Brain-Based Computing:
Using Powerful Brain Strategies to Improve Machine Learning

DRAFT Chapter X
Simple Feedforward Neural Networks

Alianna J. Maren
Northwestern University School of Professional Studies

Chapter Draft: 2017-01-01

1.1 The Multilayer Perceptron

1.1.1 The Transfer Function

We'll begin with one of the most commonly-used transfer functions, called the *sigmoid function*, because it produces an S-shaped curve.

$$y = \frac{1}{1 + exp(-\alpha x)} = (1 + exp(-\alpha x))^{-1}. \tag{1.1}$$

See this transfer function in Fig. 1.1.

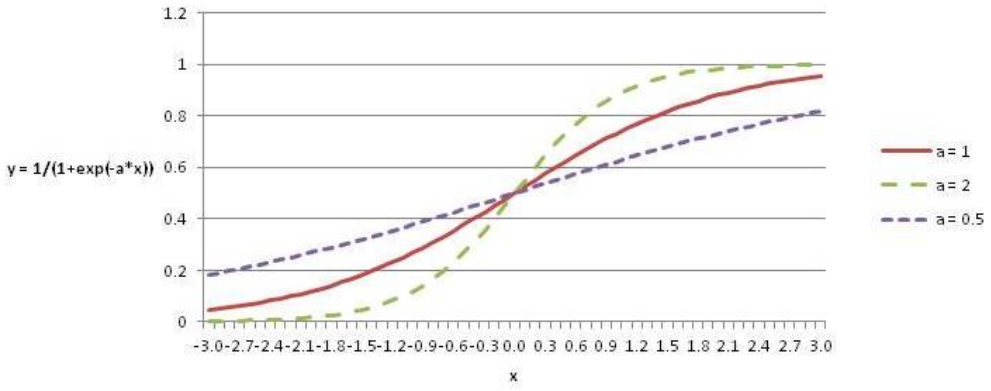

Figure 1.1: The simple transfer function.

Before we go further, let's make a few mathematical comments on this transfer function. Its purpose, in the world of neural networks, is to scale the inputs into a given node to be within a reasonable range for the outputs. That means that even if we put very large negative or positive values into the node, we want the output to be much more limited; we're choosing a transfer function here that will limit the outputs to be between 0 and 1.

We want our transfer function to be smoothly continuous; no discontinuities (sudden jumps). Also, we want its overall behavior to be consistent; we don't want something that goes up and then goes down. In other words, we want a function that is *monotonic*. In our particular case, we want a function that is *monotonically increasing*; that is, it starts with small values and moves smoothly to progressively larger values. That is exactly what we get with this particular function.

We notice that this particular function is *asymptotic* at both extremes, which is something that we want. This fulfills our previously-stated requirement, that we want the outputs to be limited within a certain range. By asymptotic, we mean that the function approaches certain limits, but never quite reaches them. When x is a very large negative, the value for y is a very small positive number; close to (but never quite reaching) zero. (As $x \to -\infty, y \to +0$.) Further, when x is a very large positive, the value for y approaches (but never quite reaches) 1. (As $x \to +\infty, y \to 1$.)

1.1.2 The Derivative of the Transfer Function

We'll need to compute the derivative of the transfer function as part of the backpropagation algorithm. We'll do this now, and mentally stash the results for future use.

The derivative is computed as:

$$\begin{aligned} \partial y/\partial x &= -(-\alpha)(1 + exp(-\alpha x))^{-2} exp(-\alpha x) \\ &= \alpha(1 + exp(-\alpha x))^{-2} exp(-\alpha x). \end{aligned} \tag{1.2}$$

As a side note, those of you who are mathematically inclined will note that we've been using the terminology of "partial derivatives" throughout, when really, there is only a single variable (x) in the equations that we have just used. The reason for using the partial derivative nomenclature is that we'll shortly be deriving the entire backpropagation algorithm, which makes extensive use of the chain rule, and which also is expressed in terms of partial derivatives. Thus, we're just setting ourselves up for an easier usage later.

We want to simplify the results that we've just obtained, and to do so, let's first study a substitution. We write an equation with the same form, but with a simpler variable; we'll let $c = exp(-\alpha x)$. Thus, our simpler equation reads

$$\partial y/\partial x = (1 + c)^{-2} c = \frac{c}{(1 + c)^2}. \tag{1.3}$$

Note that we've dropped the leading coefficient α; again, that's just so we can concentrate on working with the form of the equation; we'll put it back in before we're done.

The mathematics of this equation are such that (if you know what you're doing in advance), you can express the derivative of the transfer function in

terms of the original transfer function. To do this, we first note that the denominator of the original transfer function is now squared, compared to the original, as was shown in Eqn. 1.1.

