

# 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.

### 2.2 LINEAR APPROXIMATION APPROACH (THE GAIN-SWITCH)

Figure ????????? shows a neuron in a neural network

$$z = O \cdot g \frac{dz}{dO} = g \quad \frac{dE}{dO} = \frac{dE}{dz} \cdot \frac{dz}{dO} = \frac{dE}{dz} \cdot g \frac{dE}{dg} = \frac{dE}{dz} \cdot \frac{dz}{dg} = \frac{dE}{dz} \cdot O \quad (1)$$

### 2.2.1 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  $E_{\text{total}}$ . Then,

$$E_{\text{total}} = \sum_n E_n, \quad (2)$$

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,

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

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 3, the contribution  $E_n(0)$  of this neuron, then becomes:

$$\hat{E}_n(0) \approx E_n(O_k) - O_k \cdot \left. \frac{\partial E_n}{\partial O} \right|_{O_k} + 0.5 \cdot O_k^2 \cdot \left. \frac{\partial^2 E_n}{\partial O^2} \right|_{O_k} \quad (4)$$

Replacing  $O$  by  $O_k$  in equation 3 shows us that the error is approximated perfectly by equation 3 at  $O_k$ . Using this and equation 4 we get:

$$\Delta E_{n,k} = \hat{E}_n(0) - \hat{E}_n(O_k) = -O_k \cdot \left. \frac{\partial E_n}{\partial O} \right|_{O_k} + 0.5 \cdot O_k^2 \cdot \left. \frac{\partial^2 E_n}{\partial O^2} \right|_{O_k}, \quad (5)$$

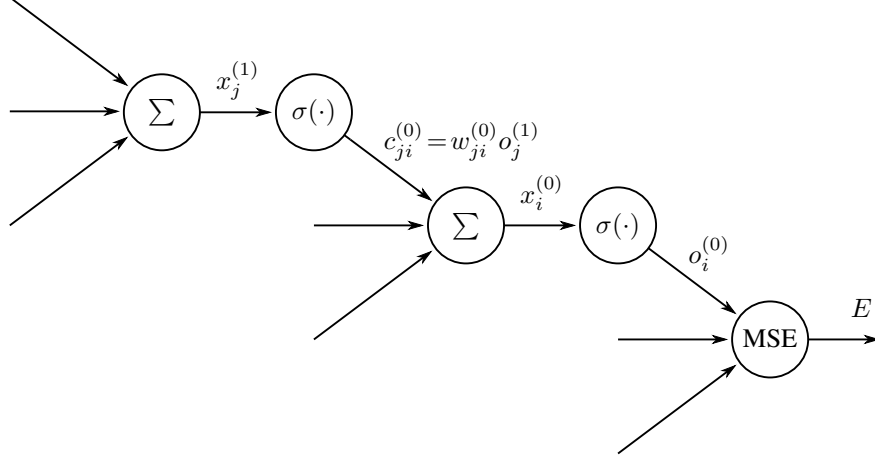
where  $\Delta E_{n,k}$  is the change in the total error of the network given a validation instance  $n$ , when exactly one neuron ( $k$ ) is turned off.

### 2.3 QUADRATIC APPROXIMATION APPROACH

We define the following network terminology here which will be used in this section and all subsequent sections unless stated otherwise. Figure 1 can be used as a reference to the terminology defined here:

$$E = \frac{1}{2} \sum_i (o_i^{(0)} - t_i)^2 \quad o_i^{(m)} = \sigma(x_i^{(m)}) \quad x_i^{(m)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} \quad c_{ji}^{(m)} = w_{ji}^{(m)} o_j^{(m+1)} \quad (6)$$

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  $i$ th output in layer  $m$  generated by the activation function  $\sigma$ , which in this paper is the standard logistic sigmoid.  $x_i^{(m)}$  is the weighted sum of inputs to the  $i$ th neuron in the  $m$ th layer, and  $c_{ji}^{(m)}$  is the contribution of the  $j$ th neuron in the  $(m+1)$ th layer to the input of the  $i$ th neuron in the  $m$ th layer.  $w_{ji}^{(m)}$  is the weight between the  $j$ th neuron in the  $(m+1)$ th layer and the  $i$ th neuron in the  $m$ th layer.



