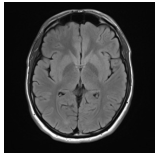
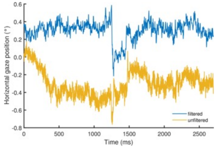
Neural Networks Introduction

Dr. U. Michelucci (TOELT)

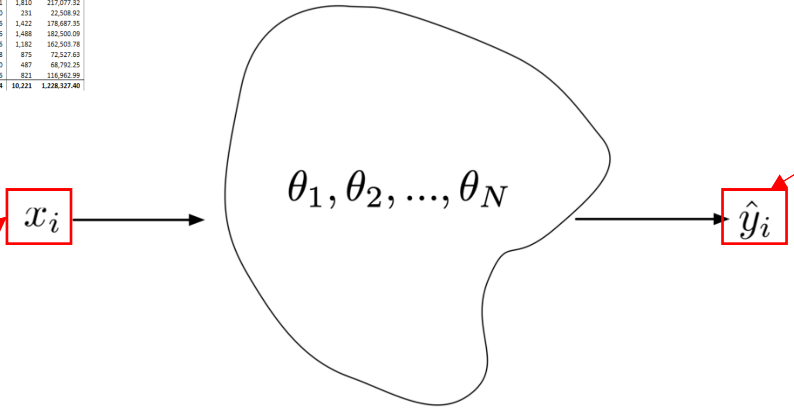
An intuitive understanding of a neural network



		2011		2012		2013		Total	
Salesperson	Country	Units	Order Amt	Units	Order Amt	Units	Order Amt	Units	Order Amt
Bromley	UK	232	24,756.89	228	40,396.64	73	9,894.51	533	75,048.04
Bromley	USA	58	7,553.95	27	3,654.00	7	1,101.20	92	12,309.15
Callahan	USA	623	49,400.07	337	43,263.95	200	18,059.50	1,160	110,723.52
Coghill	UK	81	4,029.25	39	4,657.11			120	8,686.36
Coghill	USA	885	120,626.31	520	46,505.90	405	49,945.11	1,810	217,077.32
Farnham	UK	170	14,055.87	44	5,892.65	17	2,560.40	231	22,508.92
Farnham	USA	699	89,663.20	506	73,360.59	217	15,663.56	1,422	178,687.35
Finchley	USA	699	95,850.36	487	55,787.97	302	30,861.76	1,488	182,500.09
Fuller	USA	539	71,168.14	473	73,524.18	170	17,811.46	1,182	162,503.78
Gillingham	UK	397	40,826.37	276	17,181.58	202	14,519.68	875	72,527.63
Gloucester	UK	209	31,433.16	143	19,691.89	135	17,667.20	487	68,792.25
Rayleigh	UK	422	59,827.19	268	41,903.64	131	15,232.16	821	116,962.99
Grand Total		**5,014**	**609,190.76**	**3,348**	**425,820.10**	**1,859**	**193,316.54**	**10,221**	**1,228,327.40**



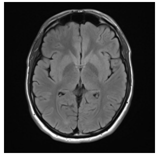
Inputs



Outputs

Class 1,
Class 2,
Class 3,
Etc.

1.73



A diagram that gives an intuitive understanding of what is a neural network. x_i are the inputs (for $i = 1, \ldots, n$), θ_i are numbers (or parameters) (for $i = 1, \ldots, N$), and $\hat{y}$ is the output of the network. The network itself is the depicted intuitively as the irregular shape in the middle of the figure.

An intuitive definition of learning

Note: a neural network is nothing else than a mathematical function that depends on a set of parameters that are tuned, hopefully in some smart way, to make the network output as close as possible to some expected output.

What does it mean "as close as possible"?

We need a mathematical way of measuring "closeness"

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

An intuitive definition of learning

Loss Function

Note: a neural network is nothing else than a mathematical function that depends on a set of parameters that are tuned, hopefully in some smart way, to make the network output as close as possible to some expected output.