As an intermediate step, we rewrite the equation so that the denominator is in two separate terms

$$\partial y/\partial x = \frac{1}{1+c} * \frac{c}{1+c}. \tag{1.4}$$

The first term on the right-hand-side is indeed the original transfer function; recall that

$$y = \frac{1}{1+exp(-\alpha x)} = \frac{1}{1+c}. \tag{1.5}$$

Thus, we'll simply substitute y for the first term on the right in Eqn. 1.4 to obtain

$$\partial y/\partial x = y * \frac{c}{1+c}. \tag{1.6}$$

Our job is half-done; we now want to express the last term on the right of Eqn. 1.6 in terms of y (the original transfer function), and not c. To do this, we rewrite the numerator of this last term, both adding and subtracting 1; because we're not changing the total value of the numerator, this is allowed.

$$\partial y/\partial x = y\Big[\frac{1+c-1}{1+c}\Big]. \tag{1.7}$$

Now we can split the resulting last term on the right-hand-side into two parts:

$$\partial y/\partial x = y\Big[\frac{1+c}{1+c} - \frac{1}{1+c}\Big]. \tag{1.8}$$

We simplify the first term within the brackets, and for the second, we notice that we have once again obtained the expression for y, so that we obtain

$$\partial y/\partial x = y\big[1-y\big]. \tag{1.9}$$

Finally, we re-introduce the term α, which we had dropped earlier in order to focus on the simplifications. (This was when we went from Eqn. 1.2 to Eqn. 1.3.) This gives us our final result

$$\partial y/\partial x = \alpha y\big[1 - y\big]. \tag{1.10}$$

We've now obtained the partial derivative of the transfer function in terms of its dependence on its (single) variable, x.

Before we go further, let's have a quick look at how this derivative compares with the original transfer function. Before we look at the actual graph, let's refer back to Fig. 1.1.

The derivative of a function is its slope. Since our transfer function is monotonically increasing, its slope is always positive. Thus, like the transfer function itself, we expect the derivative to also be consistently positive.

Further, as we can see in Fig. 1.1, the slope (rate of increase) starts off small (when x is a very large negative value), and also ends small (when x is a very large positive value). Further, when we check out the middle realm of this transfer function, we see that the rate of increase is greatest when $x = 0$, and at this point, the slope is 1.

Thus, we expect that the derivative of transfer function will be approximately bell-shaped; it will have very small positive values to the far left and the far right, and achieve a value of 1 in the center.

We notice also that having a slope of 1 when $x = 0$ is entirely dependent on the values for α; we can see from Fig. 1.1 that the slope increases faster when $\alpha > 1$, and more slowly when $\alpha < 1$. Thus, we expect that the overall function shape will be the same if we increase α, but the maximum (centered still at $x = 0$) will increase, and if we lower α, then the curve will spread out more and the maximum will decrease.

We can now view the derivative of the sigmoid transfer function in Fig. 1.2.

We notice that it behaves exactly as we predicted.

1.1.3 Feedforward Propagation of Neuron Activations

This section needs to be written

1.1.4 Backpropagation of Error: Part 1

We adjust the weights so that the feedforward propagation of neuron activations yields results (in the output layer) that are close to the desired outputs.

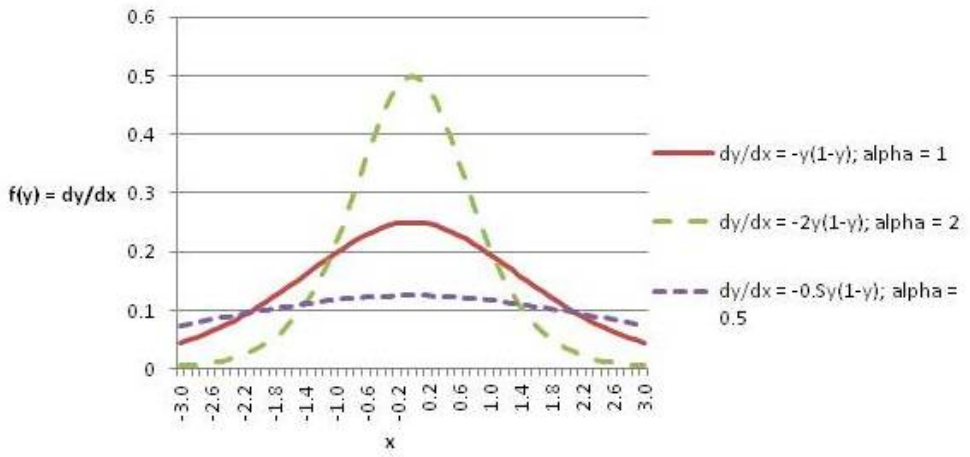

Figure 1.2: The derivative of the simple sigmoid transfer function .

