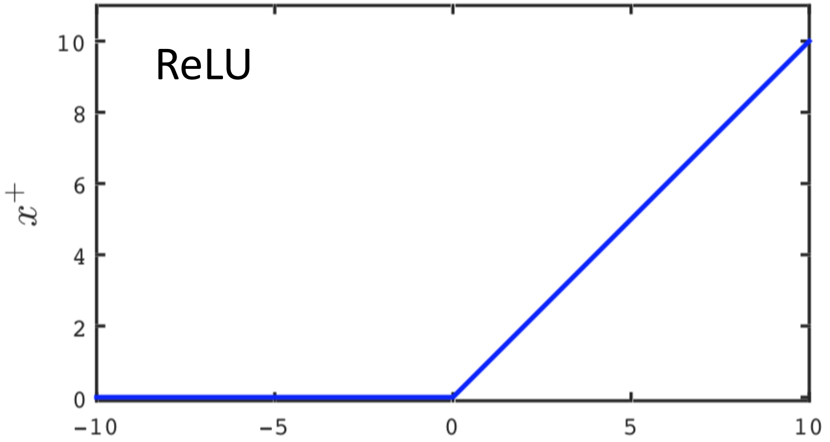


Smoothed max (*extension* of softplus) :

$$\zeta(x) = \log \sum_j \exp(x_i)$$

ReLU (Rectified Linear Unit) :

$$x^+ = \max(0, x)$$



2) Loss and Cost functions

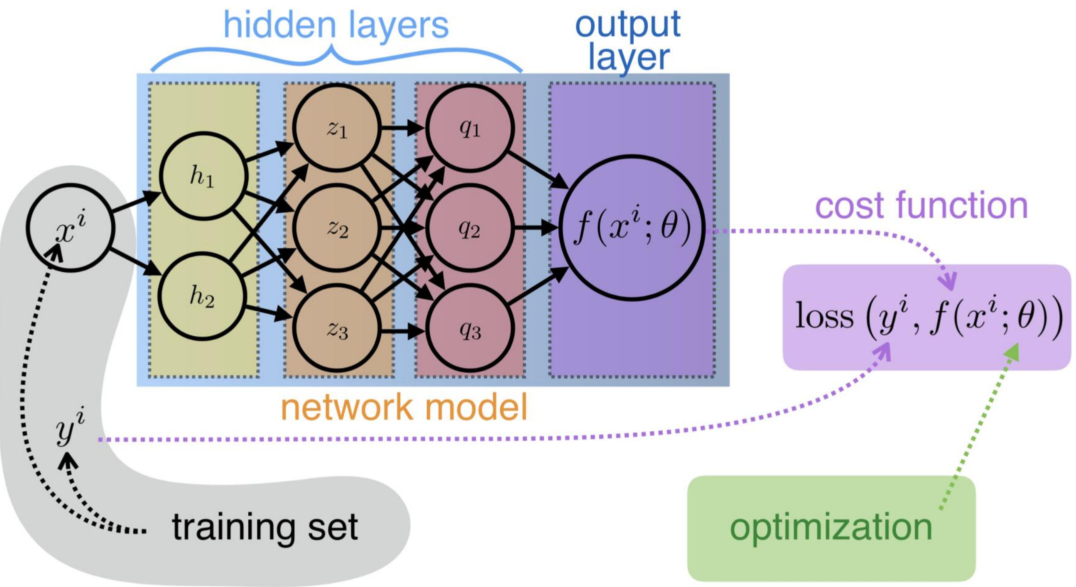
- Loss function $L\left(\hat{y}^{(i)}, y^{(i)}\right)$, also called error function, measures how different the prediction $\hat{y} = f(x)$ and the desired output y are

- 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)*
- Next, we *optimize the network parameters θ* (training)
- In addition, we have to set values for hyperparameters

Loss function choice

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

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

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

(binary classification)

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

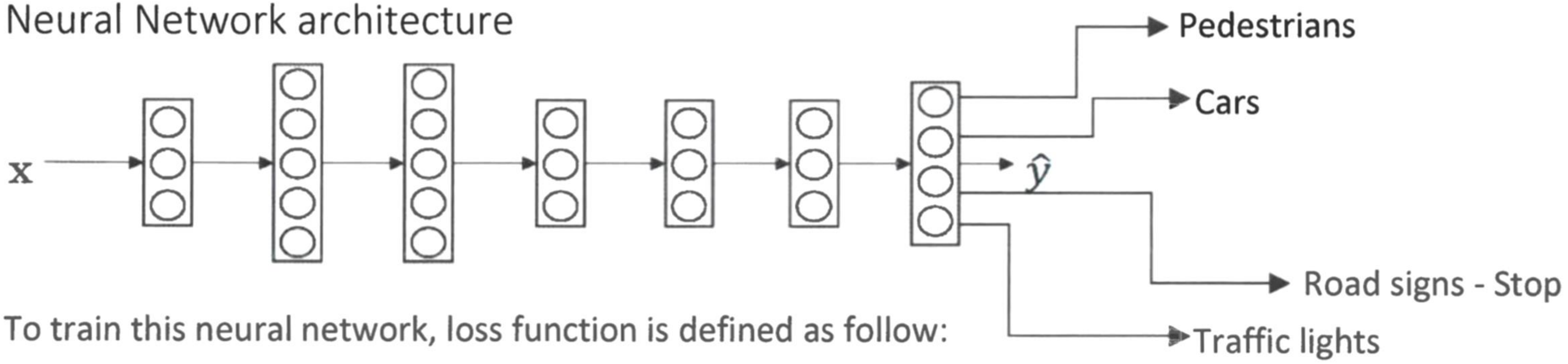
$$\hat{y} = W^\top h + b \qquad p(y|\hat{y}) = \mathrm{N}(y; \hat{y})$$

