Tufts UNIVERSITY

ARTIFICIAL NEURAL NETWORKS 2

ARTIFICIAL INTELLIGENCE | COMP 131

TODAY ON AI

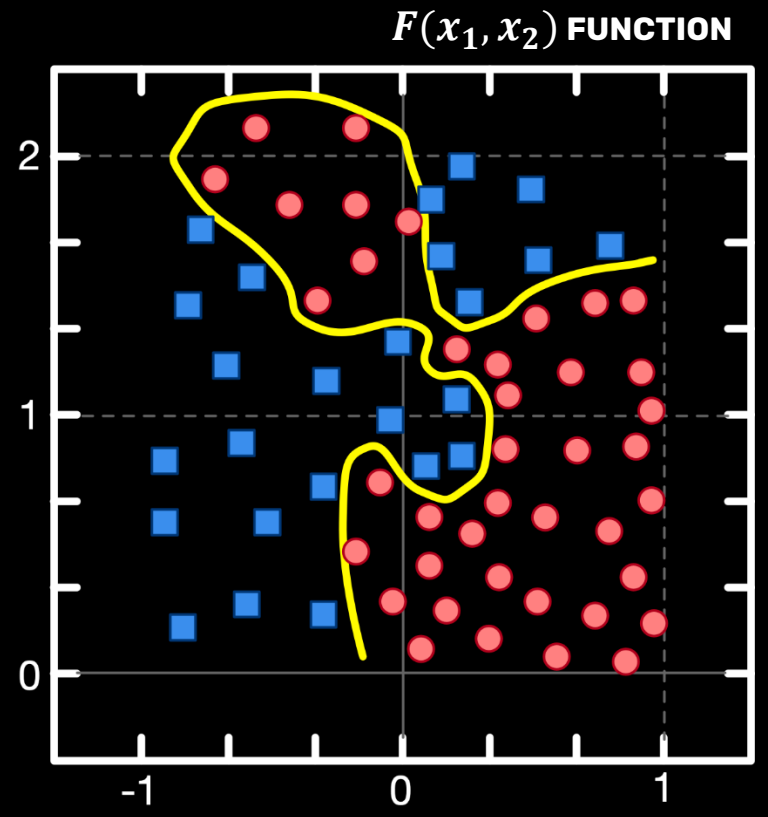
- Generalization
- Designing ANNs
- Architecture of the network
- Data preparation
- Training ANNs
- Questions?

SECTION 01

Generalization

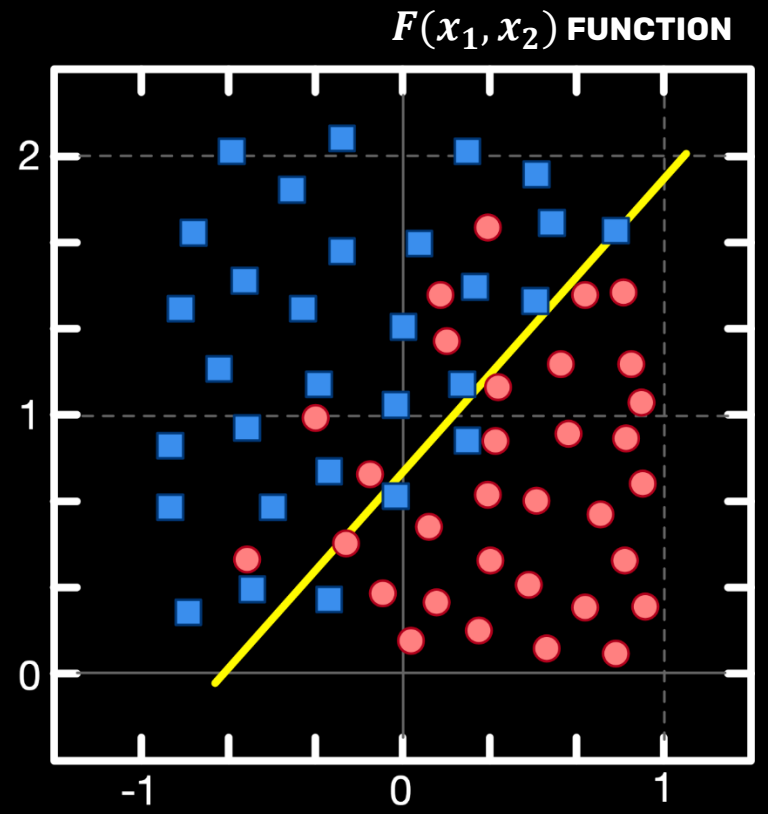
The objective of **learning** is to achieve good generalization to new cases

Generalization is the ability of learning to correctly assess new and unseen data.