We define the error term as:

$$SSE = \frac{1}{2}\sum_{o=1}^{O}(Desired_o - Actual_o)^2, \tag{1.11}$$

The multiplying factor of 1/2 is introduced so that when we take the derivative of this term, the factor of two that results from the derivative of a squared term multiplies the factor of 1/2, neutralizing the impact of constants.

The specific error associated with any given output node is:

$$Error_o = Desired_o - Actual_o \tag{1.12}$$

Notice that we square these errors before summing across all the output nodes in the network.

Our goal is to adjust the weights to as to minimize the total error produced in the network. As we do so, we will also be minimizing the errors associated with any given specific output node.

Because the things that we can adjust are the actual weights themselves, what we want to compute is the dependence of the error on the weights. Specifically, we want two entirely different sets of dependencies:

- The dependence of the error on the weights connecting hidden to output nodes, and

- The dependence of the error on the weights connecting the input to the hidden nodes.

Both of these calculations use the chain rule from calculus. Also, since any one error term is a function of multiple different inputs (to the corresponding output node), any one error term has multiple dependency paths. Thus, we do our computations using *partial* derivatives; we seek to trace the unique and specific dependence that the error has on each contributing source.

We start by computing the dependence of the Summed Squared Error, SSE, on a specific hidden-to-output weight.

$$\frac{\partial SSE}{\partial v_{h,o}} = \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial v_{h,o}} = \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial A_o}\frac{\partial A_o}{\partial v_{h,o}} \tag{1.13}$$

So far, this equation tells us that to compute the dependence of the SSE on a given weight, $v_{h,o}$ (a weight connecting hidden node h to output node o), we compute:

- The dependence of the SSE on the specific error at output node o,
- The dependence of the specific *error* at output node o on the *actual output* A at output node o, and
- The dependence of the *actual output* A at output node o on the weight connecting the hidden node h to that output node o.

Before we plug in various values for each of these multiplying terms, there is one more step that we can take. We know that the actual output A at a given node is the transfer function applied to the sum of the weighted inputs going into that node.

$$A_o = \mathscr{F}(\sum_{h=1}^{H} v_{h,o} * H_h) \tag{1.14}$$

where the sum is being taken over all of the H hidden nodes, each contributing an input to a given output node o. In this expression, H_h is the output of the h^{th} hidden node.

This equation simply states that the actual output A_o at output node o equals the transfer function $\mathscr{F}$ (which we discussed in Subsection 1.1.1, applied to the sum of the inputs to that output node. Further, each of the

inputs to that output node is made up of the *weight* $v_{h,o}$ connecting a specific hidden node h to the specific output node o, multipled by the output from that specific hidden node H_h.

For simplicity in writing the longer chain rule expression, let's identify the sum of the inputs into a given output node o as $NdInpt_o$.

Thus we can write

$$NdInpt_o = \sum_{o=1}^{O} v_{h,o} * H_h \tag{1.15}$$

and

$$A_o = \mathscr{F}(NdInpt_o) \tag{1.16}$$

Now we are ready to go back to Eqn. 1.13 and introduce a substitution taken from Eqn. 1.16

$$\frac{\partial SSE}{\partial v_{h,o}} = \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial A_o}\frac{\partial A_o}{\partial v_{h,o}} = \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial \mathscr{F}(NdInpt_o)}\frac{\partial \mathscr{F}(NdInpt_o)}{\partial v_{h,o}} \tag{1.17}$$

We introduce one more substitution; we break down the dependence of the *Error* at node o ($Error_o$) on the results of transfer function applied to the weighted node inputs into two terms:

- The dependence of the *Error* at node o ($Error_o$) on the result of the transfer function itself, and
- The dependence of the result of the transfer function on the actual sum of the weighted inputs into that node.

This gives us

$$\begin{aligned}\frac{\partial SSE}{\partial v_{h,o}} &= \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial \mathscr{F}(NdInpt_o)}\frac{\partial \mathscr{F}(NdInpt_o)}{\partial v_{h,o}} \\ &= \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial \mathscr{F}}\frac{\partial \mathscr{F}(NdInpt_o)}{\partial NdInpt_o}\frac{\partial NdInpt_o}{\partial v_{h,o}}\end{aligned} \tag{1.18}$$

This means that we can compute the dependence of the summed squared error (that which we are using as our overall error term) on a specific weight

connecting a specific hidden node to a specific output node as the multiplication of four distinct terms.

Now, all that we have to do is to compute each of these specific terms, substitute them in, and we have the backpropagation of error onto a given hidden-to-output connection weight.

