THE INCREDIBLE SHRINKING NEURAL NETWORK: PRUNING TO OPERATE IN CONSTRAINED MEMORY ENVIRONMENTS

Nikolas Wolfe, Aditya Sharma, Lukas Drude & Bhiksha Raj
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{nwolfe, adityasharma, bhiksha}@cmu.edu
{drude@nt.upb.de}

ABSTRACT

We propose and evaluate a method for pruning neural networks to operate in constrained memory environments such as mobile or embedded devices. We evaluate a simple pruning technique using first-order derivative approximations of the gradient of each neuron in an optimally trained network, and turning off those neurons which contribute least to the output of the network. We then show the limitations of this type of approximation by comparing against the ground truth value for the change in error resulting from the removal of a given neuron. We attempt to improve on this using a second-order derivative approximation. We also explore the correlation between neurons in a trained network and attempt to improve our choice of candidate neurons for removal to account for faults that can occur from the removal of a single neuron at a time. We argue that this method of pruning allows for the optimal tradeoff in network size versus accuracy in order to operate within the memory constraints of a particular device or application environment.

1 INTRODUCTION

2 METHODOLOGY

The general approach taken to prune an optimally trained neural network in the present work is to create a ranked list of all the neurons in the network based off of one of the 3 ranking criteria we discuss further in this section. This ranking is done using a validation dataset which is different from the dataset used for training the network. The effects of removing an increasing percentage of neurons based off their ranks are then analysed in the results section.

2.1 BRUTE FORCE REMOVAL APPROACH

This is perhaps the most naive yet the most accurate method for pruning the network. It is also the slowest and hence unusable on large-scale neural networks with thousands of neurons. The idea is to manually check the effect of every single neuron on the output. This is done by running a forward propagation on the validation set K times (where K is the total number of neurons in the network), turning off exactly one neuron each time (keeping all other neurons active) and noting down the change in error. Turning a neuron off can be achieved by simply setting its output to 0. This results in all the outgoing weights from that neuron being turned off. This change in error is then used to generate the ranked list.

2.1.1 TAYLOR SERIES REPRESENTATION OF ERROR

Let us denote the total error from the optimally trained neural network for any given validation dataset with N instances as E_{total}. Then,

$$E_{\text{total}} = \sum_n E_n, \tag{1}$$

where E_n is the error from the network over one validation instance. E_n can be seen as a function O, where O is the output of any general neuron in the network (In reality this error depends on each neuron's output, but for the sake of simplicity we use O to represent that). This error can be approximated at a particular neuron's output (say O_k) by using the 2nd order Taylor Series as,

$$\hat{E}_n(O) \approx E_n(O_k) + (O - O_k) \cdot E_n(O_k) + \left.\frac{\partial E_n}{\partial O}\right|_{O_k} + 0.5 \cdot (O - O_k)^2 \cdot \left.\frac{\partial^2 E_n}{\partial O^2}\right|_{O_k}, \tag{2}$$

where $\hat{E}_n(O_k)$ represents the contribution of a neuron k to the total error E_n of the network for any given validation instance n. When this neuron is pruned, its output O_k becomes 0. From equation 2, the contribution $E_n(0)$ of this neuron, then becomes:

$$\hat{E}_n(0) \approx E_n(O_k) - O_k \cdot \left.\frac{\partial E_n}{\partial O}\right|_{O_k} + 0.5 \cdot O_k^2 \cdot \left.\frac{\partial^2 E_n}{\partial O^2}\right|_{O_k} \tag{3}$$

Replacing O by O_k in equation 2 shows us that the error is approximated perfectly by equation 2 at O_k. Using this and equation 3 we get:

$$\Delta E_{n,k} = \hat{E}_n(0) - \hat{E}_n(O_k) = -O_k \cdot \left.\frac{\partial E_n}{\partial O}\right|_{O_k} + 0.5 \cdot O_k^2 \cdot \left.\frac{\partial^2 E_n}{\partial O^2}\right|_{O_k}, \tag{4}$$

where $\Delta E_{n,k}$ is the change in the total error of the network given a validation instance n, when exactly one neuron (k) is turned off.