Optimiser

Neural Network Architecture

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

Learning in the context of neural networks

Learning for a neural network means finding the best parameters θ to <u>**minimize**</u> the $\boxed{\text{loss function}}$ given a set of tuples (x_i, y_i) with $i = 1, \ldots M$.

A mathematical formula to measure "closeness"

Remember:

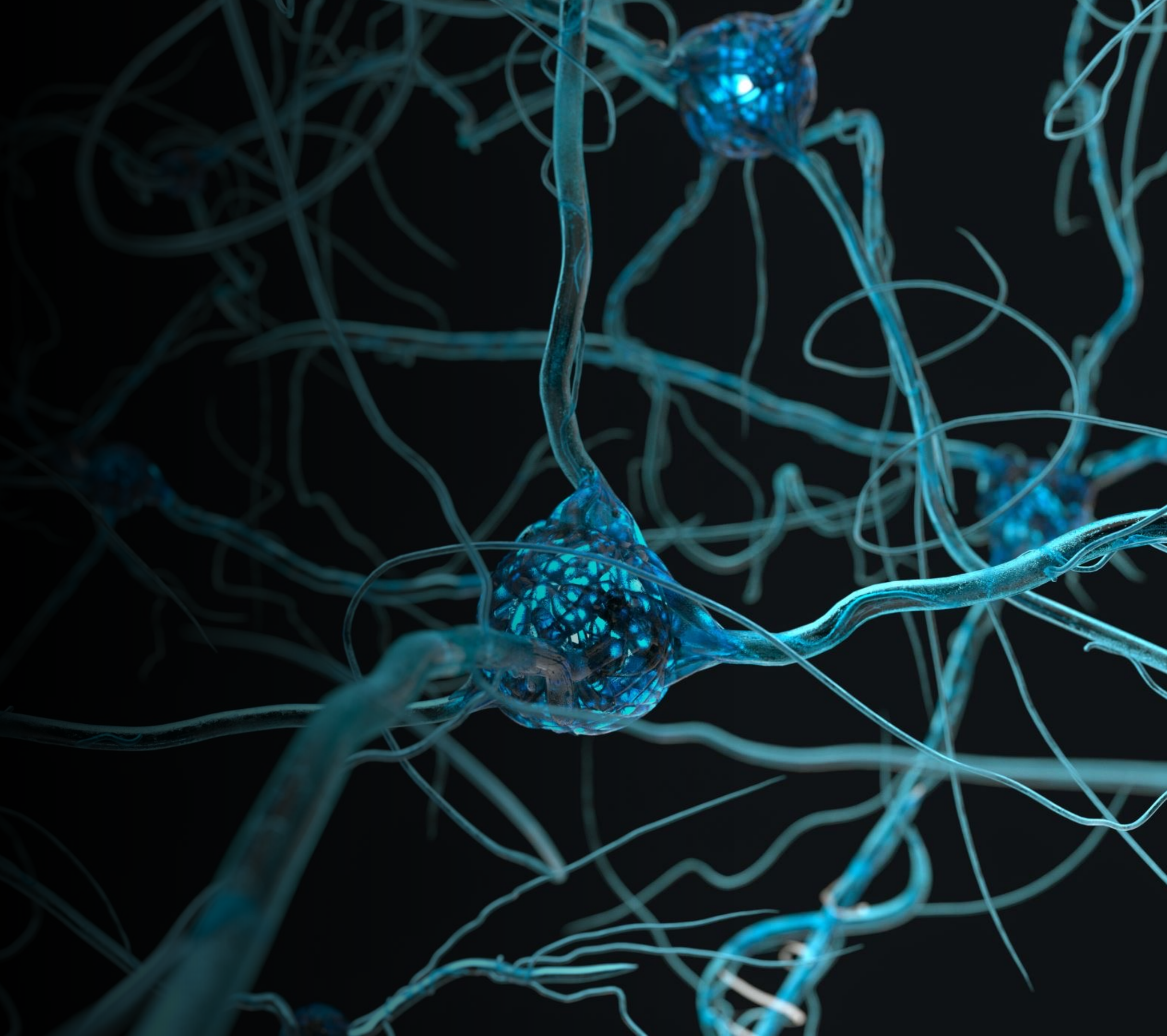
x_i → Inputs

y_i → Labels (or target variables)

