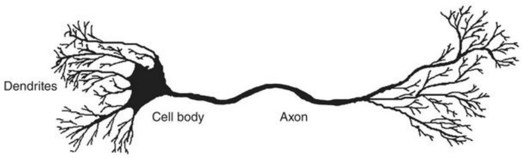
Machine learning

Neural networks

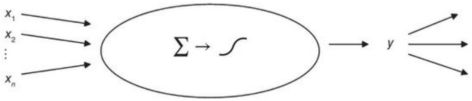
How does the brain work?

A neuron from the human brain uses dendrites to gather inputs from other neurons and combines the input information, generating a nonlinear response ("firing") when some threshold is reached, which it sends to other neurons using the axon in the form of an electrical signal.



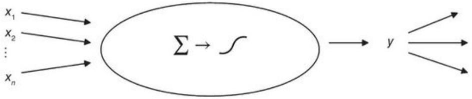
Neural networks represent an attempt at a very basic level to imitate the type of nonlinear learning that occurs in the networks of neurons found in nature, such as the human brain.

Neural networks



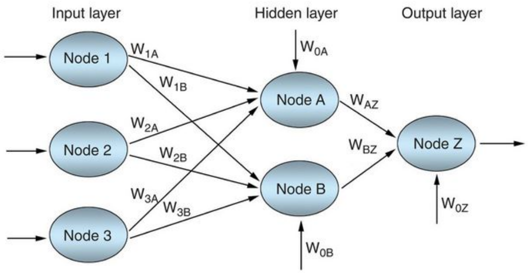
The inputs x_i are collected from the data set and combined through a combination function such as summation $\sum$, which is then input into a (usually nonlinear) activation function to produce an output response **y**, which may then be targeted downstream to other neurons.

Neural networks



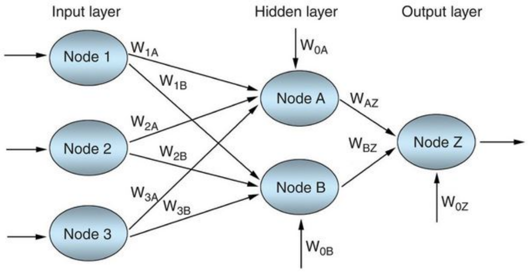
- **advantage** The main benefit of neural networks is that they are quite robust for **noisy, complicated, or nonlinear data**, due to the nonlinear nature of the activation function.
- **disadvantage** The main disadvantage of neural networks is that they are relatively non-transparent to human interpretation, unlike, say, decision trees.

Neural networks - structure



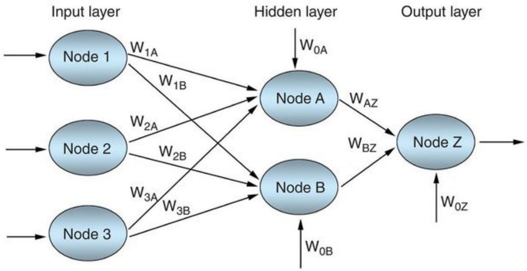
Artificial neural networks (ANNs) use simple computational elements - artificial neurons - to create systems capable of solving complex tasks.

Neural networks - structure



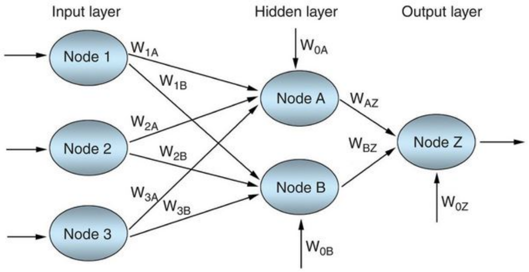
- A neural network consists of a layered, feedforward, completely connected network of artificial neurons or nodes.
- The feedforward nature of the network restricts the network to a single direction of flow and does not allow looping or cycling.
- Most networks consist of three layers: **an input layer**, **a hidden layer**, and **an output layer**.