Dependence of the Summed Squared Error on the Error Term

We begin with referencing Eqn. 1.11, and note that

$$SSE = \frac{1}{2}\sum_{q=1}^{O}(Desired_q - Actual_q)^2 = \frac{1}{2}\sum_{q=1}^{O} E_q^2. \tag{1.19}$$

Notice that we've changed the running index from o (as was used in Eqn. 1.11) to q. This is because we run the sum over all of the output errors, from all the output nodes, but want to find (in our next step) the dependence of the summed squared error on the error at a *specific* node.

We begin by substituting our expression for the summed squared error into the term for the dependence of the SSE on a specific error.

$$\frac{\partial SSE}{\partial E_o} = \frac{1}{2}\frac{\partial \sum_{q=1}^{O} E_q^2}{\partial E_o} \tag{1.20}$$

We notice right away that when we have a sum of different things, each dependent on a different node, that the only real dependence comes from the node that we are actually considering. That is, we get a dependence of the SSE on the error at node o *only* when we are dealing with the squared error from node o; none of the other squared error terms matter. That is, for the case of finding the dependence of the SSE on the error in the hidden-to-output connection weight dependence *only*, we can go from the sum to a single term.

We immediately simplify our equation by removing the sum, and just considering the node in question.

$$\frac{\partial SSE}{\partial E_o} = \frac{\partial SE}{\partial E_o} = \frac{1}{2}\frac{\partial E_{q=o}^2}{\partial E_o} \tag{1.21}$$

This is much easier now. We apply some basic calculus, recalling that

$$\frac{1}{2}\frac{dx^2}{dx} = 2 * \frac{1}{2}x = x \tag{1.22}$$

Thus we have

$$\frac{\partial SSE}{\partial E_o} = \frac{1}{2}\frac{\partial E^2_{q=o}}{\partial E_o} = E_o \tag{1.23}$$

Dependence of the Error on the Error Actual Output

We've just computed the first term from Eqn. 1.18, and are ready to compute the second, which is the dependence of the actual error on the activation at the output node. We want to find

$$\frac{\partial E_o}{\partial \mathscr{F}_o} \tag{1.24}$$

As we noticed earlier, the error term E_o is simply

$$E_o = Desired_o - Actual_o = Desired_o - \mathscr{F}_o, \tag{1.25}$$

where $\mathscr{F}_o$ is being used as shorthand for the entire activation at output node o, that is, $\mathscr{F}_o = \mathscr{F}(NdInpt_o)$.

We apply a rule of calculus, by which

$$\frac{d(a-x)}{dx} = -1 \tag{1.26}$$

to obtain

$$\frac{\partial E_o}{\partial \mathscr{F}_o} = -1. \tag{1.27}$$

Dependence of the Actual Output on the Weighted Summed Inputs

We previously computed the derivative of the transfer function in Subsection 1.1.2, Eqn.1.10, to have

$$\partial y/\partial x = \alpha y\big[1-y\big], \tag{1.28}$$

where y is identified as the transfer function $\mathscr{F}$, and x is the weighted summed input into a node.

We will use this to identify the value of the third term in Eqn. 1.18, so that

$$\frac{\partial \mathscr{F}(NdInpt_o)}{\partial NdInpt_o} = \alpha \mathscr{F}_o\big[1 - \mathscr{F}_o\big] \tag{1.29}$$

We have just one more term to compute in order to have the set of four terms that we need for backpropagating error from an output node to a weight connecting a hidden to that output node.

Dependence of the Weighted Summed Inputs on an Individual Connection Weight

Our final task, for this stage of backpropagation computation, is to determine

$$\frac{\partial NdInpt_o}{\partial v_{h,o}}. \tag{1.30}$$

We recall, from Eqn. 1.14, that $NdInpt_o$, the actual sum of weighted inputs to node o, is expressed as

$$NdInpt_o = \sum_{h=1}^{H} v_{h,o} * H_h. \tag{1.31}$$

Thus, we are seeking

$$\frac{\partial NdInpt_o}{\partial v_{h,o}} = \frac{\partial \sum_{q=1}^{H} v_{q,o} * H_h}{\partial v_{h,o}}. \tag{1.32}$$

Once again, we've changed the running index on the sum from h to q, to express that we are considering all possible contributions. Of these, however, only one term is dependent on $v_{h,o}$, so that we have

$$\frac{\partial NdInpt_o}{\partial v_{h,o}} = \frac{\partial v_{h,o} * H_h}{\partial v_{h,o}} = H_h. \tag{1.33}$$

We substitute all four terms into our previous Eqn. 1.18

$$\frac{\partial SSE}{\partial v_{h,o}} = \frac{\partial SSE}{\partial E_o}\frac{\partial E_o}{\partial \mathscr{F}}\frac{\partial \mathscr{F}}{\partial (NdInpt_o)}\frac{\partial (NdInpt_o)}{\partial v_{h,o}} \tag{1.34}$$

$$= E_o * (-1) * \alpha \mathscr{F}_o\big[1 - \mathscr{F}_o\big] * H_h \tag{1.35}$$

$$= -\alpha E_o \mathscr{F}_o\big[1 - \mathscr{F}_o\big] H_h \tag{1.36}$$

Before we move on, let's see if we can't interpret this equation just a bit. Let's keep in mind that we're discussing how the squared error at a single output node depends on one of the specific connection weights connecting a given hidden node to that output node.

