Artificial Neural Networks
&
Backpropagation

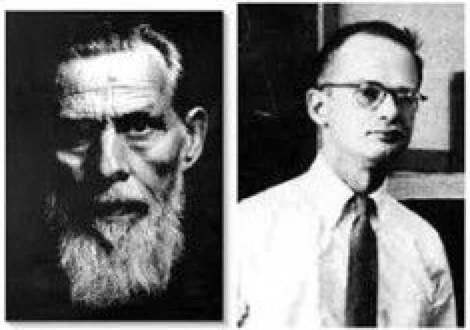
Deep Learning - Raúl Garreta - 2016

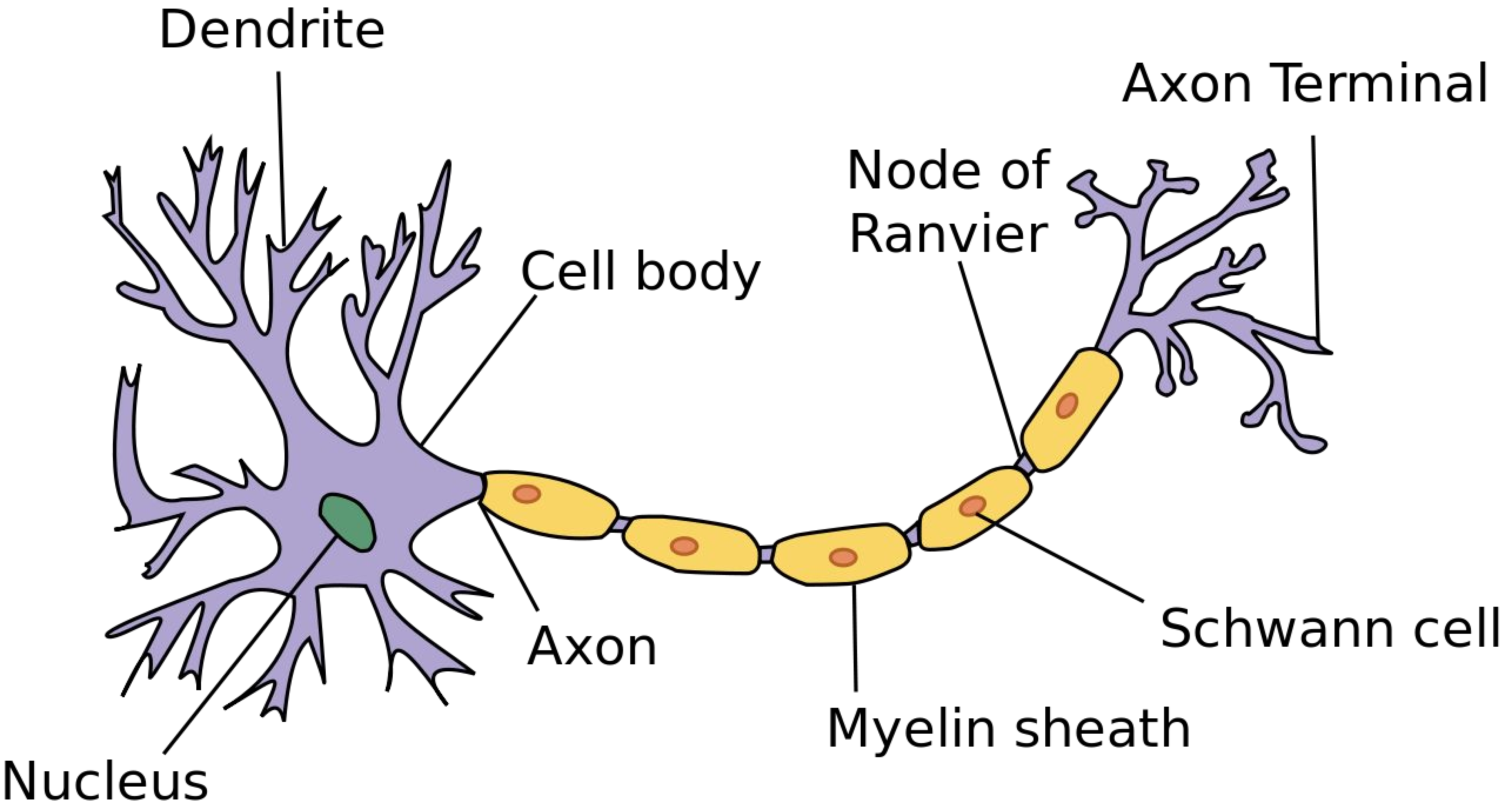
Contents

- Brief History
- Biological Neuron and Neural Networks
- Artificial Neurons
- Artificial Neural Networks (ANN)
- Machine Learning and ANNs
    - Perceptron training rule
    - Delta rule and gradient descent
    - Stochastic gradient descent (SGD)
    - Backpropagation
- Why Deep Learning?
- Advantages & Disadvantages
- Tips and Tricks