2.2 LINEAR APPROXIMATION APPROACH

We define the following network terminology here which will be used in this section and all subsequent sections unless stated otherwise. Figure ????? can be used as a reference to the terminology defined here:

$$E = \frac{1}{2}\sum_i (o_i^{(0)} - t_i)^2 \quad o_i^{(m)} = \sigma(x_i^{(m)}) \quad x_i^{(m)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} \quad c_{ji}^{(m)} = w_{ji}^{(m)} o_j^{(m+1)} \tag{5}$$

Superscripts represent the index of the layer of the network in question, with 0 representing the output layer. E is the squared-error network cost function. Note that we are dropping the E_n notation used previously as the subsequent discussion is insusceptible to the data instances. $o_i^{(m)}$ is the ith output in layer m generated by the activation function σ, which in this paper is is the standard logistic sigmoid. $x_i^{(m)}$ is the weighted sum of inputs to the ith neuron in the mth layer, and $c_{ji}^{(m)}$ is the contribution of the jth neuron in the $(m+1)$th layer to the input of the ith neuron in the mth layer. $w_{ji}^{(m)}$ is the weight between the jth neuron in the $(m+1)$th layer and the ith neuron in the mth layer.

We can use equation 4 to get the linear error approximation of the change in error due to the kth neuron being turned off and represent it as ΔE_k^1 as follows:

$$\Delta E_k^1 = -o_k \cdot \left.\frac{\partial E}{\partial o_j^{(m+1)}}\right|_{o_k} \tag{6}$$

The derivative term above is the first-order gradient which represents the change in error with respect to the output of a given neuron o_j in the $(m+1)$th layer. This term can be collected during back-propagation. The derivative term above can be calculated as follows:

$$\frac{\partial E}{\partial o_j^{(m+1)}} = \sum_i \frac{\partial E}{\partial x_i^{(m)}} \cdot w_{ji}^{(m)} \tag{7}$$

The full step-by-step mathematical derivation of the above equation can be found in the appendix.

2.3 Quadratic Approximation Approach

As seen in equation 4, $\Delta E_{n,k}$ which can now be represented as ΔE_k^2 is the quadratic approximation of the change in error due to the kth neuron being turned off. The quadratic term in equation 4 requires some discussion which we provide here. A more detailed and step-by-step mathematical derivation can be found in the appendix.

Let us reproduce equation 4 in our new terminology here:

$$\Delta E_k^2 = -o_k \cdot \left.\frac{\partial E}{\partial o_j^{(m+1)}}\right|_{o_k} + 0.5 \cdot o_k^2 \cdot \left.\frac{\partial^2 E}{\partial {o_j^{(m+1)}}^2}\right|_{o_k} \tag{8}$$

The second term here involves the second-order gradient which represents the second-order change in error with respect to the output of a given neuron o_j in the $(m+1)$th layer. This term can be generated by performing back-propagation using second derivatives. A full derivation of the second derivative back-propagation can be found in the appendix. We will quote some results from the derivation here. The second-order derivative term can be represented as:

$$\frac{\partial^2 E}{\partial {o_j^{(m+1)}}^2} = \sum_i \frac{\partial^2 E}{\partial {c_{ji}^{(m)}}^2} \left(w_{ji}^{(m)}\right)^2 \tag{9}$$

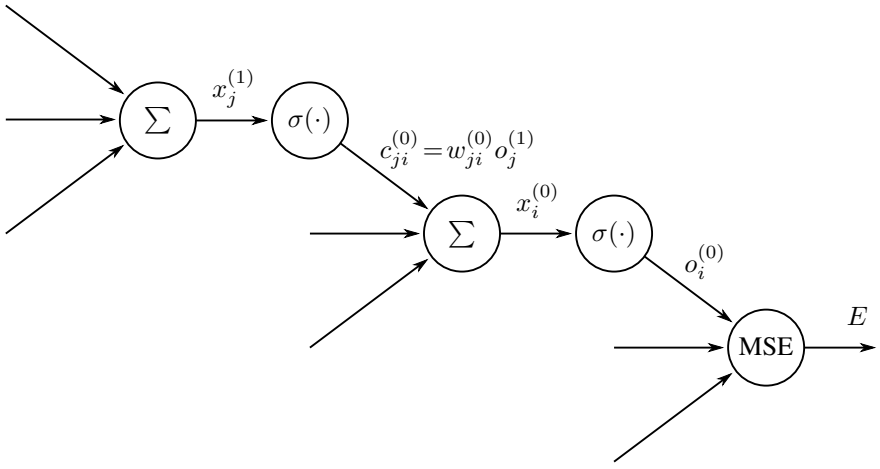

Figure 1: A computational graph of a simple feed-forward network illustrating the naming of different variables, where $\sigma(\cdot)$ is the nonlinearity, MSE is the mean-squared error cost function and E is the overall loss.