Neural networks - structure



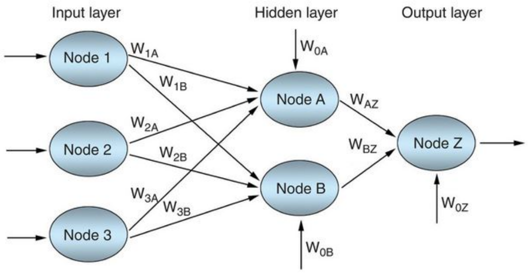
- There may be more than one hidden layer, although most networks contain only one, which is sufficient for most purposes.
- The neural network is **completely connected**, meaning that every node in a given layer is connected to every node in adjoining layers, **although not to other nodes in the same layer**.
- Each connection between nodes has a weight w_{1A} associated with it.

Neural networks - structure



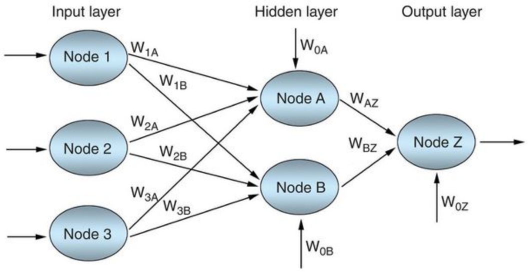
- The input layer accepts inputs from the data set, such as attribute values, and simply passes these values along to the hidden layer without further processing.
- There may be more than one hidden layer, although **most networks contain only one**, which is sufficient for most purposes.
- The feedforward nature of the network restricts the network to a single direction of flow and does not allow looping or cycling.

Neural networks - structure



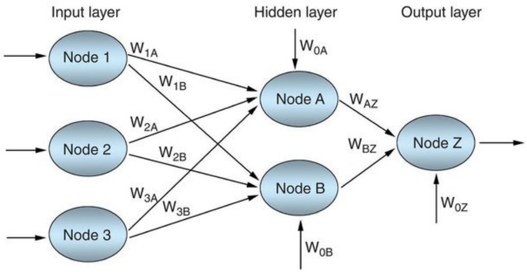
- Most networks consist of three layers: **an input layer**, **a hidden layer**, and **an output layer**.
- There may be more than one hidden layer, although most networks contain only one, which is sufficient for most purposes.

Neural networks - how many nodes should one have in the hidden layer?



- The number of hidden layers and the number of nodes in each hidden layer are both configurable by the analyst.
- Since having more nodes in the hidden layer increases the power and flexibility of the network for identifying complex patterns, one might be tempted to have a large number of nodes in the hidden layer.
- However, an overly large hidden layer leads to overfitting, memorizing the training set at the expense of generalizability to the validation set.

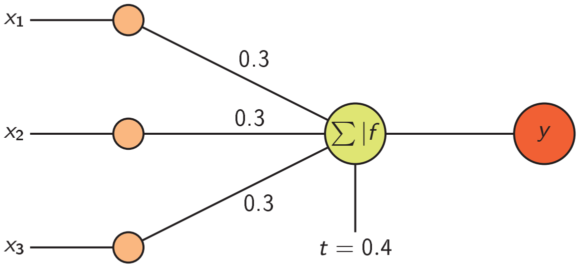
Neural networks - how many nodes should one have in the hidden layer?



If overfitting is occurring, one may consider reducing the number of nodes in the hidden layer.

If the training accuracy is unacceptably low, one may consider increasing the number of nodes in the hidden layer.

Perceptron



Perceptron– the simplest neural network, consisting of one or more independent neurons, implementing a supervised learning algorithm.

