Northwestern

# MSDS 458
Artificial Intelligence and Deep Learning

## Week 4: The Backpropagation Learning Rule – Continued -
*Input-to-Hidden Node Connection Weights*

**Original Deck: A.J. Maren**
**Revision History:**
- Original Slidedeck: *Oct., 2018*

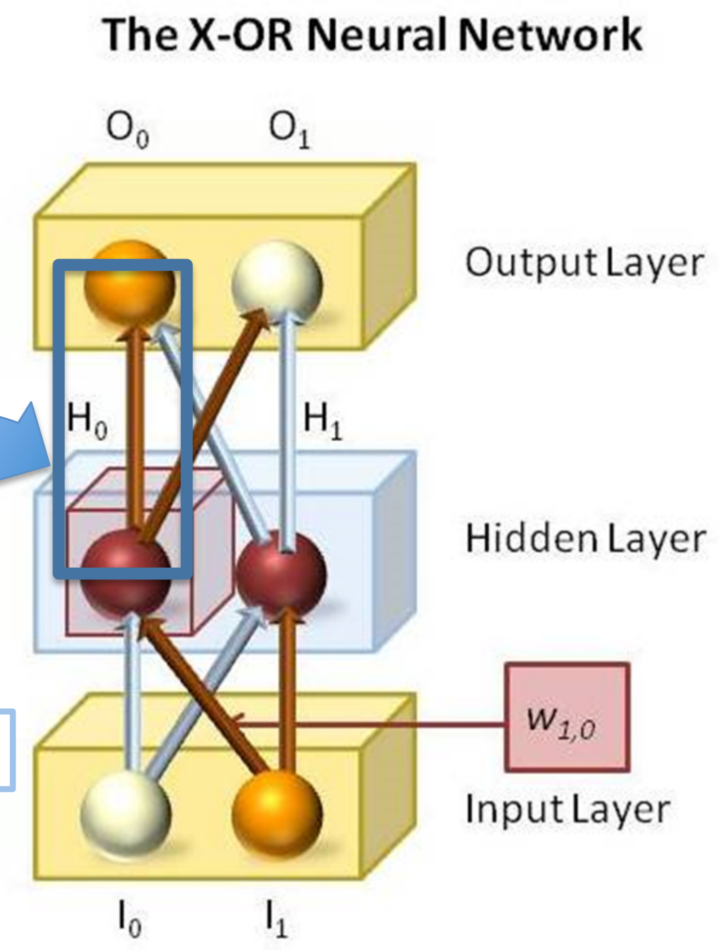
## Quick Review of Week 2: The X-OR Network

**Simple (Simplest-possible) NN:**

- Classify (0,0) and (1,1) inputs as (0,1) outputs
- Classify (1,0) and (0,1) inputs as (1,0) outputs
- Actually a challenging problem!

**We will look first at the dependence of the weight *v(0,1)* on the Summed-Squared Error (SSE).**

## *Simple X-OR Neural Network*



This is connection weight *v(0,1)*

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

## The Backpropagation Learning Rule (Review): Hidden-to-Output Nodes

**Dependence of the Hidden-to-Output connection weights (the *v(i,j)*) was straightforward**

- E.g., weight *v(0,1)* connects hidden node $H_0$ t output node $O_0$.
- The Summed-Squared-Error (SSE):
  - Compute error at $O_0$ and $O_1$.
  - Square each of the error terms.
  - Add them together for SSE.

- When we take the derivative of SSE with respect to *v(0,1)*, ***only one of the SSE terms is important – the one involving $O_0$.***
- The squared error at $O_0$ is a direct result of the impact of connection weight *v(0,1).*
- The squared error at $O_1$ has nothing to do with the connection weight *v(0,1).*
- So we don't need the SUMMED squared error, we just need the squared error at the node to which the connection weight is pointing.
- This makes the calculations a lot easier.