Here,$c_{ji}^{(m)}$ is one of the component terms of $x_i^{(m)}$, as follows from the equations in 5. Hence, it can be easily proved that (full proof in appendix):

$$\frac{\partial^2 E}{\partial {c_{ji}^{(m)}}^2} = \frac{\partial^2 E}{\partial {x_i^{(m)}}^2} \tag{10}$$

Now, the value of $x_i^{(m)}$ can be easily calculated through the steps of the second-order back-propagation using Chain Rule. The full derivation can again, be found in the appendix.

$$\frac{\partial^2 E}{\partial {x_i^{(m)}}^2} = \frac{\partial^2 E}{\partial {o_i^{(m)}}^2} \left(\sigma'\left(x_i^{(m)}\right)\right)^2 + \frac{\partial E}{\partial o_i^{(m)}} \sigma''\left(x_i^{(m)}\right) \tag{11}$$

2.4 Pruning Algorithm: The Gain-switch

We propose the Gain-switch algorithm for pruning an optimally trained neural network here. We define gain as the quadratic error approximation ΔE_k^2 we got from the Taylor Series.

The first step is to decide a stopping criterion. This can vary depending on the application but some intuitive stopping criteria can be the maximum number of neurons to remove, percentage scaling needed, maximum allowable accuracy drop etc. The Gain-switch algorithm performs the pruning in a greedy manner. It performs a forward propagation followed by a second-order back-propagation and collects the linear and quadratic gradients. It is to be noted that there is no weight update taking place during the back-propagation step as the network is already trained.This step is only used to collect the gradients. The algorithm then ranks all the neurons in the network based on their respective gain values and removes the neuron with the least value of the gain. This whole process is repeated until the stopping criterion is met.

[H] optimally trained network, training set A pruned network after applying the Gain-switch algorithm initialize and define stopping criterion stopping criterion is not met $gain = \Delta E_k^2$ perform forward propagation over the training set perform second-order back-propagation without updating weights and collect linear and quadratic gradients rank the remaining neurons based on $gain$ remove the neuron with the least value of $gain$ The proposed Gain-switch pruning algorithm

The advantage of taking a greedy approach is that while removing the neurons, we are take into account the dependencies the neurons might have with one another. A negative value of the gain will indicate a neuron contributing to the output of another neuron in an inhibiting way. As mentioned in **?**, these

3 Experiments

We propose several experiments to compare our discussed algorithm for determining which neurons to switch off with the empirically determined best ordering. First, we do brute force ordering. Using the training data, we get the sum of squared errors for the unaltered network. Then, we switch off one neuron at a time, and do one forward pass through the training data to get the change in the output error. We use this value to rank the neurons. This is the ground truth.

3.1 Correlation of Ranking vs. Ground Truth

3.2 Change in Ground Truth Error vs. Expected Improvement

3.3 Sequential Turn-Off

3.4 Iterative Re-Estimation of Next Best Neuron

4 Experimental Results

5 Conclusions

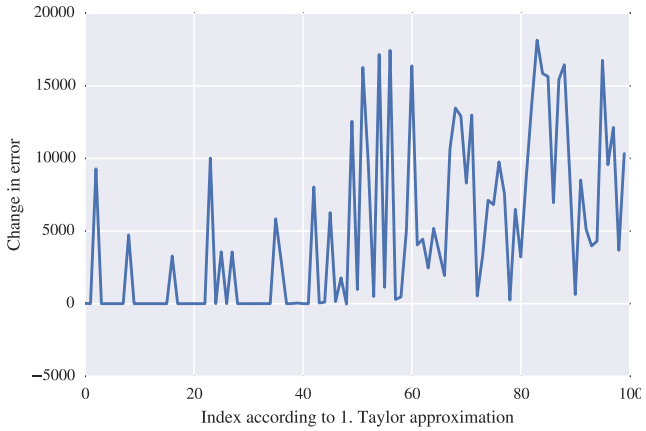

Figure 2: Error when using first Taylor Series approximation.

