Deep Learning
How to use Deep Learning for Marketing?

University of Zurich[UZH]

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Luca Gaegauf - luca.gaegauf@bf.uzh.ch

Making machines think like humans.

How (Shallow) Neural Networks work

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Luca Gaegauf - luca.gaegauf@bf.uzh.ch

This session has 3 learning goals

After this lecture you should be able to:

1. Understand the concept of neural networks.
2. Understand the basic design building blocks of neural networks.
3. Understand how to estimate neural networks.

Design and initialize a neural network

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Luca Gaegauf - luca.gaegauf@bf.uzh.ch

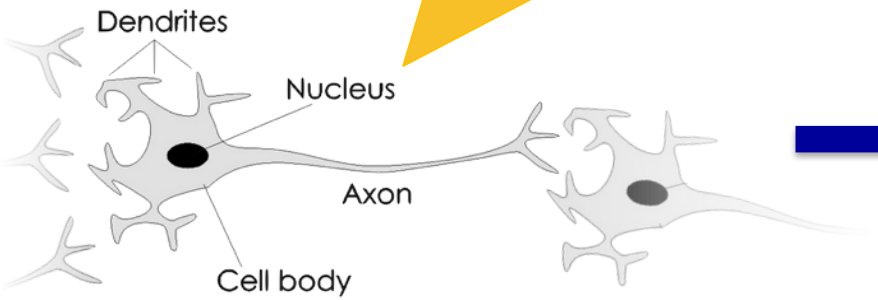
What is an artificial neural network?

https://hackernoon.com/interview-with-deep-learning-researcher-and-the-ganfather-dr-ian-goodfellow-cd300863ecff

Why neural...

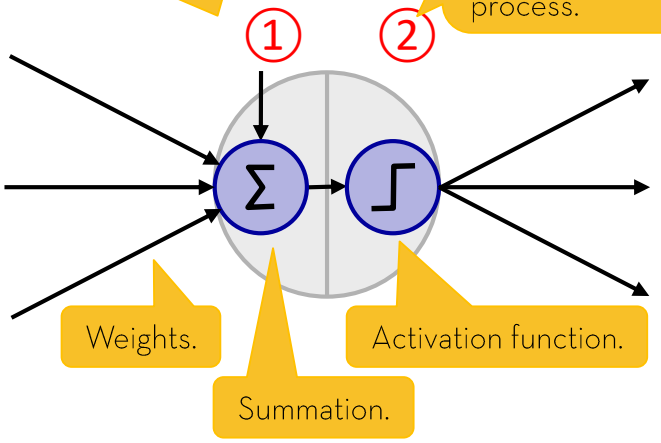
The basic building block of an artificial neural network is a neuron.

For the **biological neuron**, electrical signals from other neurons are conveyed to the cell body by dendrites; resultant electrical signals are sent along the axon to be distributed to other neurons.



The operation of the **artificial neuron** is analogous to (though much simpler than) the operation of the biological neuron: Activations from other neurons are summed at the neuron and passed through an activation function, after which the value is sent to other neurons.

Calculating the value of a neuron is a 2-step process.

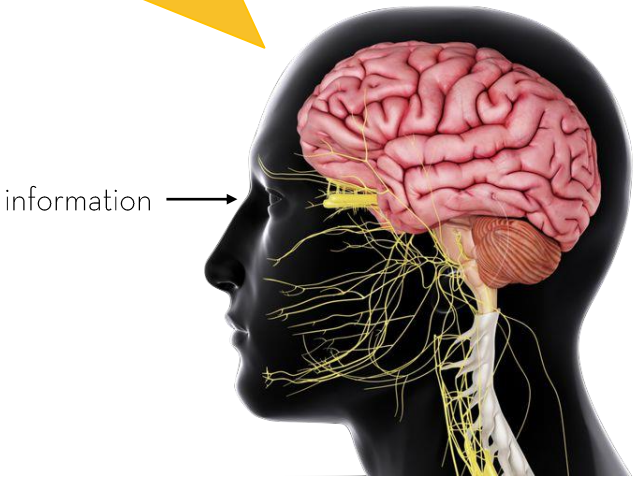


http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html

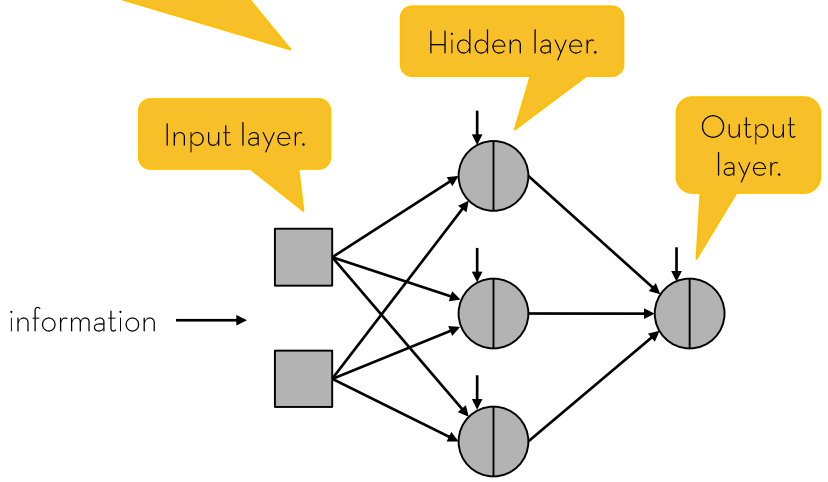
Why network...

We can stack layers of neurons together to form networks.

The **brain** is a collection of biological neurons that form pathways. Along these pathways flow chemical and electric signals. The brain processes information which is taken in through sensory organs.

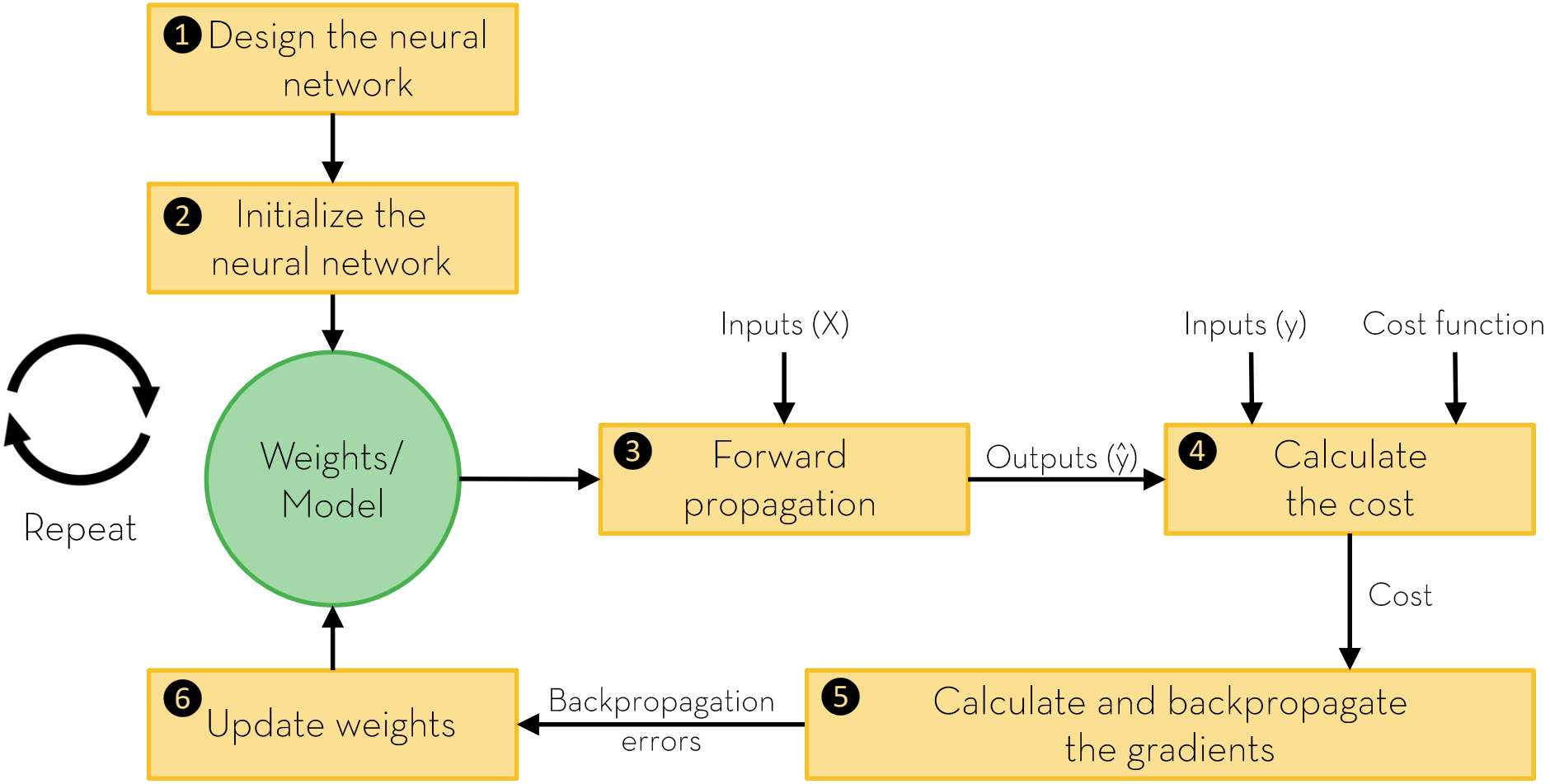


The operation of an **artificial neural networks** is analogous to (though much simpler than) the operation of the brain: input information flows through the pathways of the network and is processed by weighted summation and activation operations of the neurons.



http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html

Steps of estimating neural networks



https://datathings.com/blog/post/neuralnet/

Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

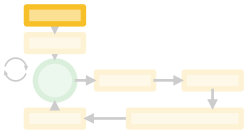
Repeat 3. – 6. until convergence criteria is met.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

Repeat 3. – 6. until convergence criteria is met.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network

1) Normalize the input data.

2) Based on the data, design the neural network:

   - Output Layer: Define number of output nodes.
   - Input Layer: Define number of input nodes.
   - Hidden Layer: Define number of hidden nodes.
   - Define the activation function on the hidden and output layer.
   - Define connections between nodes.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
a) Normalize the input data

Y	X1	X2
0	2	1.5
0	2.8	1.2
0	1.5	1
...	...	...
1	2.5	2
0	7.7	3.5

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

We **normalize our data** using the Z-Score algorithm:

X' = (X – Xmean) / (Xstandard_deviation)

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
b) Based on the data, design the neural network (1/8)

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

ON

Output nodes define what kind of data our neural network will result in.
As we have one positive output class, i.e. a binary classification problem, we choose one output node. The data is transformed in the correct output specification in this step.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
b) Based on the data, design the neural network (2/8)

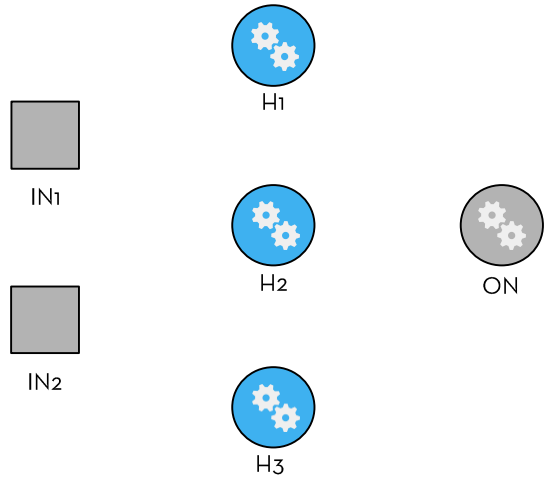
Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

Input nodes make sure our neural network receives the data it needs. Because we have two inputs, we choose two input nodes.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
b) Based on the data, design the neural network (3/8)

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

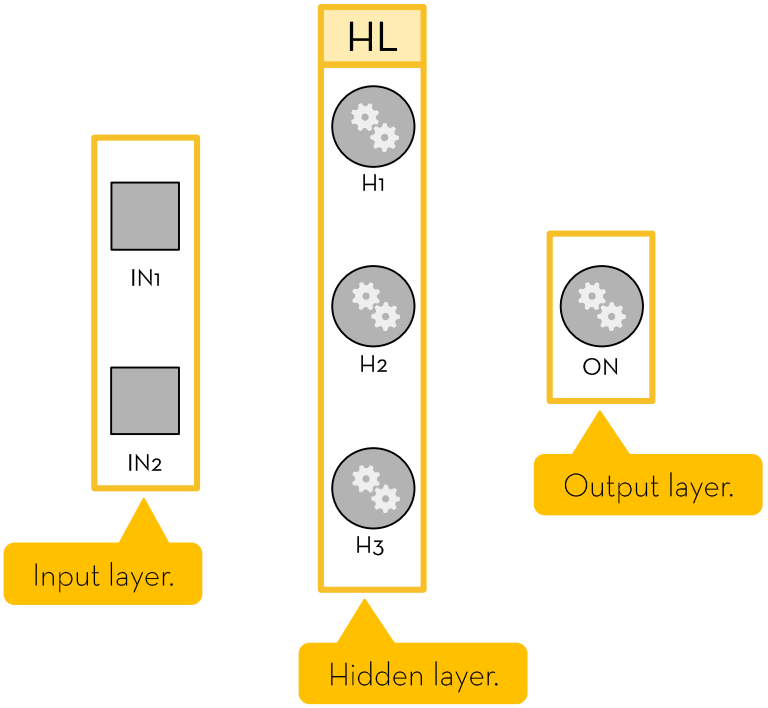


Hidden nodes further transform the data in order to make it easier to «solve the problem». The amount of hidden nodes is up to you.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
b) Based on the data, design the neural network (4/8)

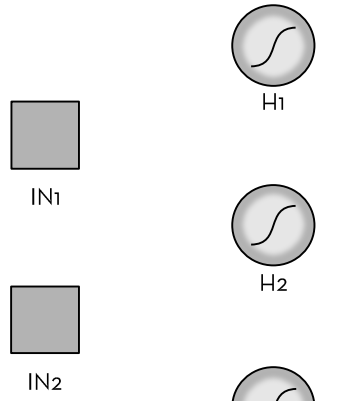
Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
b) Based on the data, design the neural network (5/8)

Y	X_1	X_2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34



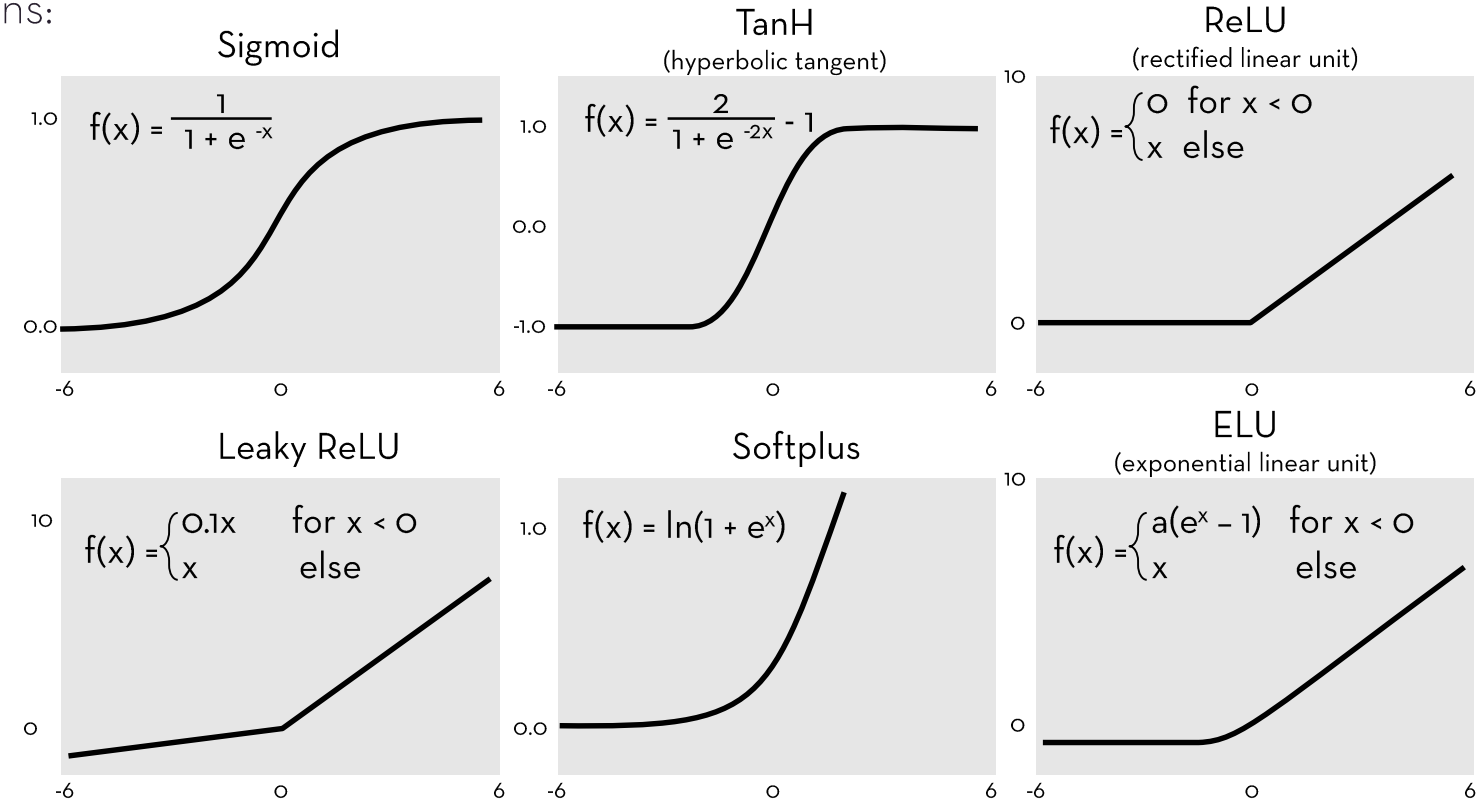
Activation functions allow the neural network to make non-linear approximations. We use the **sigmoid function** for the hidden and output layer.

