University of Guilan

Computational Intelligence

Subject3: ANNs architectures, models, and parameters

Instructor: Ali Tourani

A.Tourani1991@gmail.Com

Agenda

- ANNs basic terms
- Working with data
- Neural Networks types
- Linear classification
- Gradient descent algorithm

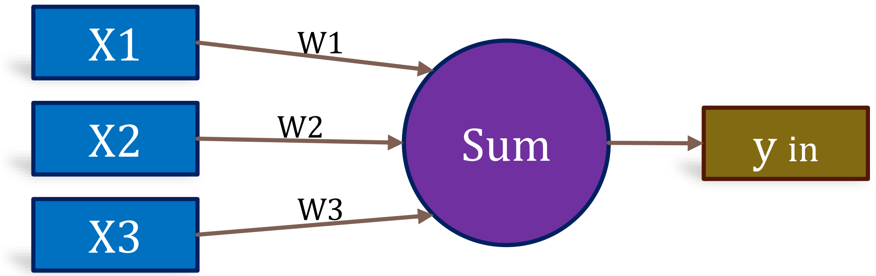
Before we begin …

- Teaching Assistants of the course:
  - Amir Abbasi (amir.abbasi.rose@gmail.com)
  - Reza Khan Mohammadi (rezanecessary@gmail.com)
- Join the **Telegram Group** of the course to get updated
- Introduce your final project team members
  - **Deadline: October 15, 2020 – 23 Mehr 1399 (Next Wednesday)**
  - Send an email to introduce your team **(Student-ID + Full-Name)**
  - Maximum number of members: **3**

ANNs Basic Terms

Let's review what we have learned …

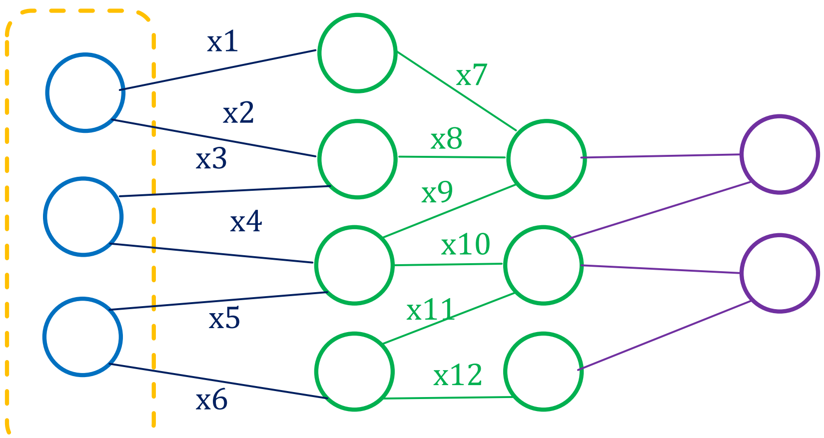
- ***Neurons:*** the basic unit of the neural network
  - A summation function to sense the input signals
  - An Activation Function (AF) to generate outputs within an acceptable range
    - Samples: Sigmoid, TanH, ReLU, etc.



ANNs Basic Terms

Let's review what we have learned ...

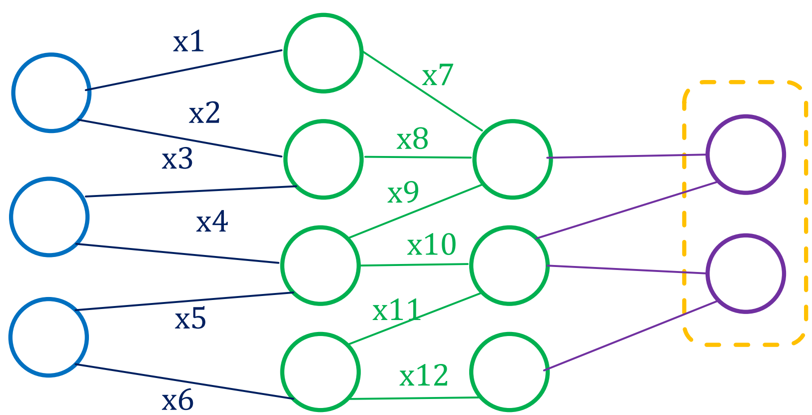
- ***Inputs:*** source data fed into the neural network
  - Why? To make a decision or predict about data
  - Each input is fed to a single neuron in the input layer



ANNs Basic Terms

Let’s review what we have learned …

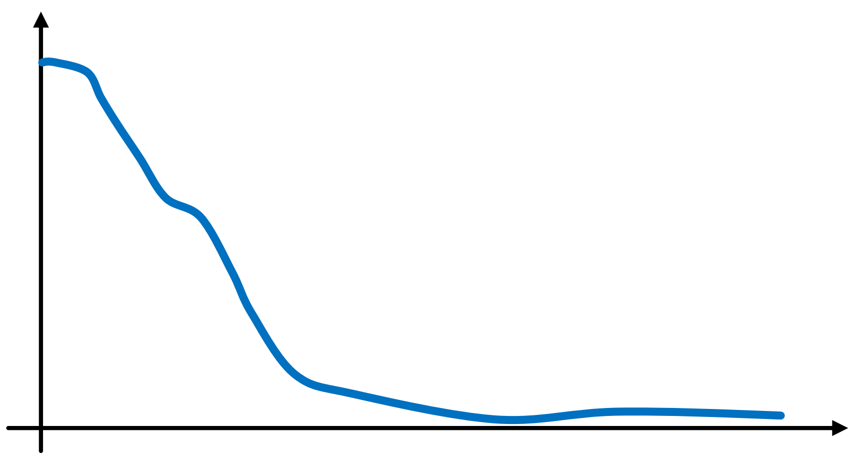
- ***Outputs:*** generated data
  - Typically a set of real values or Boolean decisions
  - Each output is generated from a single neuron in the output layer



ANNs Basic Terms

Let's review what we have learned ...

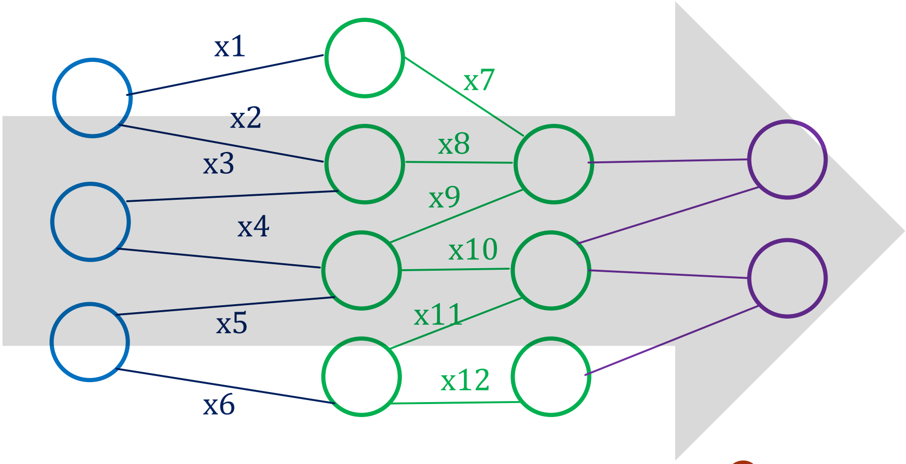
- ***Error Function:*** shows how far the actual output is from the predicted
  - Goal#1: to minimize the error function
  - Goal#2: to bring output as close as possible to the correct value



ANNs Basic Terms

Let's review what we have learned ...

- ***Forward Pass (Propagation):*** the main flow of calculation
  1. Taking the inputs
  2. Passing them through the networks for process
  3. Generate outputs
  4. Pass them to the next layer
  5. Do while generating the F.O

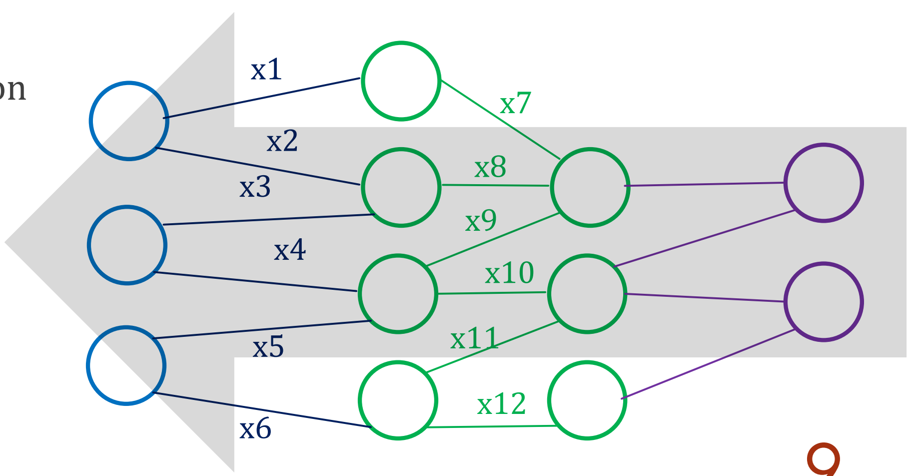


ANNs Basic Terms

University of Guilan

Let’s review what we have learned …

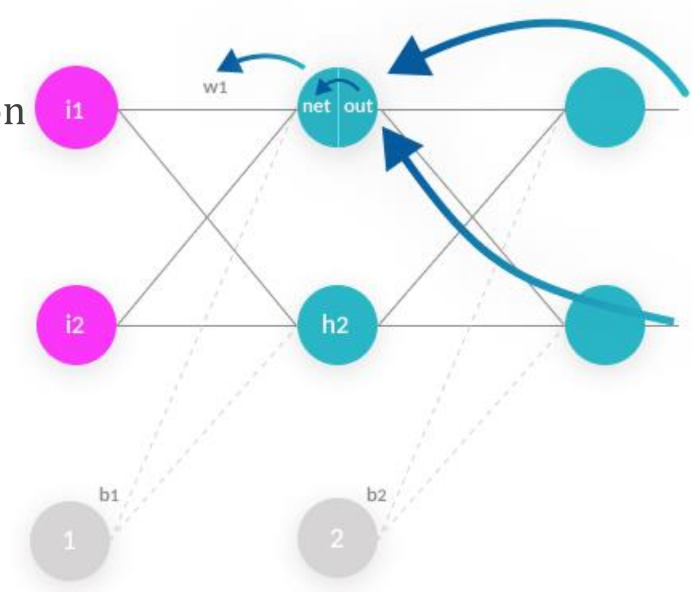
- ***Backpropagation:*** the backward pass
  - Why?
    - To discover the optimal weights for the neurons
    - To minimize the loss function
    - To generate the best prediction
  - How? by tracking the AFs
  - **Gradient descent** algorithm



