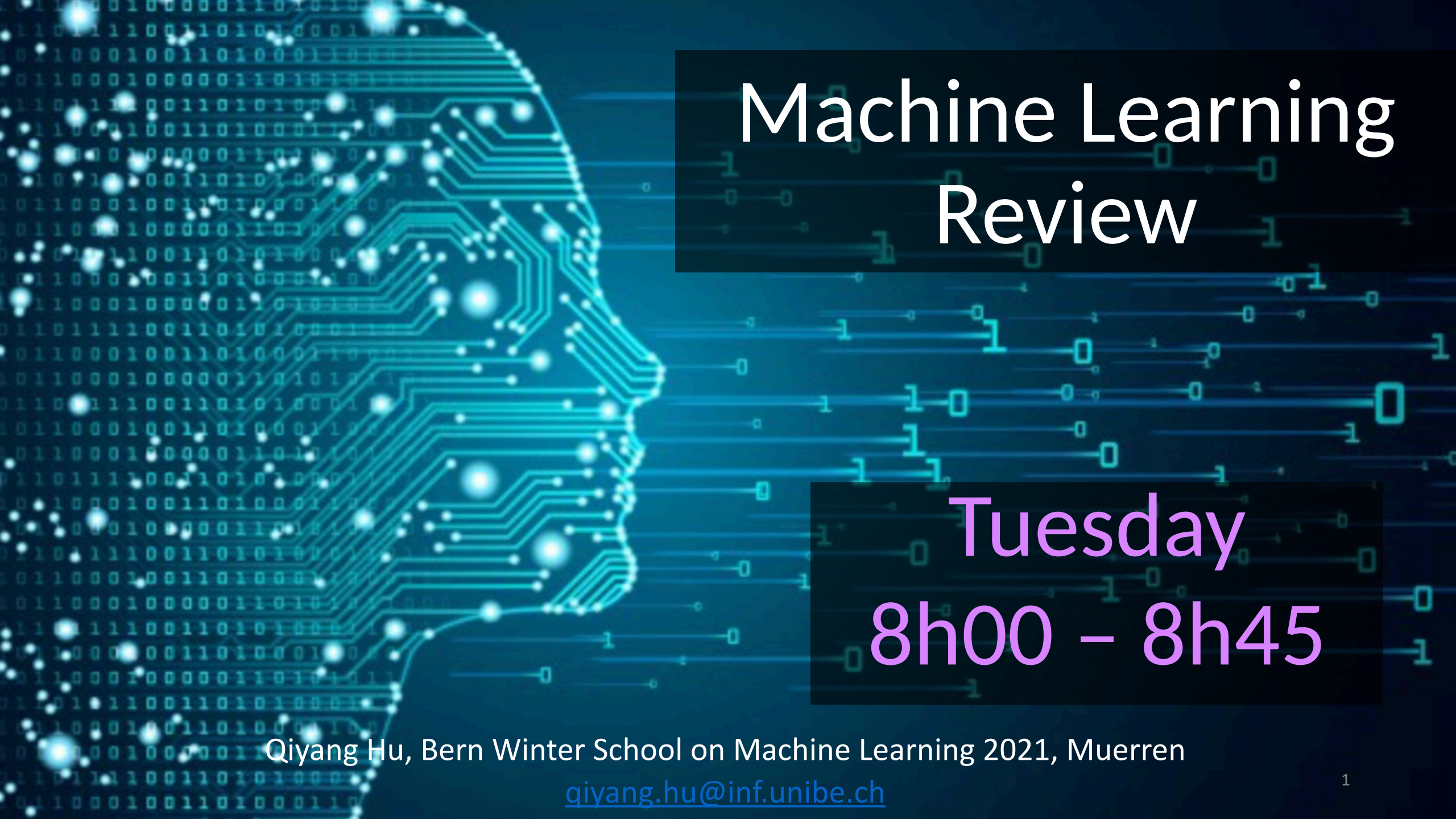
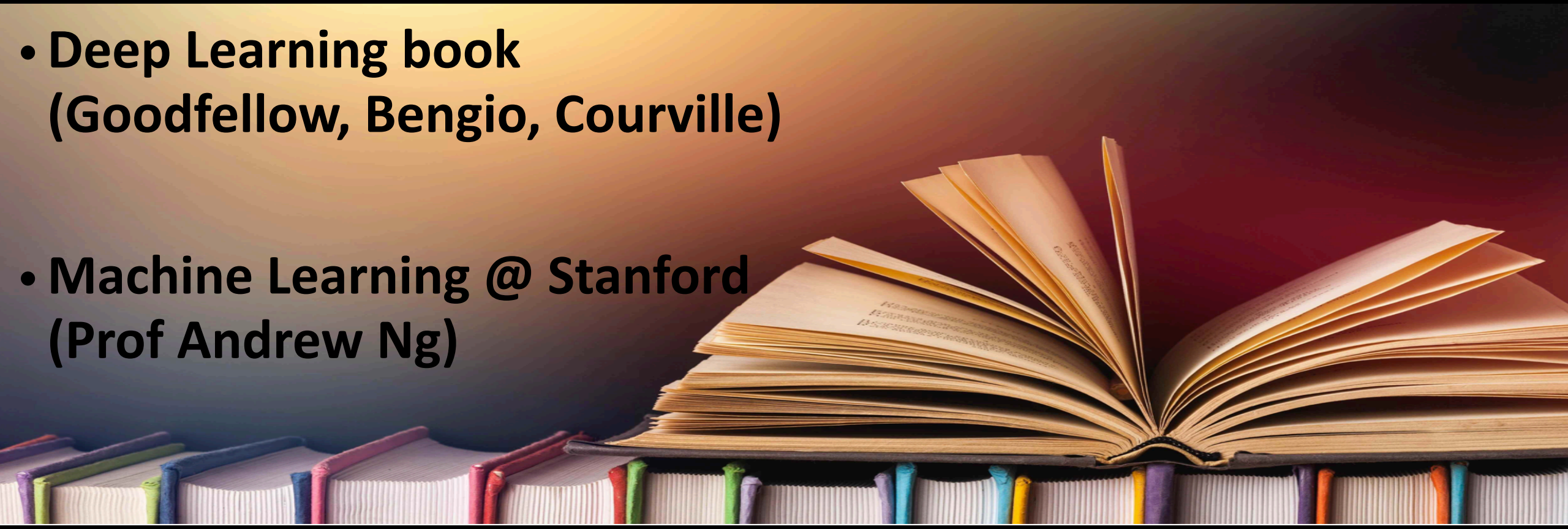


Machine Learning Review

Tuesday
8h00 – 8h45

Qiyang Hu, Bern Winter School on Machine Learning 2021, Muerren
qiyang.hu@inf.unibe.ch

Bibliography



- **Deep Learning book (Goodfellow, Bengio, Courville)**
- **Machine Learning @ Stanford (Prof Andrew Ng)**

What is Machine Learning ?

"Can machines do what we (as thinking entities) can do?"
(Turing)

Let’s warm up !

- Can you list **2 examples** of Machine Learning use in society ?
    1.
    2.
- Is Machine Learning used in your company/institution ? If yes, can you cite **1 use case** ?

- Do you know any “technical terms” about Machine Learning (algorithm names, …) ? If yes, can you list **up to 3 keywords** ?

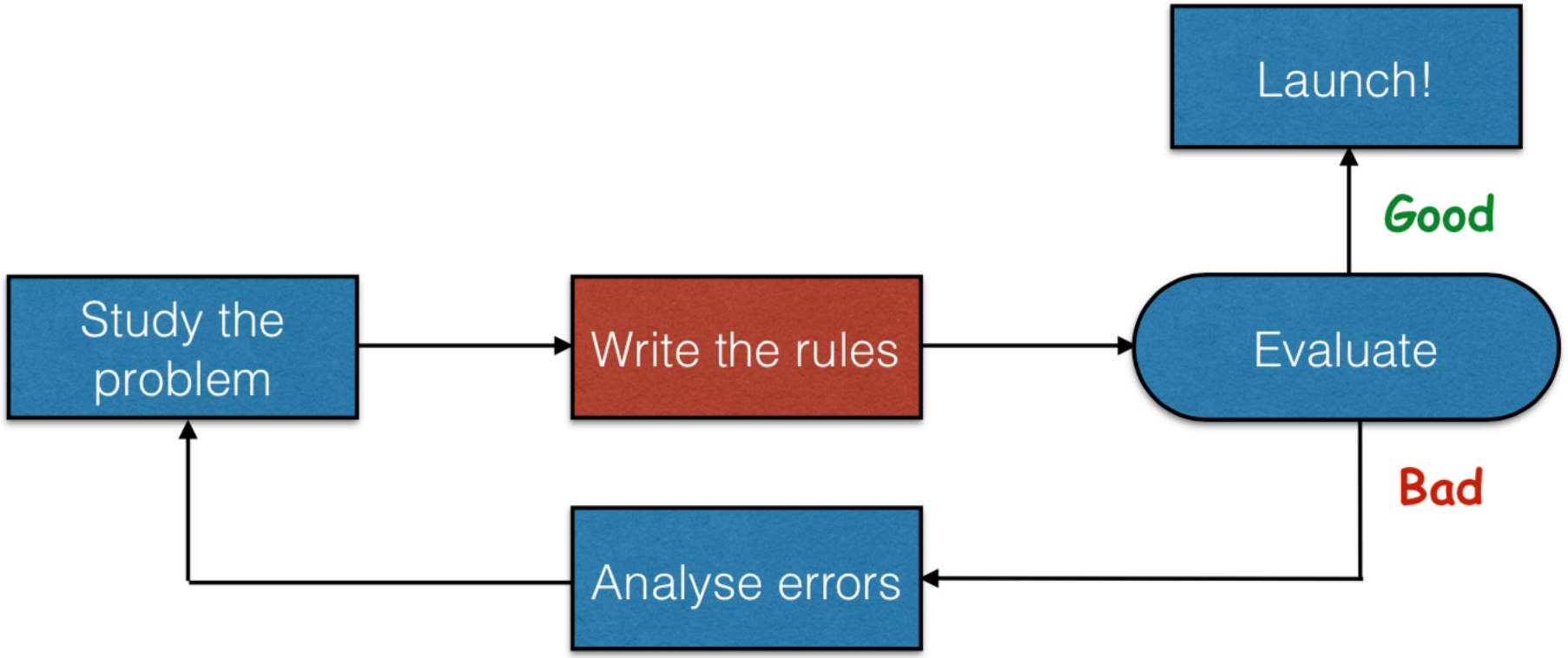
- How do you **feel** about Machine Learning and AI in general ?

Machine Learning Definition… in words

« A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.»

Tom M. Mitchell (1997)

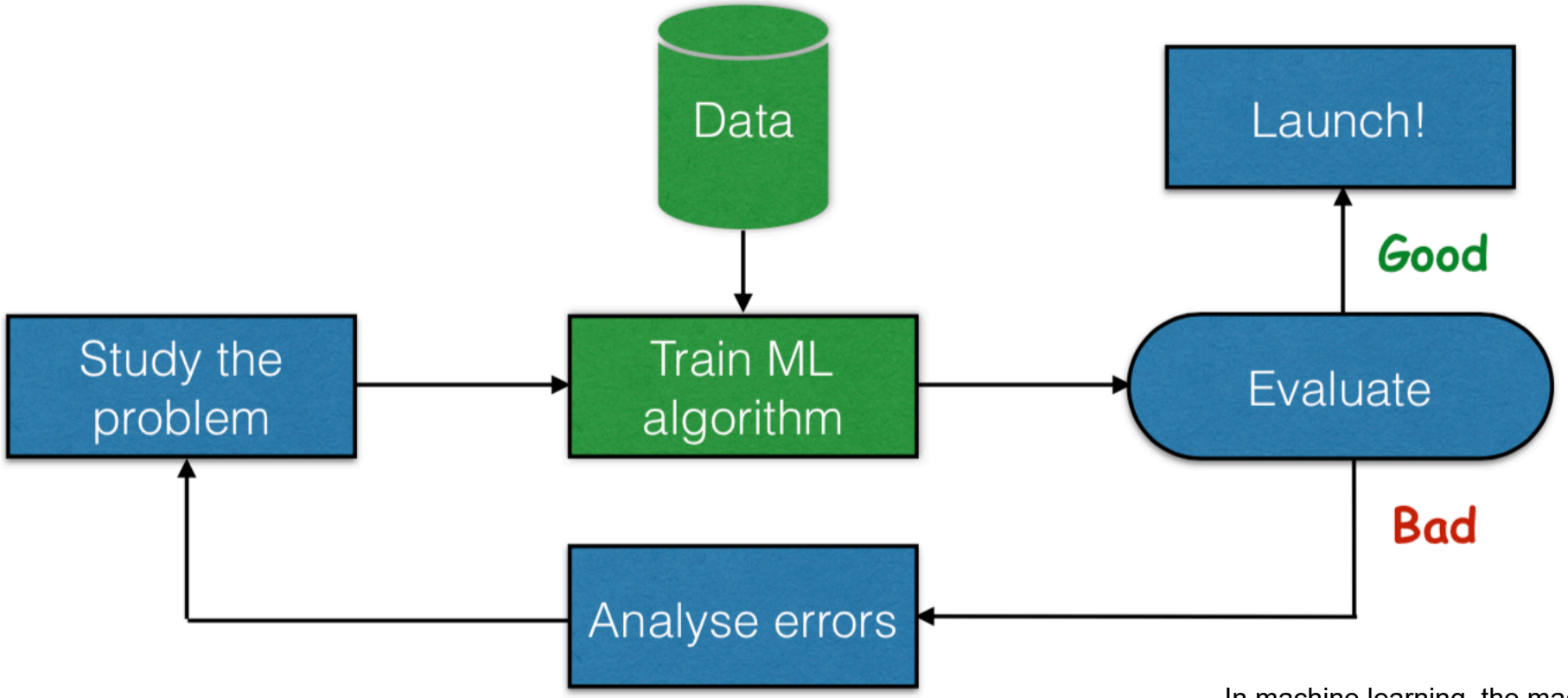
Traditional approach (Software 1.0)



Traditionally you give the machine an algorithm by which to make decisions.

List of all the knowledge and formal rules

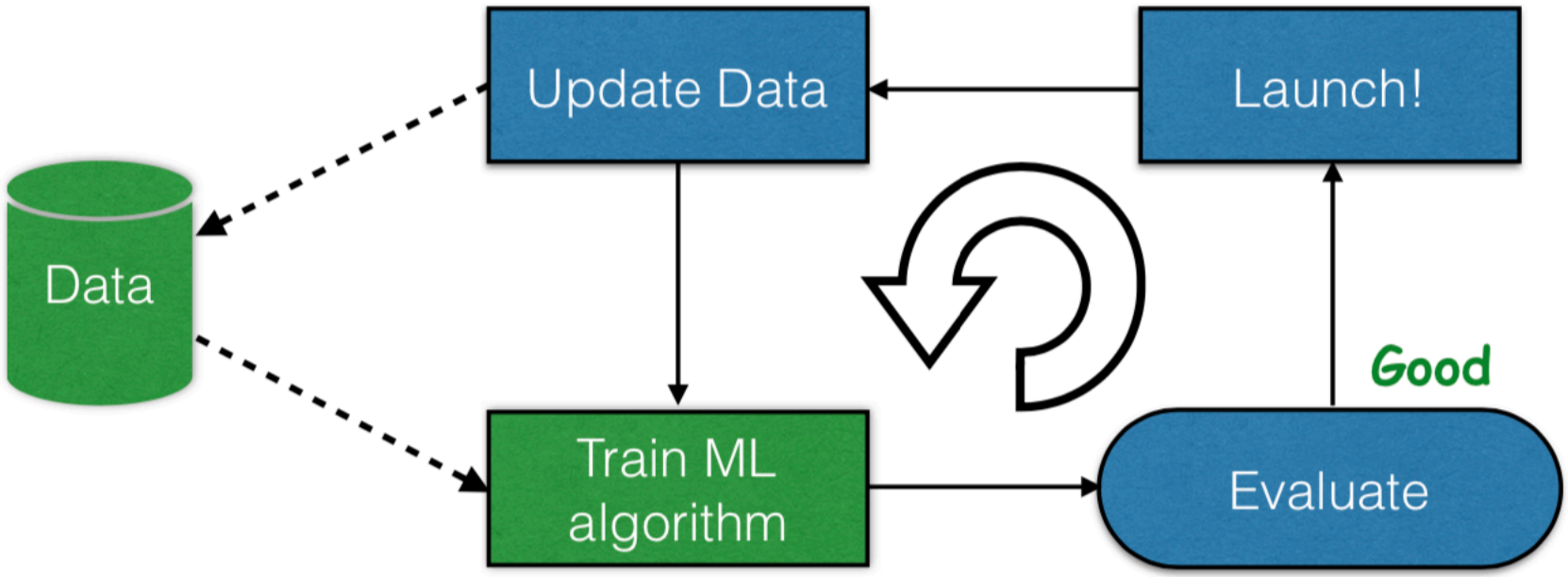
Machine Learning approach (Software 2.0)



In machine learning, the machine creates the algorithm by itself.

