Manchester Metropolitan University

Faculty of Science and Engineering

Alexander J. Johnson

Mathematics

Neural Networks with Python and TensorFlow

February 25, 2020

School of Computing, Mathematics and Digital Technology

Abstract

Artificial Intelligence (AI) still remains as one of the greatest challenges in scientific research to this date, but much progress in the field has been made using artificial neural networks. The design of artificial neural networks is loosely inspired by that of biological brains, and serves as an expansion of an earlier concept called the perceptron (Rosenblatt, 1958). By using multiple layers of these artificial neurons, we can form a highly connected system that is referred to as a neural network, these networks can then be trained on a large data set to predict the output with high accuracy.

The range applications for neural networks is wide: they can be used to classify data, predict future states of chaotic systems, apply stylisations to images, and control physical/physically-based system in real-time.

`TODO`: *Abstract*.

Declaration

With the exception of any statement to the contrary, all the material presented in this report is the result of my own efforts. In addition, no parts of this report are copied from other sources. I understand that any evidence of plagiarism and/or the use of unacknowledged third party materials will be dealt with as a serious matter.

Signed

[illegible]

Contents

Chapter 1

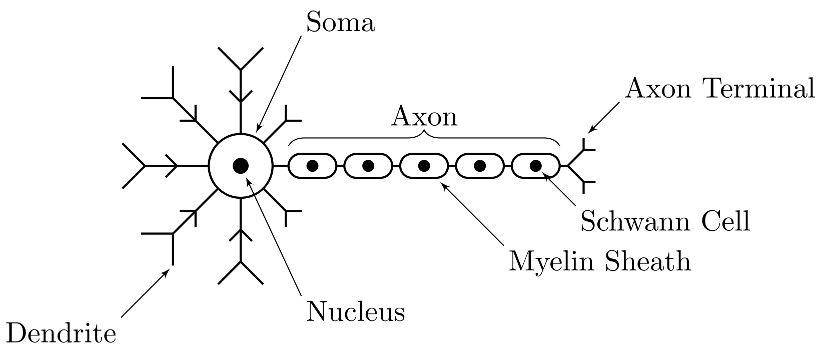
Introduction to Neural Networks

TODO: *Chapter: Introduction*

Biological Neurons

Biological neurons are electrically excitable cells that are found in almost all animals. These neurons can transmit and receive electrical signals to one another via synaptic connections, which maybe either excitatory or inhibitory. Any given neuron will be either active or inactive depending on whether or not its input exceeds a threshold.

Figure 1.1: Diagram of a biological neuron.



Signals are received by the neuron via connections to dendrites and soma. If the threshold is met, electrical signals are sent along the axon to the terminal, where it is connect to more neurons or to a controllable cell such as a neuromuscular junction.

TODO: *Section: Biological Neurons*

Artificial Intelligence

The idea of artificial beings capable of human intelligence can be traced back to mythical stories from ancient Greece. One such story was that of a mythical automaton called Talos, who circled an island's shores to protect it from pirates and other invaders.

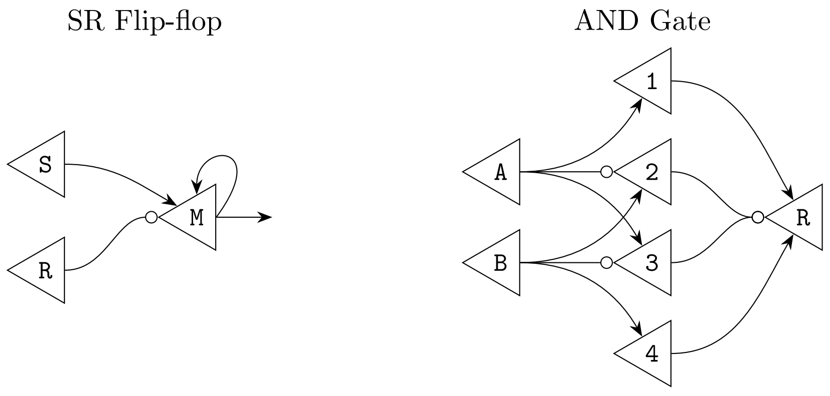
By the 19th century, other notions of artificial intelligence were explored by fiction in stories, such as Mary Shelley's "Frankenstein", and Karel Čapek's "R.U.R.". Some of the fictional writings of the 20th century further continued to explore the concept in novels such as Isaac Asimov's "I, Robot".

Academic research into artificial intelligence began around the 1940's, primarily due to findings in neurological research at the time. The first explorations of artificial neural networks was done by McCulloch and Pitts, 1943, who investigated how simple logic functions might be performed by idealised networks. The neurons within these networks operated using some basic logic rules, applied to a discrete time system which, can be summarised using the expression

$$N(t) = (E_1(t-1) \vee E_2(t-1) \vee \dots) \wedge \neg(I_1(t-1) \vee I_2(t-1) \vee \dots),$$

where $N(t)$ is the state of a neuron at time t, and $E_i(t-1)$ and $I_i(t-1)$ are the states of the excitatory and inhibitory connections from the previous time step respectively. The result is such that the neuron will only be active if at least one excitatory connection is active and all inhibitory connections are inactive. The versatility of this definition is demonstrated in the following examples.