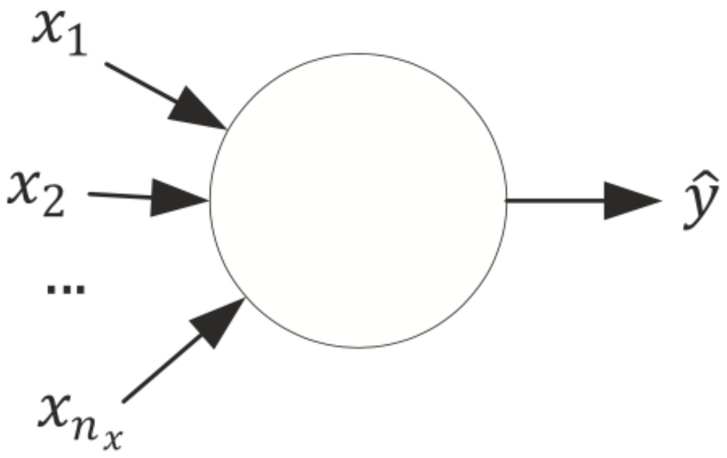
The **3 main components** of a *neural network model*

1) The *neural network architecture* (type of network, number of layers, etc.)
2) The *loss function* (the one we want to minimise)
3) The *optimiser* (the algorithm to find the minimum of the loss function)

Structure of a neuron

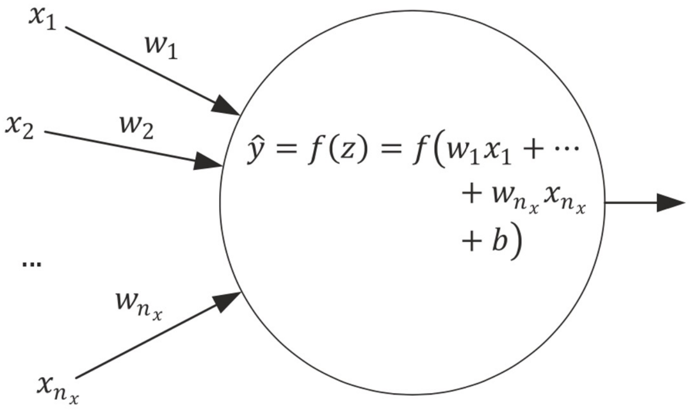


A neuron without details



Simplest network possible

The simplest neural network that we can create is made of one single neuron.



Note Practitioners mostly use the following nomenclature: w_i refers to weights, b bias, x_i input features, and f the activation function.

Source:Umberto Michelucci, Applied Deep Learning - A Case-Based Approach to Understanding Deep Neural Networks, APRESS 2018

Computational Steps of a single neuron

Let's summarize the neuron computational steps again.

Weights

1. Combine linearly all inputs x_i, calculating

$$z = \boxed{w_1} x_1 + \boxed{w_2} x_2 + \cdots + \boxed{w_{n_x}} x_{n_x} + \boxed{b};$$

Bias

2. Apply f to z, giving the output

$$\hat{y} = f(z) = \boxed{f}\left(w_1 x_1 + w_2 x_2 + \cdots + w_{n_x} x_{n_x} + b\right).$$