It can also be defined as a mathematical interpolation or regression over a set of training points.

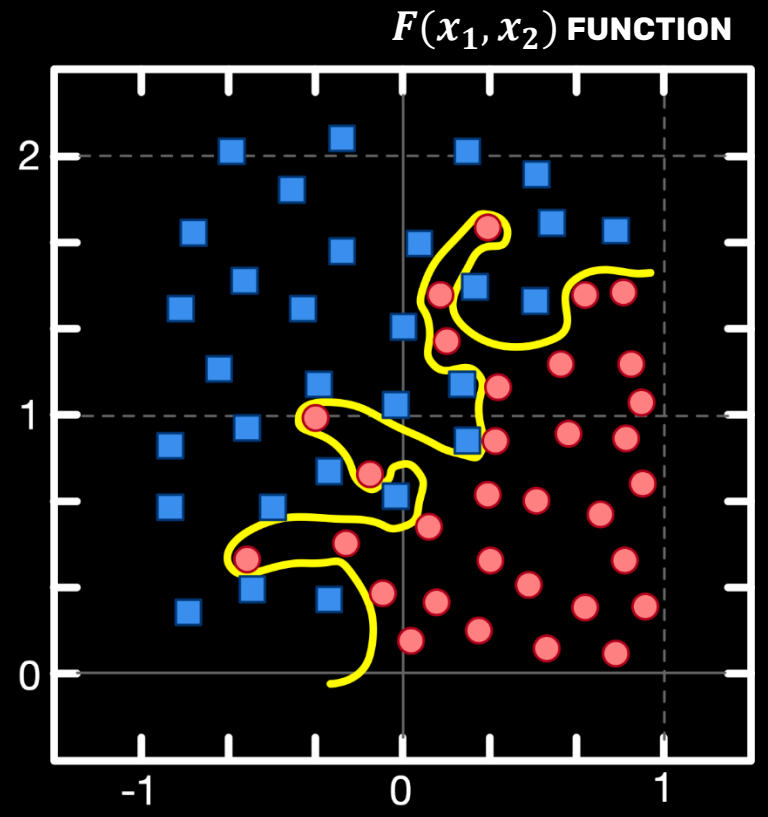
- ANNs cannot always find the perfect function for the problem at hand
- There is (almost) always a small classification error
- A **learned model** helps the system to perform F **better** as compared to no learning.



Let’s assume that we want to separate the blue squares from the red circles.

- If the ANN is over-trained, it will not generalize correctly
- It is equivalent to overfit data points with a very complex curve
- The simplest model which explains most of the data is usually the best

Occam’s Razor (1300s): “Plurality should not be assumed without necessity”



SECTION 02

Designing ANNs

1. **Architecture of the network**:
   - Connectivity structure
   - Number of input neurons
   - Number of output neurons
   - Number of hidden layers
   - Number of hidden neurons
   - Layer policy

2. **Data preparation**:
   - Pre-processing and filtering
   - Output class representation

3. **Structure of the neurons**:
   - Selection of the activation function
   - Learning rules

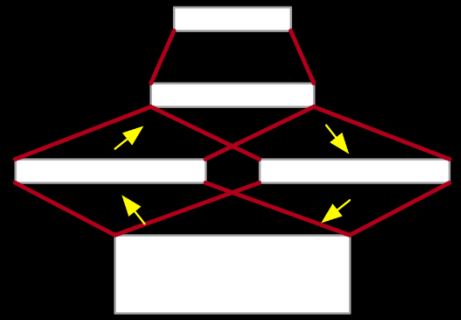
4. **Weights**:
   - Range of initialization
   - Pruning vs non-pruning