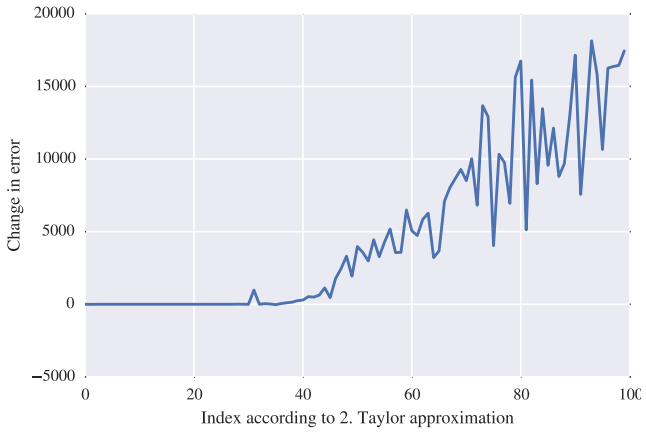

Figure 3: Error when using first Taylor Series approximation.

6 APPENDIX A: SECOND DERIVATIVE BACK-PROPAGATION

Name and network definitions:

$$E = \frac{1}{2}\sum_i (o_i^{(0)} - t_i)^2 \quad o_i^{(m)} = \sigma(x_i^{(m)}) \quad x_i^{(m)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} \quad c_{ji}^{(m)} = w_{ji}^{(m)} o_j^{(m+1)} \tag{12}$$

Superscripts represent the index of the layer of the network in question, with 0 representing the output layer. E is the squared-error network cost function. $o_i^{(m)}$ is the ith output in layer m generated by the activation function σ, which in this paper is is the standard logistic sigmoid. $x_i^{(m)}$ is the weighted sum of inputs to the ith neuron in the mth layer, and $c_{ji}^{(m)}$ is the contribution of the jth neuron in the $m+1$ layer to the input of the ith neuron in the mth layer.

6.1 FIRST AND SECOND DERIVATIVES

The first and second derivatives of the cost function with respect to the outputs:

$$\frac{\partial E}{\partial o_i^{(0)}} = o_i^{(0)} - t_i \tag{13}$$

$$\frac{\partial^2 E}{\partial {o_i^{(0)}}^2} = 1 \tag{14}$$

The first and second derivatives of the sigmoid function in forms depending only on the output:

$$\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right) \tag{15}$$
$$\sigma''(x) = \sigma'(x)\left(1 - 2\sigma(x)\right) \tag{16}$$

The second derivative of the sigmoid is easily derived from the first derivative:

$$\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right) \tag{17}$$
$$\sigma''(x) = \frac{\mathrm{d}}{\mathrm{d}x} \underbrace{\sigma(x)}_{f(x)} \underbrace{\left(1 - \sigma(x)\right)}_{g(x)} \tag{18}$$
$$\sigma''(x) = f'(x)g(x) + f(x)g'(x) \tag{19}$$
$$\sigma''(x) = \sigma'(x)(1 - \sigma(x)) - \sigma(x)\sigma'(x) \tag{20}$$
$$\sigma''(x) = \sigma'(x) - 2\sigma(x)\sigma'(x) \tag{21}$$
$$\sigma''(x) = \sigma'(x)(1 - 2\sigma(x)) \tag{22}$$

And for future convenience:

$$\frac{\mathrm{d}o_i^{(m)}}{\mathrm{d}x_i^{(m)}} = \frac{\mathrm{d}}{\mathrm{d}x_i^{(m)}}\left(o_i^{(m)} = \sigma(x_i^{(m)})\right) \tag{23}$$
$$= \left(o_i^{(m)}\right)\left(1 - o_i^{(m)}\right) \tag{24}$$
$$= \sigma'\left(x_i^{(m)}\right) \tag{25}$$
$$\frac{\mathrm{d}^2 o_i^{(m)}}{\mathrm{d}{x_i^{(m)}}^2} = \frac{\mathrm{d}}{\mathrm{d}x_i^{(m)}}\left(\frac{\mathrm{d}o_i^{(m)}}{\mathrm{d}x_i^{(m)}} = \left(o_i^{(m)}\right)\left(1 - o_i^{(m)}\right)\right) \tag{26}$$
$$= \left(o_i^{(m)}\left(1 - o_i^{(m)}\right)\right)\left(1 - 2o_i^{(m)}\right) \tag{27}$$
$$= \sigma''\left(x_i^{(m)}\right) \tag{28}$$

