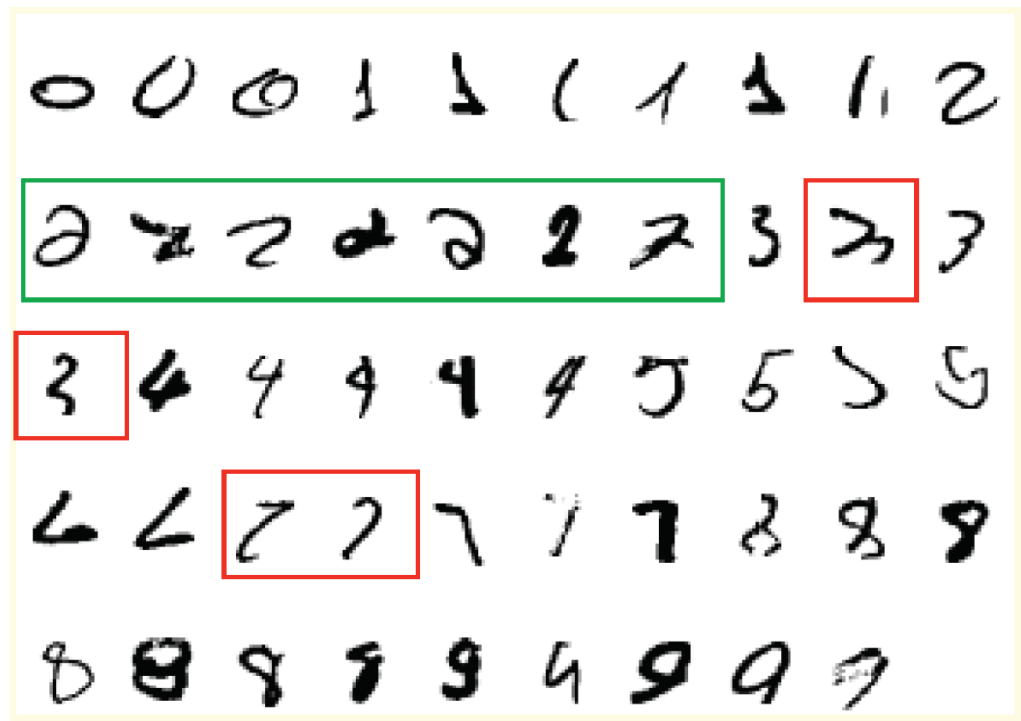


Neural Networks

Knut Hinkelmann

Motivation: Recognizing Numbers

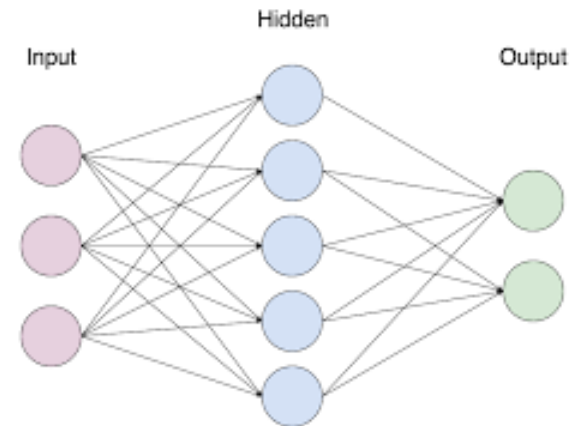
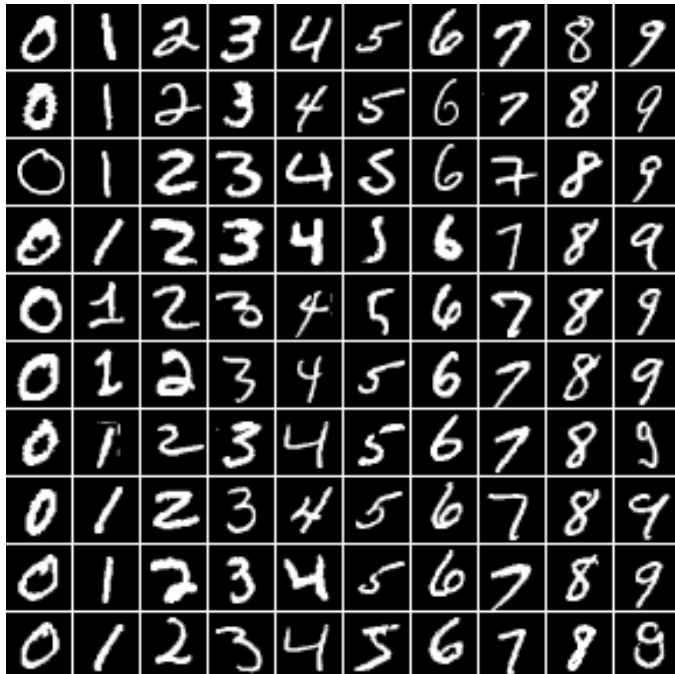
- It is very hard to specify what makes a «2»



- It is nearly impossible to create or learn symbolic rules.

Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec1.pdf

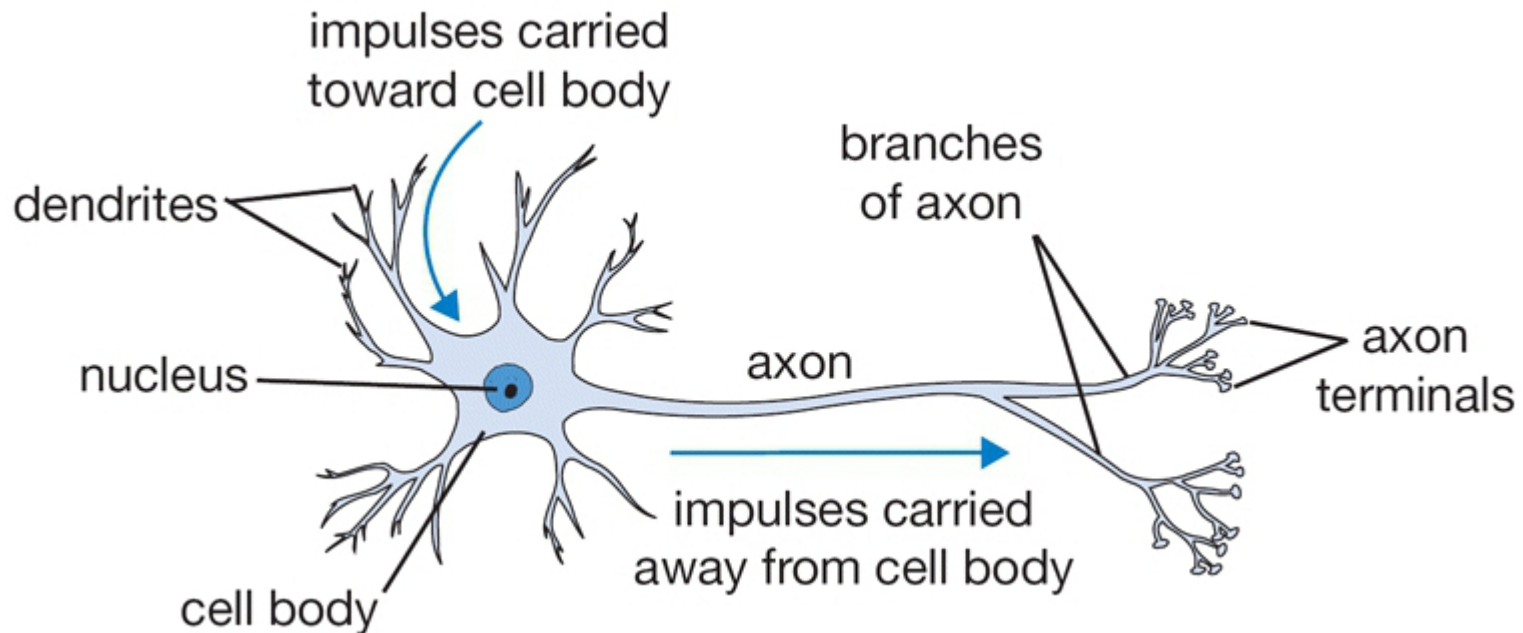
“Backprops” (Neural Networks)



- Classify the numbers

Biological Neuron

The basic computational unit of the brain is a **neuron**. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} — 10^{15} **synapses**.