ANNs Basic Terms

Let's review what we have learned …

- ***Backpropagation:*** the backward pass
  - Forward pass: generating the initial prediction
  - Backward pass: discovering the optimal weights based on the error function
  - Don't worry! There are implemented libraries for backpropagation in TensorFlow or Keras!



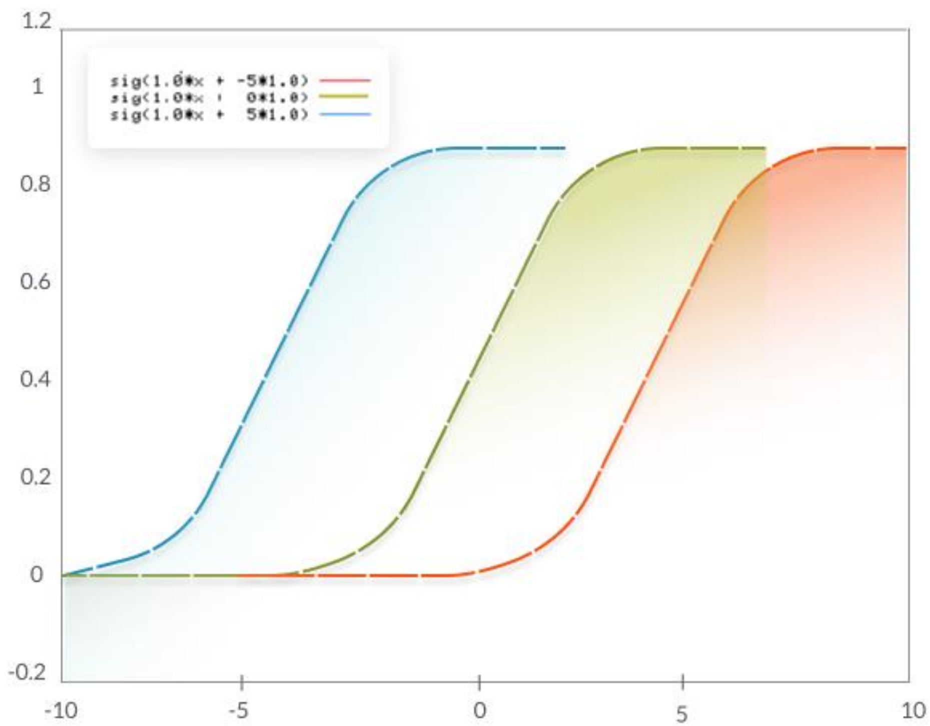
ANNs Basic Terms

- ***Bias (neuron):*** the neuron to move the AF in the graph
  - A special neuron added to each layer in the neural network
    - Note: commonly it is employed in the input layer only
  - The value of the bias neuron = 1
  - They also carry a weight (w_b)
  - Why do we use it?
    - It is essential to move the AF to the left/right/up/down
    - Without it we might face many problems
      - Especially when dealing with zero-valued signals
    - To make the output more reliable by allowing them to employ more complex logic

University of Guilan

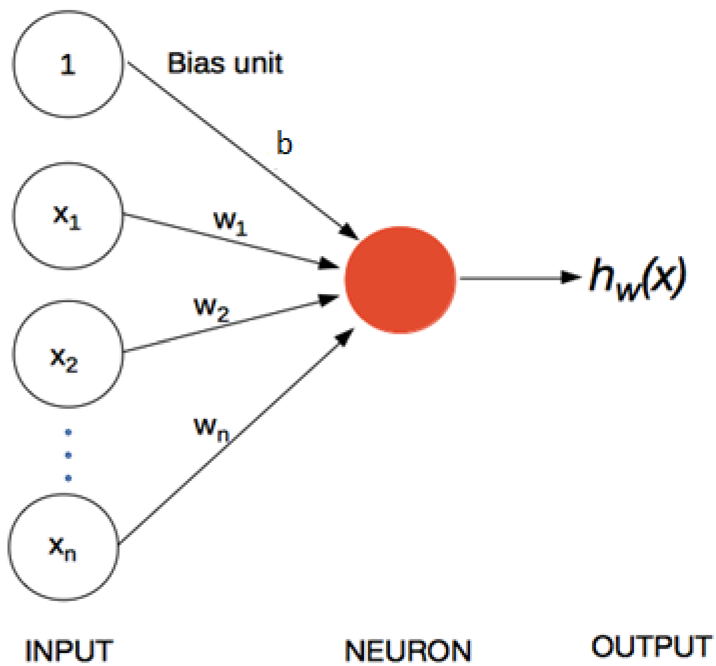
ANNs Basic Terms

- ***Bias (neuron):*** the neuron to move the AF in the graph



ANNs Basic Terms

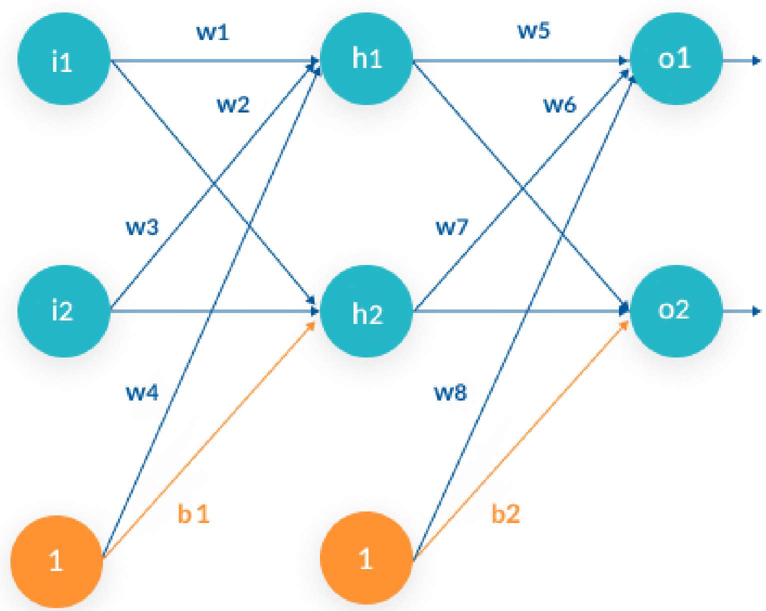
- ***Bias (neuron):*** the neuron to move the AF in the graph



$$y_{in} = (1 \times b) + \sum_{i=1}^{n} x_i w_i$$

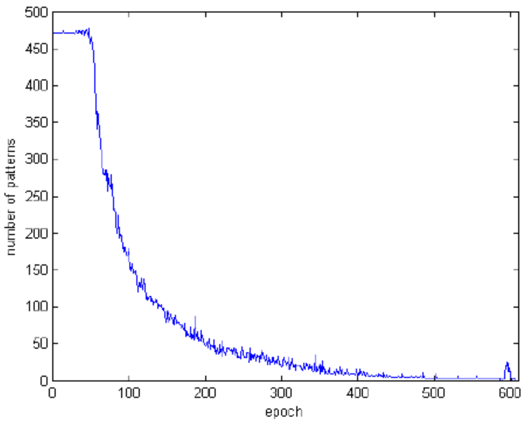
ANNs Basic Terms

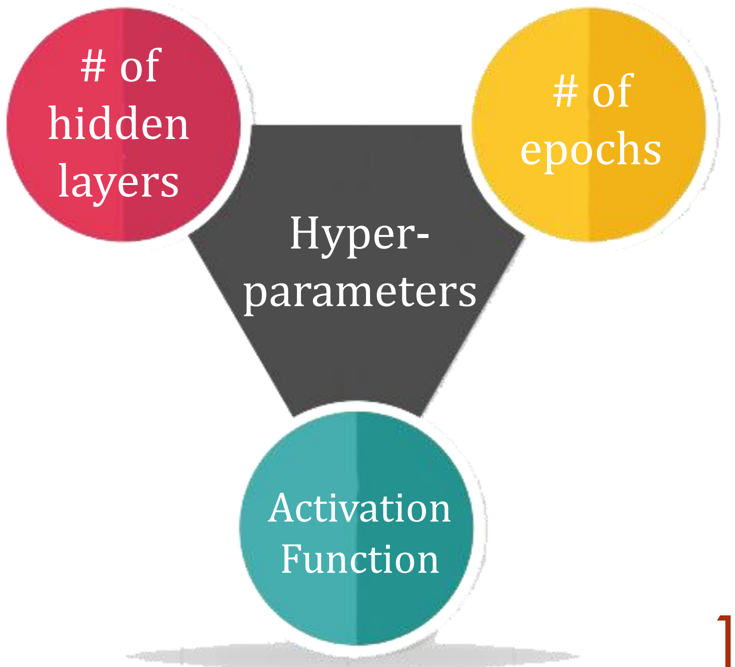
- ***Bias (neuron):*** the neuron to move the AF in the graph
  - Note: bias neurons can be added to each layer of the NN



