Introduction to Artificial Neural Networks (ANNs)

Jaya Nilakantan

Dawson College

jnilakantan@dawsoncollege.qc.ca

Recap

We've seen 3 Machine Learning algorithms so far:

- Linear regression: used to predict a value based on feature(s). Assumes a continuous linear relationship
- k-nearest neighbours: used to predict a classification based on feature(s).
  - fine tuning of *k* through thew *k-fold* cross-validation
- naive Bayes classifier: used to classify based on the assumption that past frequency of features accurately predict the future probability

What did these algorithms have in common?

- supervised learning algorithms: assumes labeled data
- pretty fast algorithms / optimization algorithms
- assumes some understanding of the data and the best algorithm
  - for example, you realize that a linear relationship exists between feature(s) and output

So why use neural networks to solve the same type of problems?

- neural networks can represent almost any function, as stated by the universal approximation theorem.
  - much much easier than deriving the complicated math for a multi-variate nonlinear relationship
  - think about how hard is is to predict the weather
- scale well for huge datasets
- can yield accuracy that is much better than the statistical/algorithmic machine learning models

What are artificial neural networks?

...computing systems vaguely inspired by the biological neural networks that constitute animal brains.
— *Wikipedia*

What are artificial neural networks?

...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.
— *Dr. Robert Hecht-Nielsen*

What are artificial neural networks?

...a massively parallel combination of simple processing unit which can acquire knowledge from environment through a learning process and store the knowledge in its connections.

— *Simon Haykin*

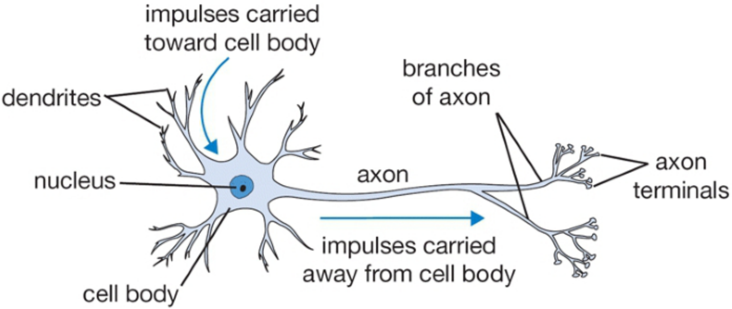
Reasons why neural computation was studied

- to understand how the brain works
  - computer simulation won't harm someone!
- to implement the same kind of parallel computation that neurons perform
  - very different from out typical sequential algorithms
  - should be good at things the brain is good at (e.g., vision, recognition)
  - should be bad at thing the brain is bad at (e.g., computation like 23 x 71)
- to solve problems using algorithms *inspired* by how the brain works
  - but not necessarily exactly how the brain would do it

Biological motivation

- the basic computational unit of the human brain is a *neuron*
- the human brain is composed of $\sim 10^{11}$ neurons
- they connect to each other through *synapses*
- each connection is *weighted* according to its importance

Biological neuron

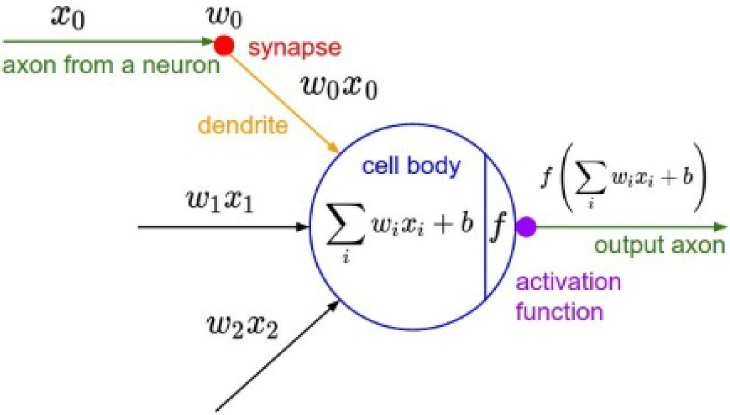


- signals from axons of previous neurons to dendrites
- spike of activity in the axon -> charge through the synapse into the next neuron
- generates the spike if enough charges have flowed through its dendrites
- synapses adapt => this is learning