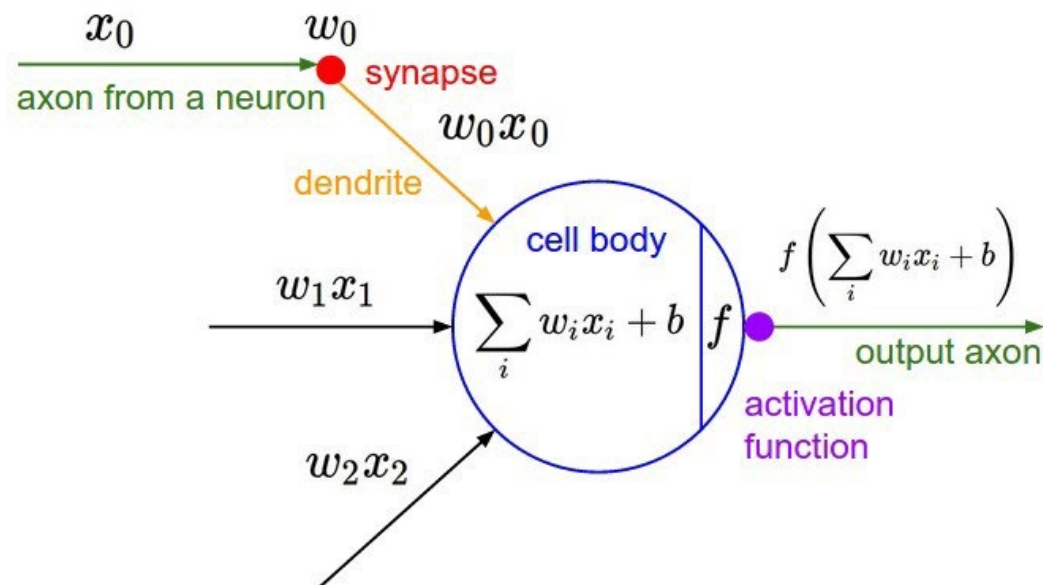


Source: David Fumo: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>



Mathematical Model of a Neuron

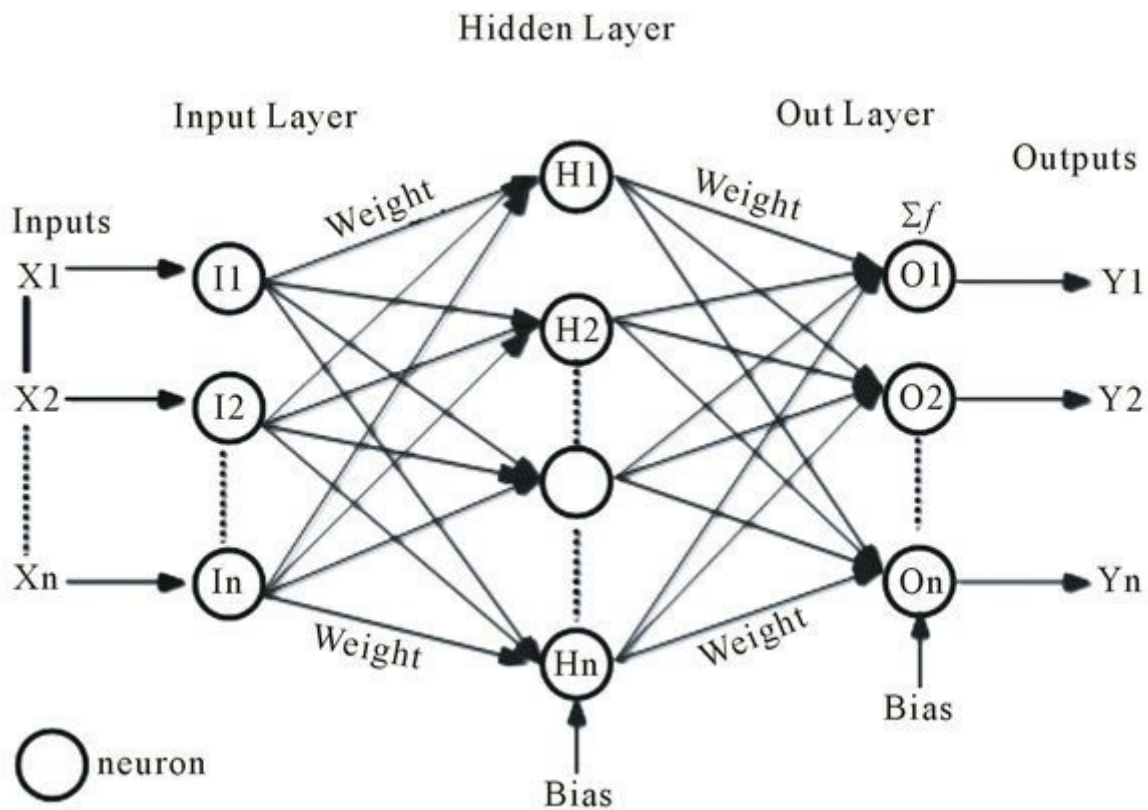
The basic unit of computation in a neural network is the neuron (often called a node or unit). It receives input from some other nodes, or from an external source and computes an output. Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function to the weighted sum of its inputs.