Working with Data

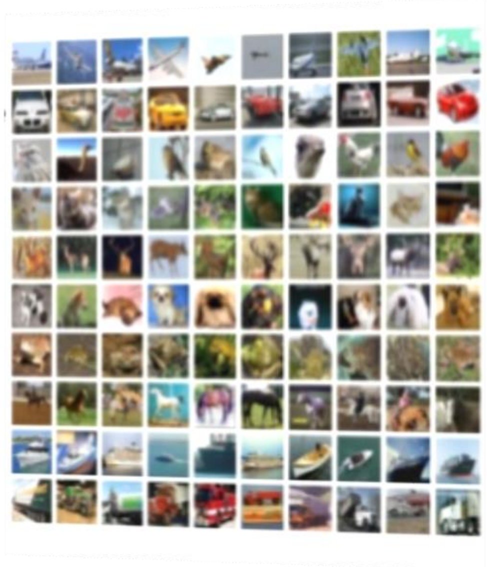
- ***Hyperparameters:*** the setting or configuration of the ANNs
  - Note: tuning them is very important!
  - Benefits of a good setting:
    - Accurate predictions
    - Magnificent results





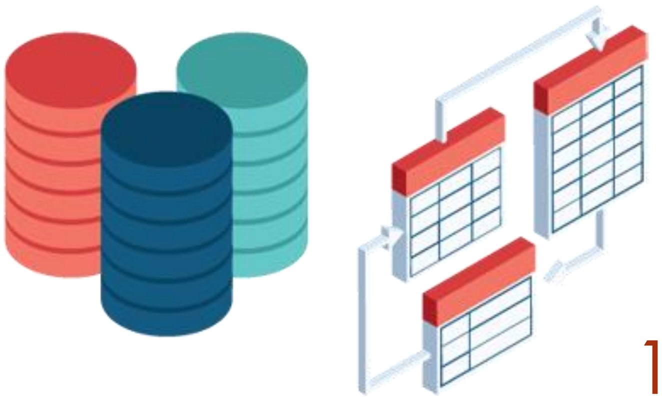
Working with Data

- ***Training-set:*** a set of inputs used to train the neural network
  - A dataset of examples used during the learning process
  - Used to fit the parameters (e.g., weights) of the NN
  - Should follow a general probability distribution
  - Supervised approaches (the correct outputs are known)
  - Large amounts of instances (dataset size)
    - May depend on variation of classes
  - Find some of them here



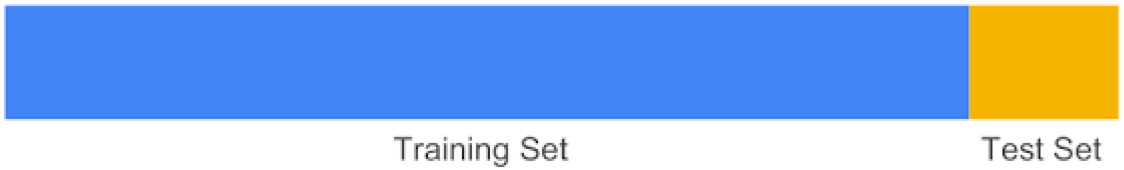
Working with Data

- ***Validation-set:*** a dataset of examples used to <u>tune the hyperparameters</u>
  - Should follow the same probability distribution as the training-set
  - Usage:
    - Decreasing the probability of **overfitting** *(what is it?!)*
    - Evaluation of the error function using another dataset
  - AKA “development-set” or “dev set”



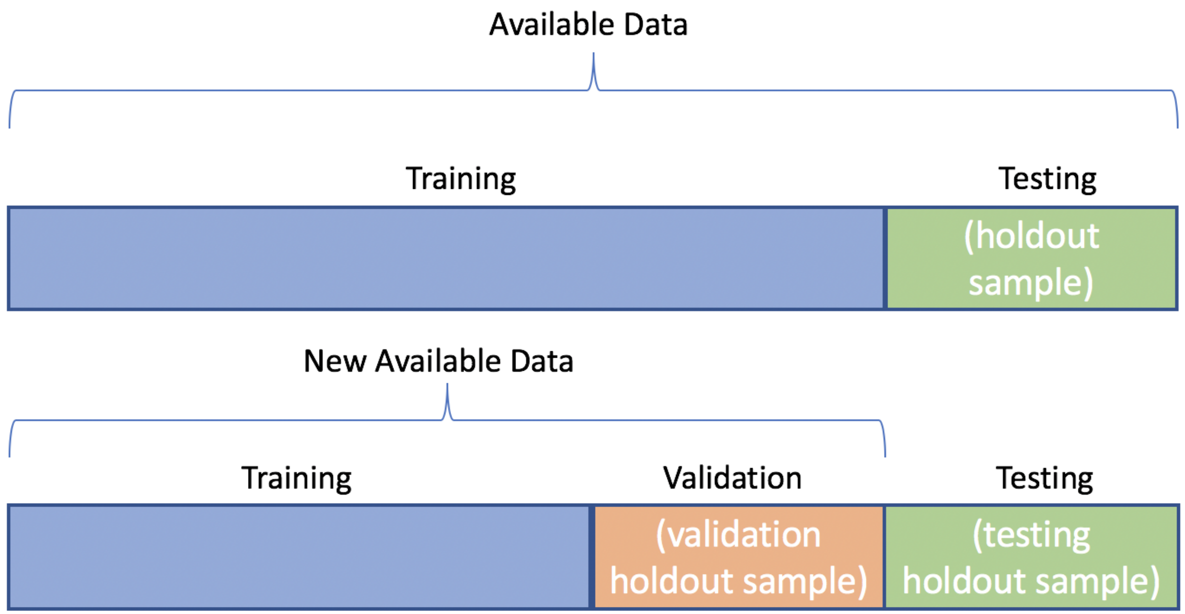
Working with Data

- ***Test-set:*** a dataset of examples used
  - Used to assess the performance of the NN
  - Independent of the training dataset
  - Should follow the same probability distribution as the training-set
  - **Holdout:** isolating a part of the training-set for test *(evaluation)* as the test-set and do not use them for training the network