Figure 1.2: Two common logic circuits using McCullochs neurons.



$N_M(t+1) = (N_S(t) \vee N_M(t)) \wedge \neg N_R(t)$ $\qquad$ $N_R(t+2) = N_A(t) \vee N_B(t)$

Where the arrows and circles indicate excitatory and inhibitory connections respectively. While this model provided insight into the mechanisms by which neurons operate, the structure was static, and incapable of learning.

McCulloch's work was later cited by psychologist Hebb, 1949, who proposed that the structure of biological neural networks was dynamic, and that frequently repeated stimuli caused gradual development. His theory was supported by research conducted by himself and others that showed that intelligence-test performance in mature patients was often unaffected by brain operations that would have prevented development in younger patients. This hypothesis became known as Hebbian learning.

`TODO`: *Find a copy of "Simulation of self-organizing systems by digital computer" - B Farley, W Clark*

`TODO`: *Section: Artificial Intelligence*

Perceptrons

The idea of the perceptron was originally conceived by Rosenblatt, 1958, to represent a simplified model of intelligent systems free from particularities of biological organisms, whilst maintaining some of their fundamental properties.

The perceptron was built as a dedicated machine that consisted of a number of photovoltaic, analogous to a retina, that feed into an "association area". This association area contains a number of cells that each calculate a weighted sum of the receptor values and output a signal if it exceeds a threshold. These value weights were implemented using variable resistance wires that the perceptron could adjust automatically. The outputs from the association area are then connected to response cells, which operate in a similar fashion to the association cells. The activation of these response cells are the outputs of the perceptron, and indicated the classification of the input.

The methods by which the perceptron adjusted it weights were based on which cells were active, and whether the correct output was produced. By doing this, the system undergoes a process analogous to Hebbian learning.

This machine was initially trained to reliably identify three different shapes: a square, a circle, and a triangle; and did so with a better than chance probability. When attempting to use the perceptron for more complicated tasks, such as character recognition, it failed to produce better than chance results.

`TODO`: *Subsection: Perceptrons*

Backpropagation

In order for a neural network to learn, it must undergo some form of optimisation process. For the perceptron, this process was one of positive and negative reinforcement.

TODO: *Need better tangent* Another optimisation method that later became the norm was that of gradient descent, which originated from control theory.

TODO: *Subsection: Backpropagation*

Types of Neurons

TODO: *Section: Types of Neurons*

CNN

RNN

LSTM

Chapter 2

Neural Networks in Python

Python is a general-purpose programming language designed by Guido van Rossum, with an emphasis on readability and reusability (Rossum, 1996). It comes with an extensive standard library and is one of the most popular programming languages.

There are multiple options for interacting with Python, these include:

- typing commands into an interpreter,
- writing files and running them with an interpreter,
- using an online service such as Google Colab.