Brief History

- End of **19th century**
  - existence of nerve cells and their interconnection in functional structures was widely accepted.
- End of **1930**, nerve fibers were known to
  - conduct electrical impulses
  - excitation and inhibition of individual cells was demonstrated.
- **1943 Warren McCulloch and Walter Pitts**
  - First computational model of a neural network.
  - The logical model of activation of neurons.
  - The neurons are interconnected in networks to build higher complexity structures.
- **1960-1980** new research: continuous activations, gradient descent, backpropagation
- **2010s** returned again



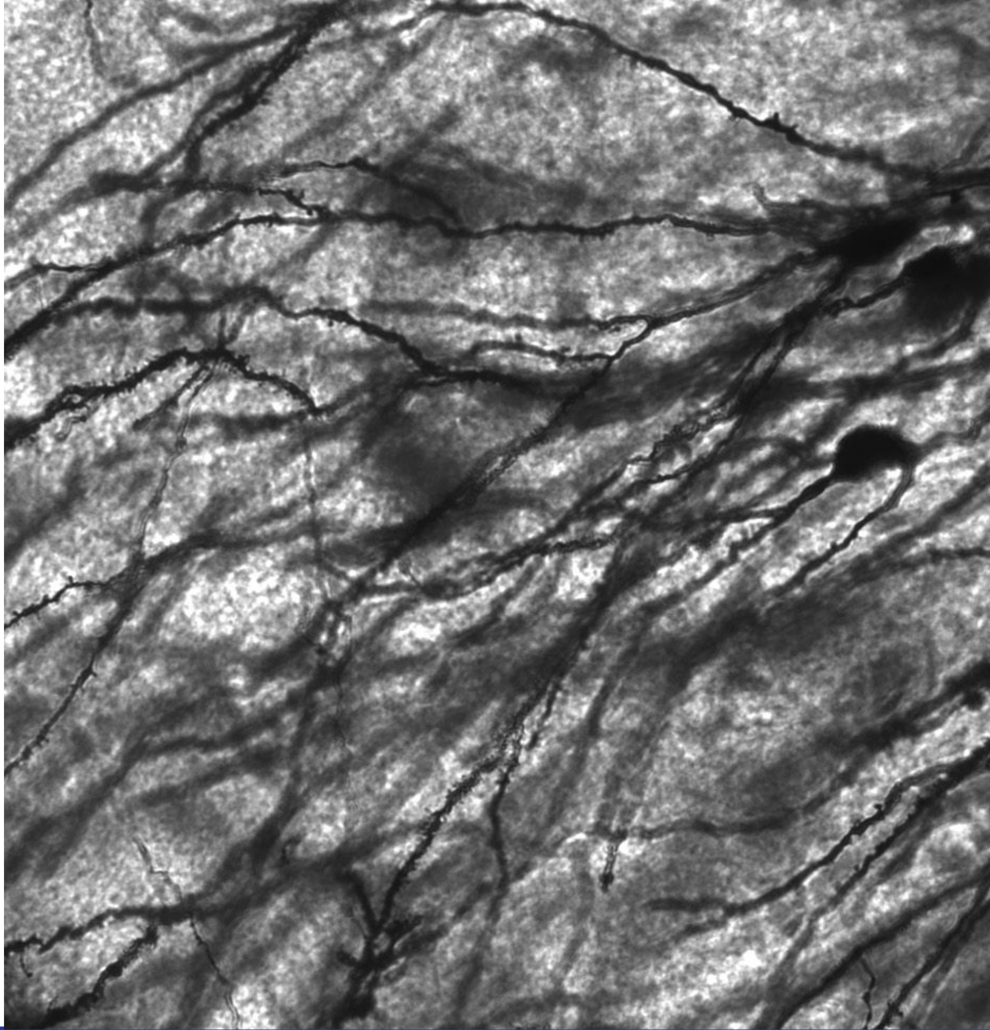

Dendrite
Axon Terminal
Node of Ranvier
Cell body
Axon
Schwann cell
Myelin sheath
Nucleus

Main Assumptions

- The nervous system is a **network of neurons**.
- Each neuron has a **soma** and an **axon**, **synapses** are always between the axon of one neuron and the soma of another.
- Synaptic signals may be **excitatory** or **inhibitory**. If the net excitation received by a neuron over a short period of time is large enough (some threshold), the neuron generates a brief pulse called an **action potential**, which originates at the soma and propagates rapidly along the axon, activating synapses onto other neurons as it goes.
- At any instant the neuron has some **threshold**, which excitation must exceed to initiate an impulse.