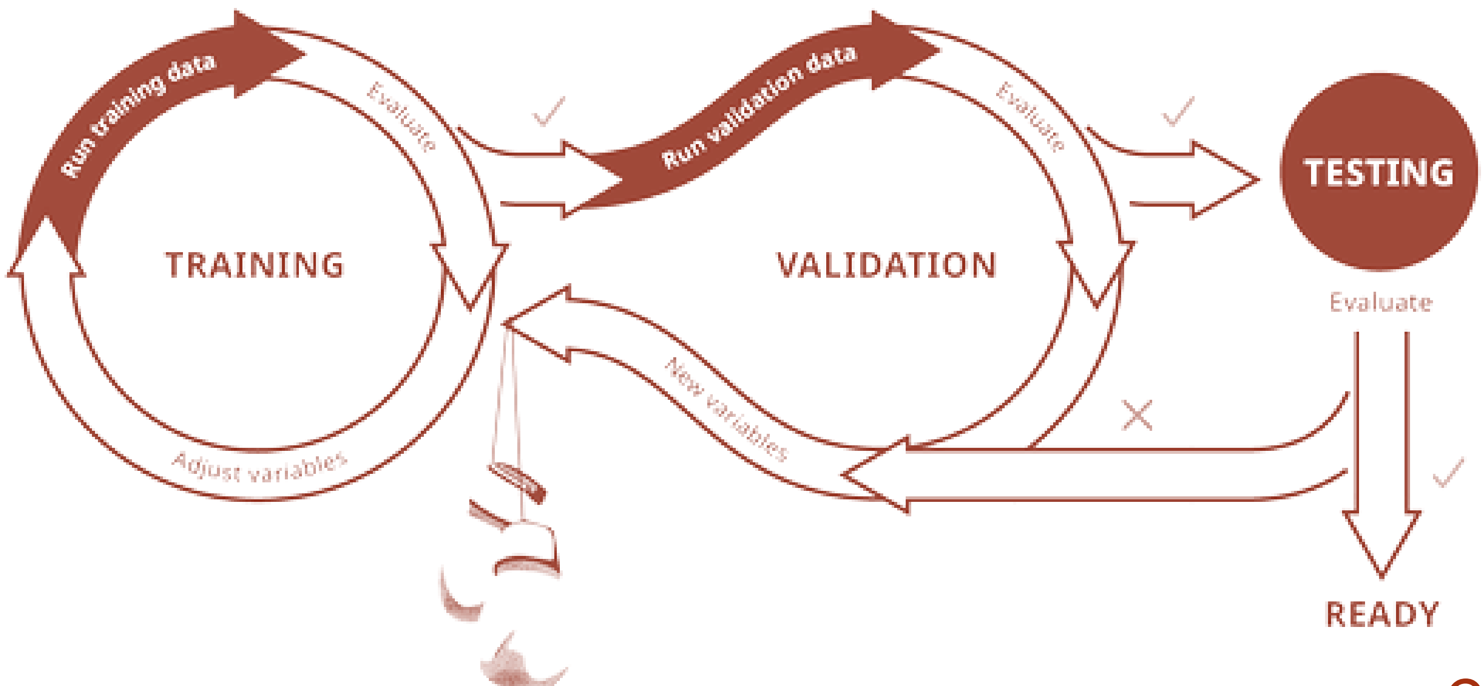
Working with Data

- ***Training-set, Validation-set, and Test-set***



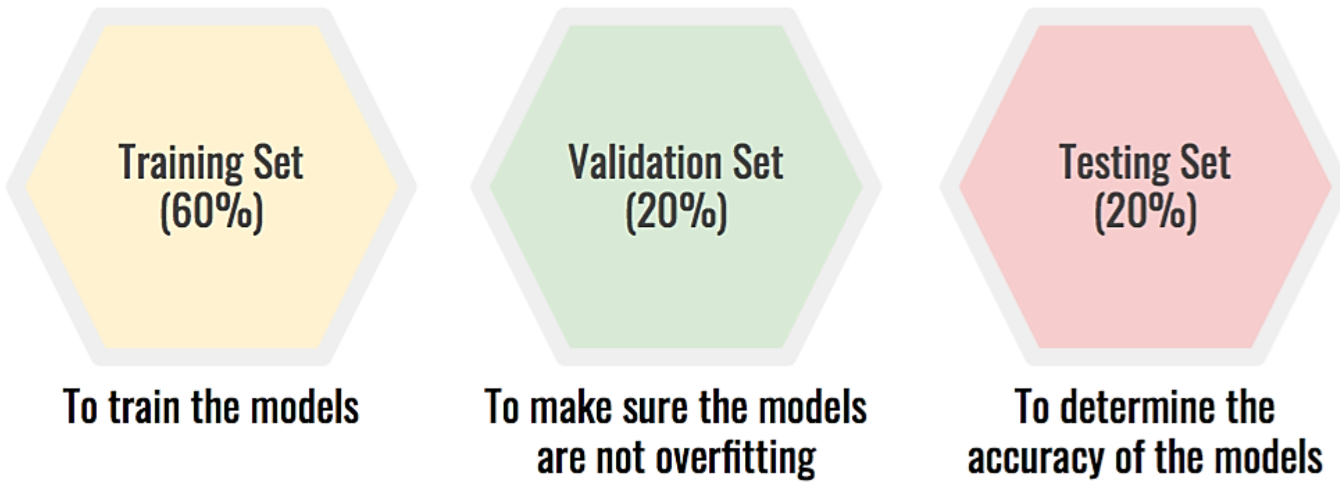
Working with Data

- ***Training-set, Validation-set, and Test-set***



Working with Data

- ***Training-set, Validation-set, and Test-set***



Working with Data

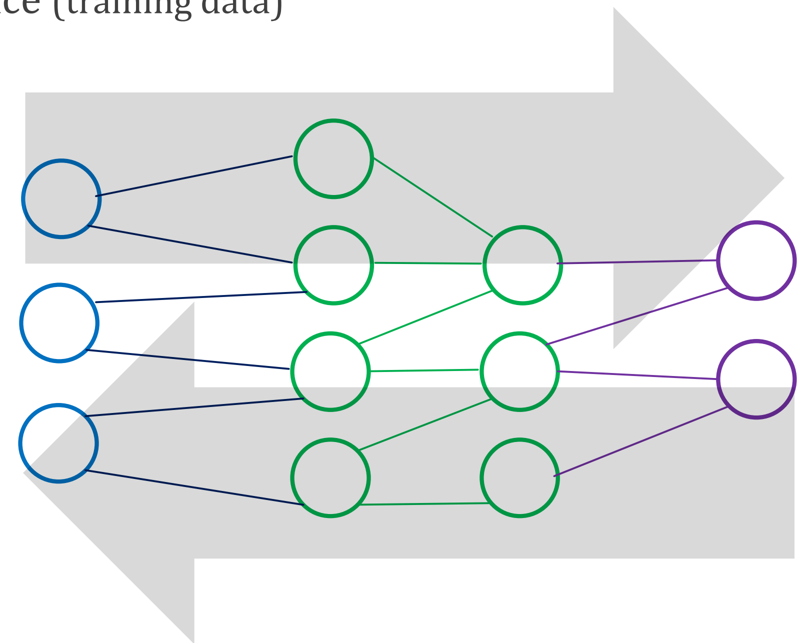
- ***Pass in NNs***
  - A forward and backward pass
- ***Batch size***
  - The number of training data used in one forward/backward pass
- ***Iteration***
  - Number of passes, each pass using [batch size] number of instances
- ***Epoch***
  - The number of times the algorithm sees the entire training set
  - Forward and backward pass of all the training instances

Working with Data

- ***Pass, Batch-size, Iteration, and Epoch***
  - ***Example***: 10,000 images of human face (training data)
  - *Batch-size*: 500
  - *# of iterations*: 20
  - We need 20 iterations to complete

  One epoch

  - Normally, we need several epochs



Working with Data

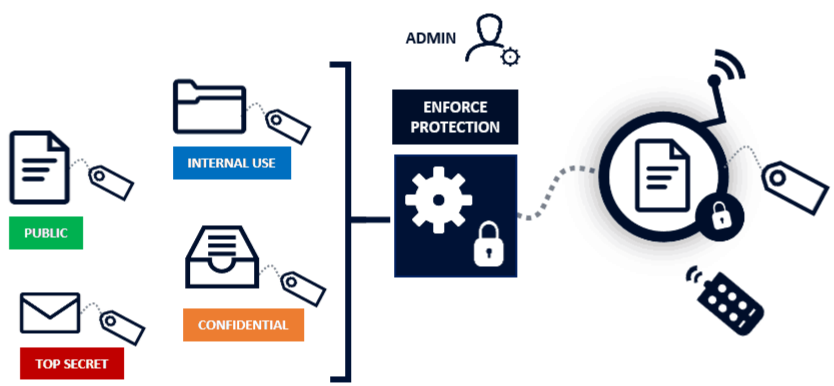
- ***Overfitting problem***
  - Exploiting relationships in the training data that do not hold in general
  - Generalization Problem:
    - The NN learns very good from the training-set, but it cannot generalize beyond it
    - Accurate predictions for the training-set
    - Inaccurate predictions for the validation-set and test-set
  - *Minimal overfitting:* a model fit to the training-set also fits the test-set
- ***Underfitting problem***
  - the network is not able to generate accurate predictions on the training set

Working with Data

- ***Overfitting problem – How to avoid?***
  - Method 1: Retraining Neural Networks
    - Running the same NN model on the same training set, but each time with different initial weights to find the lowest performance (more general) NN
  - Method 2: Multiple Neural Networks
    - Running several NNs in parallel on the same training set, but with different initial weights and calculate the average of outputs
  - Method 3: Early Stopping
    - Monitor the error on the validation set after each training iteration to see when the error increases

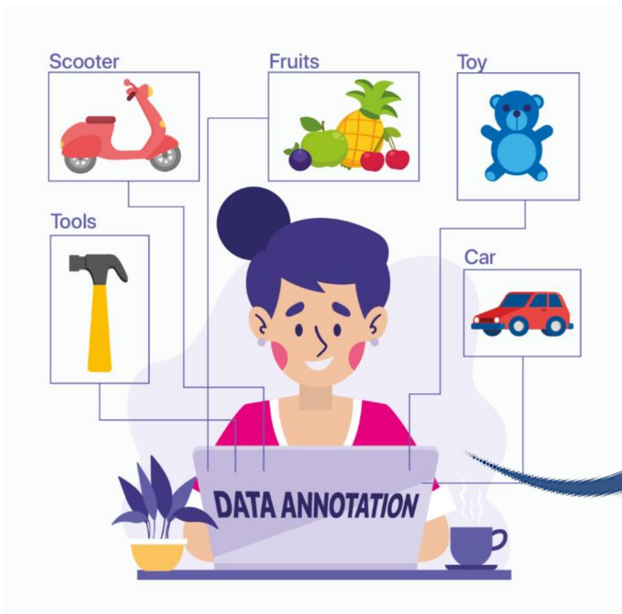
Working with Data

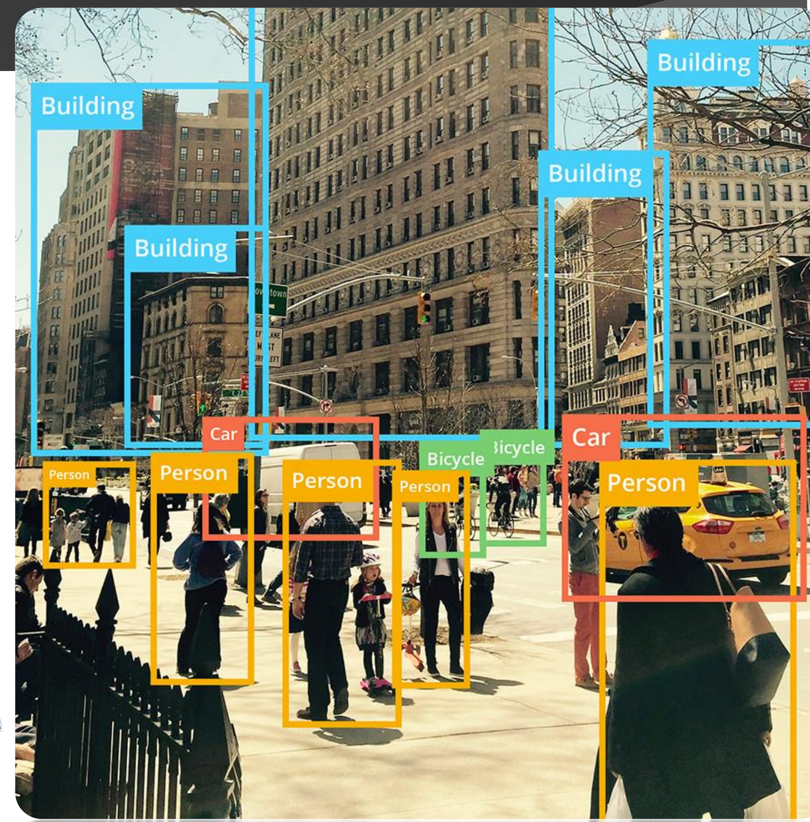
- ***Data labeling (data annotation)***
  - The process of marking up or annotating data to show the target
  - The target is the answer you want your model to predict
  - Sample tasks: tagging, annotation, moderation, or classification