Perceptron learning

```
$D = \{(x_i, y_i)\}$, $i = 1, ..., n$ -training data set
weights $w$ initialise with any values
**repeat**
**for** each observation $(x_i, y_i) \in D$
    **do**
        compute the predicted value $\hat{y}_i^{(k)}$
        **for** each weight $w_j^{(k)}$
            **do**
                $w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$


**until** stop
```

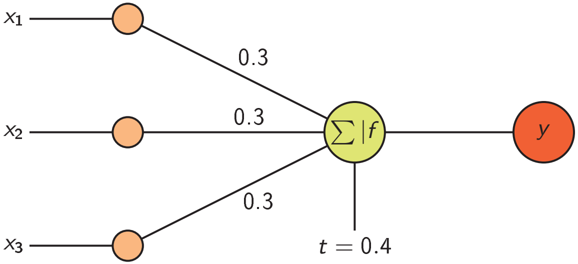
The parameter λ controls the step of the method (learning rate). Usually $\lambda \in [0, 1]$.

perceptron.R

Perceptron - example

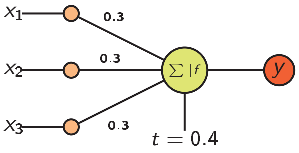
data set

x_1	x_2	x_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Perceptron

Perceptron - example



- link weights represent connections between neurons;
- model response $\hat{y}$ - weighted sum and bias t;

$$\hat{y} = \begin{cases} 1 & , 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0 \\ -1 & , 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 \leq 0 \end{cases}$$

or

$$\hat{y} = sgn(w_n x_n + ... + w_1 x_1 + w_0 x_0)$$

where $w_1, ..., w_n$ are link weights, $x_1, ..., x_n$ input values, function sgn is **activation function**, $w_0 = -t$, i $x_0 = 1$.

The activation function is called the function that allows us to calculate the value of the output of the neural network.

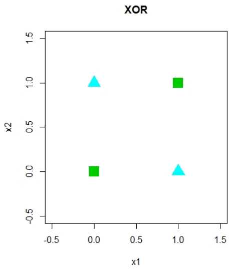
Interpretation of the weights

- The weights in a neural network determine the influence of the input variable on the output variable.
- A low weight value means that the variable has no effect on the neuron's excitation, while a higher weight value affects the excitation (firing) of the neuron and the output variable.

Perceptron learning

XOR

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1



The perceptron learning algorithm converges for sets where the data are linearly separable.

If the data are not linearly separable, then the algorithm generally does not converge. Furthermore,for nonlinearly separable problems, perceptron learning algorithm will fail because no linear hyperplane can separate the data perfectly.

Activation function

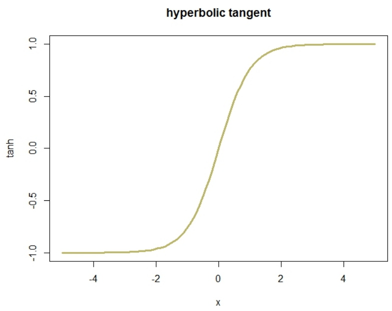
In artificial neural networks, there are several different neuron models, each of which uses a different type of activation function.

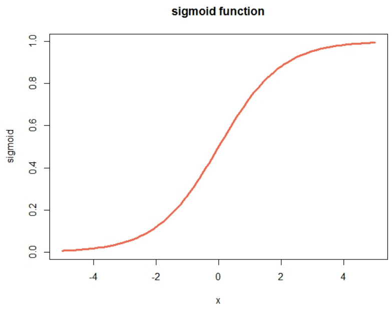
A number of different functions can be used as activation functions, including the linear function, unit jump function, **sigmoidal function, hyperbolic tangent** and others.

```
#sigmoidal function
sigmoid = function(x) {
  1 / ( 1 + exp(-x) )
  }
plot(sigmoid,-5,5)
#hyperbolic tangent
x<- seq(-5, 5, by=0.1)
t<-tanh(x)
plot(x,t, type="l", ylab="tanh")
```

