Learning Neural Networks with Java

Ryan Sanders

# Table of Contents

[1 Table of Contents 2](#_Toc365194949)

[2 Introduction 3](#_Toc365194950)

[2.1 Overview and History of Neural Networks 3](#_Toc365194951)

[Part 1 6](#_Toc365194952)

[3 Feedforward Neural Networks 7](#_Toc365194953)

[3.1 Why Feedforward Neural Networks? 7](#_Toc365194954)

[3.2 What Can They Do? 7](#_Toc365194955)

[3.3 Limitations and Criticisms 7](#_Toc365194956)

[4 8](#_Toc365194957)

# Introduction

The aim of this book is to provide the reader with a good understanding of the theory behind feedforward neural networks trained using back-propagation, combined with a walkthrough of implementing such a network in Java. It is meant to be a combination of theory and practice, where the practice helps reinforce the theory and make things a bit more concrete, and hopefully more interesting.

This book is structured as follows:

Introduction This section. Includes an overview and history of neural networks.

Part 1 Goes through the theory underpinning feedforward neural networks and how to train a network using back-propagation. Implements such a network using Java. The network is used to learn a simple function (the XOR function) to prove it works.

Part 2 Delves into real world uses of feedforward neural networks. Builds upon the theory and code from part 1 to create a more interesting neural network that using advanced concepts.

## Overview and History of Neural Networks

A number of terms and concepts are presented here with minimal explanation. Later in this book we will revisit these concepts in detail; they are mentioned only as an overview.

In the late 19th century Alexander Bain, and later William James, began to understand how the brain works. As part of the artificial intelligence field, it made sense to think how the brain could be replicated in software in order to create a system that could learn.

A biological neural network, such as the brain, is made up of a vast number of neurons. A biological neuron is shown in Figure 1.

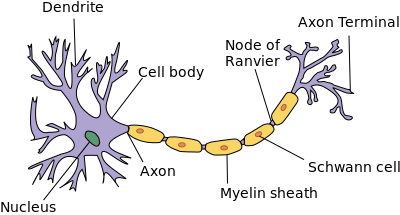


Figure 1: Biological neuron

For the purpose of comparing to an artificial neuron, we focus only on the dendrites, the cell body and the axon. The dendrites collect the output of other neurons such that they become the input to this neuron. The neuron processes the input and distributes it to the dendrites of other neurons through the axon.

This abstraction makes more sense when compared to an artificial neuron, such as the one in Figure 2.



Figure 2: Artificial neuron

Warren McCulloch and Walter Pitts created the artificial neuron in 1943. In this particular artificial neuron, there is one input and one output. The input is multiplied by a weight, in this case 1. If the result is greater than the threshold, 2, the neuron fires, or outputs 1. Otherwise the output is 0.

Now that we have an artificial neuron, the next step is to connect a number of them in some way to make an artificial neural network. This book focuses on feedforward neural networks, where the neurons are organized into layers and each neuron in a layer only connects to a neuron in a later layer.

The perceptron is the simplest form of feedforward neural network. It was invented by Frank Rosenblatt in 1957 and is comprised of two layers, and input layer and an output layer—see Figure 3 for an example.



Figure 3: An example of a perceptron, with two neurons in the input layer and one in the output layer

The alternative to feedforward networks is recurrent networks, where any neuron can connect to any other neuron. An example of a recurrent neural network is a Hopfield network, invented by John Hopfield in 1982.

Of course, an artificial neural network by itself is only part of the story. The goal is to make that network learn. The psychologist Donald Hebb came up with Hebbian theory to describe how biological neurons learn. The theory describes "associative learning", in which simultaneous activation of cells leads to pronounced increases in synaptic strength between those cells. The theory is often summarized as “cells that fire together, wire together”.

Hebbian learning was the first learning technique to be implemented in artificial neural networks. It is implemented such that the weight between two neurons increases if they activate simultaneously and reduces otherwise. Hopfield networks learn in this way and function as associative memory; after the network trains on a set of examples a new input will be classified as the example in the training set that most closely resembles the new input.

Frank Rosenblatt’s perceptron also included an algorithm for learning that is based on stochastic gradient descent. However, it turns out that the perceptron was quite limited in what it could learn; in particular, it can only categorize data that is linearly separable. As an example, this means that while it is capable of learning the AND function, it is incapable of learning something as simple as the XOR function. This limitation caused interest in neural networks to wane.

Learning more complex functions requires additional layers, but training such a network was not possible until the invention of the back-propagation algorithm. The algorithm was first invented in 1969 by Arthur Bryson and Yu-Chi Ho. However, it wasn’t until the mid-1980s that it was used to train artificial neural networks. Now neural networks could learn more complex functions, and interest in neural networks was revived.



# Feedforward Neural Networks

## Why Feedforward Neural Networks?

This book focuses on feedforward neural networks. A reasonable question is, why? There are a number of different neural networks, including recurrent neural networks such as the Hopfield network mentioned in Overview and History of Neural Networks (section 2.1, page 3).

The reason is that feedforward neural networks form a small but significant part of a larger field, and examining them in detail covers a lot of ground. Also, a number of other networks, such as self organizing maps, are extensions or variations of feedforward networks. So focusing on feedforward neural networks gives a good base for further exploration into neural networks.

## What Can They Do?

Another reasonable question is, what can feedforward networks do?

Neural networks are good at solving classification problems and for approximating functions that are tolerant of some imprecision and do not have a known deterministic solution. To be successful they generally need lots of available training data.

## Limitations and Criticisms

In covering what neural networks can do we immediately uncover some limitations. They are limited to certain types of problems. They need a lot of training data to learn a solution. In addition, that training data needs to be diverse.

Other limitations include that once a solution is arrived at, it cannot be examined to understand how the problem was solved. Learning also tends to be slow.

Another common criticism is that, although neural networks were meant to mimic how the human brain learns, they do not yet do so. One counter argument is that when the Wright brothers achieved human flight, it was not by mimicking birds.

# 