Working with Data

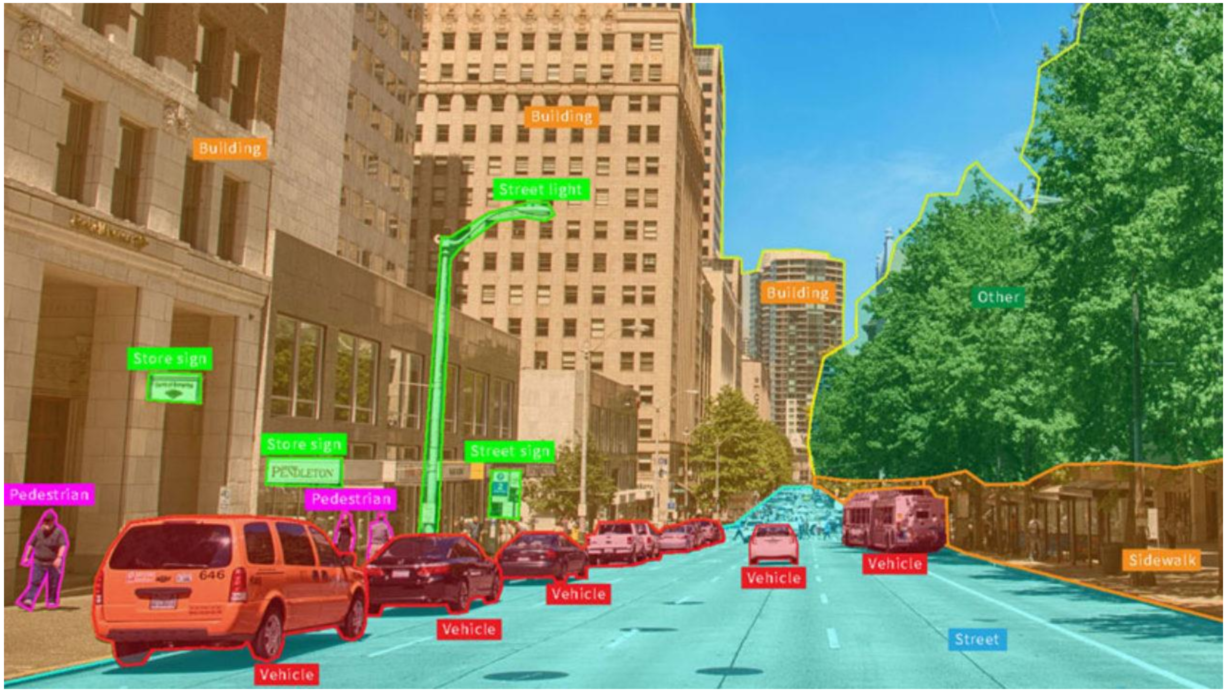
- ***Data labeling (data annotation)***
  - Manually/Automatically labeling data