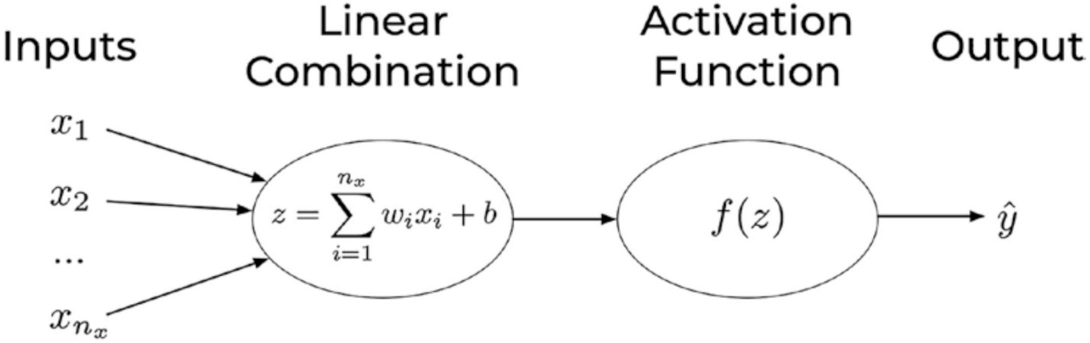
Activation Function

Source:Umberto Michelucci, Applied Deep Learning - A Case-Based Approach to Understanding Deep Neural Networks, APRESS 2018

Components of a neuron

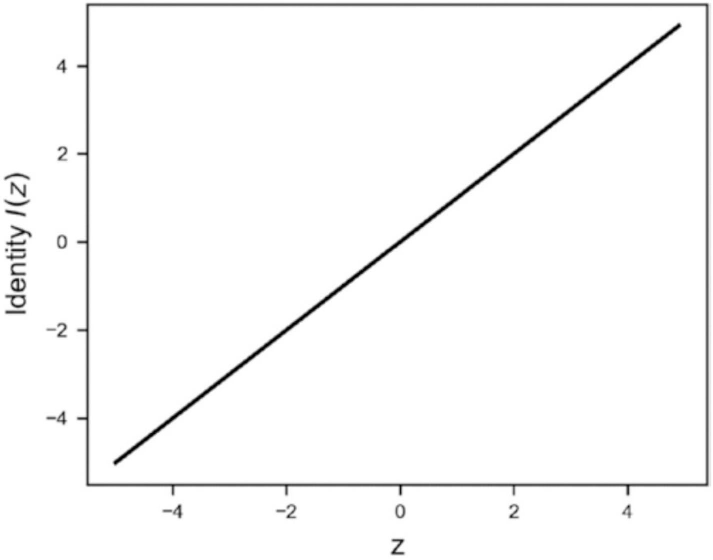
A neuron has two main components:

- The weights (w_i) (the bias b is sometime included in the weights)
- The *activation function* (the f in the previous slide)



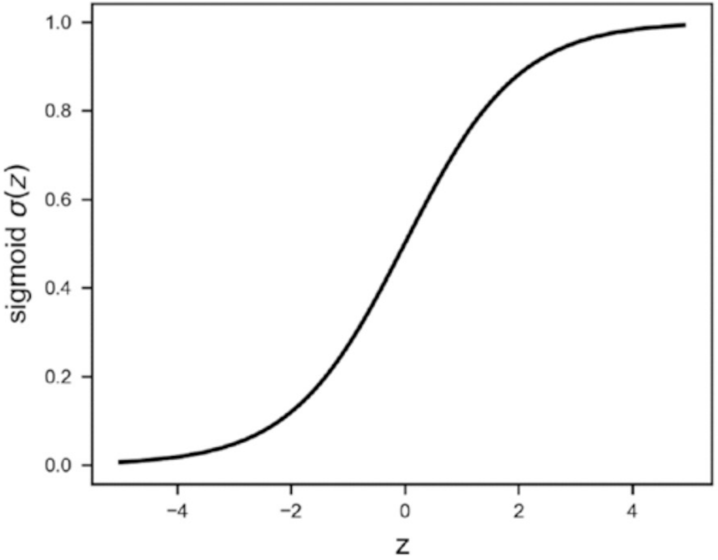
Activation Functions ($f(z)$)