As seen in equation 5,  $\Delta E_{n,k}$  which can now be represented as  $\Delta E_k$  is the quadratic approximation of the change in error due to the  $k$ th neuron being turned off which becomes the basis of our ranking in this approach. The quadratic term in equation 5 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 5 in our new terminology here:

$$\Delta E_k = -o_k \cdot \left. \frac{\partial E}{\partial o_j^{(m+1)}} \right|_{o_k} + 0.5 \cdot o_k^2 \cdot \left. \frac{\partial^2 E}{\partial o_j^{(m+1)2}} \right|_{o_k} \quad (7)$$

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:

$$\frac{\partial^2 E}{\partial o_j^{(m+1)2}} = \sum_i \frac{\partial^2 E}{\partial c_{ji}^{(m)2}} \left( w_{ji}^{(m)} \right)^2 \quad (8)$$

Here,  $c_{ji}^{(m)}$  is one of the component terms of  $x_i^{(m)}$ , as follows from the equations in 6. Hence, it can be easily proved that (full proof in appendix):

$$\frac{\partial^2 E}{\partial c_{ji}^{(m)2}} = \frac{\partial^2 E}{\partial x_i^{(m)2}} \quad (9)$$

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.

$$\frac{\partial^2 E}{\partial x_i^{(m)2}} = \frac{\partial^2 E}{\partial o_i^{(m)2}} \left( \sigma' \left( x_i^{(m)} \right) \right)^2 + \frac{\partial E}{\partial o_i^{(m)}} \sigma'' \left( x_i^{(m)} \right) \quad (10)$$

### 3 EXPERIMENTAL RESULTS

### 4 CONCLUSIONS

## 5 APPENDIX A: SECOND DERIVATIVE BACK-PROPAGATION

Name and network definitions:

$$E = \frac{1}{2} \sum_i (o_i^{(0)} - t_i)^2 \quad o_i^{(m)} = \sigma(x_i^{(m)}) \quad x_i^{(m)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} \quad c_{ji}^{(m)} = w_{ji}^{(m)} o_j^{(m+1)} \quad (11)$$

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  $i$ th output in layer  $m$  generated by the activation function  $\sigma$ , which in this paper is the standard logistic sigmoid.  $x_i^{(m)}$  is the weighted sum of inputs to the  $i$ th neuron in the  $m$ th layer, and  $c_{ji}^{(m)}$  is the contribution of the  $j$ th neuron in the  $m+1$  layer to the input of the  $i$ th neuron in the  $m$ th layer.

### 5.1 FIRST AND SECOND DERIVATIVES

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

$$\frac{\partial E}{\partial o_i^{(0)}} = o_i^{(0)} - t_i \quad (12)$$

$$\frac{\partial^2 E}{\partial o_i^{(0)2}} = 1 \quad (13)$$

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

$$\sigma'(x) = \sigma(x) (1 - \sigma(x)) \quad (14)$$

$$\sigma''(x) = \sigma'(x) (1 - 2\sigma(x)) \quad (15)$$

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

$$\sigma'(x) = \sigma(x) (1 - \sigma(x)) \quad (16)$$

$$\sigma''(x) = \frac{d}{dx} \underbrace{\sigma(x)}_{f(x)} \underbrace{(1 - \sigma(x))}_{g(x)} \quad (17)$$

