ISLab

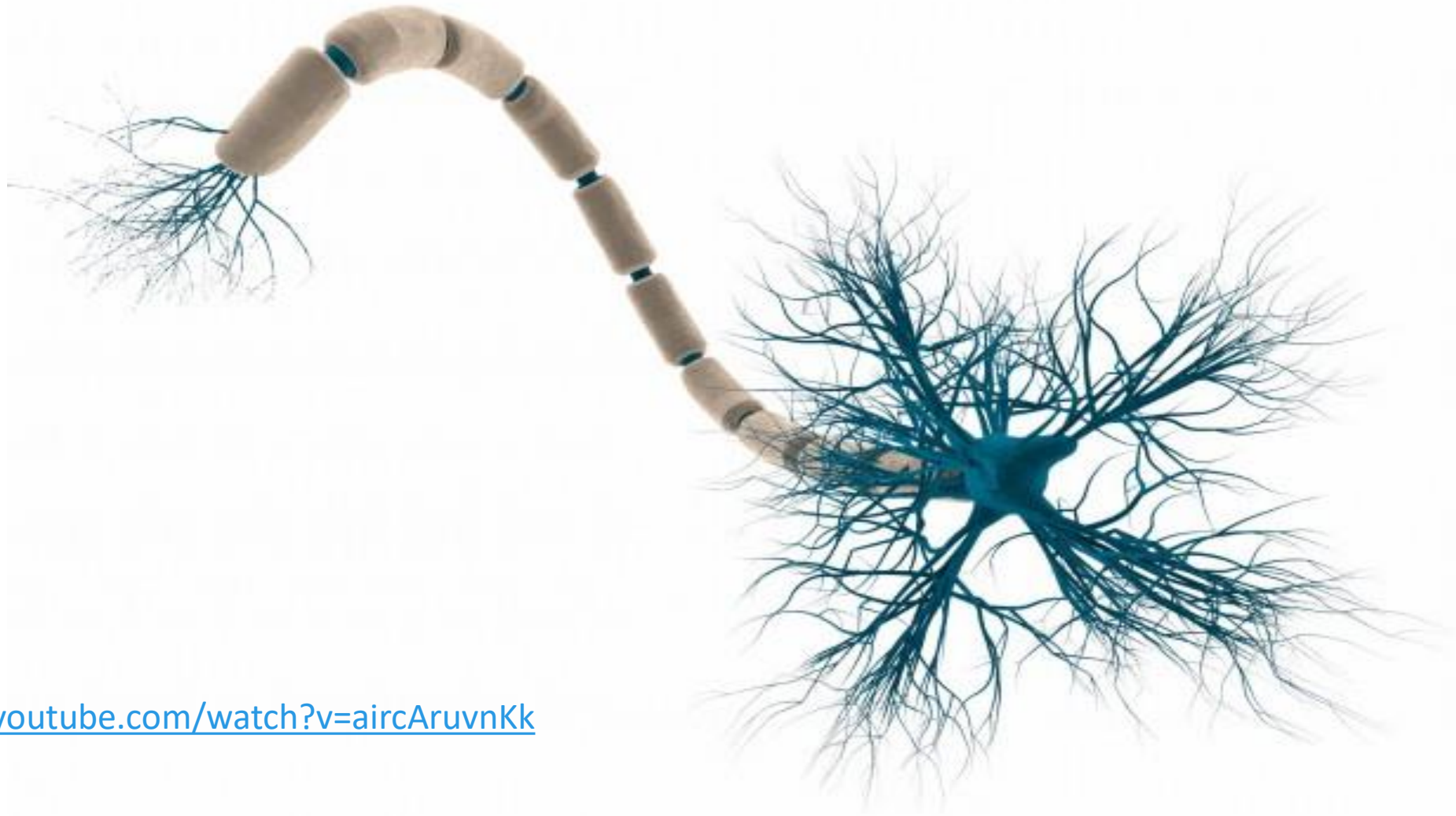
Universidade do Minho
Escola de Engenharia
Departamento de Informática

Neural Networks

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Perfil SI :: Computação Natural

ISLab
Synthetic Intelligence Lab

Inspiration

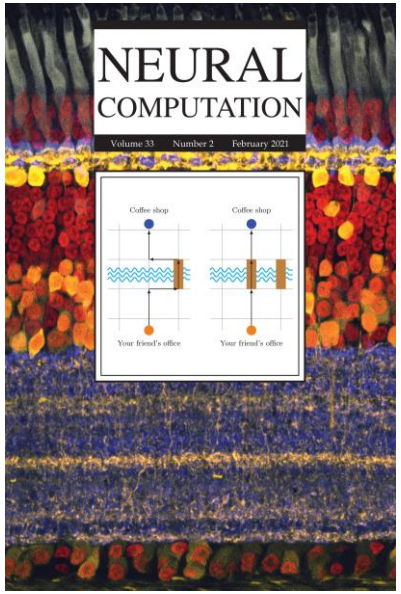


https://www.youtube.com/watch?v=aircAruvnKk

Neural Computation

Neural Computation

Neural computation is the information processing performed by networks of neurons.



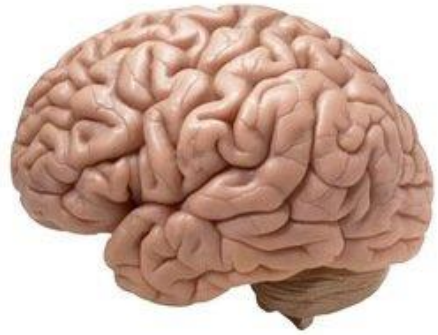
Source: Neural Computation
https://www.mitpressjournals.org/loi/neco

ISLab
Synthetic Intelligence Lab

Neural Networks

- An analogy to biological neural systems;
- Understand biological systems through computational shaping;
- Massive parallelism allows for computational efficiency;
- Distributed (neural) representations vs local representation;
- Intelligent behavior - an emergent property of a large number of simple units;
- "**The human brain is a highly complex, non-linear, and parallel computer, responsible for information-processing abilities that humans possess**".

Image source: https://www.roboticshell.com/Tutorial/NeuralNetwork



Source: https://www.freeimages.com/pt/photo/no-description-1301831

ISLab
Synthetic Intelligence Lab

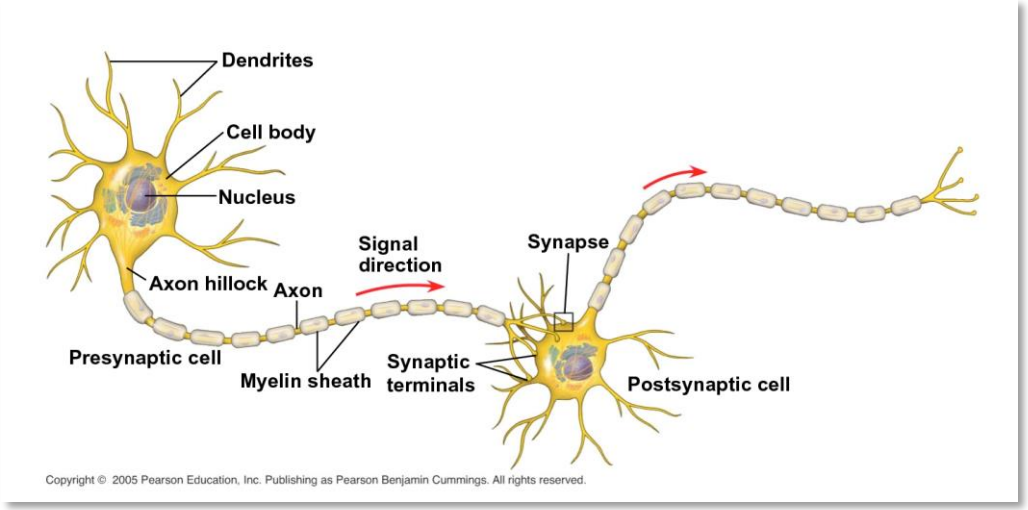
Common Types of Applications

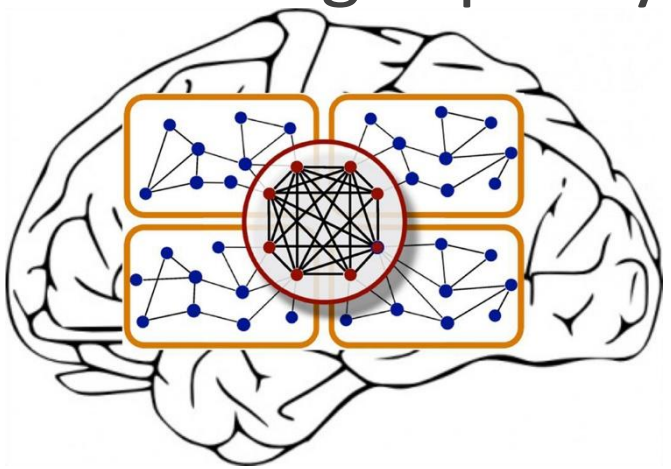
- Associative memory;
- Classification/Diagnosis;
- Pattern recognition;
- Regression;
- Control;
- Optimization;
- Data filtering / compression.

ISLab
Synthetic Intelligence Lab

Artificial Neural Network

- An **Artificial neural network** (ANN) is a connection-based computational system for problem solving.
- An ANN is designed based on a simplified model of human central nervous system.
- An ANN is defined by an interconnected structure of computational units, called neurons, with learning capacity.





ISLab
Synthetic Intelligence Lab

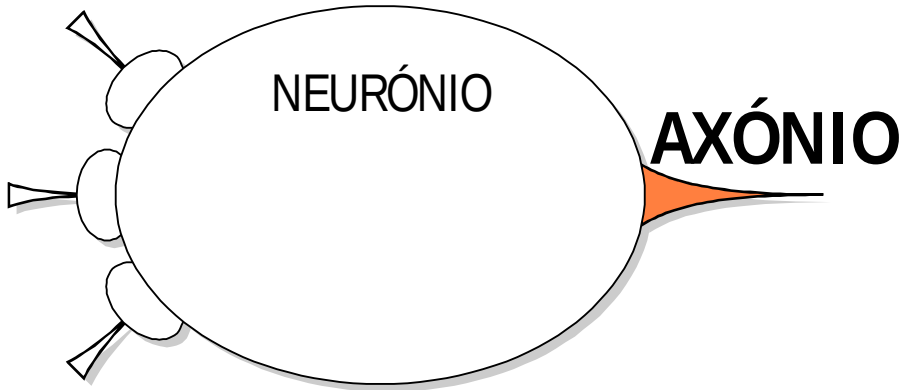
Neuron

- **Computational unit** of ANN composition.
- **Identified** by its **position** in the network.
- Characterized by the **state value**.

ISLab
Synthetic Intelligence Lab

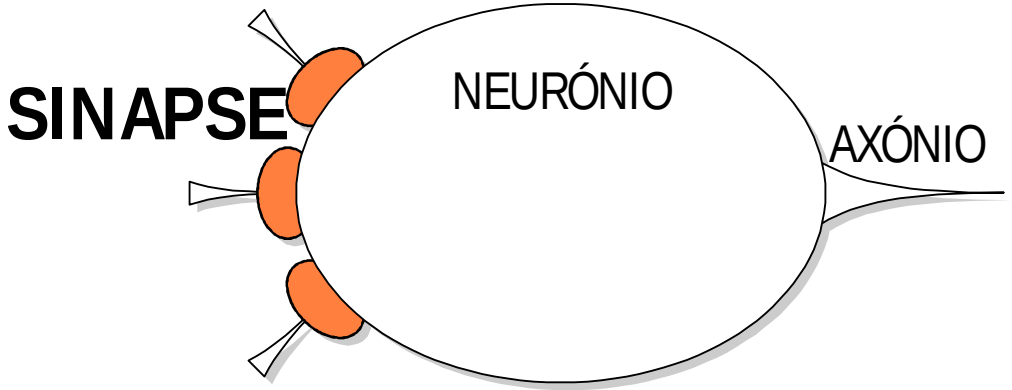
Axon

- **Communication pathway** between neurons.
- It can **connect any neuron,** including itself.
- Connections may **differ over time**.
- Information circulates in **one direction**.



Synapse

- **Connection point** between axons and neurons.
- The **synapse value** determines the **weight** (importance) of the signal to enter the neuron: excitatory, inhibitory or null.
- The **variation in time determines the learning** of ANN.



ISLab
Synthetic Intelligence Lab

Activation

- The activation value is represented by a **single value**.
- The activation value **changes with time**.
- The range of values differs with the model adopted (it is usually dependent on inputs and some memory effect).

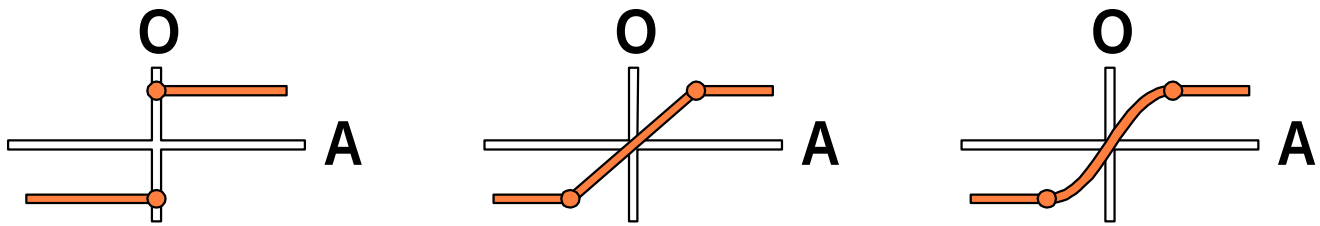
ISLab
Synthetic Intelligence Lab

Transfer

- The transfer value of a neuron determines the **value** that is **placed at the exit** (transferred through the axon).
- It is calculated as a function of the activation value (possibly with some memory effect).

Tasks of a neuron

- Calculation of the output value (output = Oi), function of the activation value: $O_i = f_T (A_i)$



Activation value (A_j).