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

Loss function example

- NN does simultaneously several tasks (multi-task)

Neural Network architecture



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

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

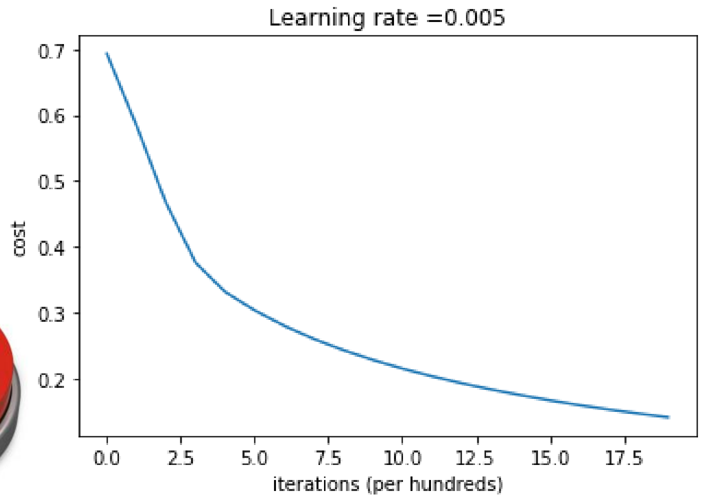
Training

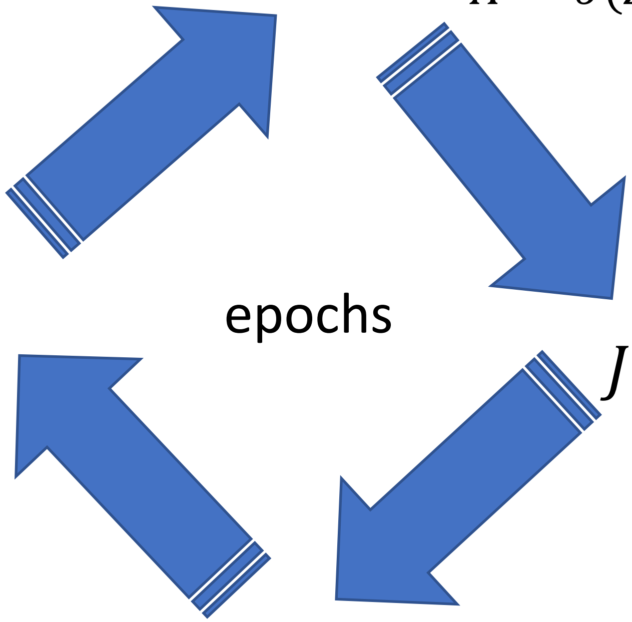
- *Iterative* process

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

$A = \sigma(Z)$

Learning curve





Parameter update
(gradient descent)