Biological Neuron Facts

- The human brain has ~ 10^{11} neurons.
- Each neuron is connected with ~ 10^4 neurons.
- The fastest neuron activation times ~ 10^{-3} seconds, (quite slow compared with computer times 10^{-10}).



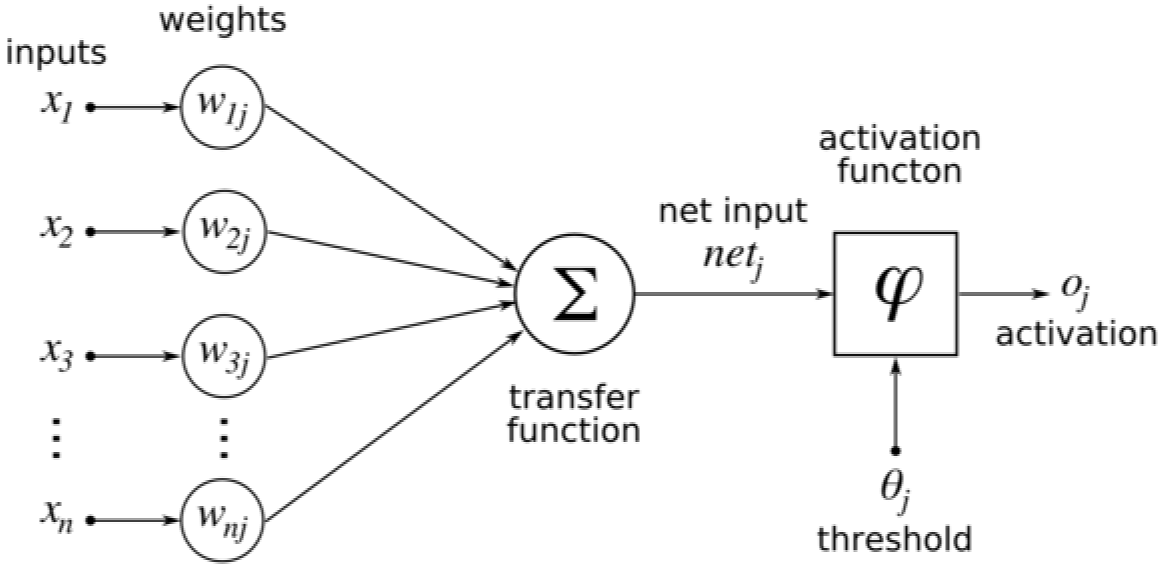
Intuition for Artificial Neural Networks

- Humans are able to make complex decisions surprisingly quickly.
  - For example, it is estimated that a person can recognize the face of his mother in around 10^{-1} seconds.
- Dividing by the average activation time of a neuron:
  - At most a few **hundred steps or layers** of neurons to do all the processing.
- From this observation, we can conclude that the biological neural system must have a **high parallel processing** and **distributed representations**.

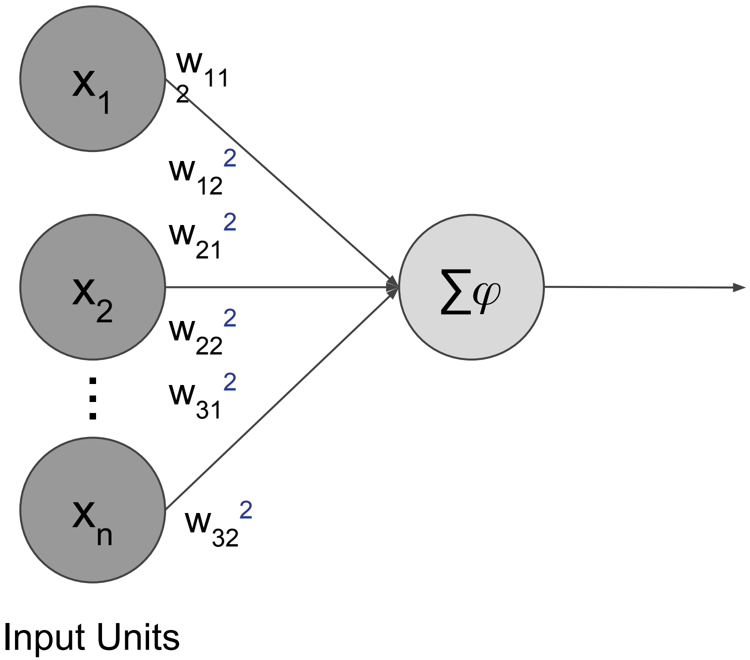
Artificial Neurons

- Were conceived (initially) as a mathematical structure to model biological neurons.
- The usual transformation in a AN takes the inputs and performs a weighted sum of the inputs that then is passed through a nonlinear function commonly known as the activation function:

$$o_j = \varphi\Big(\sum_{i=0}^{n} w_{ji} x_i \Big)$$



weights
inputs
x_1
w_{1j}
x_2
w_{2j}
x_3
w_{3j}
x_n
w_{nj}
Σ
transfer
function
net input
net_j
activation
functon
φ
o_j
activation
θ_j
threshold

