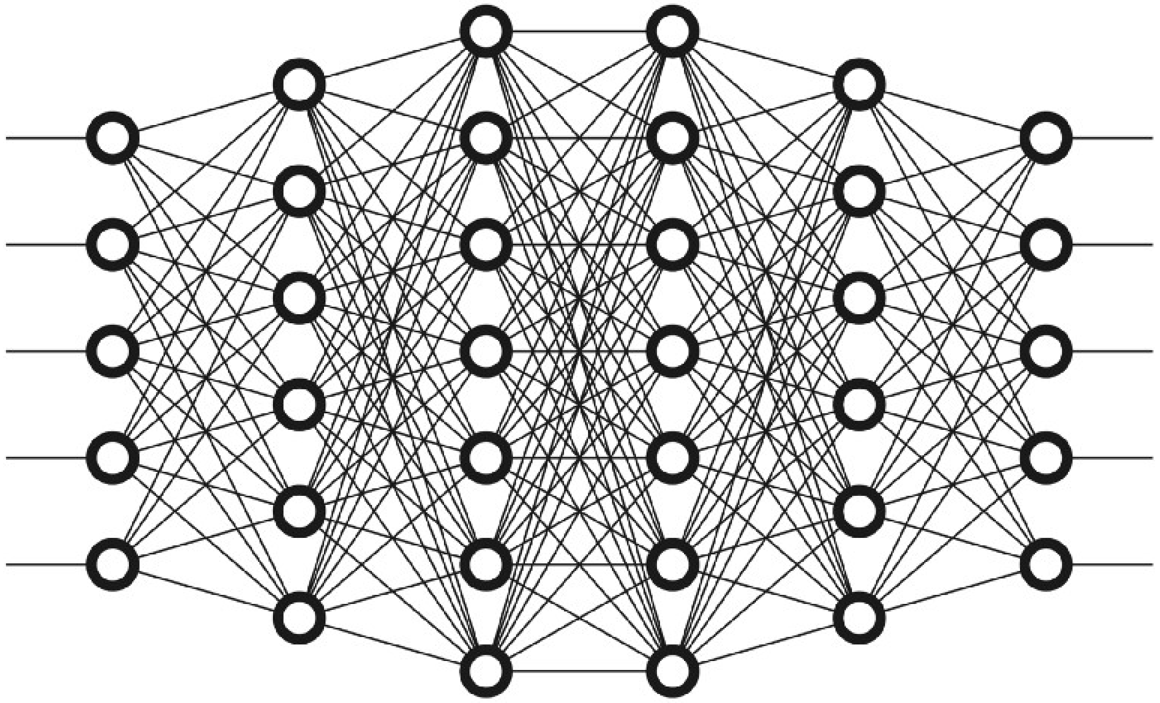
Neural networks



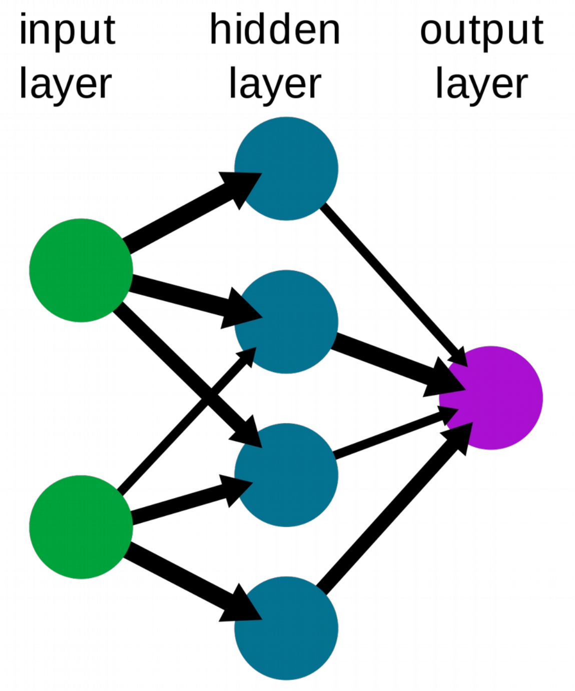
by Maurycy Borkowski

What is inside?

Neural network is build of layers of neurons connected with weighted edges. Each neuron and weight of edge is just a number (usually between 0 and 1) .

We can group layers into: input, hidden, output.

A simple neural network



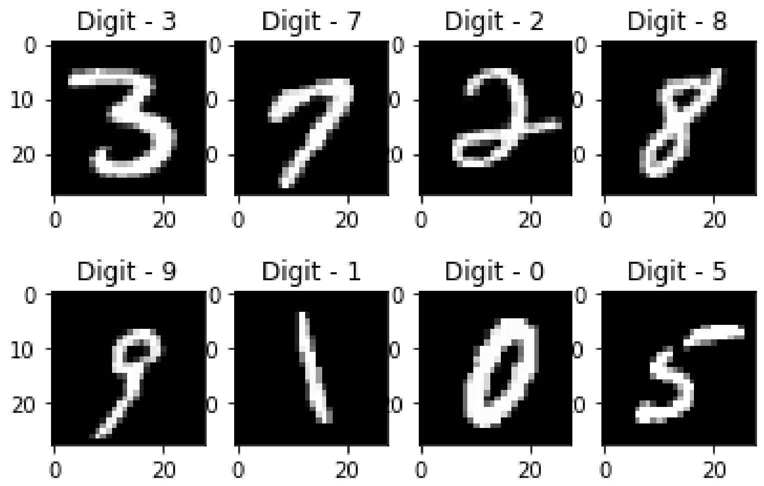
What is inside? example

To understand a little bit what is going on here let's go by example. As an ilustration we will take handwritten numbers recognition network (it’s like *Hello World!* In machine learning):

Our input data is black and white picture (i.e 28x28 = 784 pixels) so we can just set input layer neurons values to grayscale number of pixel.