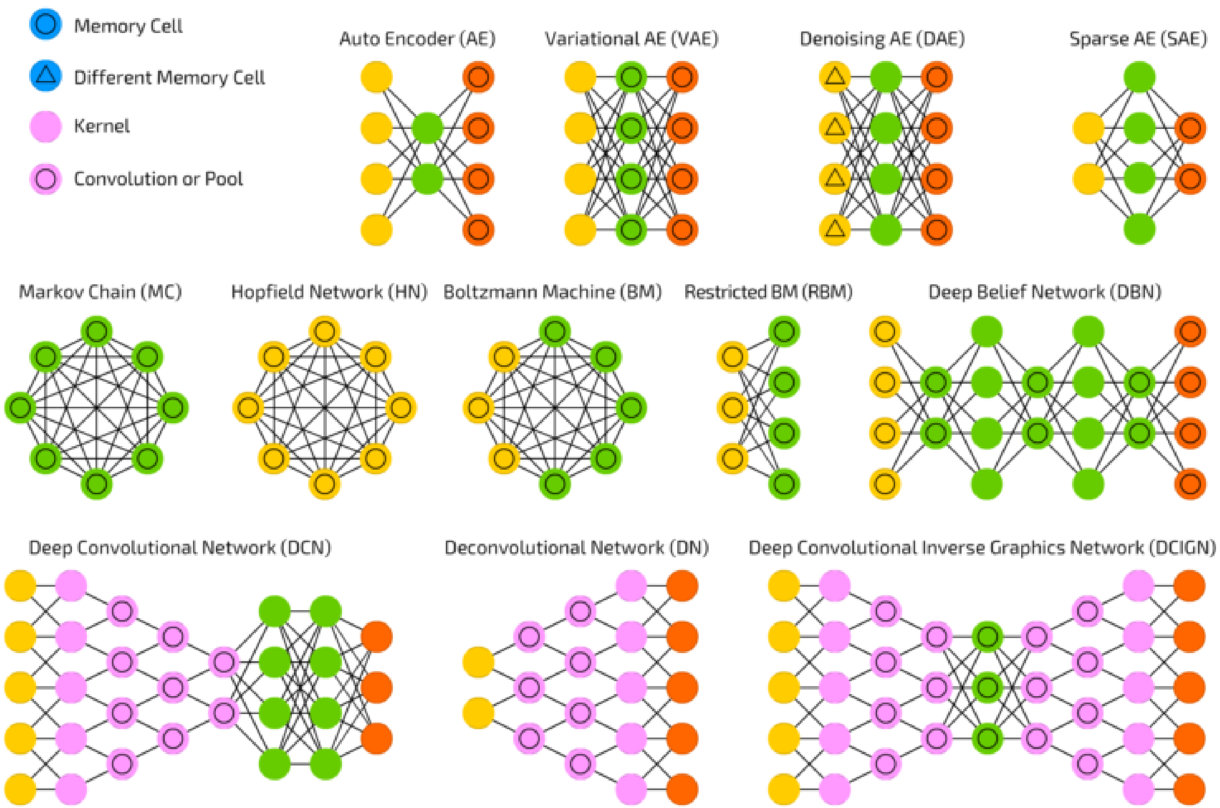


Working with Data

- ***Data labeling** (data annotation)*



Neural Networks Types

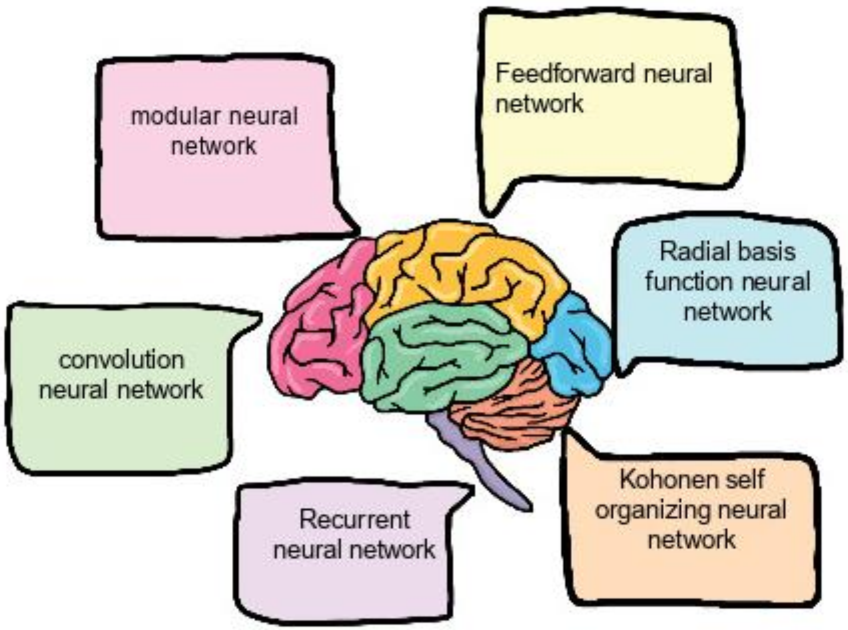


Reference: https://kim.hfg-karlsruhe.de/neural-network-chart/

Neural Networks Types

Typical Neural Networks

- Feed Forward
- Recurrent
- Competitive
- Probabilistic



Neural Networks Types

Typical Neural Networks

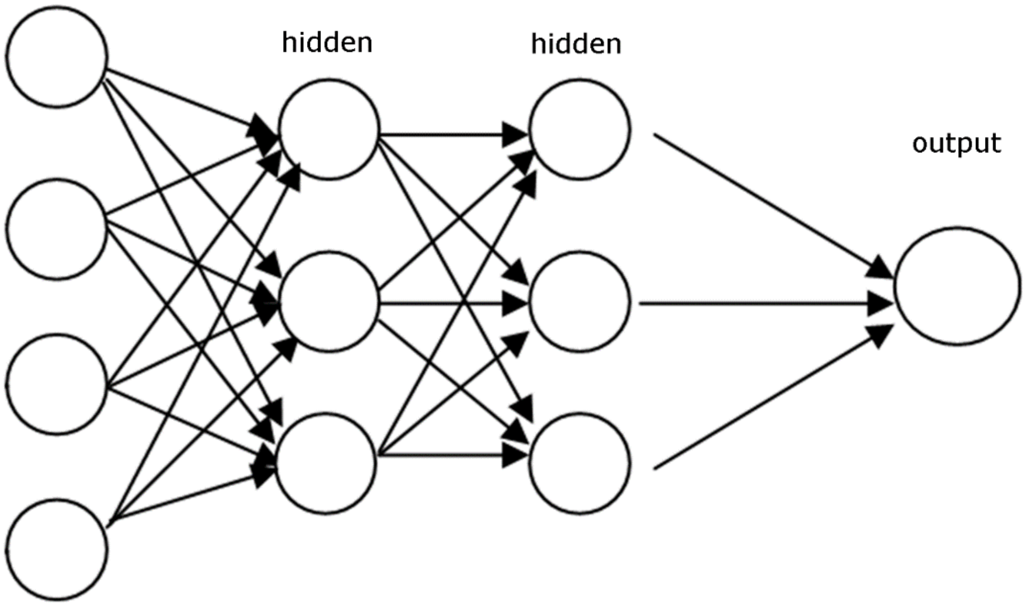
- Feed Forward
  - All paths from the input nodes to the output layer (no loop)
  - The information moves in **only one direction**
  - Connections between the nodes do not form a cycle
  - Samples:
    - Single-layer perceptron
    - Multi-layer perceptron
    - Convolutional Neural Networks (CNNs)
    - Radial Basis Function networks (RBF)

Neural Networks Types

University of Guilan

Typical Neural Networks

- Feed Forward



Neural Networks Types

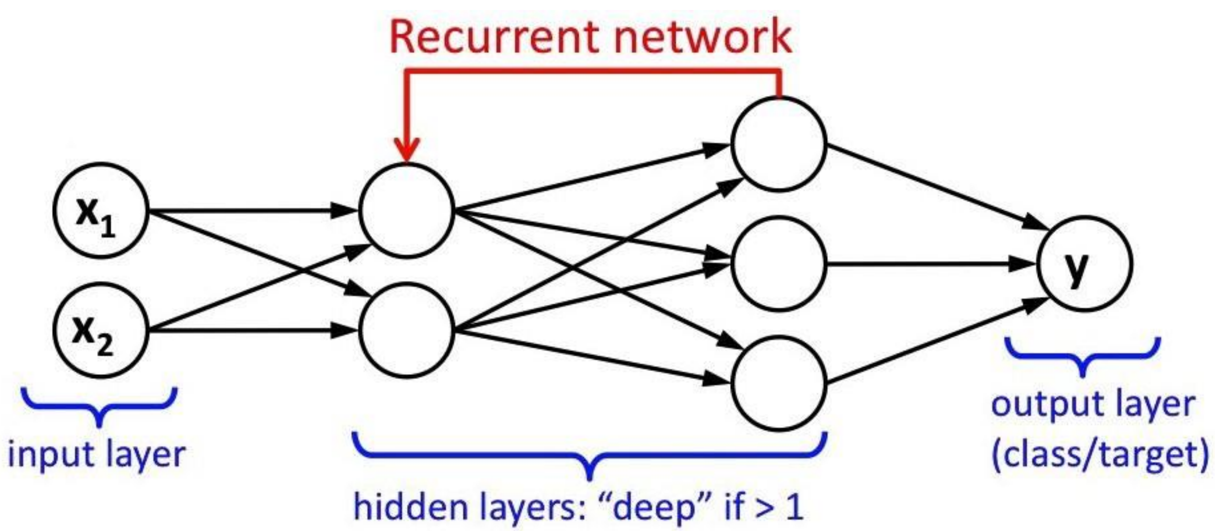
Typical Neural Networks

- Recurrent
  - The information moves in **both forward and backward direction**
  - Connections between the nodes forms a directed graph
  - Samples:
    - Elman and Jordan networks
    - Hopfield networks
    - Recursive neural networks

Neural Networks Types

Typical Neural Networks

- Recurrent



Neural Networks Types

University of Guilan

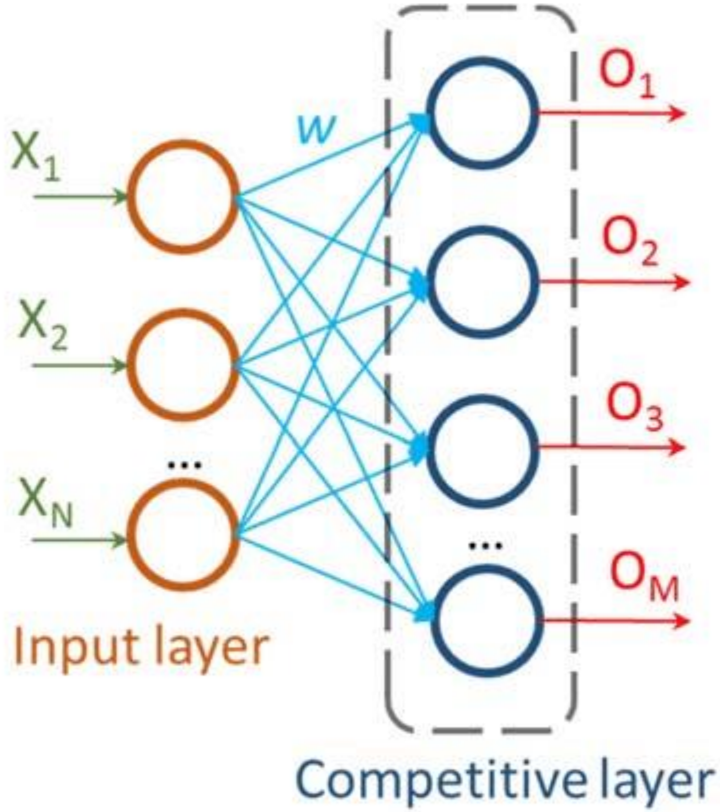
Typical Neural Networks

- Competitive
  - The neurons **compete** to be activated (not just by a simple AF)
  - How? Using a function of distance from a selected data point
  - The highest value: the neuron closest to the data point (strength)
  - High potential to be used in unsupervised learning
  - Samples:
    - Kohonen self-organizing maps

Neural Networks Types

Typical Neural Networks

- Competitive - algorithm
  - Apply input data
  - Find the strongest
  - Increase the weights of the strongest matches
  - Repeat



Neural Networks Types

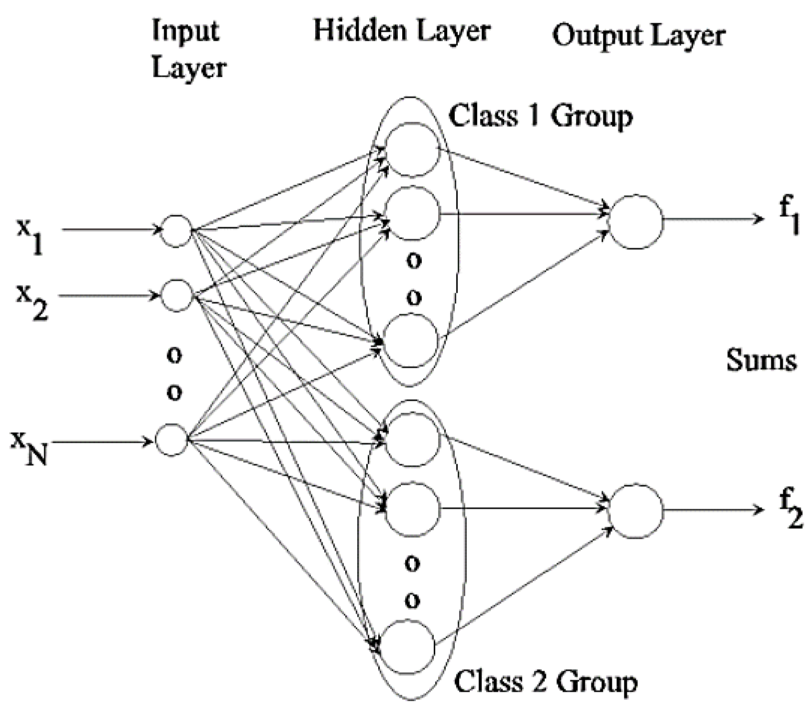
Typical Neural Networks

- Probabilistic
  - Uses exponential functions as AFs
  - Each node in the hidden layer receives data from all input nodes
  - There are $K>0$ classes, placed in the hidden layer,
    - We also have $K>0$ outputs (one output for each class)
  - Each node of the class k corresponds to a probabilistic (e.g. Gaussian) function
  - All of the probabilistic functions are summed to shape the final output