SECTION 02

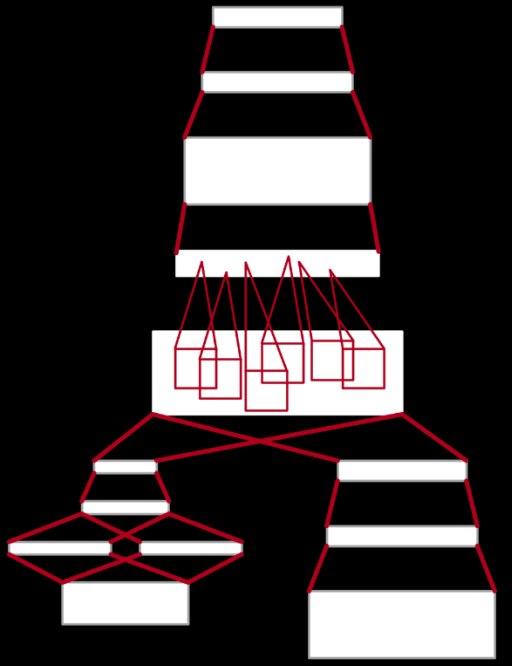
Architecture of the network

The selection of the **connectivity structure** or **topology** depends on the concept of model or hypothesis space one is training the network with:

FEED-FORWARD NETWORKS

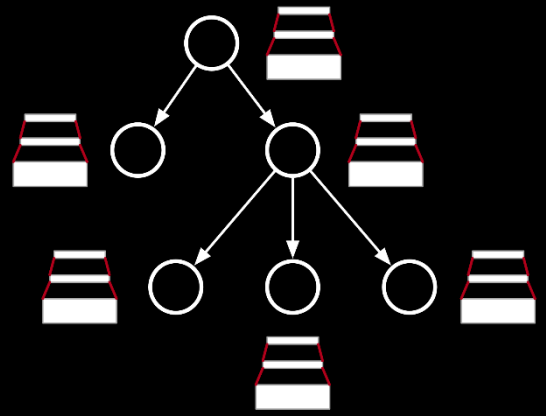


RECURRENT NETWORKS



DEEP NETWORKS

SELF-ORGANIZING MAPS (SOM)



GRAPH NETWORKS

The **number of input neurons** depends on the number of features used during training and production:

- **Too many inputs**: the ANN becomes too big and it won't train well. This case is normally handled:
  - Applying dimensionality reduction algorithms like **Principal Component Analysis** (**PCA**) or **Independent Component Analysis** (**ICA**) to the data.
  - Remove redundant and/or correlating attributes
  - Combine attributes via sum, multiplication, difference, etc.

- **Too few inputs**: the ANN won't be able to generalize well enough because unable to distinguish between different examples. This case is normally handled increasing the number of input features to the ANN.

Data features are normally represented by different types. Some examples include:

- Category symbols: **banana**, **apple**, **orange**
- Ordinal discrete ranking: **1st**, **2nd**, **3rd**
- Continuous numbers: **0.23**, **-45.2**, **500.43**
- Images: **color channels per pixel**

ANNs trained with a Back-propagation algorithm can accept **only continuous numeric** values. It typically depends on the activation function, so the range can be between 0 and +1, or -1 and +1.

CATEGORY SYMBOLS and DISCRETE RANKING

- **Encoding**: the set has 3 categories: 2 bits are enough, 2 input (banana = **01**, apple = **10**, orange = **11**)
- **One-hot**: the set has 3 categories: 3 bits are required, 3 neurons (banana = **100**, apple = **010**, orange = **001**)

PIXEL COLOR CHANNELS

- The single-color channel values can be normalized. Each channel would correspond to an input neuron

The number of output neurons strictly depends on the problem one is training the ANN with:

- Binary output: the layer neurons produce only binary values (i.e. 1s and 0s)
- Degree of certainty: the output neurons produce scalar values of certainty or probability of correct answer

PROBLEM Caution must be used when representing categories and discrete ranking. They must be converted in a binary form

Theoretically, algorithms like Back-propagation can function on an unlimited number of hidden layers.

However, two problems force it to a limited number:

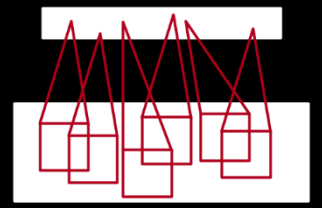
- The **gradient-fading problem**: the error signal becomes too small to be used in layers closer to the input.
- Too many **hidden layers** tend to over-train and not to generalize. It is always better to start with a lower number of hidden neurons and then increase it in a latter training session if learning does not progress.