How the brain works (Hinton's abridged version)

- each neuron receives inputs from 1 or more other neurons
- the effect of each input on the neuron is controlled by a synaptic weight
  - positive or negative
- the synaptic weight adapts -> this is learning

Artificial neuron - example: Linear neuron



- each input has a weight: the relative importance of that input vs the other
- sum of all the weighted inputs + bias -> activation function -> output through the axon to the next neurons

Artificial neuron - example: Binary threshold neuron

- sum of all the weighted inputs + bias
- if it exceeds a threshold=> output of 1, else output of 0
- think of it as logical combination of True and False signals

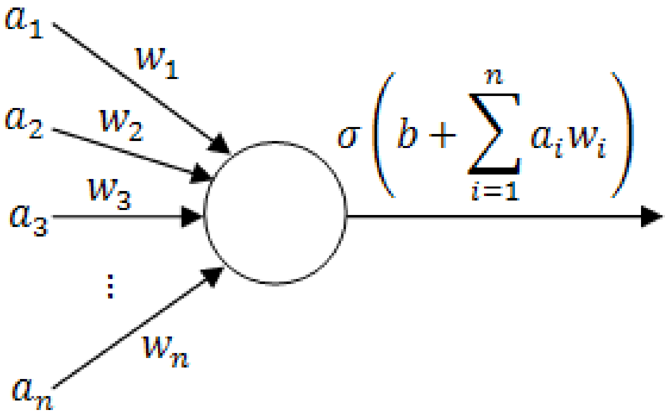
General idea behind ANNs

- the weights w are learnable
- they control the strength of the influence of one neuron on another
- weights can be positive ("excite" the next neuron, causing the sum to exceed the theshold so it fires) or negative ("inhibit" the next neuron from firing)
- activation function models the firing rate - frequency of spikes on the axon

ANN architecture - Feedforward

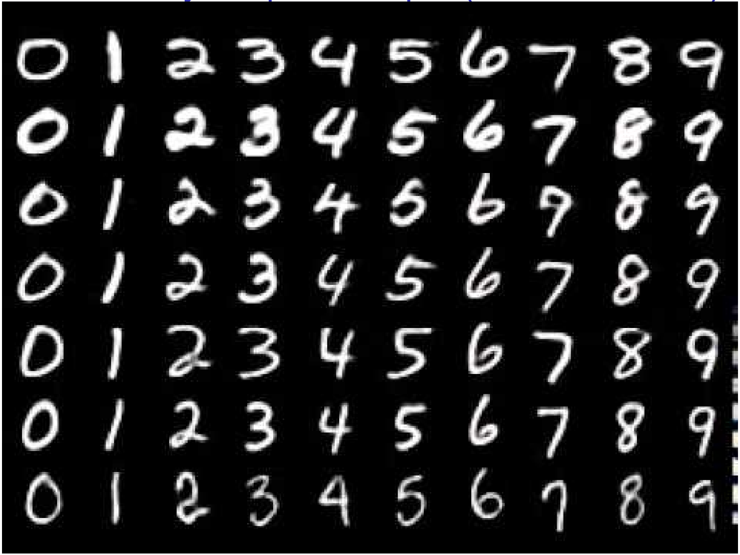
- Feedforward neurons only send information in one direction: the connections can never form a circle

Single layer Perceptron



- simplest feedforward ANN
- input (training/test data) feeds the output layer based on weights
- no computations done at input, so it is not counted as a layer

A very simple example (source: Hinton)



Consider the famous MNIST database of handwritten digits. It is often used for classification experiments.