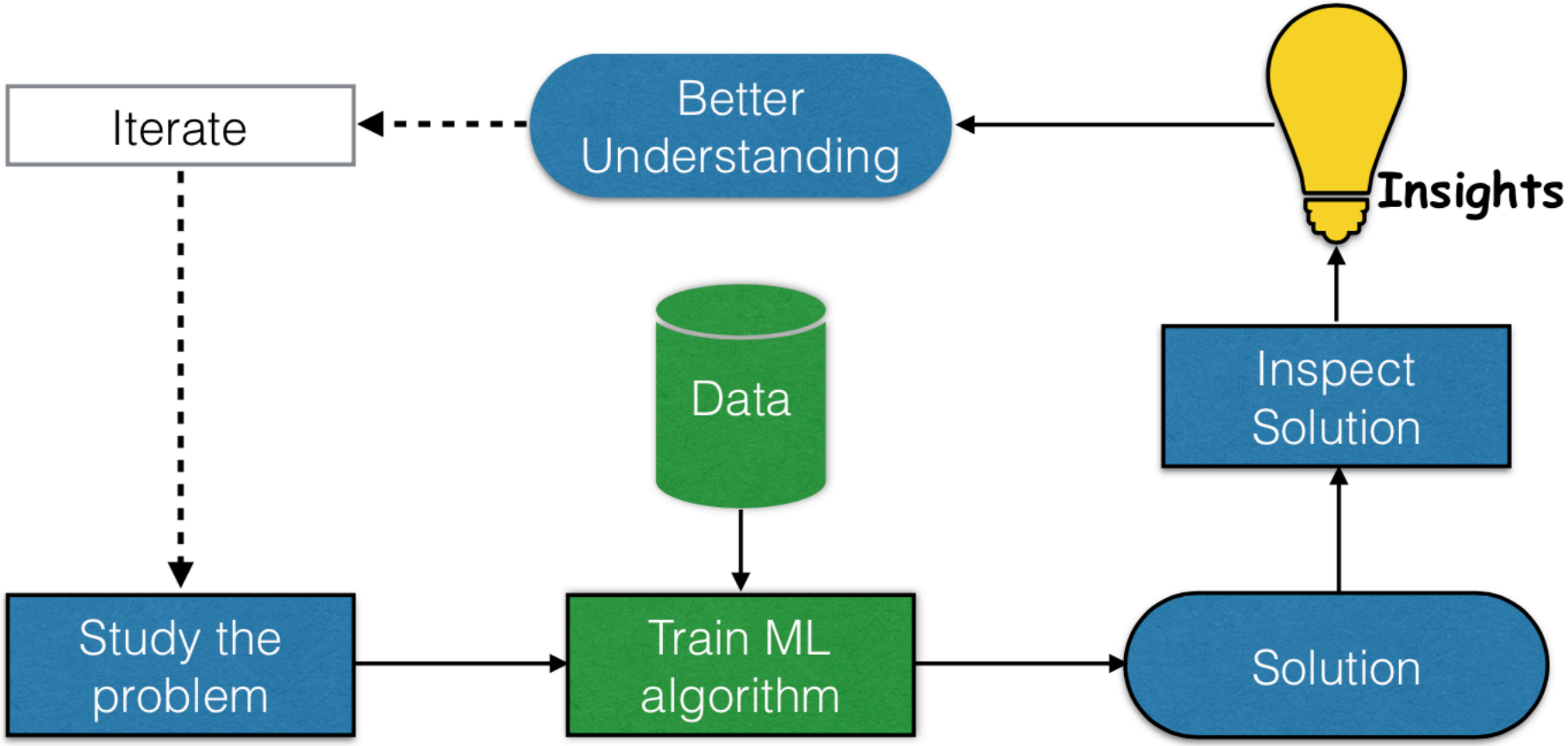
Learning from examples

Machine Learning approach (Software 2.0)



Adapting to change

Machine Learning approach (Software 2.0)



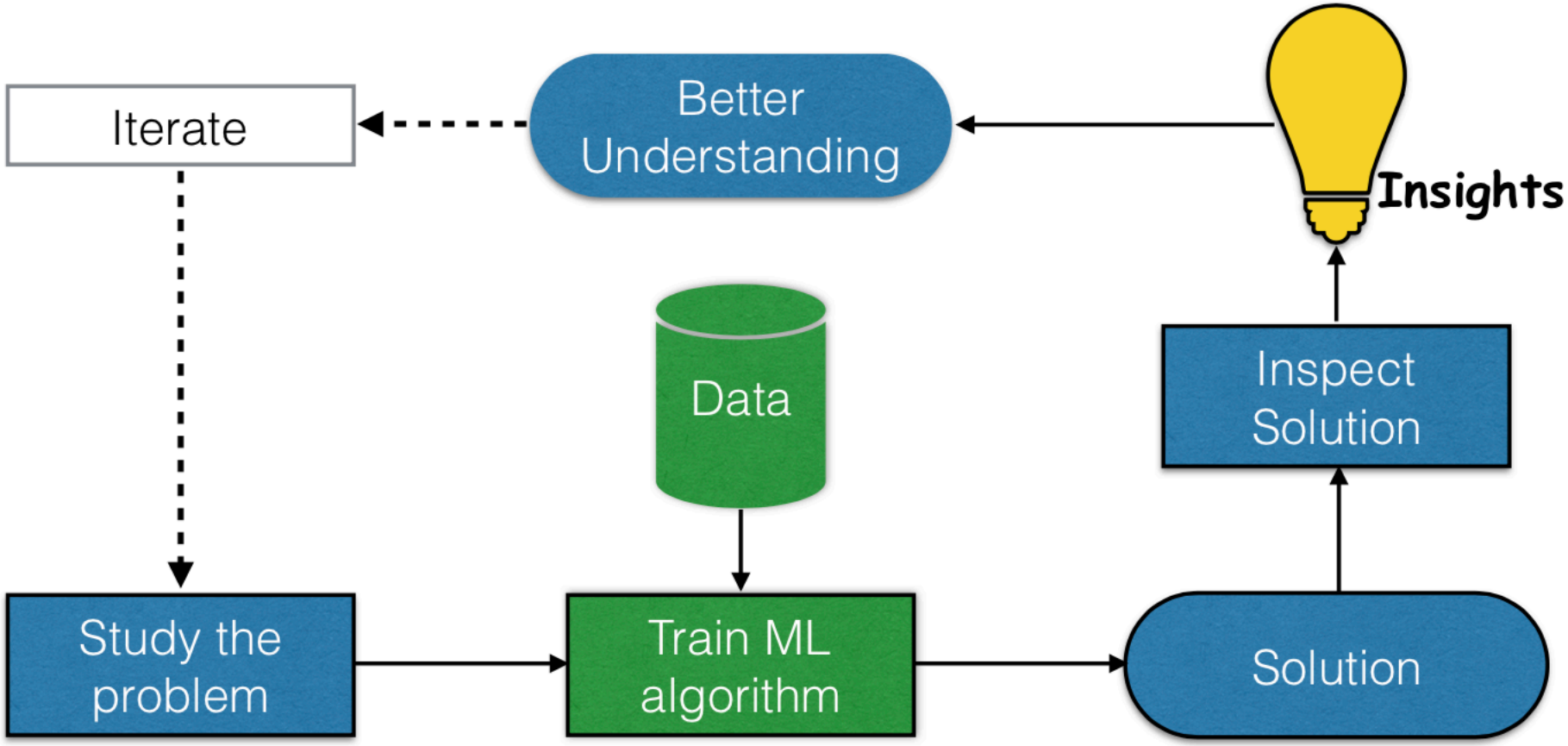
Help Humans learn

Now we are at a point where ML might teach humans about problems and solution algorithms.

Machine [illegible]vare 2.0)

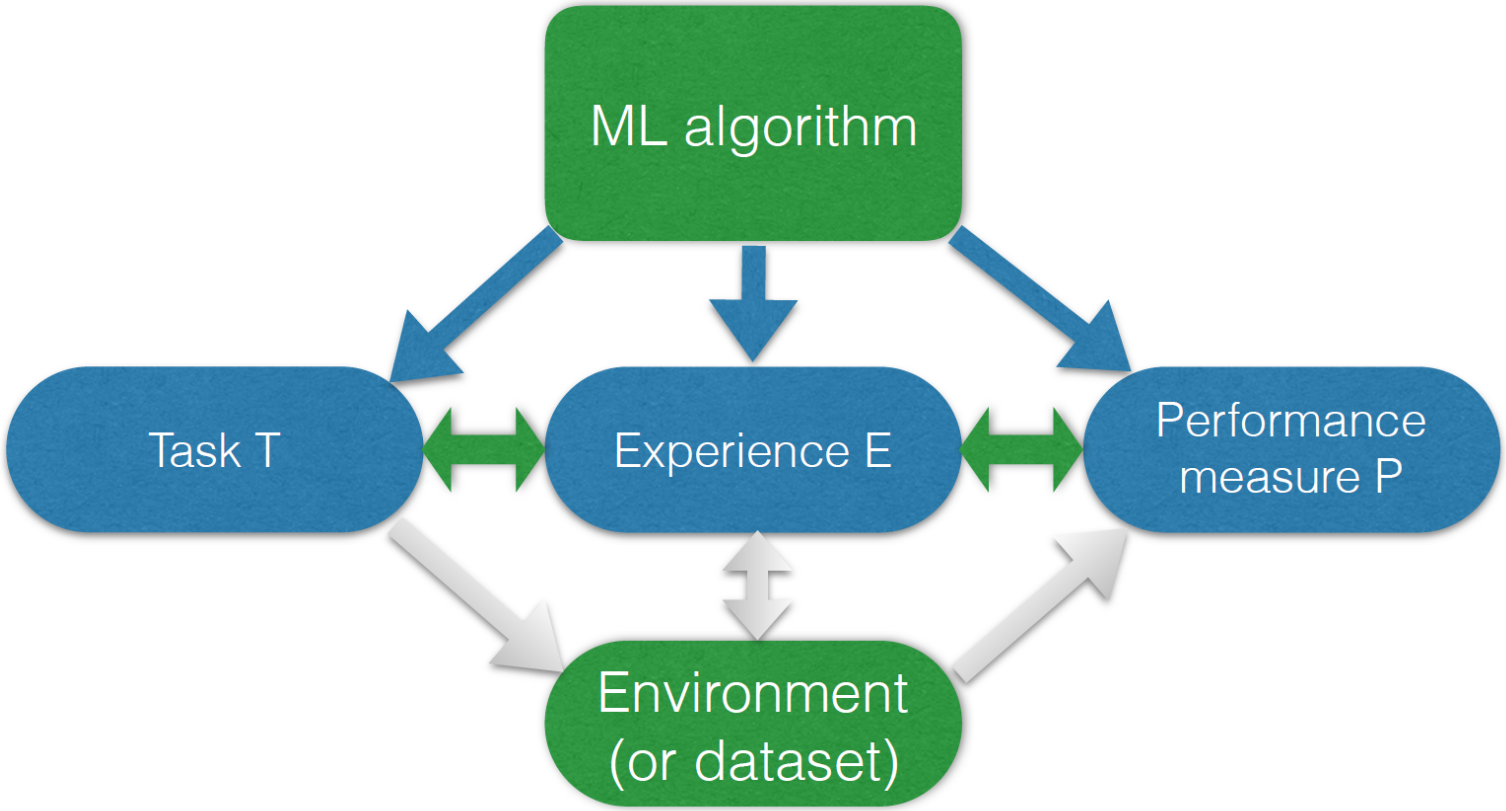
Help Humans learn

Machine Learning approach (Software 2.0)



Help Humans learn

Definition… schematically



Experience E

What data to use to solve the task

Learning Pillars : How much information is given to the ML algorithm

Task T

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

Performance Measure P

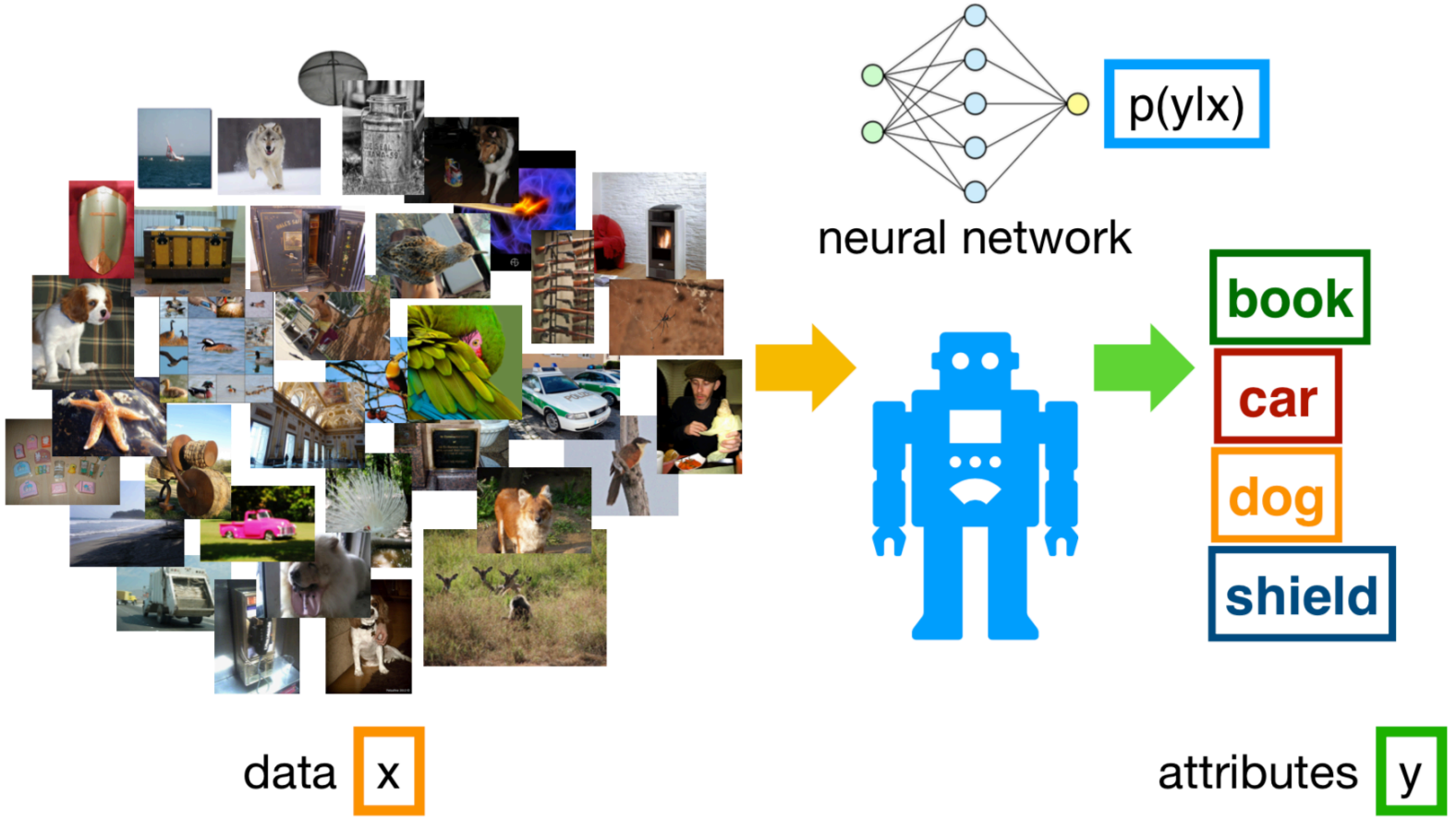
Estimate the ML algorithm performance on task T using the validation set

Learning Pillars

Supervised Learning

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

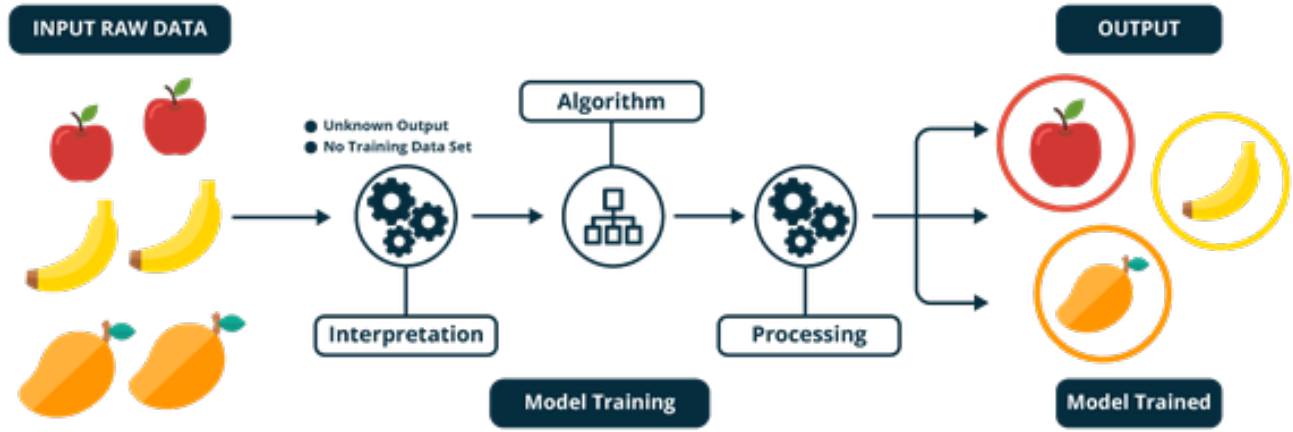
Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples.In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances.