Identity



$$f(z) = I(z) = z$$

Sigmoid

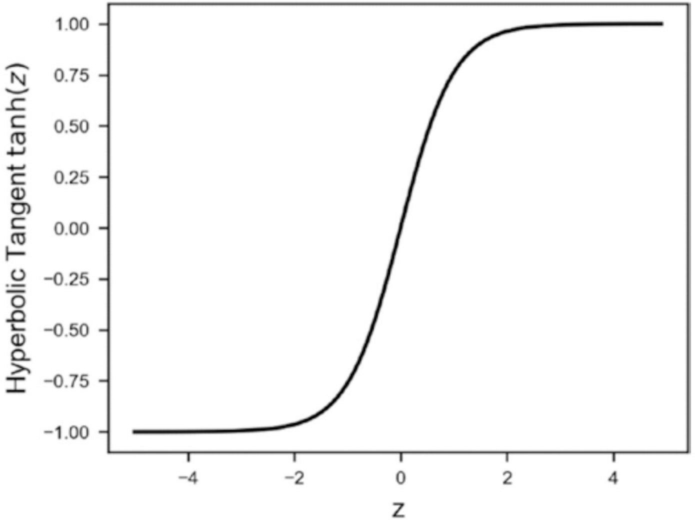


$$f(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

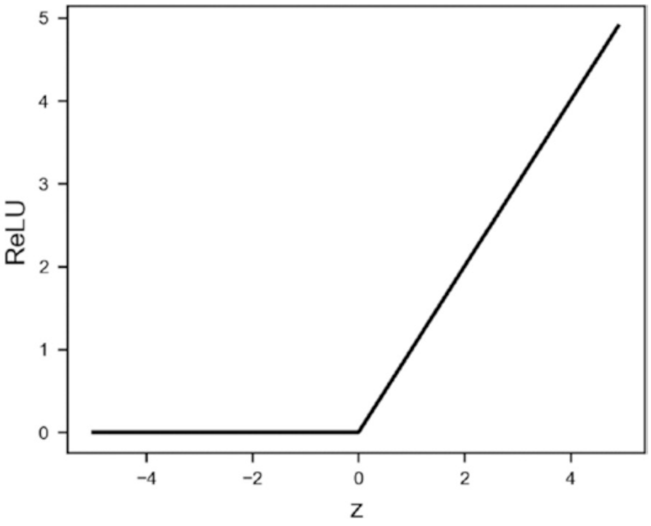
Activation Functions ($f(z)$)

Hyperbolic Tangent



$$f(z) = \tanh(z)$$

ReLU (Rectified Linear Unit)



$$f(z) = \max(0, z)$$

U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer Nature, 2022

Activation Functions ($f(z)$)

Many others:

- Leaky ReLU
- Swish
- ArcTan
- Exponential Linear Unit (ELU)
- Softplus

Learning for a neuron includes 3 components

- A neural network (defined by its architecture, in other words the computational steps to evaluate the network output)

- A loss function (a function to measure how good or bad the network is predicting the outputs)

- An algorithm to minimize the loss function (also called **optimizer**)

Source:Umberto Michelucci, Applied Deep Learning - A Case-Based Approach to Understanding Deep Neural Networks, APRESS 2018

Quiz: What can a neuron do?

- We have discussed how a neuron looks like. But how can a neuron solve any type of problem?
- Can a neuron solve a classification and a regression problem? How can we do that?

Question to you: which component of a neuron is responsible for the type of problem we want to solve?

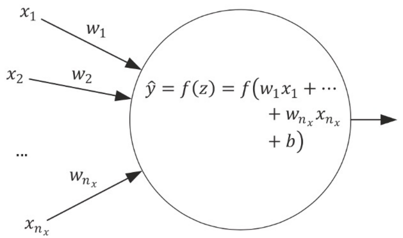
Which portion of a *neural network model* is responsible for the type of problem that can be solved?

Two main components are responsible for the type of problem that can be solved:

- The output activation function
- The loss function

→ The optimiser is not related in any way to the type of problem solved.