First, the lead term is α. The higher the value for α, the greater the dependence of the error on the connection weight. We're going to adjust this connection weight by some small fraction of the overall dependence term; it's going to be like taking a linear approximation of a slope, and incrementing the connection weight by a small value based on this linear slope approximation. What this means is that the larger the alpha, the larger the (approximately linear slope) adjustment, which means a faster change to the connection weight. In short, increasing α means that we'll typically change the connection weight more than if we had a smaller alpha. The overall impact of this depends on the overall nature of the gradient curve for which we're attempting to find a minimum, and that is beyond this particular discussion. We'll simply summarize by saying, " bigger α means faster change."

The size of the error term has an impact also. A bigger error term will also induce a faster change. As we get closer to a workable set of connection weights, the error terms will decrease in size; the adjustments to the connection weights will become smaller.

To assess the value of the terms involving the transfer function, we consult Figure 1.2. Let us recall that the notation $\mathscr{F}_o$ means that the transfer function is applied to the set of summed inputs to the output node O_o. The more that this set of inputs is either a large positive or large negative number, the more that the total set of terms involving the transfer function is small. This means that for very large positive or negative values of the total summed inputs, we decrease the amount of change that we make to a specific connection weight.

When we consider this, it makes sense. The set of terms involving the transfer function reflect the input of all the hidden nodes into into that

particular output node. If the sum of those weighted inputs is large (at least in magnitude), it means that either there are very strongly activated hidden nodes sending inputs to that output node, or the weights connecting those hidden nodes to the output node are strong, or both. In either case, we're looking at the influence of hidden nodes other than the one that has the particular connection to the output that we're seeking to modify. This means that we should probably not make any great changes, as we'd distort the input of those other hidden nodes.

On the other hand, when the sum of the weighted activations going into that transfer function are more moderate - centered more-or-less around zero - then we can say that either the overall inputs from these other hidden nodes are not that strong, or that they are canceling each other out. In either of these cases, we have more leeway in adjusting the connection weight in question.

Finally, we note the impact of the last term, H_h. This is the actual activation of the hidden node h. The stronger that this activation is, in either a positive or negative sense, the more that it will impact the extent to which we change the connection weight.

This concludes the derivation of the dependence of the squared error on a specific hidden-to-output connection weight. The next step, in the next subsection, is to consider the dependence of the squared error on a given input-to-hidden connection weight.

1.1.5 Backpropagation of Error: Part 2

As a next step, we want to compute the dependence of the summed squared error on the input-to-hidden weights.

The key difference between this calculation and the previous one is that earlier, when we were interested in the dependence of the summed squared error (SSE) on a specific hidden-to-output connection weight $v_{h,o}$, we only needed to think about the squared error at output node O_o. The other output nodes did not impact the dependence of the SSE on this output node.

In contrast, this time, when we want to consider the dependence of the SSE on a given input-to-hidden connection weight $w_{i,h}$, we now have to think about the influence of the squared errors from *each* of the output nodes. This is because the input-to-hidden connection weight $w_{i,h}$ influences the activation (resulting output) of hidden node H_h. The activation of this hidden node, however, impacts all of the output nodes. Therefore, we need to con-

sider the SSE at all the output nodes, and their back-propagated influence on H_h, and from that hidden node to $w_{i,h}$. This is shown in Figure 1.3.