## *Dependence of connection weight v (0,1) on the SSE*

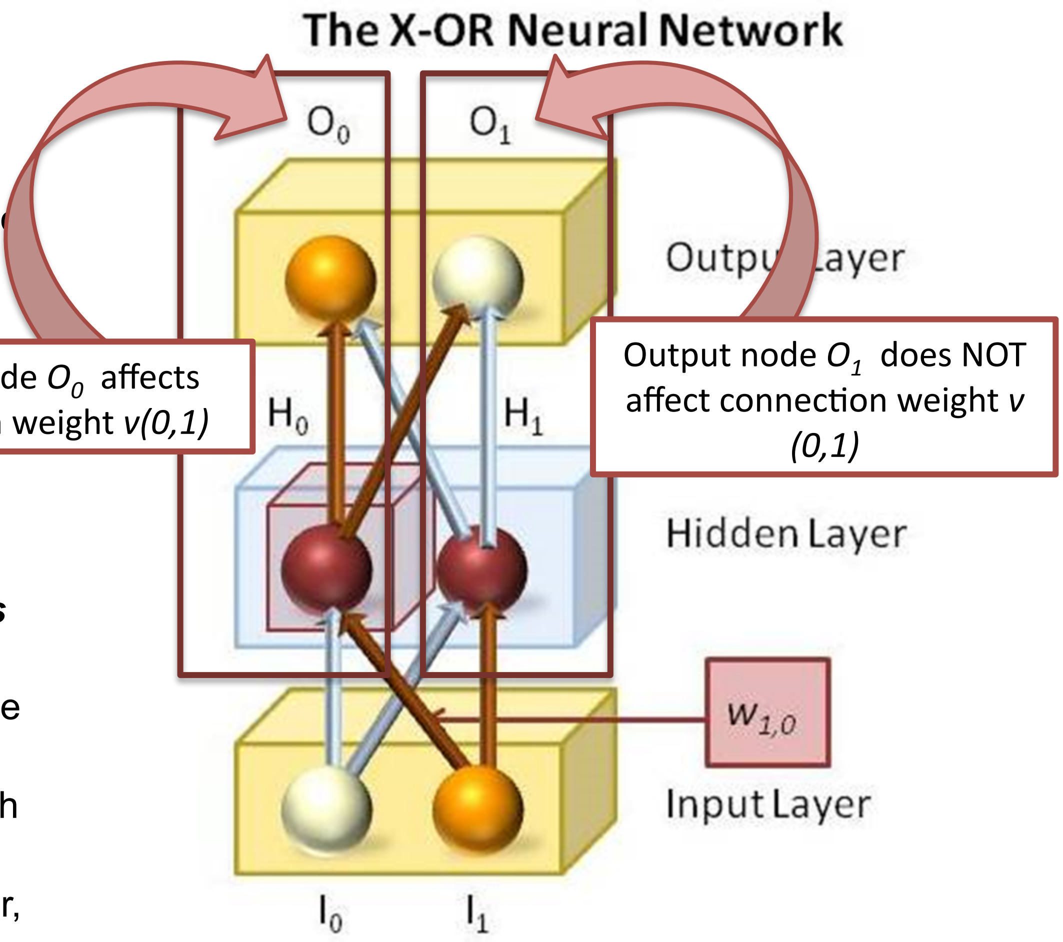

The X-OR Neural Network
$O_0$
$O_1$
Output Layer
Output node $O_0$ affects connection weight *v(0,1)*
Output node $O_1$ does NOT affect connection weight *v (0,1)*
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

## The Backpropagation Learning Rule (Review): Input-to-Hidden Nodes

**Dependence of the Input-to-Hidden connection weights (the *w(i,j)*) is more complex**

- E.g., weight *w(1,0)* connects input node $I_1$ to hidden node $H_0$.
- The Summed-Squared-Error (SSE):
  - Compute error at $O_0$ and $O_1$.
  - Square each of the error terms.
  - Add them together for SSE.
- The impact on connection weight *w(1,0)* comes from both (all) of the output nodes.
- This is because they both (all) affect activation of hidden node $H_0$.
- And the activation at hidden node $H_0$ affects each of the weights connecting to it.
- In this case, *w(1,0).*

## *Dependence of connection weight w (0,1) on the SSE*

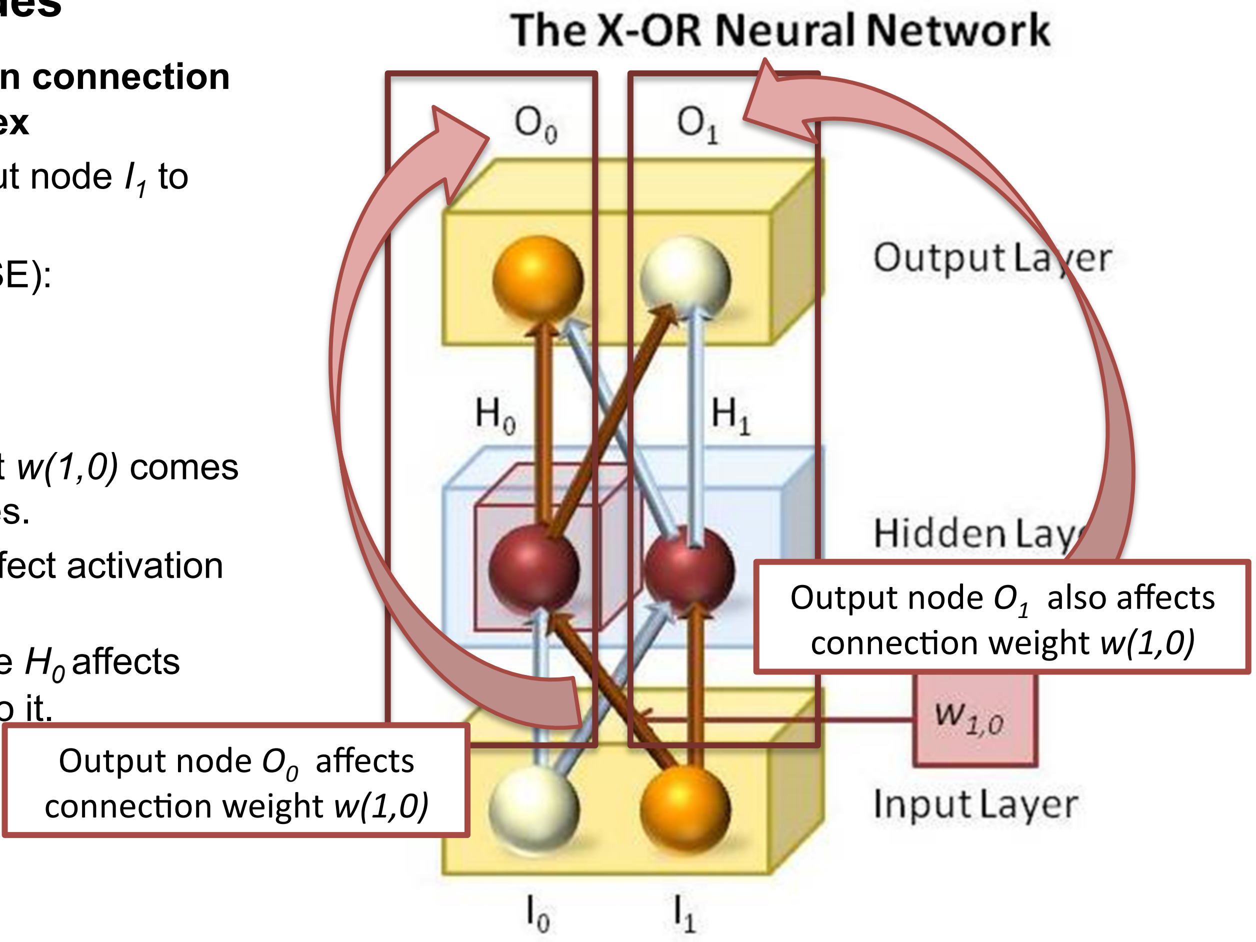

The X-OR Neural Network
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
Output node $O_1$ also affects connection weight *w(1,0)*
$w_{1,0}$
Input Layer
Output node $O_0$ affects connection weight *w(1,0)*
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