Varies over time with its own value and that of other entries (w_i ; I):

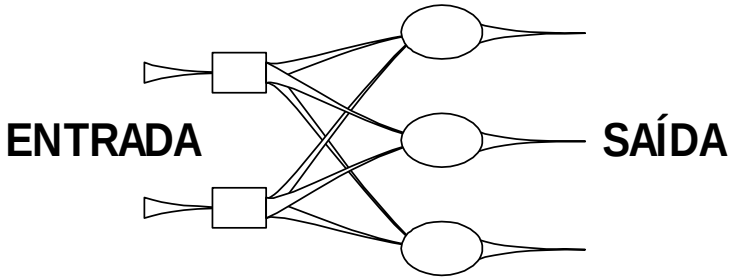
$$A_j = \mathcal{F}(A_{j-1}; I_j; \sum w_{i,j} \times O_i)$$

Learning: weight modification rules (w_i).

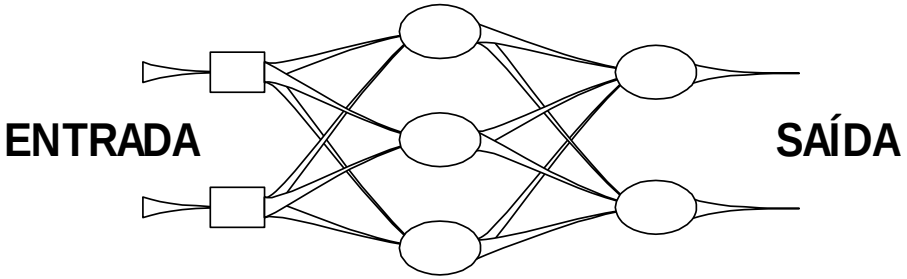
ISLab
Synthetic Intelligence Lab

Artificial Neural Network

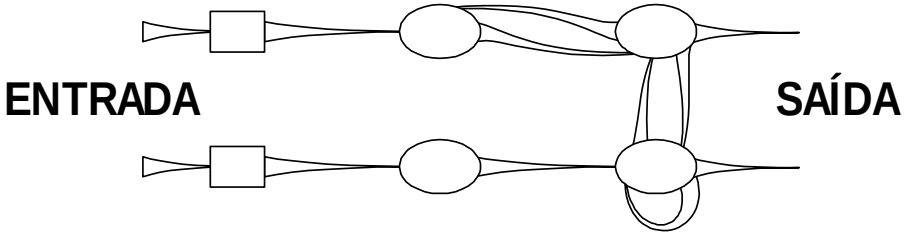
Feedforward, Single layer:



Feedforward, multi-layer:



Recurrent:

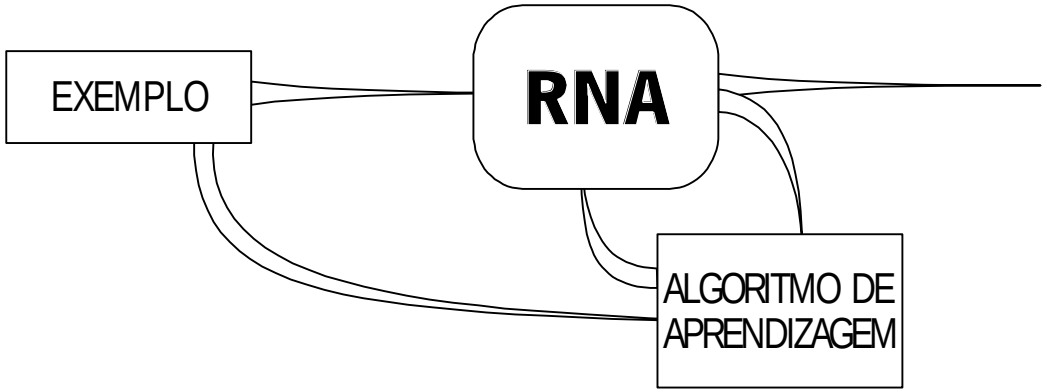


ISLab
Synthetic Intelligence Lab

Learning paradigms

Without supervision

(e.g., when two adjacent neurons have variations in activation in the same direction, then the weight of the bond should be progressively increased.)



ISLab
Synthetic Intelligence Lab

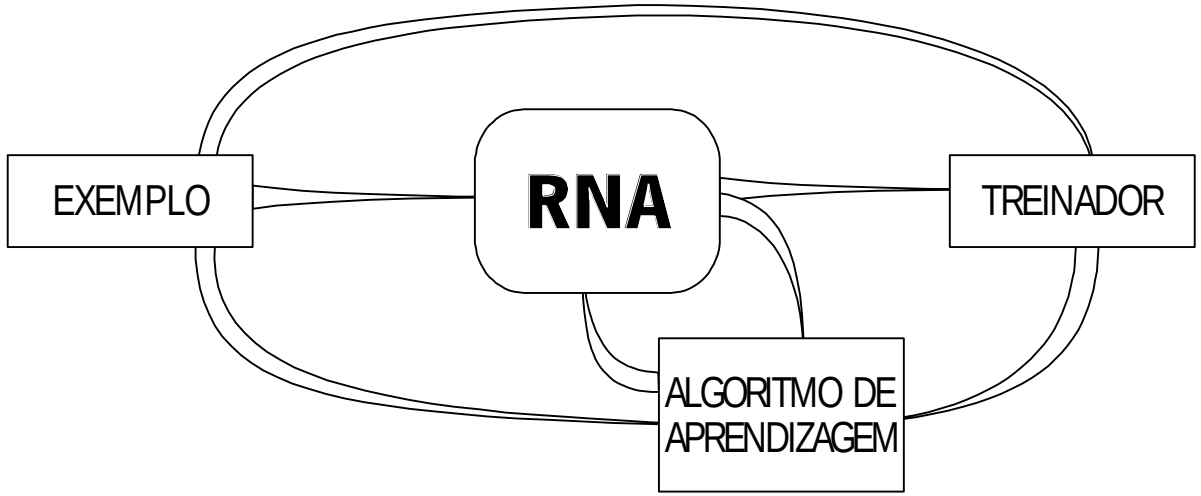
Learning paradigms

With supervision:

(e.g., adjustments to the connection weights are made in order to minimize the error produced by the ANN results.)

Reinforcement:

the example contains only an indication of the correction of the result.



Training rules (learning)

The training of an ANN corresponds to the application of learning rules, in order to vary the connection weights (synapses);

Examples of more common learning rules are:

- Hebbian;
- Competitive;
- Stochastic;
- Memory-based;
- Descending gradient.

Specification

Number of neurons:
- in the input layer;
- in the output layer;
- in the intermediate layers;

Levels (or layers) of the RNA;

Links between neurons;

Connection topology;

Scheme for assigning and updating weights;

Functions:
- transfer;
- activation;
- of learning;

Training Methods.

ISLab
Synthetic Intelligence Lab

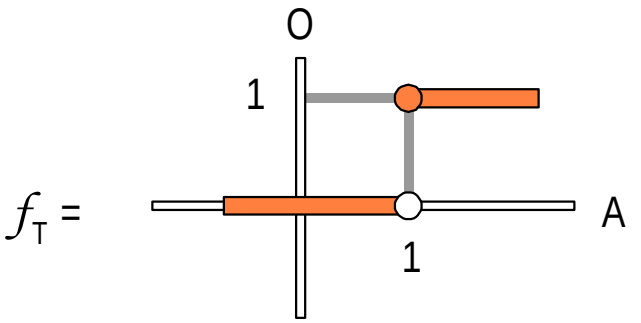
XOR Problem

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Activation function:

$\mathcal{F}_A = \sum$ inputs x weights

Transfer function:

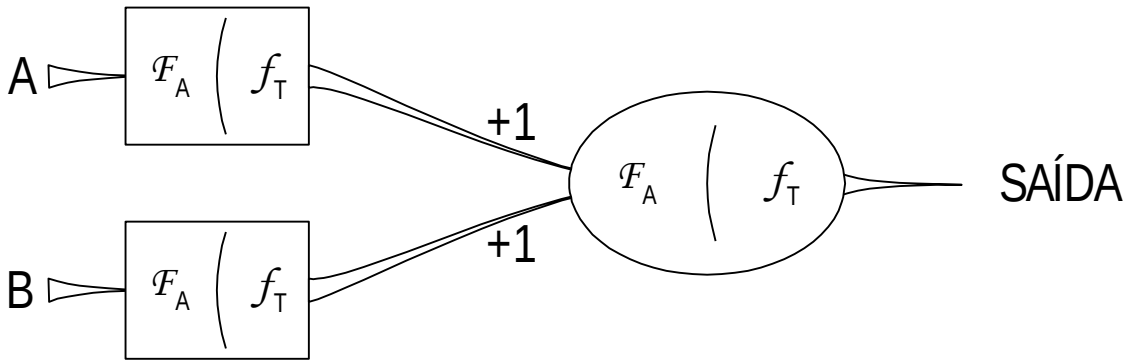


ISLab
Synthetic Intelligence Lab

XOR Problem

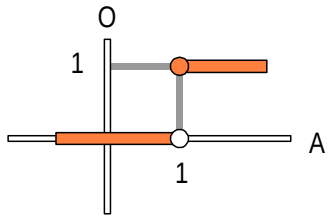
Feedforward RNA, completely bound, with layers 2-1;

Assume the training result given by::



$\mathcal{F}_A = \sum$ inputs x weights

$f_T =$

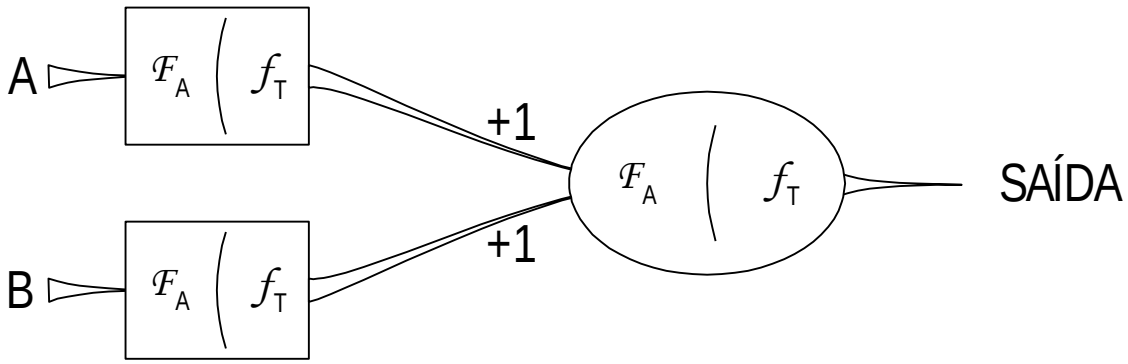


ISLab
Synthetic Intelligence Lab

XOR Problem

Feedforward RNA, completely bound, with layers 2-1;

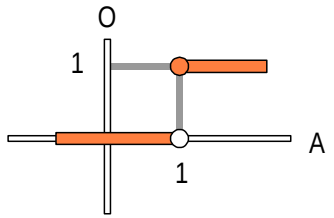
Assume the training result given by:



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	1

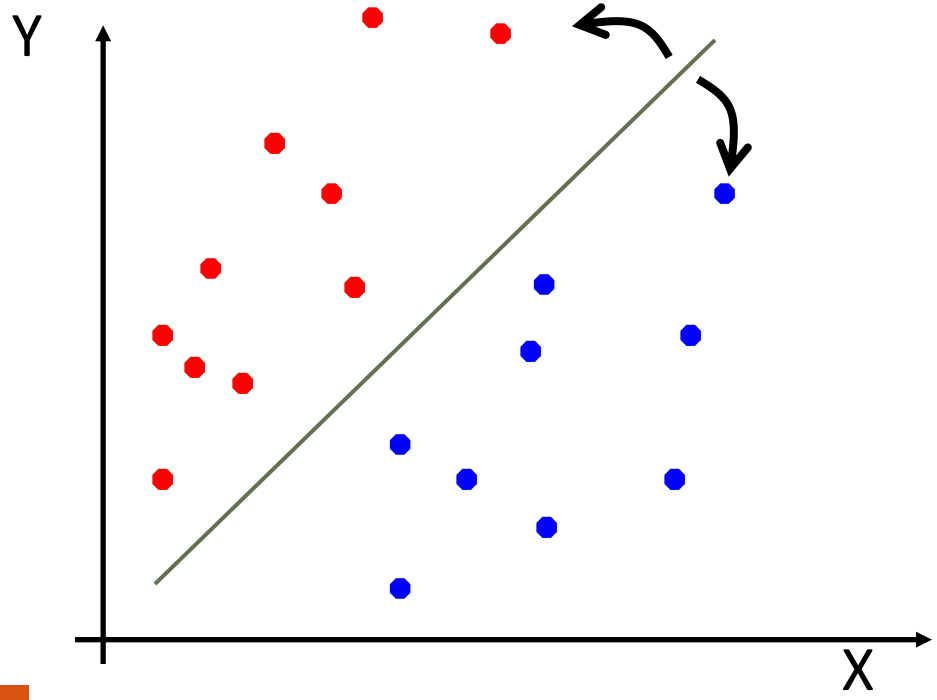
$\mathcal{F}_A = \sum$ imputs x weights

$f_T =$



Perceptron

Since perceptron uses linear threshold function, it is searching for a linear separator that discriminates the classes.