Unsupervised Learning

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

Unsupervised learning (UL) is a type of algorithm that learns patterns from untagged data. The hope is that through mimicry, the machine is forced to build a compact internal representation of its world. In contrast to Supervised Learning (SL) where data is tagged by a human, eg. as "car" or "fish" etc, UL exhibits self-organization that captures patterns as neuronal predelections or probability densities.[1]



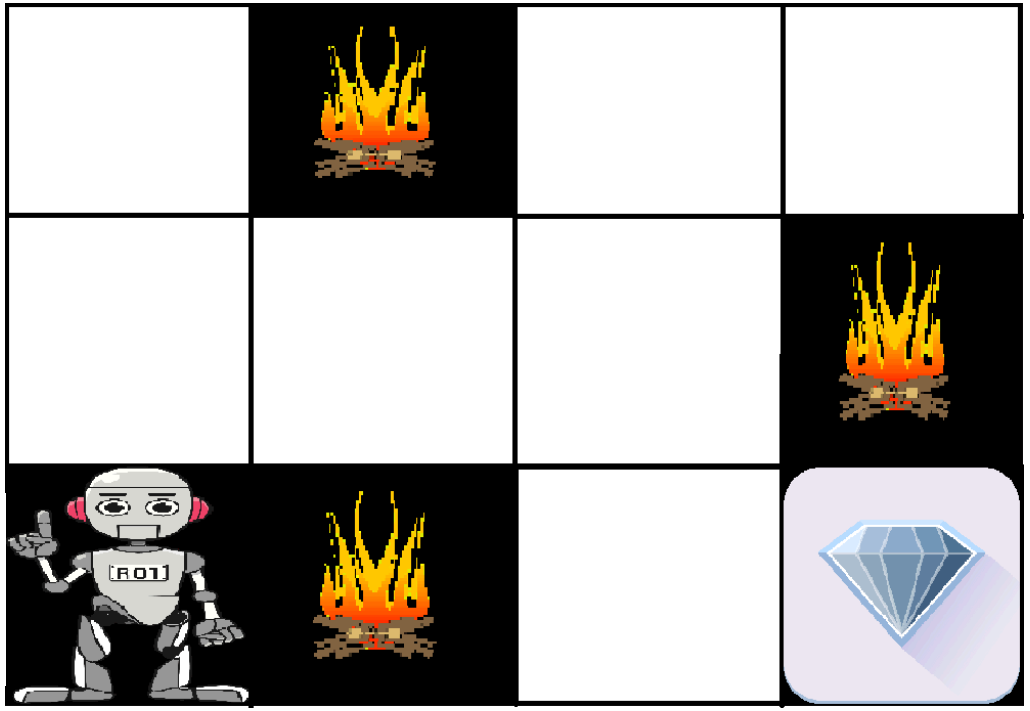
The data is clustered due to similarities but the machine doesn't know any labels for it.

Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning $concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. The machine is not necessarily given strict labelled data but much rather rewards for good behaviour and punishments for undesirable behaviour.

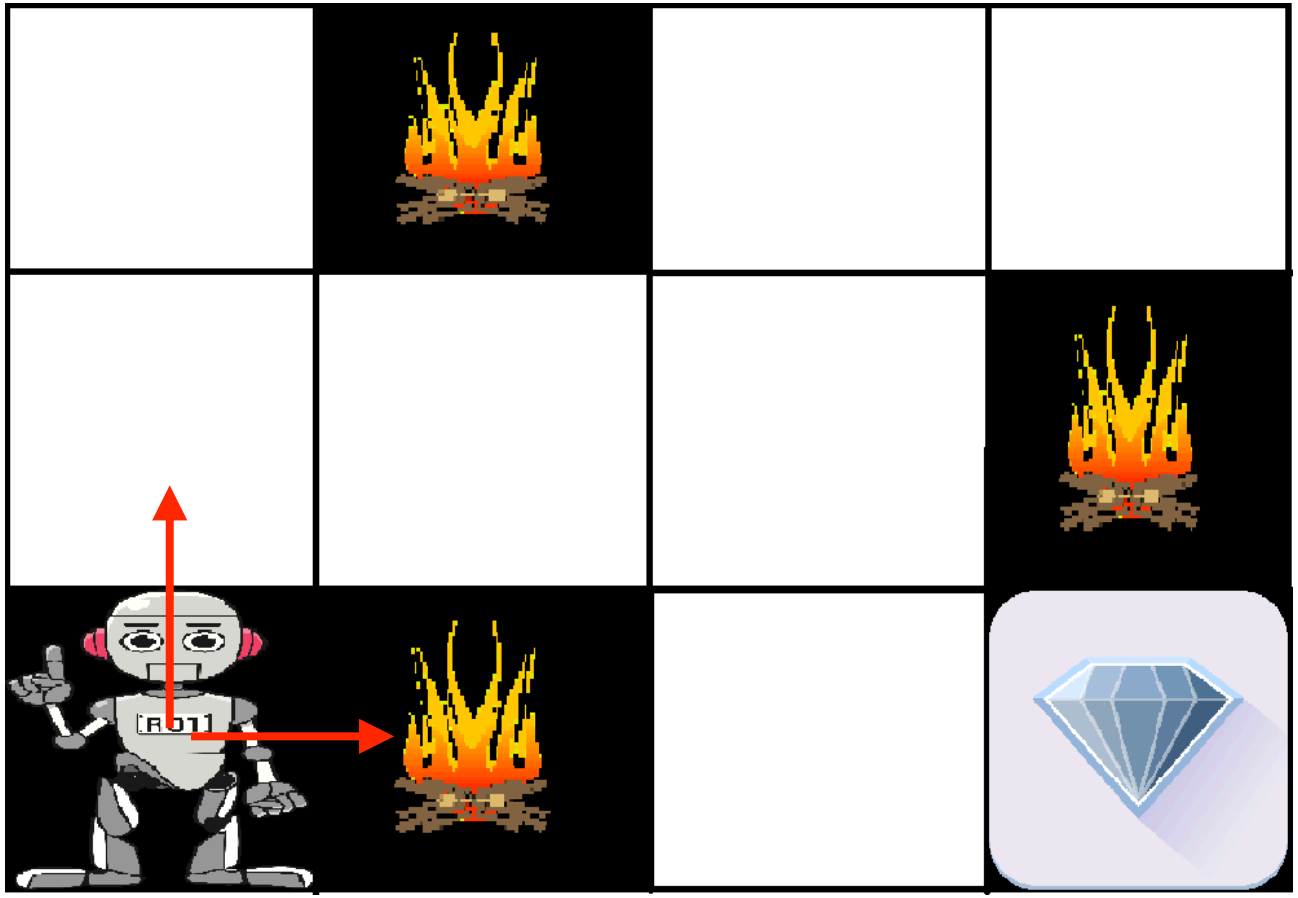
- Science of making optimal decisions.
  - Formulate reward-motivated behaviour exhibited by living species
  - Chess Game / Robotics for industrial automation

?



Reinforcement Learning

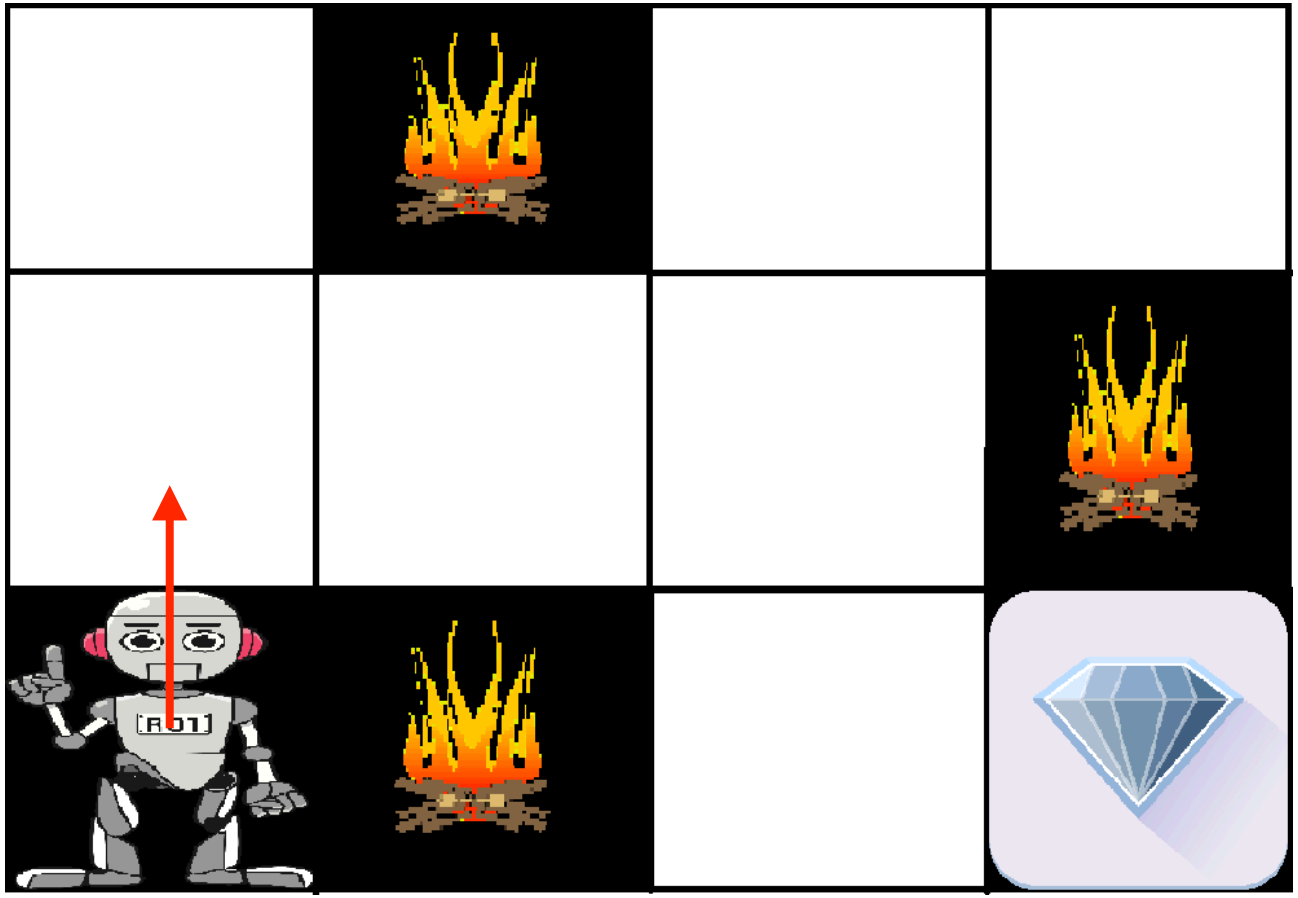
Robot need to find the diamond without stepping on the fire



Reinforcement Learning

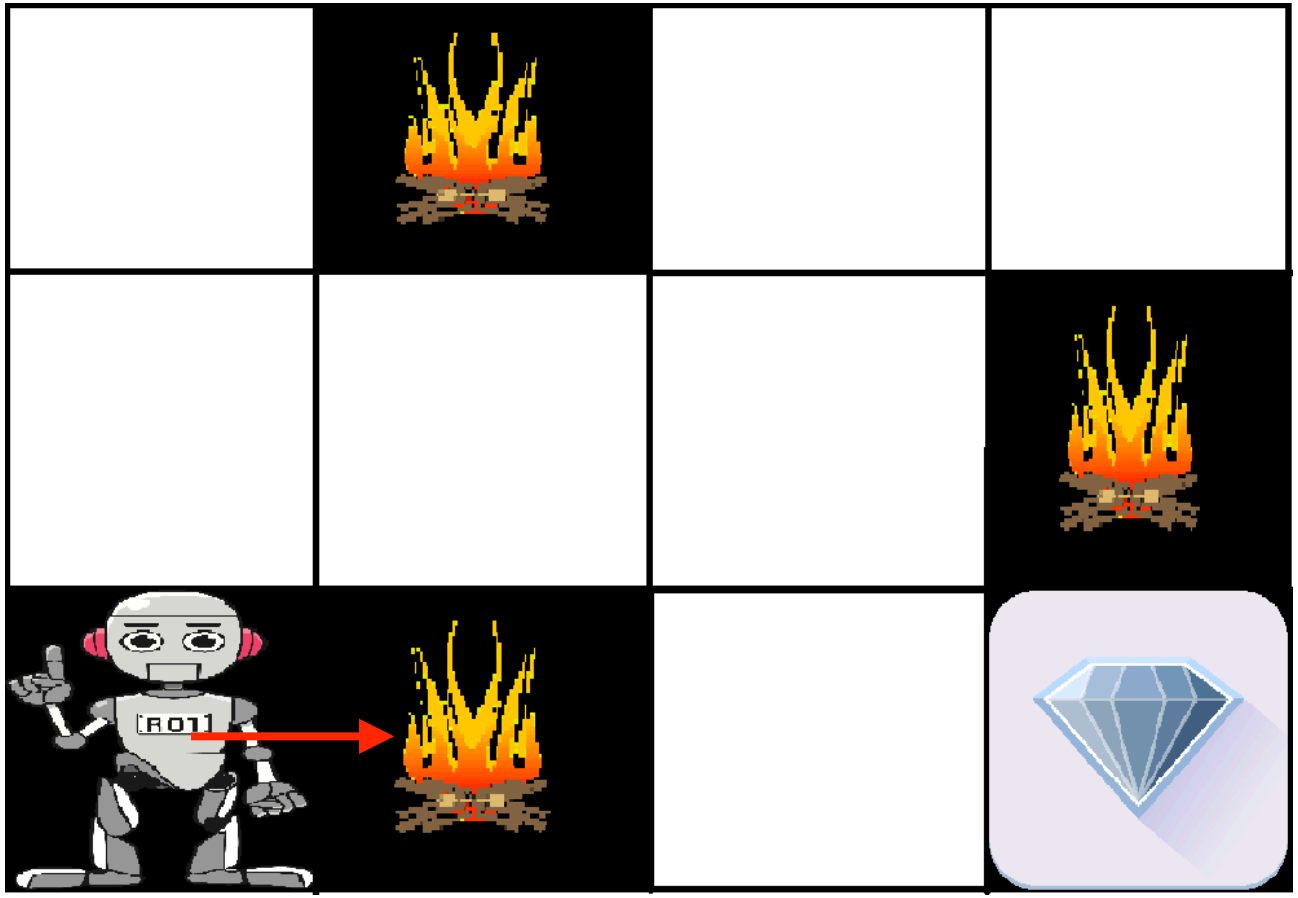
Robot need to find the diamond without stepping on the fire

√ Add reward



Reinforcement Learning

Robot need to find the diamond without stepping on the fire



X subtract reward

Training/Development/Test sets

- Separate the data into 2(3) sets
  - Training set for training
  - Dev. (Validation) set to find the best parameters
  - (Test set to estimate the performance)
- Separation depends on size of the dataset