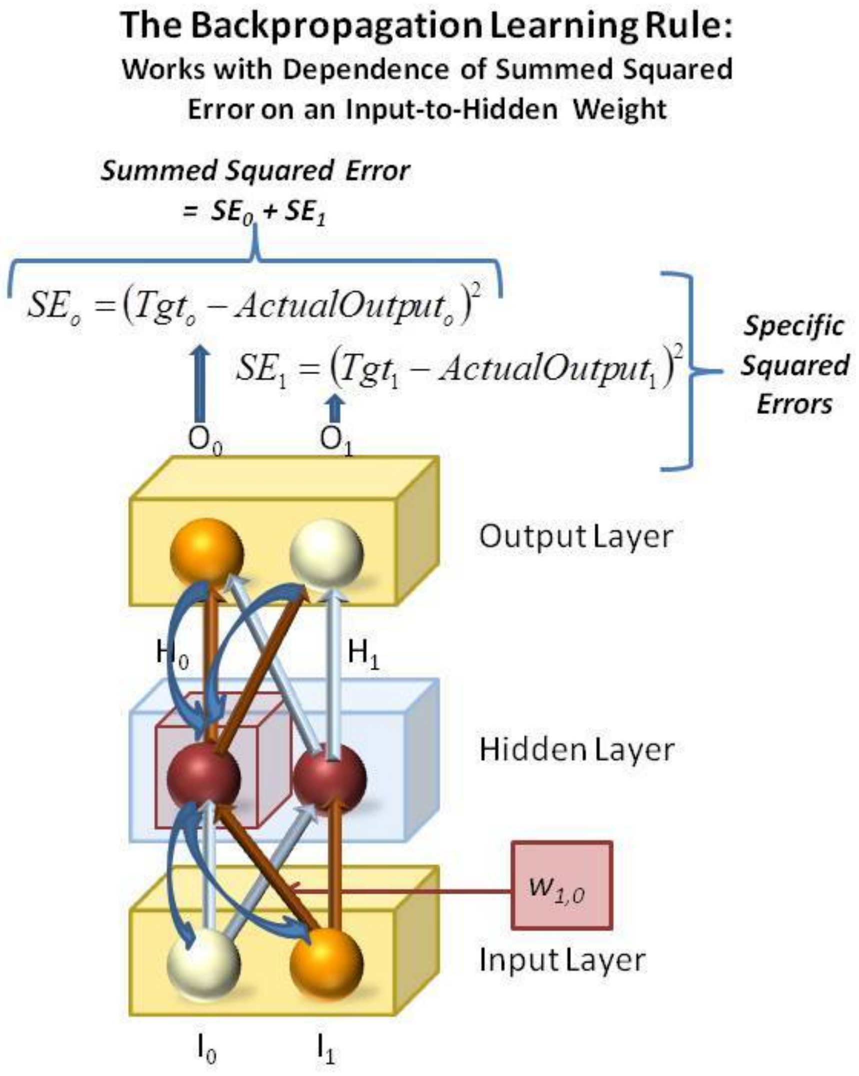

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 1:**

- Take the derivative of the SSE with respect to each of the terms in the SSE; each of the ***squared errors***.
- Take the derivative of (each of the) squared errors with respect to the ***error*** itself.
- Take the derivative of the error with respect to that which contributes to the error - which is the activation of the output function.
- Note: *Error* = *Tgt* – *Actual_Output*; the *Tgt* is a constant and doesn't count in the derivative. The *Actual_Output* is a function, and does count in the derivataive.