Usually **one hidden layer** is enough to handle relatively complex functions.

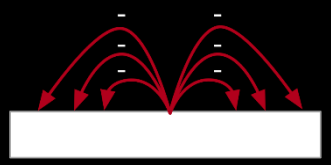
A **connection policy** is the connection behavior that the single neurons or group of neurons express in the network:

- **Fully connected**: a neuron is connected to all the neurons of the previous layer.

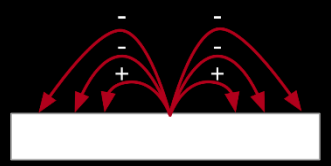
- **Convolution**: a neuron is connected only to a small group of neurons of the previous layer.



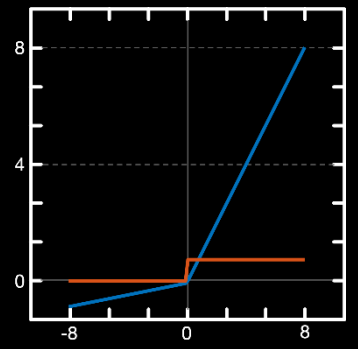
- **Winner-takes-all**: the neuron that produces the highest output suppresses all the others in the same layer.



- **Mexican-hat**: the neuron is connected with positive weights to a small group of neighbors. It is also connected with negative weights to a ring of neighbors outside the immediate neighbors.



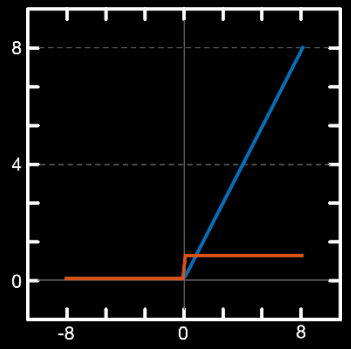
LEAKY RELU



$$O(p) = \begin{cases} p & \text{for} \quad p \geq 0 \\ 0.01p & \text{for} \quad p < 0 \end{cases}$$

$$O'(p) = \begin{cases} 1 & \text{for} \quad p \geq 0 \\ 0.01 & \text{for} \quad p < 0 \end{cases}$$

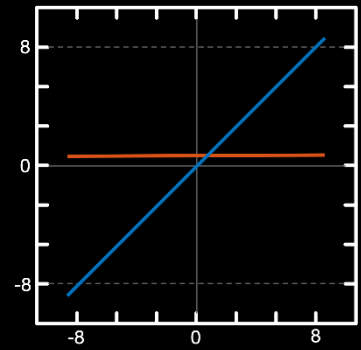
RELU



$$O(p) = \begin{cases} p & \text{for} \quad p \geq 0 \\ 0 & \text{for} \quad p < 0 \end{cases}$$

$$O'(p) = \begin{cases} 1 & \text{for} \quad p \geq 0 \\ 0 & \text{for} \quad p < 0 \end{cases}$$

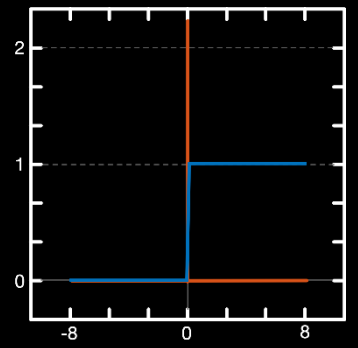
LINEAR / IDENTITY



$$O(p) = p$$

$$O'(p) = 1$$

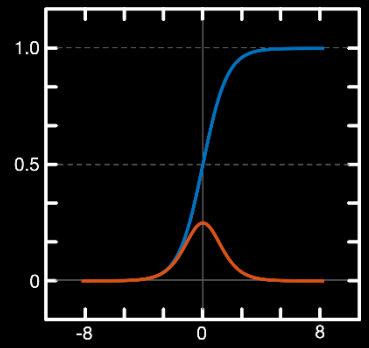
STEP



$$O(p) = \begin{cases} 1 & \text{for} \quad p \geq 0 \\ 0 & \text{for} \quad p < 0 \end{cases}$$

$$O'(p) = \begin{cases} 0 & \text{for} \quad p \neq 0 \\ ? & \text{for} \quad p = 0 \end{cases}$$