Source: David Fumo: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>



Multi-layer Perceptron



Source: David Fumo: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>



Basic Concepts of Neural Networks

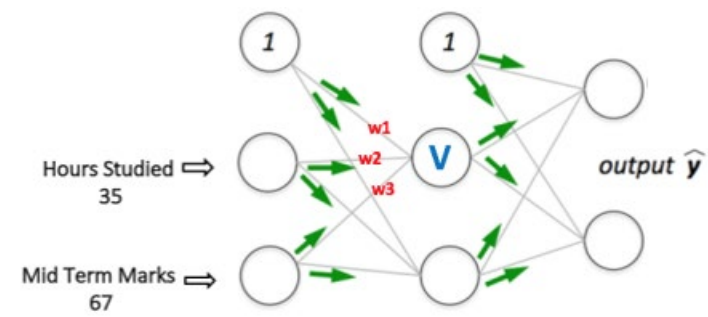
- **Input Nodes (input layer):** No computation is done here within this layer, they just pass the information to the next layer.
- **Output Nodes (output layer):** Here we finally use an activation function that maps to the desired output format.
- **Hidden nodes (hidden layer):** Hidden nodes transfer the weights from the input layer to the following layer (another hidden layer or to the output layer).
- **Connections and weights:** The *network* consists of connections, each connection transferring the output of a neuron i to the input of a neuron j . Each connection is assigned a weight W_{ij} .
- **Activation function:** Defines the output of that node given an input or set of inputs. *Nonlinear* activation functions allows such networks to compute nontrivial problems using only a small number of nodes.
- **Learning rule:** The *learning rule* modifies weights and thresholds of the neural network, in order for a given input to the network to produce a favored output.

Source: David Fumo: <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>



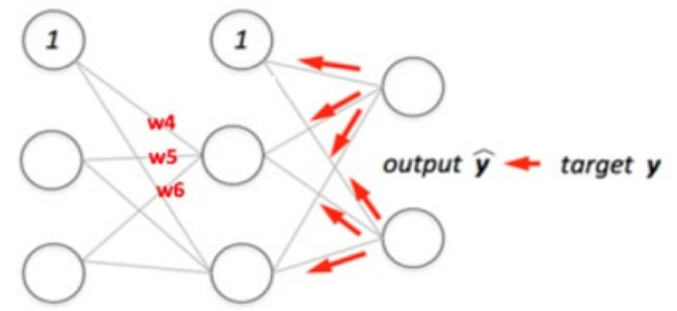
Principle of Application and (Supervised) Learning

- Application: The input values are **propagated forward** to the output



- Learning: The supervisor **corrects** the network whenever it makes mistakes (supervised learning). Total error at the output nodes are **propagated back** through the network leading to new weights

Backpropagation
+
Weights Adjusted

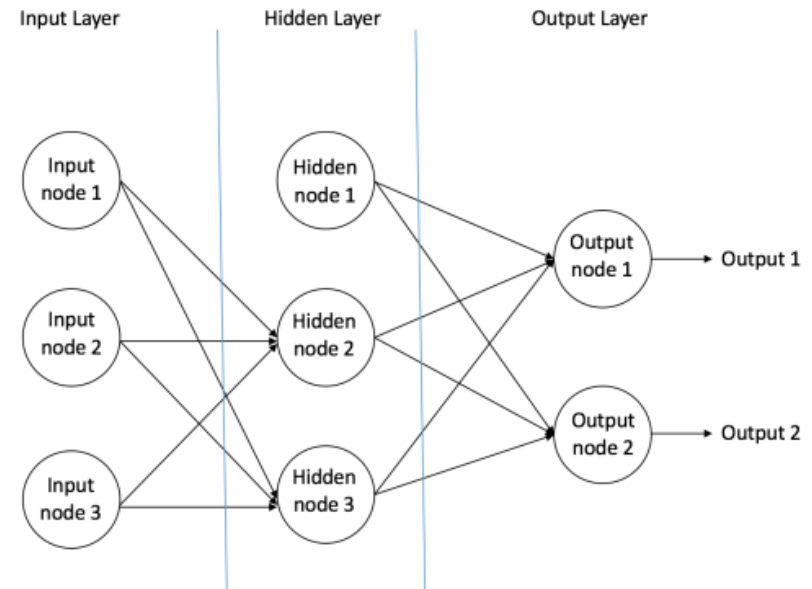


<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>



Feedforward Network

- In a feedforward network, the information moves in only one direction – forward – from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network
- Two examples of feedforward networks are given below:
 - ◆ **Single Layer Perceptron** – This is the simplest feedforward neural network and does not contain any hidden layer.
 - ◆ **Multi Layer Perceptron** – A Multi Layer Perceptron has one or more hidden layers.