Since we have a binary classification problem, the output node must have an activation function that restricts the output values to a range between 0 and 1

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

There are multiple activation functions, i.e. ways to introduce non-linearities in the neural network

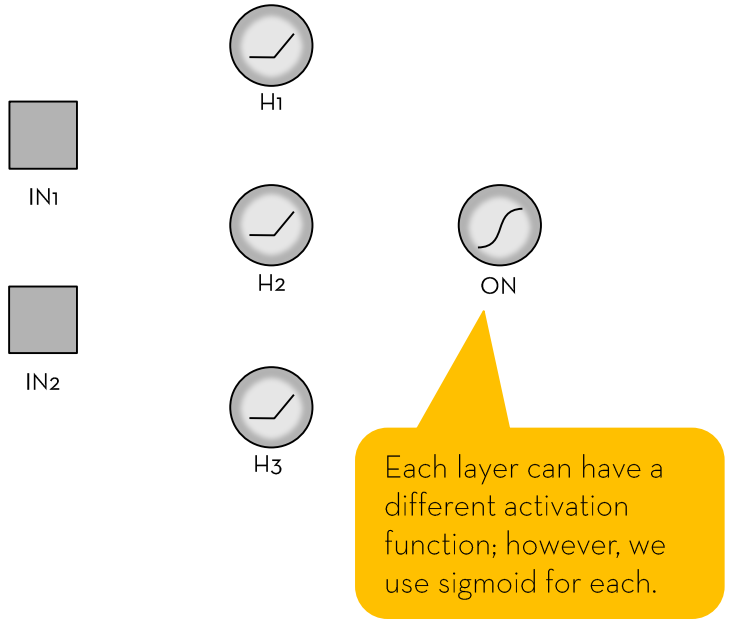
Instead of the sigmoid function, we could also use any other of the many activation functions:



https://stats.stackexchange.com/questions/115258/comprehensive-list-of-activation-functions-in-neural-networks-with-pros-cons
https://en.wikipedia.org/wiki/Activation_function

Step 1: Design the neural network
b) Based on the data, design the neural network (6/8)

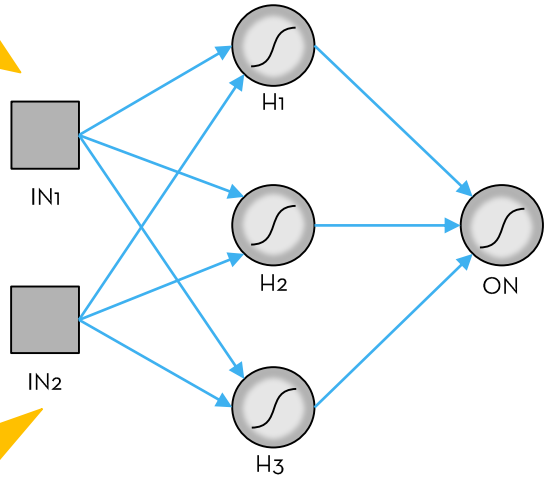
Y	X_1	X_2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34



Step 1: Design the neural network
b) Based on the data, design the neural network (7/8)

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

In our example of a traditional neural network, we **connect all nodes of a layer with all nodes from the next layer**. This is called a **dense** network. In advanced neural networks, some nodes are connected with only a few specific other nodes.



The nodes are **connected from one layer to each other**, forming a flow from the start to the end. This is called a **feedforward** network.

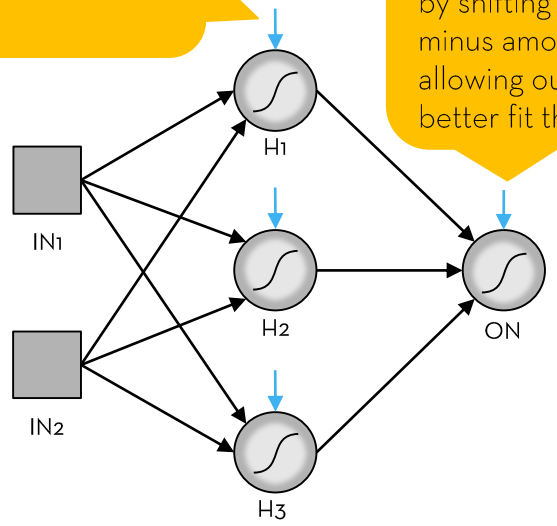
http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 1: Design the neural network
b) Based on the data, design the neural network (8/8)

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

In order to assist the hidden nodes and output node in transforming the data, we give each node an **individual bias**.

The **bias** allows us to further transform the data by shifting it by a plus or minus amount, therefore allowing our network to better fit the data.

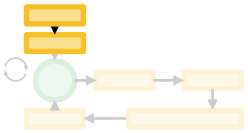


http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

Repeat 3. – 6. until convergence criteria is met.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 2: Initialize the neural network

a) Set the weights and biases: Set to random values between 1 and -1.

b) Set the inputs: Set to available values from the data.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

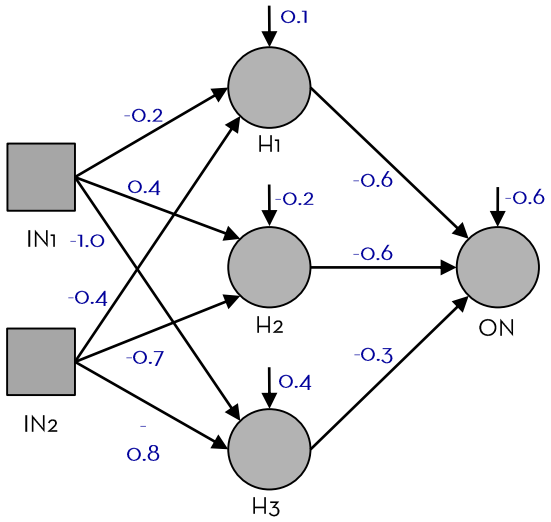
Step 2: Initialize the neural network
a) Set the weights and biases

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

Weights and biases are set randomly.

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

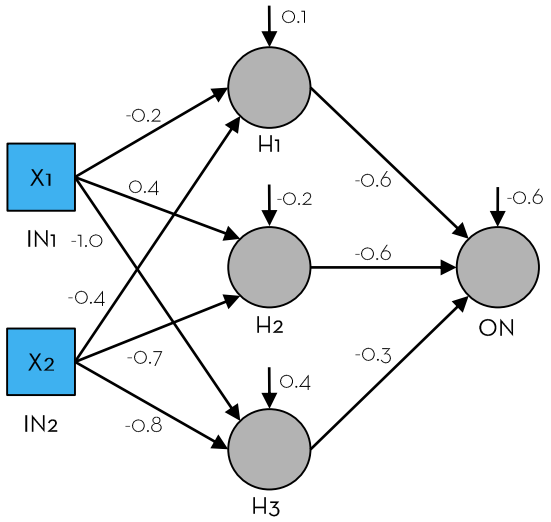


http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 2: Initialize the neural network
b) Set the inputs

Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

As inputs, we take X1 and X2.



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Estimating a neural network: Forward Propagation & Error calculation

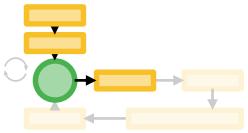
Dr. Markus Meierer - markus.meierer@business.uzh.ch | Luca Gaegauf - luca.gaegauf@bf.uzh.ch

Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

Repeat 3. – 6. until convergence criteria is met.

We will first illustrate the general principle of the forward pass before looking at a numerical example.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 3: Forward propagation
General principle

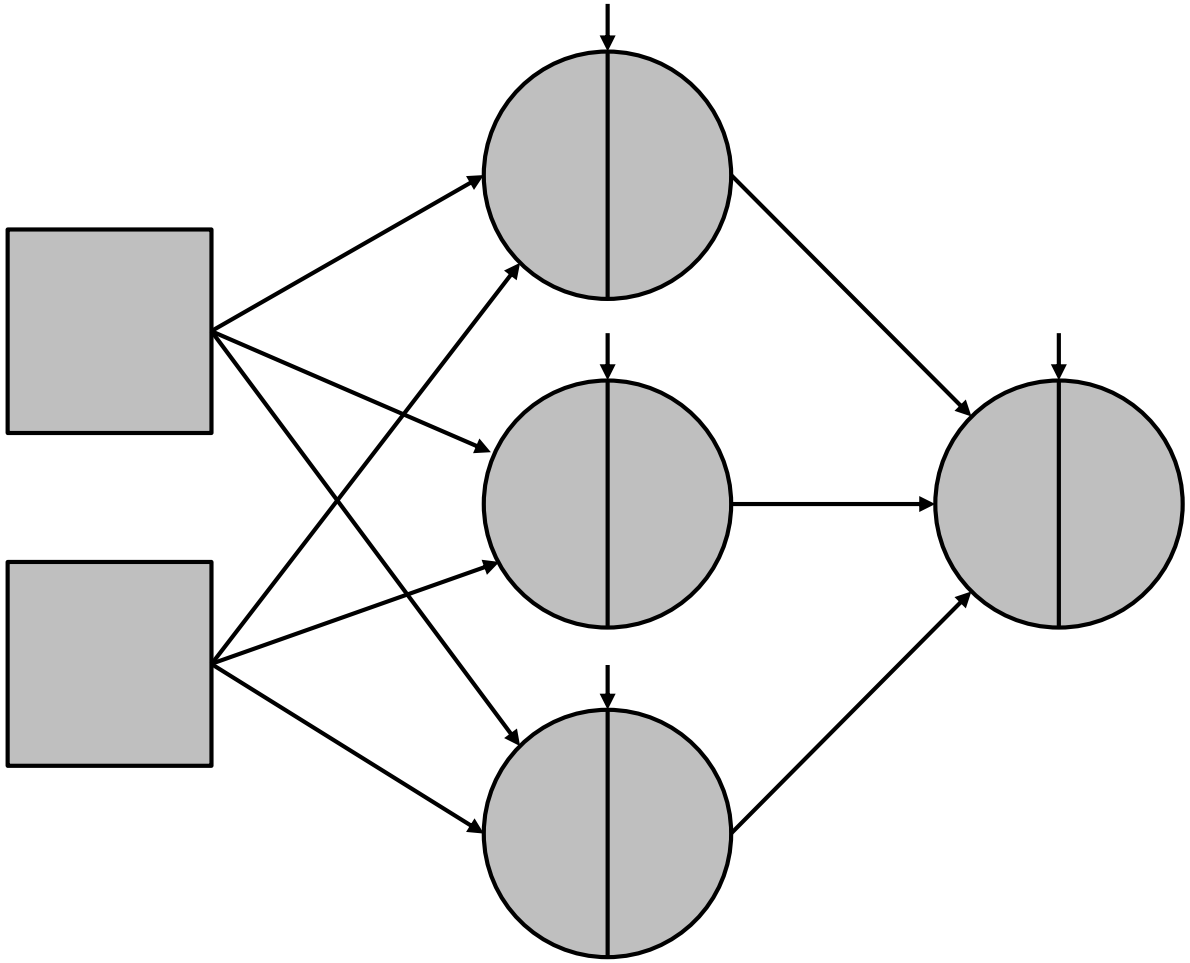
a) Calculate the values of the hidden layer.

b) Calculate the values of the output layer.

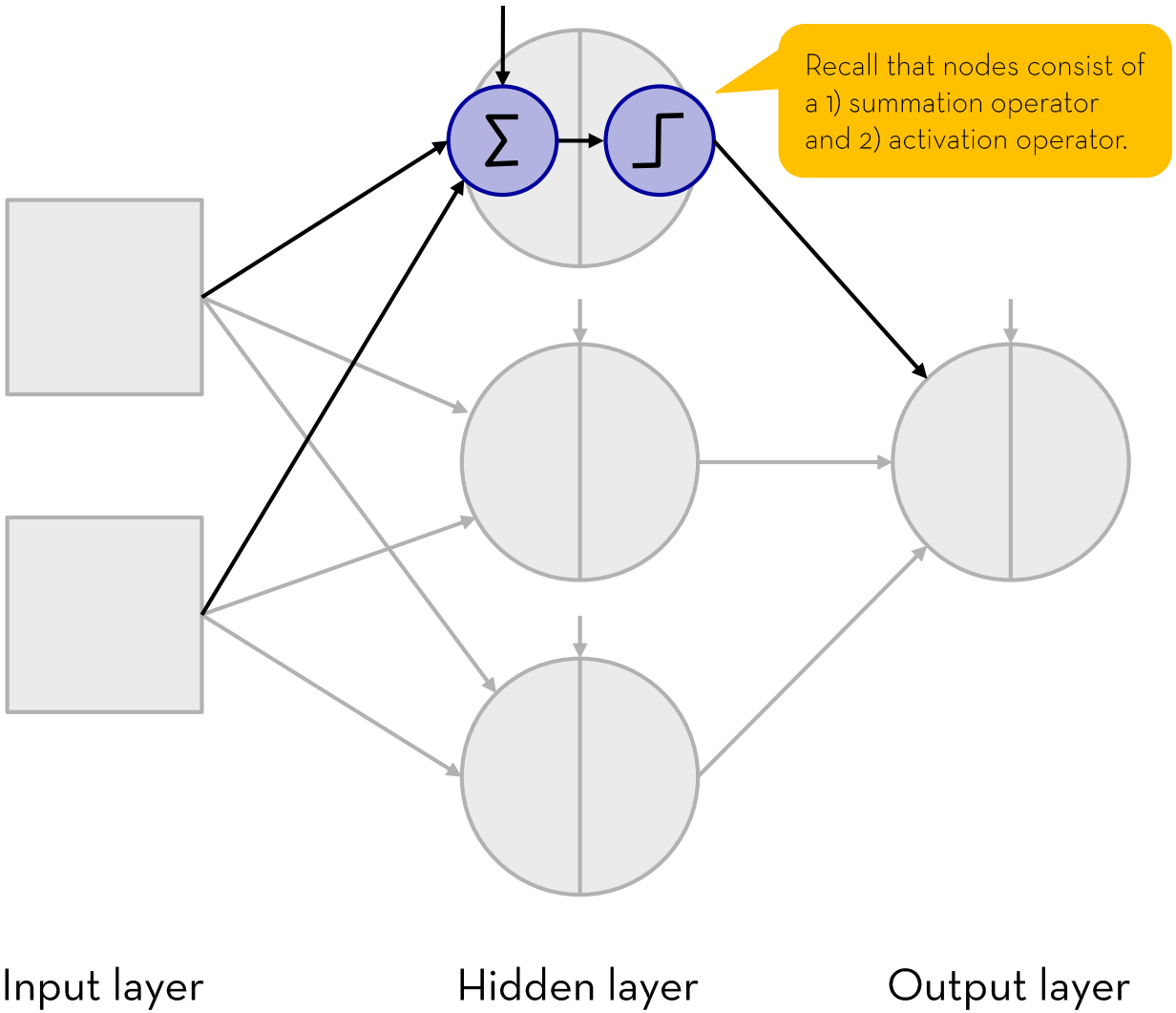
We work our way forward through the network.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