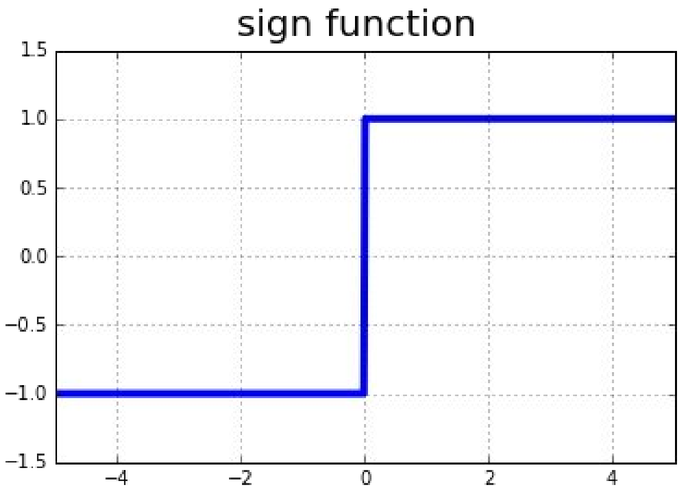

x_1
w_{11}
2
w_{12}^{2}
w_{21}^{2}
x_2
w_{22}^{2}
w_{31}^{2}
x_n
w_{32}^{2}
$\sum\varphi$
Input Units

Activation Functions Characteristics

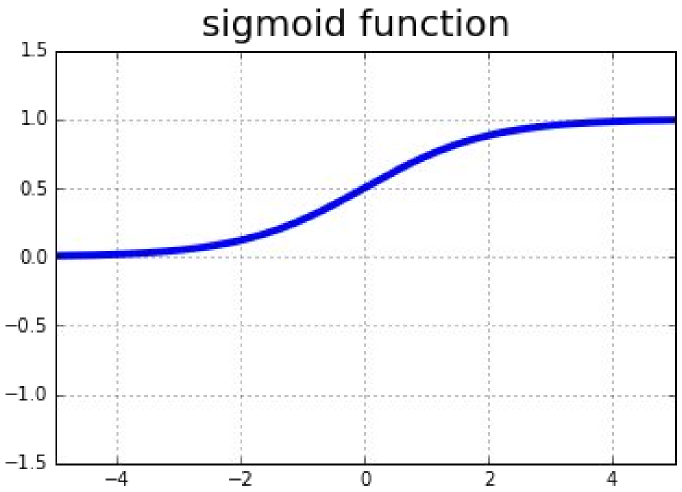
- **Monotonically Increasing** (the magnitude of the output increases as the magnitude of the input increases)
- **Continuous** (roughly speaking, small changes in the input produce small changes in the output).
- **Differentiable** (can calculate its derivative efficiently).
- **Bounded** (a function that returns an output that can be bounded, like squashing all the input in a founded range)

Activation Functions

$$sign(x) = \begin{cases} 1 : x > 0 \\ -1 : otherwise \end{cases}$$



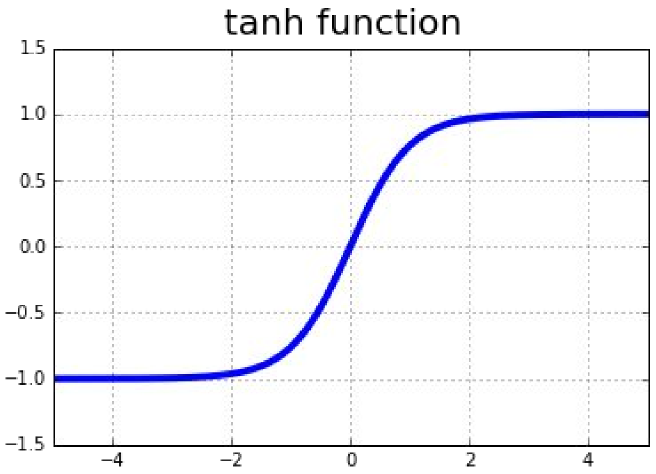
$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