A single neuron can do a lot



Logistic regression (binary classification)

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

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = -\left(y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right)$$

Linear regression

$$f(z) = z$$

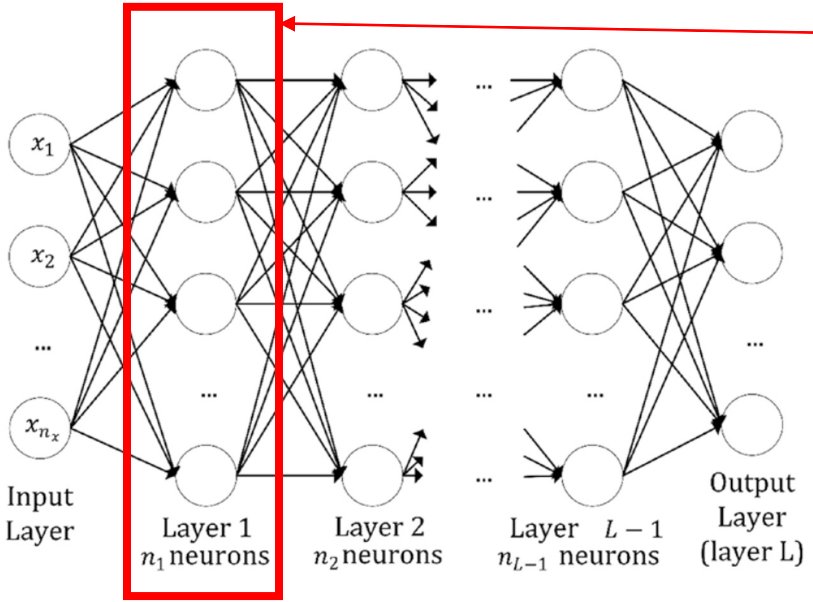
$$L(\hat{y}^{(i)}, y^{(i)}) = (y^{(i)} - \hat{y}^{(i)})^2$$

$\hat{y}^{(i)}$ Output of network for observation i

$y^{(i)}$ Expected output for observation i

$$z = w_1 x_1 + \ldots + w_{n_x} x_{n_x} + b$$

Multiple neurons / multiple layers



Layer (no intra-layer connection)

Quiz: Do you think there is any rule or best practice in deciding how many neurons/layers you should use?

- L: Number of hidden layers, excluding the input layer but including the output layer
- n_l: Number of neurons in layer l

$$N_{neurons} = n_x + \sum_{i=1}^{L} n_i = \sum_{i=0}^{L} n_i$$

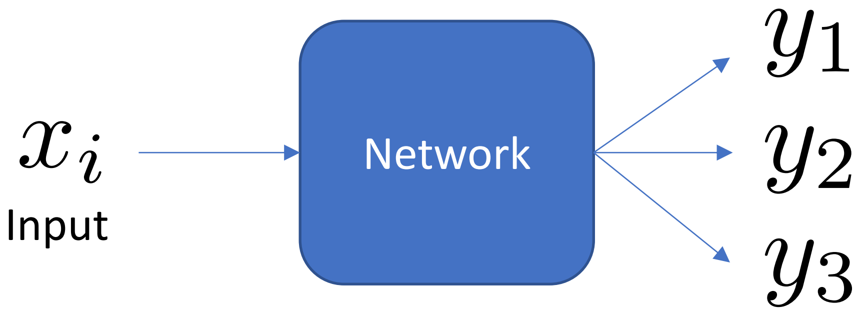
Source:Umberto Michelucci, Applied Deep Learning - A Case-Based Approach to Understanding Deep Neural Networks, APRESS 2018

How to do classification if the output is a real number?

Classification Problem and output activation function

The Softmax function

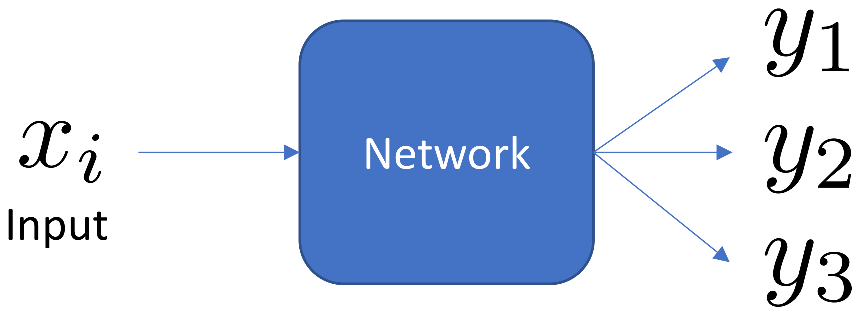
Neural Network Architecture (example with 3 classes)



How should we interpret those numbers?