$$\sigma''(x) = f'(x)g(x) + f(x)g'(x) \quad (18)$$

$$\sigma''(x) = \sigma'(x)(1 - \sigma(x)) - \sigma(x)\sigma'(x) \quad (19)$$

$$\sigma''(x) = \sigma'(x) - 2\sigma(x)\sigma'(x) \quad (20)$$

$$\sigma''(x) = \sigma'(x)(1 - 2\sigma(x)) \quad (21)$$

And for future convenience:

$$\frac{do_i^{(m)}}{dx_i^{(m)}} = \frac{d}{dx_i^{(m)}} \left( o_i^{(m)} = \sigma(x_i^{(m)}) \right) \quad (22)$$

$$= \left( o_i^{(m)} \right) \left( 1 - o_i^{(m)} \right) \quad (23)$$

$$= \sigma' \left( x_i^{(m)} \right) \quad (24)$$

$$\frac{d^2 o_i^{(m)}}{dx_i^{(m)2}} = \frac{d}{dx_i^{(m)}} \left( \frac{do_i^{(m)}}{dx_i^{(m)}} = \left( o_i^{(m)} \right) \left( 1 - o_i^{(m)} \right) \right) \quad (25)$$

$$= \left( o_i^{(m)} \left( 1 - o_i^{(m)} \right) \right) \left( 1 - 2o_i^{(m)} \right) \quad (26)$$

$$= \sigma'' \left( x_i^{(m)} \right) \quad (27)$$

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

$$\frac{\partial E}{\partial x_i^{(0)}} = \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \quad (28)$$

$$= \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (12)}} \underbrace{\sigma\left(x_i^{(0)}\right) \left(1 - \sigma\left(x_i^{(0)}\right)\right)}_{\text{from (14)}} \quad (29)$$

$$= \left(o_i^{(0)} - t_i\right) \left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right) \quad (30)$$

$$= \left(o_i^{(0)} - t_i\right) \sigma'\left(x_i^{(0)}\right) \quad (31)$$

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

$$\frac{\partial^2 E}{\partial x_i^{(0)2}} = \frac{\partial}{\partial x_i^{(0)}} \left( \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \right) \quad (32)$$

$$= \frac{\partial^2 E}{\partial x_i^{(0)} \partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} + \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial^2 o_i^{(0)}}{\partial x_i^{(0)2}} \quad (33)$$

$$= \frac{\partial^2 E}{\partial x_i^{(0)} \partial o_i^{(0)}} \underbrace{\left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right)}_{\text{from (14)}} + \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (12)}} \underbrace{\left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right) \left(1 - 2o_i^{(0)}\right)}_{\text{from (15)}} \quad (34)$$

$$\left( \frac{\partial^2 E}{\partial x_i^{(0)} \partial o_i^{(0)}} \right) = \frac{\partial}{\partial x_i^{(0)}} \frac{\partial E}{\partial o_i^{(0)}} = \frac{\partial}{\partial x_i^{(0)}} \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (12)}} = \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} = \underbrace{\left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right)}_{\text{from (14)}} \quad (35)$$

$$\frac{\partial^2 E}{\partial x_i^{(0)2}} = \left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right) \left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right) \left(1 - 2o_i^{(0)}\right) \quad (36)$$

$$= \left(\sigma'\left(x_i^{(0)}\right)\right)^2 + \left(o_i^{(0)} - t_i\right) \sigma''\left(x_i^{(0)}\right) \quad (37)$$

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:

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \frac{\partial x_i^{(0)}}{\partial c_{ji}^{(0)}} \quad (38)$$

$$= \underbrace{\left(o_i^{(0)} - t_i\right)}_{\text{from (12)}} \underbrace{\left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right)}_{\text{from (14)}} \frac{\partial x_i^{(0)}}{\partial c_{ji}^{(0)}} \quad (39)$$

$$\left( \frac{\partial x_i^{(m)}}{\partial c_{ji}^{(m)}} \right) = \frac{\partial}{\partial c_{ji}^{(m)}} \left( x_i^{(m)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} \right) = \frac{\partial}{\partial c_{ji}^{(m)}} \left( c_{ji}^{(m)} + k \right) = 1 \quad (40)$$