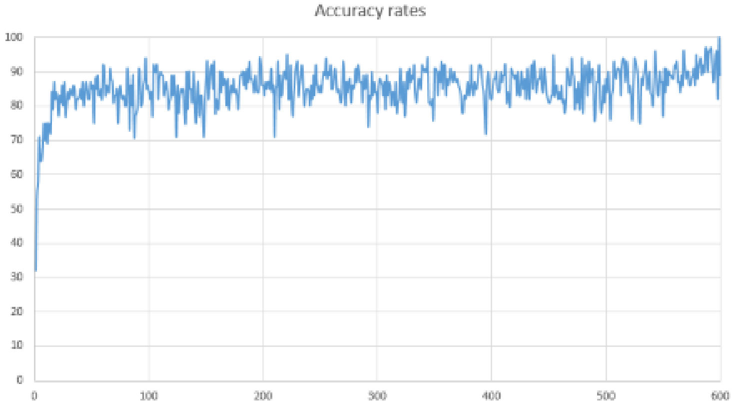
Solving with a Single layer perceptron

- each digit is 28x28 bitmap -> consider that there are 784 inputs, one for each pixel
- input value represents the pixel intensity
- the weight of each pixel is initially randomly chosen
  - the weights represent the “votes” that each pixel has regarding the output(s)
  - each pixel can vote for several different outputs
- the output layer are the 9 possible digits. In other words, the possible classifications.
  - the shape with the most votes wins

How does the learning happen?

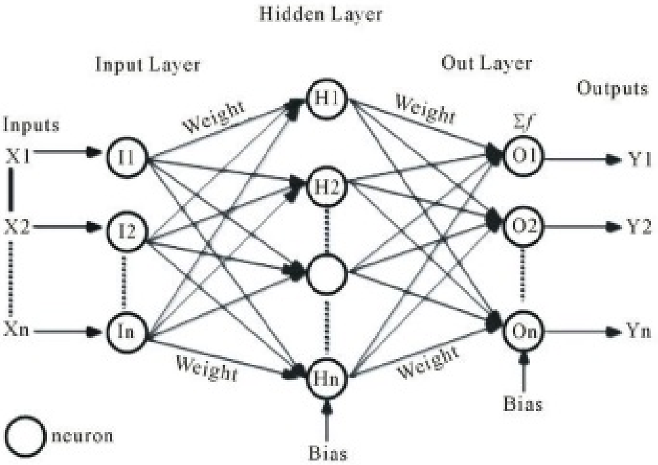
- after every training sample, check the guess with the actual shape
- for every correct guess, increment the weights from the active pixels to the correct class
- decrement the weights from the active pixels to any wrong guess

Results (simple PHP code)



- a one-layer network is equivalent to a template for each shap: the winner is the template with that highest overlap
- hand-written shapes have too much variance: there will always be a relatively high error rate with this approach

Multi-layer perceptron



- input layer -> training/test data
- hidden layer -> processing
- output layer -> result

Other architectures

- Convolutional NN
- Recurrent NN

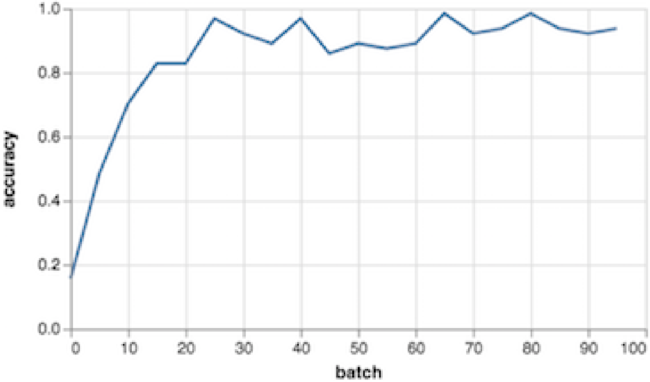
Introduction to Tensorflow

- Tensorflow is Google's open sourced Deep Learning library
- you can deploy on Google Cloud, other cloud providers, desktops, Android, browser
- main languages: implemented in C++, Python client, C API
  - Python has a high level API called Keras
- TensorFlow.js
  - requires Node.js, and NPM or Yarn (dependency management)

Using a Convolutional NN to recognize hadwritten digits

- follow the tutorial at TensorFlow.js

Results:



*

References

https://en.wikipedia.org/wiki/Universal_approximation_theorem
https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc
https://medium.com/all-of-us-are-belong-to-machines/the-gentlest-introduction-to-tensorflow-248dc871a224
https://core.ac.uk/download/pdf/82123892.pdf
http://www.cs.toronto.edu/~tijmen/csc321/lecture_notes.shtml
https://www.tensorflow.org/
https://js.tensorflow.org/tutorials/