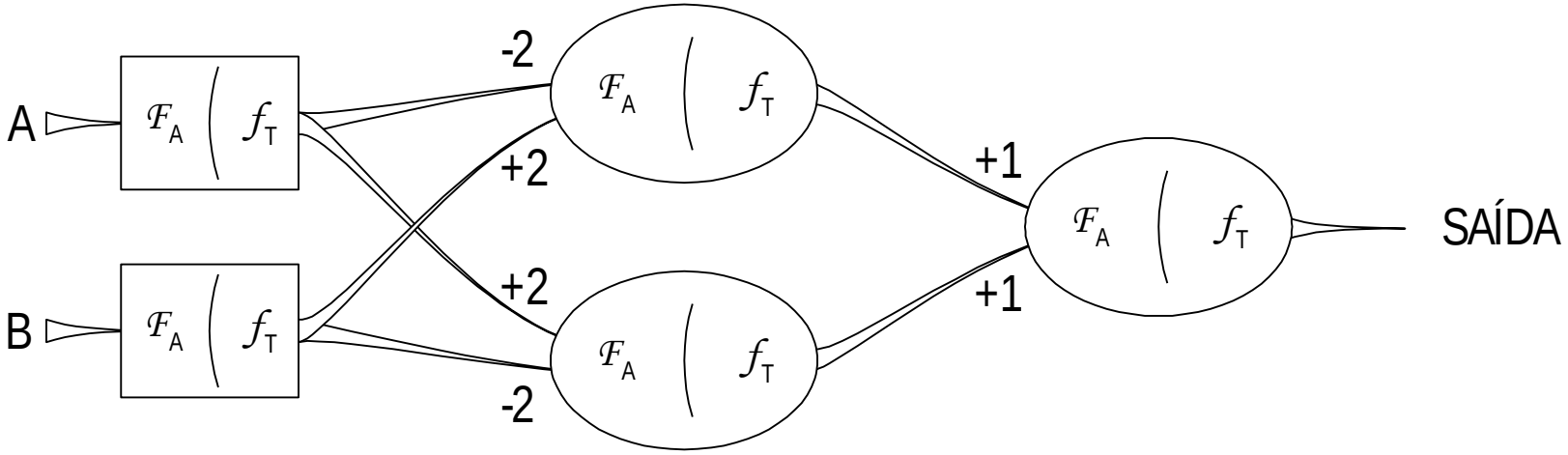
(As we know) Logistic regression is only able to find hyperplanes (basically straight lines) that separate the subspaces of each class.
linear discriminants

Historically the inefficiency of the Perceptron networks to solve this problem caused the “NN winter”

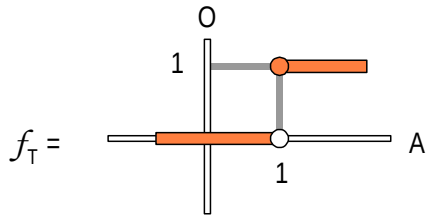
XOR Problem

Feedforward RNA, completely bound, with layers 2-2;

Assume the training result given by:



$\mathcal{F}_A = \sum$ inputs x weights

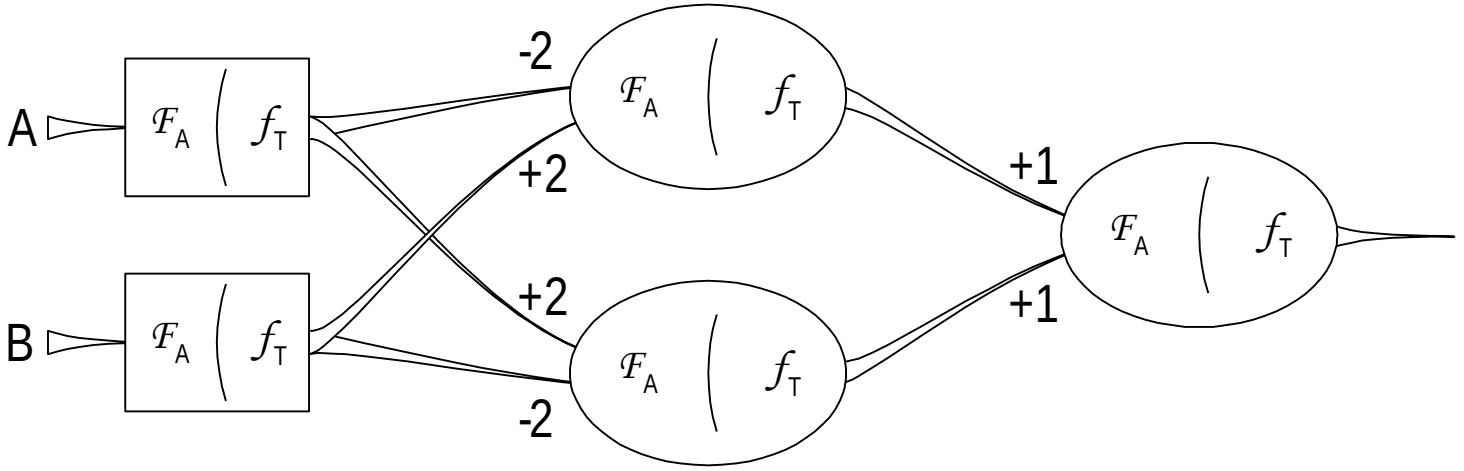


ISLab
Synthetic Intelligence Lab

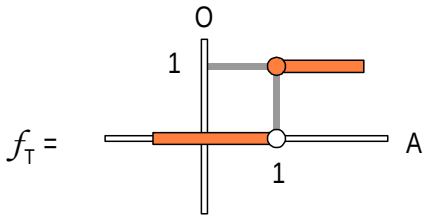
XOR Problem

Feedforward RNA, completely bound, with layers 2-2;

Assume the training result given by:



$\mathcal{F}_A = \sum$ inputs x weights



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

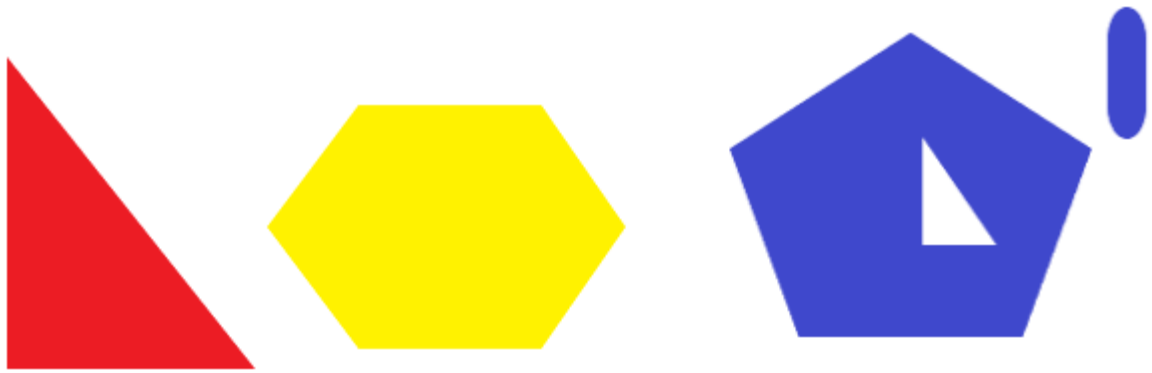
ANN: issues

The number of hidden layers, typically:

- one layer has the ability to approximate any linear decision area;
- two layers approximate any continuous decision area;
- three layers approximate any decision area.

In the hidden layers, the number of neurons:

- A short number might not be enough to model the desired decision area;
- A high value might lead to overfitting;
- Values to test should be between half and double of the number of neurons in the input layer.



ISLab
Synthetic Intelligence Lab

RNA's training

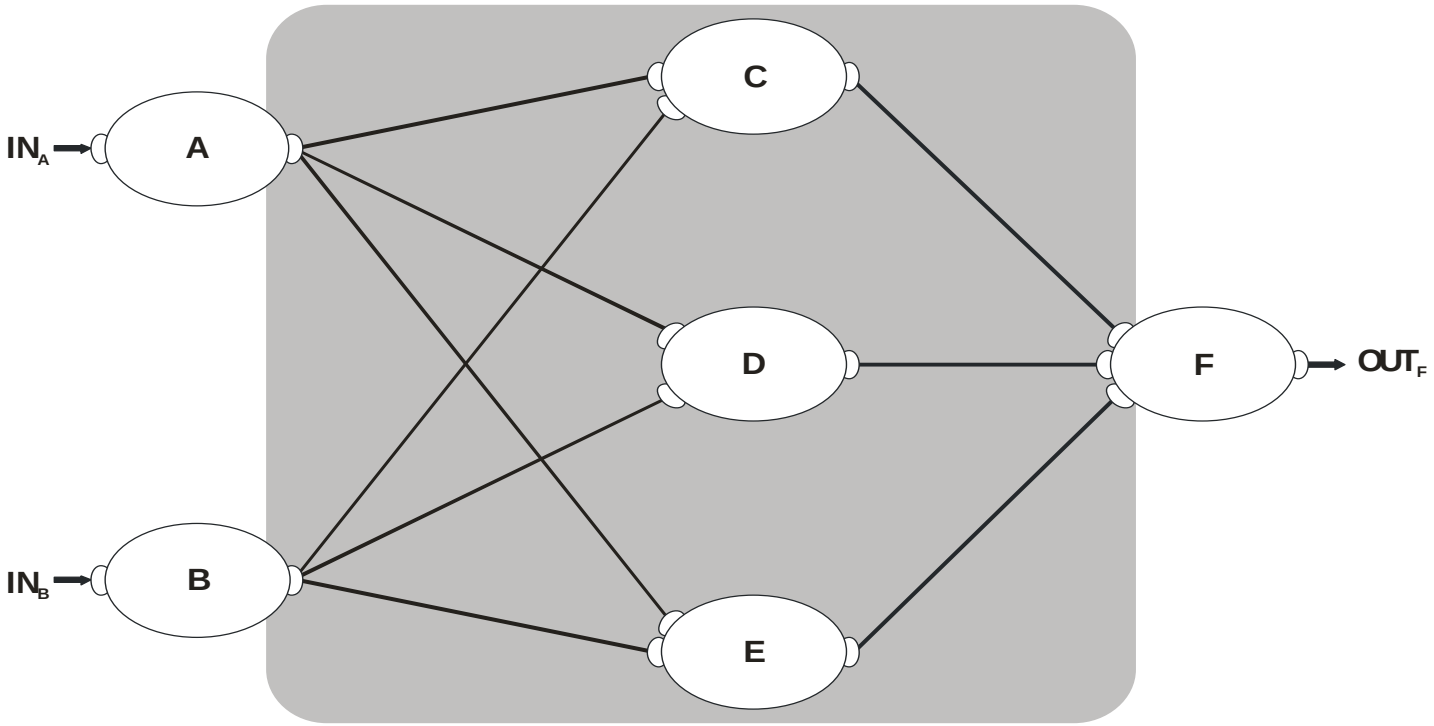
Consider the following Artificial Neuronal Network ...

RNA's training

... structure: 2 neurons at the input and 1 at the output ...