***We're doing this for each of the output nodes, so we'll have a summation term.***

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

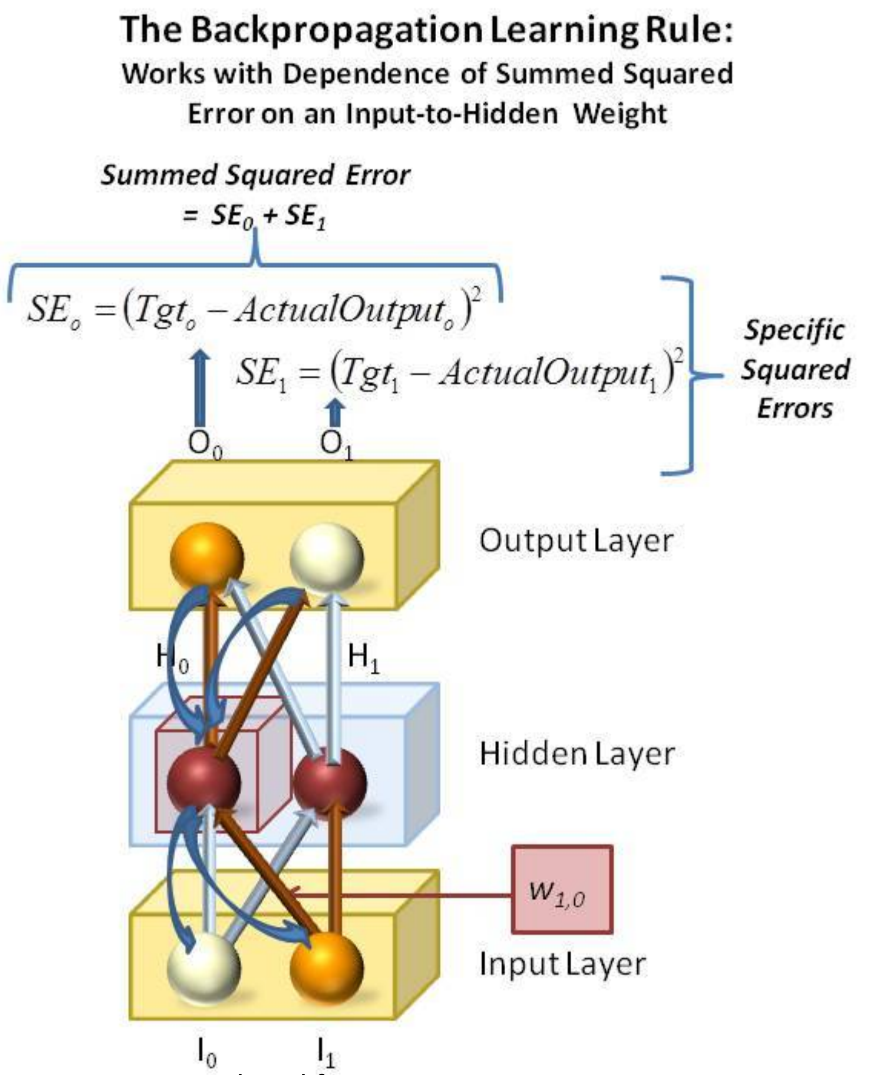

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error
= $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 2:**

***For <u>each</u> of the output nodes:***

- Take the derivative of the activation of the output node (*ActualOutput*) with respect to the transfer function in the output node. (This is what operates on the inputs into the output node.)

***Think about the activation in <u>just one</u> of those output nodes, e.g., $O_0$.***

***This activation function is the result of a transfer function applied to the sum of inputs from all the hidden nodes. (This means: there's <u>another</u> summation sign involved.)***

So, we'll have two things to think about:

- The derivative of the transfer function itself (with respect to the whole thing going into the transfer function), and
- The derivative of the whole thing going into the transfer function with respect to each particular thing that contributes to the whole thing.