<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>

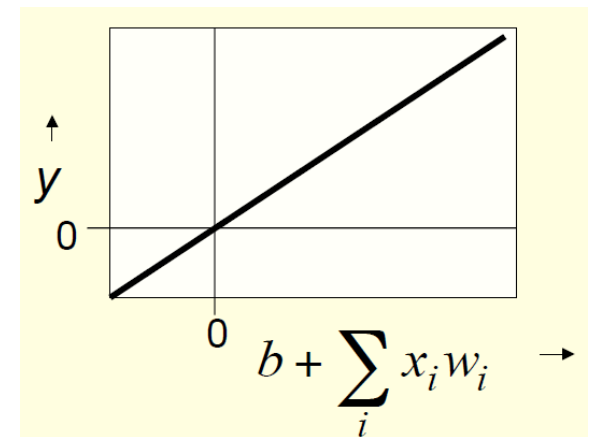
Simple Type of Neuron: Linear Neuron

- Simple but computationally limited

$$y = b + \sum_i x_i w_i$$

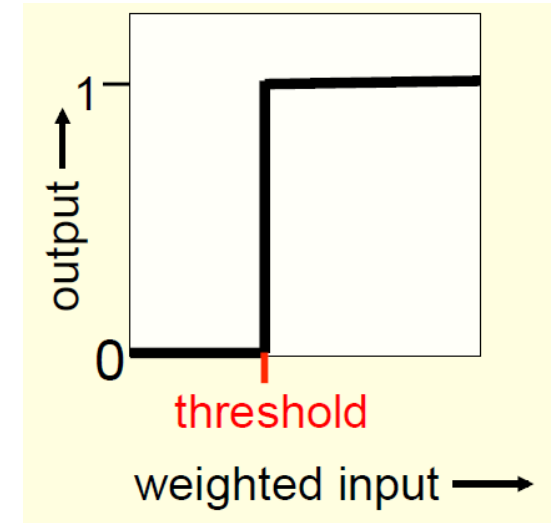
Diagram illustrating the linear neuron equation: $y = b + \sum_i x_i w_i$.

- y : output
- b : bias
- x_i : i^{th} input
- w_i : weight on i^{th} input
- i : index over input connections



Binary Threshold Neurons

- McCulloch-Pitts (1943)
 - ◆ First compute a weighted sum of the inputs.
 - ◆ Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
- There are two equivalent ways to write the equations for a binary threshold neuron:



$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

$$\theta = -b$$

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides lec1.pdf

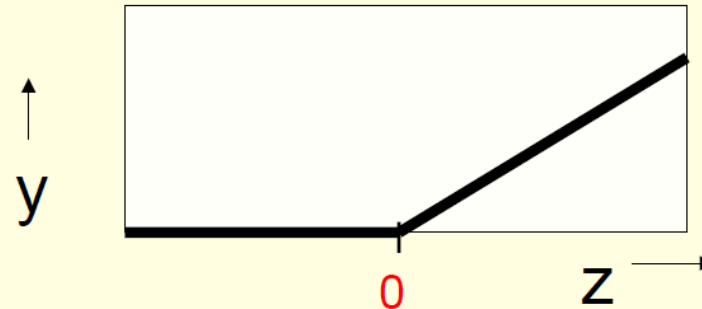


Rectified Linear Neurons

- They compute a *linear* weighted sum of their inputs.
- The output is a *non-linear* function of the total input.

$$z = b + \sum_i x_i w_i$$

$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

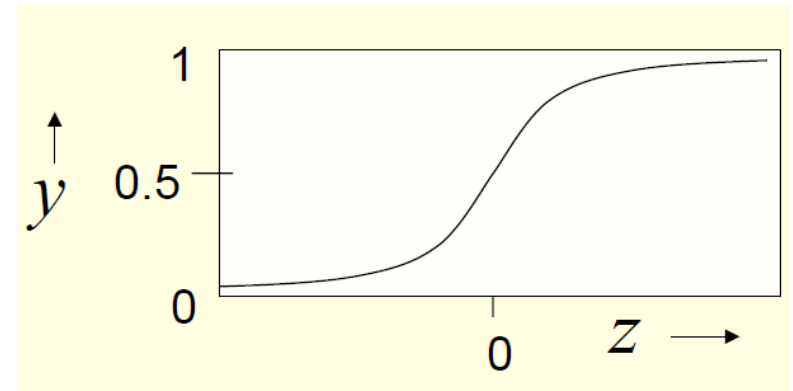


Sigmoid Neuron

- These give a real-valued output that is a smooth and bounded function of their total input.
- – Typically they use the logistic function