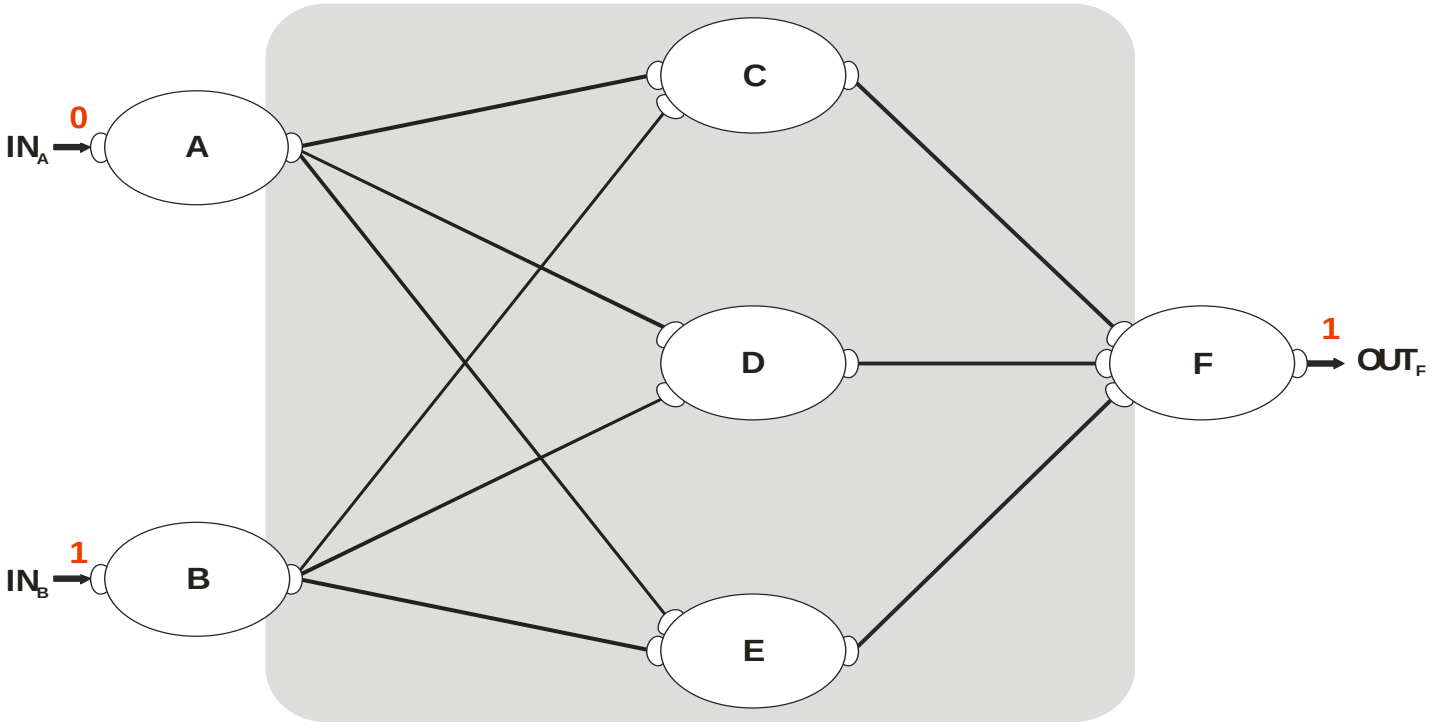
RNA's training

... feedforward, completely connected



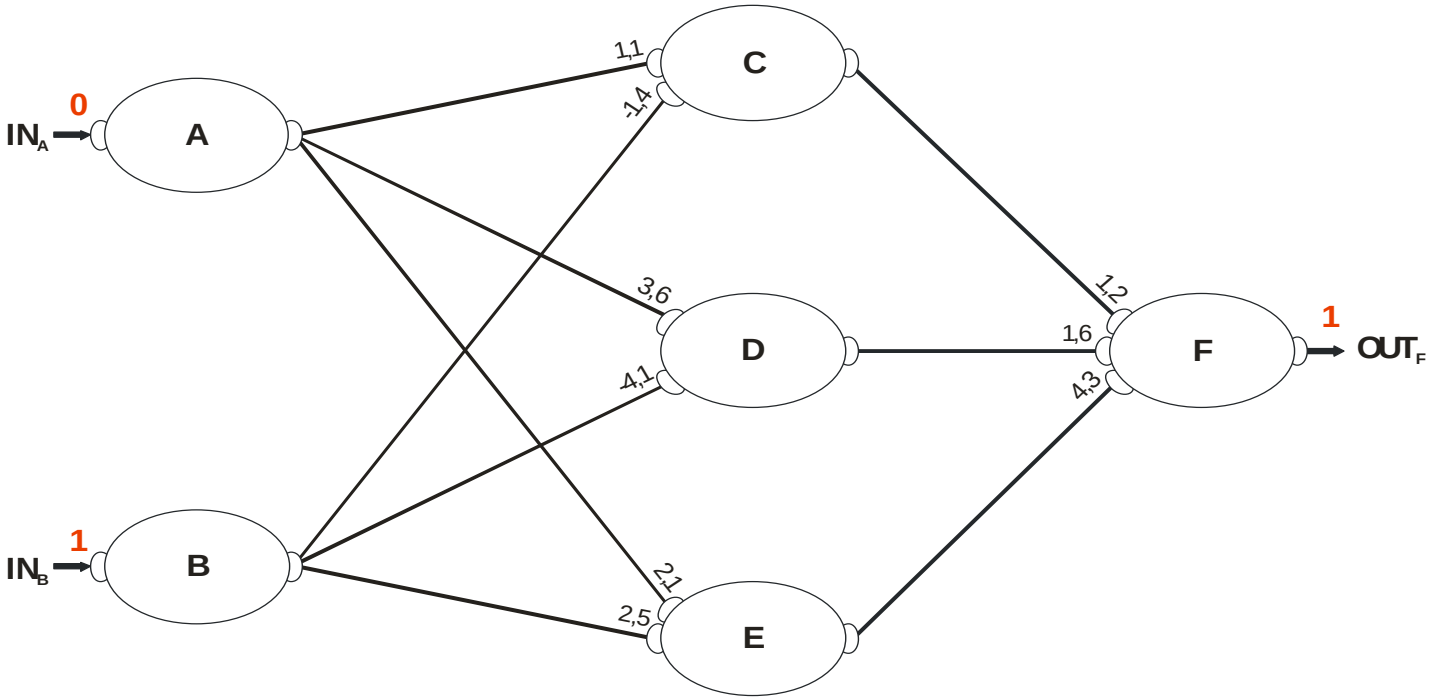
RNA's training

The training examples contain the desired results.



RNA's training

Random assignment of weights to synapses.

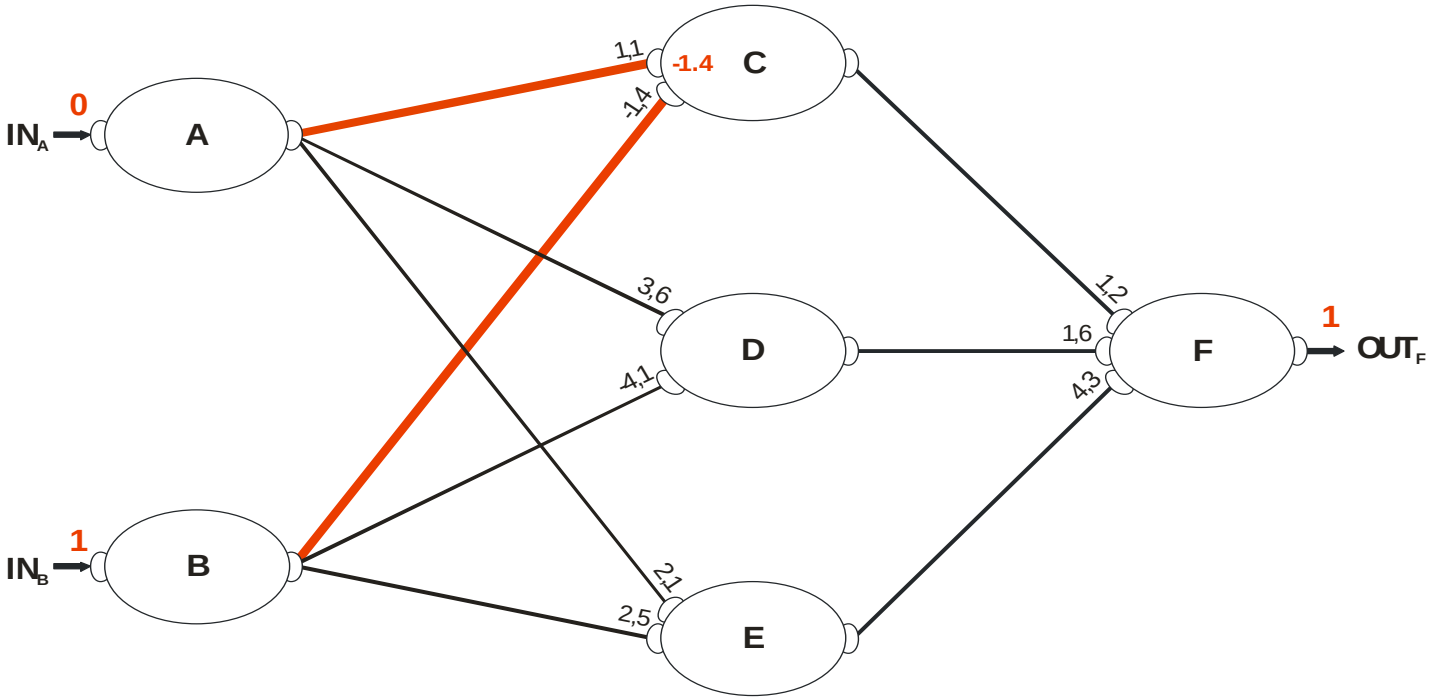


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

Computation of activation value ...

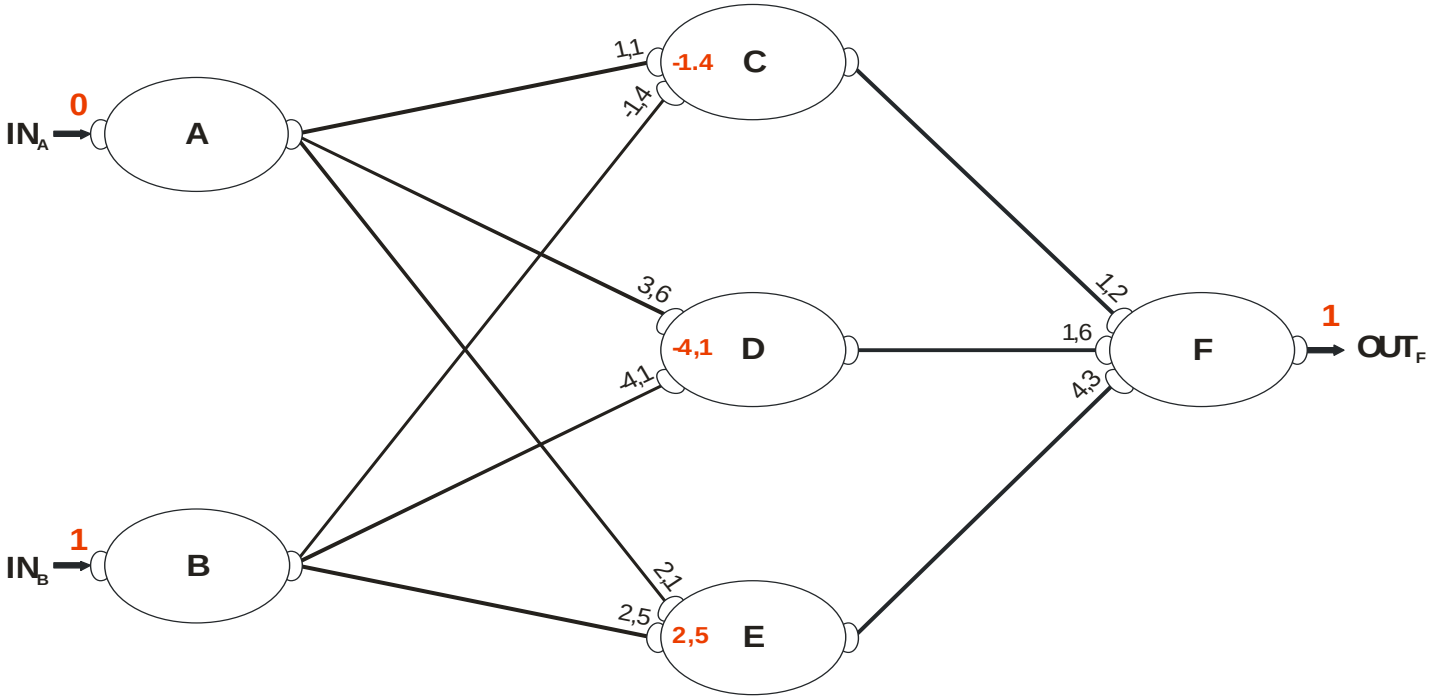


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

... for all neurons in the middle layer.

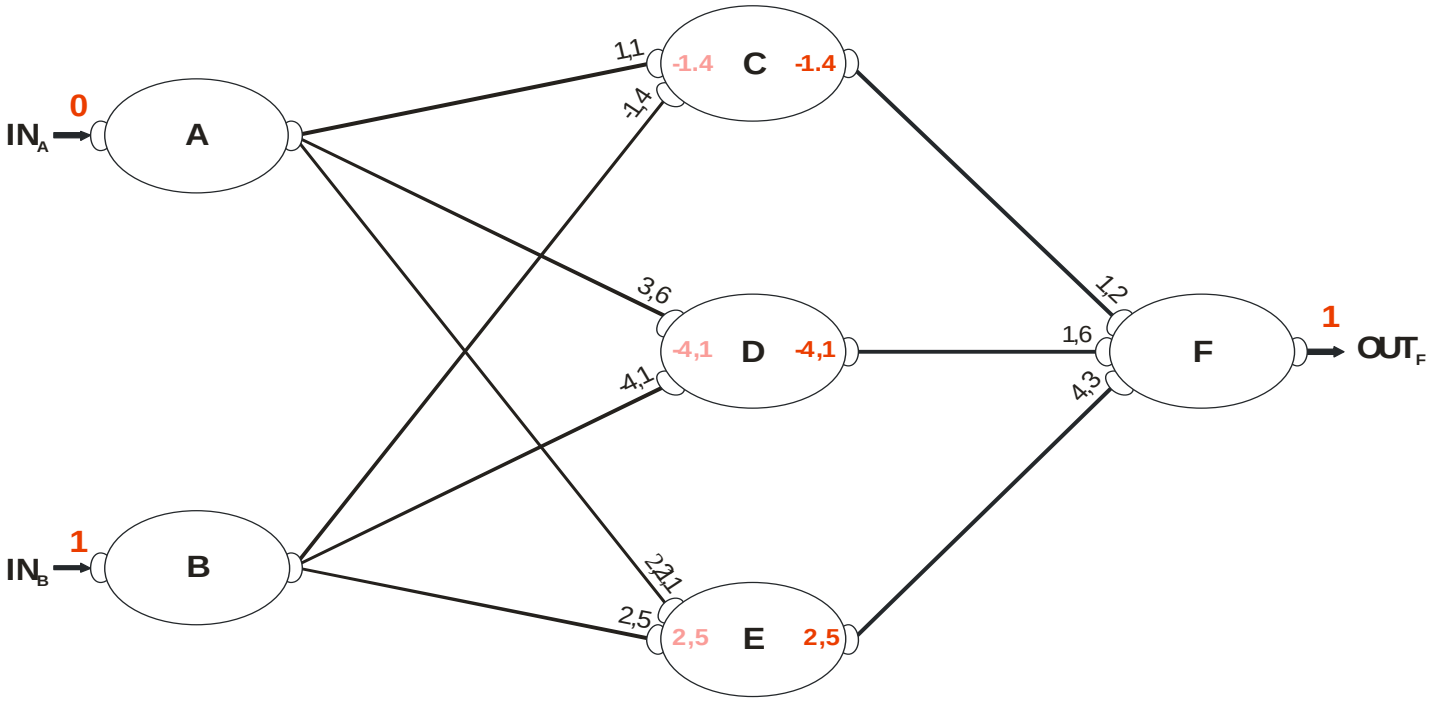


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

Estimate of the transfer amount.

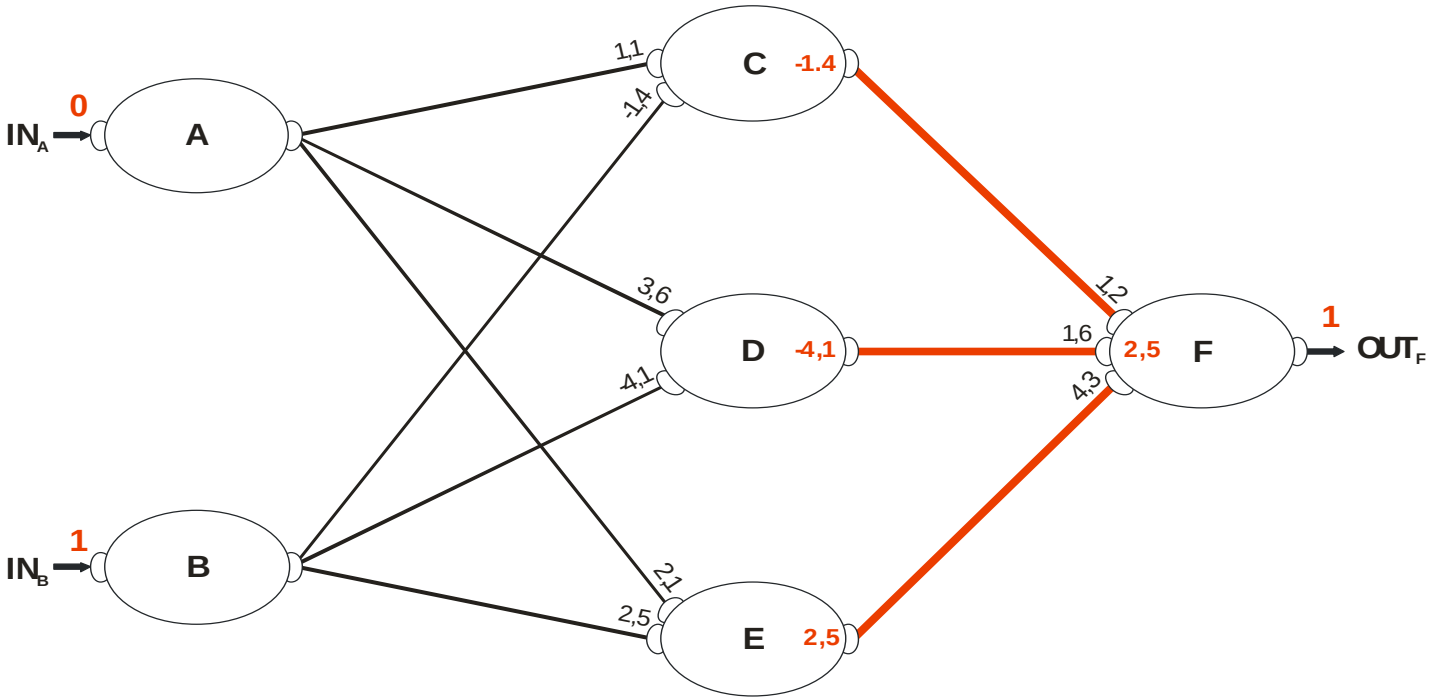


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

Activation value in the output layer ...