epochs

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

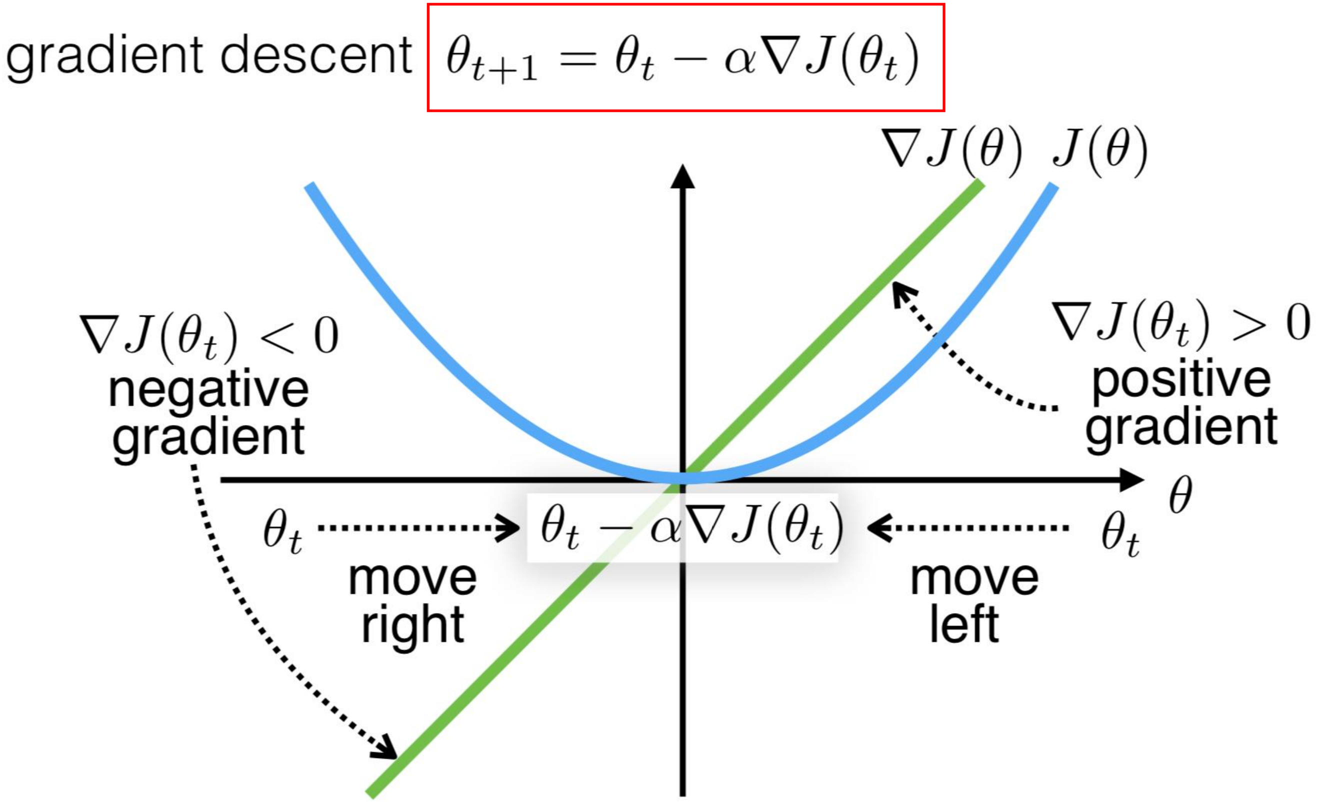
learning rate α

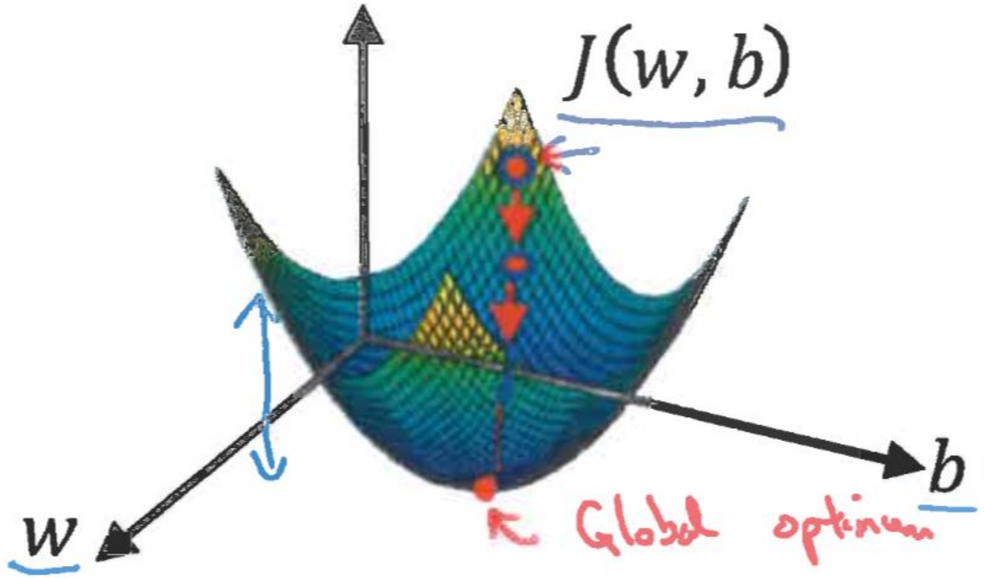
$$\boxed{\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)}$$

Backward propagation
(dJ/dw, dJ/db)