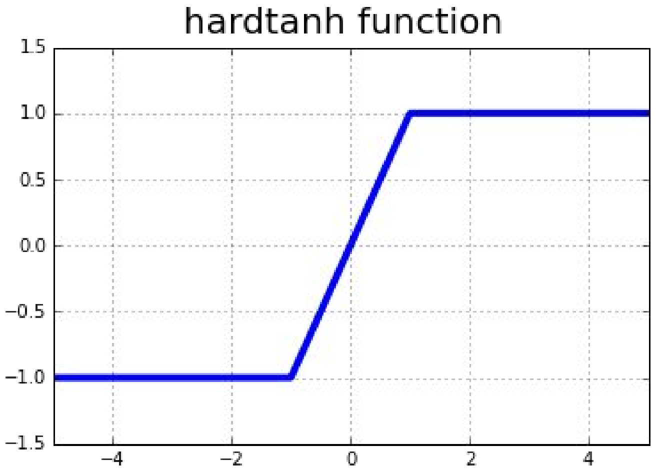


Activation Functions

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

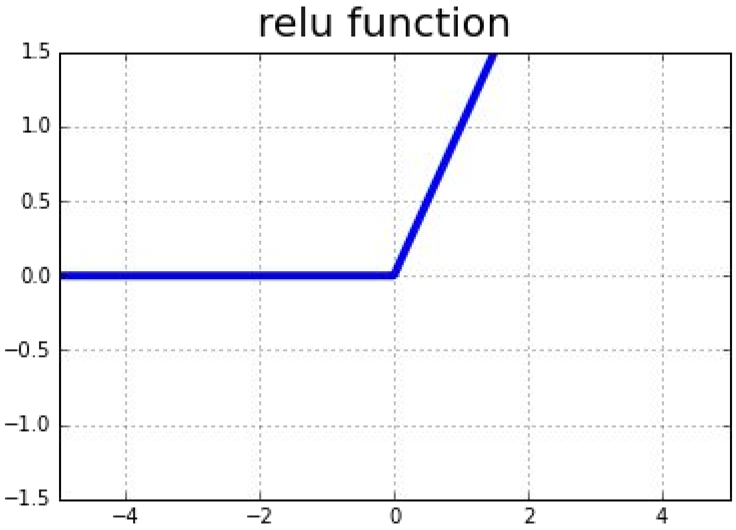
$$hardtanh(x) = \begin{cases} -1 : x < -1 \\ x : -1 <= x <= 1 \\ 1 : x > 1 \end{cases}$$





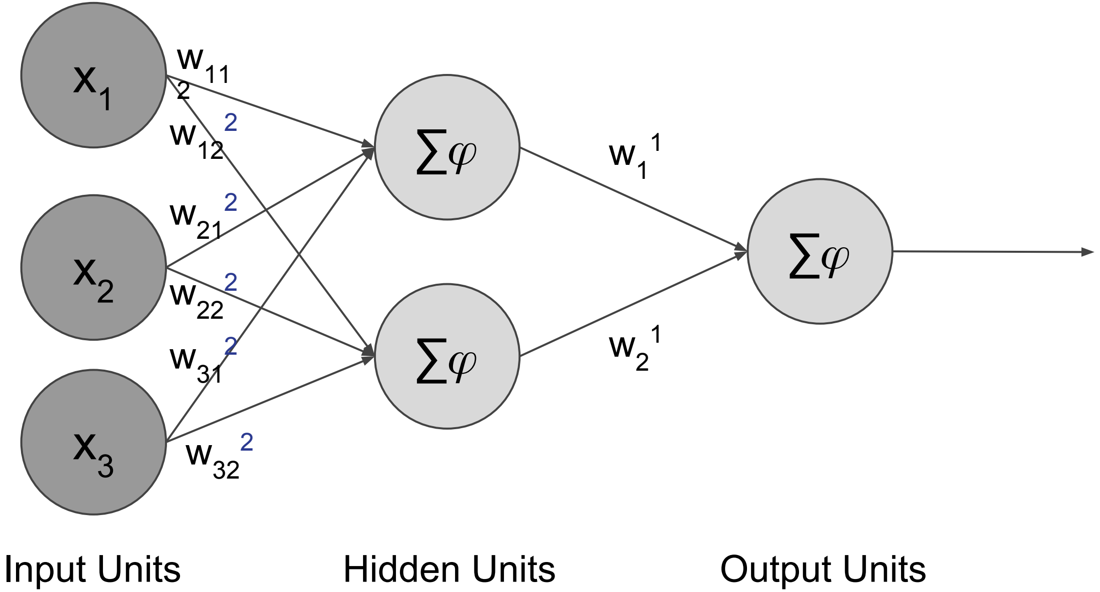
Activation Functions

$$relu(x) = max(x, 0)$$



Artificial Neural Networks (ANN)

- Usually will be interested in **Networks of Artificial Neurons**.
- When combining them (connecting outputs of neurons to inputs of others) we can represent a huge variety of functions. Eg: **Feedforward Architecture**:



Machine Learning and Neural Networks

- Technically we haven't talked anything about machine learning :)
- The interesting part comes when you can **dynamically change, adapt and improve the connections** between neurons to create the desired output given a particular input.
- That's when **machine learning** comes into the show.