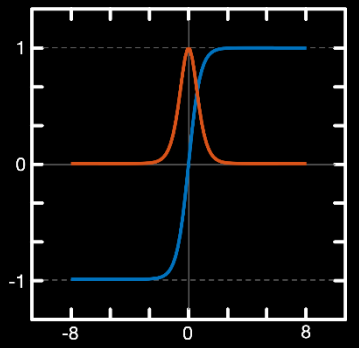
SIGMOID / LOGISTIC



$$O(p) = \frac{1}{1 + e^{-p}}$$

$$O'(p) = O(p)[1 - O(p)]$$

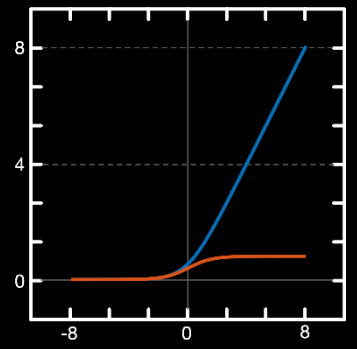
TANH



$$O(p) = \tanh(p)$$

$$O'(p) = 1 - O(p)^2$$

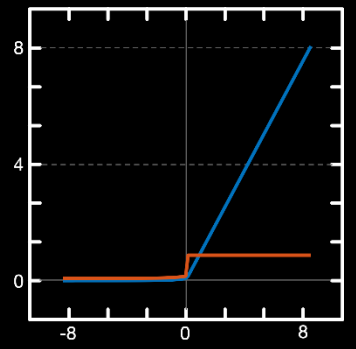
SOFTPLUS



$$O(p) = \ln(1 + e^p)$$

$$O'(p) = \frac{1}{1 + e^{-p}}$$

ELU



$$O(p) = \begin{cases} p & \text{for} \quad p \geq 0 \\ \alpha(e^p - 1) & \text{for} \quad p < 0 \end{cases}$$

$$O'(p) = \begin{cases} 1 & \text{for} \quad p \geq 0 \\ O(p) + \alpha & \text{for} \quad p < 0 \end{cases}$$

Selection of the **learning rule** depends on several factors:

- The layer policy
- The nature of the problem one is training to ANN with

In literature there are many learning rules:

- Hebb learning rule
- Delta learning rule (or Widrow-Hoff learning rule)
- Momentum descent
- Many more…

SECTION 03

Data preparation

- The **quality of results** relates directly to quality of the data
- 50%-70% of ANN development time is usually spent on data preparation
- The three steps of data preparation:
  1. Consolidation and cleaning
  2. Selection and preprocessing
  3. Transformation and encoding

Data must be prepared and cleaned before using it for training:

- **Determine** the appropriate input attributes
- **Consolidate** data into a working database
- **Eliminate** or **estimate** missing values
- **Remove** outliers (obvious exceptions)

- **De-correlate** example attributes via a value normalization method:
  - **Euclidean**: $y_e = \frac{x_e}{\sqrt{\sum_e x_e^2}}$
  - **Percentage**: $y_e = \frac{x_e}{\sum_e x_e}$
  - **Variance**: $y_e = \frac{x_e - \mu}{\sigma^2}$
- **Scale** values using:
  - A linear transformation if data is uniformly distributed
  - A non-linear (log, power) transformation if data has a skewed distribution

SECTION 04

Training ANNs

How do you ensure that a network has been well trained? One tests the **generalization accuracy** on new examples.

In order to do that, one needs:

- **Establish** a maximum acceptable error rate
- **Train** the network with the **training set**
- **Validate** the network with a **validation test** during training
- At the end of the training phase, **test** the network against a separate **test set** (also called **production set**)

Measuring the **performance** of the model is done via several metrics:

- **Accuracy**: $$Accuracy = \frac{Number\ of\ correct\ classifications}{Total\ number\ of\ test\ cases}$$

- **Confusion matrix**:

		ACTUAL OUTPUT: CLASS 1	ACTUAL OUTPUT: CLASS 2	ACTUAL OUTPUT: CLASS 3
PREDICTED OUTPUT	CLASS 1	95%	4%	1%
	CLASS 2	0%	97%	3%
	CLASS 3	0%	0%	100%