Experience E

What data to use to solve the task

Learning Pillars : How much information is given to the ML algorithm

Task T

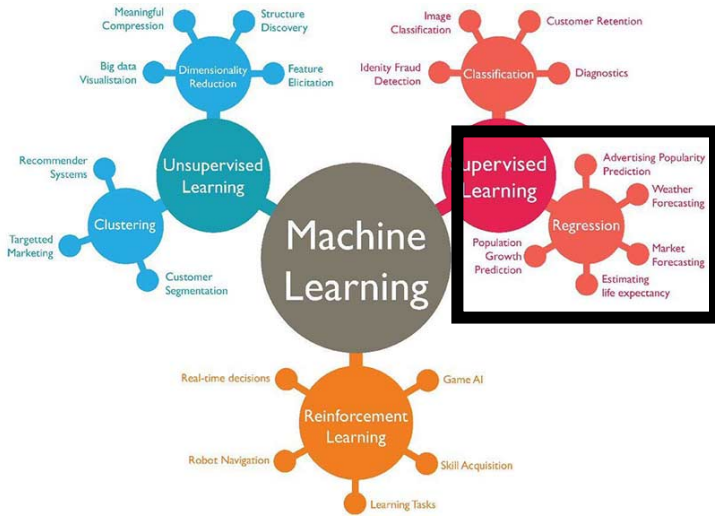
Find the function f that satisfies f(x) = y using the training set

Performance Measure P

Estimate the ML algorithm performance on task T using the validation set

Regression

Predict results within *a continuous output*



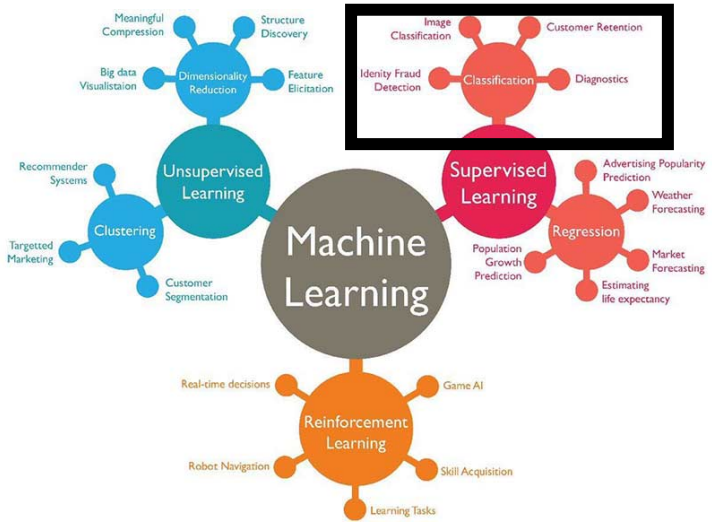
$82000

$55500

???

Classification

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

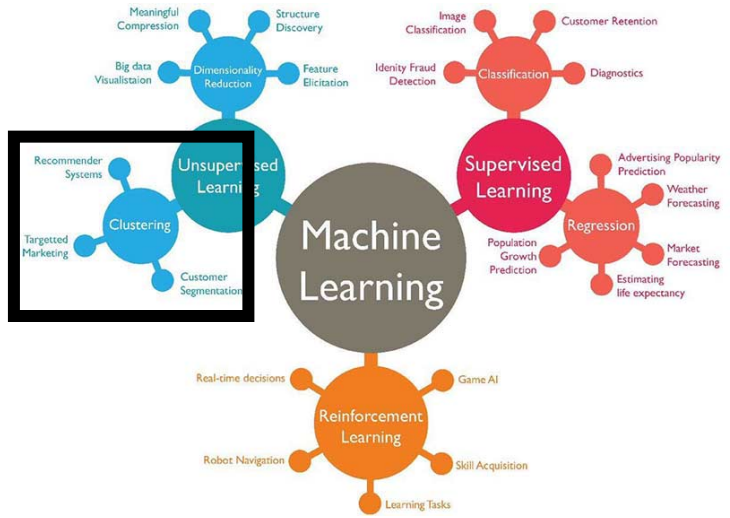


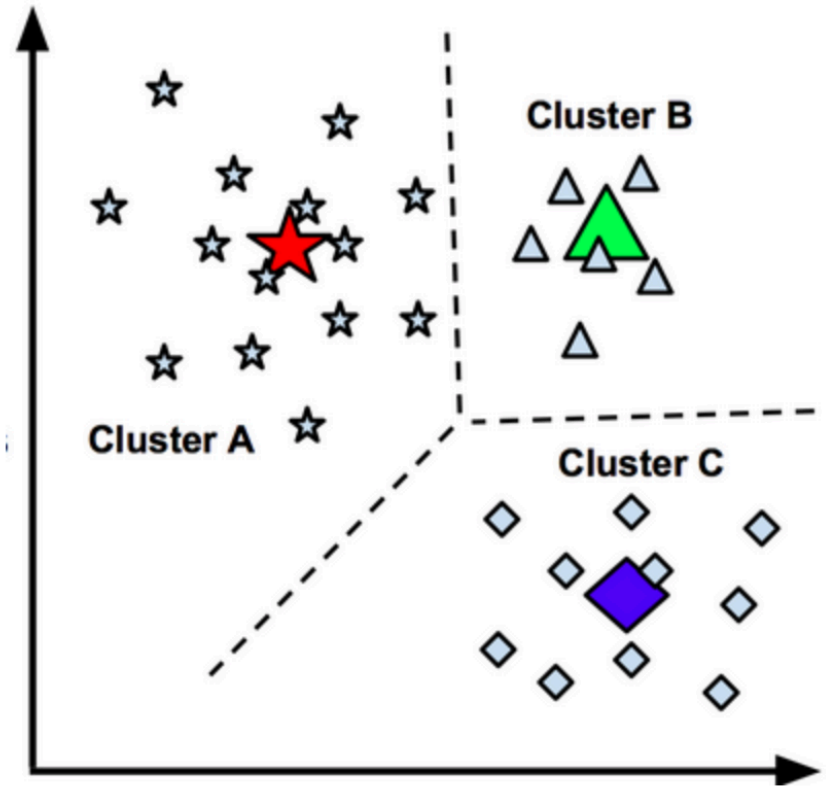
Dog: 96%
Cat: 29%
Duck: 2%
Bird: 0%

Dog: 36%
Cat: 94%
Duck: 2%
Bird: 1%

Clustering

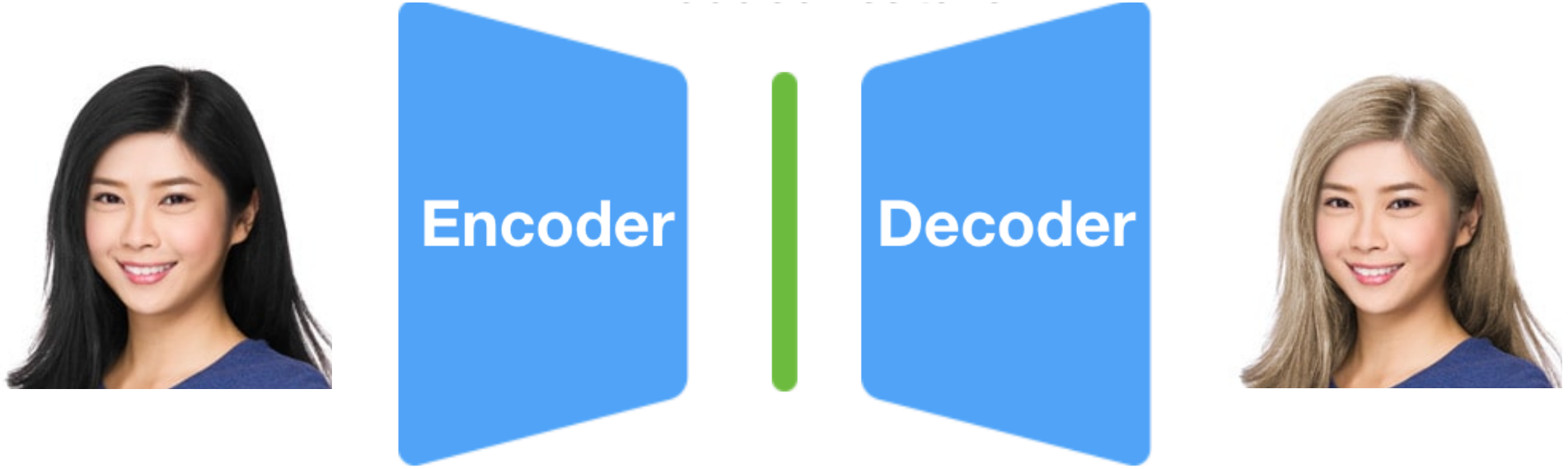
- create a set of categories, for which individual data instances have a set of common or similar characteristics.





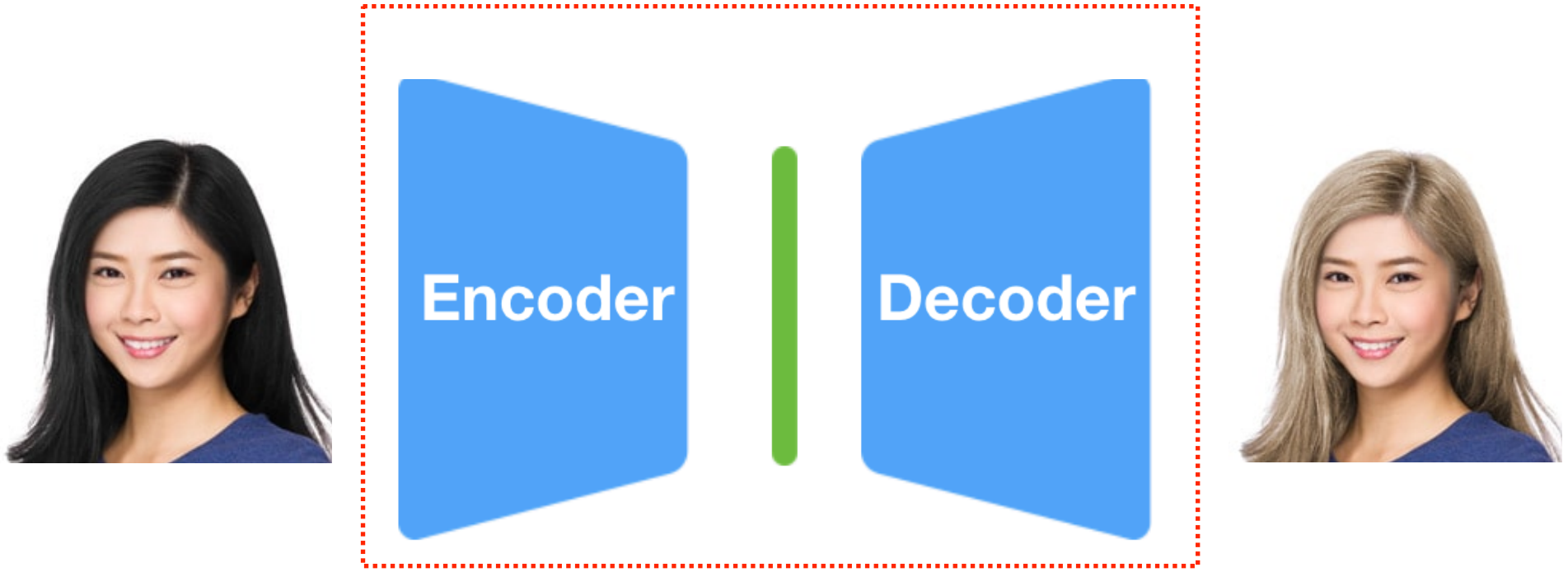
Data Generation

- generate appropriately novel data



Data Generation

- generate appropriately novel data

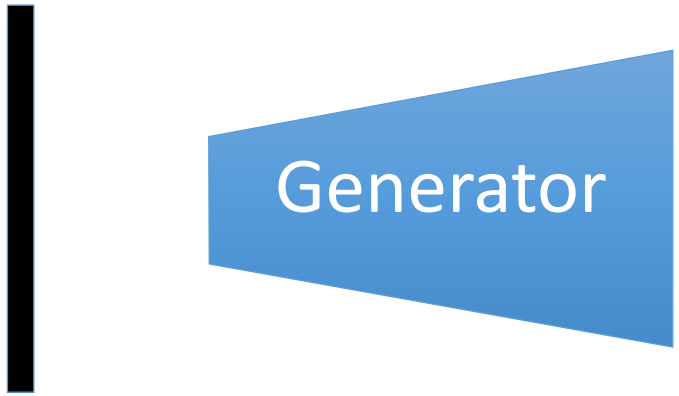


Autoencoder

Data Generation

- generate appropriately novel data

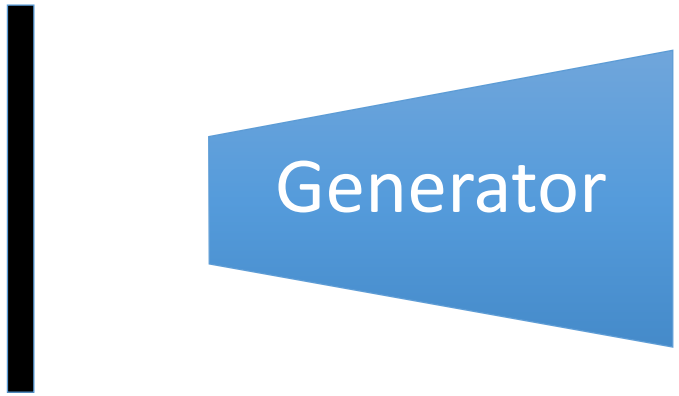
Noise vector



Data Generation

- generate appropriately novel data

Noise vector



Generative Adversarial Networks (GANs)

Experience E

What data to use to solve the task

Learning Pillars : How much information is given to the ML algorithm

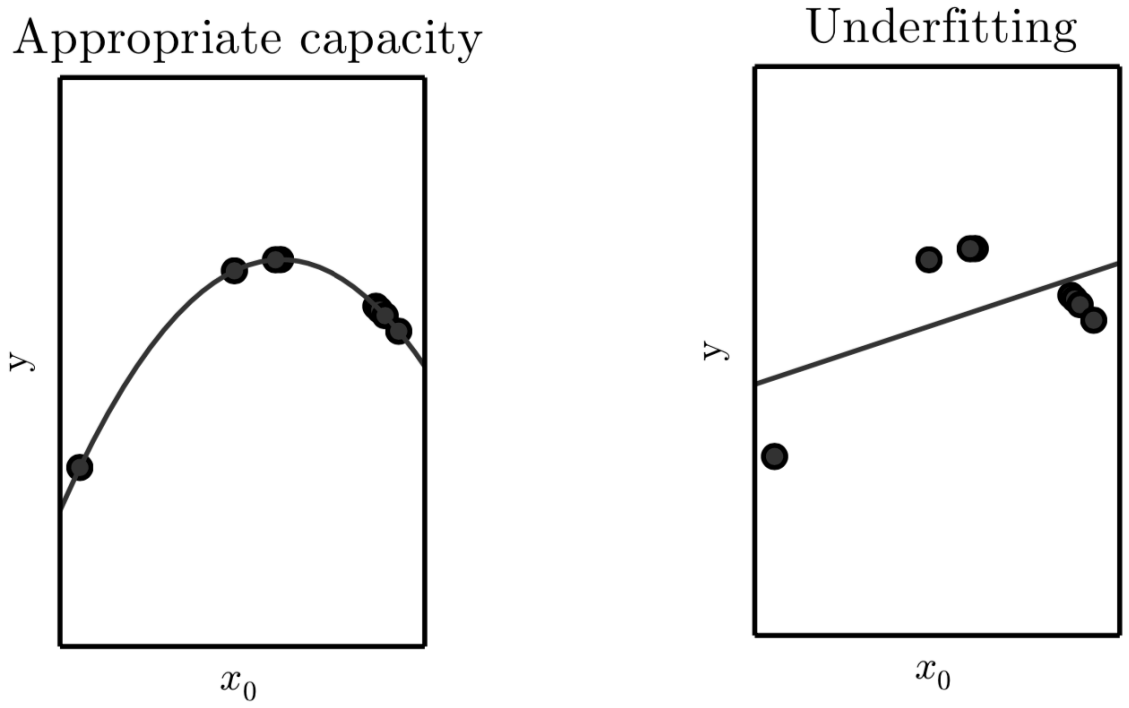
Task T

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

Performance Measure P

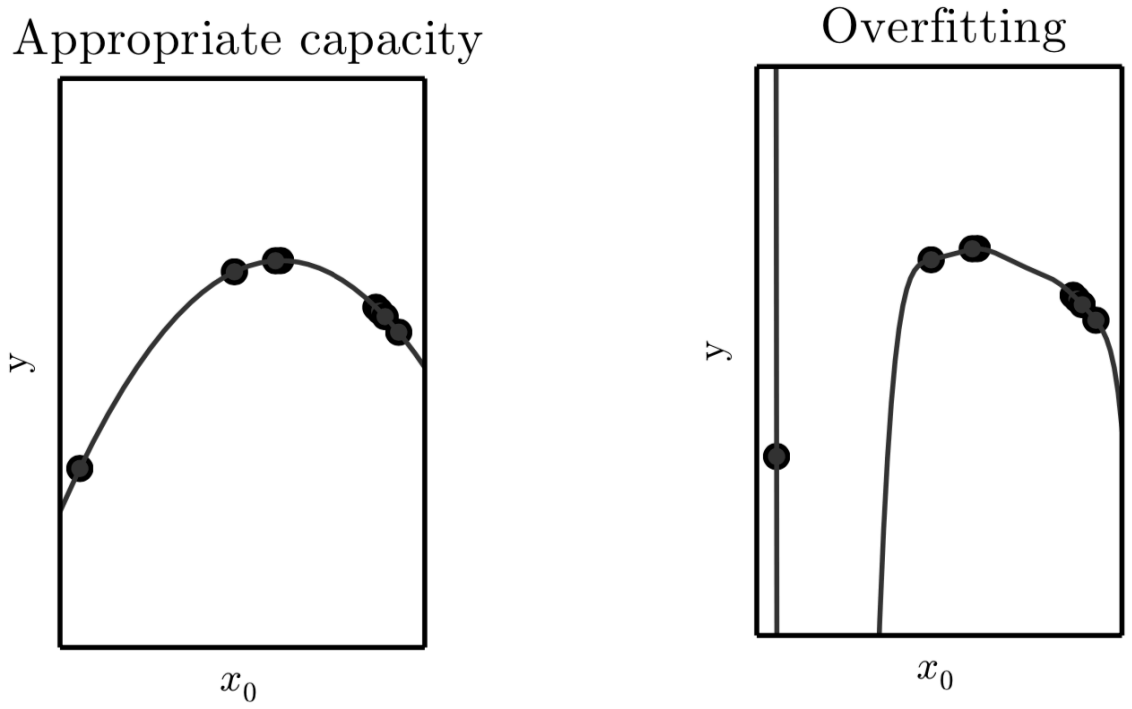
Estimate the ML algorithm performance on task T using the validation set