Learning weights of a single neuron

1. Begin with an **initial set of weights** (eg: start with random weights).
2. Iteratively **input an example** into the unit and **adjust appropriately** whenever the output is different to the expected result.
3. **Repeat** step 2 as many times as necessary until the outputs for every training example are correct.

- Different algorithms mainly differ in how they adjust the weights to correct the output of the neuron:

$$w \leftarrow w + \Delta w$$

- The most popular:
    - **Perceptron Rule**
    - Delta Rule, Gradient Descent, **Stochastic Gradient Descent** (SGD).

Perceptron Training Rule

- Simple rule to adjust the weights:

$$\Delta w_i = \alpha(t - o)x_i$$

- Where dw_i is the adjust made to weight w_i when the input is x_i with target output t and obtained output is o.
- α is the **learning rate**, moderate the step of the updates.
- Converges in a finite number of iterations to a set of weights that make the unit correctly classify all the training examples, provided that they are linearly separable.
- **It may not converge** when the training examples are not linearly separable.

Gradient Descent

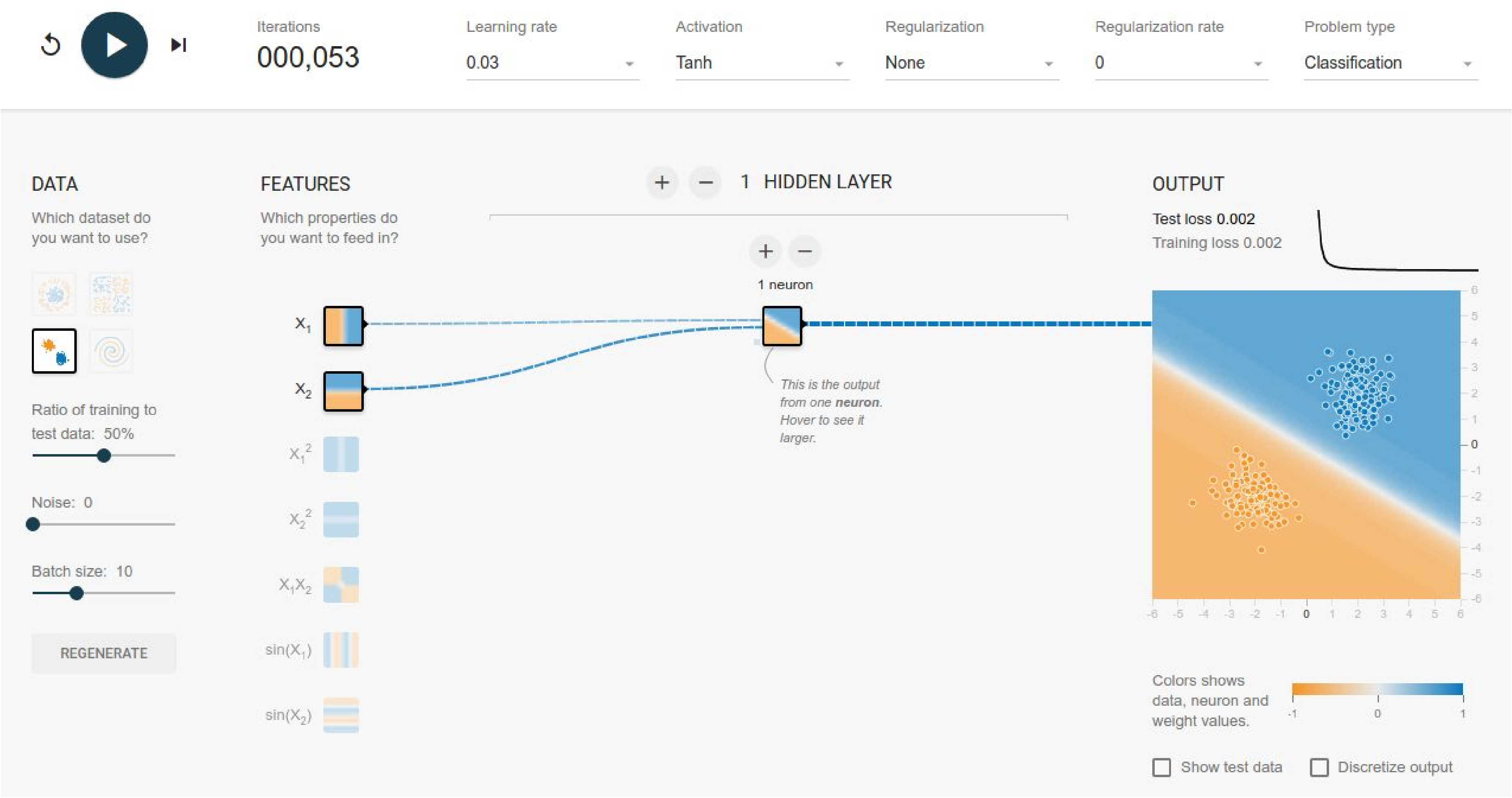
- This rule always converges to the best possible approximation to the desired output.
- Minimize squared error:

