Introduction to Neural Networks

By Eric Schles

Intro

Neural networks are the “gateway” technique into machine learning.

They were first invented in 1943!

Uses of neural networks

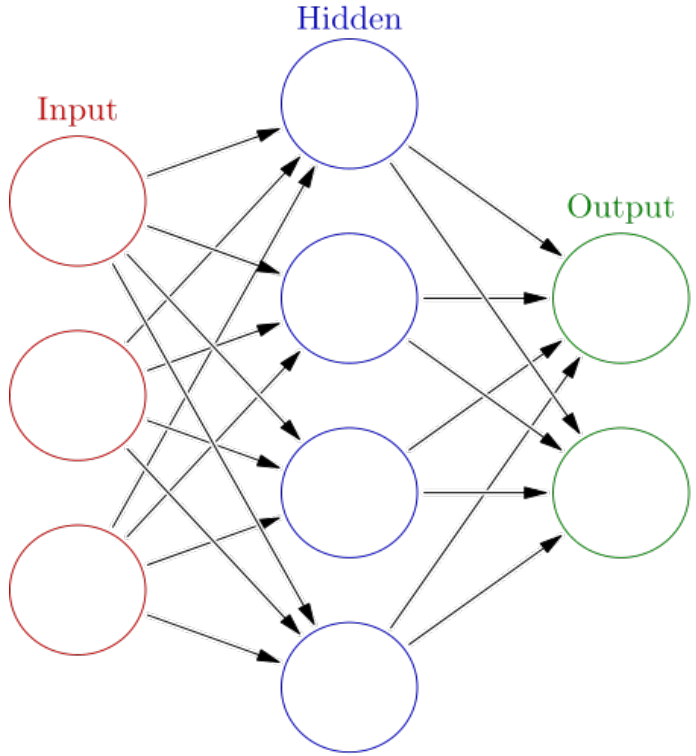
- function approximation:
  - time series
  - fitness approximation
- classification:
  - novelty detection, pattern recognition
- Data processing:
  - filtering
  - clustering

definition

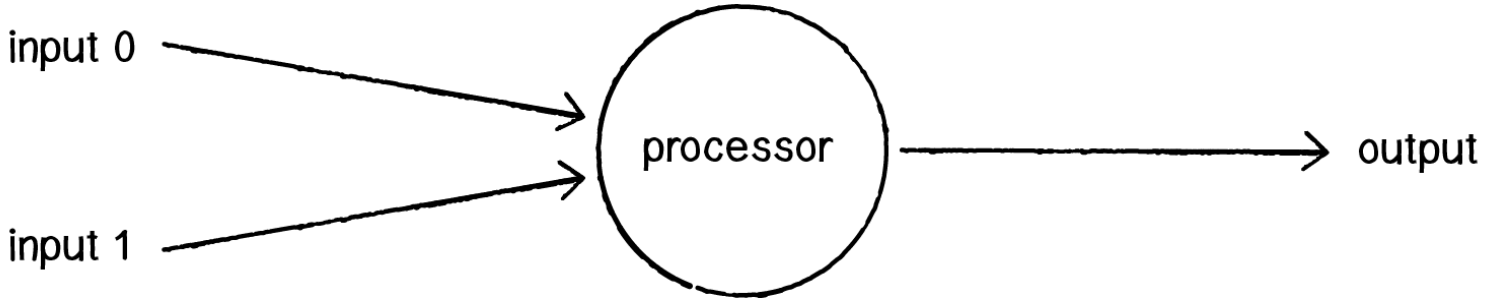
A neural network is a computer science model of the human brain.

Decomposition of Neural networks

1. The interconnection pattern between the different layers of neurons
2. The learning process for updating the weights of the interconnections
3. The activation function that converts a neuron's weighted input to its output activation.



Simple ANN: single perceptron



The perceptron takes in inputs and “applies” weights, then producing an output. These weights are randomly initialized. This is known as the feedforward step.