$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1 + e^{-z}}$$



Source: Geoffrey Hinton, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides lec1.pdf

Learning: Backpropagation

- **Backward Propagation of Errors**, often abbreviated as BackProp is one of the several ways in which an artificial neural network (ANN) can be trained.
- It is a supervised training scheme, which means, it learns from labeled training data.
- To put in simple terms, BackProp is like “**learning from mistakes**“. The supervisor **corrects** the ANN whenever it makes mistakes.

Backprop Algorithm

- Initially all the edge weights are randomly assigned.
- For every input in the training dataset, the ANN is activated and its output is observed. This output is compared with the desired output that we already know, and the error is “propagated” back to the previous layer. This error is noted and the weights are “adjusted” accordingly.
- This process is repeated until the output error is below a predetermined threshold.
- Once the above algorithm terminates, we have a “learned” ANN.

<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>



Backprop Example (1)

- We want to learn the probability of a student to pass or fail an exam

- Training set:

Hours Studied	Mid Term Marks	Final Term Result
35	67	1 (Pass)
12	75	0 (Fail)
16	89	1 (Pass)
45	56	1 (Pass)
10	90	0 (Fail)

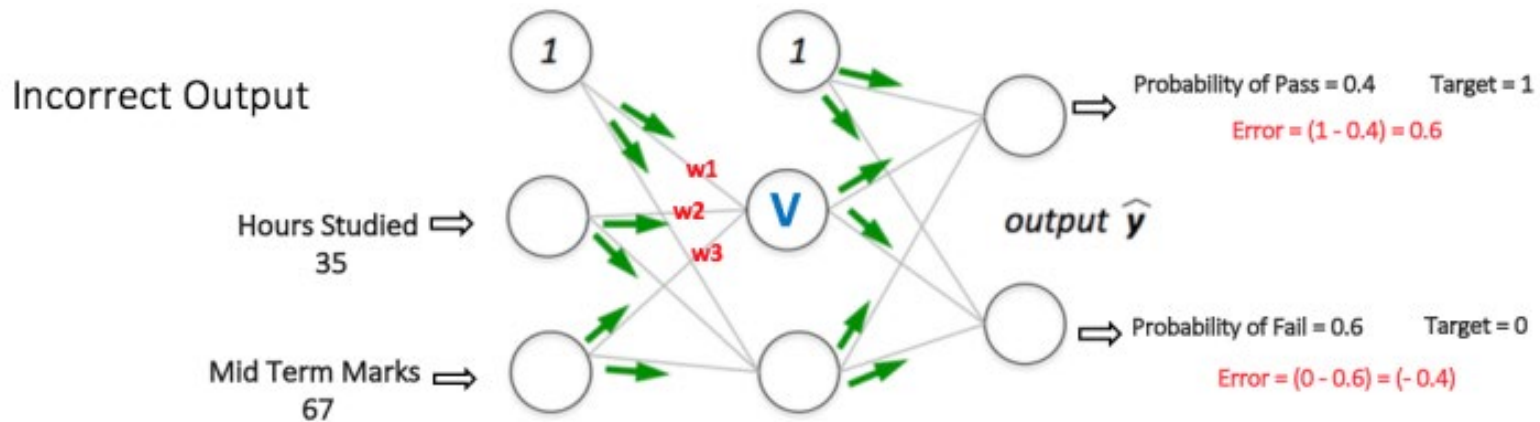
- The two input columns show the number of hours the student has studied and the mid term marks obtained by the student. The Final Result column can have two values 1 or 0 indicating whether the student passed in the final term.

<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>



Backprop Example (2)