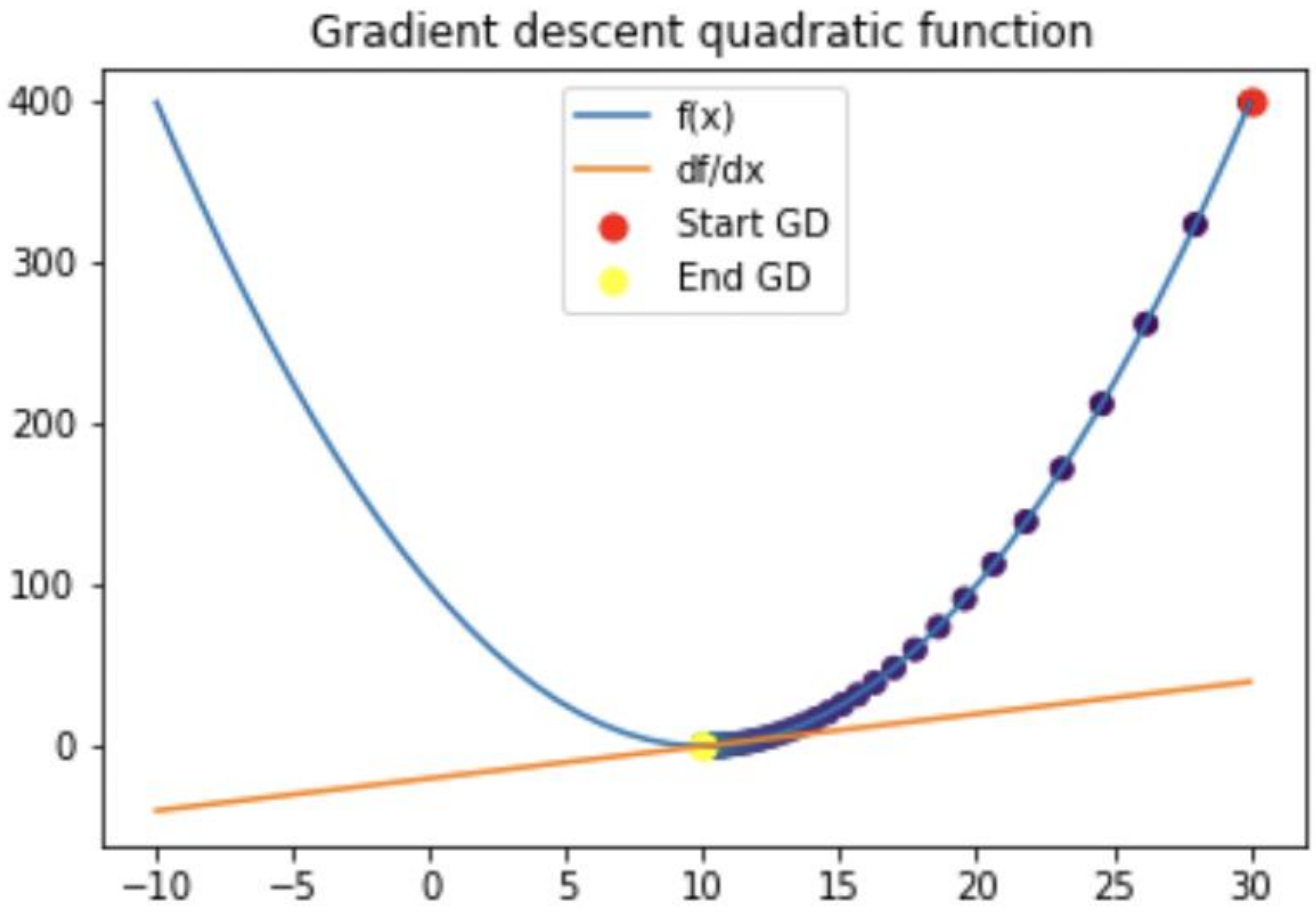
Gradient Descent

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

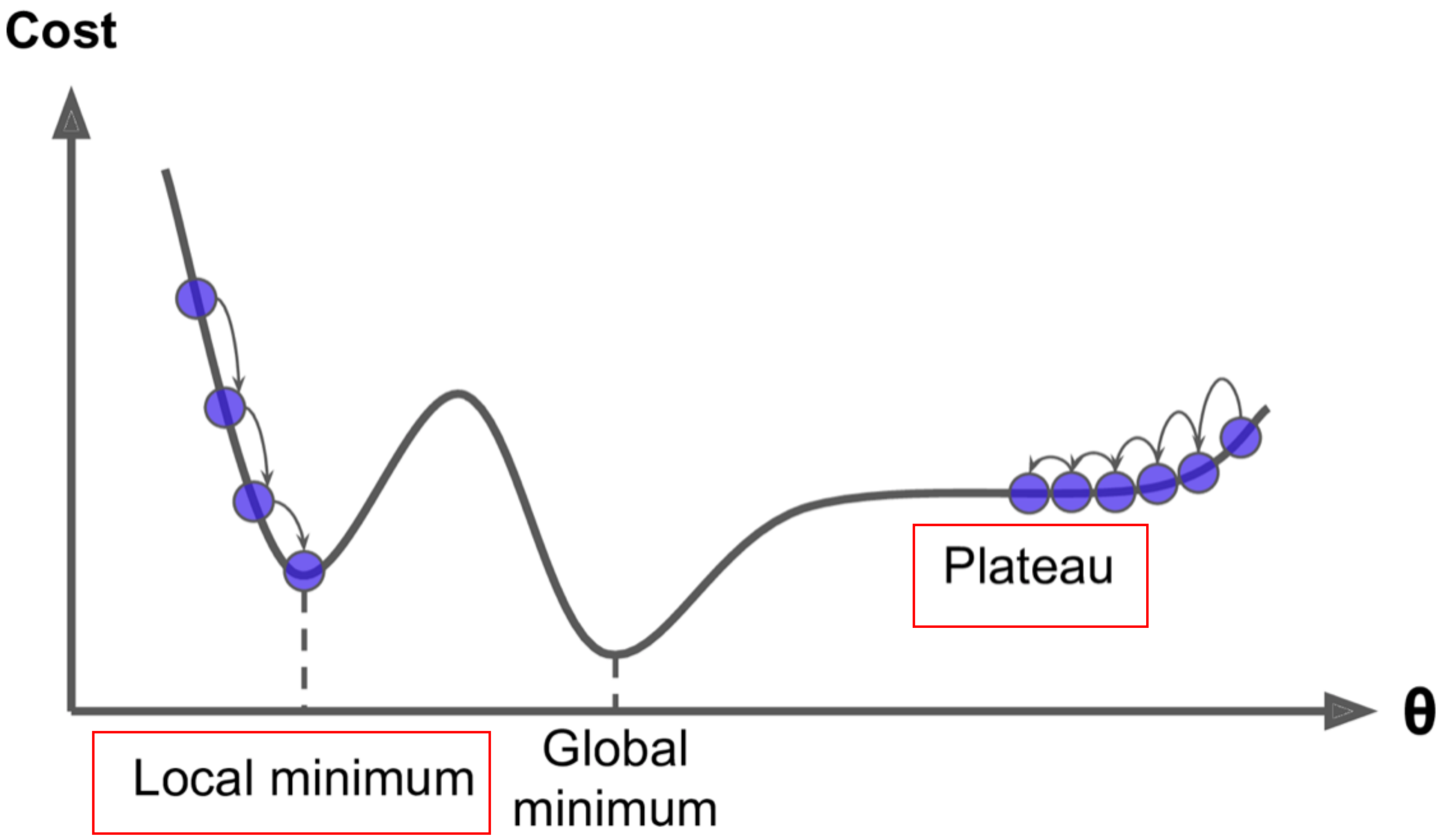