Northwestern

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

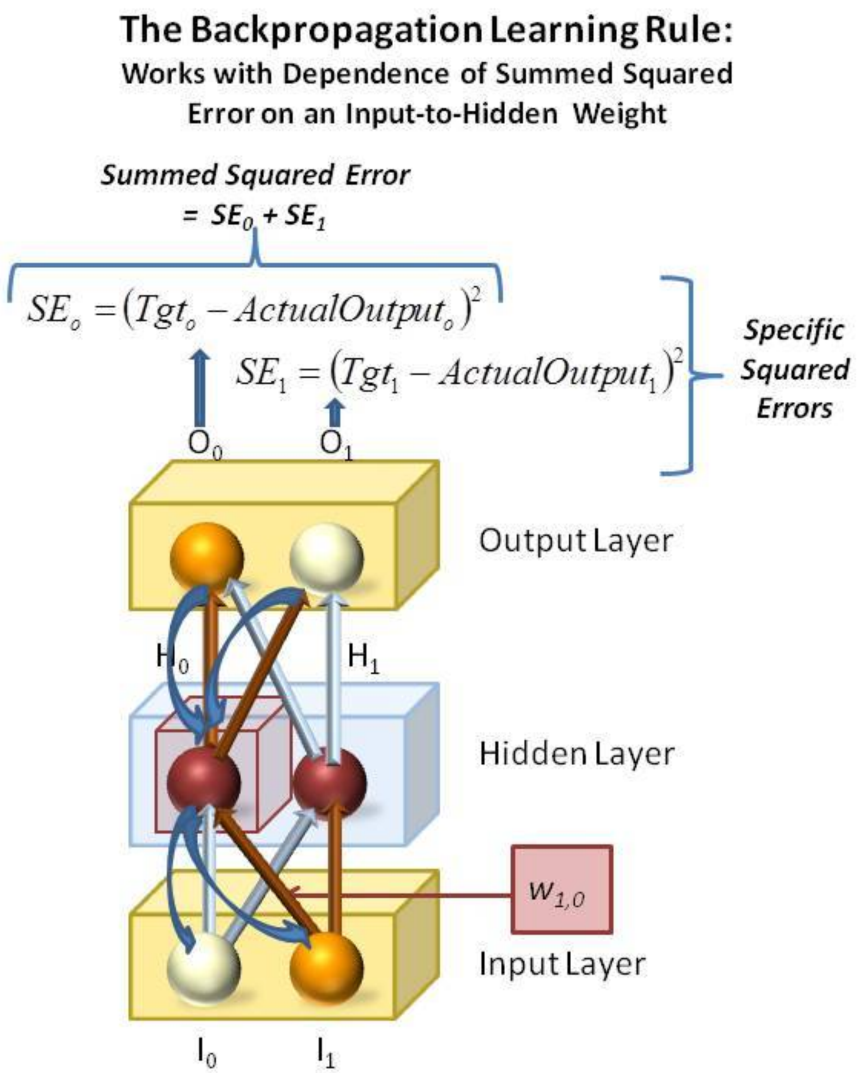

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 3:**

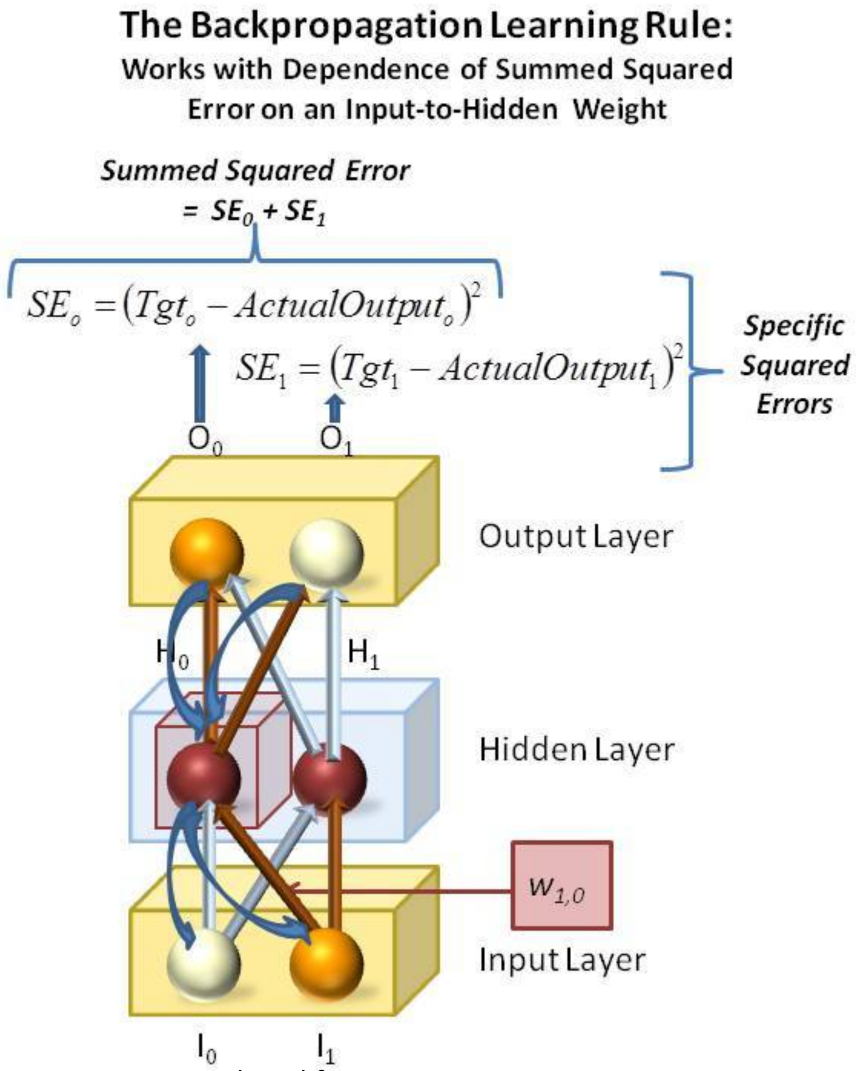
***From the last step of Part 2:*** We computed the derivative of the whole thing going into the transfer function (for each output node) with respect to each particular thing that contributes to the whole thing.

Each of these "particular things" going into the output node will be the activation in a specific hidden node times its connection weight to that particular output node.

This is *v(j,k)*Activ_H(j)*

- We're ***generalizing our notation***; *v(j,k)* is the connection weight from the jth hidden node (*H(j)* or $H_j$) to the kth output node (*O(k)*, or $O_k$ ).
- When we compute the derivative of the sum of things going into the output node, we get a ***sum of derivatives***, each with regard to one of those particular things going into the output node.
- When we compute the derivative for one of those particular things, it will be with respect to the thing that CAN change – the hidden node activation. Because the thing that CAN'T change for this calculation is the hidden-to-output connection weight. (We did those in the separate, preceding calculation.)

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE



The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 3 (Cont.):**

***So we going to take the derivative of v(j,k) *Activ_H(j).***

***We'll do this for each hidden node $H_j$*** *going into the output node $O_k$* , and we do this (see two slides back) for each of the different *$O_k$* .

***The question is: what are we taking the derivative with respect to?***

*For our purposes – getting the dependence of the SSE on a particular input-to-hidden connection weight, the* ***hidden-to-output connection weight is a constant.***

***We fussed with that in the preceding calculation*** *(for hidden-to-output connection weights). That's done now. (Temporarily.) So … we treat v(j,k) as a constant.*

*So we're going to take the next derivative as the* ***derivative of a constant times the activation of a hidden node****; the derivative of c*Activ_H(j) with respect to Activ_H(j).*

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

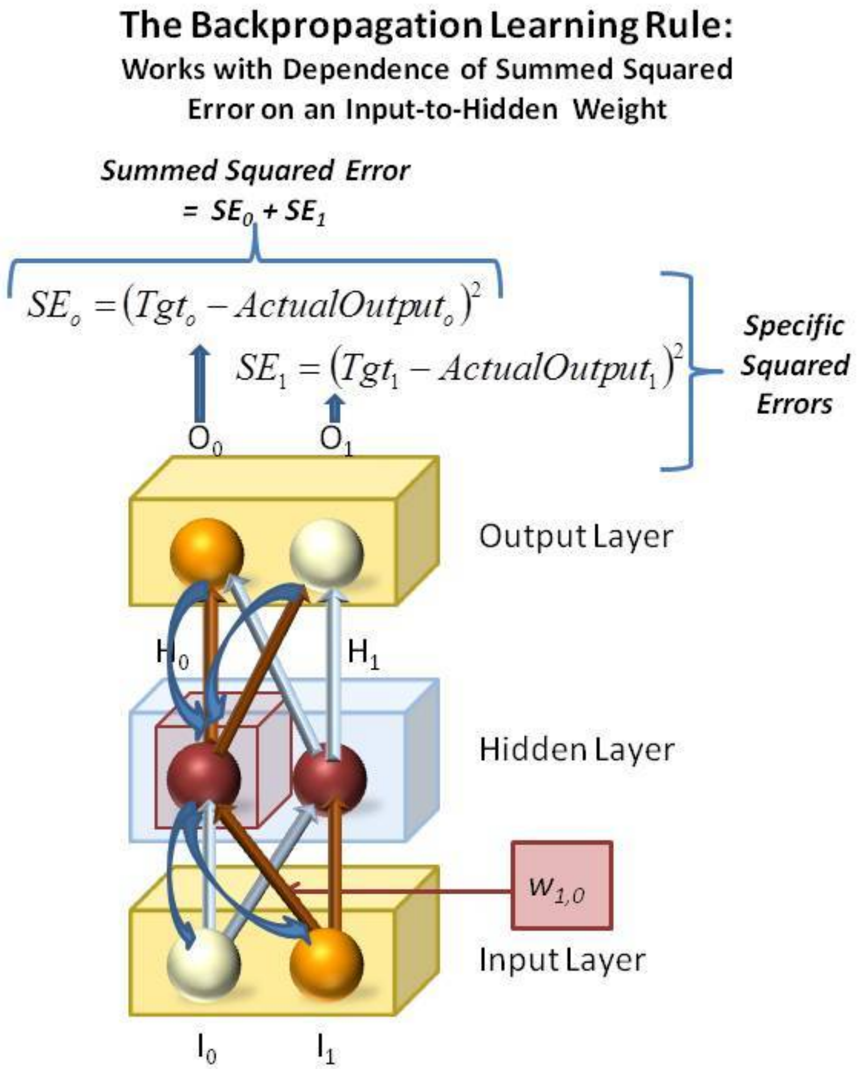

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 3 (Cont. some more):**

***So now, we going to take the derivative of something that looks like c*Activ_H(j).***

*(We know this. We're just starting to get a bit blurry by now.)*

***The derivative of c*x with respect to x is just c.***

***However … it's not just the derivative of c*x.*** *It's the derivative of c*x times whatever the derivative of x is with respect to whatever is inside of x. (Whatever makes up x.)*

