Proofs for the Four Fundamental Equations of the Backpropagation and Algorithms in Feedforward Neural Networks

Hao Li
Department of Mathematics
Knox College
`haoli25@uw.edu`

Abstract

This research paper delves into the intricate algorithm of **backpropagation** in the realm of machine learning, with a particular emphasis on its application in training feedforward neural networks. The discourse extends to a meticulous proof of the four fundamental equations of backpropagation and demonstrates its practical implementation, in tandem with the stochastic gradient descent technique, for recognizing handwritten digits. While portions of the proof draw upon Michael Nielsen's work [2], this paper endeavors to furnish additional details and basic definitions pertaining to the gradient.

1 Introduction

Machine learning, a subset of artificial intelligence, has witnessed the pervasive utilization of the **backpropagation** algorithm, especially in the training of feedforward neural networks [4]. Initially introduced by Rumelhart, Hinton, and Williams in 1986, *backpropagation* has been heralded for its efficacy in rapidly computing the gradient of the cost function relative to the weights and biases across all network layers [3]. The algorithm, when synergized with the stochastic gradient descent technique, facilitates the optimization of weights and biases, thereby minimizing the cost function, which quantifies the discrepancies between predicted outcomes and actual results.

The ensuing sections of this paper will meticulously dissect the backpropagation algorithm, providing a comprehensive proof of its four fundamental equations. Furthermore, a practical implementation of the algorithm, in conjunction with the stochastic gradient descent technique, will be illustrated through an application aimed at recognizing handwritten digits. While certain segments of the proof are derived from Michael Nielsen's online book, *Neural Networks and Deep Learning* [2], this paper seeks to augment his contributions by providing an in-depth exploration of his proofs and introducing fundamental definitions related to the gradient.

2 Preliminary Definitions and Notations

Notations: Let $w^l_{j,k}$ denote the weight for the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer. Let b^l_j denote the j^{th} bias in the l^{th} layer. The activation a^l_j can be expressed as

$$a^l_j = \sigma\left(\sum_k w^l_{j,k} \cdot a^{l-1}_k + b^l_j\right), \tag{1}$$

where σ represents our activation function. Let C denote the cost function, which measures the error between the prediction a^L and the correct answer, where the variables are the weights and biases.

Definition 1. *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a scalar field. The gradient of g at $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is denoted $\nabla g(\mathbf{x})$ and is defined by*

$$\nabla g(\mathbf{x}) = \left(\frac{\partial g}{\partial x_1}(\mathbf{x}), \frac{\partial g}{\partial x_2}(\mathbf{x}), \ldots, \frac{\partial g}{\partial x_n}(\mathbf{x}) \right). \tag{2}$$

Definition 2. *Let $\mathbf{v}$ be a unit vector. The directional derivative of g at $\mathbf{x}$ in the direction of $\mathbf{v}$ is defined by*

$$D_v g(\mathbf{x}) = \mathbf{v} \cdot \nabla g(\mathbf{x}). \tag{3}$$

Definition 3. *The error of the j^{th} neuron in layer l is defined by:*

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}, \tag{4}$$

where $z_j^l = \sum_k w_{j,k}^l \cdot a_k^{l-1} + b_j^l$.

3 Theorems and Derivations

Consider a cost function C defined as the mean square error function, given by

$$C(\mathbf{w}, \mathbf{b}) = \frac{1}{2n} \sum_{\mathbf{x}} ||y(\mathbf{x}) - \mathbf{a}||^2, \tag{5}$$

where $\mathbf{w}$ and $\mathbf{b}$ represent the collections of all weights and biases, respectively, and n denotes the total number of training inputs. Furthermore, $\mathbf{a}$ is the vector of network outputs (predictions), and $y(\mathbf{x})$ is the desired output (correct answer) when $\mathbf{x}$ is the input.

Theorem 4. *The partial derivatives of the cost function with respect to any weights and biases can be computed using the following four fundamental equations:*

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \, \sigma'(z_j^L), \tag{BP1}$$

$$\delta_j^l = \sum_k w_{k,j}^{l+1} \, \delta_k^{l+1} \, \sigma'(z_j^l), \tag{BP2}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l, \tag{BP3}$$

$$\frac{\partial C}{\partial w_{j,k}^l} = a_k^{l-1} \, \delta_j^l. \tag{BP4}$$

Equation (BP1) elucidates the computation of the error in the final layer of a feedforward network. Subsequently, equation (BP2) enables the back-propagation of the error from layer $l+1$ to layer l by expressing the error in layer l in terms of the error in layer $l+1$. Equations (BP3) and (BP4) establish that the partial derivative of the cost function with respect to any bias is equal to the error in that layer, and the partial derivative with respect to any weight is the product of the error in layer l and the activation in layer $l-1$, respectively.