Derivative of the error with respect to the ith neuron's input $x_i^{(0)}$ in the output layer:

$$\frac{\partial E}{\partial x_i^{(0)}} = \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \tag{29}$$

$$= \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (13)}} \underbrace{\sigma\left(x_i^{(0)}\right)\left(1 - \sigma\left(x_i^{(0)}\right)\right)}_{\text{from (15)}} \tag{30}$$

$$= \left(o_i^{(0)} - t_i\right)\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right) \tag{31}$$

$$= \left(o_i^{(0)} - t_i\right)\sigma'\left(x_i^{(0)}\right) \tag{32}$$

Second derivative of the error with respect to the ith neuron's input $x_i^{(0)}$ in the output layer:

$$\frac{\partial^2 E}{\partial {x_i^{(0)}}^2} = \frac{\partial}{\partial x_i^{(0)}}\left(\frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}}\right) \tag{33}$$

$$= \frac{\partial^2 E}{\partial x_i^{(0)} \partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} + \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial^2 o_i^{(0)}}{\partial {x_i^{(0)}}^2} \tag{34}$$

$$= \frac{\partial^2 E}{\partial x_i^{(0)} \partial o_i^{(0)}} \underbrace{\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right)}_{\text{from (15)}} + \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (13)}} \underbrace{\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right)\left(1 - 2o_i^{(0)}\right)}_{\text{from (16)}} \tag{35}$$

$$\left(\frac{\partial^2 E}{\partial x_i^{(0)} \partial o_i^{(0)}}\right) = \frac{\partial}{\partial x_i^{(0)}} \frac{\partial E}{\partial o_i^{(0)}} = \frac{\partial}{\partial x_i^{(0)}} \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (13)}} = \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} = \underbrace{\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right)}_{\text{from (15)}} \tag{36}$$

$$\frac{\partial^2 E}{\partial {x_i^{(0)}}^2} = \left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right)\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right)\left(1 - 2o_i^{(0)}\right) \tag{37}$$

$$= \left(\sigma'\left(x_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right)\sigma''\left(x_i^{(0)}\right) \tag{38}$$

First derivative of the error with respect to a single input contribution $c_{ji}^{(0)}$ from neuron j to neuron i with weight $w_{ji}^{(0)}$ in the output layer:

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \frac{\partial x_i^{(0)}}{\partial c_{ji}^{(0)}} \tag{39}$$

$$= \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (13)}} \underbrace{\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right)}_{\text{from (15)}} \frac{\partial x_i^{(0)}}{\partial c_{ji}^{(0)}} \tag{40}$$

$$\left(\frac{\partial x_i^{(m)}}{\partial c_{ji}^{(m)}}\right) = \frac{\partial}{\partial c_{ji}^{(m)}}\left(x_i^{(m)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)}\right) = \frac{\partial}{\partial c_{ji}^{(m)}}\left(c_{ji}^{(m)} + k\right) = 1 \tag{41}$$

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \left(o_i^{(0)} - t_i\right)\left(o_i^{(0)}\left(1 - o_i^{(0)}\right)\right) \tag{42}$$

$$= \underbrace{\left(o_i^{(0)} - t_i\right)\sigma'\left(x_i^{(0)}\right)}_{\text{from (32)}} \tag{43}$$

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \frac{\partial E}{\partial x_i^{(0)}} \tag{44}$$

Second derivative of the error with respect to a single input contribution $c_{ji}^{(0)}$:

$$\frac{\partial^2 E}{\partial {c_{ji}^{(0)}}^2} = \frac{\partial}{\partial c_{ji}^{(0)}} \left(\frac{\partial E}{\partial c_{ji}^{(0)}} = \underbrace{\left(o_i^{(0)} - t_i\right) \sigma'\left(x_i^{(0)}\right)}_{\text{from (43)}} \right) \tag{45}$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \left(\sigma\left(x_i^{(0)}\right) - t_i\right) \sigma'\left(x_i^{(0)}\right) \tag{46}$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \left(\sigma\left(\sum_j w_{ji}^{(m)} o_j^{(m+1)}\right) - t_i\right) \sigma'\left(\sum_j w_{ji}^{(m)} o_j^{(m+1)}\right) \tag{47}$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \left(\sigma\left(\sum_j c_{ji}^{(0)}\right) - t_i\right) \sigma'\left(\sum_j c_{ji}^{(0)}\right) \tag{48}$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \underbrace{\left(\sigma\left(c_{ji}^{(0)} + k\right) - t_i\right)}_{f\left(c_{ji}^{(0)}\right)} \underbrace{\sigma'\left(c_{ji}^{(0)} + k\right)}_{g\left(c_{ji}^{(0)}\right)} \tag{49}$$