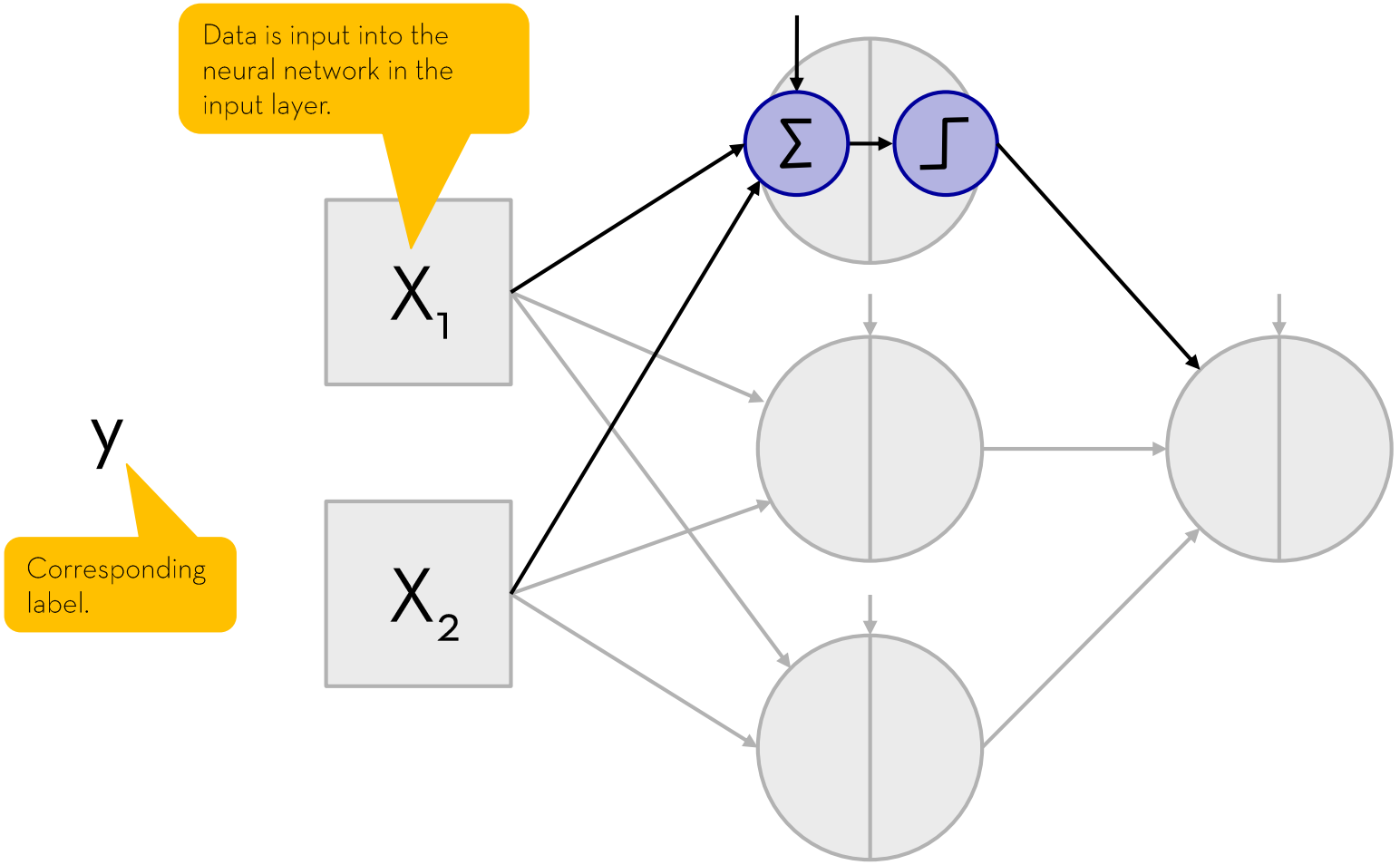
The general principle of forward propagation



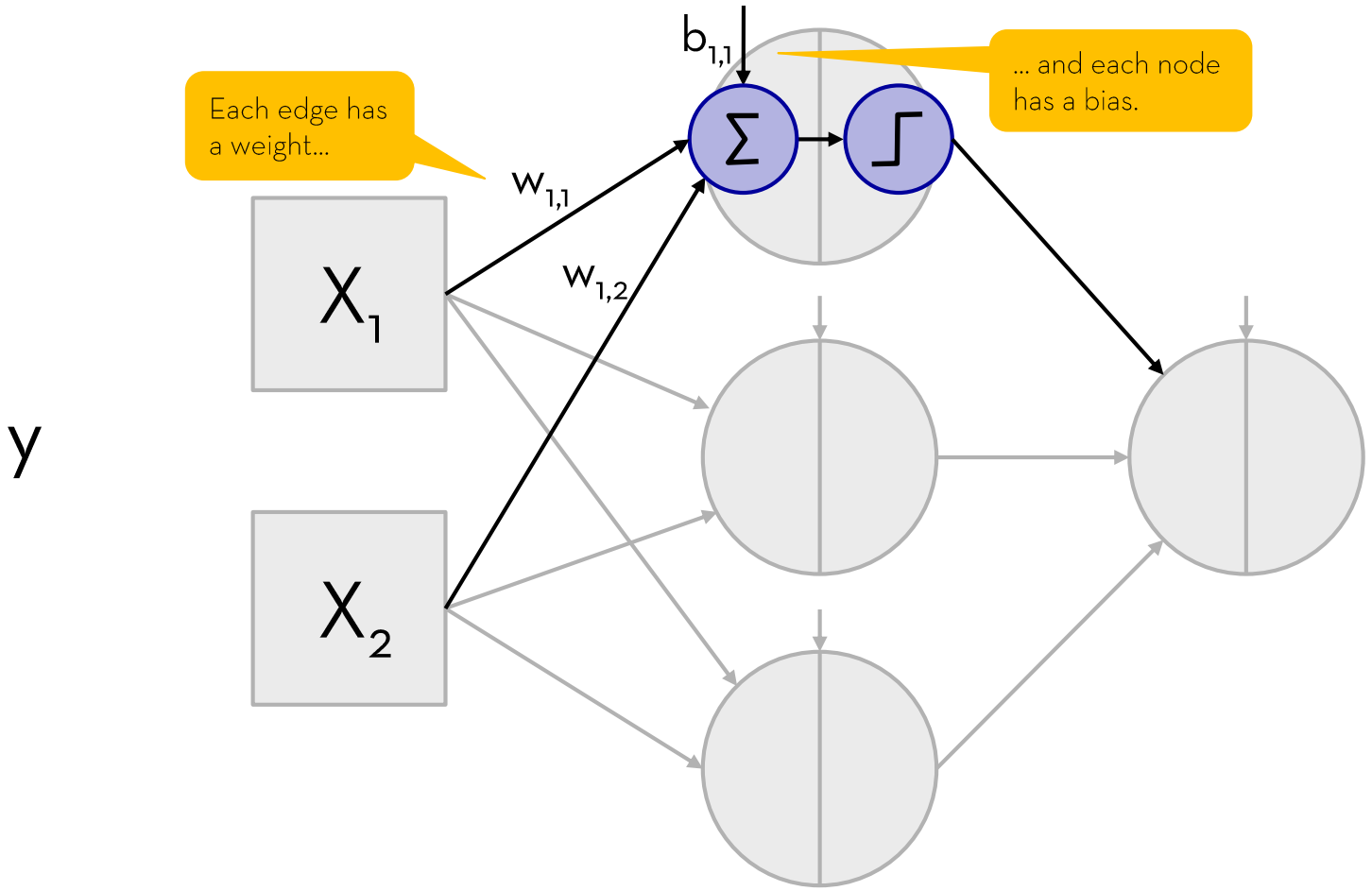
The general principle of forward propagation
Nodes process the data in 2 steps



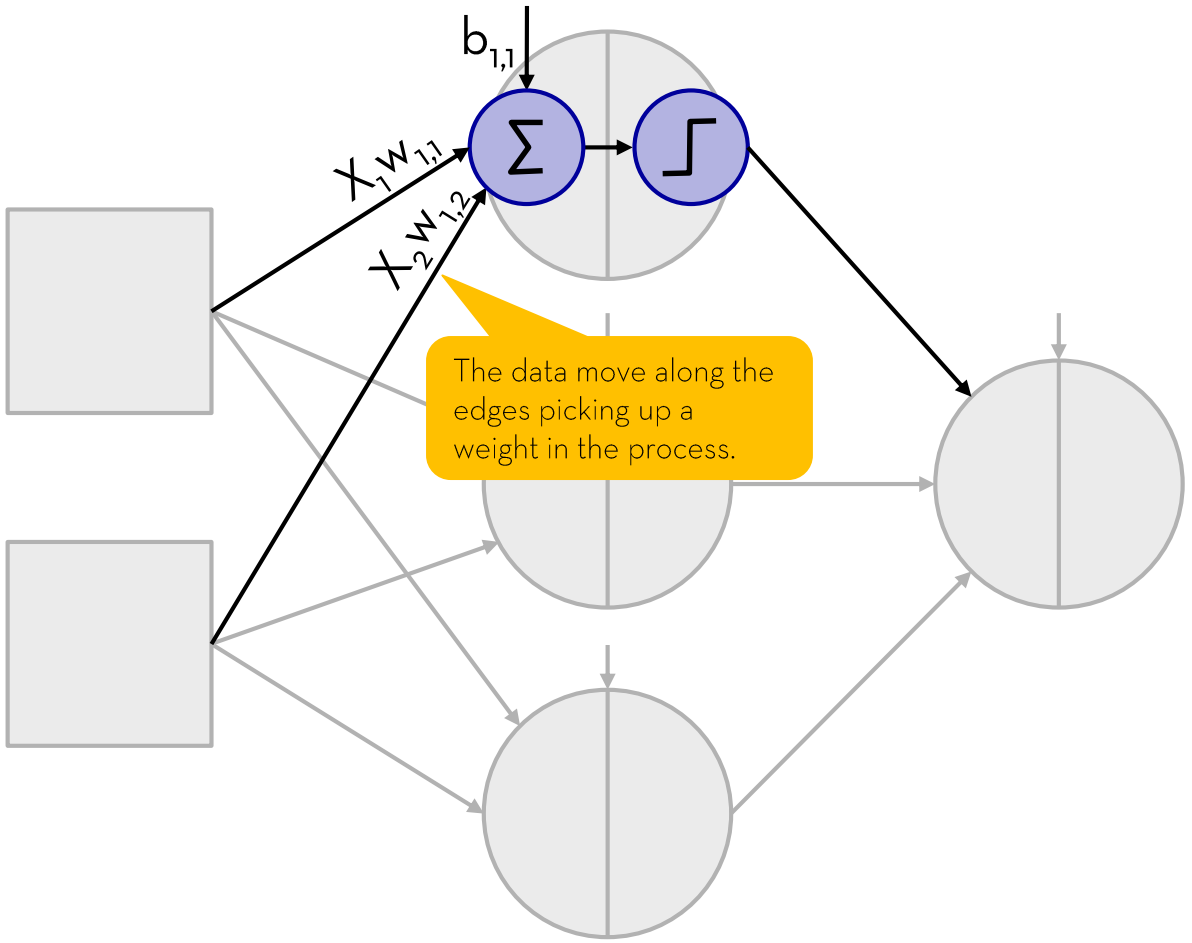
The general principle of forward propagation



The general principle of forward propagation
a) Calculate the values of the hidden layer (1/5)



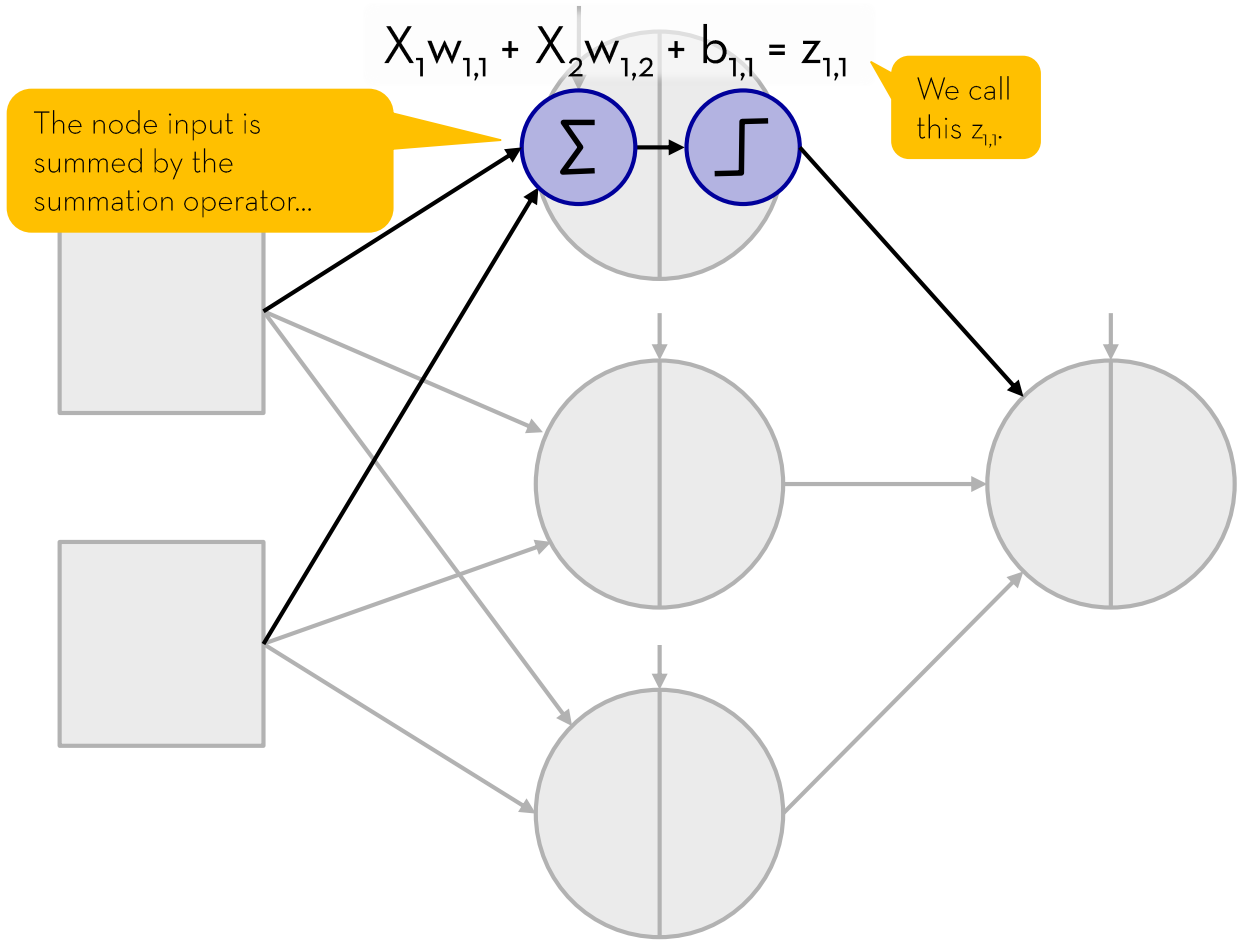
The general principle of forward propagation
a) Calculate the values of the hidden layer (2/5)