- Step 1: forward propagation step in a multi layer perceptron



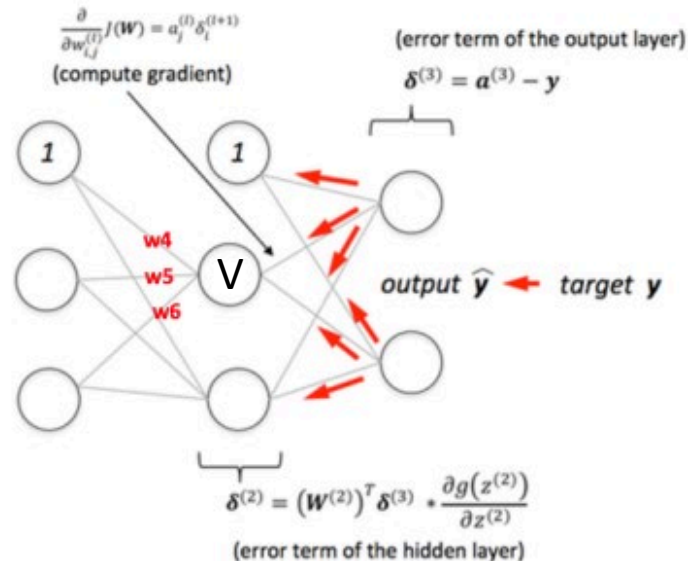
In this example the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively)

<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>

Backprop Example (3)

- We calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation (we ignore the mathematical equations in the figure for now).
- This leads to new weights w_4 , w_5 and w_6 for node V

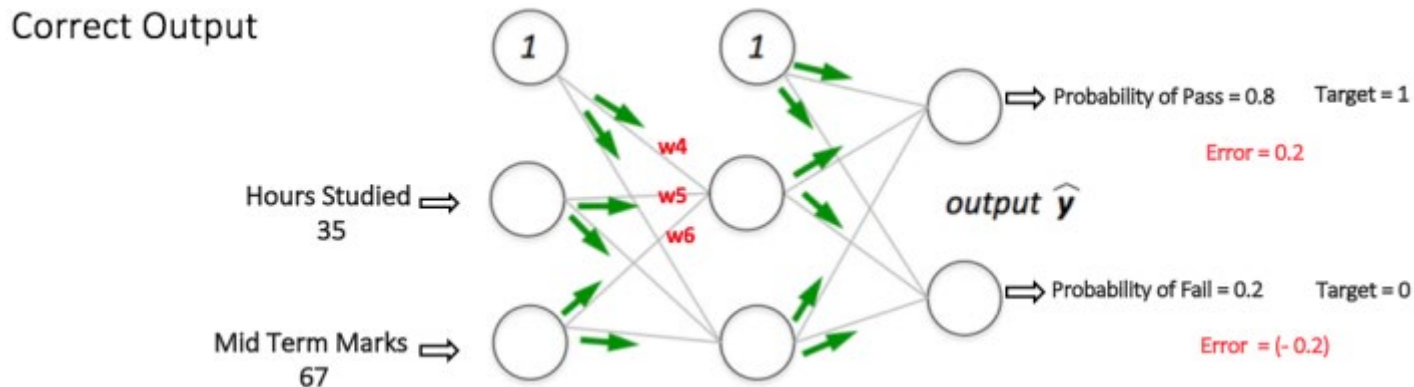
Backpropagation
+
Weights Adjusted



<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>

Backprop Example (4)

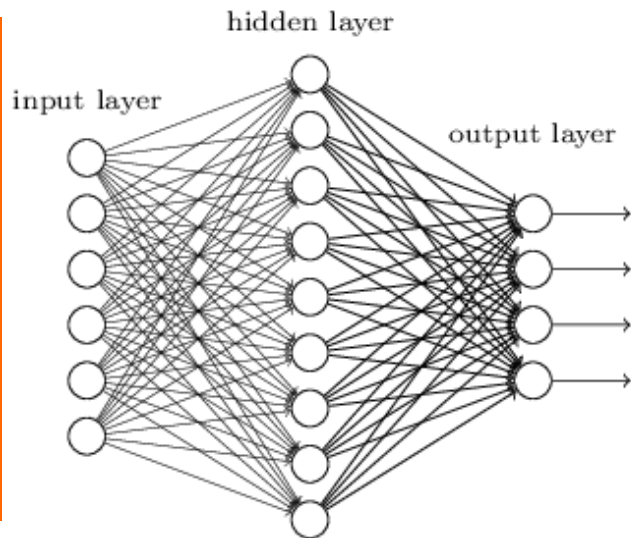
- Using the same input data for the network with the adjusted weights leads to much better results