$$E(w) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- Use the derivative to find the steepest descent along the error surface towards the minimum:

$$\Delta w = -\alpha \nabla E(w)$$

- ∇***E(w)*** is a vector whose components are the partial derivatives of ***E*** respect to each of the vector ***w*** components.



Example with Tensorflow Playground

Learning weights of a Network: Backpropagation

- **Generalization of the gradient descent** to learn weights for **multilayer neural networks**.
- Adjusts the weights in a ANN to minimize the error between the obtained output and the desired output of the network when feeding the network with its inputs.

Main Differences

- With **multiple outputs** the minimization will be calculated by taking the sum of the errors of all the outputs of the network.
- We have to **adjust weights of all the units** in the network (it's a much larger search space).
- The error surface can have **multiple local minima**, it's not guaranteed that the algorithm converges to the global minima.
- Despite these difficulties, backpropagation obtains very good results in many practical applications.

Backpropagation Algorithm

- For each example <x, t> in training set:
  - Propagate the input forward through the network:
    - 1. Input the instance x to the network and compute the output of every unit u in the network.
  - Propagate the error back through the network:
    - 2. For each output unit k, calculate the error term δ_k:

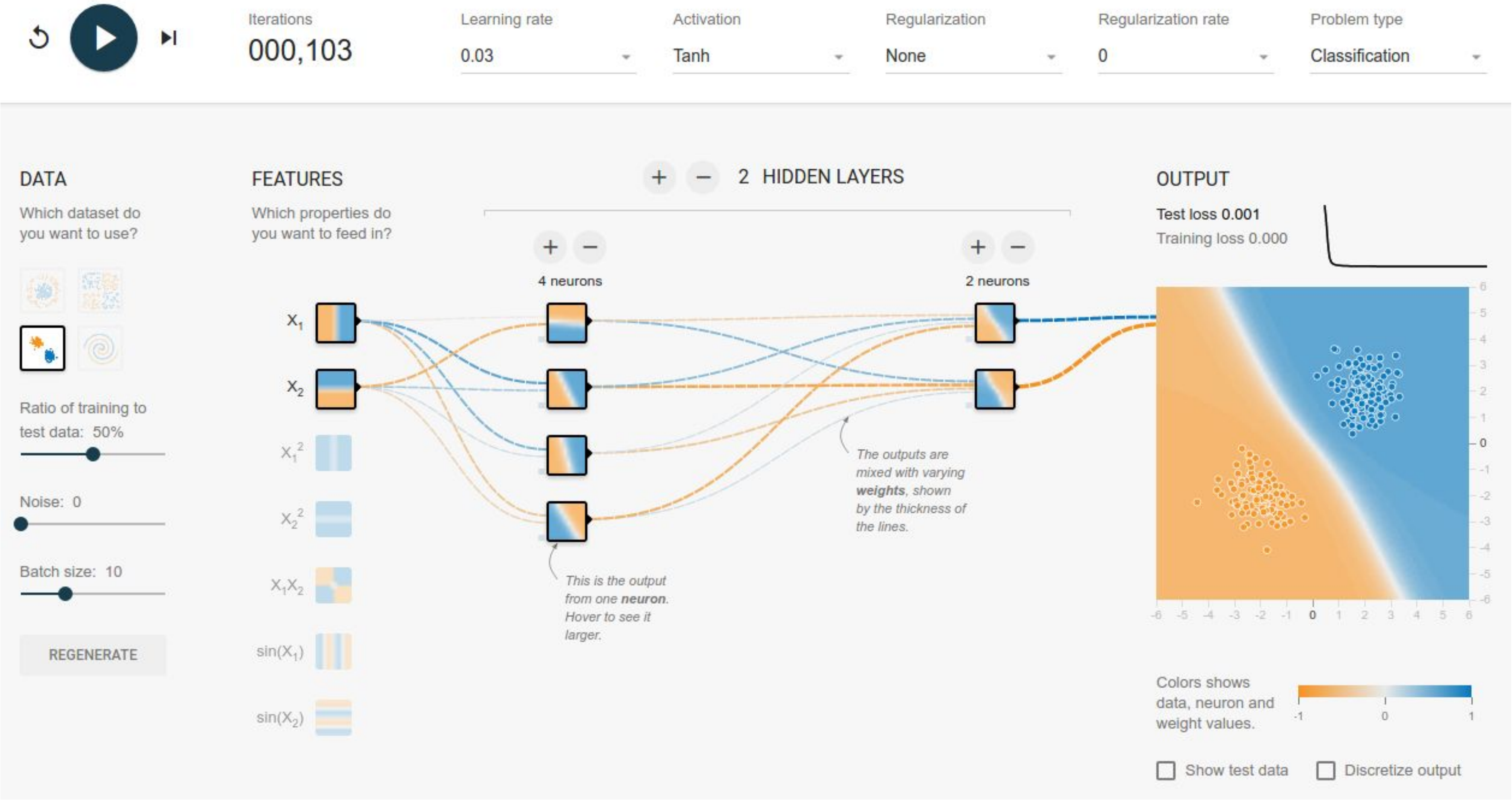
$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