We would like to interpret them as the probability of the input observation to be in class 1, 2 or 3.

Neural Network Architecture (example with 3 classes)

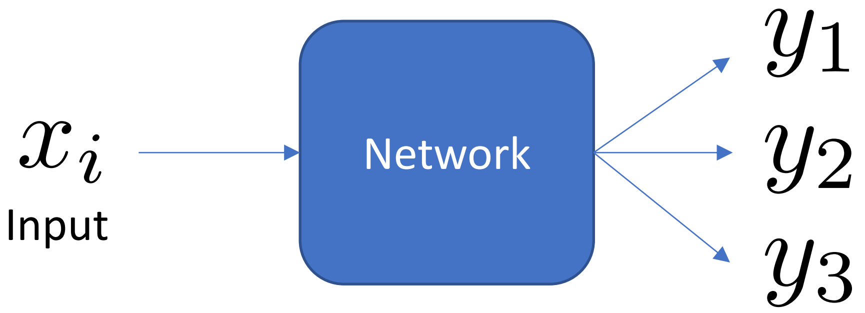


If we do not pay attention it can happen that the values of the y_i are large or negative. For example it can happen that

$$
\begin{aligned}
y_1 &= 5.4 \\
y_2 &= 0.4 \\
y_3 &= 12.5
\end{aligned}
$$

But how can those be interpreted as probabilities? REMEMBER: proability < 1 and the sum should be 1 (100%)

Neural Network Architecture (example with 3 classes)

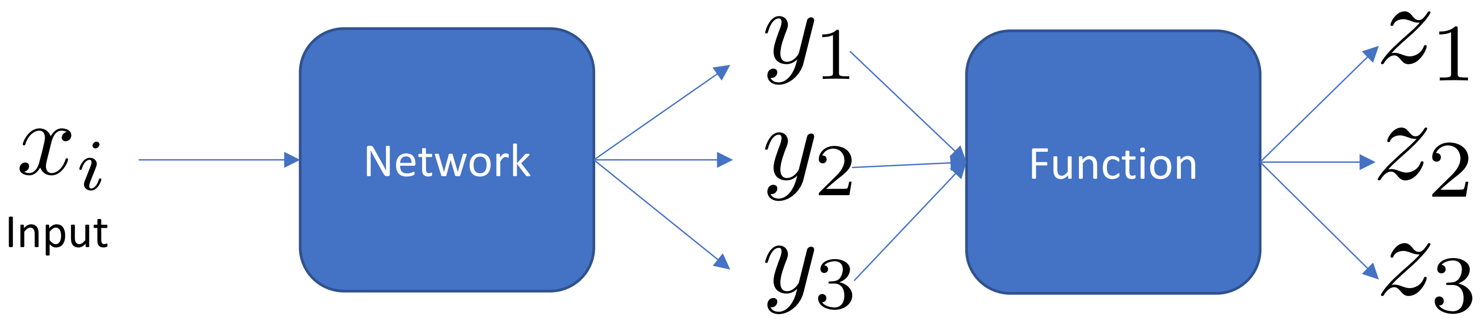


$y_1 = 5.4$

$y_2 = 0.4$

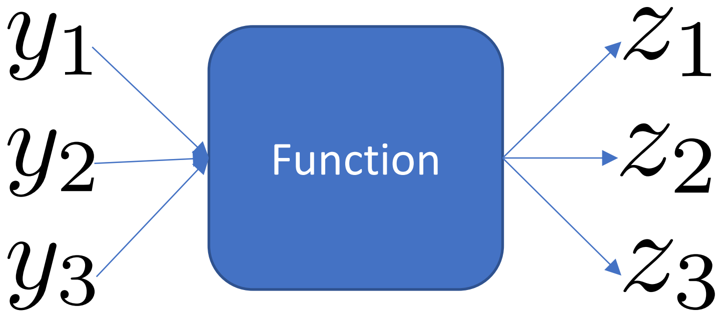
$y_3 = 12.5$

Neural Network Architecture (example with 3 classes)



