Senior Honours Project

Freeing Neural Training Through Surfing

Georg Wölflein
170011885
Supervisor: Dr. Mike Weir

April 9, 2020

Word count: 1096 words

Abstract

TODO

Declaration

“I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 1096 words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.”

Georg Wölflein

Contents

Chapter 1

Introduction

Describe the problem you set out to solve and the extent of your success in solving it. You should include the aims and objectives of the project in order of importance and try to outline key aspects of your project for the reader to look for in the rest of your report. **TODO**

Chapter 2

Context survey

Surveying the context, the background literature and any recent work with similar aims. The context survey describes the work already done in this area, either as described in textbooks, research papers, or in publicly available software. You may also describe potentially useful tools and technologies here but do not go into project-specific decisions.

- TensorFlow
- keras

TODO

Chapter 3

Requirements specification

Primary objectives:

1. Design a generic framework that can be used for various neural training algorithms with a clear set of inputs and outputs at each step. This framework should include benchmarking capabilities.

2. For a simple case of this framework (when the dimensionality of the control space and output space are suitably low), implement a visualisation tool that shows the algorithm's steps.

3. Implement a particular training algorithm for the framework that uses potential field techniques.

4. Evaluate the performance of this and other algorithms on tasks of differing complexity, especially with regard to the local minimum problem and similar issues.

Secondary objectives:

1. Investigate how this approach can be generalized to any numerical optimisation problems.

Chapter 4

Theory

4.1 Supervised learning

Regression model In machine learning, a regression model f is defined as a mathematical function of the form

$$f(\mathbf{x}) = \hat{y} = y + \epsilon \tag{4.1}$$

that models the relationship between a D-dimensional feature vector $\mathbf{x} \in \mathbb{R}^D$ of independent (*input*) variables and the dependent (*output*) variable $y \in \mathbb{R}$. Given a particular $\mathbf{x}$, the model will produce a *prediction* for y which we denote $\hat{y}$. Here, the additive error term ϵ represents the discrepancy between y and $\hat{y}$.

Supervised learning A supervised learning algorithm for a regression task infers the function f given in (4.1) from a set of *labelled training data*. This dataset consists of N tuples of the form $\langle \mathbf{x}_i, y_i \rangle$ for $i = 1, \ldots, N$. For each feature vector $\mathbf{x}_i$, the corresponding y_i represents the observed output, or *label* [Burkov, 2019]. We use the vector

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix}^\mathsf{T} \tag{4.2}$$

to denote all the labelled outputs in the dataset, and

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{bmatrix}^\mathsf{T} \tag{4.3}$$

is the $N \times D$ matrix representing the corresponding feature vectors.

Similarly,

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_N \end{bmatrix}^\mathsf{T} \tag{4.4}$$

denotes a particular prediction for each training sample.

4.2 Artifical neural networks

Artifical neural networks (ANNs) take inspiration from the human brain and can be regarded as a set of interconnected neurons. More formally, an ANN is a directed graph of n neurons (referred to as *nodes* or *units*) with weighted edges (*links*). Each link connecting two units i and j is directed and associated with a real-valued weight $w_{i,j}$.

A particular unit i's *excitation*, denoted ex_i, is calculated as the weighted sum

$$ex_i = \sum_{j=1}^{n} w_{j,i} a_j + b_i \tag{4.5}$$

where $a_j \in \mathbb{R}$ is another unit j's *activation* and $b_i \in \mathbb{R}$ is the ith unit's *bias*. Notice that if there exists no link between unit i and a particular j then simply $w_{i,j} = 0$ and therefore j will not contribute to i's excitation.

The unit i's activation is its excitation applied to a non-linear *activation function*, g. We have

$$a_i = g\left(ex_i\right) = g\left(\sum_{j=1}^{n} w_{j,i} a_j + b_i\right). \tag{4.6}$$

Activation functions will be explored in more detail in Section 4.4.

ANNs as regression models We can employ an ANN to model a regression problem of the form given in (4.1). To do so, we need at least $D+1$ neurons in the network. We consider the first D units to be the *input* neurons, and the last neuron, n, is the output unit. Furthermore, we require $w_{j,k} = 0$ for $j, k \in \mathbb{Z}^+$ where $j \leq n$ and $k \leq D$ to ensure that there are no links feeding into the input neurons.