    - 3. For each hidden unit h, calculate the error term δ_h:

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} (w_{kh}\delta_k)$$

    - 4. Update unit weights w_{ji}:

$$w_{ji} \leftarrow w_{ji} + \alpha\delta_j x_{ji}$$



Example with TensrofFlow Playground

Why Deep Learning?

- **New research / techniques**
  - Both academia and industry
- **Large amount of layers** (deep networks)
  - Intuition from biology: time from eye signal to response / activation time =~ 10.
- **Automatic feature extraction** (lower layers)
  - Training features separately w/ unsupervised techniques, eg: word2vec.
- **New architectures**
  - Convolutional networks, Recurrent neural networks, Autoencoders
- **Massive amounts of data**.
  - Easy to capture, easy to store.
- **Tools**
  - **Software**: frameworks with calculations that reduce error propagation between layers. Analytical gradient instead of numerical. Parallelization.
  - **Hardware**: Moore's law, parallelization with GPUs, specialized HW (eg: google chip).

When Deep Learning is a good option?

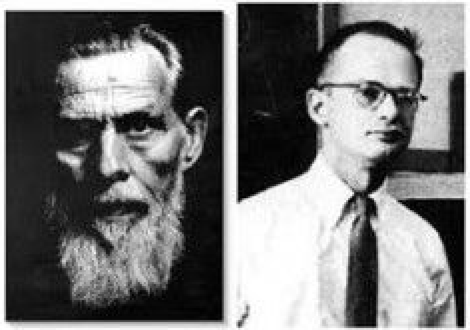
- Instances are represented by **large amounts of attributes**.
- The target function:
  - Continuous Real
  - Discrete
  - Vector
- Training **examples can contain errors**:
  - Neural network models are robuts to errors.
- **Long training times** are acceptable.
- **Fast prediction times**.
- Humans don't need to understand the learned model.

Difficulties with Deep Learning

- Huge search space
  - Large amount of parameters to adjust.
- Long training times.
- Large amount of training examples needed.
- Large amount of "hyperparameters" to select:
  - Learning rate
  - Architecture
  - Number of neurons
- Termination criteria

Tips & Tricks

- **Learning rate**:
  - Start with small learning rate. Avoid large learning rates that make the model to overshoot the minima.
  - Decrease learning rate over time.
- Use **momentum** term.
- Use **stochastic** gradient descent (SGD) instead of standard gradient descent.
- **Weight initialization**:
  - Random
  - Train multiple networks initialized with different weights.
- **Regularization**
- **Termination criterion**:
  - iterating the training until training error falls below a certain threshold is not a good idea: **overfitting**. Use a separate validation set.



Warren McCulloch - Walter Pitts

Geoffrey Hinton
U. Toronto & Google

Yann LeCun
AT&T - NYU - Facebook

Yoshua Bengio
MIT - AT&T - U. Montreal

Tomas Mikolov
U. Brno - Google - Facebook

Ilya Sutskever
U. Stanford - Google - OpenAI

Richard Socher
U. Stanford - Salesforce

Ronan Collobert
NEC - Facebook