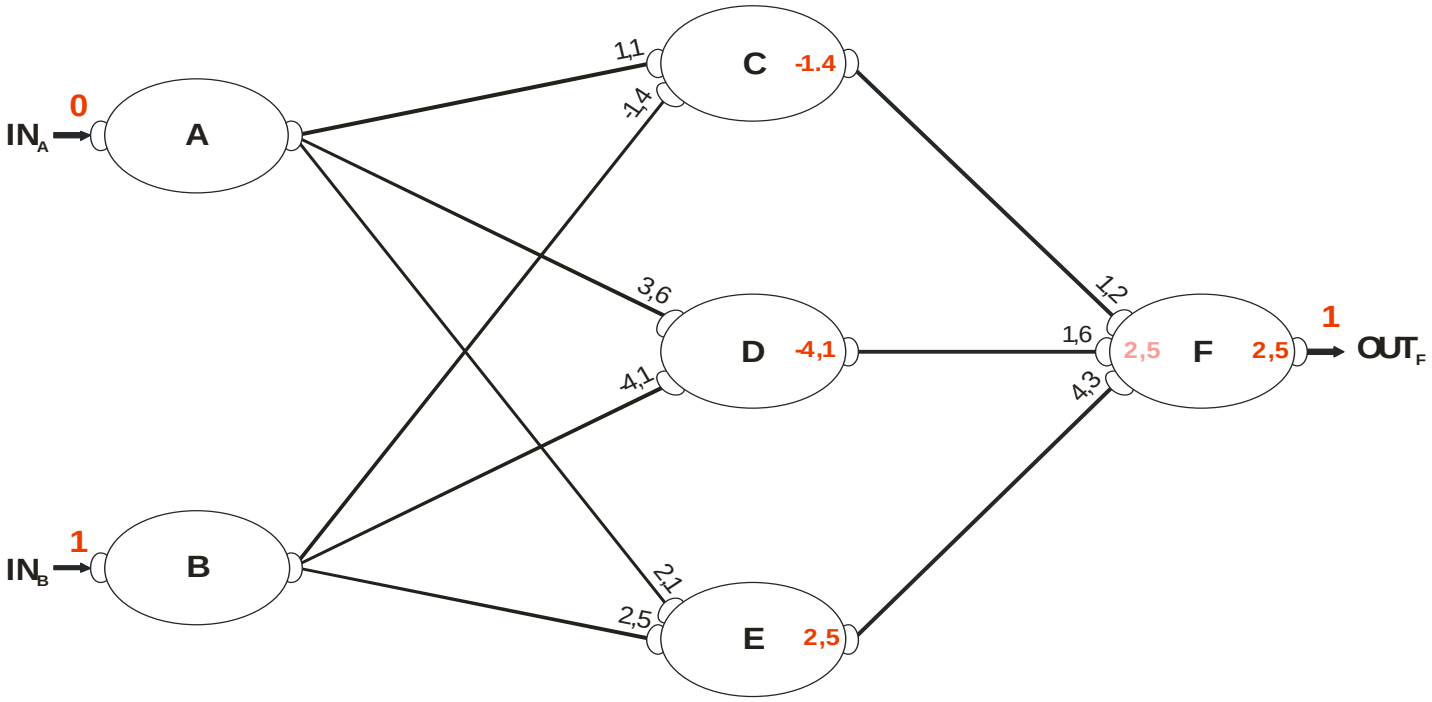


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

... and its transfer value.

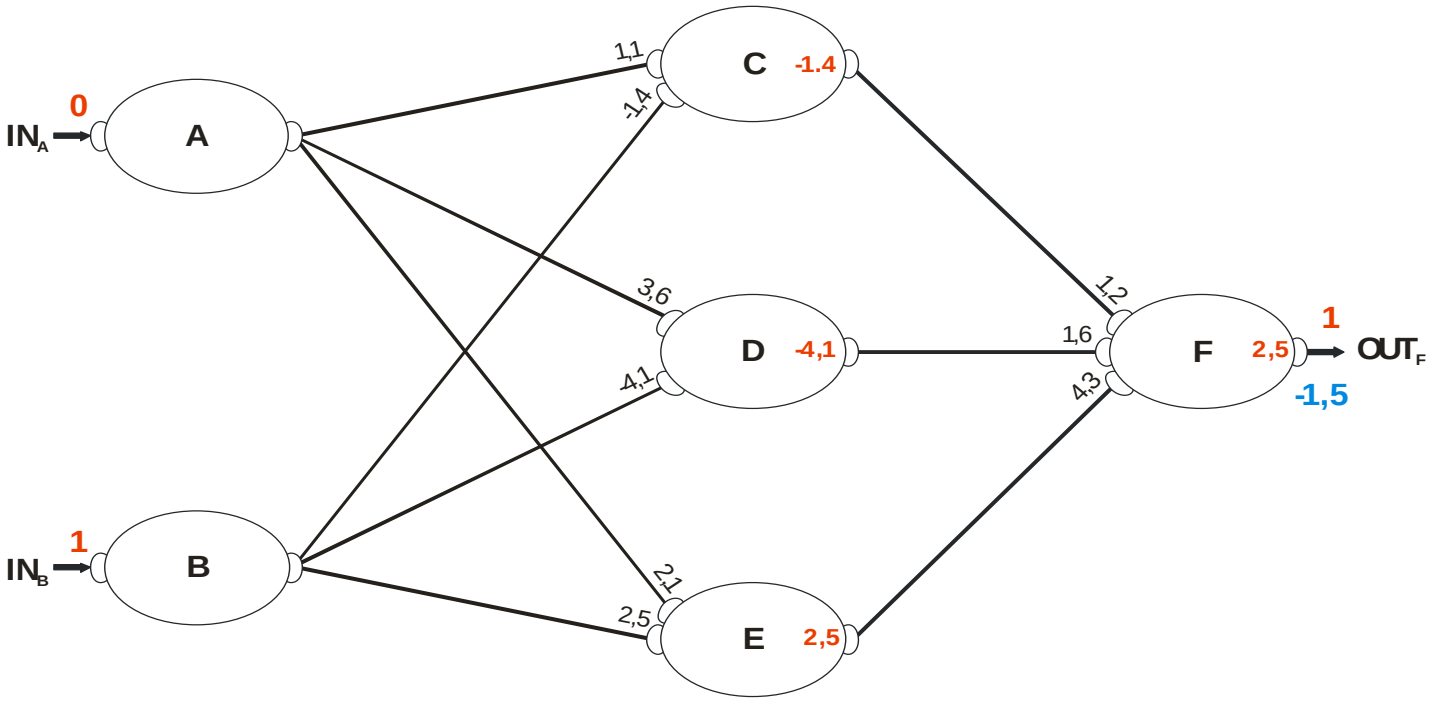


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

Computation of the error in the output layer ...

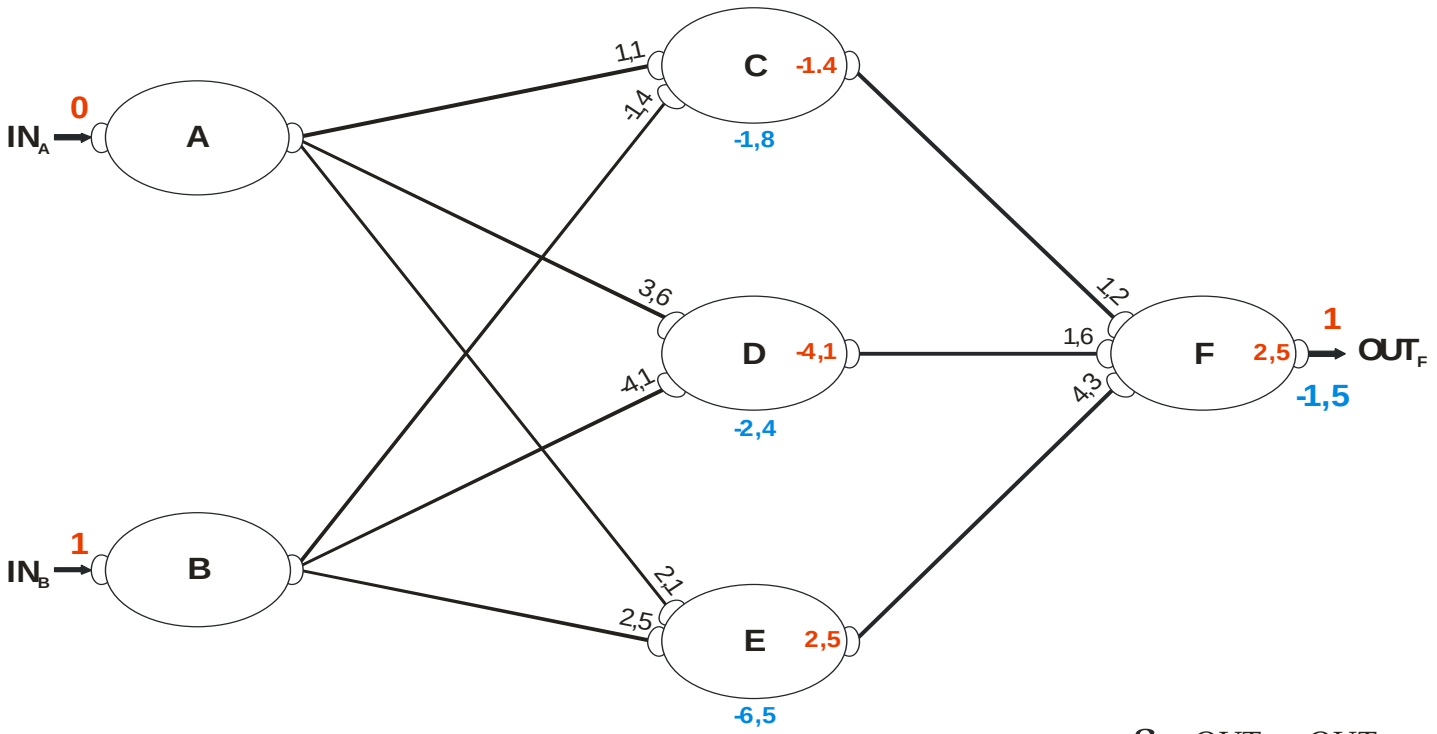


$$\mathcal{E} = OUT_{\mathcal{D}} - OUT_{C}$$

$$\mathcal{E}_{\leftarrow} = \mathcal{E} \times P$$

RNA's training

... and calculating the estimated value of the error in the middle layer.