We now make use of the abbreviations f and g:

$$= f'\left(c_{ji}^{(0)}\right) g\left(c_{ji}^{(0)}\right) + f\left(c_{ji}^{(0)}\right) g'\left(c_{ji}^{(0)}\right) \tag{50}$$

$$= \sigma'\left(c_{ji}^{(0)} + k\right) \sigma'\left(c_{ji}^{(0)} + k\right) + \left(\sigma\left(c_{ji}^{(0)} + k\right) - t_i\right) \sigma''\left(c_{ji}^{(0)} + k\right) \tag{51}$$

$$= \sigma'\left(c_{ji}^{(0)} + k\right)^2 + \left(o_i^{(0)} - t_i\right) \sigma''\left(c_{ji}^{(0)} + k\right) \tag{52}$$

$$\left(c_{ji}^{(0)} + k = \sum_j c_{ji}^{(0)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} = x_i^{(0)}\right) \tag{53}$$

$$\frac{\partial^2 E}{\partial {c_{ji}^{(0)}}^2} = \underbrace{\left(\sigma'\left(x_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right) \sigma''\left(x_i^{(0)}\right)}_{\text{from (38)}} \tag{54}$$

$$\frac{\partial^2 E}{\partial {c_{ji}^{(0)}}^2} = \frac{\partial^2 E}{\partial {x_i^{(0)}}^2} \tag{55}$$

6.1.1 Summary Of Output Layer Derivatives

$$\frac{\partial E}{\partial o_i^{(0)}} = o_i^{(0)} - t_i \qquad \frac{\partial^2 E}{\partial {o_i^{(0)}}^2} = 1 \tag{56}$$

$$\frac{\partial E}{\partial x_i^{(0)}} = \left(o_i^{(0)} - t_i\right) \sigma'\left(x_i^{(0)}\right) \qquad \frac{\partial^2 E}{\partial {x_i^{(0)}}^2} = \left(\sigma'\left(x_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right) \sigma''\left(x_i^{(0)}\right) \tag{57}$$

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \frac{\partial E}{\partial x_i^{(0)}} \qquad \frac{\partial^2 E}{\partial {c_{ji}^{(0)}}^2} = \frac{\partial^2 E}{\partial {x_i^{(0)}}^2} \tag{58}$$

6.1.2 HIDDEN LAYER DERIVATIVES

The first derivative of the error with respect to a neuron with output $o_j^{(1)}$ in the first hidden layer, summing over all partial derivative contributions from the output layer:

$$\frac{\partial E}{\partial o_j^{(1)}} = \sum_i \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \frac{\partial x_i^{(0)}}{\partial c_{ji}^{(0)}} \frac{\partial c_{ji}^{(0)}}{\partial o_j^{(1)}} = \sum_i \underbrace{\left(o_i^{(0)} - t_i\right) \sigma'\left(x_i^{(0)}\right)}_{\text{from (32)}} w_{ji}^{(0)} \tag{59}$$

$$\frac{\partial c_{ji}^{(m)}}{\partial o_j^{(m+1)}} = \frac{\partial}{\partial o_j^{(m+1)}} \left(c_{ji}^{(m)} = w_{ji}^{(m)} o_j^{(m+1)}\right) = w_{ji}^{(m)} \tag{60}$$

$$\frac{\partial E}{\partial o_j^{(1)}} = \sum_i \frac{\partial E}{\partial x_i^{(0)}} w_{ji}^{(0)} \tag{61}$$

Note that this equation does not depend on the specific form of $\frac{\partial E}{\partial x_i^{(0)}}$, whether it involves a sigmoid or any other activation function. We can therefore replace the specific indexes with general ones, and use this equation in the future.

$$\frac{\partial E}{\partial o_j^{(m+1)}} = \sum_i \frac{\partial E}{\partial x_i^{(m)}} w_{ji}^{(m)} \tag{62}$$