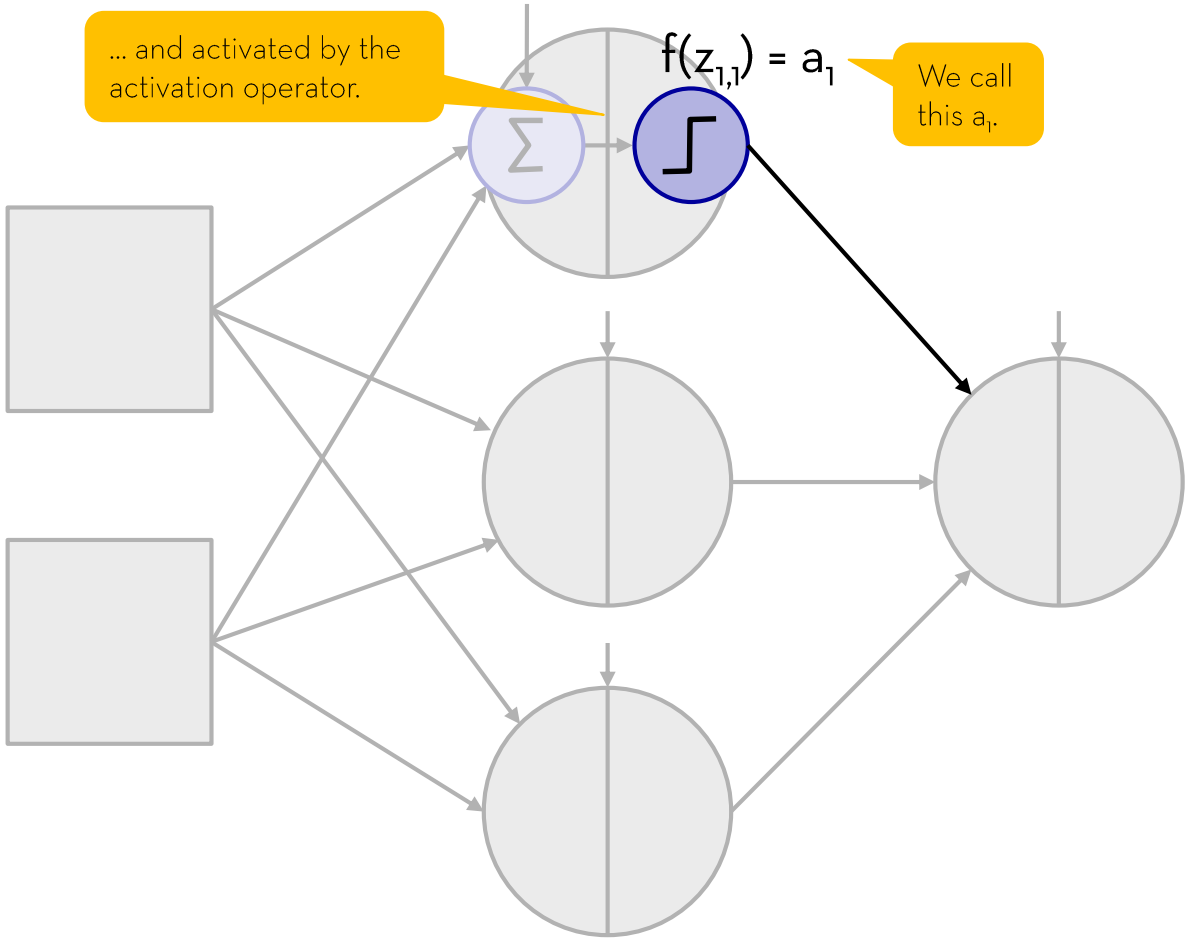
y

Label Input layer Hidden layer Output layer

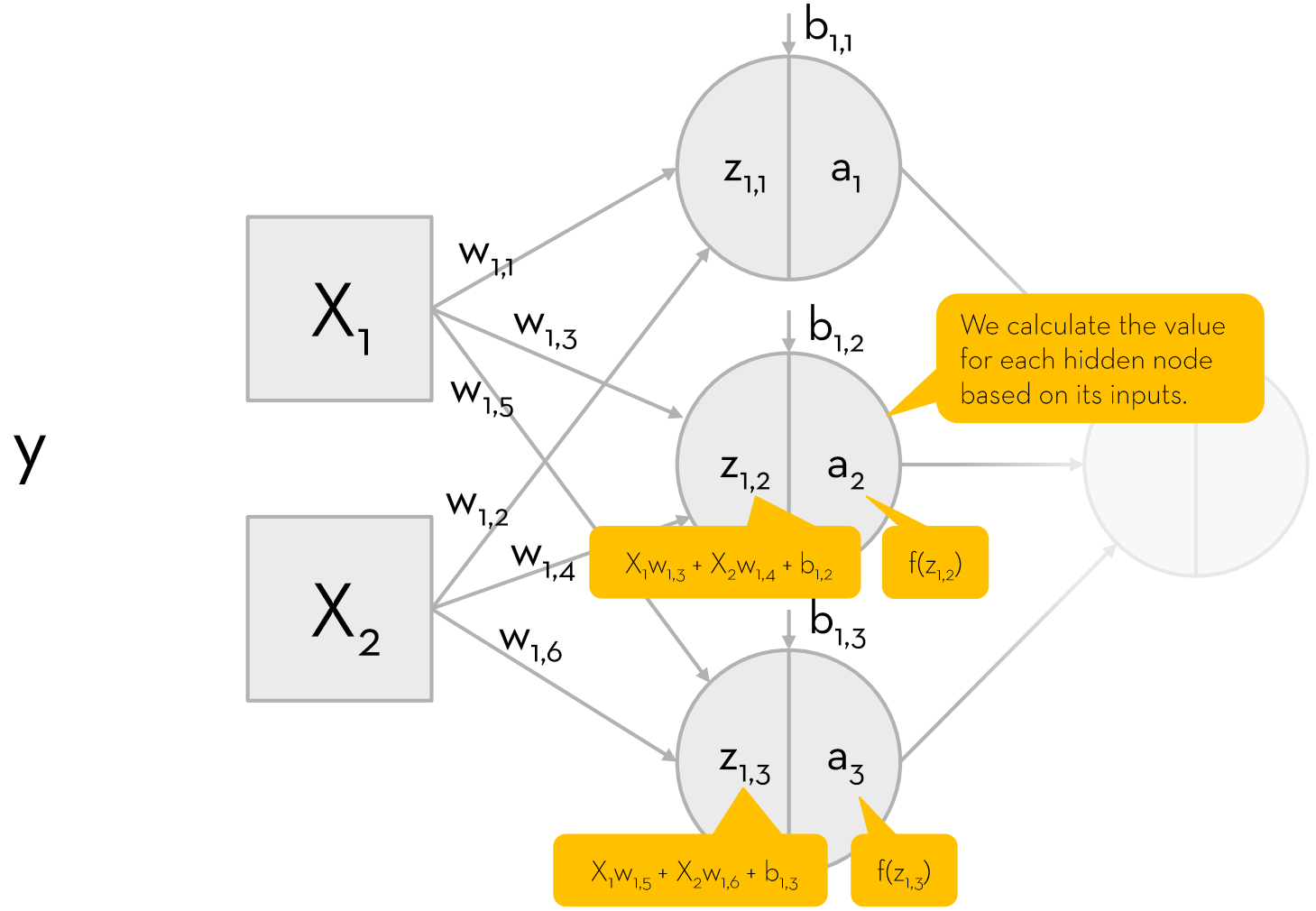
The general principle of forward propagation
a) Calculate the values of the hidden layer (3/5)



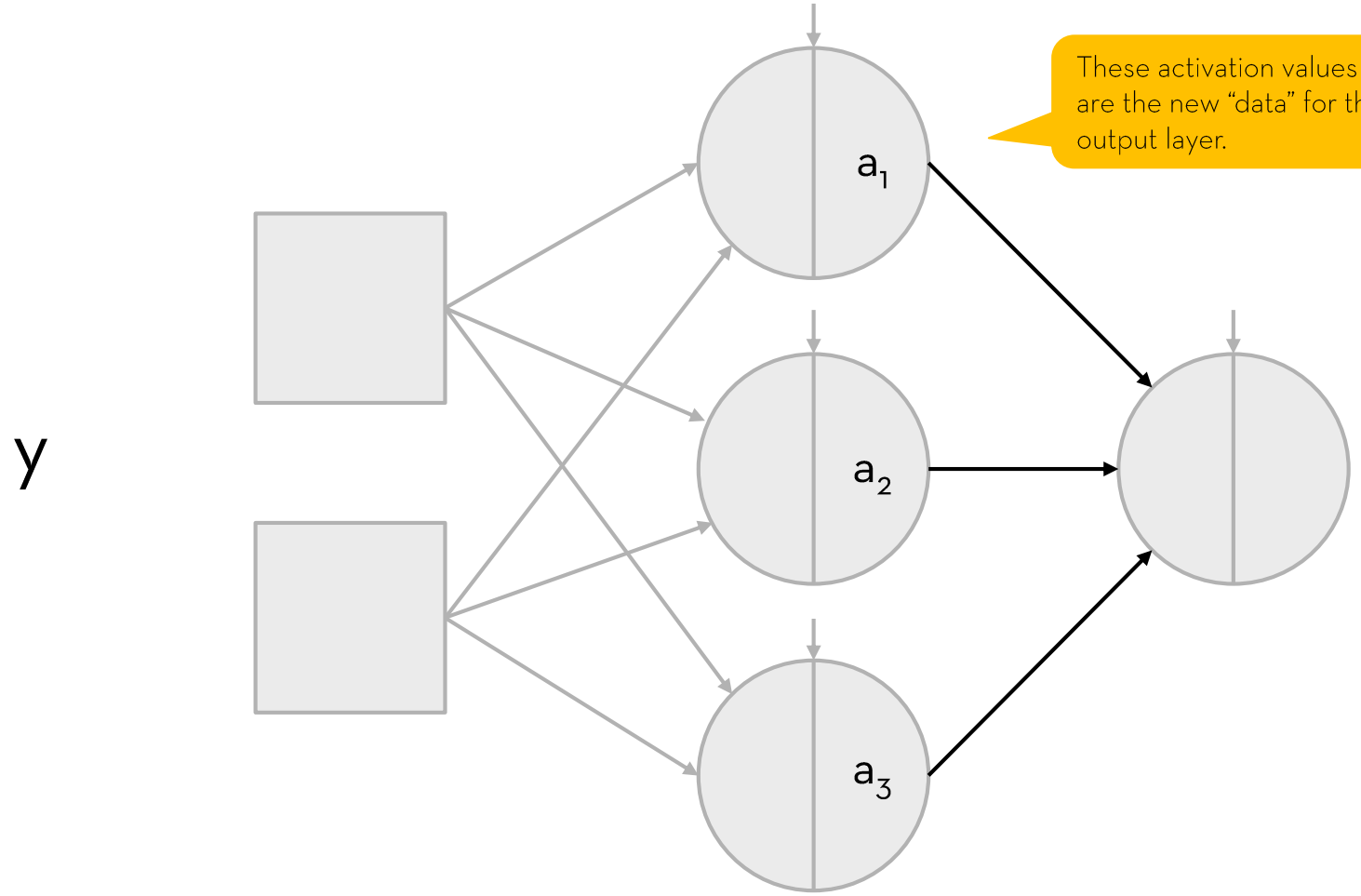
The general principle of forward propagation
a) Calculate the values of the hidden layer (4/5)



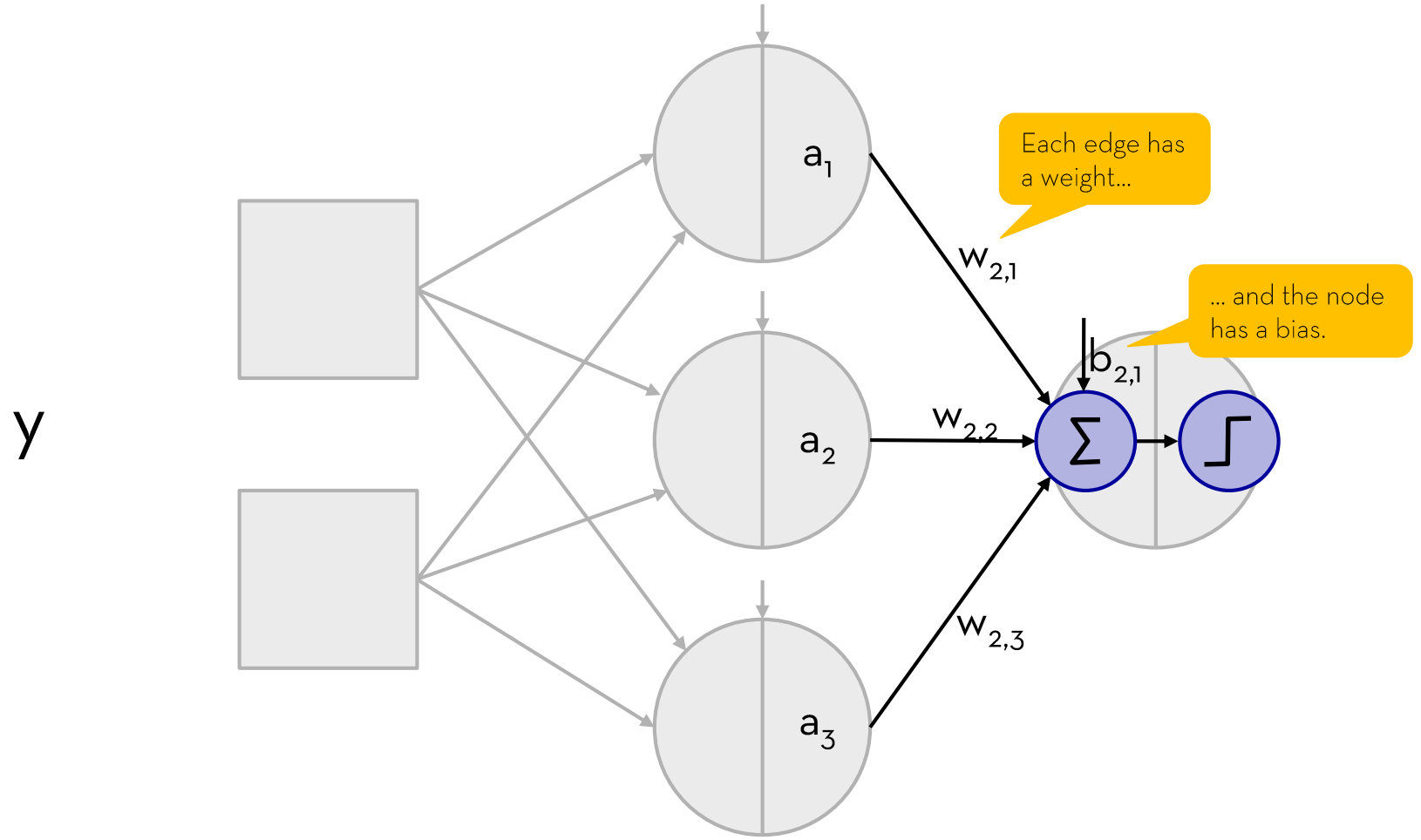
The general principle of forward propagation
a) Calculate the values of the hidden layer (5/5)



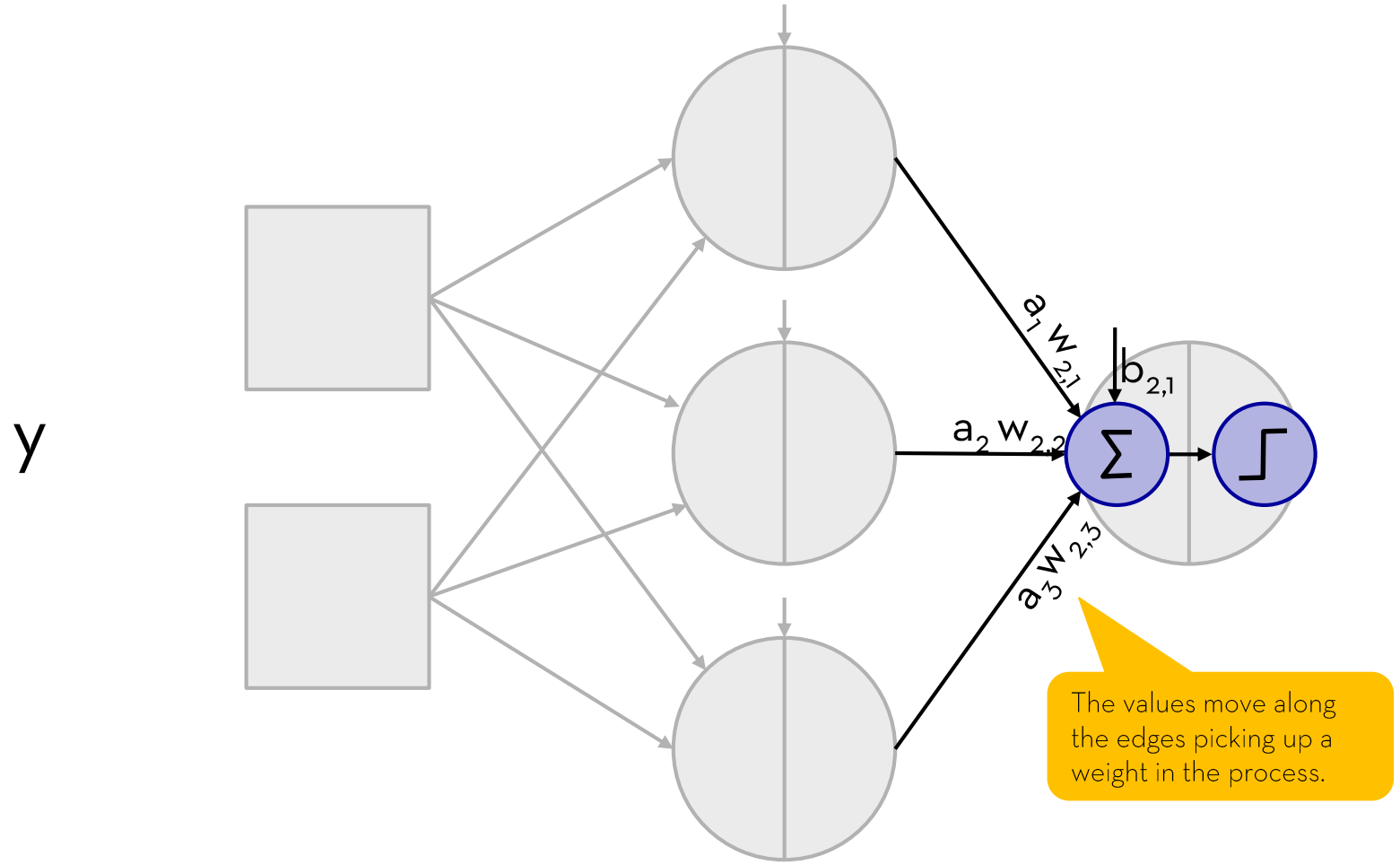
The general principle of forward propagation
b) Calculate the values of the output layer (1/5)