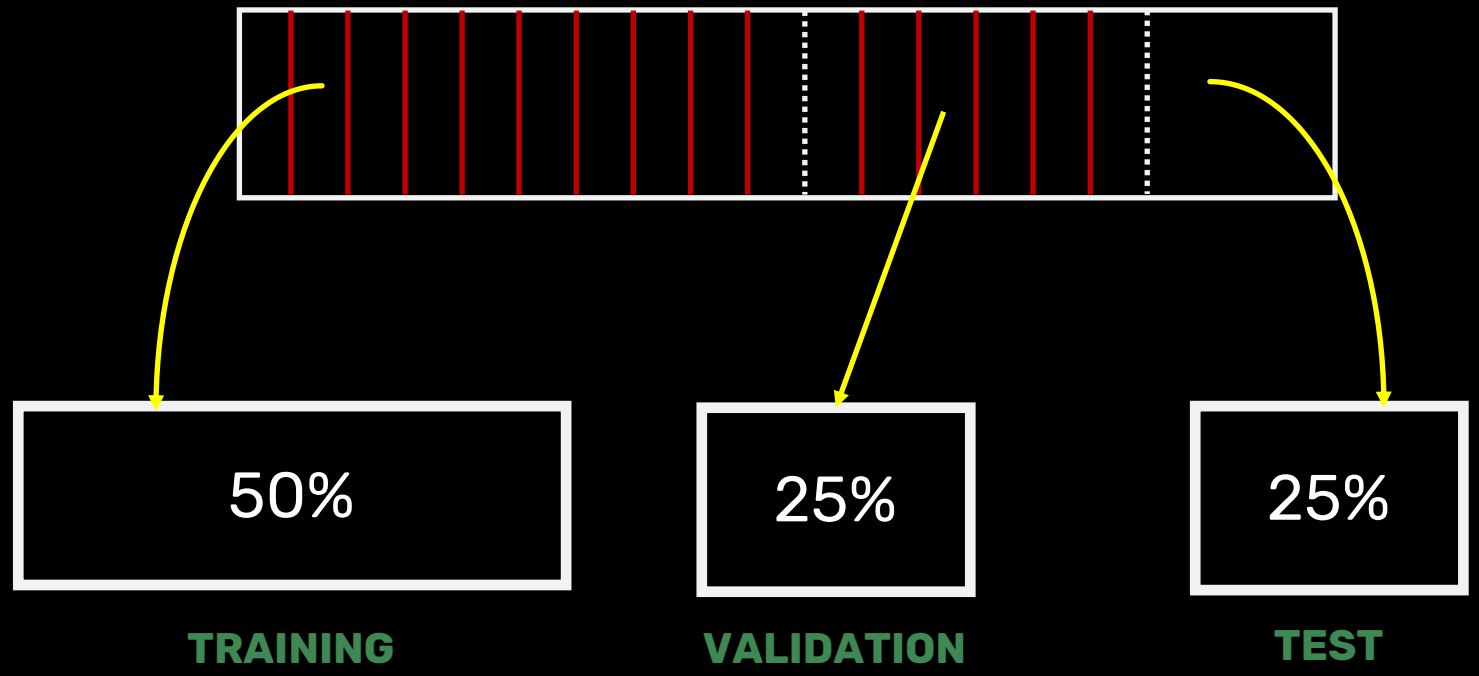
CLASS 1 examples are correctly classified as **CLASS 1** in 95% of the cases; 4% are erroneously classified as **CLASS 2**; 1% are erroneously classified as **CLASS 3**.

- **RMSE / MSE**: $$RMSE = \left[\frac{1}{N}\sum_{c=1}^{N}(o_p - o_a)^2\right]^{\frac{1}{2}} \qquad MSE = \frac{1}{N}\sum_{c=1}^{N}(o_p - o_a)^2$$

- **Effectiveness of training**:
  - ANNs start with random weights and improves them
  - If improvement stops, we stop the training process
  - There is no guarantee that we found the best set of weights
- **Convergence**:
  - Back-propagation algorithms use gradient decent
  - Naïve implementations can:
    - **Over-correct** weights
    - **Under-correct** weights
  - In either case, convergence can be poor
  - It can be improved with Simulated annealing, genetic algorithms, more sophisticated gradient descents, etc.