The second derivative of the error with respect to a neuron with output $o_j^{(1)}$ in the first hidden layer:

$$\frac{\partial^2 E}{\partial {o_j^{(1)}}^2} = \frac{\partial}{\partial o_j^{(1)}} \frac{\partial E}{\partial o_j^{(1)}} \tag{63}$$

$$= \frac{\partial}{\partial o_j^{(1)}} \sum_i \frac{\partial E}{\partial x_i^{(0)}} w_{ji}^{(0)} \tag{64}$$

$$= \frac{\partial}{\partial o_j^{(1)}} \sum_i \left(o_i^{(0)} - t_i\right) \sigma'\left(x_i^{(0)}\right) w_{ji}^{(0)} \tag{65}$$

If we now make use of the fact, that $o_i^{(0)} = \sigma\left(x_i^{(0)}\right) = \sigma\left(\sum_j \left(w_{ji}^{(0)} o_j^{(1)}\right)\right)$, we can evaluate the expression further.

$$\frac{\partial^2 E}{\partial {o_j^{(1)}}^2} = \frac{\partial}{\partial o_j^{(1)}} \sum_i \underbrace{\left(\sigma\left(\sum_j w_{ji}^{(0)} o_j^{(1)}\right) - t_i\right)}_{f\left(o_j^{(1)}\right)} \underbrace{\sigma'\left(\sum_j w_{ji}^{(0)} o_j^{(1)}\right)}_{g\left(o_j^{(1)}\right)} w_{ji}^{(0)} \tag{66}$$

$$= \sum_i \left(f'\left(o_j^{(1)}\right) g\left(o_j^{(1)}\right) + f\left(o_j^{(1)}\right) g'\left(o_j^{(1)}\right)\right) \tag{67}$$

$$= \sum_i \sigma'\left(\sum_j w_{ji}^{(0)} o_j^{(1)}\right) w_{ji}^{(0)} \; \sigma'\left(\sum_j w_{ji}^{(0)} o_j^{(1)}\right) w_{ji}^{(0)} + \ldots \tag{68}$$

$$\sum_i \left(\sigma\left(\sum_j w_{ji}^{(0)} o_j^{(1)}\right) - t_i\right) \sigma''\left(\sum_j w_{ji}^{(0)} o_j^{(1)}\right) \left(w_{ji}^{(0)}\right)^2 \tag{69}$$

$$= \sum_i \left(\left(\sigma'\left(x_i^{(0)}\right)\right)^2 \left(w_{ji}^{(0)}\right)^2 + \left(o_i^{(0)} - t_i\right) \sigma''\left(x_i^{(0)}\right) \left(w_{ji}^{(0)}\right)^2\right) \tag{70}$$

$$= \sum_i \underbrace{\left(\left(\sigma'\left(x_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right) \sigma''\left(x_i^{(0)}\right)\right)}_{\text{from (38)}} \left(w_{ji}^{(0)}\right)^2 \tag{71}$$

Summing up, we obtain the more general expression:

$$\frac{\partial^2 E}{\partial {o_j^{(1)}}^2} = \sum_i \frac{\partial^2 E}{\partial {x_i^{(0)}}^2} \left(w_{ji}^{(0)}\right)^2 \tag{72}$$

Note that the equation in (72) does not depend on the form of $\frac{\partial^2 E}{\partial {x_x^{(0)}}^2}$, which means we can replace the specific indexes with general ones:

$$\frac{\partial^2 E}{\partial {o_j^{(m+1)}}^2} = \sum_i \frac{\partial^2 E}{\partial {x_i^{(m)}}^2} \left(w_{ji}^{(m)}\right)^2 \tag{73}$$

At this point we are beginning to see the recursion in the form of the 2nd derivative terms which can be thought of analogously to the first derivative recursion which is central to the back-propagation algorithm. The formulation above which makes specific reference to layer indexes also works in the general case.

Consider the ith neuron in any layer m with output $o_i^{(m)}$ and input $x_i^{(m)}$. The first and second derivatives of the error E with respect to this neuron's *input* are:

$$\frac{\partial E}{\partial x_i^{(m)}} = \frac{\partial E}{\partial o_i^{(m)}} \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} \tag{74}$$

$$\frac{\partial^2 E}{\partial {x_i^{(m)}}^2} = \frac{\partial}{\partial x_i^{(m)}} \frac{\partial E}{\partial x_i^{(m)}} \tag{75}$$

$$= \frac{\partial}{\partial x_i^{(m)}} \left(\frac{\partial E}{\partial o_i^{(m)}} \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} \right) \tag{76}$$