To obtain the prediction $\hat{y}$ given the D-dimensional feature vector $\mathbf{x}$, we set the activation of the ith unit to the value the ith element in $\mathbf{x}$ for $i = 1, \ldots, D$. Then, we propagate the activations using (4.6) until finally the prediction is the activation of the last neuron, $\hat{y} = a_n$. This process is often called *forward propagation* or *forward pass* [Russell and Norvig, 2010].

Single-layer perceptron A single-layer perceptron (SLP) is a type of ANN that consists of two layers, an input and an output layer. Every input node is connected to every output node, but there are no intra-layer links (i.e. there are no links between any two input nodes or any two output nodes). This is what we call a *fully-connected feedforward* architecture. SLP architectures will always form a *directed acyclic graph* (DAG) because there are no intra-layer or backwards connections. Figure 4.1 depicts the DAG for an example SLP and **TODO**

Let us consider a SLP with m inputs and n outputs. Since every output unit i only has connections from every input unit j, we can adapt (4.6) to give

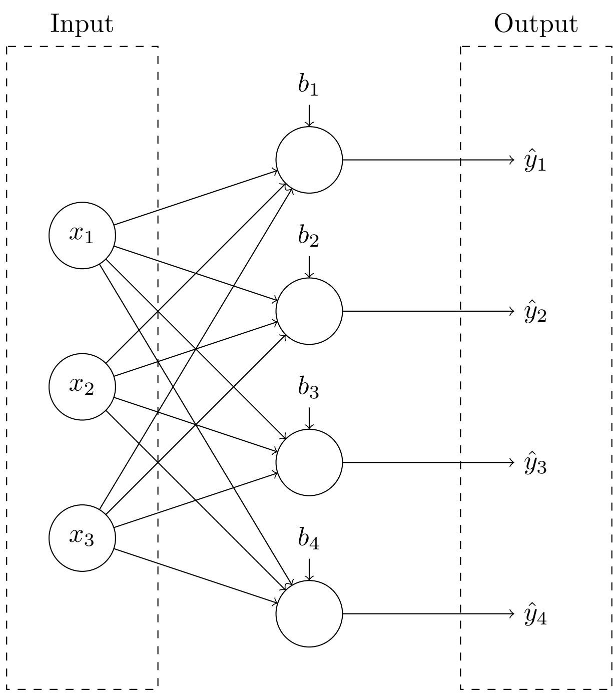

Figure 4.1: A single-layer perceptron with 3 input and 4 output neurons.

the activation of a particular output neuron i as

$$a_i = y_i = g\left(ex_i\right) = g\left(\sum_{j=1}^{m} w_{j,i} x_j + b_i\right) = g\left(\mathbf{w}_i \mathbf{x}_i + b_i\right) \tag{4.7}$$

where $\mathbf{w}_i = \begin{bmatrix} w_{1,i} & w_{2,i} & \cdots & w_{m,i} \end{bmatrix}$ represents the weights of all the edges that connect to output unit i. If we use the $n \times m$ matrix

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_n \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{m,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,n} & w_{2,n} & \cdots & w_{m,n} \end{bmatrix} \tag{4.8}$$

to capture all weights and the vector $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}^{\mathsf{T}}$ for the biases, we can give a mathematical formula describing the relationship between the inputs and outputs as

$$\mathbf{f_{SLP}}(\mathbf{x}) = \hat{\mathbf{y}} = \mathbf{W}\mathbf{x} + \mathbf{b}. \tag{4.9}$$

Multi-layer perceptron A multi-layer perceptron (MLP) is a fully-connected feedforward ANN architecture with multiple layers which we will define in terms of multiple nested SLPs [Burkov, 2019].

employed in this project

retain DAG property

4.3 Weight and output spaces

We define the weight space $\mathcal{W}$ **TODO**

The output space $\mathcal{O}$ spans the space of all possible output predictions on the training set, $\hat{\mathbf{y}}$, so $\mathcal{O} = \mathbb{R}^N$ considering the fact that the training set has N samples.

4.4 Activation functions

Although units within a network can have different activation functions, this project solely employs networks where every unit uses the same g. Common activation functions include the sigmoid

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4.10}$$

TODO

Ideas

Generalize to classification as regression with multiple output variables?

Bibliography

[Burkov, 2019] Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Andriy Burkov.

[Russell and Norvig, 2010] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson, 3rd edition.