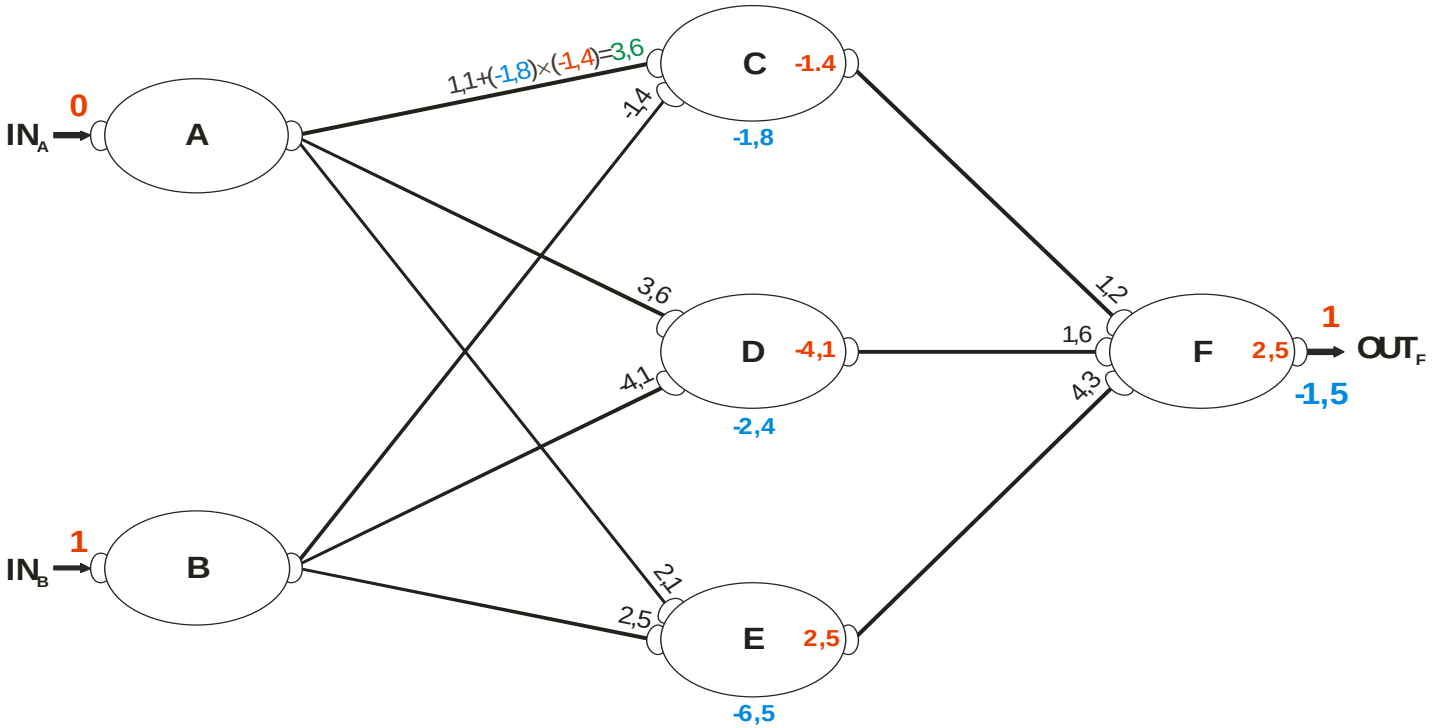
$$\mathcal{E} = OUT_D - OUT_C$$

$$\mathcal{E}_{\leftarrow} = \mathcal{E} \times P$$

ISLab
Synthetic Intelligence Lab

RNA's training

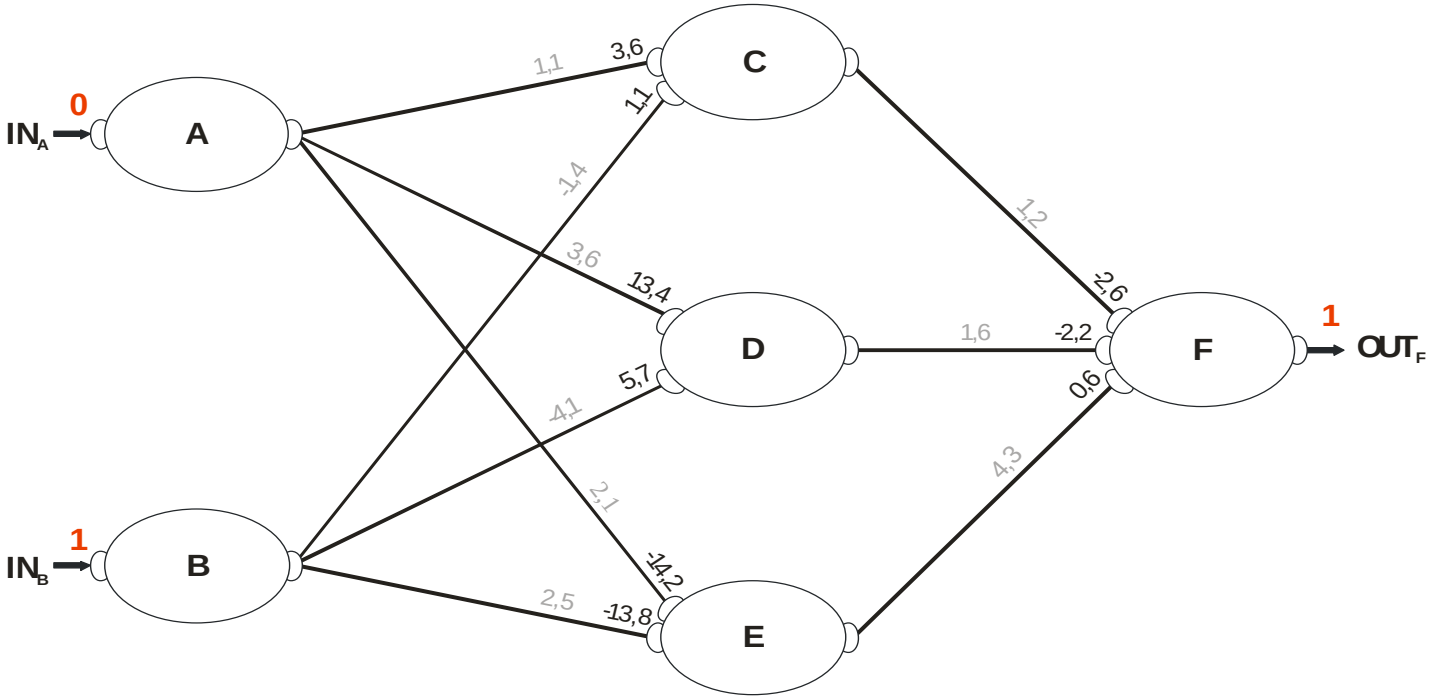
Application of a rule for updating synapse weights ...



$$P_{i+1} = P_i + \varepsilon \times f_T$$

RNA's training

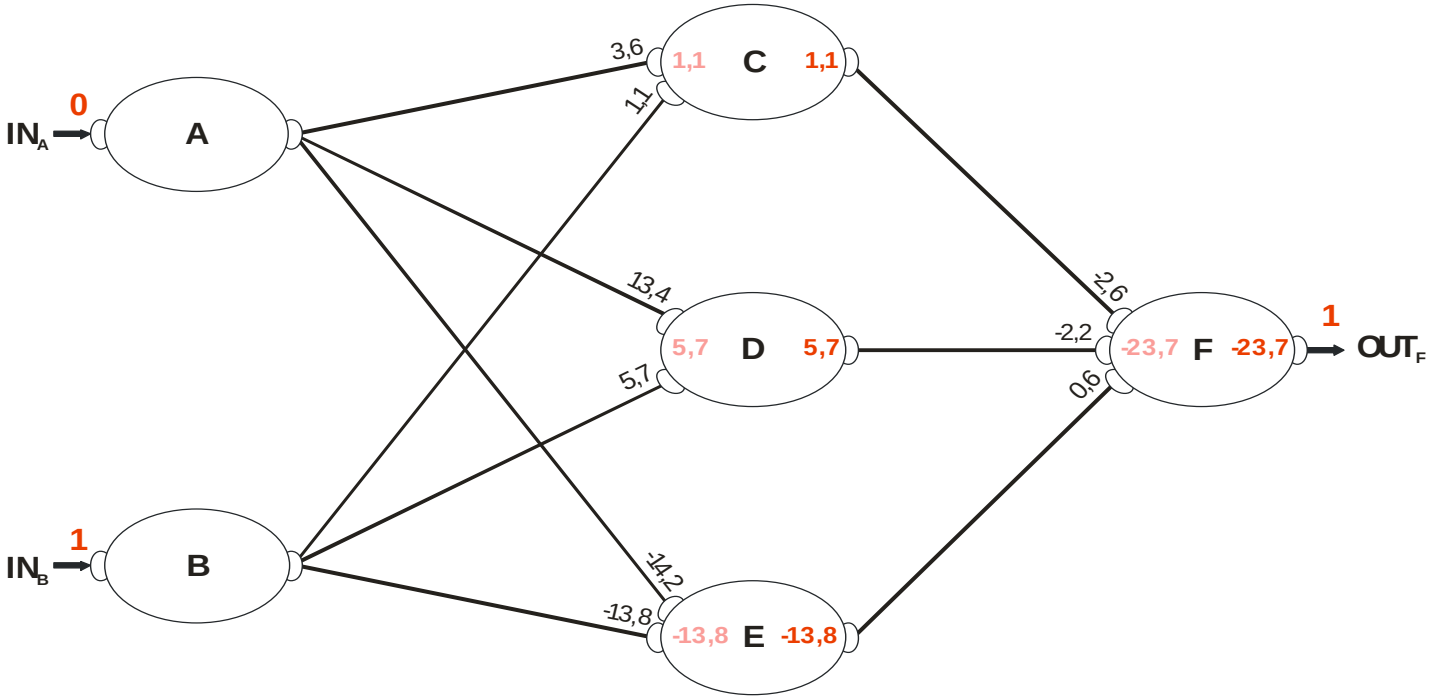
... to update the synapse values of all neurons.



$$P_{i+1} = P_i + \mathcal{E} \times f_T$$

RNA's training

Second iteration of the spread of the training case ...

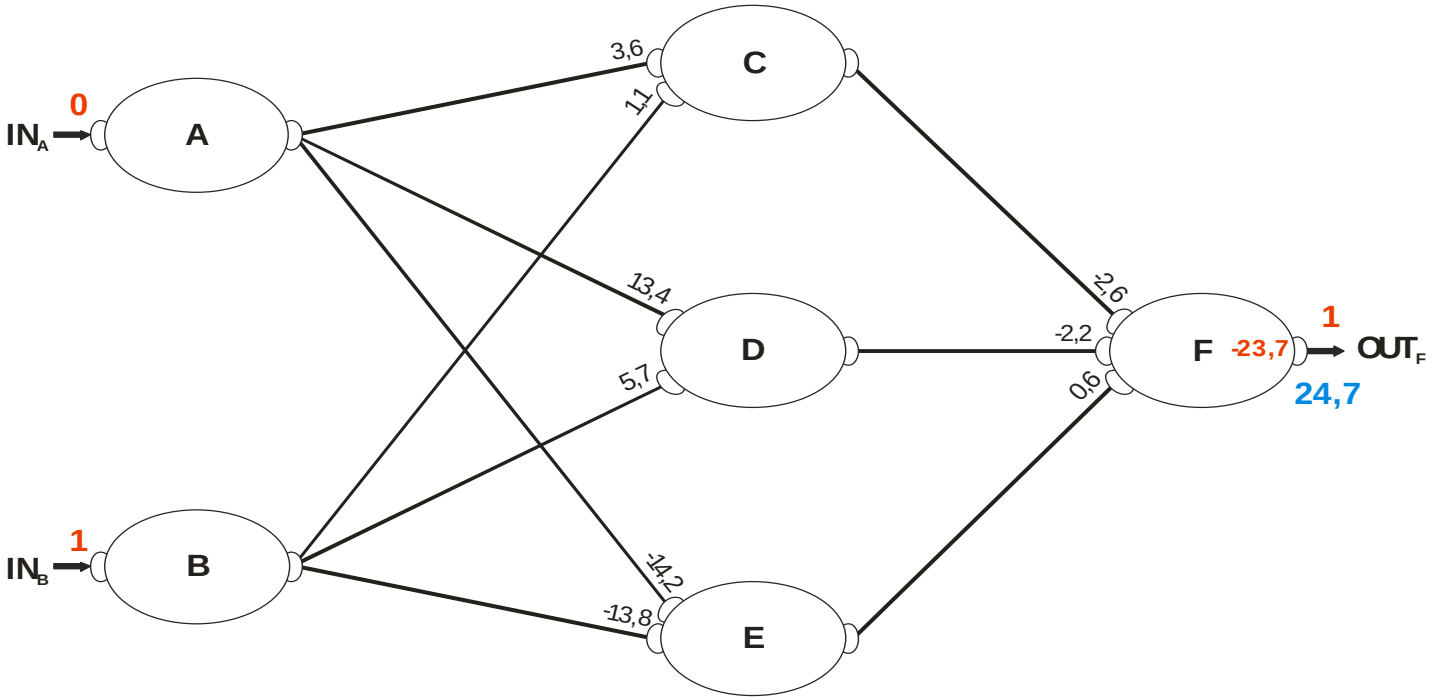


$$f_A(P,E) = \sum P \times E$$

$$f_T(A) = A$$

RNA's training

... and calculating the error produced by RNA in the second iteration.



Fundamental concepts: summary

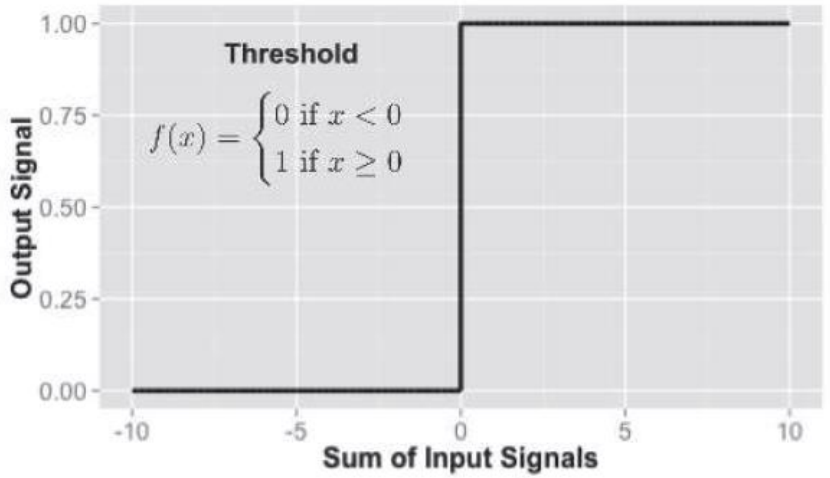
Although there are numerous variants of neural networks, each can be defined in terms of:

- An **activation function**, which transforms a neuron's net input signal into a single output signal to be broadcasted further in the network;
- A **network topology** (or architecture), which describes the number of neurons in the model as well as the number of layers and manner in which they are connected;
- The **training algorithm** that specifies how connection weights are set in order to inhibit or excite neurons in proportion to the input signal.