Neural Networks Types

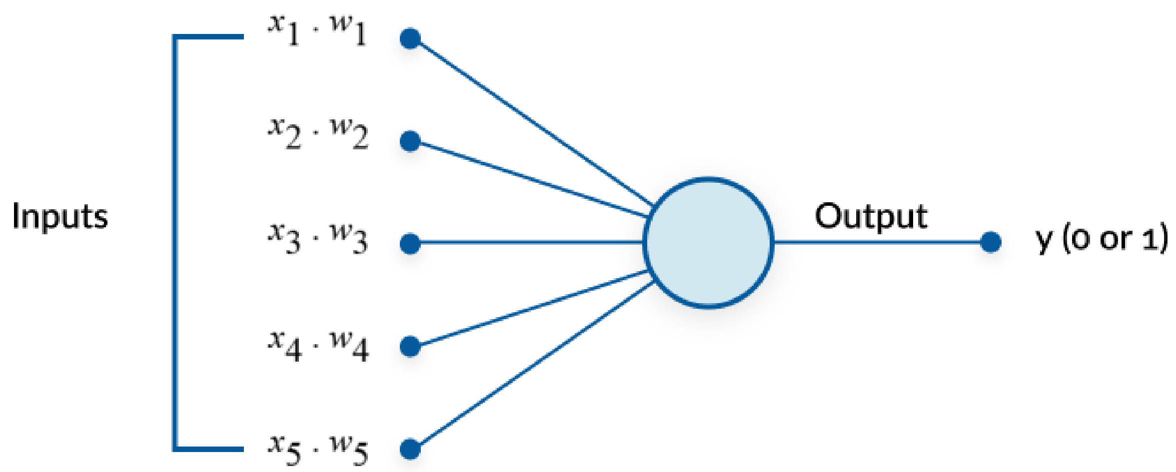
Typical Neural Networks

- Probabilistic



Linear Classification

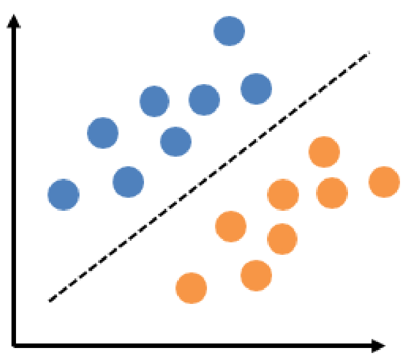
- ***Singe-layer Perceptron:*** the simplest type of a neural network
  - A feedforward neural network with no hidden units
  - Able to learn linearly separable patterns



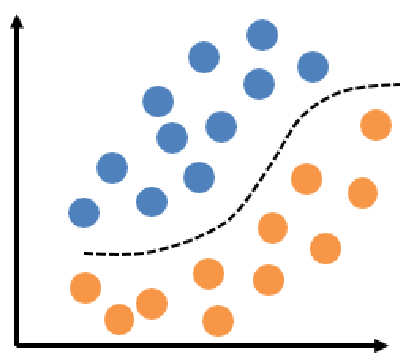
Linear Classification

- Can we divides the input space into two classes (e.g. 0 and 1)?
  - 2D space: a line 3D space: a hyperplane
  - We need to adjust the weights for slope and bias for intercept $y = wx + b$

Linear



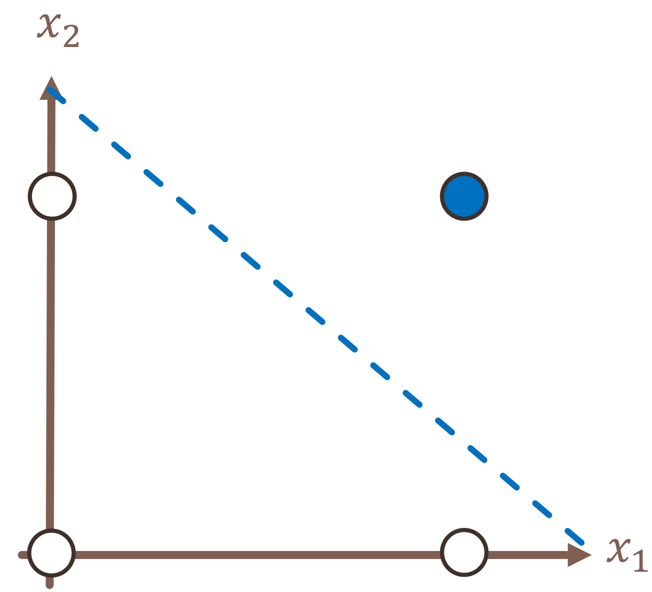
Nonlinear



Linear Classification

- A single-layer perceptron can only solve **Linearly Separable** problems
  - **Binary AND**

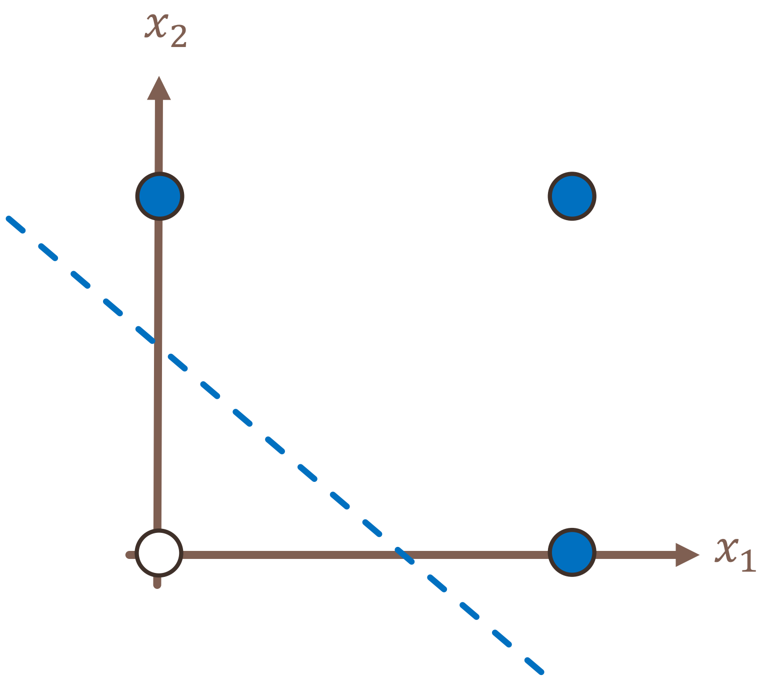
Input		***Output***
x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0



Linear Classification

- A single-layer perceptron can only solve **Linearly Separable** problems
  - **Binary OR**

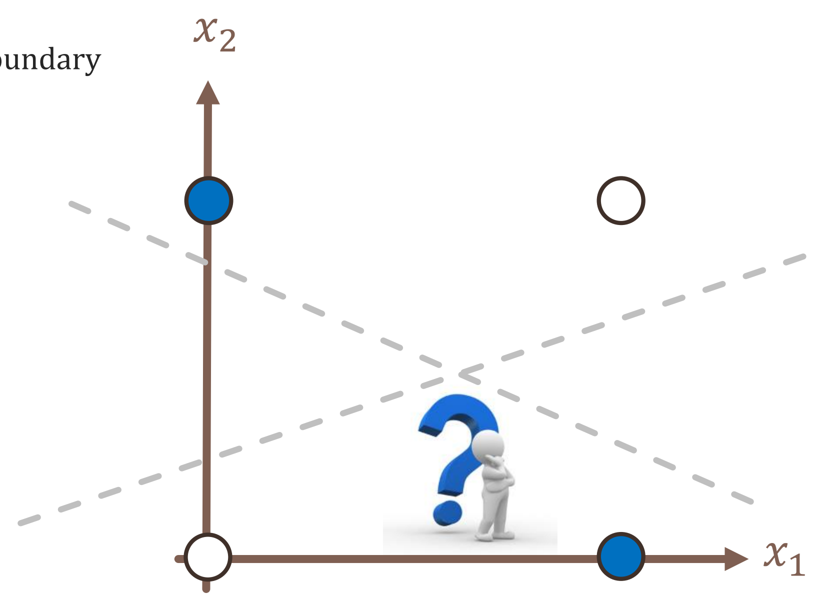
Input		*Output*
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0



Linear Classification

- A single-layer perceptron can <u>only</u> solve **Linearly Separable** problems
  - **Binary XOR????**
    - Impossible to find a linear boundary

Input		*Output*
x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

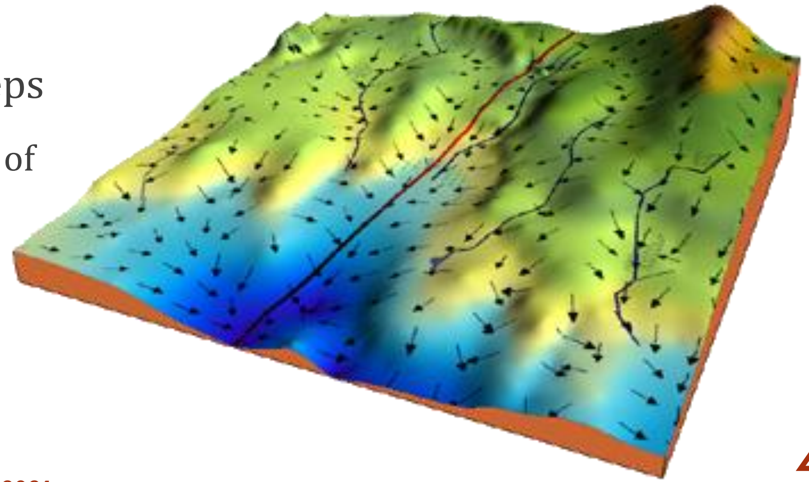


Gradient descent algorithm

- An iterative optimization algorithm for finding a local minimum
  - How? by iteratively moving in the direction of the steepest descent
  - Like moving from a mountain to the sea
  - An step-to-step downhill in the direction with negative gradient