Proof.

- For equation (BP1), we first express the error in the last layer using its definition:

$$\delta_j^L = \frac{\partial C}{\partial z_j^L}. \tag{6}$$

  Given that $a_j^L = \sigma(z_j^L)$, we can apply the chain rule to obtain:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L}. \tag{7}$$

  Since $\frac{\partial a_j^L}{\partial z_j^L} = \sigma'(z_j^L)$, substituting into equation (7) yields:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \, \sigma'(z_j^L). \tag{8}$$

  Equation (8) corresponds to equation (BP1), thereby validating its formulation.

- For equation (BP2), we will compute the partial derivative of the cost with respect to z_k in layer $l+1$. Applying the chain rule, we obtain

$$\begin{aligned} \delta_j^l &= \sum_k \frac{\partial C}{\partial z_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \cdot \delta_k^{l+1}. \end{aligned} \tag{9}$$

  Note that if the z_k^{l+1} in the denominator and numerator cancel out, this is simply another way to express the error δ_j^l. The last equality holds due to the definition of δ_k^{l+1}. Now, by the definition of z_k^{l+1}, we have

$$z_k^{l+1} = \sum_j w_{k,j}^{l+1}\, a_j^l + b_k^{l+1} = \sum_j w_{k,j}^{l+1}\, \sigma(z_j^l) + b_k^{l+1}. \tag{10}$$

  Differentiating, we obtain

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{k,j}^{l+1}\, \sigma'(z_j^l). \tag{11}$$

  Substituting back into equation (9), we derive

$$\delta_j^l = \sum_k w_{k,j}^{l+1}\, \delta_k^{l+1}\, \sigma'(z_j^l). \tag{12}$$

  Equation (12) corresponds to equation (BP2), thereby validating its formulation.

- For equation (BP3), first note that

$$z_j^l = \sum_k w_{j,k}^l\, a_k^{l-1} + b_j^l, \tag{13}$$

  Taking the derivative with respect to b_j^l, we have

$$\frac{\partial z_j^l}{\partial b_j^l} = 1. \tag{14}$$

  Multiplying $\frac{\partial z_j^l}{\partial b_j^l}$ to both sides of the error formula, we obtain

$$\begin{aligned} \delta_j^l &= \frac{\partial C}{\partial z_j^l} \cdot \frac{\partial z_j^l}{\partial b_j^l} \\ &= \frac{\partial C}{\partial b_j^l}. \end{aligned} \tag{15}$$

  Equation (15) corresponds to equation (BP3), thereby validating its formulation.

- For equation (BP4), similarly, taking the derivative of z_j^l with respect to $w_{j,k}^l$, we get:

$$\frac{\partial z_j^l}{\partial w_{j,k}^l} = a_k^{l-1}. \tag{16}$$

  Multiplying this equation to both sides of the error formula, we derive

$$\begin{aligned} \delta_j^l\, a_k^{l-1} &= \frac{\partial C}{\partial z_j^l} \cdot \frac{\partial z_j^l}{\partial w_{j,k}^l} \\ &= \frac{\partial C}{\partial w_{j,k}^l}. \end{aligned} \tag{17}$$

  Equation (17) corresponds to equation (BP4), thereby validating its formulation.

$\square$

Now we can quickly find gradients of the cost function with respect to any weights and biases. Let me introduce the gradient descent technique. With this technique, we will be able to find the best weights and biases that minimize the cost function.

4 Algorithms

Consider a function $g : \mathbb{R}^n \to \mathbb{R}$ and let $\mathbf{x}^{(0)}$ denote the initial point, with η being a positive constant. In the context of our network, the vector $\mathbf{x}$ represents the weights and biases $\mathbf{w}, \mathbf{b}$.

Gradient Descent Algorithm:

1. Evaluate g at the initial point $\mathbf{x}^{(0)}$.
2. Determine a direction from $\mathbf{x}^{(0)}$ that results in the steepest decrease in the value of g.
3. Move a certain distance in this direction and denote the new value as $\mathbf{x}^{(1)}$.
4. Repeat steps 1 through 3, substituting $\mathbf{x}^{(0)}$ with $\mathbf{x}^{(1)}$ [1].