Underfitting



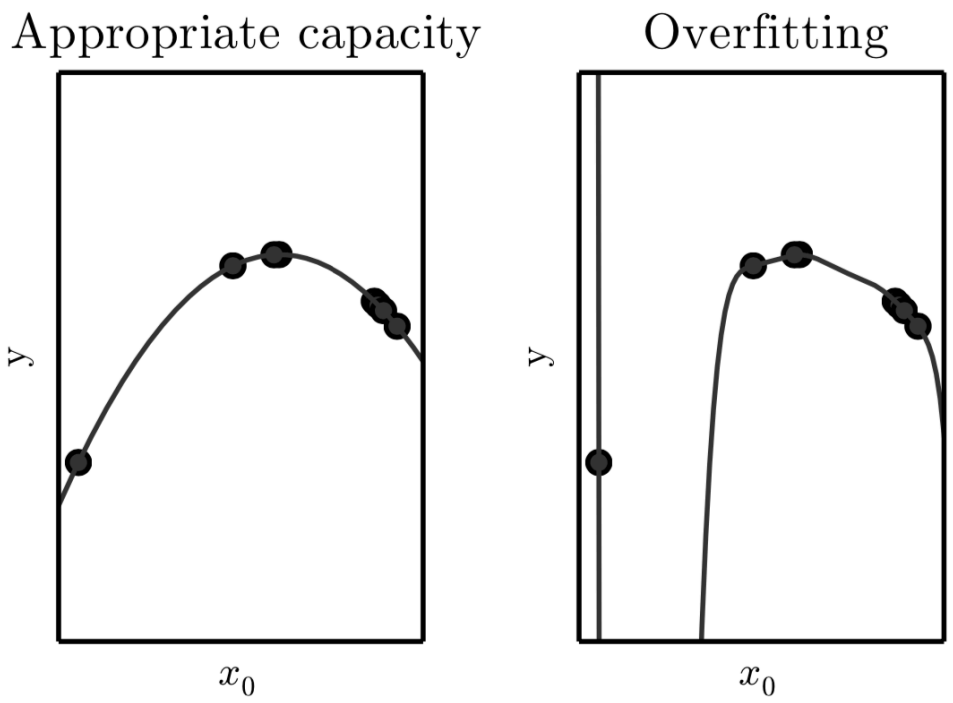
High bias

Overfitting

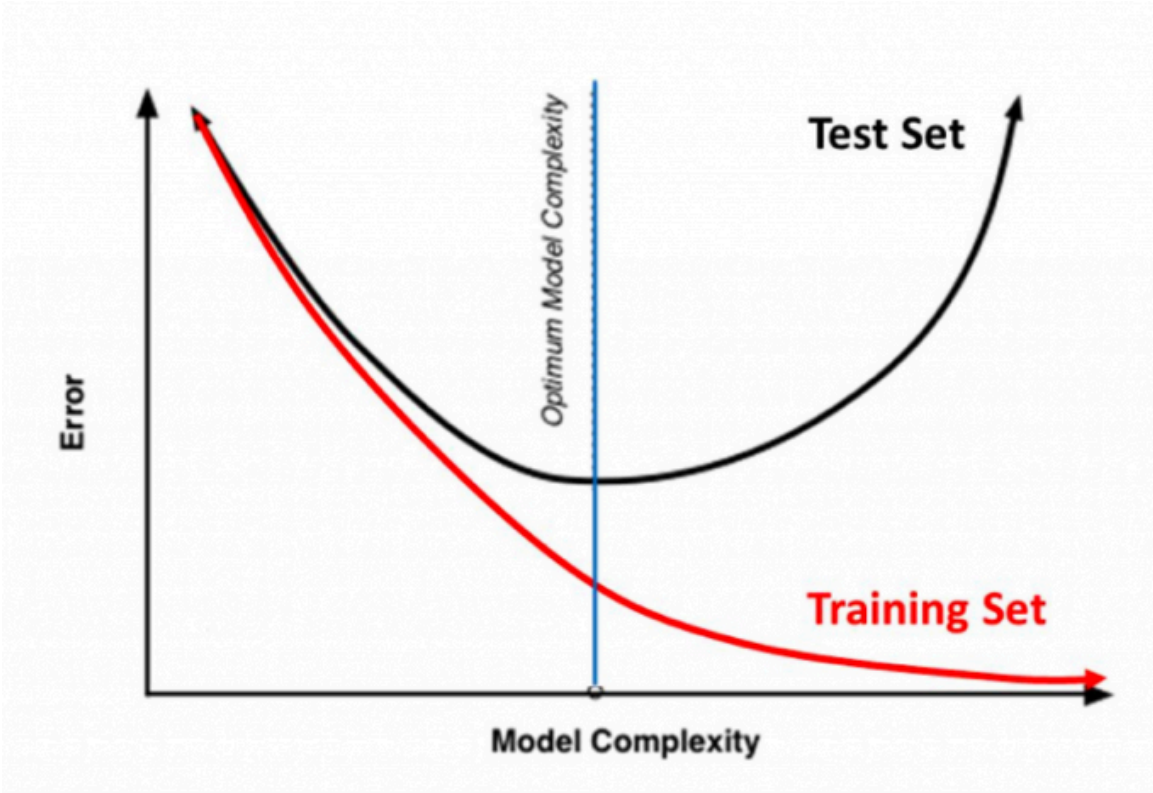


High variance

Overfitting



High variance



minimize cumulated error

Overfitting

Deep Learning

« Build a machine that can learn from experience and understand the world as a hierarchy of concepts »

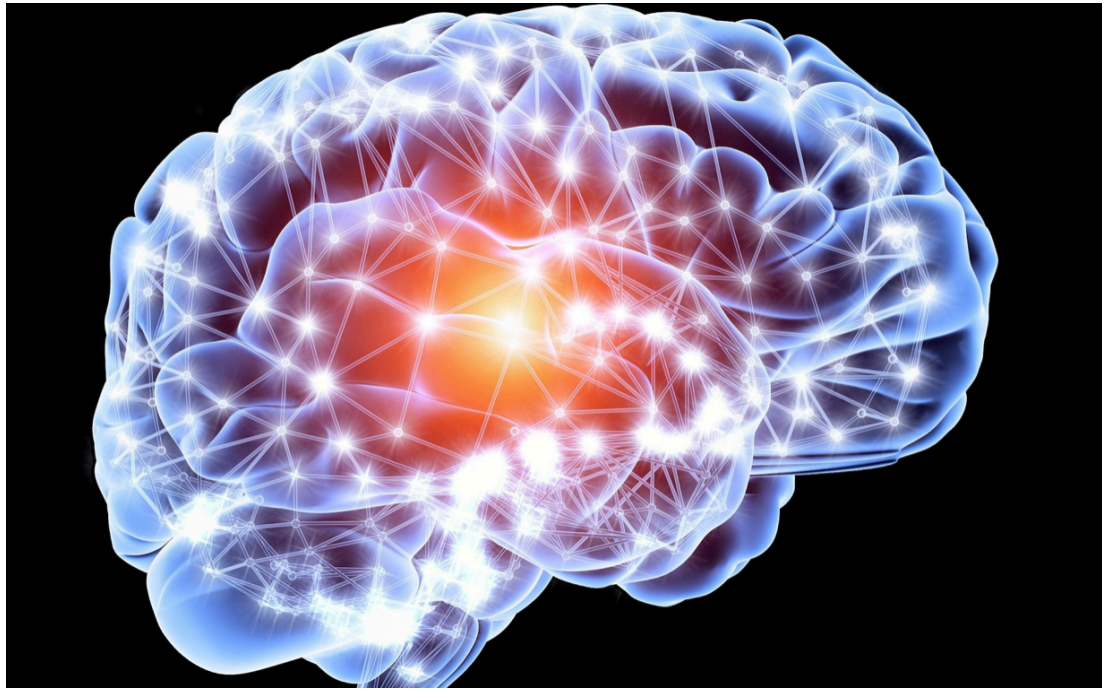
Neural Network (NN)

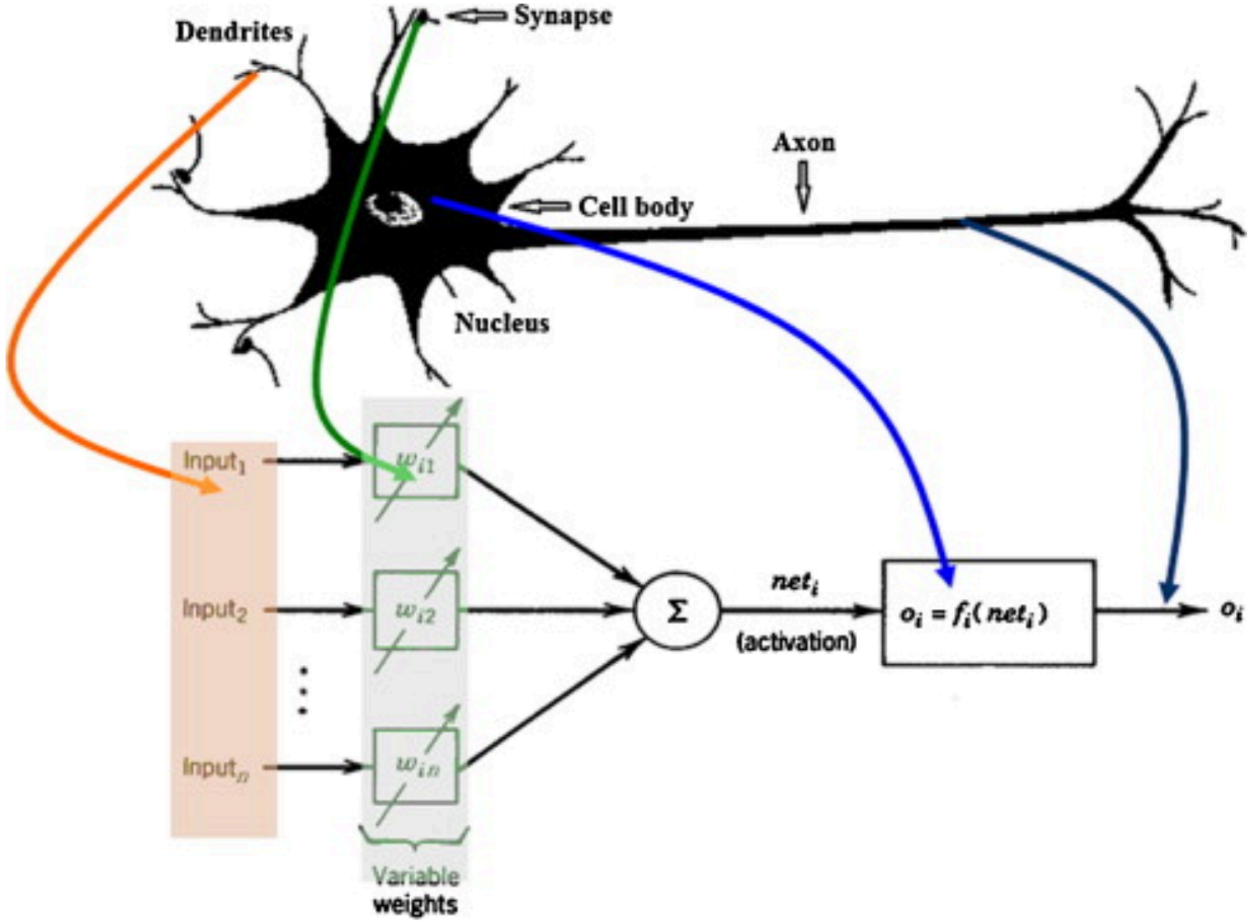
A NN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it (some function) and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Neural networks learn (or are trained) by processing examples, each of which contains a known "input" and "result," forming probability-weighted associations between the two, which are stored within the data structure of the net itself. The training of a neural network from a given example is usually conducted by determining the difference between the processed output of the network (often a prediction) and a target output. This is the error. The network then adjusts its weighted associations according to a learning rule and using this error value. Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output. After a sufficient number of these adjustments the training can be terminated based upon certain criteria.