ISLab
Synthetic Intelligence Lab

Activation Function

- Artificial neuron processes information and passes it throughout the network
- Threshold activation function - output results only once a specified input threshold is reached



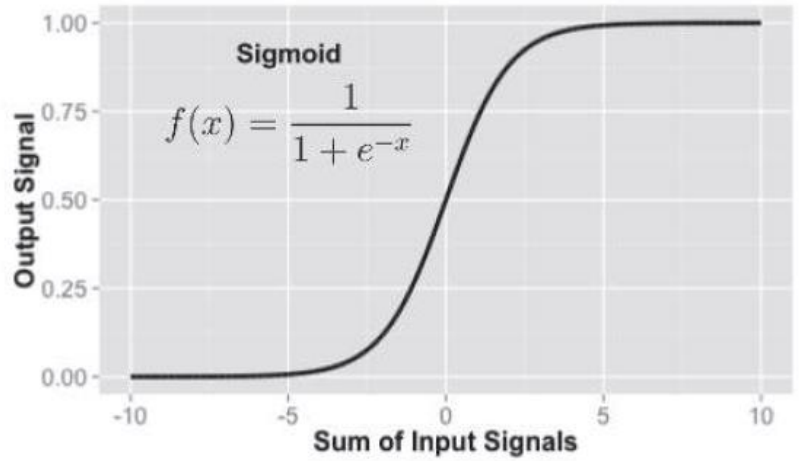
Unit step threshold activation function

The neuron fires when the sum of input signals is at least zero

Activation Function

Sigmoid activation function

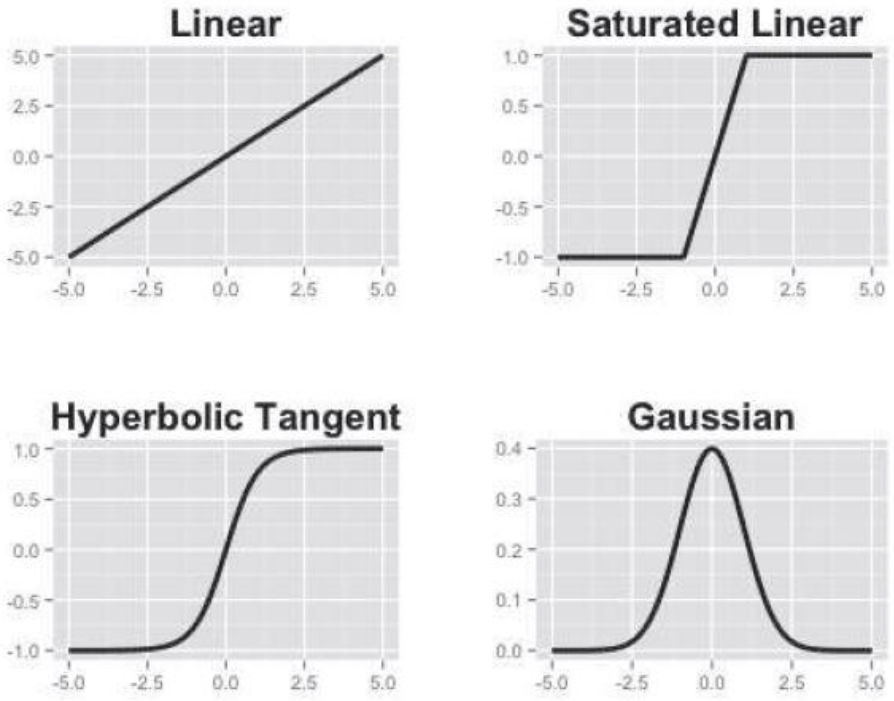
Logistic sigmoid, where e is the base of natural algorithms



- Most common
- Shares a similar step or S shape with the threshold activation function, but the output signal is no longer binary;
- output values can fall anywhere in the range from 0 to 1
- is differentiable, which means that it is possible to calculate the derivative across the entire range of inputs
  - Very important for backpropagation

allows the input to take any value in the range [-∞, + ∞] and compress it to the range [0, +1]

Activation Functions



The primary detail that differentiates among these activation functions is the output signal range

(0,1), (-1, +1), (-inf, + inf)

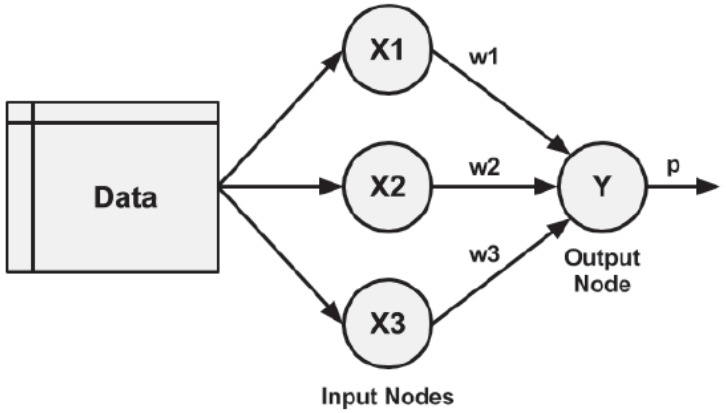
- a linear activation function results in a neural network very similar to a linear regression model
- a Gaussian activation function results in a model called a Radial Basis Function (RBF) network
  - allows input to take on any value in the range [-∞, + ∞] and compress it to the range [-1, +1]

Network Topology

- The capacity of a neural network to learn is rooted in its topology, or the patterns and structures of interconnected neurons;
- Determines the complexity of tasks that can be learned by the network;
  - generally, larger and more complex networks are capable of identifying more subtle patterns and complex decision boundaries
  - however, the power of a network is not only a function of the network size, but also the way units are arranged

1. The number of layers
2. The direction of information flow
3. The number of nodes within each layer of the network

Number of layers



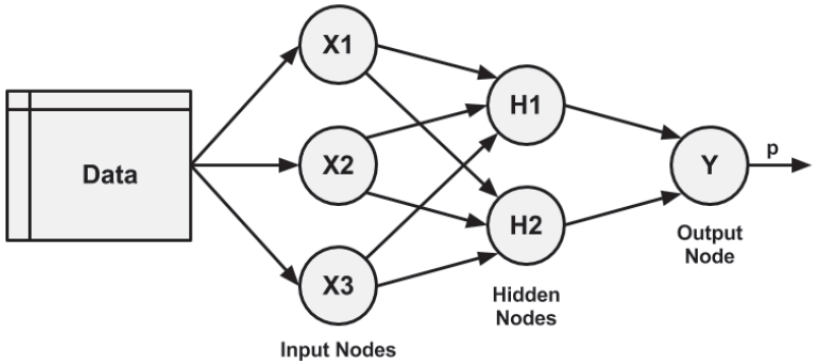
- Input Nodes receive unprocessed signals directly from the input data
- Each input node is responsible for processing a single feature in the dataset
- The signals resulting from the input nodes are received by the Output Node
- Output Node which uses its own activation function to generate a final prediction (denoted here as p)

The input and output nodes are arranged in groups known as layers

Single-layer networks can be used for basic pattern classification, particularly sophisticated networks are required for patterns that are linearly separable, but more sophisticated networks are required for most learning tasks

Number of layers

- Complex networks are created by adding additional layers -> multilayer network
  - Hidden layers

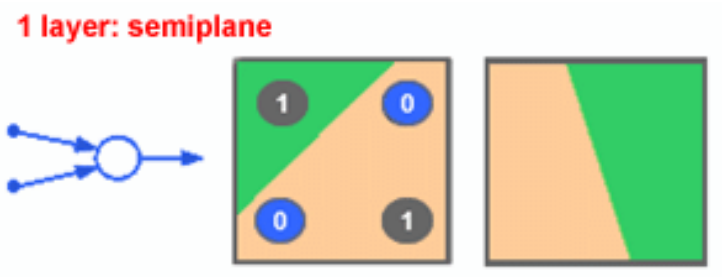


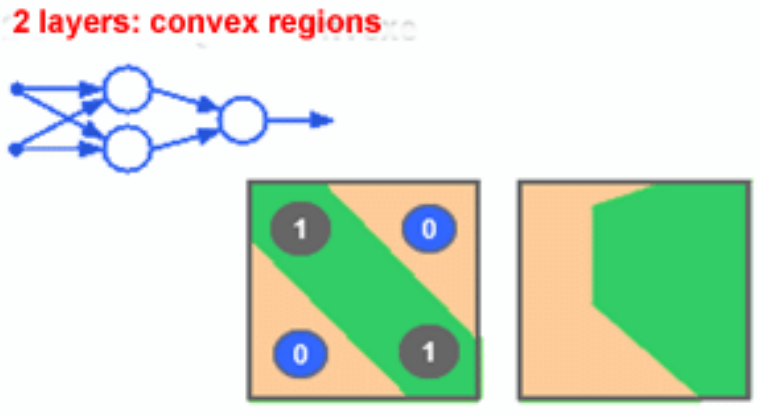
fully connected when every node in one layer is connected to every node in the next layer, but this is not required

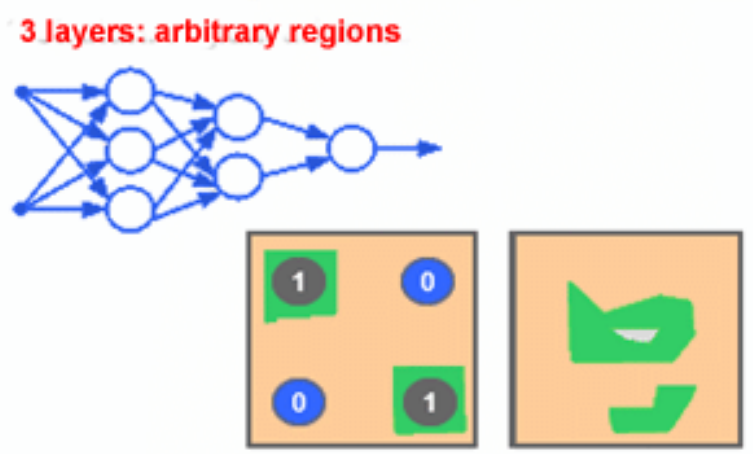
Deep Neural Networks

The term ‘deep’ refers to the number of layers in a neural network. While deep neural networks can have as many as two hundred layers, traditional neural networks only have a few, usually around three

Decision regions of multilayer perceptrons





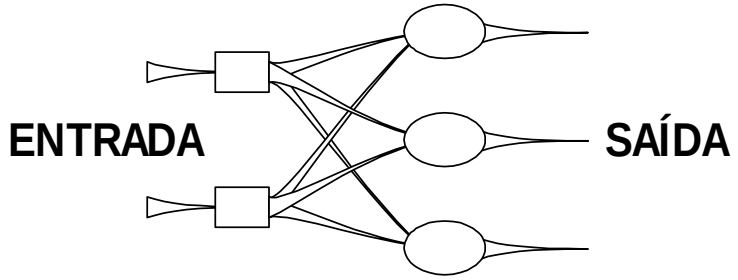


ISLab
Synthetic Intelligence Lab

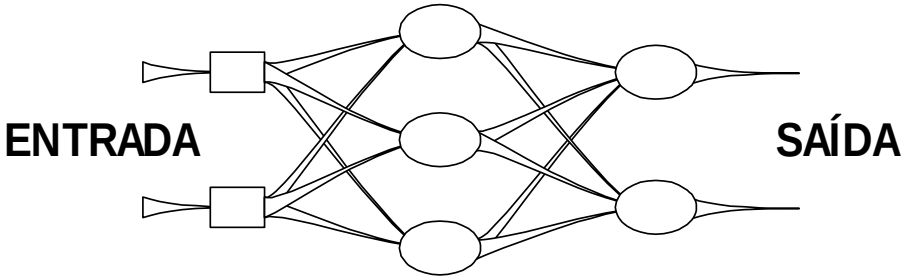
Direction of information flow

Feedforward, Single layer

The input signal is fed continuously in one direction from connection-to-connection until reaching the output layer

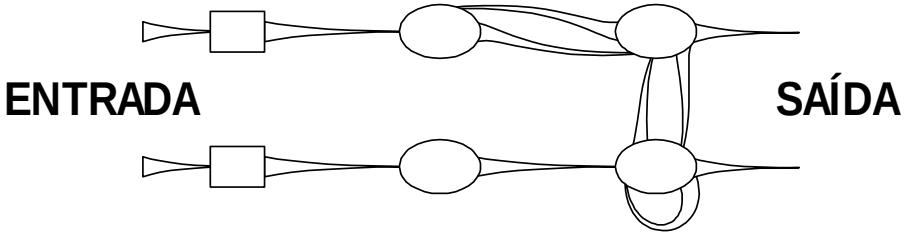


Feedforward, multi-layer:



Recurrent:

Allows signals to travel in both directions using loops



ISLab
Synthetic Intelligence Lab

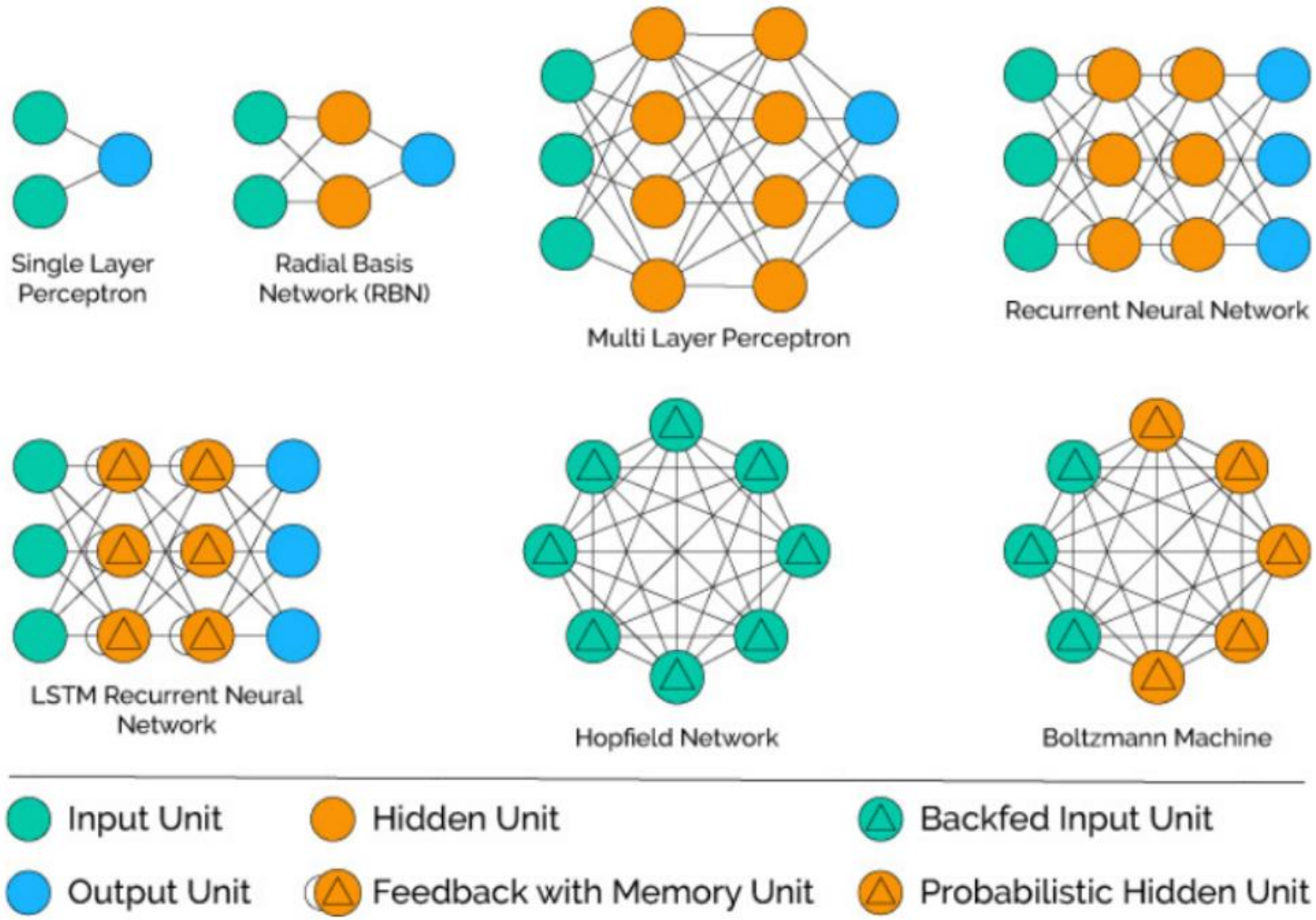
Number of nodes in each layer

- The number of input nodes is predetermined by the number of features in the input data;
- The number of output nodes is predetermined by the number of outcomes to be modeled or the number of class levels in the outcome;
- The number of hidden nodes is left to the user to decide prior to training the model - there is no reliable rule to determine the number of neurons in the hidden layer:
  - A greater number of neurons will result in a model that more closely mirrors the training data, but this runs a risk of overfitting -> it may generalize poorly to new data
  - Large neural networks can also be computationally expensive and slow to train