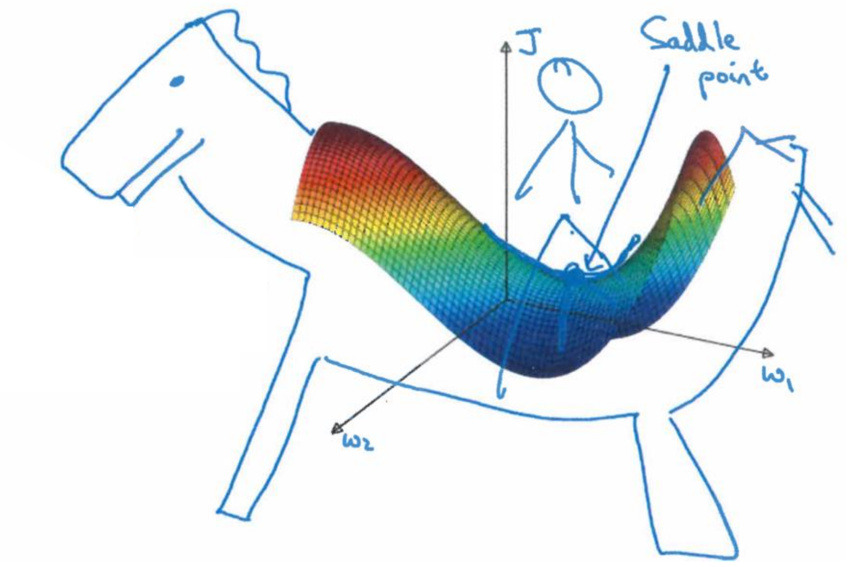
$$\nabla J(w) = \frac{dJ(w,b)}{dw} \qquad \nabla J(b) = \frac{dJ(w,b)}{db}$$