When the amount of available data is **large**, the proportion of data for the training, validation and test sets is usually **50:25:25**:

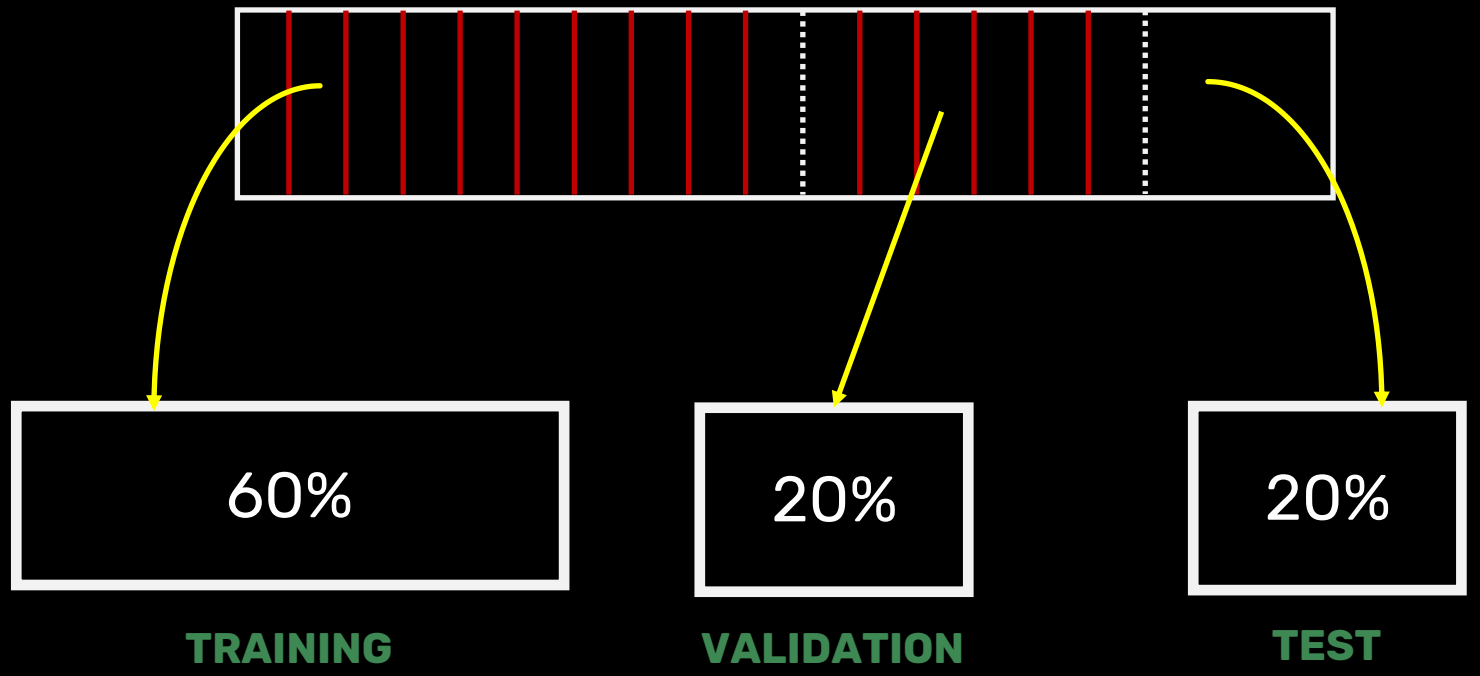
1. The training and validation set and test set are **selected** at random
2. The examples in the sets **must** uniformly represent all the classes of data



When the amount of available data is **small**, the proportion of data for the training, validation and test sets is usually **60:20:20**.

If the amount of available data is very small, a **leave-some-out, multi-fold cross-validation** technique can be applied.