Use the fewest nodes that result in adequate performance on a validation dataset

ISLab
Synthetic Intelligence Lab

Types of Neural Networks



Types of Neural Networks

Perceptron Model in Neural Networks – A perceptron model Neural Network is having two input units and one output units with no hidden layers. These are also known as 'single layer perceptrons'.

Radial Basis Function Neural Network – These networks are similar to the feed-forward Neural Network except radial basis function is used as the activation function of these neurons.

Multilayer Perceptron Neural Network – These networks use more than one hidden layer of neurons, unlike single layer perceptron. These are also known as Deep Feedforward Neural Networks.

Recurrent Neural Network – Type of Neural Network in which hidden layer neurons has self-connections. Recurrent Neural Networks possess memory. At any instance, hidden layer neuron receives activation from the lower layer as well as its previous activation value.

Long Short-Term Memory Neural Network (LSTM) – Type of Neural Network in which memory cell is incorporated into hidden layer neurons.

Hopfield Network – A fully interconnected network of neurons in which each neuron is connected to every other neuron. The network is trained with input pattern by setting a value of neurons to the desired pattern. Once trained for one or more patterns, the network will converge to the learned patterns. It is different from other Neural Networks.

Boltzmann Machine Neural Network – These networks are similar to the Hopfield network however some neurons are input, whereas others are hidden in nature. The weights are initialised randomly and learn through backpropagation algorithm.

Training rules (learning)

Hebbian:

- If two nodes on each side of a connection are activated simultaneously, then the strength of that connection is progressively increased;
- If two nodes on each side of a connection are activated asynchronously, then the connection is progressively weakened or eliminated;
- Rule used in unsupervised learning

Competitive:

- The outputs of the nodes of the same layer compete with each other to become active, with only one node being activated at any given time. When a pattern is supplied to the network, one of the elements will respond better than the others, is therefore allowed to reinforce the weight of its connections.
- SOM (Self-Organinzing Maps) and the Kohonen

Stochastic:

- Connection weights adjusted in a probabilistic way. Simulated Annealing

Training rules (learning)

Memory-based:

- Stores all (or almost all) past experiences. We have explicitly a large memory of pairs input/output.
- Radial-Basis Functions

Descending gradient:

- Supervised learning: it is intended to reduce the error between the desired value and the output value. The error as a control mechanism, with successive adjustments;
- Back-propagation algorithm implementation.

The training algorithm

Backpropagation

- As the neural network processes the input data, connections between the neurons are strengthened or weakened similar to how a baby's brain develops as he or she experiences the environment;
- Algorithm that uses a strategy of back-propagating errors.

Multilayer feedforward networks that use the backpropagation algorithm.

Strengths	Weaknesses
• Can be adapted to classification or numeric prediction problems • Among the most accurate modeling approaches • Makes few assumptions about the data's underlying relationships	• Reputation of being computationally intensive and slow to train, particularly if the network topology is complex • Easy to overfit or underfit training data • Results in a complex black box model that is difficult if not impossible to interpret

Backpropagation algorithm

- Iterates through many cycles of two processes;
- Each iteration of the algorithm is known as an epoch;
- As the network contains no apriori (existing) knowledge, typically the weights are set randomly prior to beginning;
- the algorithm cycles through the two processes until a stopping criterion is reached;

Forward phase

- The neurons are activated in sequence from the input layer to the output layer, applying each neuron's weights and activation functions.

Backward phase

- The network's output signal resulting from the forward phase is compared to the true target value in the training data.
- Difference between the network's output signal and the true value results is an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors

ISLab
Synthetic Intelligence Lab

Backpropagation algorithm

- Difference between the network's output signal and the true value results is an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors;
- How does the algorithm determine how much (or whether) weight should be changed? Gradient descent:
  - Uses the derivative of each neuron's activation function to identify the gradient in the direction of each of the incoming weights—hence the importance of having a differentiable activation function;
  - The gradient suggests how steeply the error will be reduced or increased for a change in the weight;
  - The algorithm will attempt to change the weights that result in the greatest reduction in error by an amount known as the learning rate;
  - The greater the learning rate, the faster the algorithm will attempt to descend down the gradients, which could reduce training time at the risk of overshooting the valley.

ISLab
Synthetic Intelligence Lab

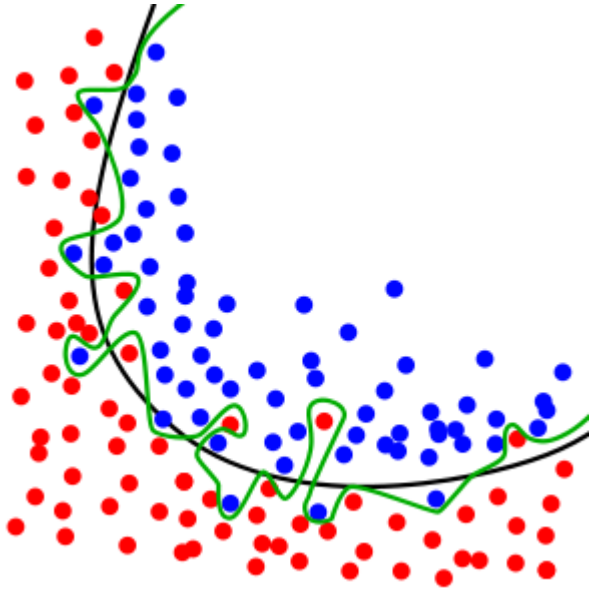
Data Preprocessing

- Data integrity check;
- Data representation;
- Data Scaling;
- Dimension reduction:
- Noise filtering.

Overfitting

“The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably“;

- An overfitted model is a statistical model that contains more parameters than can be justified by the data;
- The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e., the noise) as if that variation represented the underlying model structure;.
- The model remembers a huge number of examples instead of learning to notice features;
- Underfitting occurs when a statistical model cannot adequately capture the underlying structure of the data.
- “Over-training" and “Under-training".



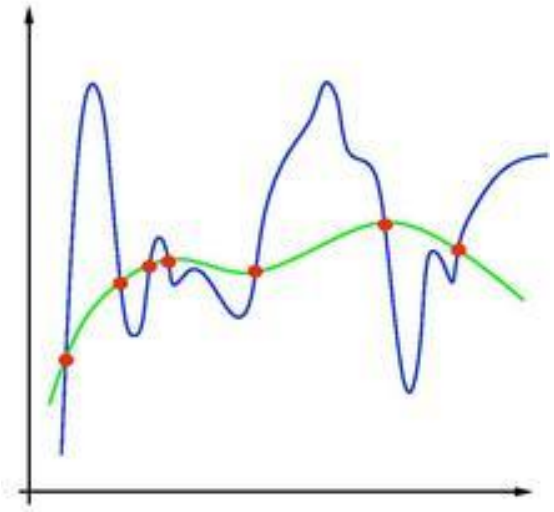
Regularization

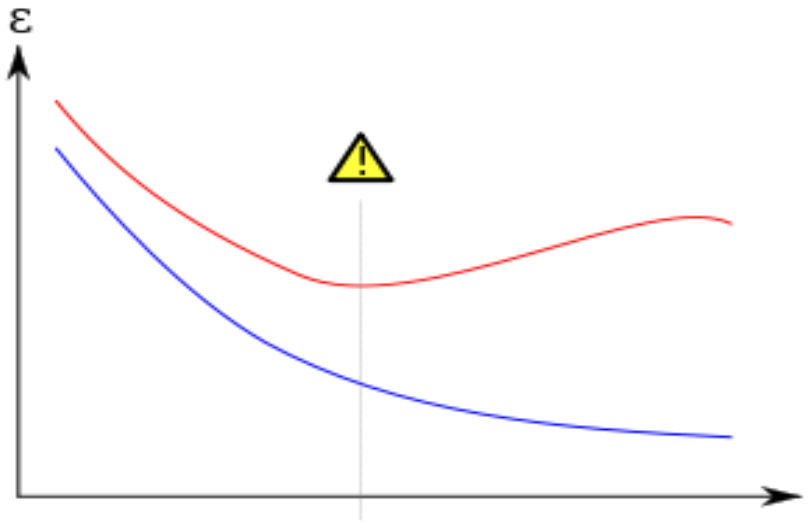
- Large neural nets trained on relatively small datasets can overfit the training data;
- The model learning the statistical noise in the training data, which results in poor performance when the model is evaluated on new data.
- **Dropout** is a regularization method that approximates training a large number of neural networks with different architectures in parallel.
  - During **training**, some number of layer outputs are randomly ignored or "*dropped out*." This has the effect of making the layer be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "*view*" of the configured layer.
  - During training, randomly set some activations to 0
    - Typically "drop" 50% of activations in layer;
    - Forces networks to not rely on any node.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

Regularization

- **Early stopping** is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent.
- Such methods update the learner so as to make it better fit the training data with each iteration. Up to a point, this improves the learner's performance on data outside of the training set. Past that point, however, improving the learner's fit to the training data comes at the expense of increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit.





Prechelt L. (1998) Early Stopping - But When?. In: Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-49430-8_3

Different techniques of ANNs

- **Classification Neural Network**
  - In a Classification Neural Network, the network can be trained to classify any given patterns or datasets into a predefined class. It uses Feedforward Networks to do this.
- **Prediction Neural Network**
  - In a Prediction Neural Network, the network can be trained to produce outputs that are expected from a given input. The network 'learns' to produce outputs that are similar to the representation examples given in the input.
- **Clustering Neural Network**
  - The Neural network can be used to identify a unique feature of the data and classify them into different categories without any prior knowledge of the data.
  - Following networks are used for clustering:
    - Competitive networks;
    - Adaptive Resonance Theory Networks;
    - Kohonen Self-Organising Maps.
- **Association Neural Network**
  - An Association Neural Network can be trained to remember a particular pattern, which enables any noise patterns presented to the network to be associated with the closest one in the memory or discard it.

ISLab
Synthetic Intelligence Lab

Artificial Neural Networks

Advantages

- Classification accuracy is usually high, even for complex problems;
- Distributed processing, knowledge is distributed through connection weights;
- Robust in handling examples even if they contain errors;
- Handle redundant attributes well, since the weights associated with them are usually very small;
- Results can be discrete, real values, or a vector of values (discrete or real).

Disadvantages

- Difficult to determine the optimal network topology for a problem;
- Difficult to use - have many parameters to define;
- Requires specific data pre-processing;
- Requires long training time;
- Difficult to understand the learning function (weights);
- Discovered knowledge is not readable;
- Do not provide explanations of the results;
- Domain knowledge incorporation is not easy.

@Tensorflow

```
# Load Data

(x_train, y_train), (x_test, y_test) =
tensorflow.keras.datasets.mnist.load_data()

# Scale image pixel values to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255

# Transform image shape to vector
x_train = numpy.expand_dims(x_train, -1)
x_test = numpy.expand_dims(x_test, -1)

# Convert class vectors to binary class matrices
y_train = tensorflow.keras.utils.to_categorical(y_train, num_classes)
y_test = tensorflow.keras.utils.to_categorical(y_test, num_classes)
```

```
# Build the model
model = keras.Sequential(
    [keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax")])

# Train the model
model.compile(loss="categorical_crossentropy",
optimizer="adam", metrics=["accuracy"])
model.fit(x_train, y_train, batch_size=128, epochs=15,
validation_split=0.1)
```

ISLab
Synthetic Intelligence Lab

References

Minsky, M., & Papert, S., Perceptrons. M.I.T. Press, 1969.

R. P. Lippman, An Introduction to Computing with Neural Nets, IEEE Acoustics, Speech and Signal Processing Magazine, 1987.

Christopher Bishop, Neural Networks for Pattern Recognition, Clarendon Press;, 1 edition, 1996.

Paulo Cortez e José Neves, Redes Neuronais Artificiais, Universidade do Minho, 2000.

Pang-Ning Tan; Michael Steinbach; Anuj Karpatne; Vipin Kuma, Introduction to Data Mining, 2nd Edition, Pearson, ISBN: 9780133128901, 2019.

Prechelt L. (1998) Early Stopping - But When?. In: Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 1524. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-49430-8_3

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

ISLab

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Neural Networks

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Perfil SI :: Computação Natural