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

Optimization pitfalls



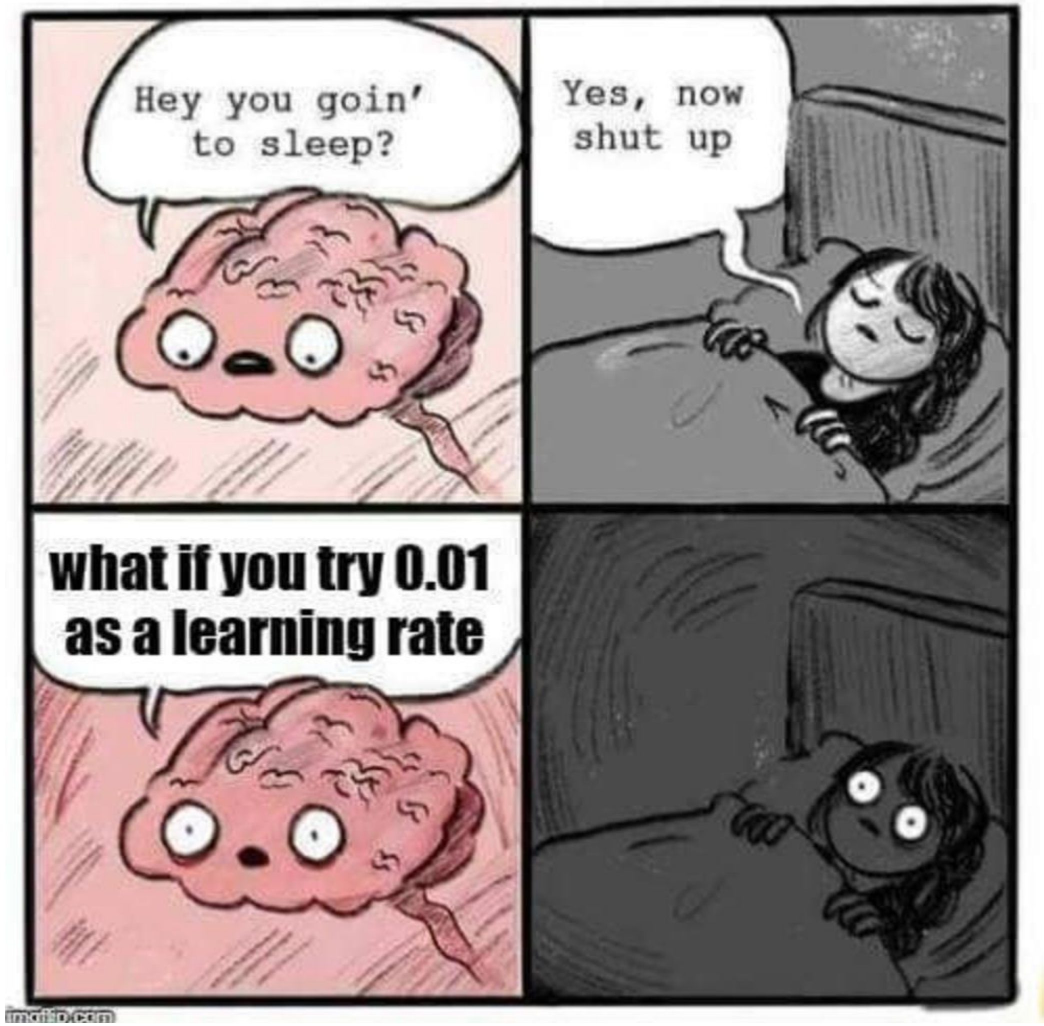
Saddle point



Hyperparameters

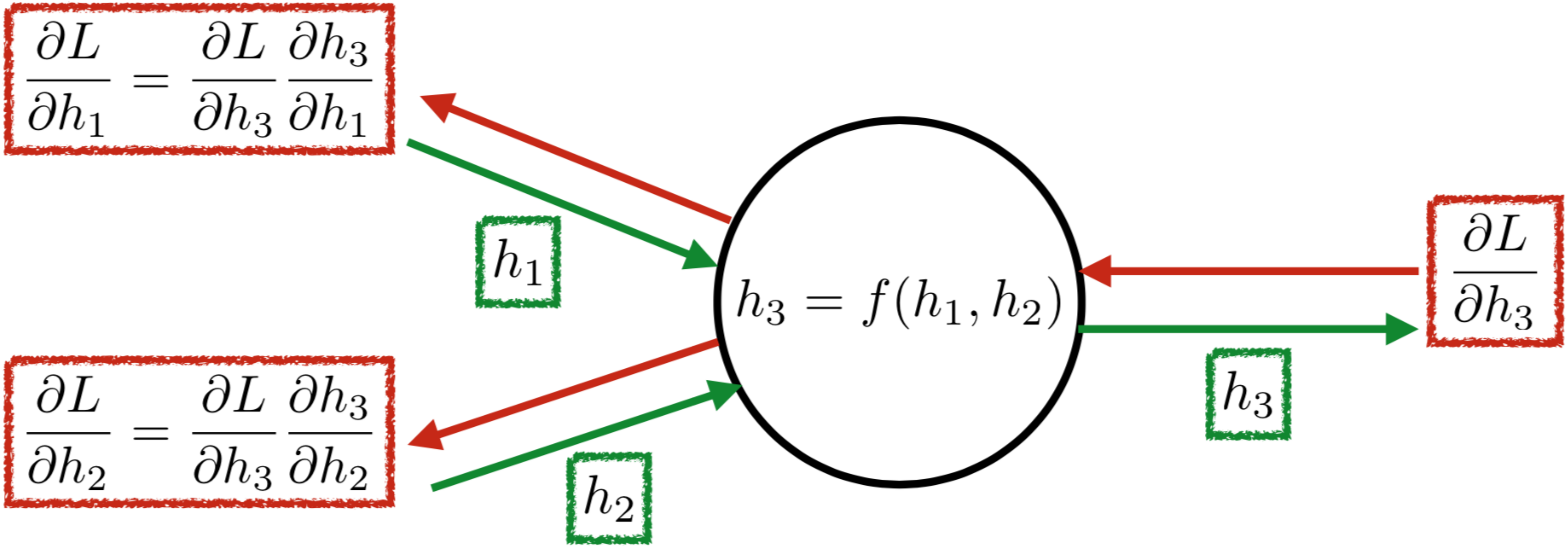
- Parameters that cannot be learned directly from training data