Neural Network (NN)

- Learning algorithm inspired by *how the brain works*





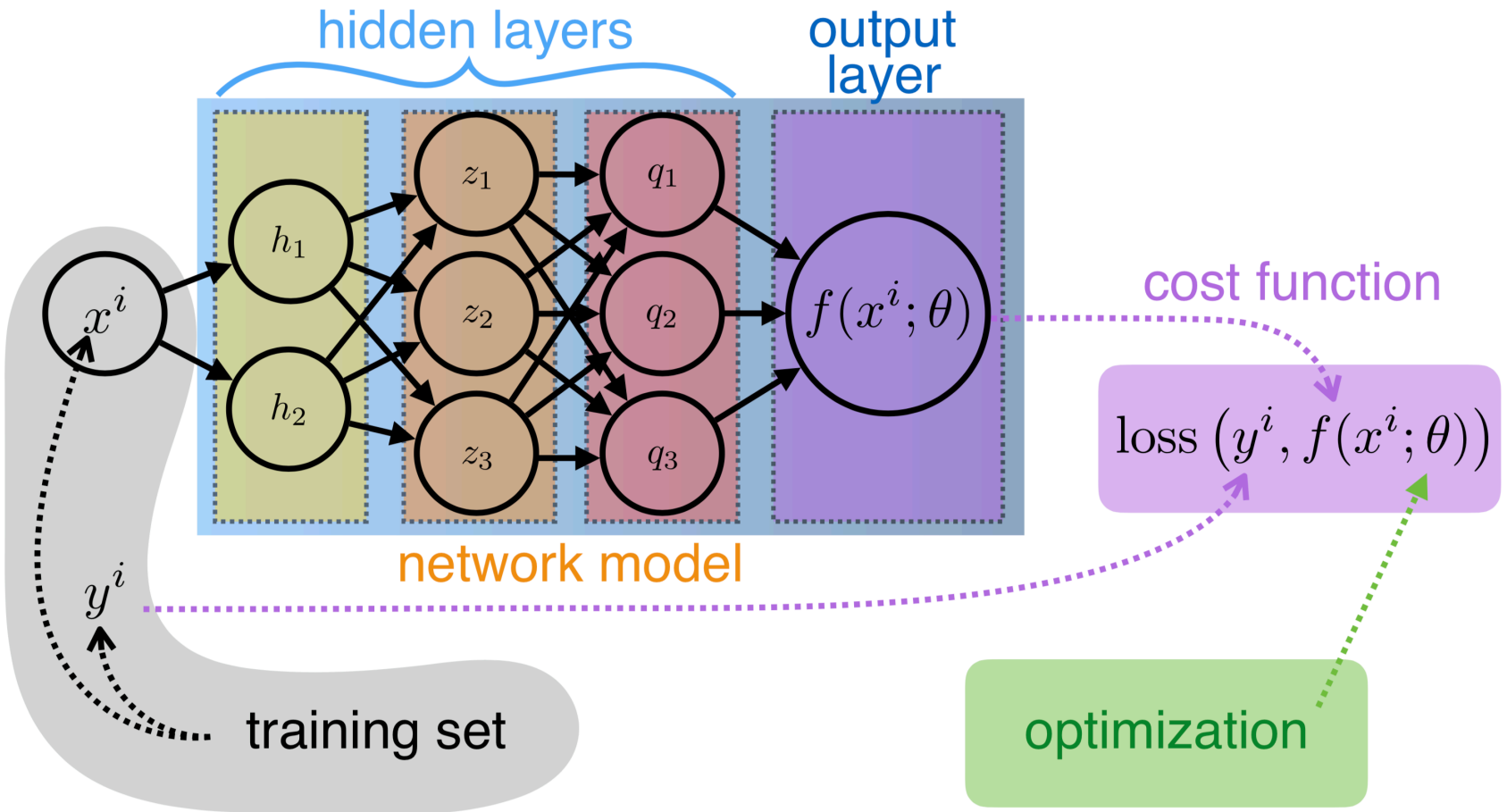
Deploying a Neural Network

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

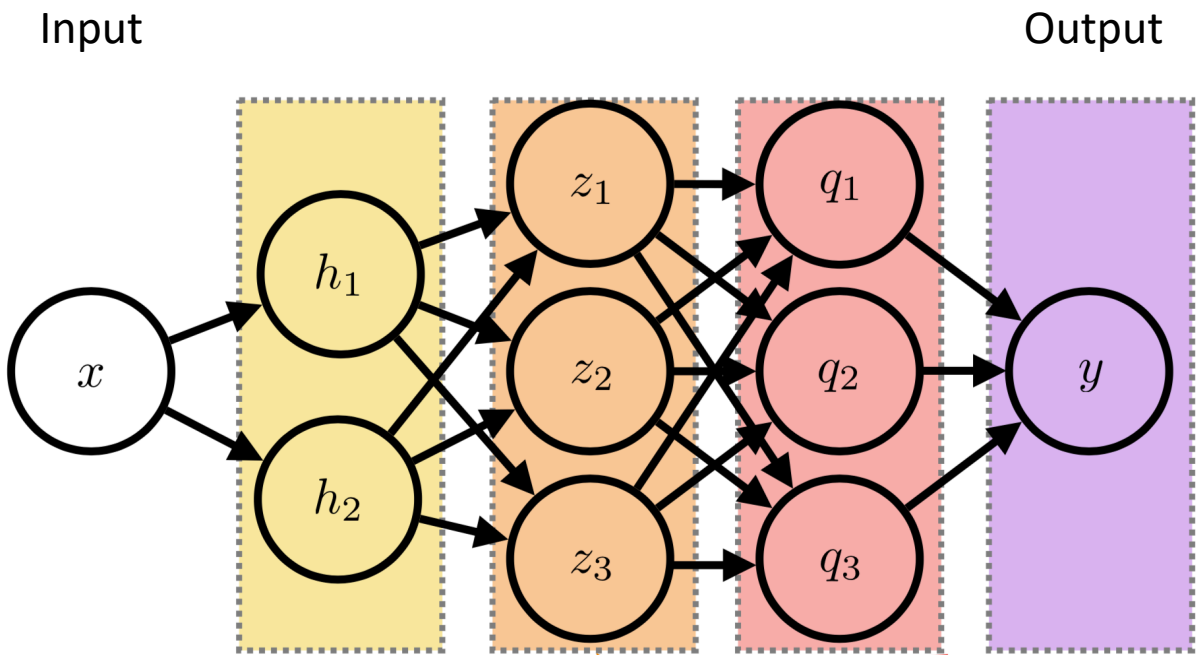
1) Network model

2) Cost function

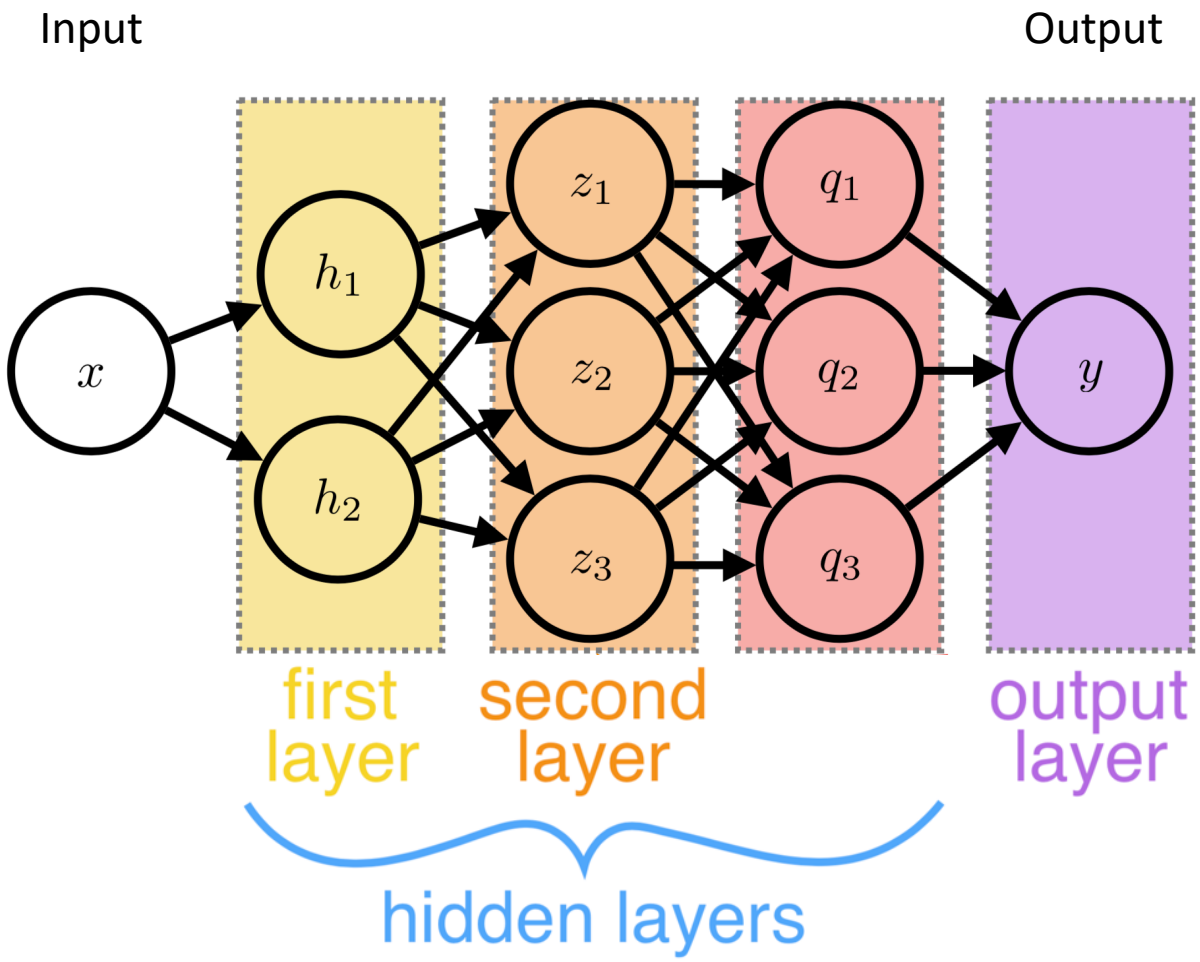
3) Optimization



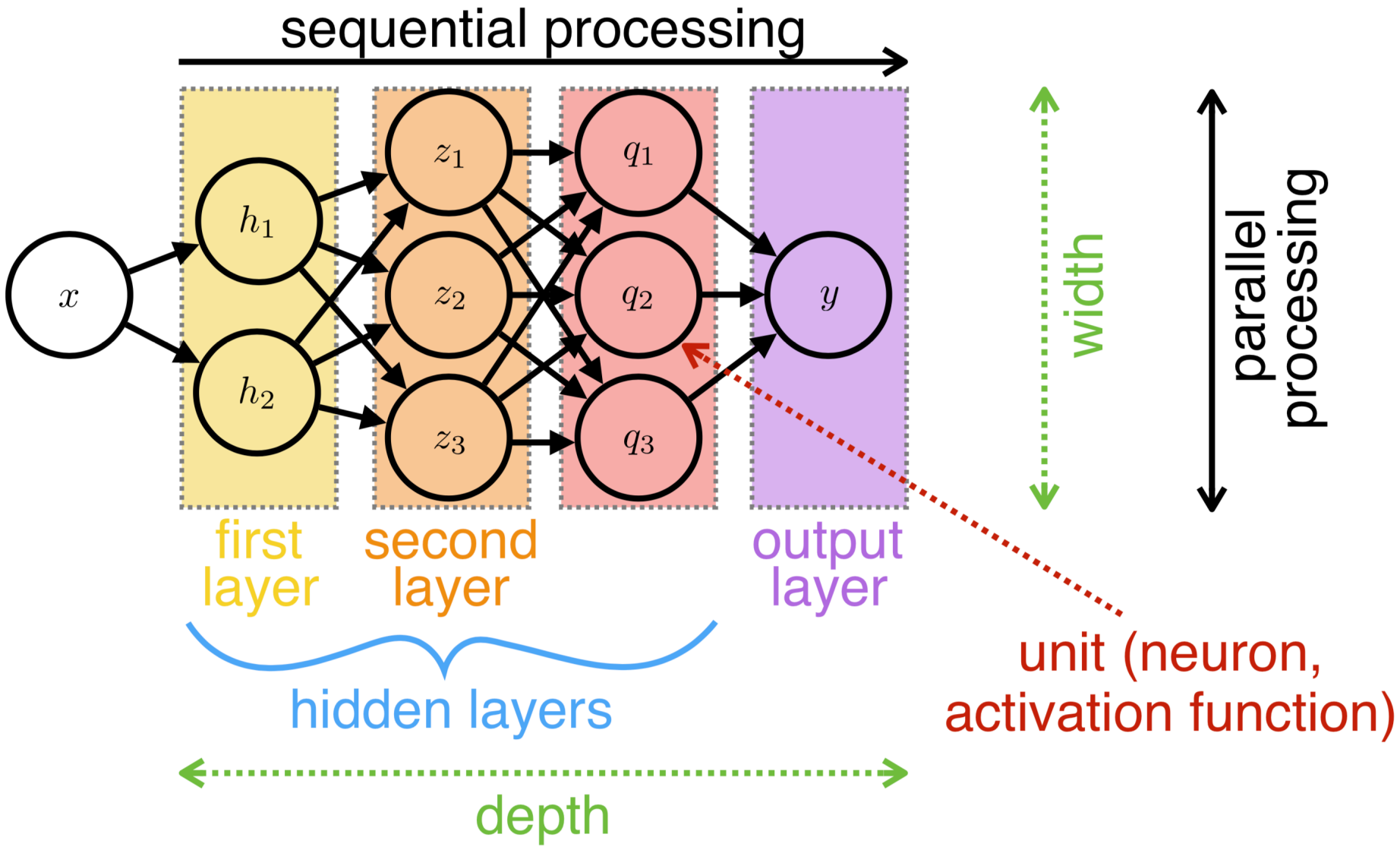
Network model



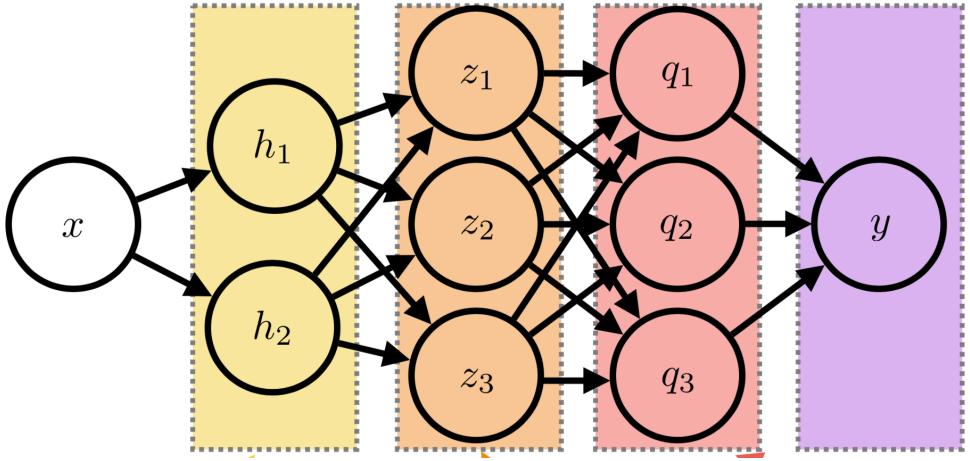
Network model



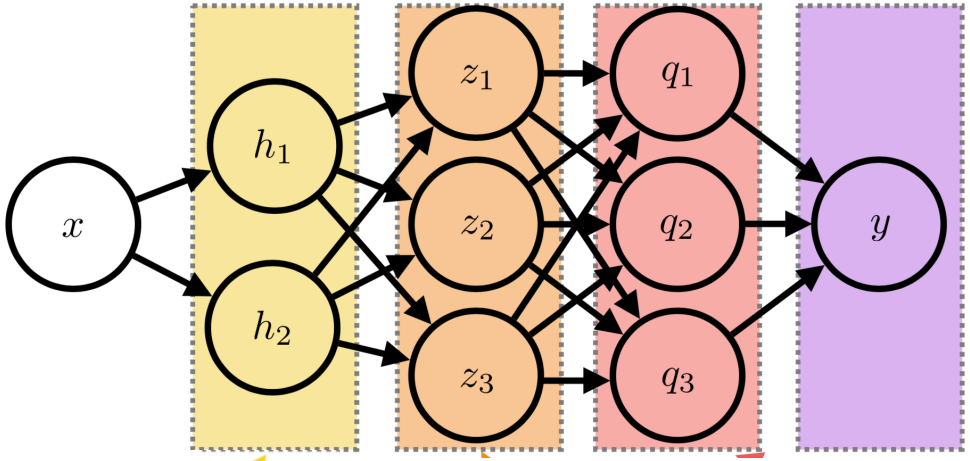
Network model



Activation functions

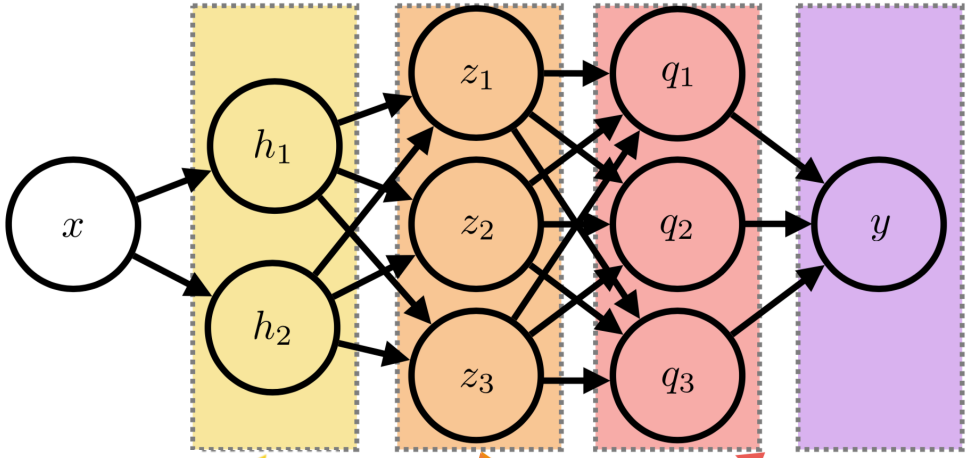


Activation functions



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

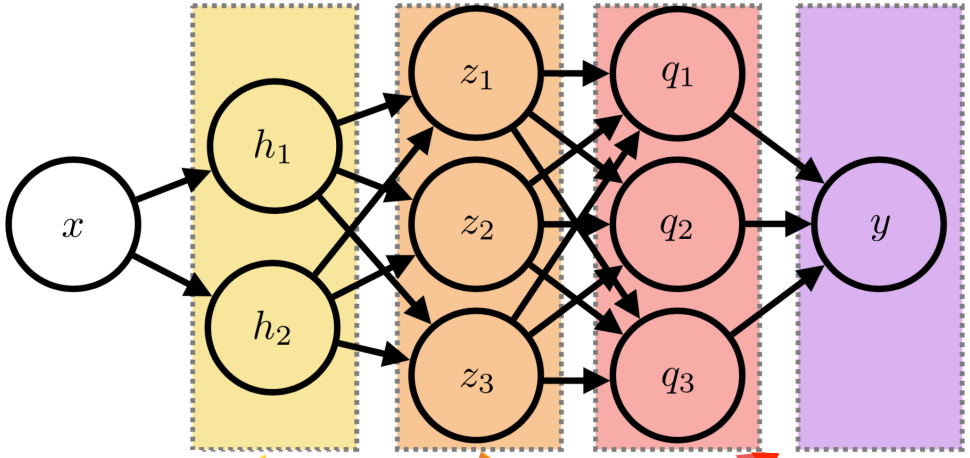
Activation functions



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

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

Activation functions

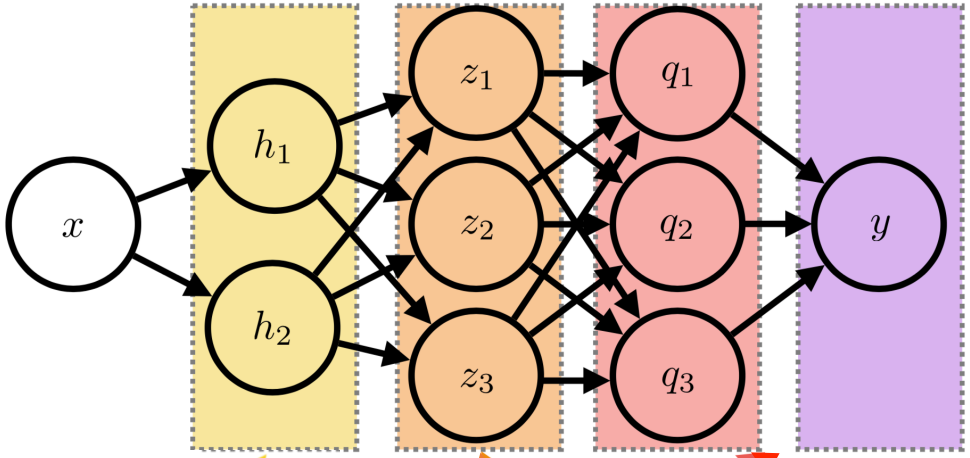


$$h_1 = f_{1,1}(x)$$
$$h_2 = f_{1,2}(x)$$

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

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

Activation functions



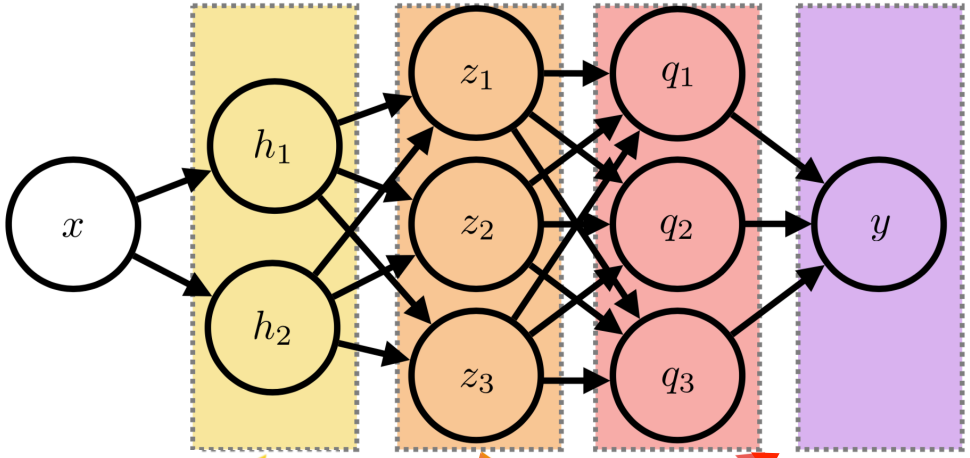
$$y = f_4(q_1, q_2, q_3)$$

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

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

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

Activation functions



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

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

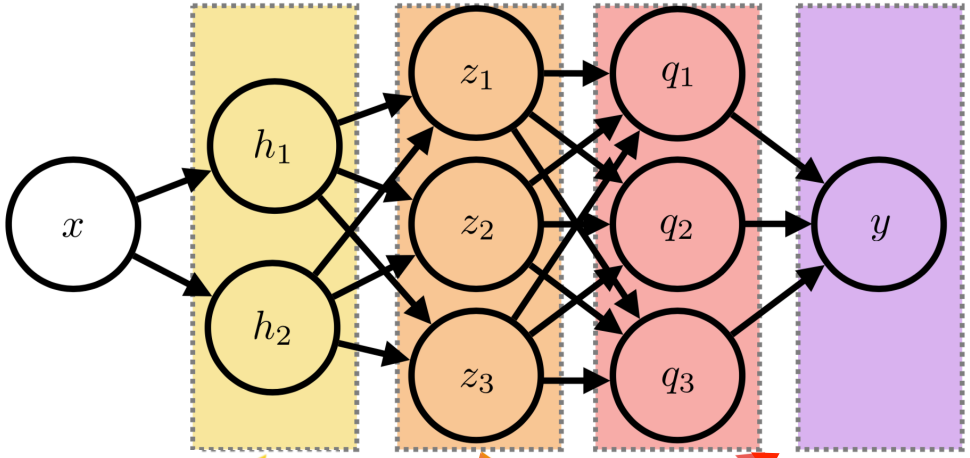
Hierarchical representation

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

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

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

Activation functions



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

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

Hierarchical representation

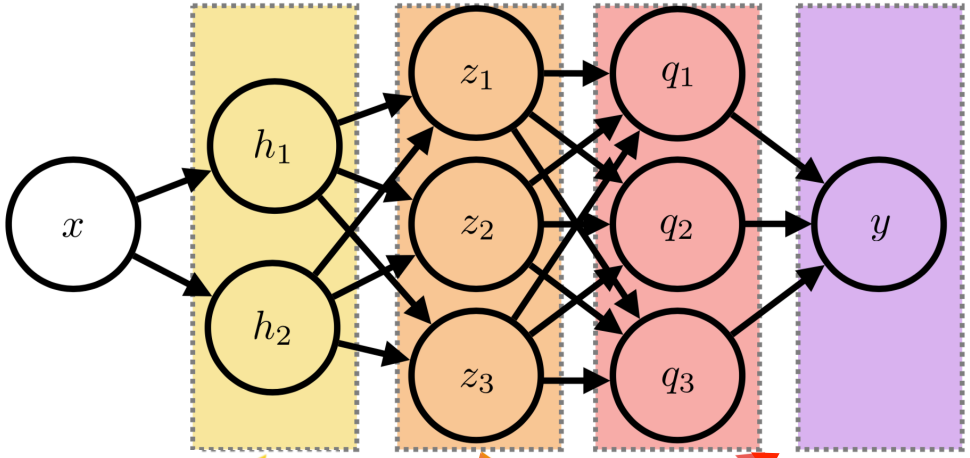
$$h_1 = f_{1,1}(x)$$
$$h_2 = f_{1,2}(x)$$

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

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

$$f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2$$

Activation functions



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

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

Hierarchical representation

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

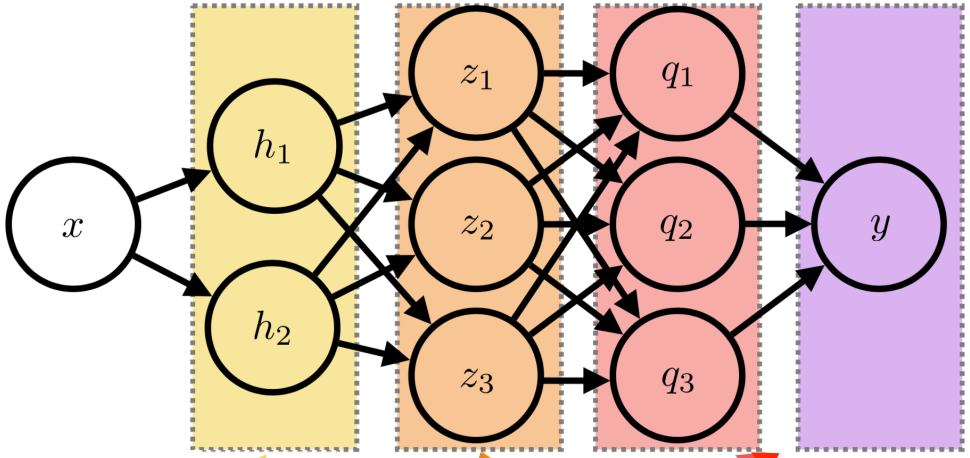