*So … we get a multiplying factor of c, which in our case is really v(j,k).*

*Now we need to take the derivative of "x" – which in our case is Activ_H(j),* ***with whatever contributes to making up Activ_H(j).***

***We move forward … to Part 4 …*** *knowing that we want to multiply whatever we do next by the "constant" that we got from our most recent derivative; v(j,k).*

Northwestern

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

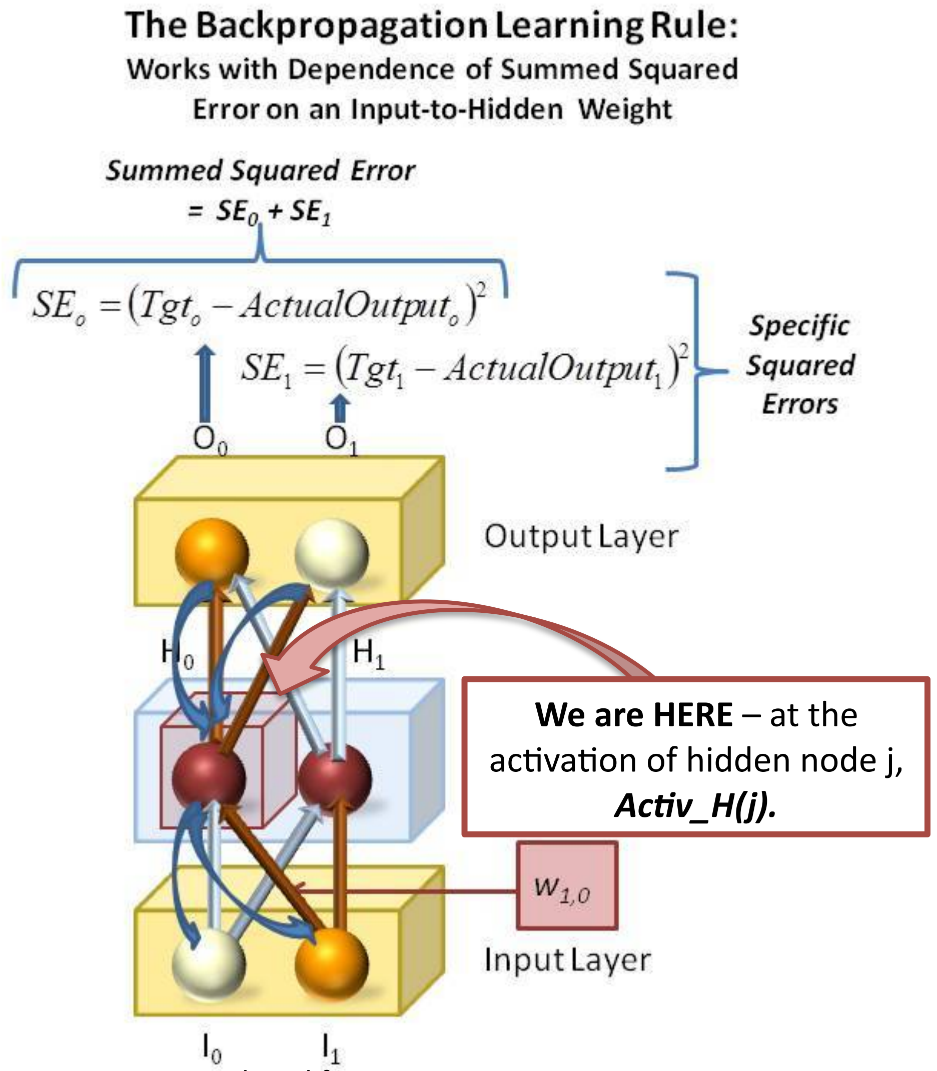

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
We are HERE – at the activation of hidden node j, Activ_H(j).
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 4:**

***So now, we need the derivative of Activ_H(j) with whatever makes up Activ_H(j).***

*(We just did something like this when we took the derivative of the activation at an output node.)*

*Once again, we get:*

- *The derivative of the activation at H(j) with respect to the transfer function, and then*
- *The derivative of what is inside the transfer function with respect to whatever goes into THAT.*
- *AND … that which goes into THIS PARTICULAR transfer function is (of course) the sum of connection weights times the values of the input nodes.*

*We don't talk about "activations" of the hidden nodes; they just have values – provided via the input data set.*