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \left(o_i^{(0)} - t_i\right) \left(o_i^{(0)} \left(1 - o_i^{(0)}\right)\right) \quad (41)$$

$$= \underbrace{\left(o_i^{(0)} - t_i\right) \sigma'\left(x_i^{(0)}\right)}_{\text{from (31)}} \quad (42)$$

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \frac{\partial E}{\partial x_i^{(0)}} \quad (43)$$

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

$$\frac{\partial^2 E}{\partial c_{ji}^{(0)2}} = \frac{\partial}{\partial c_{ji}^{(0)}} \left( \underbrace{\frac{\partial E}{\partial c_{ji}^{(0)}}}_{\text{from (42)}} = \left( o_i^{(0)} - t_i \right) \sigma' \left( x_i^{(0)} \right) \right) \quad (44)$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \left( \sigma \left( x_i^{(0)} \right) - t_i \right) \sigma' \left( x_i^{(0)} \right) \quad (45)$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \left( \sigma \left( \sum_j w_{ji}^{(m)} o_j^{(m+1)} \right) - t_i \right) \sigma' \left( \sum_j w_{ji}^{(m)} o_j^{(m+1)} \right) \quad (46)$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \left( \sigma \left( \sum_j c_{ji}^{(0)} \right) - t_i \right) \sigma' \left( \sum_j c_{ji}^{(0)} \right) \quad (47)$$

$$= \frac{\partial}{\partial c_{ji}^{(0)}} \underbrace{\left( \sigma \left( c_{ji}^{(0)} + k \right) - t_i \right)}_{f(c_{ji}^{(0)})} \underbrace{\sigma' \left( c_{ji}^{(0)} + k \right)}_{g(c_{ji}^{(0)})} \quad (48)$$

$$= f' \left( c_{ji}^{(0)} \right) g \left( c_{ji}^{(0)} \right) + f \left( c_{ji}^{(0)} \right) g' \left( c_{ji}^{(0)} \right) \quad (49)$$

$$= \sigma' \left( c_{ji}^{(0)} + k \right) \sigma' \left( c_{ji}^{(0)} + k \right) + \left( \sigma \left( c_{ji}^{(0)} + k \right) - t_i \right) \sigma'' \left( c_{ji}^{(0)} + k \right) \quad (50)$$

$$= \sigma' \left( c_{ji}^{(0)} + k \right)^2 + \left( o_i^{(0)} - t_i \right) \sigma'' \left( c_{ji}^{(0)} + k \right) \quad (51)$$

$$\left( c_{ji}^{(0)} + k = \sum_j c_{ji}^{(0)} = \sum_j w_{ji}^{(m)} o_j^{(m+1)} = x_i^{(0)} \right) \quad (52)$$

$$\frac{\partial^2 E}{\partial c_{ji}^{(0)2}} = \underbrace{\left( \sigma' \left( x_i^{(0)} \right) \right)^2 + \left( o_i^{(0)} - t_i \right) \sigma'' \left( x_i^{(0)} \right)}_{\text{from (37)}} \quad (53)$$

$$\frac{\partial^2 E}{\partial c_{ji}^{(0)2}} = \frac{\partial^2 E}{\partial x_i^{(0)2}} \quad (54)$$

### 5.1.1 SUMMARY OF OUTPUT LAYER DERIVATIVES

$$\frac{\partial E}{\partial o_i^{(0)}} = o_i^{(0)} - t_i \quad \frac{\partial^2 E}{\partial o_i^{(0)2}} = 1 \quad (55)$$

$$\frac{\partial E}{\partial x_i^{(0)}} = \left( o_i^{(0)} - t_i \right) \sigma' \left( x_i^{(0)} \right) \quad \frac{\partial^2 E}{\partial x_i^{(0)2}} = \left( \sigma' \left( x_i^{(0)} \right) \right)^2 + \left( o_i^{(0)} - t_i \right) \sigma'' \left( x_i^{(0)} \right) \quad (56)$$

$$\frac{\partial E}{\partial c_{ji}^{(0)}} = \frac{\partial E}{\partial x_i^{(0)}} \quad \frac{\partial^2 E}{\partial c_{ji}^{(0)2}} = \frac{\partial^2 E}{\partial x_i^{(0)2}} \quad (57)$$