- 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 !*



Backpropagation

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



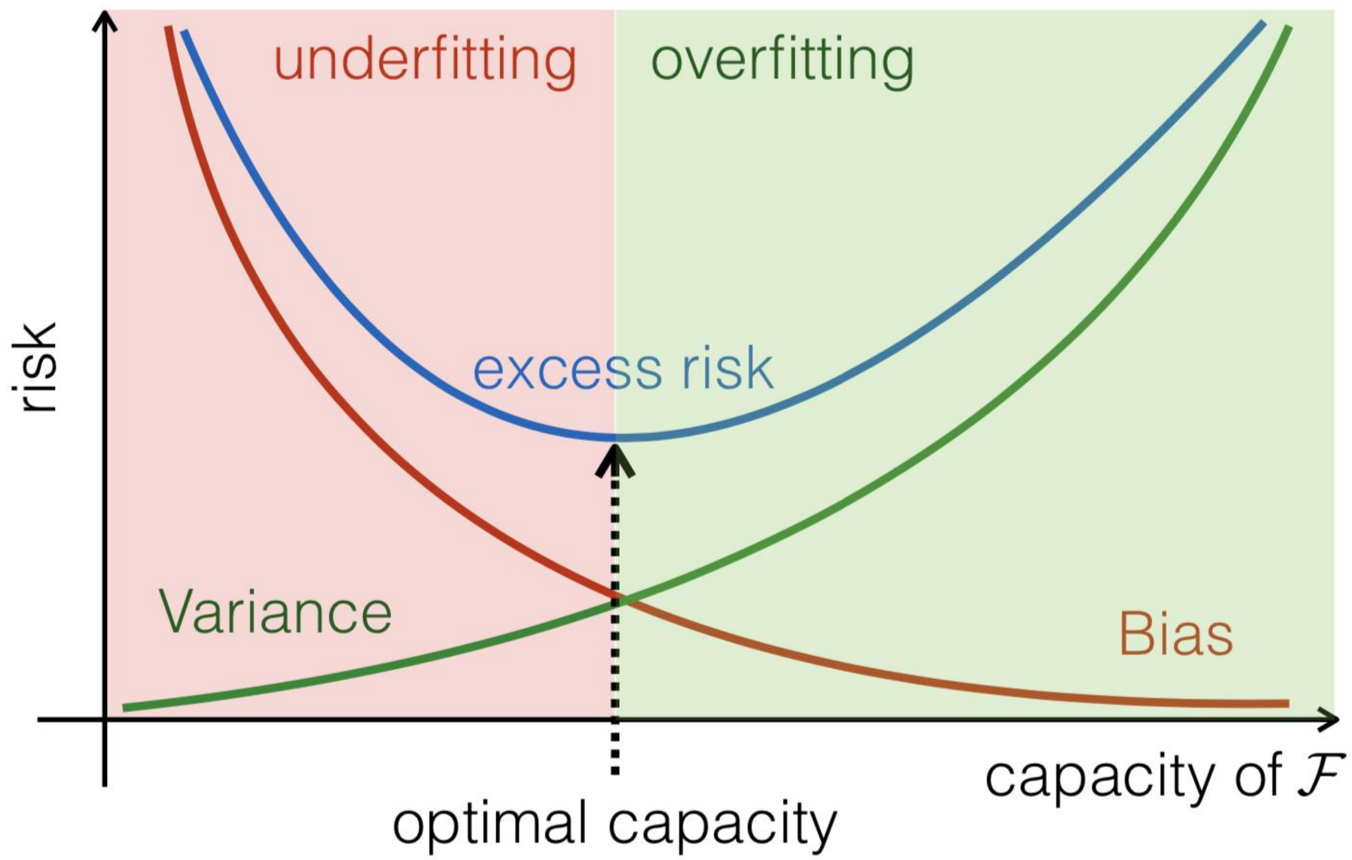
Performance Measure P

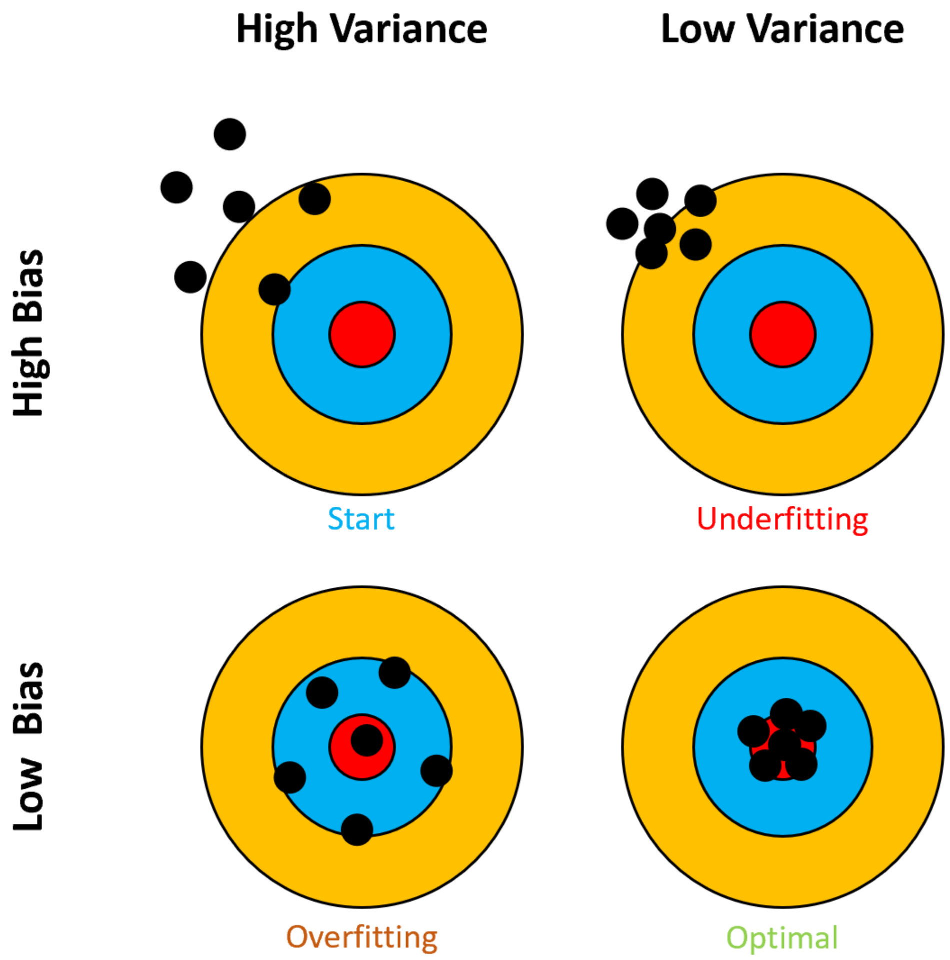
- To evaluate a ML algorithm, we need a way to measure **how well it performs on the task**

- It is measured **on a separate set** (test set) from what we use to build the function f (training set)