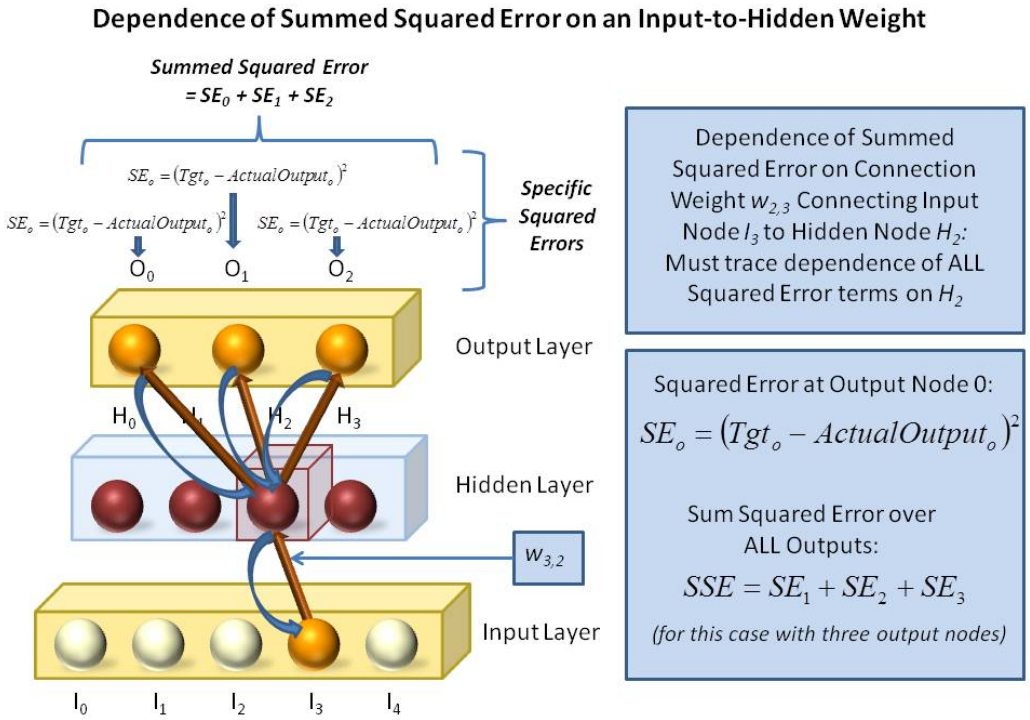

Figure 1.3: Dependence of the Summed Squared Error (SSE) on the input-to-hidden connection weight $w_{i,h}$ requires tracing the dependence of the squared error at each output node through hidden node h and from there to the connection weight $w_{i,h}$. This is shown for the case where the connection weight is between input node 3 and hidden node 2, $w_{3,2}$, where the nodes in each layer are numbered in the manner of Python code, beginning with the zeroth node in each layer.

We thus desire

$$\frac{\partial SSE}{\partial w_{i,h}} = \sum_{o=1}^{O} \left[\frac{\partial SE}{\partial E_o} \frac{\partial E_o}{\partial w_{i,h}} \right] = \sum_{o=1}^{O} \left[\frac{\partial SE}{\partial E_o} \frac{\partial E_o}{\partial A_o} \frac{\partial A_o}{\partial w_{i,h}} \right] \tag{1.37}$$

Thus far, we are identical with the previous initial step in backpropagation, given in Eqn. 1.13, with the exception that the last term involves the

dependence of the output activation on a specific input-to-hidden weight, instead of hidden-to-output weight. Also, we are summing over the full set of squared errors at each of the output nodes.

Before we go further, we're going to write the dependence of the activation (actual output) at output node O_o on the input-to-hidden connection weight connecting input node I_i to hidden node H_h. That is, we will write the expression for

$$\frac{\partial A_o}{\partial w_{i,h}} \tag{1.38}$$

We know already that the activation (actual output) of the output node o is a result of the transfer function being applied to the summed inputs to that node.

$$A_o = \mathscr{F}(NdInpt_o) \tag{1.39}$$

Now, what we need is to work through the dependence of A_o on the connection weight $w_{i,h}$ using this relation.

$$\frac{\partial A_o}{\partial w_{i,h}} = \frac{\partial \mathscr{F}(NdInpt_o)}{\partial w_{i,h}} = \frac{\partial \mathscr{F}(NdInpt_o)}{\partial (NdInpt_o)} \frac{\partial (NdInpt_o)}{\partial w_{i,h}} \tag{1.40}$$

We have previously found the dependence of the transfer function $\mathscr{F}$ on the node input; we already know that

$$\frac{\partial \mathscr{F}(NdInpt_o)}{\partial NdInpt_o} = \alpha \mathscr{F}_o\big[1 - \mathscr{F}_o\big] \tag{1.41}$$

We can substitute this into Eqn. 1.40 to obtain

$$\frac{\partial A_o}{\partial w_{i,h}} = \frac{\partial \mathscr{F}(NdInpt_o)}{\partial w_{i,h}} = \alpha \mathscr{F}_o\big[1 - \mathscr{F}_o\big] \frac{\partial (NdInpt_o)}{\partial w_{i,h}} \tag{1.42}$$