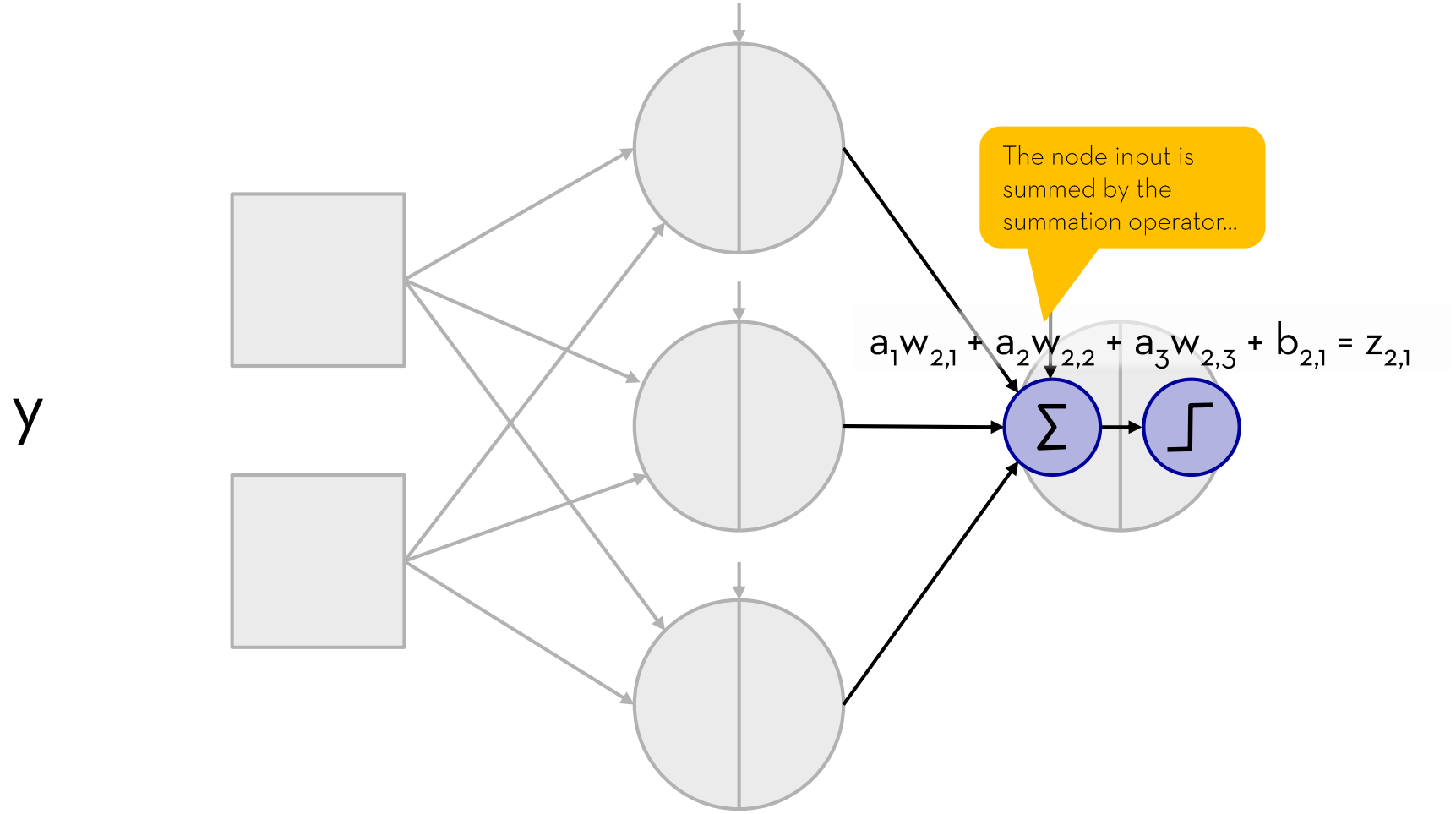
The general principle of forward propagation
b) Calculate the values of the output layer (2/5)



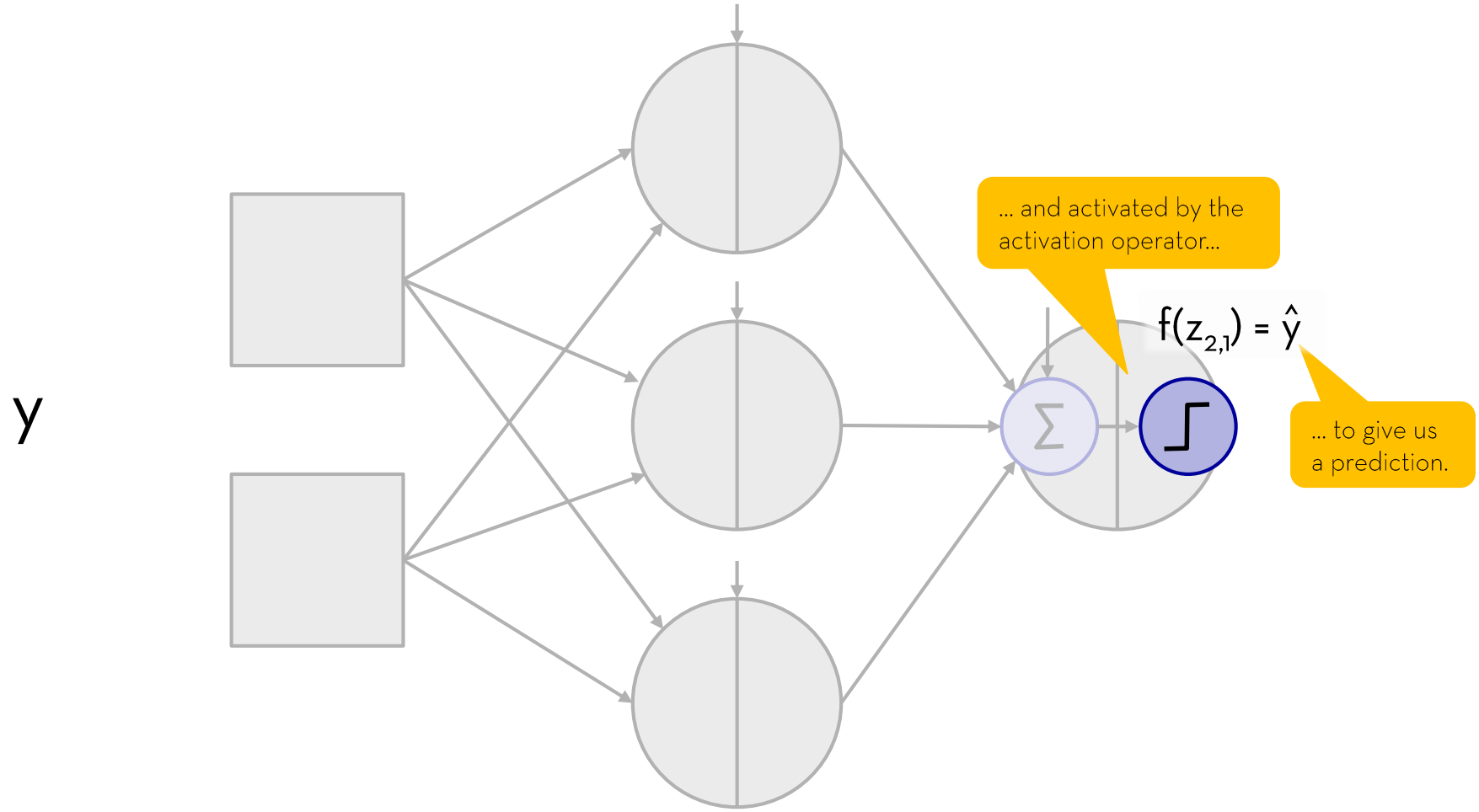
The general principle of forward propagation
b) Calculate the values of the output layer (3/5)



The general principle of forward propagation
b) Calculate the values of the output layer (4/5)

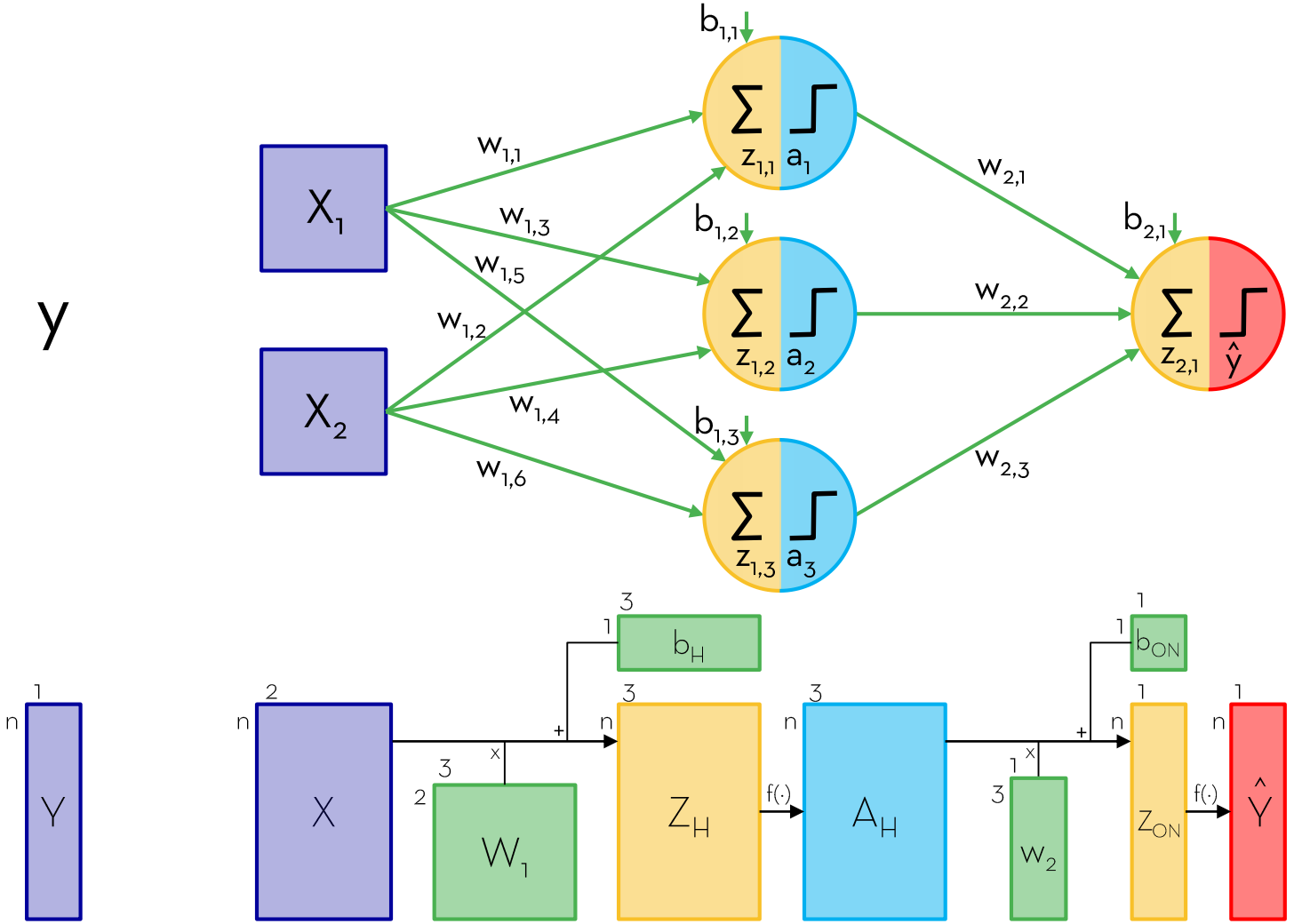


The general principle of forward propagation
b) Calculate the values of the output layer (5/5)



A summary of the general principle of forward propagation

y



http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Step 3: Forward propagation
Numerical example

a) Calculate the values of the hidden layer.

b) Calculate the values of the output layer.

We work our way forward through the network.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

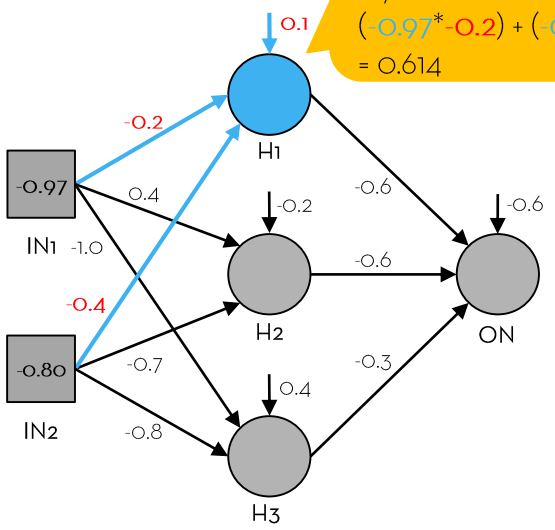
Step 3: Forward propagation
a) Calculate the values of the hidden layer (1/5)

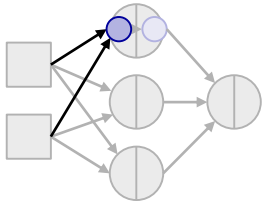
Y	X1	X2
0	-0.97	-0.80
0	-0.69	-0.97
0	-1.14	-1.09
...	...	...
1	-0.79	-0.52
0	1.04	0.34

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

We calculate the **input sum** for each neuron of the hidden layer:
(-0.97*-0.2) + (-0.80*-0.4) + 0.1
= 0.614





Step 3: Forward propagation
a) Calculate the values of the hidden layer (2/5)

Y	X1	X2	H1	H2	H3
0	-0.97	-0.80	0.65		
0	-0.69	-0.97			
0	-1.14	-1.09			
...	...	...			
1	-0.79	-0.52			
0	1.04	0.34			

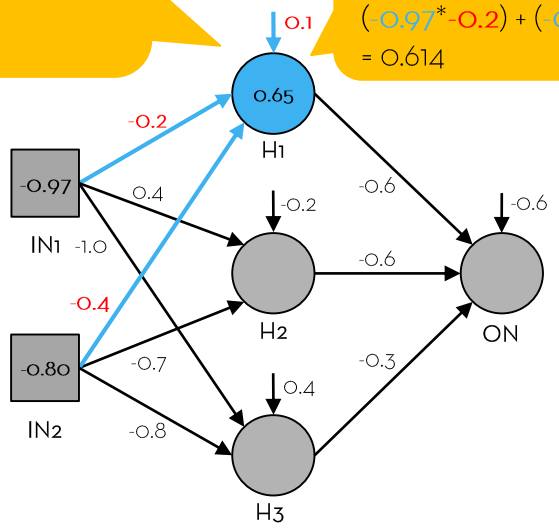
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

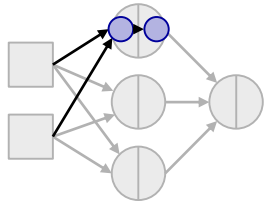
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

We then apply the **sigmoid function**, to get the output of a node (which is its value):

$f(x) = 1/(1+e^{-x})$
Here: f(0.614) = 0.65

We calculate the **input sum** for each hidden layer:
(-0.97*-0.2) + (-0.80*-0.4) + 0.1
= 0.614



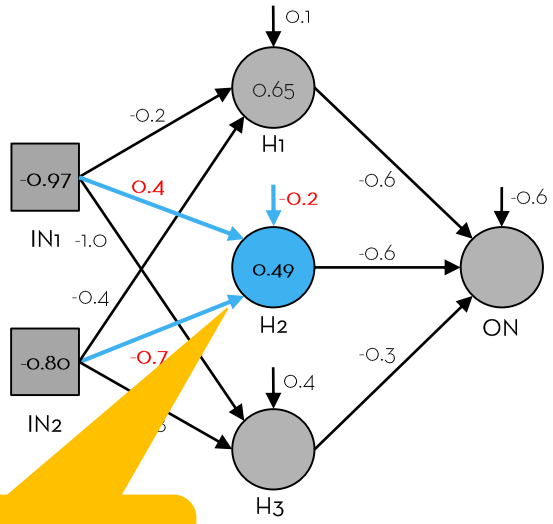


Step 3: Forward propagation
a) Calculate the values of the hidden layer (3/5)