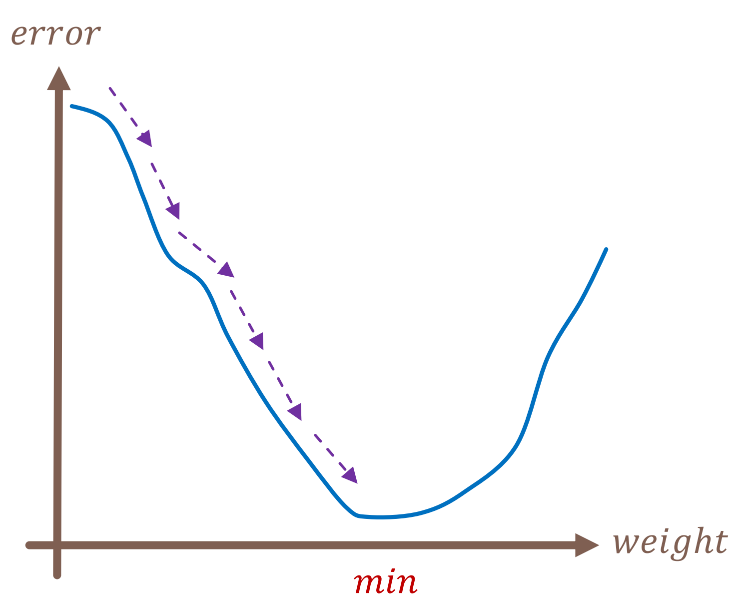
  - *Learning Rate*: the size of steps
    - The higher LR, the more risk of

    finding the local minimum



Gradient descent algorithm

- **Usage**
  - We use this method to predict how good our model is working
  - Cost/Loss function
  - Parameters: weight and bias

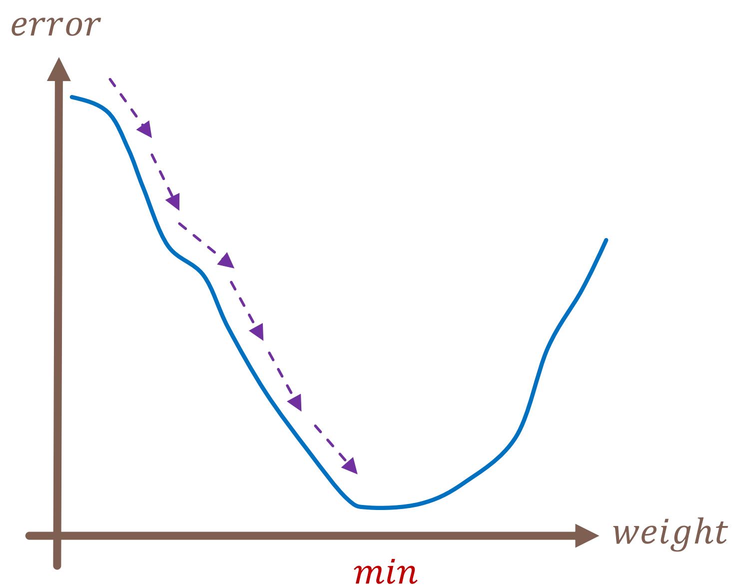


Gradient descent algorithm

- **Calculations**
  - We know how to calculate MSE

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(R_n - P_n)^2$$

  - Real value R_n
  - Predicted value P_n
  - Number of samples N



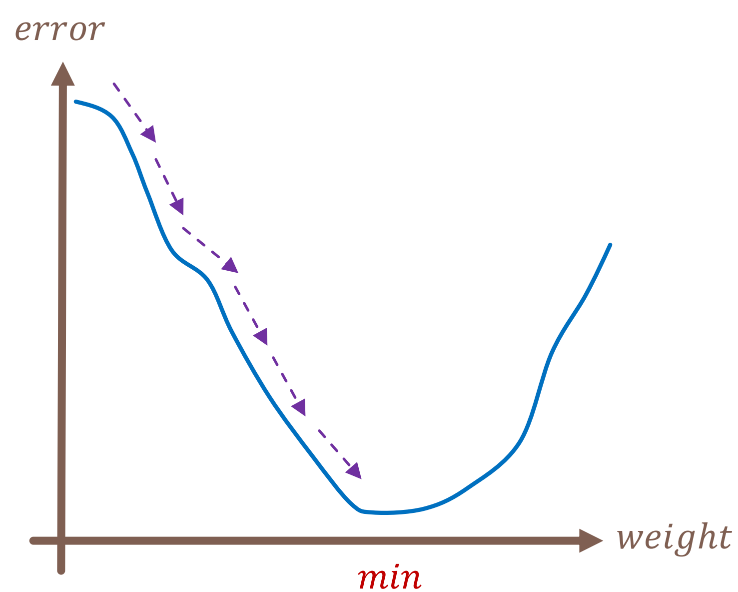
Gradient descent algorithm

- **Calculations**
  - In this regard, for cost function:

$$f(w,b) = \frac{1}{N}\sum_{n=1}^{N}(y_n - (wx_n + b)\,)^2$$

  - Thus, the gradient will be:

$$f'(w,b) = \begin{bmatrix} \frac{df}{dw} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -2x_n(y_n - (wx_n + b)) \\ \frac{1}{N}\sum -2\,(y_n - (wx_n + b)) \end{bmatrix}$$



Gradient descent algorithm

University of Guilan

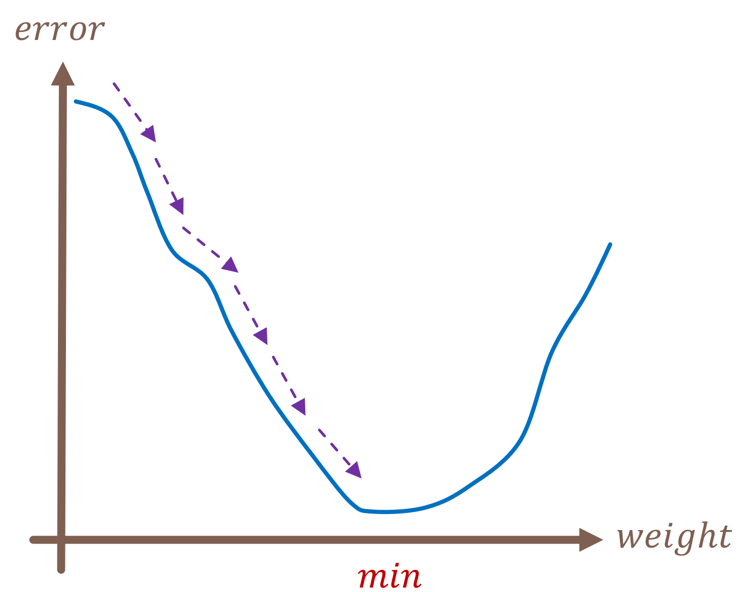
- **Calculations**
  - Relation among weights:

$$w_i(t) = w_i(t-1) + \Delta w_i(t)$$

  - Where $\Delta w_i(t)$ is:

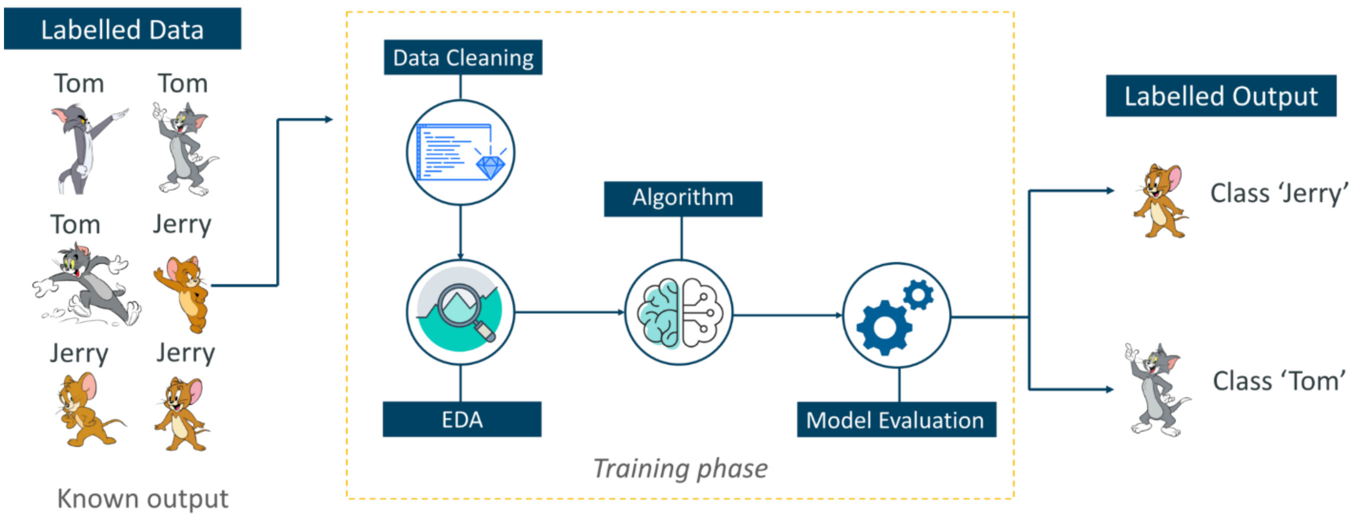
$$\Delta w_i(t) = \mu(-\frac{\delta E}{\delta w})$$

    - Learning rate μ
    - MSE E



What's Next?

- ANNs Learning Process



Questions?