Now, we need to find

$$\frac{\partial (NdInpt_o)}{\partial w_{i,h}} \tag{1.43}$$

To do this, we recall from Eq. 1.31 that

$$NdInpt_o = \sum_{h=1}^{H} v_{h,o} * H_h. \tag{1.44}$$

We substitute this into Eqn. 1.43, while changing the index on the summation from h to q, to obtain

$$\frac{\partial(NdInpt_o)}{\partial w_{i,h}} = \frac{\partial\Big(\sum_{q=1}^{H} v_{q,o} * H_q\Big)}{\partial w_{i,h}} \tag{1.45}$$

Since the dependence holds only in the case where $q = h$, we can simplify this as

$$\frac{\partial(NdInpt_o)}{\partial w_{i,h}} = v_{h,o}\frac{\partial H_h}{\partial w_{i,h}} \tag{1.46}$$

We can write an expression for the activation of the hidden node H_h as

$$H_h = \mathscr{F}(\sum_{i=1}^{I} w_{i,h} * Input_i) = \mathscr{F}(NdInput_h) \tag{1.47}$$

We can write the node inputs to hidden node h as

$$NdInpt_h = \sum_{i=1}^{I} w_{i,h} * Input_i. \tag{1.48}$$

We can substitute from Eqn. 1.48 into Eqn. 1.47 to obtain

$$H_h = \mathscr{F}(NdInput_h) = \mathscr{F}\Big(\sum_{i=1}^{I} w_{i,h} * Input_i\Big) \tag{1.49}$$

We can substitute this expression for H_h into Eqn. 1.44 to obtain

$$NdInpt_o = \sum_{h=1}^{H} v_{h,o} * H_h = \sum_{h=1}^{H} v_{h,o} * \mathscr{F}\Big(\sum_{i=1}^{I} w_{i,h} * Input_i\Big). \tag{1.50}$$

Now, let's use this expression to figure out the dependence of the node inputs at output node o on the input-to-hidden connection weight $w_{i,h}$. Specifically, going back to Eqn. 1.43, we introduce a substitution based on Eqn. 1.50 and write

$$\frac{\partial(NdInpt_o)}{\partial w_{i,h}} = \frac{\partial\Big[\sum_{h=1}^{H} v_{h,o} * \mathscr{F}\Big(\sum_{i=1}^{I} w_{i,h} * Input_i\Big)\Big]}{\partial w_{i,h}} \tag{1.51}$$

The hidden-to-output connection weights $v_{h,o}$ are a constant with regard to the specific input-to-hidden connection weight $w_{i,h}$, and so we can take these terms (along with their summation) outside of the partial derivative. We also want to distinguish between the sum over all possible hidden nodes and the specific one with which we want to compute a dependence, and so we change the running index on the sum over the hidden nodes to be q. Thus we write

$$\frac{\partial(NdInpt_o)}{\partial w_{i,h}} = \sum_{q=1}^{H} v_{q,o} \frac{\partial\Big[\mathscr{F}\Big(\sum\limits_{i=1}^{I} w_{i,q} * Input_i\Big)\Big]}{\partial w_{i,h}} \tag{1.52}$$

We know that the transfer function is being taken, in each case, at hidden node q, and we know that the derivative of this is given as

$$\begin{aligned}
\frac{\partial\Big[\mathscr{F}\Big(\sum\limits_{i=1}^{I} w_{i,q} * Input_i\Big)\Big]}{\partial w_{i,h}} &= \frac{\partial\mathscr{F}(NdInput_q)}{\partial NdInput_q}\frac{\partial NdInput_q}{\partial w_{i,h}} \\
&= \alpha\mathscr{F}_q(1-\mathscr{F}_q) * \frac{\partial NdInput_q}{\partial w_{i,h}} \\
&= \alpha\mathscr{F}_q(1-\mathscr{F}_q) * \frac{\partial\Big(\sum\limits_{i=1}^{I} w_{i,q} * Input_i\Big)}{\partial w_{i,h}} \\
&= \alpha\mathscr{F}_q(1-\mathscr{F}_q)Input_i
\end{aligned} \tag{1.53}$$

We note that this dependence occurs only when $w_{i,q} = w_{i,h}$ (all other terms drop out in the last sum), so that $q = h$, and we can rewrite the previous Eqn. 1.53 as

$$\frac{\partial\Big[\mathscr{F}\Big(\sum\limits_{i=1}^{I} w_{i,q} * Input_i\Big)\Big]}{\partial w_{i,h}} = \alpha\mathscr{F}_h(1-\mathscr{F}_h)Input_i \tag{1.54}$$