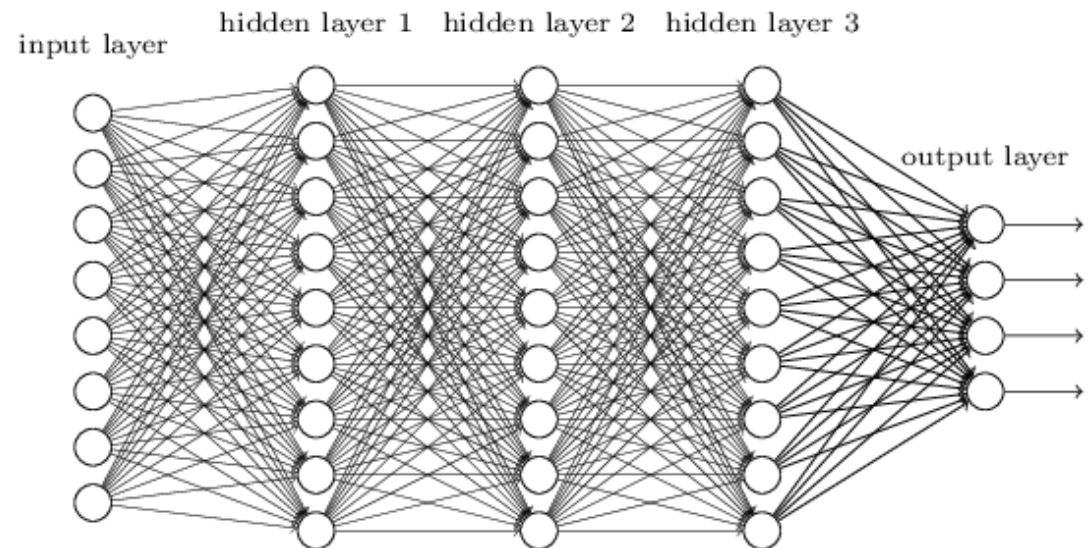
- This process is repeated with all other training data sets.
- When the network has *learnt* these examples, it can find the output probabilities for Pass and Fail also for new input data through the forward propagation step and.

Deep Neural Networks

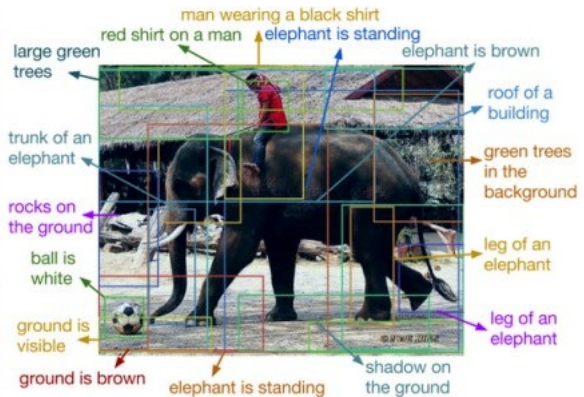
"Non-deep" feedforward neural network



Deep neural network



Applications of Deep Learners

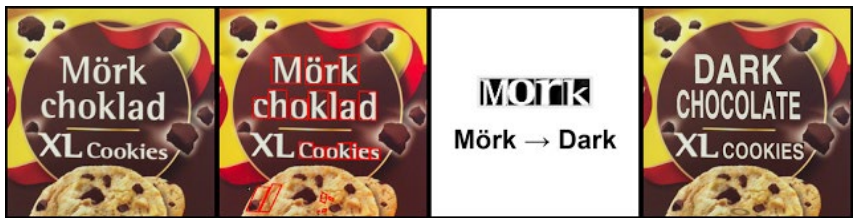


Describing photos

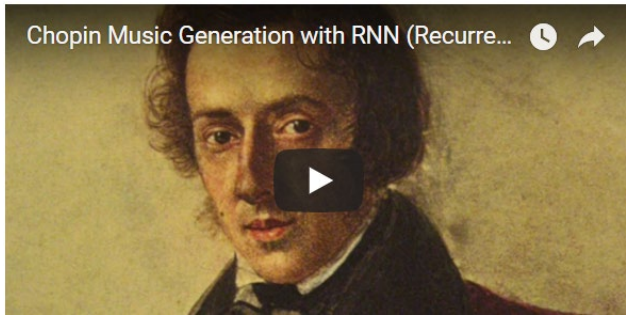


Colorado National Park, 1941 Textile Mill, June 1937 Berry Field, June 1909 Hamilton, 1936

Picture coloring



Translation



Musik composition



Self-driving Cars

© <http://www.yaronhadad.com/deep-learning-most-amazing-applications/> [28.02.2018]



Neural Networks vs. Decision Trees/Rules

- Classification with Neural Networks is very good
- Decisions with neural networks are not comprehensible
- Decision Trees and Rules are often more trustworthy
 - ◆ Decisions with trees and rules are comprehensible and explainable
 - ◆ One can see which rules are applied to make a decision
- In applications, in which trust in the decision or explainability is important, people prefer decision trees or rules