Y	X1	X2	H1	H2	H3
0	-0.97	-0.80	0.65	0.49	
0	-0.69	-0.97			
0	-1.14	-1.09			
...	...	...			
1	-0.79	-0.52			
0	1.04	0.34			

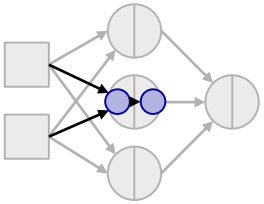
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6



Input sum:
(-0.97*0.4) + (-0.80*-0.7) + -0.2
= -0.026

f(-0.026) = 0.49

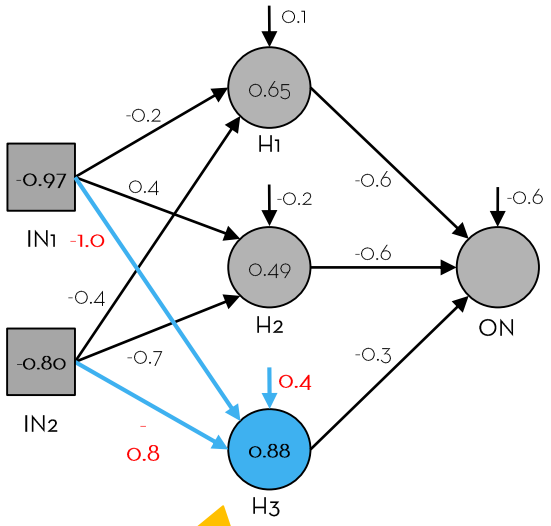


Step 3: Forward propagation
a) Calculate the values of the hidden layer (4/5)

Y	X1	X2	H1	H2	H3
0	-0.97	-0.80	0.65	0.49	0.88
0	-0.69	-0.97			
0	-1.14	-1.09			
...	...	...			
1	-0.79	-0.52			
0	1.04	0.34			

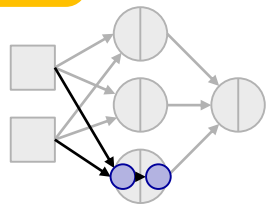
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6



Input sum:
(-0.97*-1.0) + (-0.80*-0.8) + 0.4
= 2.01

f(2.01) = 0.88



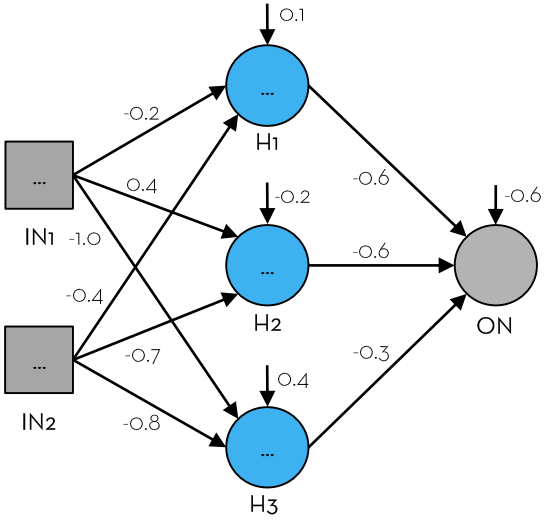
Step 3: Forward propagation
a) Calculate the values of the hidden layer (5/5)

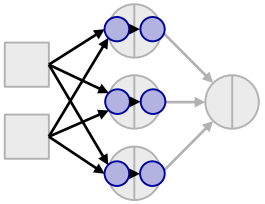
Y	X1	X2	H1	H2	H3
0	-0.97	-0.80	0.65	0.49	0.88
0	-0.69	-0.97	0.65	0.55	0.87
0	-1.14	-1.09	0.68	0.53	0.92
...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83
0	1.04	0.34	0.44	0.49	0.29

We do the same for all other observations.

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6



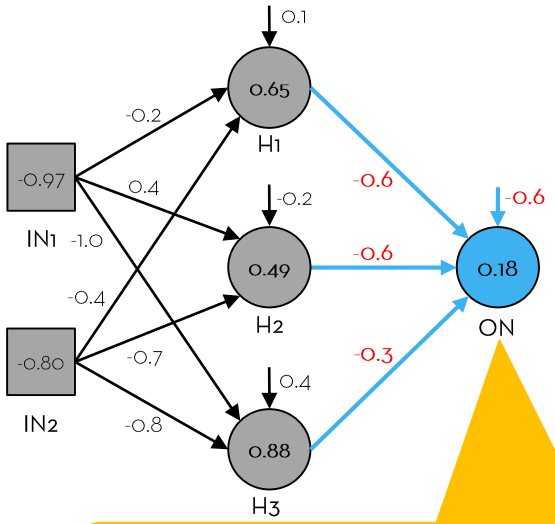


Step 3: Forward propagation
b) Calculate the values of the output layer (2/3)

Y	X1	X2	H1	H2	H3	Output
0	-0.97	-0.80	0.65	0.49	0.88	0.18
0	-0.69	-0.97	0.65	0.55	0.87	
0	-1.14	-1.09	0.68	0.53	0.92	
...	...	...	...	...	...	
1	-0.79	-0.52	0.61	0.46	0.83	
0	1.04	0.34	0.44	0.49	0.29	

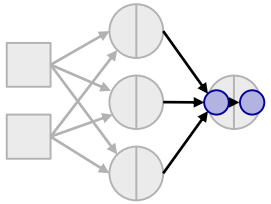
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6



Input sum:
(0.65*-0.6) + (0.49*-0.6) + (0.88*-0.3) + -0.6
= -1.55

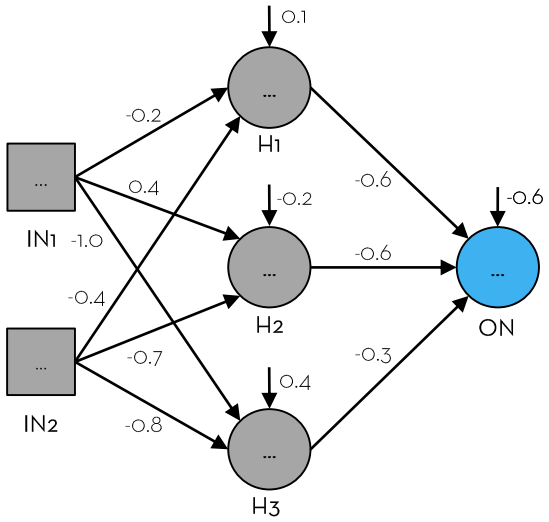
f(-1.55) = 0.18



Step 3: Forward propagation
b) Calculate the values of the output layer (3/3)

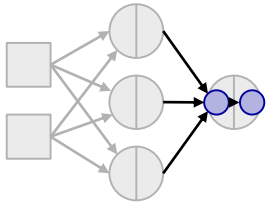
Y	X1	X2	H1	H2	H3	Output
0	-0.97	-0.80	0.65	0.49	0.88	0.18
0	-0.69	-0.97	0.65	0.55	0.87	0.17
0	-1.14	-1.09	0.68	0.53	0.92	0.17
...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18
0	1.04	0.34	0.44	0.49	0.29	0.22

We do the **same** for all other observations.



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

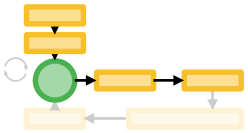
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6



Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

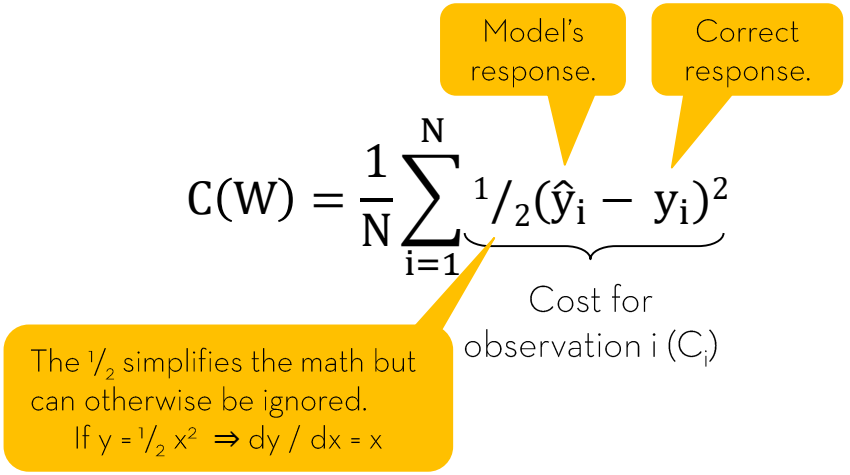
Repeat 3. – 6. until convergence criteria is met.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

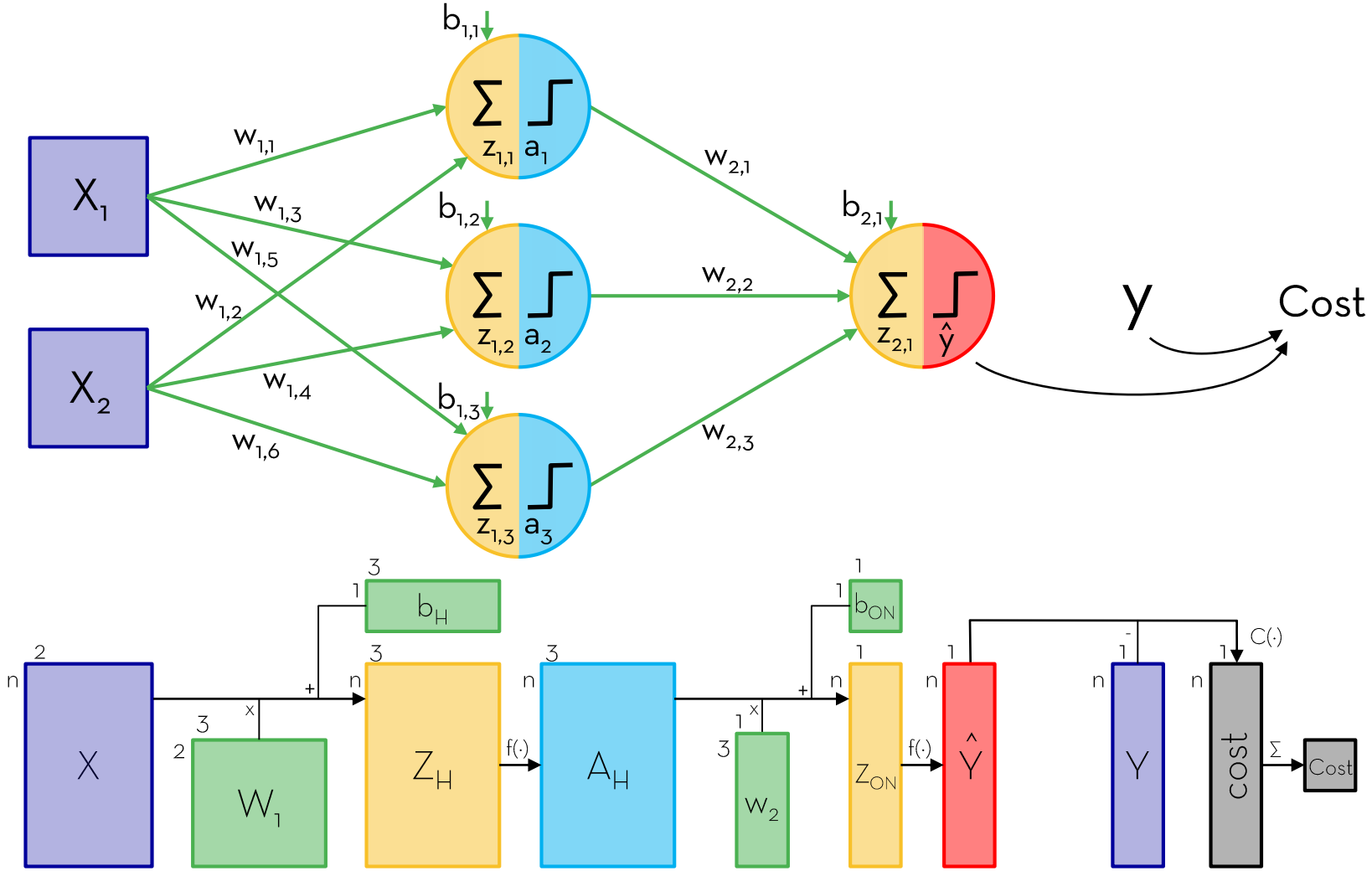
The general principle of calculating the cost

- The **cost function** (a.k.a. loss function) produces a scalar value that quantifies the model's error by **comparing the model's response with the correct response.**
- It is used to **evaluate the possible alternative solutions**, i.e. if one combination of parameters provides a better model fit than another combination of parameters.
- We need to define the cost function for our task. We will choose **mean squared error**:



http://cs231n.github.io/neural-networks-2/

A summary of the general principle of calculating the cost



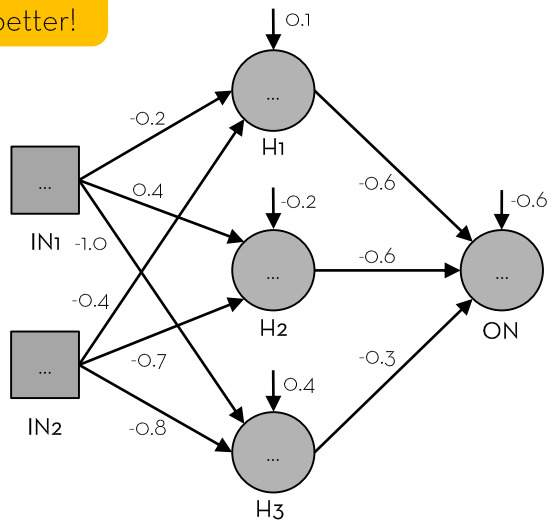
http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Step 4: Calculate the cost
Is our model good or bad? (1/2)

Our $\hat{y}$ Is labeled "output".

$Cost_i$ = $^1/_2 (output_i - y_i)^2$
→ The smaller, the better!

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.02
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.01
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.01
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.33
0	1.04	0.34	0.44	0.49	0.29	0.22	0.02



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

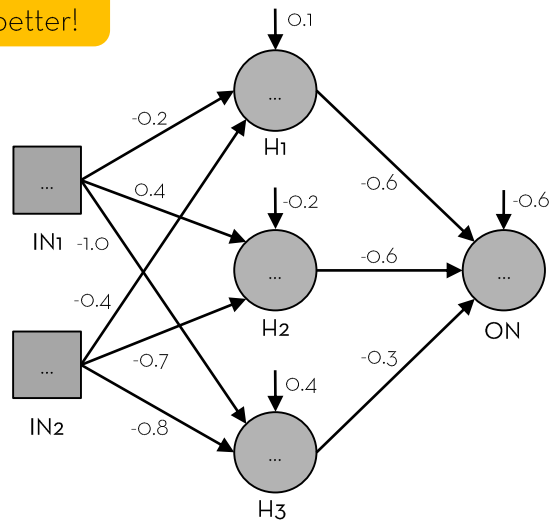
$$C(W) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{{}^1/_2(\hat{y}_i - y_i)^2}_{\text{Cost for observation i } (C_i)}$$

Step 4: Calculate the cost
Is our model good or bad? (2/2)

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.02
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.01
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.01
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.33
0	1.04	0.34	0.44	0.49	0.29	0.22	0.02
							Σ = 0.16

Cost$_i$ = $^1/_2$ (output$_i$ - y_i)^2
→ The smaller, the better!

Our **averaged up cost function** returns 0.16.



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

$$C(W) = \frac{1}{N}\sum_{i=1}^{N} {}^1/_2(\hat{y}_i - y_i)^2$$

Estimating a neural network: Backpropagation

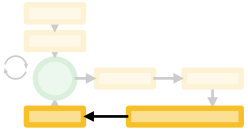
Dr. Markus Meierer - markus.meierer@business.uzh.ch | Luca Gaegauf - luca.gaegauf@bf.uzh.ch

Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

Repeat 3. – 6. until convergence criteria is met.