$$z_1 = f_{2,1}(h_1, h_2)$$
$$z_2 = f_{2,2}(h_1, h_2)$$
$$z_3 = f_{2,3}(h_1, h_2)$$

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

Bias (constant)

$$f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}$$

Activation functions



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

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

Hierarchical representation

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

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

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

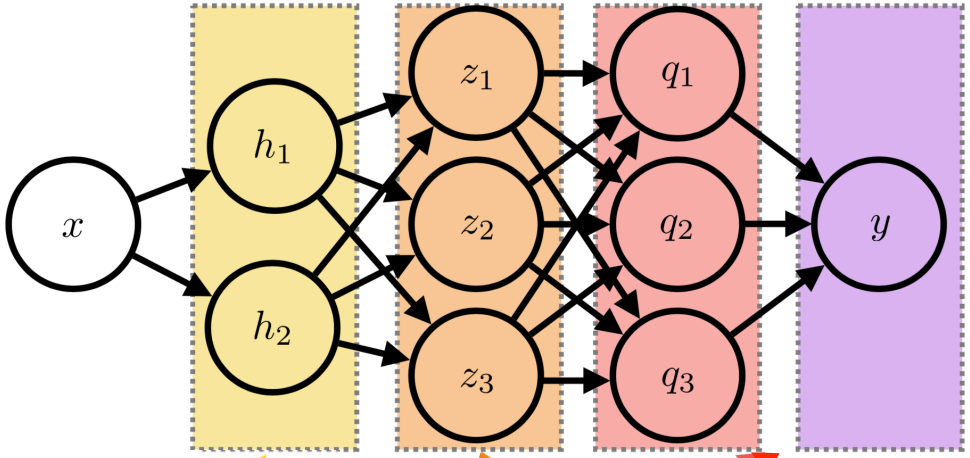
Bias (constant)

$$f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}$$

Weights w and bias b parameters to optimise

Activation functions

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



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

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

Hierarchical representation

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

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

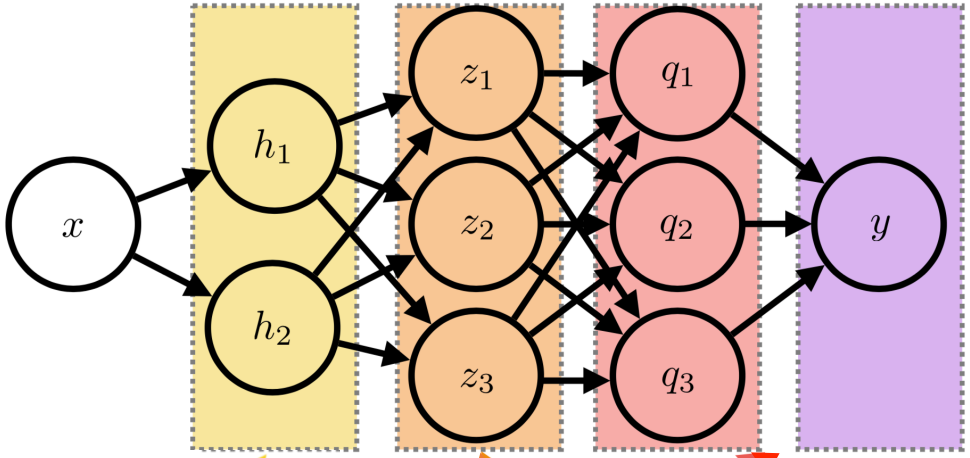
$$q_1 = f_{3,1}(z_1, z_2, z_3)$$
$$q_2 = f_{3,2}(z_1, z_2, z_3)$$
$$q_3 = f_{3,3}(z_1, z_2, z_3)$$

Bias (constant)

$$f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}$$

Weights w and bias b parameters to optimise

Activation functions



Fully connected

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

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

Hierarchical representation

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

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

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

Bias (constant)

$$f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}$$

Weights w and bias b parameters to optimise

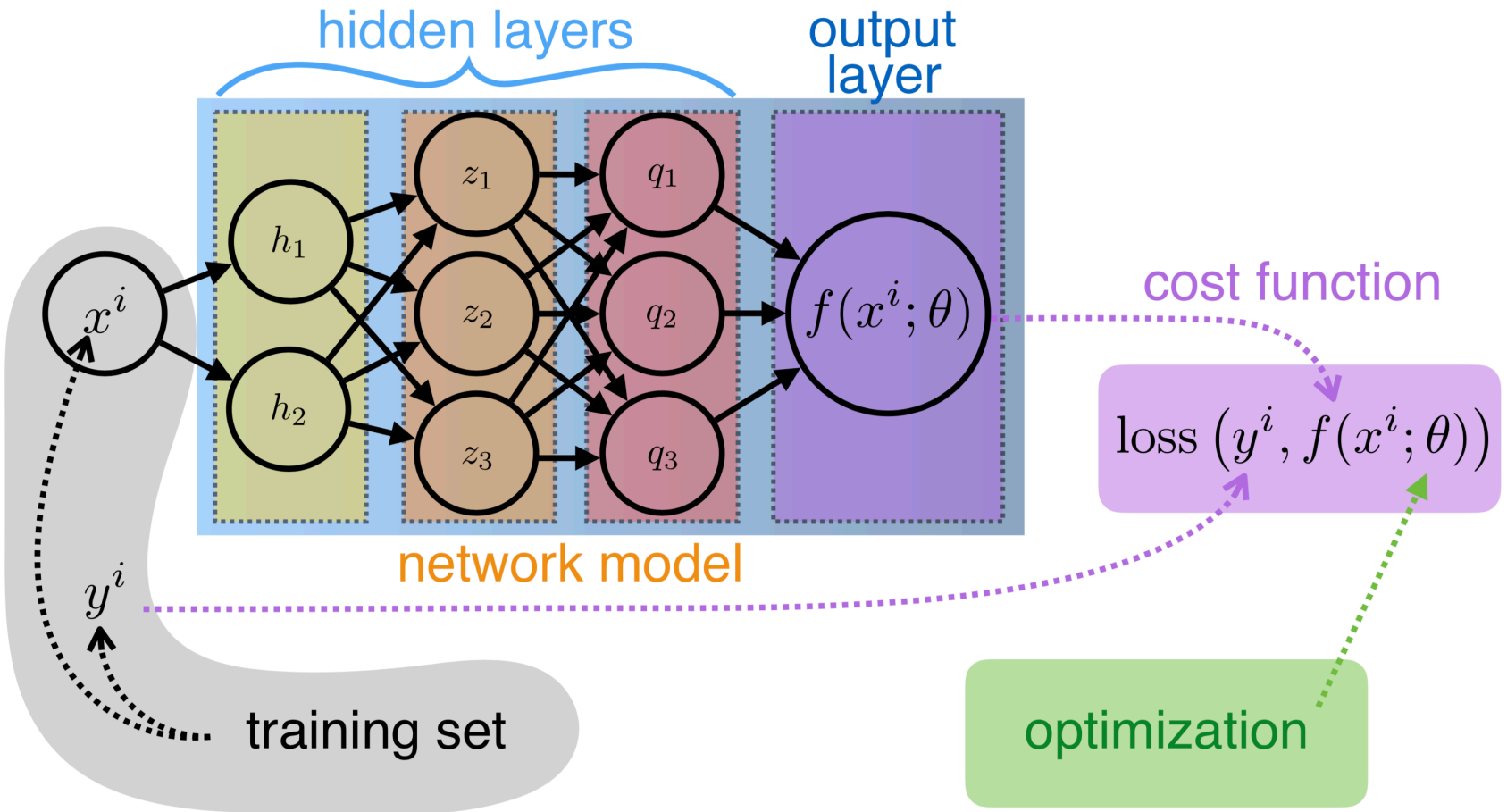
Deploying a Neural Network

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

1) Network model

2) Cost function

3) Optimization



2) Loss and Cost functions

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

$$L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

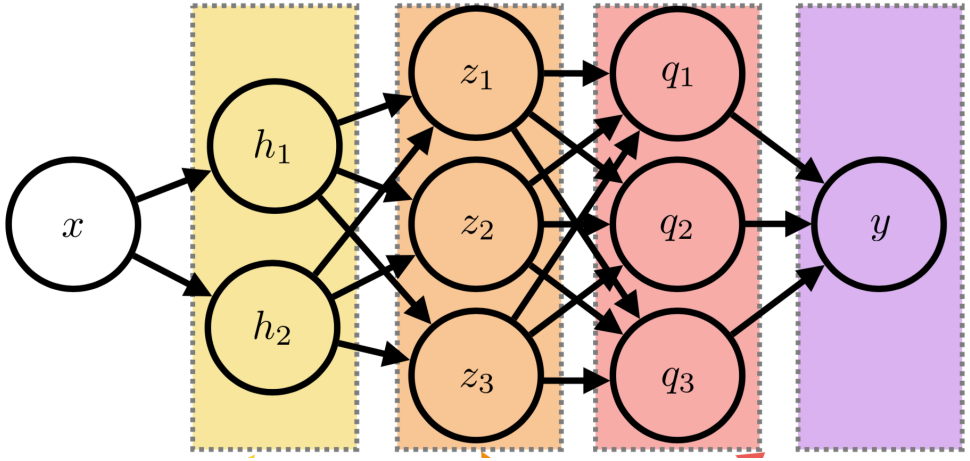
- Cost function $J(w, b)$ is the average of the loss function on the *entire training set*

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

Cost function: A function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number.