Algorithm

output[i] = weight[i] * input[i]

- output = sum

if Activate(sum) > 0 then first result, otherwise second result

Review of OOP

Classes

- https://docs.python.org/2/tutorial/classes.html
- http://learnpythonthehardway.org/book/ex40.html
- http://www.tutorialspoint.com/python/python_classes_objects.htm
- http://en.wikibooks.org/wiki/A_Beginner's_Python_Tutorial/Classes
- http://www.openbookproject.net/thinkcs/python/english2e/index.html

Intro to classes

https://github.com/EricSchles/neuralnet/blob/master/intro_to_oop.py

Intro to references

https://github.com/EricSchles/linkedlists

https://github.com/EricSchles/BinarySearchTree

In code

https://github.com/EricSchles/neuralnet/blob/master/first.py

Learning

I thought our models were supposed to be “learning”

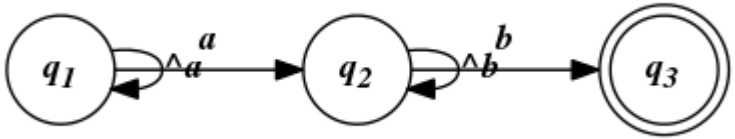
Enter backpropagation.

A “slightly” more complex ANN

We’ve seen a simple example of a neural network, one perceptron applying random weights.

However if we want to optimize our output we need to iteratively test our weights until we find the ones that are “good enough”

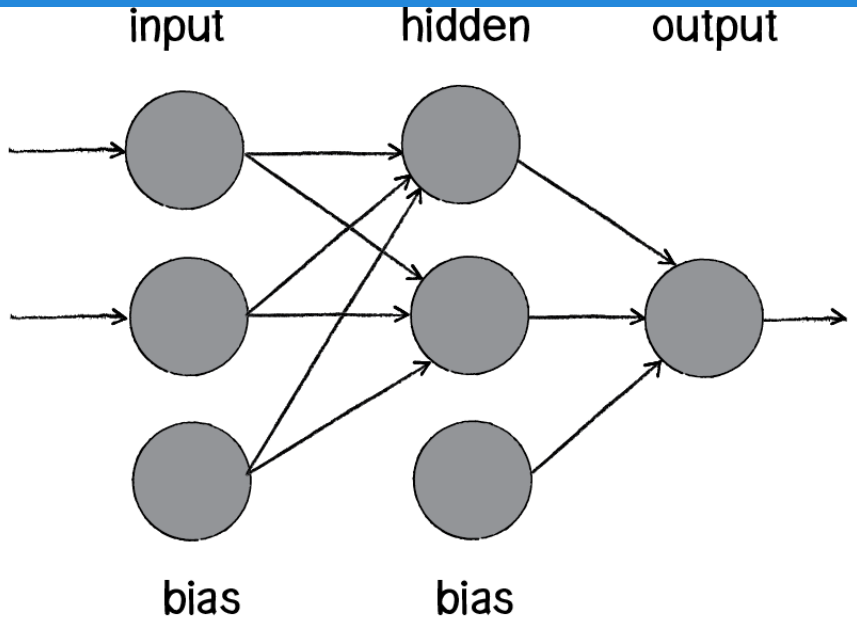
Enter backpropagation



Back propagation allows us to regenerate our weights. Thus if our model doesn't perform well the first time, we can still do a decent job.

Great theoretical intro: http://neuralnetworksanddeeplearning.com/chap2.html

The typical back propagation diagram



We consider the hidden nodes as the regenerated weights.

In code

Implementing xor

If none of that made any sense

Enter PyBrain

implementing xor in pybrain

https://github.com/EricSchles/neuralnet/blob/master/xor.py

Some examples

Image Recognition

Prediction

Character recognition

https://github.com/EricSchles/neuralnet/blob/master/character_recognition.py

Image Processing

- http://scikit-image.org/
  - http://scikit-image.org/docs/0.10.x/contents.html
- http://docs.scipy.org/doc/scipy/reference/tutorial/ndimage.html
- http://pythonvision.org/basic-tutorial
- http://stackoverflow.com/questions/16112161/pillow-2-0-0-tutorial
- http://docs.python-guide.org/en/latest/scenarios/imaging/

Image processing on my face

https://github.com/EricSchles/neuralnet/blob/master/cleaner.py

let’s see if we can learn on my face!

Danger - this example *may* not work

references

http://natureofcode.com/book/chapter-10-neural-networks/

https://github.com/shiffman/The-Nature-of-Code-Examples/