`TODO`: *Chapter: Neural Networks in Python*

Single Perceptron Boston Housing Data

`TODO`: *Section: Single Perceptron Boston Housing Data*

Multi Layer Perceptron Boston Housing Data

`TODO`: *Section: Multi Layer Perceptron Boston Housing Data*

XOR Gate

`TODO`: *Section: XOR Gate*

Chapter 3

Introduction To TensorFlow

TensorFlow `TODO`: *Chapter: Introduction To TensorFlow*

Linear Regression

`TODO`: *Section: Linear Regression*

XOR

`TODO`: *Section: XOR*

Boston Housing with Keras

`TODO`: *Section: Boston Housing with Keras*

Chapter 4

Deep Learning

TODO: *Chapter: Deep Learning*

Recurrent Neural Networks

TODO: *Section: Recurrent Neural Networks*

Convolutional Neural Networks

TODO: *Section: Convolutional Neural Networks*

Image Processing

TODO: *Subsection: Image Processing*

Supervised Learning

TODO: *Section: Supervised Learning*

Unsupervised Learning

TODO: *Section: Unsupervised Learning*

Autoencoding

TODO: *Section: Autoencoding*

Reinforcement Learning

TODO: *Section: Reinforcement Learning*

Chapter 5

Null

notes.txt

```
Biological Neurons

electrically excitable cell
communicates with other via synapses
three types of neuron
        sensory (touch, sound, smell, light, temperature, taste)
        motor (muscle control)
        interneuron (connects sensory and motor to the central nervous system)
most synapses signal from ones axon to anothers dendrites
synaptic signals may be excitatory or inhibitory
all-or-none response
        greater stimulation can increase firing frequency
        threshold logic
present in almost all animals



wiki summary (https://en.wikipedia.org/wiki/Artificial_neural_network)

computing systems inspired by biological neural networks
image recognition
        large set of manually labeled data
        no prior knowledge of what constitutes the label
        auto' generates identifying characteristics from data
collection of connected artificial neurons (nodes)
        receive signal from previous layer (edges)
        take weighted sum with bias
        non-linear function
        output new signal to next layer
applications
        computer vision
        speech recognition
        machine translation
        social network filtering
        playing games
        medical diagnosis
confirmations are empirical, not theoretical

history

1943: Warren McCulloch and Walter Pitts, created a computational model for neural networks [3]
1949: D. O. Hebb, Learning hypothesis based on neural plasticity (Hebbian learning) [5]
1954: Farley and Wesley A. Clark, first to simulate a Hebbian Network [6]
1958: Rosenblatt, perceptron [7,8]
1960: Henry J. Kelley, control theory backpropagation [15]
1961: Arthur Earl Bryson Jr, backpropagation [16]
1965: Ivakhnenko and Lapa, first functional with many layers [9,10,11]
1969: Marvin Lee Minsk and Seymour Aubrey Papert, limit of perceptron [21]
1970: Seppo Linnainmaa, automatic differentiation [17,18]
1973: Stuart Dreyfus, adapt parameters in proportion to error gradients [19]
1975: Paul John Werbos, pratical multi-layer network training [?]
1982: Paul John Werbos, applied Linnainmaa's AD method to neural networks [12,20]
1992: John (Juyang) Weng, Narendra Ahuja and Thomas S. Huang, max-pooling (CNN) [22,23,24]
1992: J rgen Schmidhuber, multi-level hierarchy of networks pre-trained unsupervised, backpropagation [25]
2006: Geoffrey Hinton et al, learning proposition using successive layers with restricted Boltzmann machine [27]
2010: Ciresan and colleagues, GPU backpropagation feasibility [29,30]
2012: Andrew Yan-Tak Ng and Jeffrey Adgate Dean, high-level concepts without labels [28]



Recurrent Neural Network

can use their internal memory to process variable length inputs

LSTM (https://en.wikipedia.org/wiki/Long_short-term_memory)
```

```
RNN architecture
feedback connections
can process entire sequences of data
applications
        connected handwriting recognition
        speech recognition
        anomaly detection
        time series predictions
components
        cell
                remembers values over arbitrary time interval
        input gate
        output gate
        forget gate
partially solves the vanishing gradient problem

Recursive

apply the same set of weights recursively
structured input, structured prediction, variable-size input structure
input traversed in topological order
applications
        natural language processing

history

1997: Sepp Hochreiter and J rgen Schmidhuber, constant error carousel units [1,5]
1999: Felix Gers, J rgen Schmidhuber and Fred Cummins, forget gate introduced [5,6]
2000: Felix Gers, J rgen Schmidhuber and Fred Cummins, peephole connections [5,7]
2014: Kyunghyun et al, simplified variant [8]



Autoencoder

network learns to copy the input to the output
encoding section is narrower than the input/output

        x       x'
            u
        y       y'
            v
        z       z'

sparse autoencoder
        improve performance on classification by encouraging sparsity
        hidden layer may have more nodes than input, but only a small number may be active at a given time (dropout)



Deep Belief Network

probabilistically reconstruction inputs
acts as feature detector



Vanishing Gradient Problem

affects gradient-based learning methods and backpropagation
caused by taking the product of many small values
prevents value from changing, partially due to rounding errors
solutions
        multi-level heirarchy
                pre-trained one level at a time
                fine-tuned using back propagation



Reinforcement Learning [An Introduction to Deep Reinforcement Learning]

concerned with software agents actions for cumulative rewards
balance between exploration and exploitation
no assumption of knowledge
agent's performance compared with target, difference in performance gives rise to "regret"
must reason about long term performance



Unsupervised Learning

finds previously unknown patterns within a data set
allows the modelling of probability densities of given inputs
        cluster analysis to group/segment attributes
        anomaly detection
        autoencoders
        deep belief network
                probabilistically reconstruction inputs
                acts as feature detector



Supervised Learning

learning of a function that maps an input to an output
function inferd from examples
ideally, the algorithm will correctly label unseen instances
```

```
issues
        inductive bias
        overfitting
```

TODO: *Decide parent chapter for CNN, RNN, and LSTM sections*

List of Figures

List of Tables

Bibliography

Hebb, Donald O (1949). *The organization of behavior.* John Wiley & Sons, Inc.

McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133. DOI: `10.1007/BF02478259`.

Rosenblatt, Frank (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6, p. 386. DOI: `10.1037/h0042519`.

Rossum, Guido van (1996). *Foreword for “Programming Python” (1st ed.)* URL: `https://www.python.org/doc/essays/foreword/` (visited on 02/07/2020).