1. The data is **divided** in chunks
2. The training and validation sets are **selected** among the chunks
3. The network is **trained** multiple times with **different** training and validation sets
4. The best network is **selected**



		TYPICAL	RANGE
Learning rate	η	**0.1**	**0.010 – 0.999**
Momentum	α	**0.8**	**0.100 – 0.900**
Weight	w_{ij}	**0.1**	**0.001 – 0.500**

Network weight initialization:

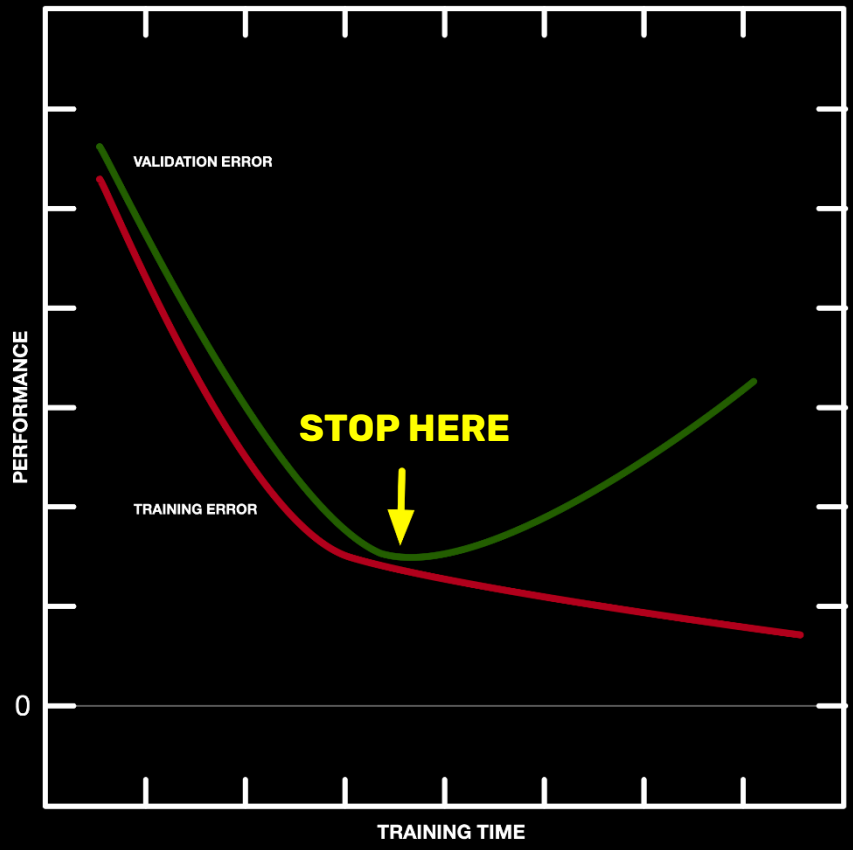
- **Smaller weight values** for neurons with **many** incoming connections
- Generate random initial values within some range that should be approximately:

$$\pm \frac{1}{\#weights}$$

An ANN is **over-trained** when it works too well on a training set, but it works poorly on a test set.

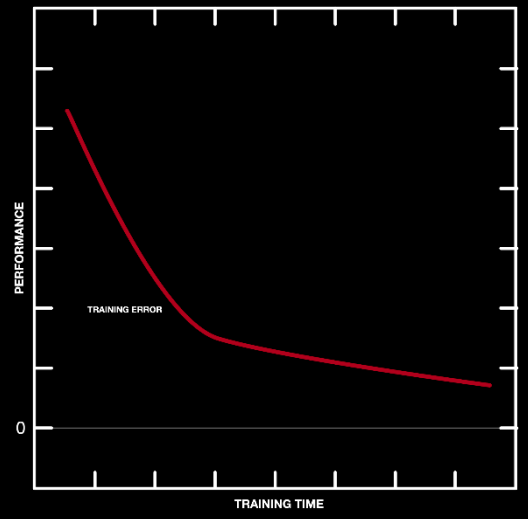
The general idea is to **stop** the training when the validation error increases again.

Sometimes, the training **continues** a bit more to see if the first increase is due to a local minimum.



The way the **total error** decreases with time might indicate few things that are happening inside the ANN:

- The training is proceeding well



- The network has possibly found a local minimum:
  - **Increase** the learning rate
  - **Increase** the momentum parameter

- The network might not be able to learn the function from the data:
  - **Reduce** the learning rate
  - **Reduce** the momentum parameter

READINGS

Chapter 18.7

QUESTIONS ?

Tufts
UNIVERSITY

ARTIFICIAL INTELLIGENCE

COMP 131

FABRIZIO SANTINI

VERSION 4.0