***And – finally getting to the end --- that which goes into the transfer function at the hidden node is:***

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

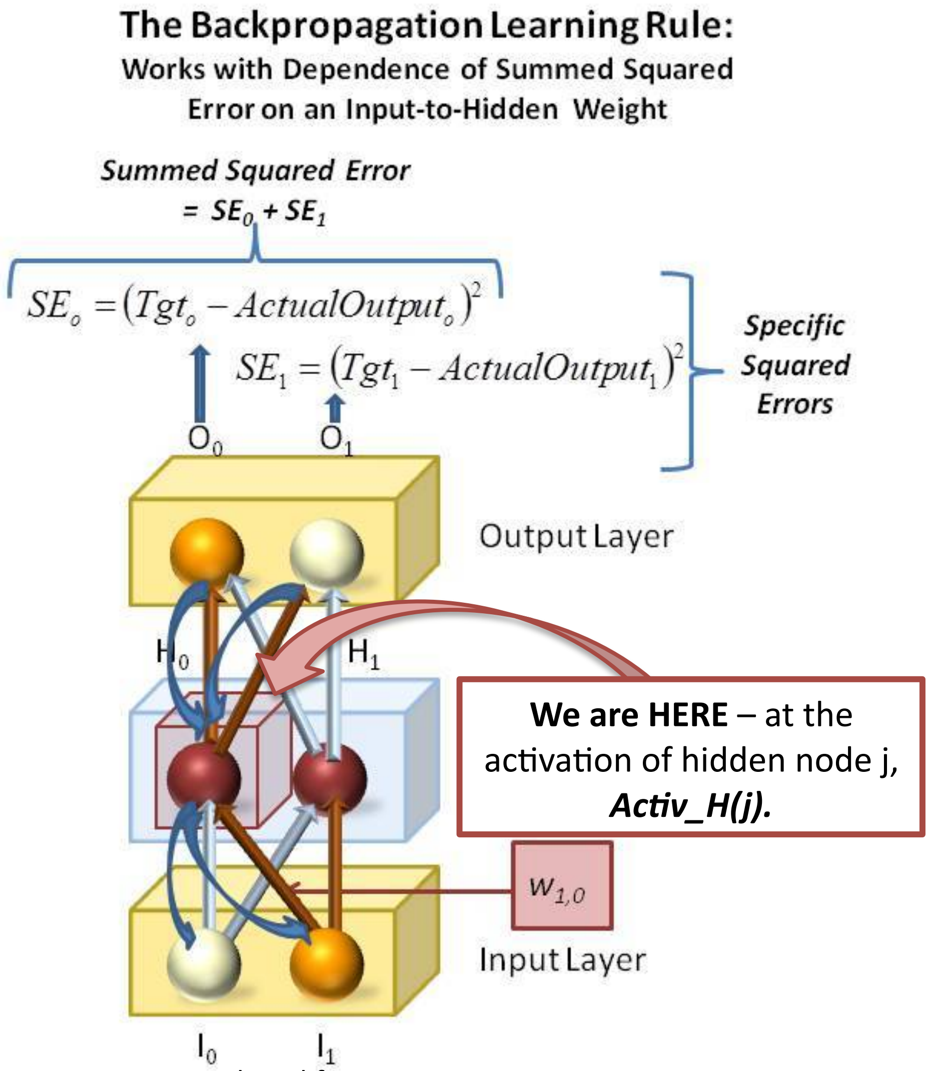

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
We are HERE – at the activation of hidden node j, Activ_H(j).
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 4 (cont):**

***And*** *– finally getting to the end --- that which goes into the transfer function at the hidden node is:* ***w(i,j) *Input(i)***

***or …***

$$\boldsymbol{w(i,j)^*\, I_i}$$

***The big difference:***

*Previously, the thing that was the constant was the connection weight, v(j,k) – because we'd used it as our variable in the whole previous calculation of getting the dependence of the SSE on the hidden-to-output connection weight, v(j,k).*

*For purposes of finding the dependence of the SSE on the input-to-hidden connection weight, w(i,j), v(j,k) is a constant.*

*Previously, the variable for our derivative calculations was the hidden node activation, H(j) or $H_j$.*

Northwestern

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

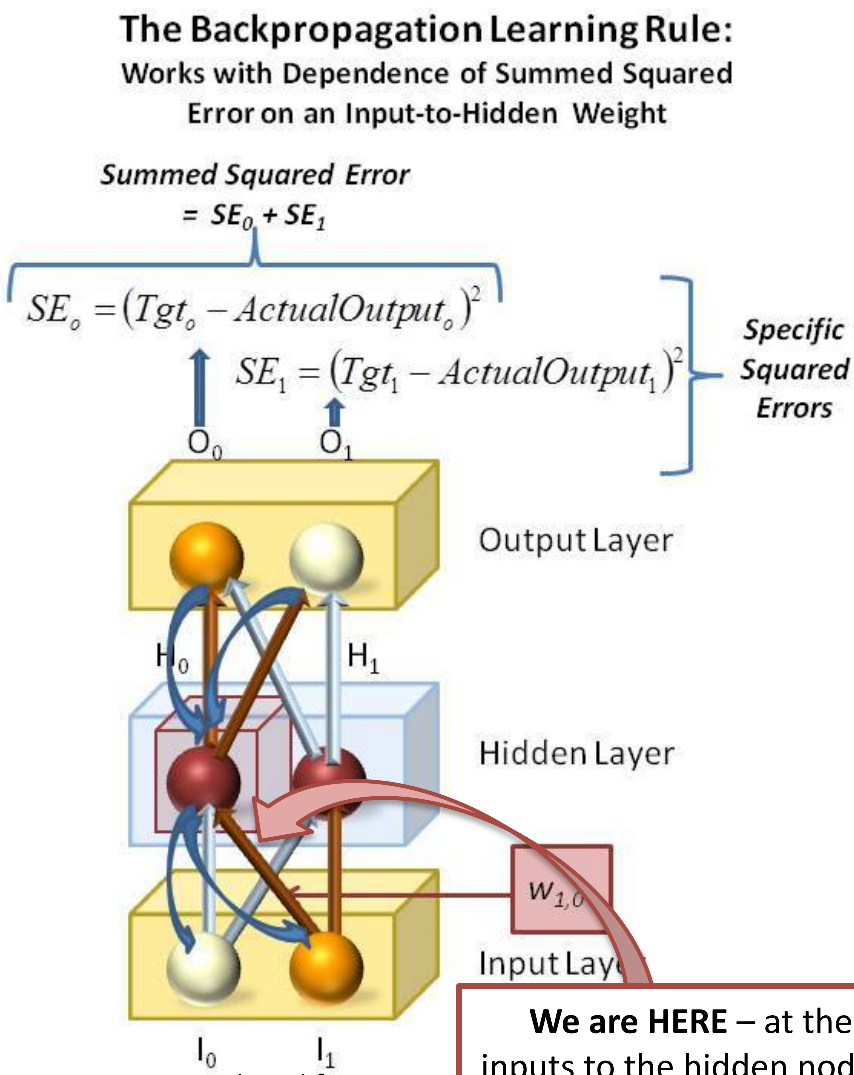

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$

Figure reproduced from:
*Statistical Mechanics, Neural*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**We are HERE** – at the inputs to the hidden node j, ***w(i,j)* $I_i$.***