## 5.1.2 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:

$$\frac{\partial E}{\partial o_j^{(1)}} = \sum_i \frac{\partial E}{\partial o_i^{(0)}} \frac{\partial o_i^{(0)}}{\partial x_i^{(0)}} \frac{\partial x_i^{(0)}}{\partial c_{ji}^{(0)}} \frac{\partial c_{ji}^{(0)}}{\partial o_j^{(1)}} = \sum_i \underbrace{\left( o_i^{(0)} - t_i \right) \sigma' \left( x_i^{(0)} \right)}_{\text{from (31)}} w_{ji}^{(0)} \quad (58)$$

$$\frac{\partial c_{ji}^{(m)}}{\partial o_j^{(m+1)}} = \frac{\partial}{\partial o_j^{(m+1)}} \left( c_{ji}^{(m)} = w_{ji}^{(m)} o_j^{(m+1)} \right) = w_{ji}^{(m)} \quad (59)$$

$$\frac{\partial E}{\partial o_j^{(1)}} = \sum_i \frac{\partial E}{\partial x_i^{(0)}} w_{ji}^{(0)} \quad (60)$$

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.

$$\frac{\partial E}{\partial o_j^{(m+1)}} = \sum_i \frac{\partial E}{\partial x_i^{(m)}} w_{ji}^{(m)} \quad (61)$$

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

$$\frac{\partial^2 E}{\partial o_j^{(1)2}} = \frac{\partial}{\partial o_j^{(1)}} \frac{\partial E}{\partial o_j^{(1)}} \quad (62)$$

$$= \frac{\partial}{\partial o_j^{(1)}} \sum_i \frac{\partial E}{\partial x_i^{(0)}} w_{ji}^{(0)} \quad (63)$$

$$= \frac{\partial}{\partial o_j^{(1)}} \sum_i \left( o_i^{(0)} - t_i \right) \sigma' \left( x_i^{(0)} \right) w_{ji}^{(0)} \quad (64)$$

$$o_i^{(0)} = \sigma \left( x_i^{(0)} \right) = \sigma \left( \sum_j \left( w_{ji}^{(0)} o_j^{(1)} \right) \right) \quad (65)$$

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

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

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

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

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

$$= \sum_i \underbrace{\left( \left( \sigma' \left( x_i^{(0)} \right) \right)^2 + \left( o_i^{(0)} - t_i \right) \sigma'' \left( x_i^{(0)} \right) \right)}_{\text{from (37)}} \left( w_{ji}^{(0)} \right)^2 \quad (71)$$

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

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:

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

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  $i$ th 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:

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



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

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

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

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

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

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

Note the form of this equation is the general form of what was derived for the output layer in (37). 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  $m$ th layer. For the first derivative, we have this from (61).

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

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

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

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

$$\frac{\partial^2 E}{\partial x_j^{(m+1)2}} = \underbrace{\sum_i \frac{\partial^2 E}{\partial x_i^{(0)2}} \left( w_{ji}^{(0)} \right)^2}_{\text{from (73)}} \left( \sigma'(x_j^{(m+1)}) \right)^2 + \underbrace{\sum_i \frac{\partial E}{\partial x_i^{(m)}} w_{ji}^{(m)}}_{\text{from (61)}} \sigma''(x_j^{(m+1)}) \quad (83)$$

And this horrible mouthful of an equation gives you a general form for any neuron in the  $j$ th 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.