We return now to the initial equation for backpropagating the dependence of the summed squared error SSE on a given input-to-hidden connection weight, Eqn. 1.37, and substitute what we have gained in this last step to obtain

$$\begin{aligned}\frac{\partial SSE}{\partial w_{i,h}} &= \sum_{o=1}^{O}\Big[\frac{\partial SE}{\partial E_o}\frac{\partial E_o}{\partial w_{i,h}}\Big] = \sum_{o=1}^{O}\Big[\frac{\partial SE}{\partial E_o}\frac{\partial E_o}{\partial A_o}\frac{\partial A_o}{\partial w_{i,h}}\Big] \\ &= \sum_{o=1}^{O}\Big[\frac{\partial SE}{\partial E_o}\frac{\partial E_o}{\partial A_o}\alpha v_{h,o}\mathscr{F}_h(1-\mathscr{F}_h)Input_i\Big] \\ &= \alpha\sum_{o=1}^{O}\Big[\frac{\partial SE}{\partial E_o}\frac{\partial E_o}{\partial A_o}v_{h,o}\mathscr{F}_h(1-\mathscr{F}_h)Input_i\Big]\end{aligned} \tag{1.55}$$

We can quickly obtain the dependence of the squared error on the error (first term inside the summation) and the dependence of the error on the activation (second term inside the summation), using results previously obtained, referencing Eqn. 1.23 which had previously given us

$$\frac{\partial SSE}{\partial E_o} = \frac{1}{2}\frac{\partial E_{q=o}^2}{\partial E_o} = E_o \tag{1.56}$$

and also Eqn. 1.27

$$\frac{\partial E_o}{\partial \mathscr{F}_o} = -1. \tag{1.57}$$

so that we can write

$$\begin{aligned}\frac{\partial SSE}{\partial w_{i,h}} &= \alpha\sum_{o=1}^{O}\Big[\frac{\partial SE}{\partial E_o}\frac{\partial E_o}{\partial A_o}v_{h,o}\mathscr{F}_h(1-\mathscr{F}_h)Input_i\Big] \\ &= \alpha\sum_{o=1}^{O}\Big[E_o\frac{\partial E_o}{\partial A_o}v_{h,o}\mathscr{F}_h(1-\mathscr{F}_h)Input_i\Big] \\ &= \alpha\sum_{o=1}^{O}\Big[E_o(-1)v_{h,o}\mathscr{F}_h(1-\mathscr{F}_h)Input_i\Big] \\ &= -\alpha\sum_{o=1}^{O}\Big[E_o v_{h,o}\mathscr{F}_h(1-\mathscr{F}_h)Input_i\Big]\end{aligned} \tag{1.58}$$

However, while the summation is over all output nodes, the terms involving the specific hidden node involved in the dependence of the SSE on the

input-to-hidden connection weight $w_{i,h}$, together with the actual value of the input node i, that is, $Input_i$, are constants relative to the output error terms. We can thus pull them out of the summation and write

$$\begin{aligned}\frac{\partial SSE}{\partial w_{i,h}} &= -\alpha \sum_{o=1}^{O} \Big[E_o v_{h,o} \mathscr{F}_h (1 - \mathscr{F}_h) Input_i \Big] \\ &= -\alpha \mathscr{F}_h (1 - \mathscr{F}_h) Input_i \sum_{o=1}^{O} v_{h,o} E_o \end{aligned} \tag{1.59}$$

1.1.6 Summary: The Two Backpropagation Equations

We have now derived the two equations needed for backpropagation of error from:

1. The dependence of the *Error* (which is the summed squared error across the output nodes) at node o ($Error_o$) on a given hidden-to-output node $v_{h,o}$, and

2. The dependence of the *Error* at node o ($Error_o$) on a given input-to-hidden node $w_{i,h}$.

In rewriting these equations, we now identify them with the changes (deltas, or δ values) to the weights (hidden-to-output, and input-to-hidden, respectively). Because we're now actually considering the actual changes that we'll make to the existing weights, we'll introduce a learning parameter η (Greek letter "eta"), which is a constant selected so that $0 < \eta <= 1$. Also, we are going to decrement the pre-existing weight by this change, so we write the delta in the weight ($\delta w_{i,h}$, or $\delta v_{h,o}$) as the negative of what we had in our previous equations. Because we're taking the negative of what was previously a negative term (in each case), the minus sign disappears.

These two *delta* equations, based on our previously identified dependencies of error on specific weights (see Eqns.), are:

$$\delta v_{h,o} = -\eta \frac{\partial SSE}{\partial v_{h,o}} = \eta \alpha E_o \mathscr{F}_o \big[1 - \mathscr{F}_o\big] H_h \tag{1.60}$$

and

$$\delta w_{i,h} = -\eta \frac{\partial SSE}{\partial w_{i,h}} = \eta \alpha \mathscr{F}_h (1 - \mathscr{F}_h) Input_i \sum_{o=1}^{O} v_{h,o} E_o \qquad (1.61)$$