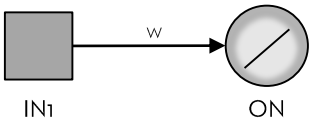
The remaining 2 steps constitute **gradient descent**. Gradient descent is the first-order optimization technique we use to train a neural network.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

If we only have one weight, it is easy to visualize the cost function

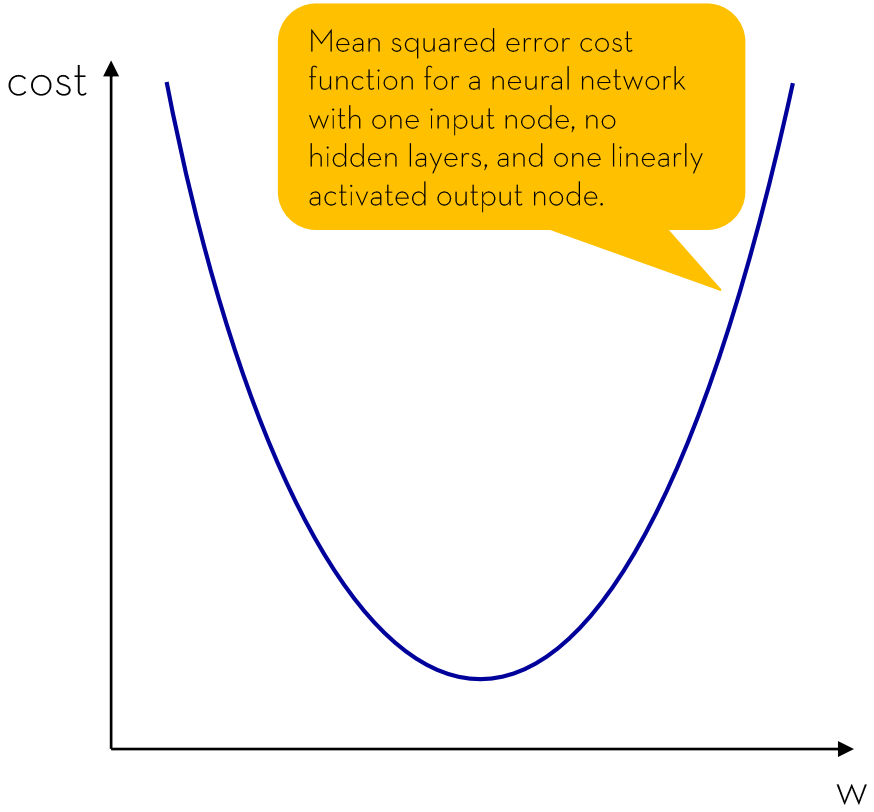
For a very simple neural network:



it is simple to visualize the cost function:

$$C(W) = \frac{1}{N}\sum_{i=1}^{N} {}^{1}/_{2}(xiw - y_i)^2.$$

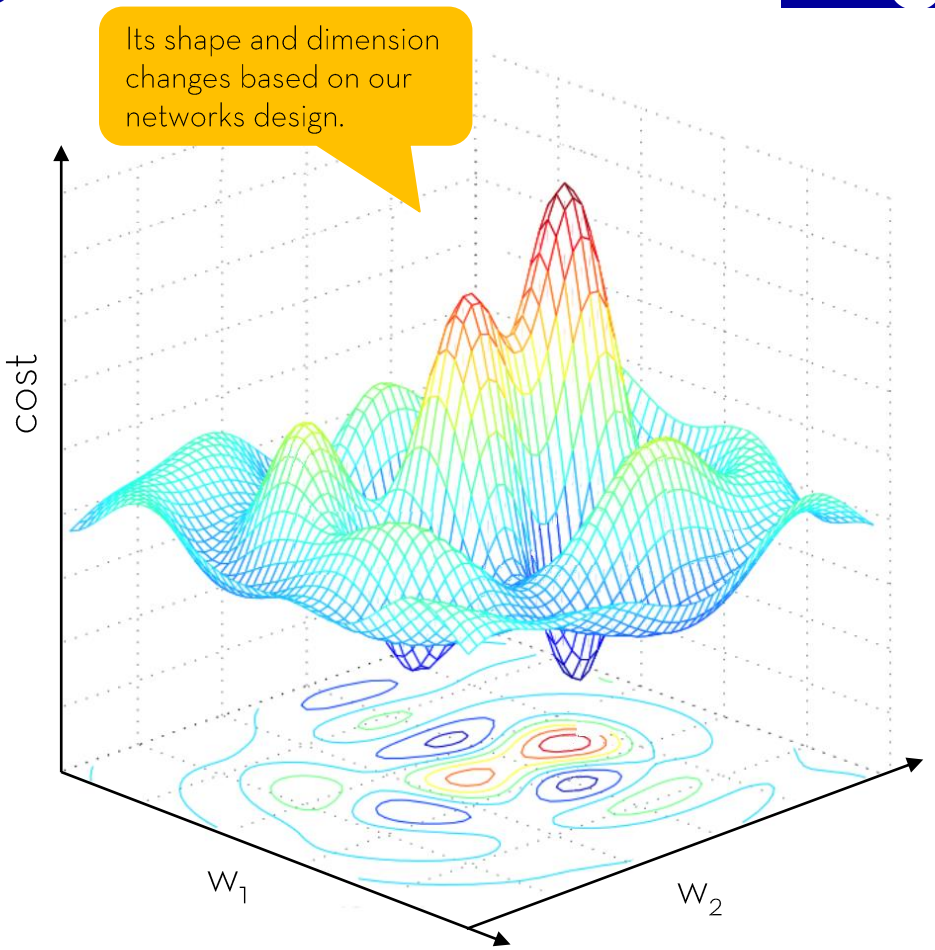
Once we have more than one weight, the cost function is harder to visualize.



Neural network cost functions can be very non-linear and difficult to optimize

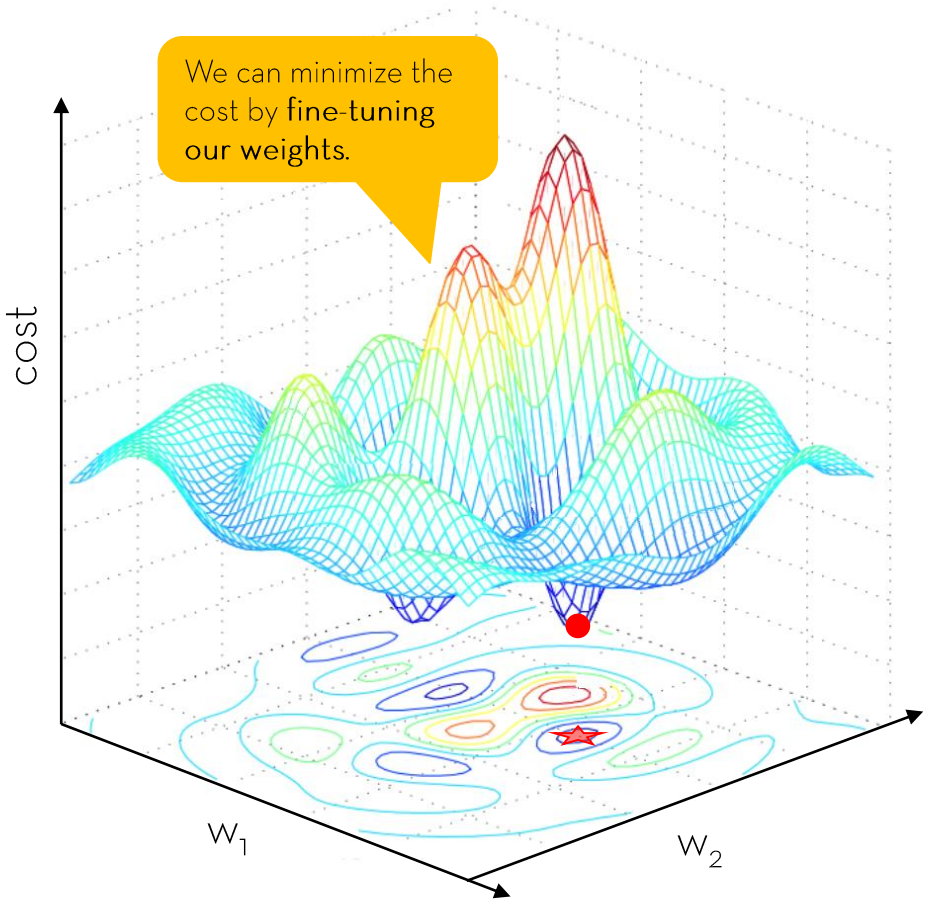
The cost is a hyper-dimensional non-linear and non-convex function whose shape **we don't know**.

- **Hyper-dimensional** because we are training many weights (in our case 9 weights and 4 biases).
- **Non-linear** and **non-convex** because of the activation functions and cost function.



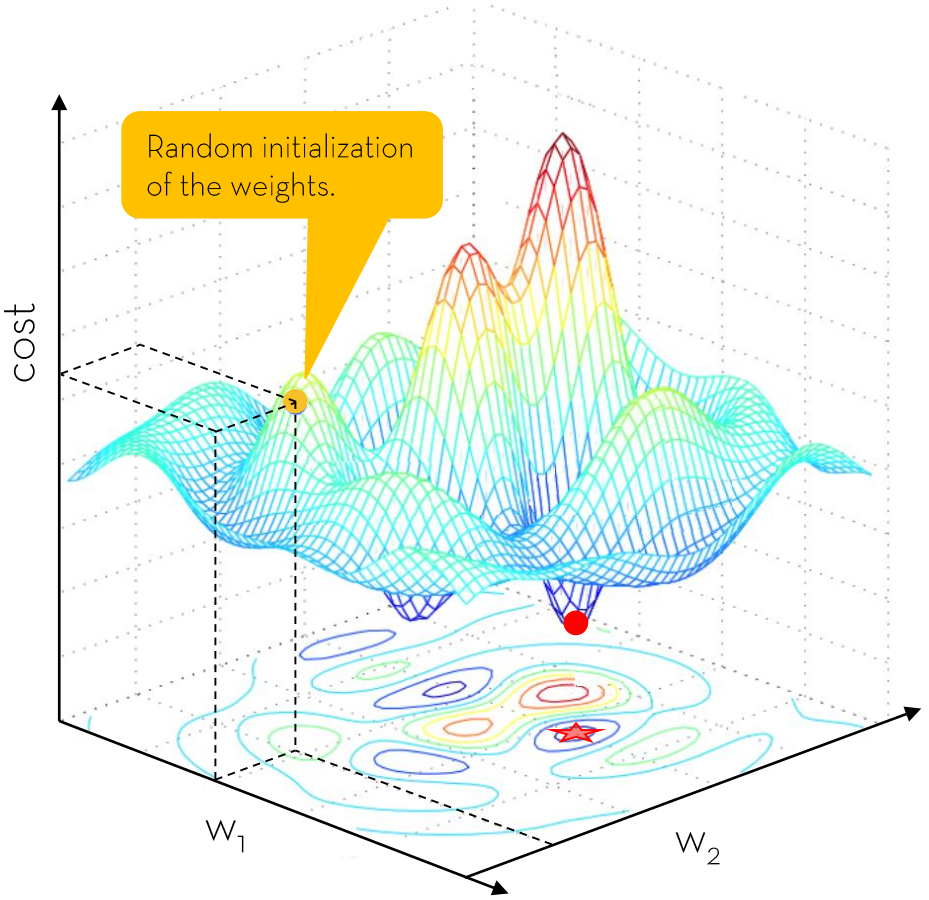
How do we optimize our weights?

- We are trying to find the **optimal weights** – i.e., the combination of weights that minimizes the cost function.



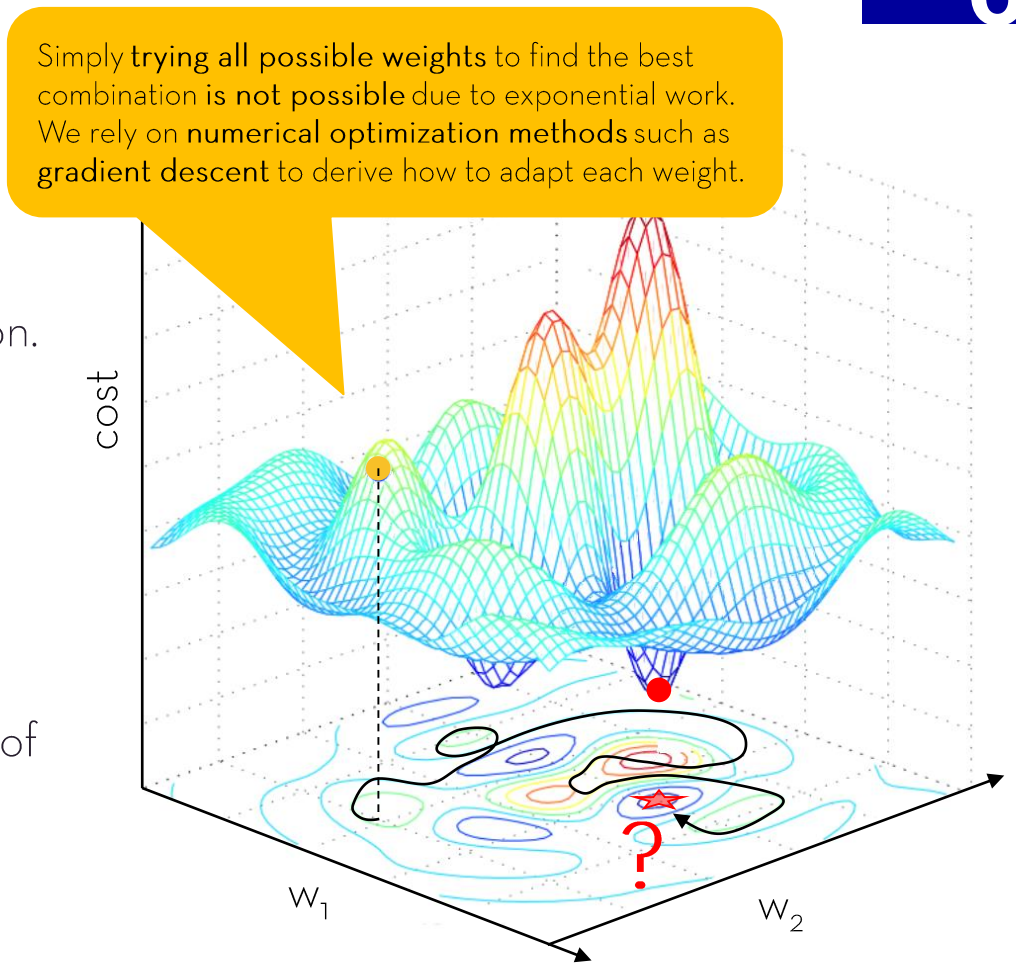
How do we optimize our weights?

- We are trying to find the **optimal weights** – i.e., the combination of weights that minimizes the cost function.
- By randomly initializing the weights (step 2) we start at a random point on the function.



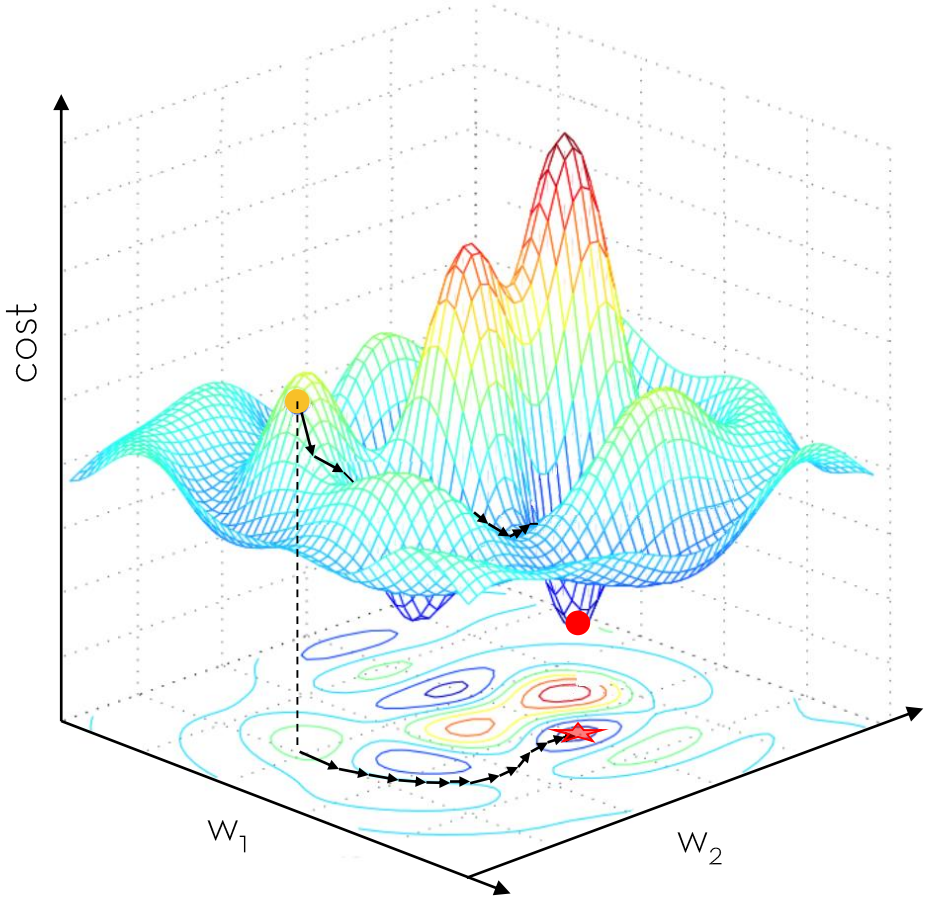
How do we optimize our weights?