For now let's leave the hidden layers discussion.

The output layer size is 10, each neuron corresponds to a number from 0 to 9. The goal product is only one highlighted neuron.



Network rewarding

We need to be able to check whether our network is working correctly or not. That’s why why need cost function.

Cost function is just sum of (squares of) differences between output and goal output. So the smaller difference, the better network.

We can think about our structure also as a function, here is nice comparison:

	Network	Cost function
input	784 pixels	edges weights
output	1 number (guess)	1 number (cost)
parameters	edges weights	784 pixels

Learning network

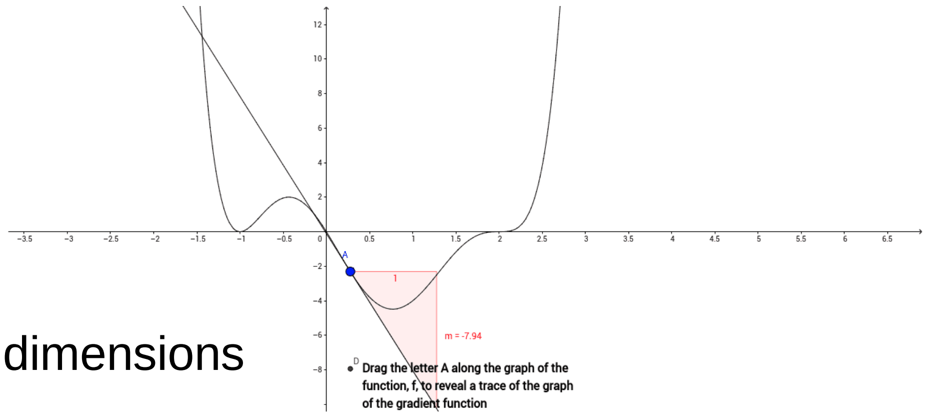
Now our goal is of course to minimize cost function.

We can interpret this problem as problem of minimizing a function in <number of all weights> dimension space…

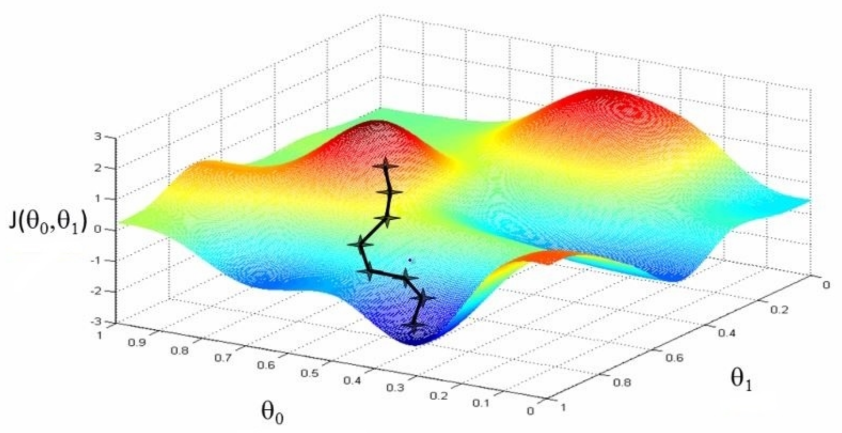
So what we do need to do is:

Algorithm:

- compete gradient of cost function: $\nabla\, cost\ function$
- do small step in $-\nabla\, cost\ function$
- repeat



2 dimensions



3 dimensions

Updating. Backtracking.

Now we know what to do but how do we should do algorithm from previous slide?

When we are processing neuron N we know should we increase or decrease value of it.

How to modify it?

Adapt *bias* (value on each neuron to operate of linear combination of edges*neurons)

Fit weight of edge in proportion neuron neighbor value

Revise neighbor amount in proportion of edge weight

Going like that from the output layer ending in input layer.

It turns out that vector average of changes (balance) for each weight (bias, neuron) for some training data is $-\nabla \mathit{cost\ function}$

Naturally more data = detailed value of gradient.

Summary

That is how neural network works in a nutshell :) I know that I skipped implementation / math details but what I wanted is give you some intuition of machine learning. I hope that know you don't think of it as a black magic.

Applications

- Automatic speech recognition
- Image recognition
- Natural language processing
- Financial fraud detection
- Military

Successes of machine learning

- Playing games Chess (1996), Go (2016),Poker (2017)
- Automatic speech recognition (Microsoft/IBM, Switchboard Corpora, about 2016)
- Knowledge competitions (Watson, Jeopardy)
- Image recognition (ImageNet, 1mln pictures):
  - 2010: about 28% mistakes
  - 2015 (4.94% , human: 5.1%)
  - Current record: 1.3% (Top5 accuracy, 2020)