The transition from step (1) to (3) can be generalized as follows:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \eta \, \nabla g(\mathbf{x}^{(0)}). \tag{18}$$

However, a direct application of this algorithm to every weight and bias is computationally expensive. To mitigate this, we estimate the gradient ∇C by computing ∇C_x for a small, randomly chosen sample of training inputs.

We denote random training inputs as $X_1, X_2, \dots, X_m$ and refer to them as a *mini-batch*. Assuming the sample size m is sufficiently large, we expect that the average value of the ∇C_{X_j} will approximate the average over all ∇C_x, i.e.,

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C. \tag{19}$$

Applying the **stochastic gradient descent technique**, the updated weights and biases are given by:

$$\mathbf{w}_k^{(1)} = \mathbf{w}_k^{(0)} - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}, \tag{20}$$

$$\mathbf{b}_l^{(1)} = \mathbf{b}_l^{(0)} - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}. \tag{21}$$

5 Implementation of the Backpropagation Algorithm

We introduce an algorithm that synergizes the stochastic gradient descent technique and backpropagation to implement a network for recognizing handwritten digits. Given a mini-batch of m training examples:

The Backpropagation Algorithm:

1. **Input a set of training examples.**
2. **For each training example $\mathbf{x}$:**
   - **Feedforward:** Compute

     $$z^{x,l} = w^l \, a^{x,l-1} + b^l \quad \text{and} \quad a^{x,l} = \sigma(z^{x,l})$$

     for layers 2 through L.
   - **Output error:** Compute the error vector at the output layer:

     $$\delta^{x,L} = \nabla_{a^L} C_x \circ \sigma'(z^{x,L}).$$

   - **Backpropagate the error:** Determine the error vector at each layer using equation (BP2).

3. **Update the weights and biases:**

$$w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} \, (a^{x,l-1})^T, \tag{22}$$

$$b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}. \tag{23}$$

   Note that within the summation, we substitute the gradient with equations (BP3) and (BP4).

After processing each mini-batch, the weights and biases are updated, and this process is repeated across multiple iterations to evaluate accuracy.

6 Conclusions

The efficacy of a feedforward neural network is fundamentally tethered to its weights and biases, as the output or prediction it generates is a function thereof. The pivotal objective in machine learning, therefore, is to discern optimal weights and biases, ensuring that the network's predictions approximate the correct answers as closely as possible. The gradient descent technique and backpropagation emerge as the linchpin in this context, enabling the rapid and precise identification of weights and biases that minimize discrepancies between predictions and actual answers.

It is imperative to acknowledge that the algorithms and some theorem proofs presented herein are derived from Michael Nielsen's online book, *Neural Networks and Deep Learning* [2]. The book can be accessed at the following URL: `http://neuralnetworksanddeeplearning.com/index.html`.

References

[1] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Thomson Brooks/Cole, 8 edition, 2005.

[2] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 1 edition, 2015.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986.

[4] Wikipedia contributors. Backpropagation — Wikipedia, the free encyclopedia, 2021. [Online; accessed 1-June-2021].

7 Appendix

In this appendix, we present a theorem and its proof to elucidate why following the gradient is the optimal strategy for minimizing the cost.

Theorem 5. *The gradient ∇f points in the direction of the maximum rate of increase of the scalar field f.*

Proof. Firstly, invoking the definition of the dot product, we have

$$\begin{aligned} D_v f(\mathbf{x}) &= |\nabla f|\,|\mathbf{v}|\ \cos\theta \\ &= |\nabla f|\ \cos\theta, \end{aligned} \tag{24}$$

where θ is the angle between ∇f and $\mathbf{v}$. Given that $\cos\theta$ ranges within $[-1, 1]$, we deduce

$$D_v f(\mathbf{x})_{\text{max}} = |\nabla f| \quad \text{when } \theta = 0^\circ, \tag{25}$$

$$D_v f(\mathbf{x})_{\text{min}} = -|\nabla f| \quad \text{when } \theta = 180^\circ. \tag{26}$$

□

The directional derivative, which indicates the rate of change of the scalar field (in our context, the cost) in the direction of $\mathbf{v}$ at point $\mathbf{x}$, implies that the most expeditious path to the scalar field's minimum is to traverse in the direction indicated by the gradient.