**Part 4 (cont. some more):**

***We want the derivative of***

$$\boldsymbol{w(i,j)^* I_i}$$

***with respect to w(i,j).***

***For our purposes, $I_i$ is a constant.*** *(Every time we run the NN, we put in a value into each $I_i$ node. This is a constant value for each training run.)*

*The variable – that which we're taking the derivative with respect to – is the connection weight,* ***w(i,j).***

*So once again, our derivative is of the form of taking the derivative of a constant times a variable; c*x.*

- *This time, c is the input value;* ***$I_i$.***
- *The variable x is* ***w(i,j).***

*The result of taking the derivative is that we just get the constant, c, which in our case is* ***$I_i$.***.

## Next: Dependence of the Input-to-Hidden Connection Weights on SSE

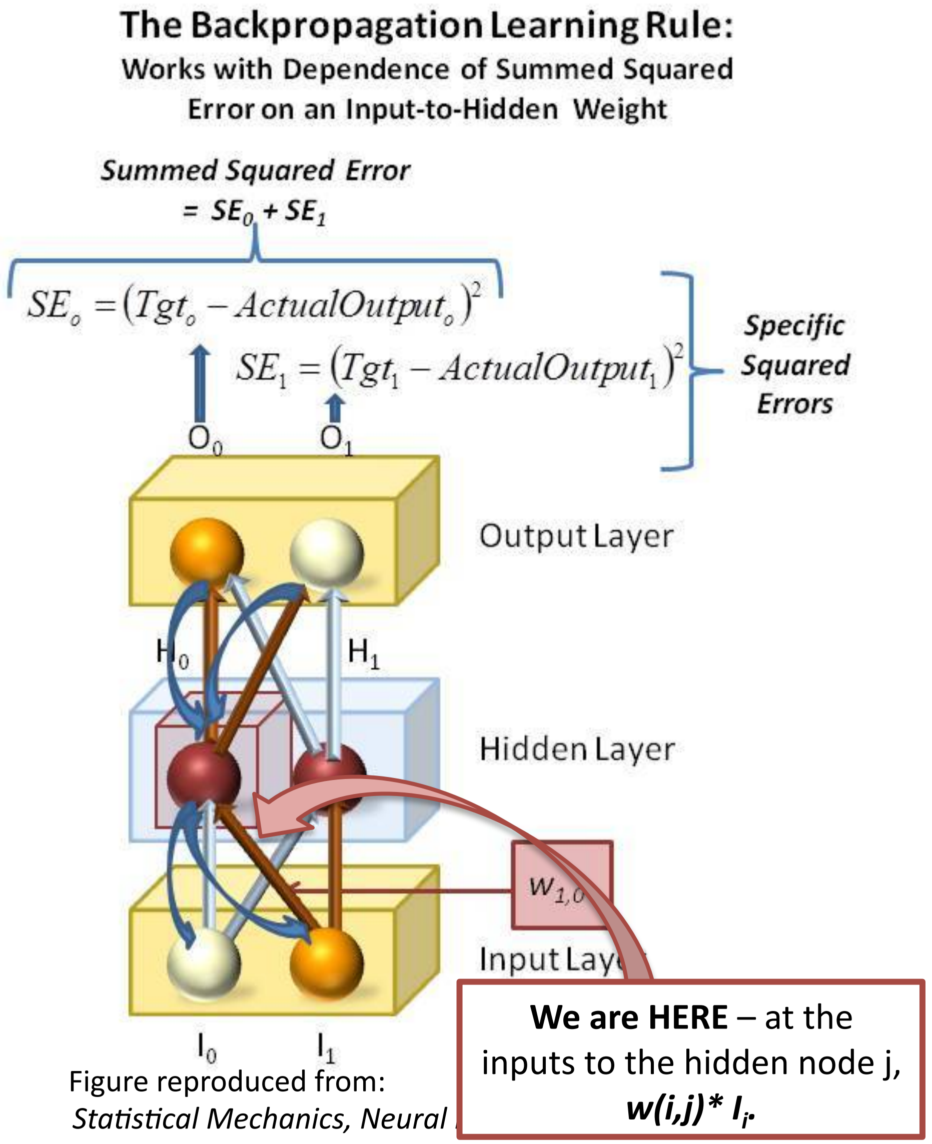

The Backpropagation Learning Rule:
Works with Dependence of Summed Squared Error on an Input-to-Hidden Weight
Summed Squared Error = $SE_0 + SE_1$
$SE_o = (Tgt_o - ActualOutput_o)^2$
$SE_1 = (Tgt_1 - ActualOutput_1)^2$
Specific Squared Errors
$O_0$
$O_1$
Output Layer
$H_0$
$H_1$
Hidden Layer
$w_{1,0}$
Input Layer
$I_0$
$I_1$
We are HERE – at the inputs to the hidden node j, $w(i,j) * I_i$.

Figure reproduced from:
*Statistical Mechanics, Neural*
by Alianna J. Maren, Ph.D. (Publication expected 2019)
Used with permission.

**Part 5:**

***We traced the dependence of the SSE on the input-to-hidden connection weight,***

***w(i,j).***

***Now we have to chain-rule all these little components together.***

***For that, please see the corresponding chapter in the book.***

*Statistical Mechanics, Neural Networks, and Artificial Intelligence*
by Alianna J. Maren, Ph.D. (In progress.)
Chapter 5: Backpropagation.