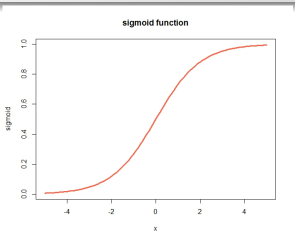
https://pl.wikipedia.org/wiki/Funkcja_aktywacji

Activation function





Sigmoid activation function

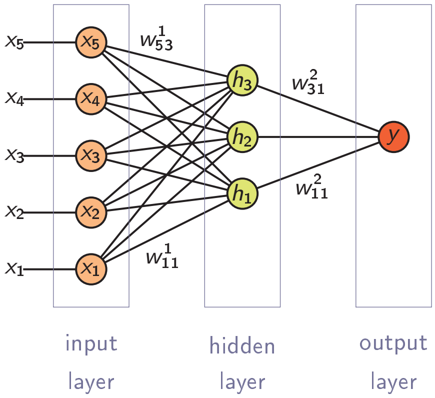


$$f(x) = \frac{1}{1 + e^{-x}}$$

A commonly used activation function is the sigmoidal function, which combines almost linear, curvilinear and almost constant behaviour, depending on the values of the input data.

Moderate increments in the value of x produce different increments of the value of f(x), depending on the location of x. Near the center, moderate increments in the value of x produce moderate increments in the value of f(x); however, near the extremes, moderate increments in the value of x produce give small increments of f(x).

Neural networks



- A neural network that contains multiple hidden layers is called a **multilayer** network.
- The **feed-forward** networks are networks in which nodes in one layer are only connected to nodes in the next layer.

Neural networks - learning

The aim of the learning algorithm **ANN** is to calculate a set of weights that minimise the mean square error MSE

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y})^2$$

where $\hat{y} = \mathrm{w} \cdot \mathrm{x}$ and

$$w_j = w_j - \lambda \frac{\partial E(w)}{\partial w_j}$$

where λ - learning rate - step length in the gradient method.

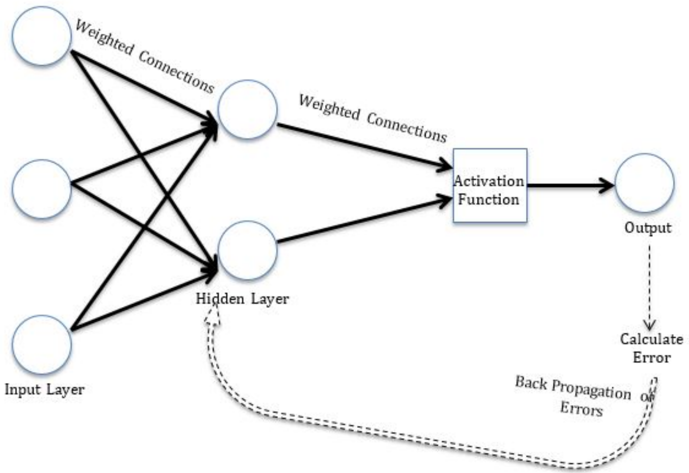
Neural networks - the backpropagation algorithm

The backpropagation algorithm **ANN** determines the optimal set of weights using the **back propagation of error**.

- there are two phases in each iteration of the algorithm: a forward phase and a backward phase;
- **during the forward phase**, the weights obtained from the previous iteration are used to calculate the output of each neuron in the network. The calculation proceeds in the forward direction: i.e. the outputs of the neurons at level k are calculated before the outputs at level k + 1 are calculated;
- **during the backward phase**, the update of the weights is applied in the reverse direction. In other words, the weights at level k + 1 are updated before the the weights at level k.

Back propagation allows the errors of the neurons in the k + 1 layer to be used to estimate the errors of the neurons in the k layer.

Neural networks - the backpropagation algorithm



Neural networks - the backpropagation algorithm

- if the learning time is short, then the resulting model is generally not accurate;
- if the network improves the weights for too long, it may become an overtrained network;
- the condition for completing the improvement is determined by a cross-validation mechanism;

Thank you for your attention!!!