$$= \frac{\partial^2 E}{\partial x_i^{(m)} \partial o_i^{(m)}} \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} + \frac{\partial E}{\partial o_i^{(m)}} \frac{\partial^2 o_i^{(m)}}{\partial {x_i^{(m)}}^2} \tag{77}$$

$$= \frac{\partial}{\partial o_i^{(m)}} \left(\frac{\partial E}{\partial x_i^{(m)}} = \frac{\partial E}{\partial o_i^{(m)}} \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} \right) \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} + \frac{\partial E}{\partial o_i^{(m)}} \sigma'' \left(x_i^{(m)}\right) \tag{78}$$

$$= \frac{\partial^2 E}{\partial {o_i^{(m)}}^2} \left(\frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} \right) + \frac{\partial E}{\partial o_i^{(m)}} \sigma'' \left(x_i^{(m)}\right) \tag{79}$$

$$\frac{\partial^2 E}{\partial {x_i^{(m)}}^2} = \frac{\partial^2 E}{\partial {o_i^{(m)}}^2} \left(\sigma' \left(x_i^{(m)}\right)\right)^2 + \frac{\partial E}{\partial o_i^{(m)}} \sigma'' \left(x_i^{(m)}\right) \tag{80}$$

Note the form of this equation is the general form of what was derived for the output layer in (38). Both of the above first and second terms are easily computable and can be stored as we propagate back from the output of the network to the input. With respect to the output layer, the first and second derivative terms have already been derived above. In the case of the $m+1$ hidden layer during back propagation, there is a summation of terms calculated in the mth layer. For the first derivative, we have this from (62).

$$\frac{\partial E}{\partial o_j^{(m+1)}} = \sum_i \frac{\partial E}{\partial x_i^{(m)}} w_{ji}^{(m)} \tag{81}$$

And the second derivative for the jth neuron in the $m+1$ layer:

$$\frac{\partial^2 E}{\partial {x_j^{(m+1)}}^2} = \frac{\partial^2 E}{\partial {o_j^{(m+1)}}^2} \left(\sigma' \left(x_j^{(m+1)}\right)\right)^2 + \frac{\partial E}{\partial o_j^{(m+1)}} \sigma'' \left(x_j^{(m+1)}\right) \tag{82}$$

We can replace both derivative terms with the forms which depend on the previous layer:

$$\frac{\partial^2 E}{\partial {x_j^{(m+1)}}^2} = \underbrace{\sum_i \frac{\partial^2 E}{\partial {x_i^{(0)}}^2} \left(w_{ji}^{(0)}\right)^2}_{\text{from (73)}} \left(\sigma' \left(x_j^{(m+1)}\right)\right)^2 + \underbrace{\sum_i \frac{\partial E}{\partial x_i^{(m)}} w_{ji}^{(m)}}_{\text{from (62)}} \sigma'' \left(x_j^{(m+1)}\right) \tag{83}$$

And this horrible mouthful of an equation gives you a general form for any neuron in the jth position of the $m+1$ layer. Taking very careful note of the indexes, this can be more or less translated painlessly to code. You are welcome, world.

6.1.3 SUMMARY OF HIDDEN LAYER DERIVATIVES

$$\frac{\partial E}{\partial o_j^{(m+1)}} = \sum_i \frac{\partial E}{\partial x_i^{(m)}} w_{ji}^{(m)} \qquad \frac{\partial^2 E}{\partial o_j^{(m+1)^2}} = \sum_i \frac{\partial^2 E}{\partial x_i^{(m)^2}} \left(w_{ji}^{(m)}\right)^2 \tag{84}$$

$$\frac{\partial E}{\partial x_i^{(m)}} = \frac{\partial E}{\partial o_i^{(m)}} \frac{\partial o_i^{(m)}}{\partial x_i^{(m)}} \tag{85}$$

$$\frac{\partial^2 E}{\partial x_j^{(m+1)^2}} = \frac{\partial^2 E}{\partial o_j^{(m+1)^2}} \left(\sigma'\left(x_j^{(m+1)}\right)\right)^2 + \frac{\partial E}{\partial o_j^{(m+1)}} \sigma''\left(x_j^{(m+1)}\right) \tag{86}$$

$$\frac{\partial E}{\partial c_{ji}^{(m)}} = ... \qquad \frac{\partial^2 E}{\partial c_{ji}^{(m)^2}} = ... \tag{87}$$