Blah Blah Blah

A thesis submitted by

Your Name Here

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Tufts University

December 2023

Advisor: ...

The abstract is a very brief summary of the dissertation's contents. It should be half a page long at most. Somebody unfamiliar with your project should have a good idea of what it's about having read the abstract alone and will know whether it will be of interest to them.

This is not needed, but common.

Chapter 1

Background

1.1 Artificial Neural Networks

Artificial neural networks (ANNs) are non-linear computational models that approximate a target function $f : \mathbb{R}^n \to \mathbb{R}^m$, where n and m are integers [1]. Given a set $X = \{(x_1, y_1), ..., (x_N, y_N)\} \subset \mathbb{R}^n \times \mathbb{R}^m$ of input-output pairs of size N the model is trained to approximate f such that $f(x_i) = y_i \forall i \in \{1, 2, \ldots, N\}$. By the Universal Approximation theorem, a neural network's approximation of a continuous function is theoretically guaranteed to be as precise as it needs to be given the network has at least one hidden layer with some finite number of nodes [2].

Consider the problem of predicting the value of one or more continuous target variables $\mathbf{t}$ provided a D-dimensional vector $\mathbf{x}$ of input variables, or what is called regression. Given a training set consisting of N observations $\{ \mathbf{x}_1, ..., \mathbf{x}_n \}$ and values $\{t_n\}_{n=1}^N$, the objective is to predict the output value for any input x as close as possible to the provided target value $\mathbf{t}$. The regression function is a linear combination over the input variables

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D \tag{1.1}$$

where $\mathbf{x}$ has D dimensions, $\mathbf{x} = (x_1, ..., x_D)^T$ and $w \in \mathbb{R}^{D+1}$ represents the parameters of the function, $w = (w_0, ..., w_D)$. This function is limited to being a linear function over the input vector $\mathbf{x}$. Non-linear basis functions ϕ

on the input variables make the function $y(\mathbf{x}, \mathbf{w})$ non-linear.

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{N} w_i \phi_i (x_i) \tag{1.2}$$

The basic neural network model extends this by making the basis functions $\phi(x)$ depend on learnable parameters along with the weights w.The neural network performs a series of linear transformations. We start with M linear combinations of the input variables where $j = 1, ..., M$.

$$a_j = \sum_{i=1}^{D} w_{ji} x_i + w_{j0} \phi (x_i) \tag{1.3}$$

The *activation* a_j is transformed using non-linear activation function h.

$$z_j = h(a_j) \tag{1.4}$$

This transformation corresponds to a *layer* in a neural network. The network has two layers - one input layer over the input variables $x_1, ..., x_D$ and one output layer that takes the activation from the input layer's transformation through an activation function suited to the provided target variables $t_1, ..., t_N$. The parameters for the input layer are represented with a superscript (1) and the parameters for the output layer are represented with a superscript (2).

$$z_j = h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) \tag{1.5}$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{1.6}$$

Equations (1.5) and (1.6) combined present the equation for a *feed-forward* pass through a 2-layer ANN.

$$y_k(\mathbf{x}, \mathbf{w}) = \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{D} w_{ji}^{(1)} + w_{j0}^{(1)} \phi (x_i) \right) + w_{k0}^{(2)} \right) \tag{1.7}$$

1.1.1 Training a neural network

The goal of learning for a neural network is to optimize the weights of the network such that the loss function $E(X, \mathbf{w})$ takes the lowest value. Continuing with the previous example for regression, we look at the sum-of-squares error function

$$E(X, \mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \| y\left(\mathbf{x}_n, \mathbf{w}\right) - \mathbf{t}_n \|^2 \tag{1.8}$$

There is typically not an analytical solution and iterative procedures are used to minimize the loss function E. The steps taken are

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta \mathbf{w}^{(\tau)} \tag{1.9}$$

where τ is the iteration step. An approach for the weight update step with $\Delta \mathbf{w}^{(\tau)}$ is to use gradient information where the gradient is of $E(X, \mathbf{w})$ with respect to the parameters $\mathbf{w}$. The weights are updated in the direction of steepest error function decrease or in the $-\nabla E(X, \mathbf{w})$ direction. Equation 1.9 thus becomes

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \alpha \nabla E\left(X, \mathbf{w}^{(\tau)}\right) \tag{1.10}$$

where $\alpha > 0$ is the learning rate controlling the size of update step taken. This iterative procedure is called *gradient descent optimization* [3]. The gradient must be calculated with respect to every layer's weights in the neural network. Consider first the case of one weight w_{ji} where j corresponds to the parameter in the layer and i corresponds to the component in the input. Provided the error function for 1 input $E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$, the derivative of E_n with respect to w_{ji} is

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni} \tag{1.11}$$

However, in a neural network there is more than 1 weight and more than 1 layer. Backpropagation seeks to calculate the derivative of the error with respect to every weight in the network from the output units backwards through all hidden units. Equation 1.11 can be rewritten as

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{1.12}$$

where a_j is the weighted sum of previous units with weight w_{ji}. This term is called an *error* where here it is the error for unit j.

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} \tag{1.13}$$

And because $\frac{\partial a_j}{\partial w_{ji}} = z_i$ we can write equation 1.12 as

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{1.14}$$

1.2 Title section

1.3 Notation

1.4 Published material

Chapter 2

Title chapter 2

2.1 Title section 2.1

2.1.1 If needed

2.1.2 If needed

2.2 Title section 2.2

2.2.1 If needed

2.2.2 If needed

2.3 Title section 2.3

2.3.1 If needed

2.3.2 If needed

Appendix A

Title of the Appendix

Bibliography

[1] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.