$z_i \in [0, 1], \quad i = 1, 2, 3$ $\qquad$ $z_1 + z_2 + z_3 = 1$

Neural Network Architecture (example with 3 classes)

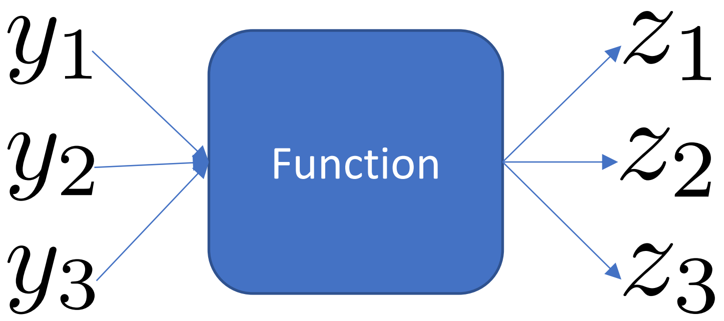


$$z_1 = \frac{e^{y_1}}{e^{y_1}+e^{y_2}+e^{y_3}}$$

$$z_2 = \frac{e^{y_1}}{e^{y_1}+e^{y_2}+e^{y_3}}$$

$$z_3 = \frac{e^{y_1}}{e^{y_2}+e^{y_2}+e^{y_3}}$$

Neural Network Architecture (example with 3 classes)

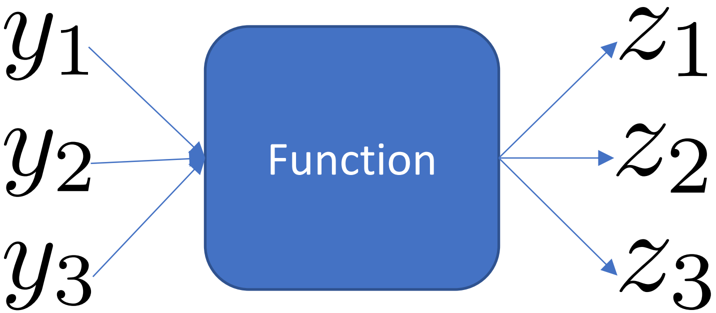


$$z_1 = \frac{e^{5.4}}{e^{5.4}+e^{0.4}+e^{12.5}}$$

$$z_2 = \frac{e^{0.4}}{e^{5.4}+e^{0.4}+e^{12.5}}$$

$$z_3 = \frac{e^{12.5}}{e^{5.4}+e^{0.4}+e^{12.5}}$$

Neural Network Architecture (example with 3 classes)



$z_1 = 0.08\%$

$z_2 = 5.5 \cdot 10^{-6}$

$z_3 = 99.91\%$

Softmax output layer for classification

$$S(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Output of ith neuron of the output layer

$$\sum_{i=1}^{k} S(\mathbf{z})_i = \sum_{i=1}^{k} \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} = \frac{\sum_{i=1}^{k} e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} = 1$$

Note that they can be interpreted as probability since the sum is equal to 1

Note We will look at $S(\mathbf{z})_i$ as a probability distribution over k with $i = 1, \ldots, k$ possible outcomes. For us, $S(\mathbf{z})_i$ will simply be the probability of our input observation being of class i.

Source:Umberto Michelucci, Applied Deep Learning - A Case-Based Approach to Understanding Deep Neural Networks, APRESS 2018

Challenges with Neural Networks

Additional difficulties

- **Regularisation** – How to avoid overfitting and make the models generalize better
- **Hyperparameter tuning** - finding the best parameters to obtain the best values for the chosen metric
- **Metric analysis** - detecting possible issues with the datasets
- **Training speed** - optimizing to get the fastest training possible
- **Big datasets** - working with amount of data that does not fit in memory anymore

Additional architectures

There are many architectures that are know to work best with different data types:

- Convolutional neural networks: for multi-dimensional data types (as images)
- Recurrent neural networks: time series, language, etc.

Many others have been developed to solve specific problem as: object localisation, segmentation (in MRI for example), etc.

From here the only limit is your creativity!