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

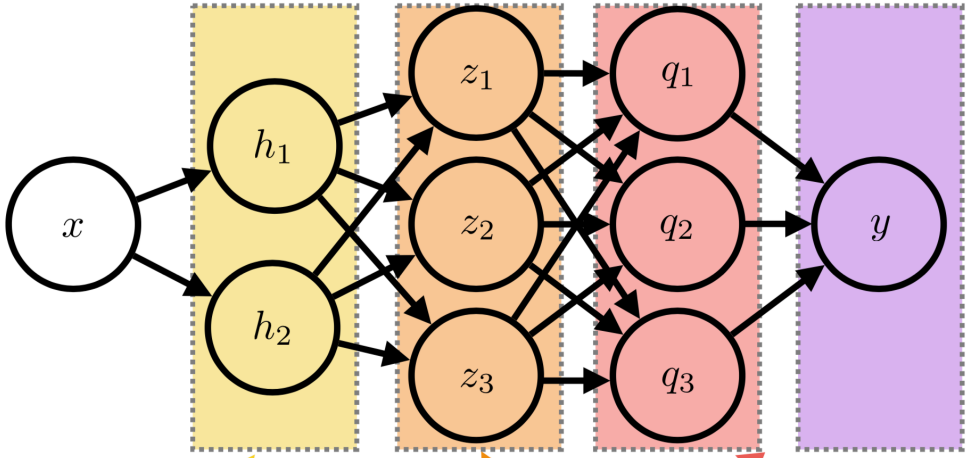
3) Optimization



Forward propagation

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

3) Optimization

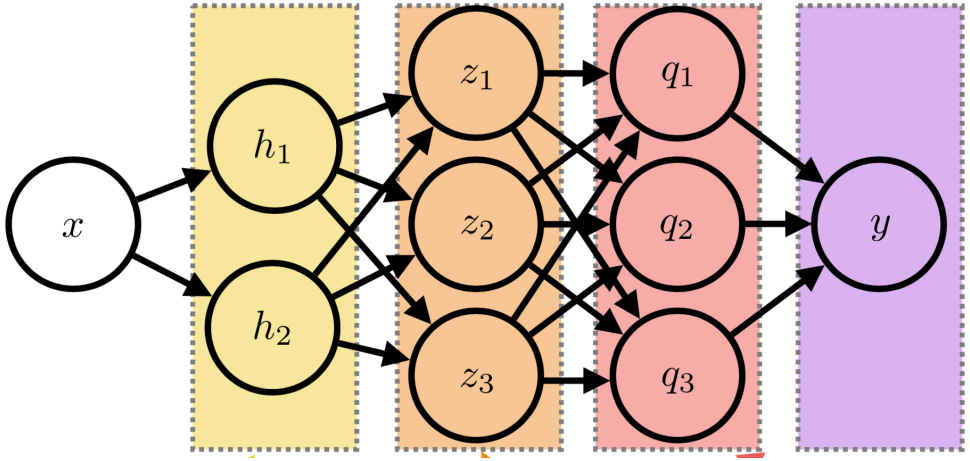


Forward propagation

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

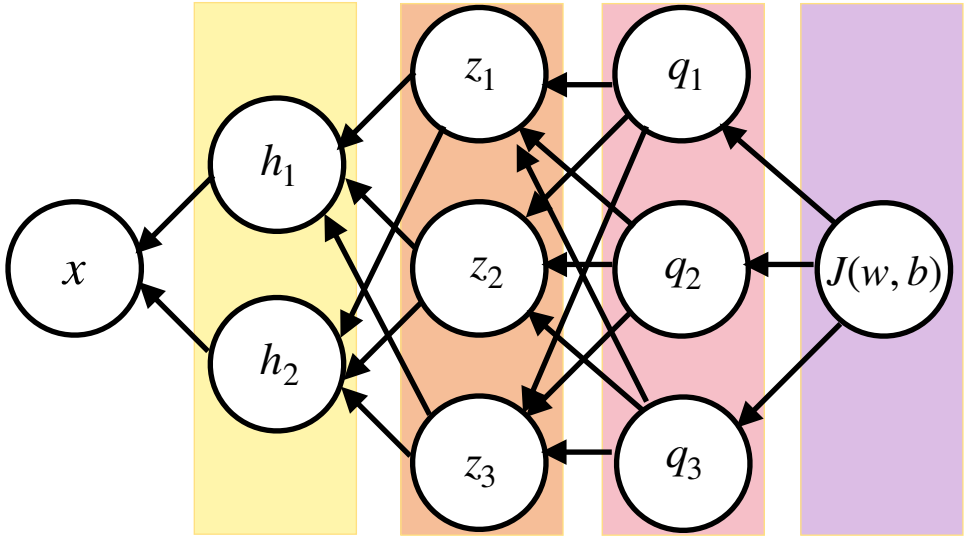
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

3) Optimization



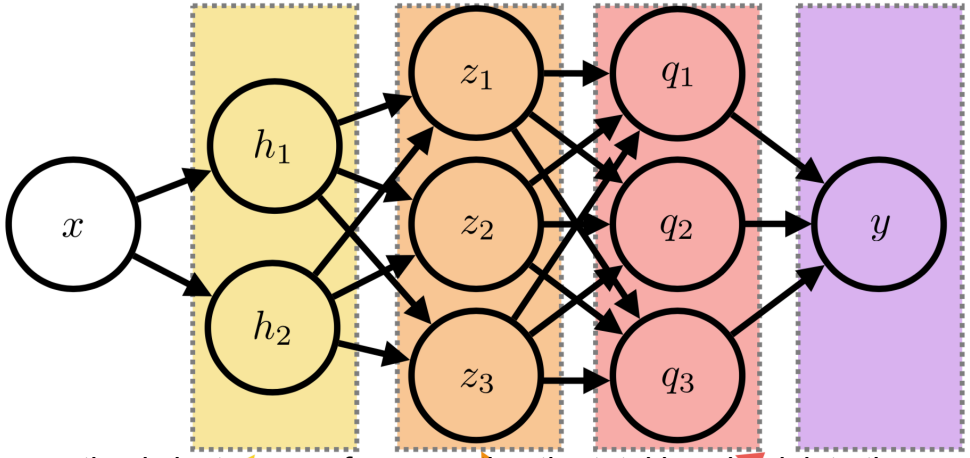
Forward propagation

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



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

3) Optimization



The input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.
Such network configurations are known as feed-forward network.

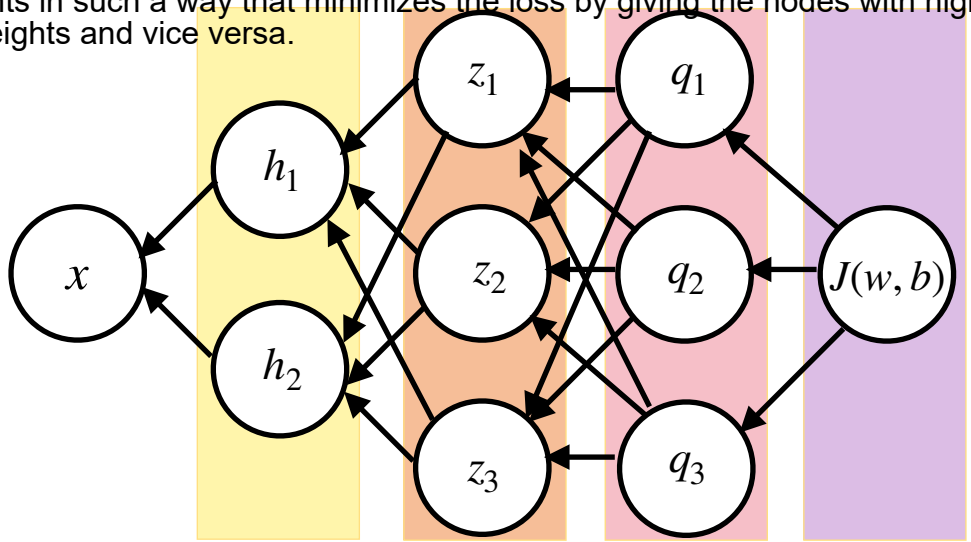
Forward propagation

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

Back-propagation is just a way of propagating the total loss back into the neural network to know how much of the loss every node is responsible for, and subsequently updating t he weights in such a way that minimizes the loss by giving the nodes with higher error rates lower weights and vice versa.

Back-propagation is the essence of neural net training. It is the practice of fine-tuning the weights of a neural net based on the error rate (i.e. loss) obtained in the previous epoch (i.e. iteration). Proper tuning of the weights ensures lower error rates, making the model reliable by increasing its generalization.

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



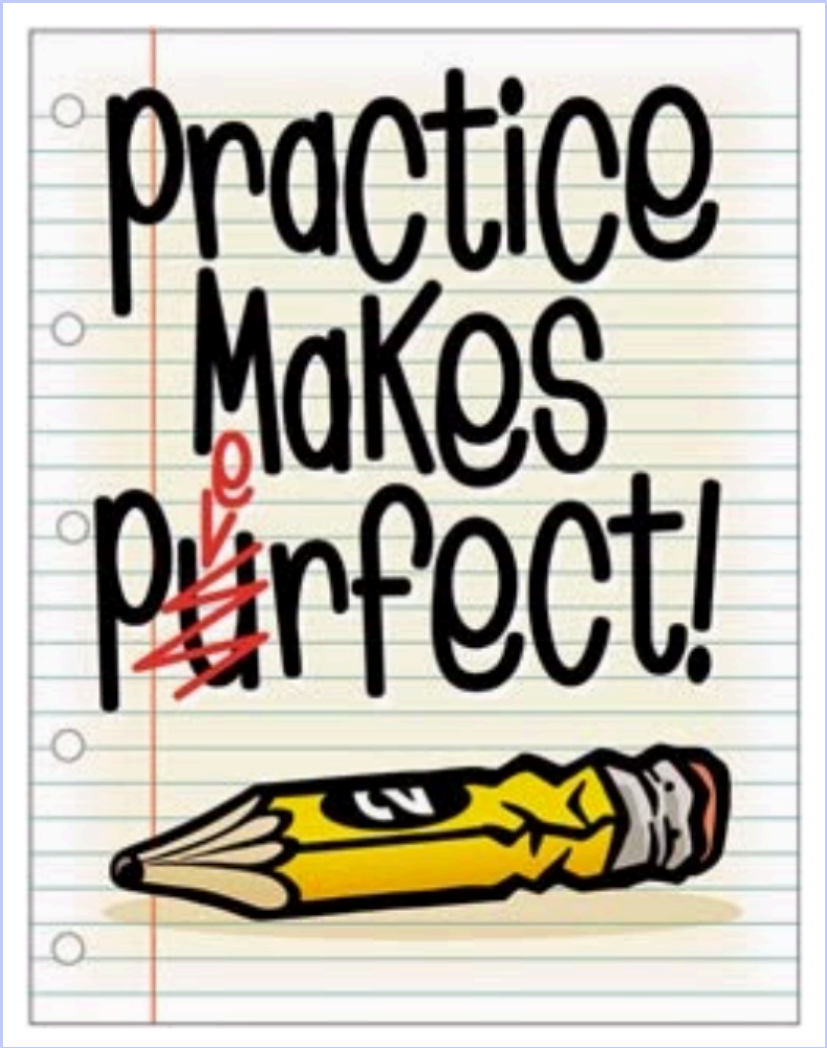
Back propagation

Back-propagation is all about feeding this loss backwards in such a way that we can fine-tune the weights.
The optimization function (Gradient Descent in our example) will help us find the weights that will — hopefully — yield a smaller loss in the next iteration.

Let’s start playing !

Tutorial 1: 9h-9h45

Introduction to Tensorflow

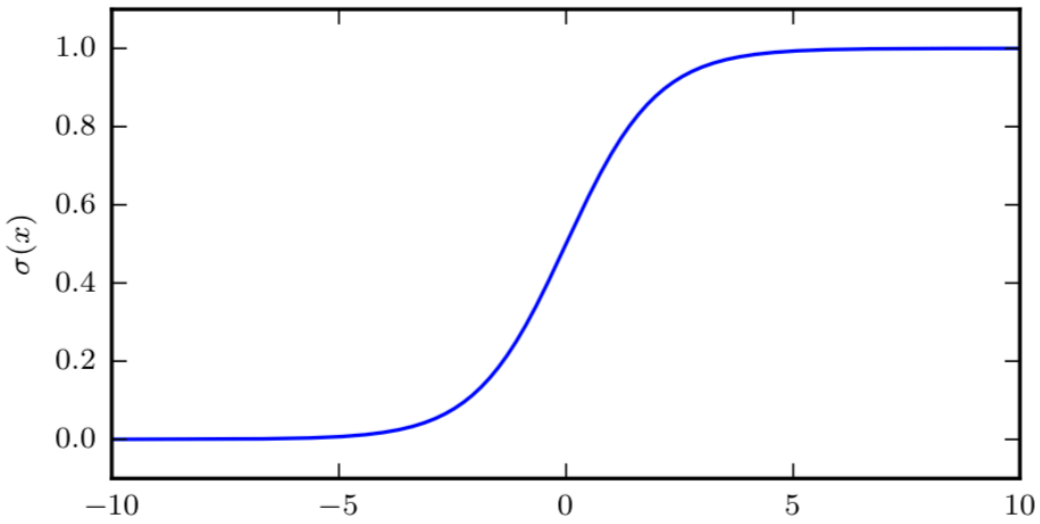


Back-up slides

Sigmoid and softmax

Sigmoid (*two-class* classifier) :

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



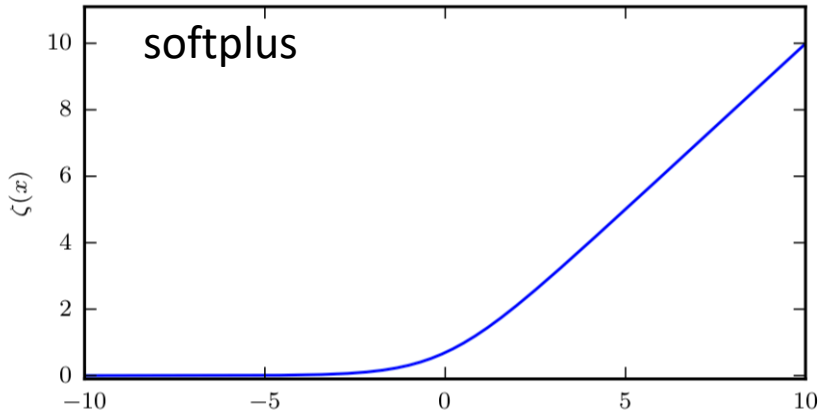
Softmax (*multi-class* classifier) :

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

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) :

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