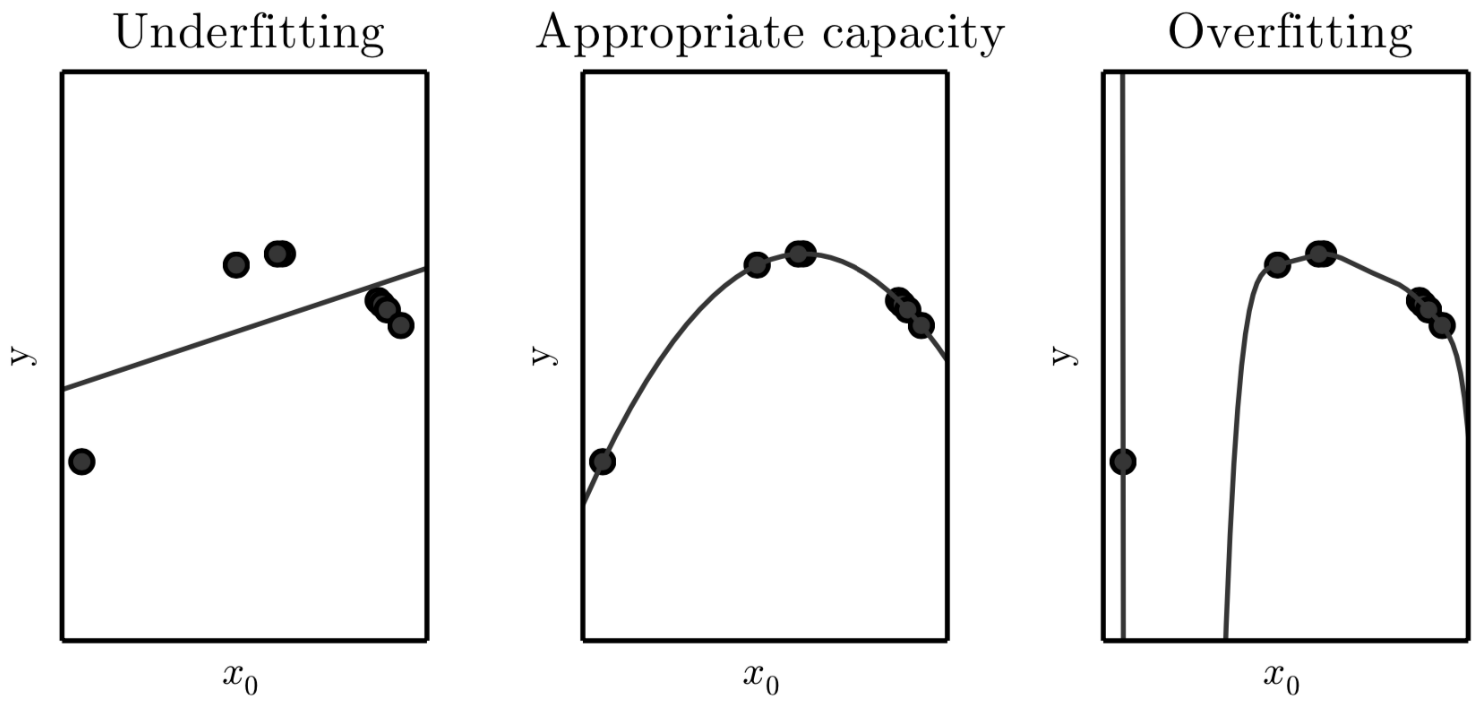
- **Examples** :
  - Classification accuracy (portion of correct answers)
  - Error rate (portion of incorrect answers)
  - Regression accuracy (e.g. least squares errors)

Bias and Variance - Overfitting and Underfitting



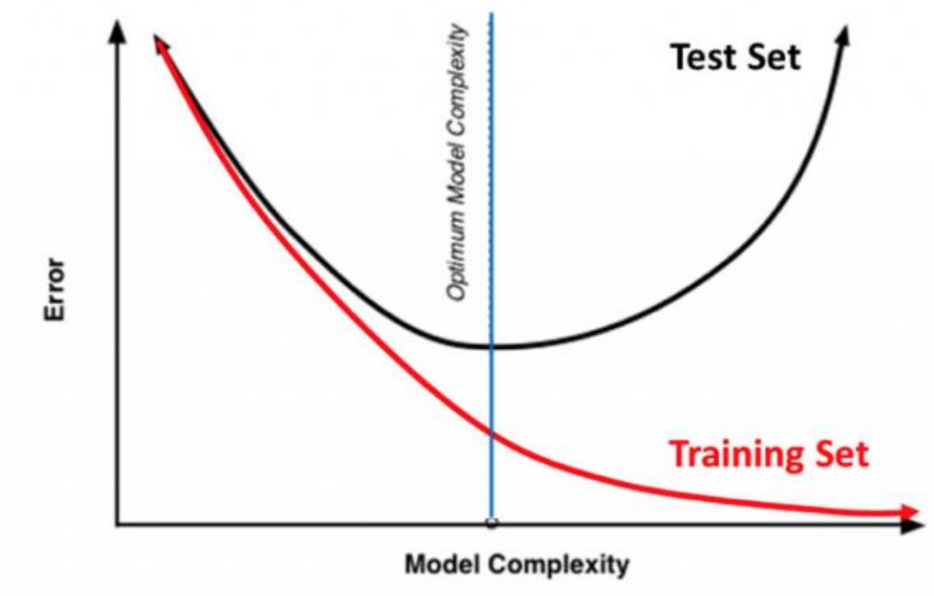


Overfitting and Underfitting



High bias

High variance



Case

- You want to find cats in images

- Classification error (the portion of wrong answers) used as an evaluation metric

Algorithm	Classification error (%)
A	**3%**
B	**5%**

➢ *Which one is best ?*

First of all, understand your data !

- Carry out manual error analysis
  - Look at *mislabeled development set* examples (*do not look at test set*)
  - For example: check by hand 500 pictures (incorrect labels ? Foggy pictures ? Other causes ?)
- Clean up incorrectly labeled data
  - Apply the same process to your dev and test sets!