- We are trying to find the **optimal weights** – i.e., the combination of weights that minimizes the cost function.
- By randomly initializing the weights (step 2) we start at a random point on the function.
- How do we find the best combination of weights? We use **gradient descent** to navigate to the lowest cost solution.



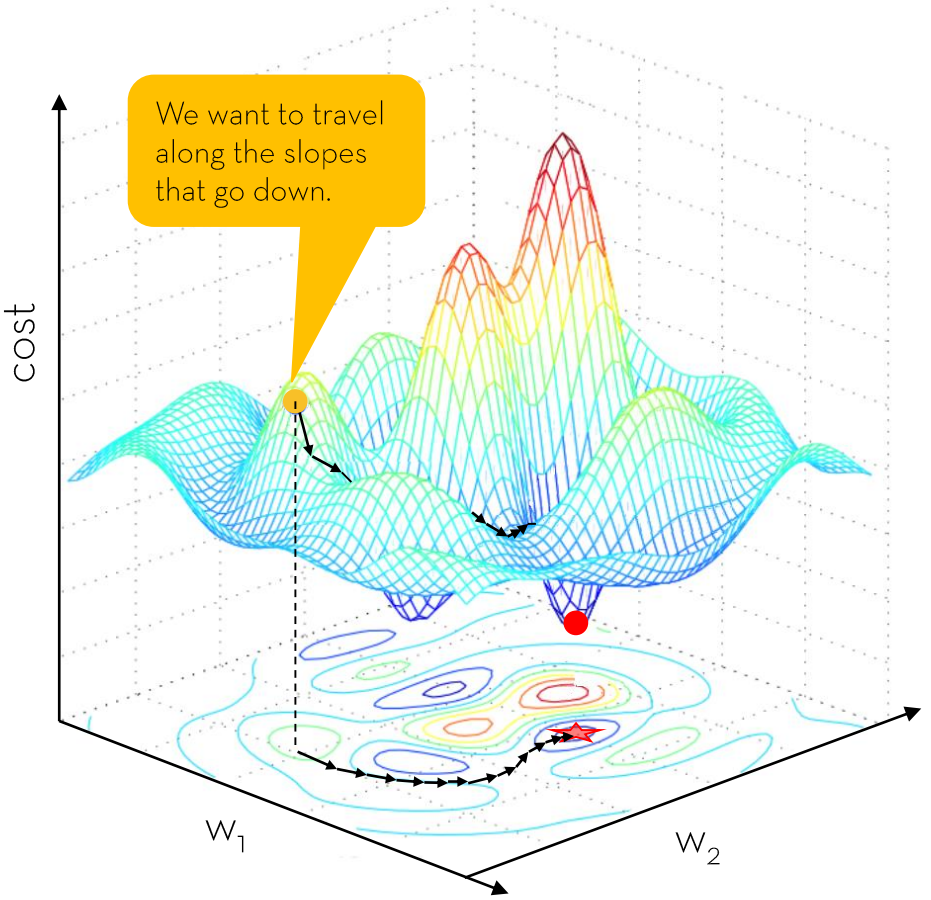
Why gradient?

- The gradient tells us in **which direction** and **how far we need to go** for each weight.



Why descent?

- The gradient tells us in **which direction** and **how far we need to go** for each weight.
- Because we are trying to **minimize the cost**, we need to travel in the direction that **descends the gradient**.
- That means that we want to update the weights such that we travel in the direction in which the gradient is negative.



A quick refresher on gradients

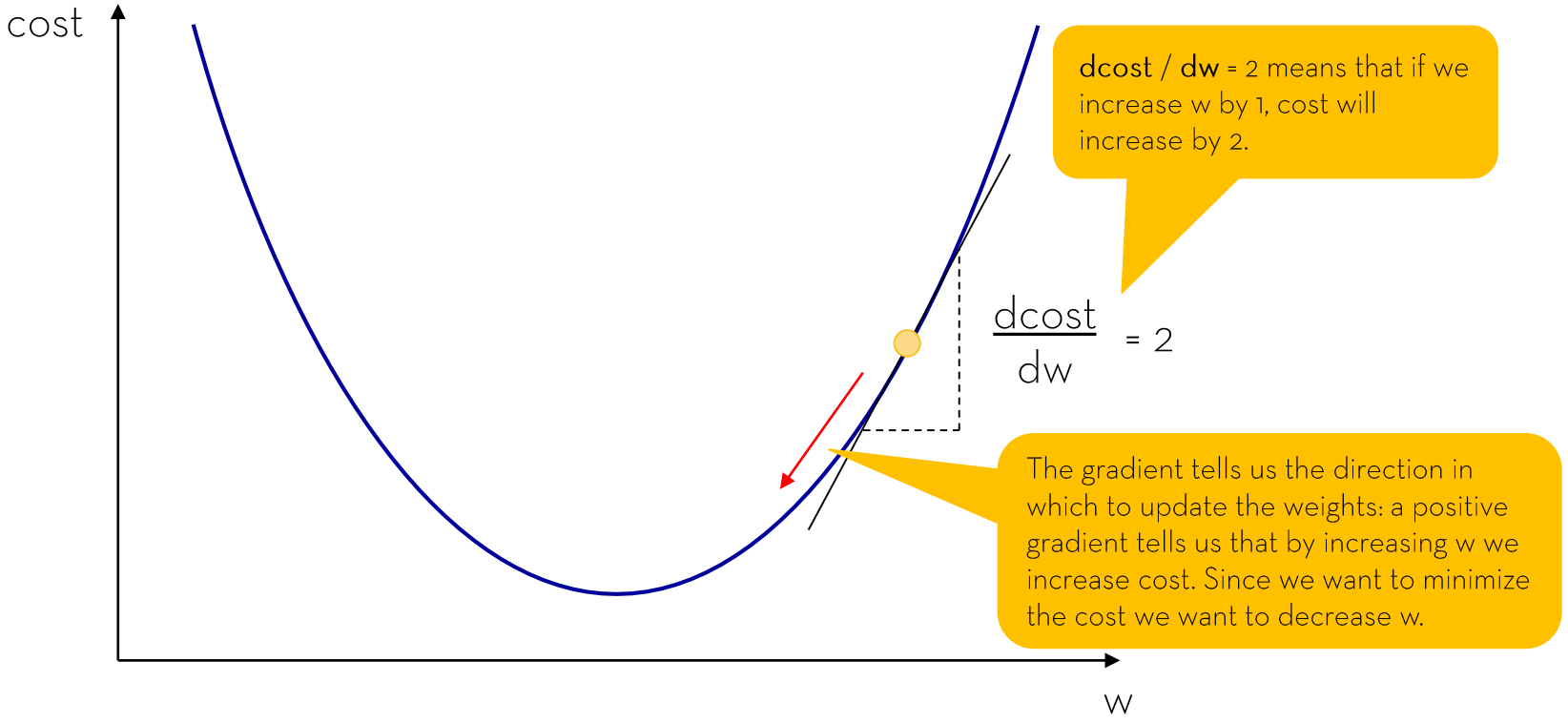
The gradient answers the question "how much does y change if we change x?"

$$\frac{dy}{dx}$$

$$f'(x)$$

In most of the lectures we will use function notation. If y = f(x) then dy/dx = f'(x).

A quick refresher on gradients

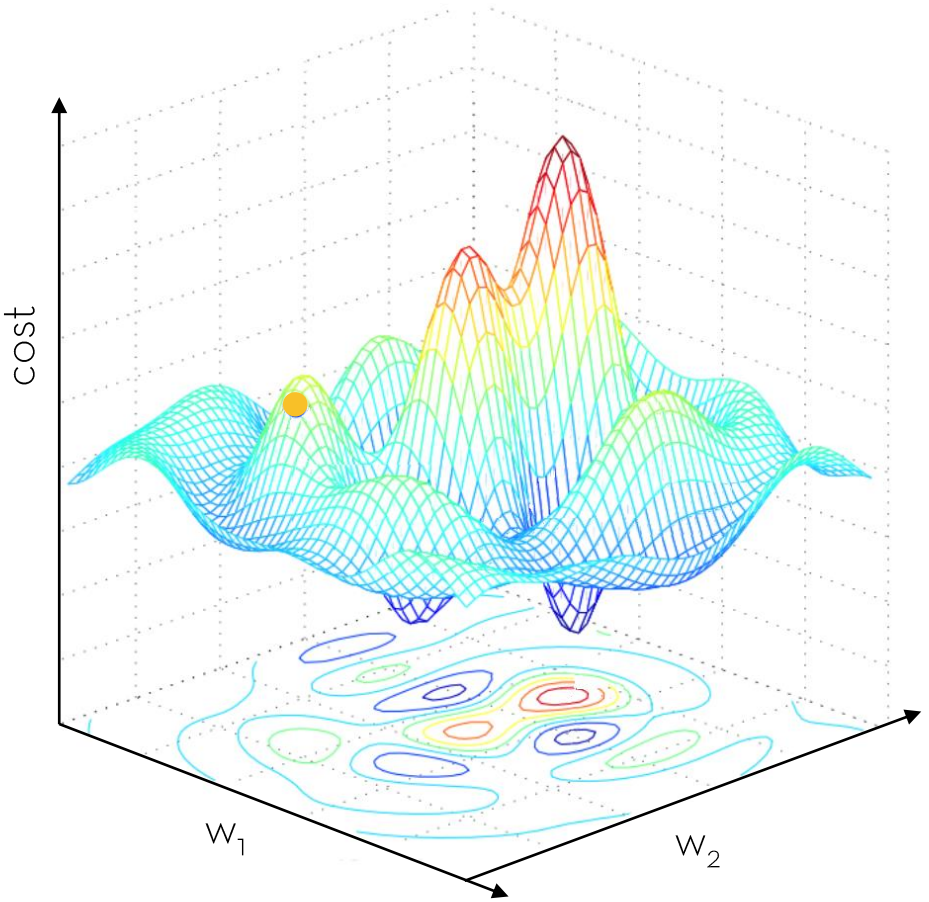


How to update the weights using gradients

Since we are optimizing many weights simultaneously, we take partial derivatives – i.e., the **derivative of the cost with respect to one weight** at a time.

$$\frac{\partial \text{cost}}{\partial w_1} \quad \frac{\partial \text{cost}}{\partial w_2}$$

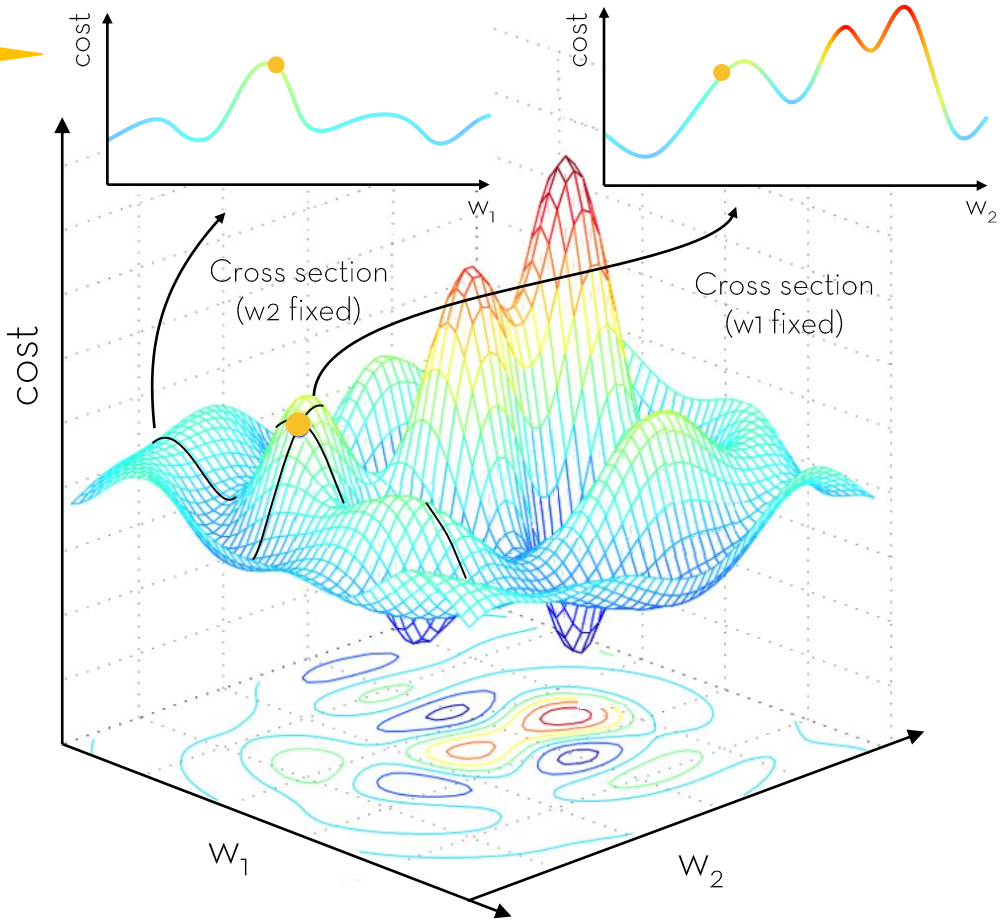
Then, once calculated we use $\partial \text{cost}/\partial w_1$ to update w1 and $\partial \text{cost}/\partial w_2$ to update w2.



How to update the weights using gradients

We can take the cross sections of the loss function **at the current point** to visualize the gradients better.

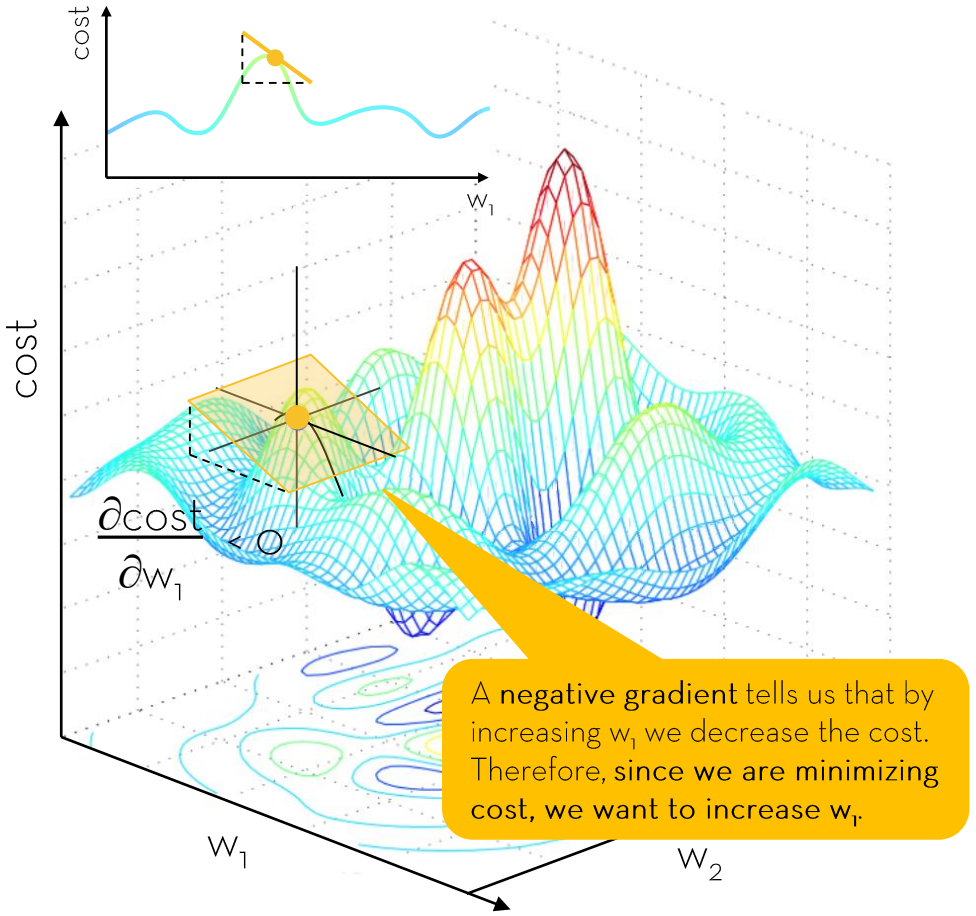
$$\frac{\partial \text{cost}}{\partial w_1} \quad \frac{\partial \text{cost}}{\partial w_2}$$



How to update the weights using gradients

- If $\partial cost/\partial w_1 > 0$, decreasing the w_1 will decrease the cost.
- If $\partial cost/\partial w_1 < 0$, increasing the w_1 will decrease the cost.

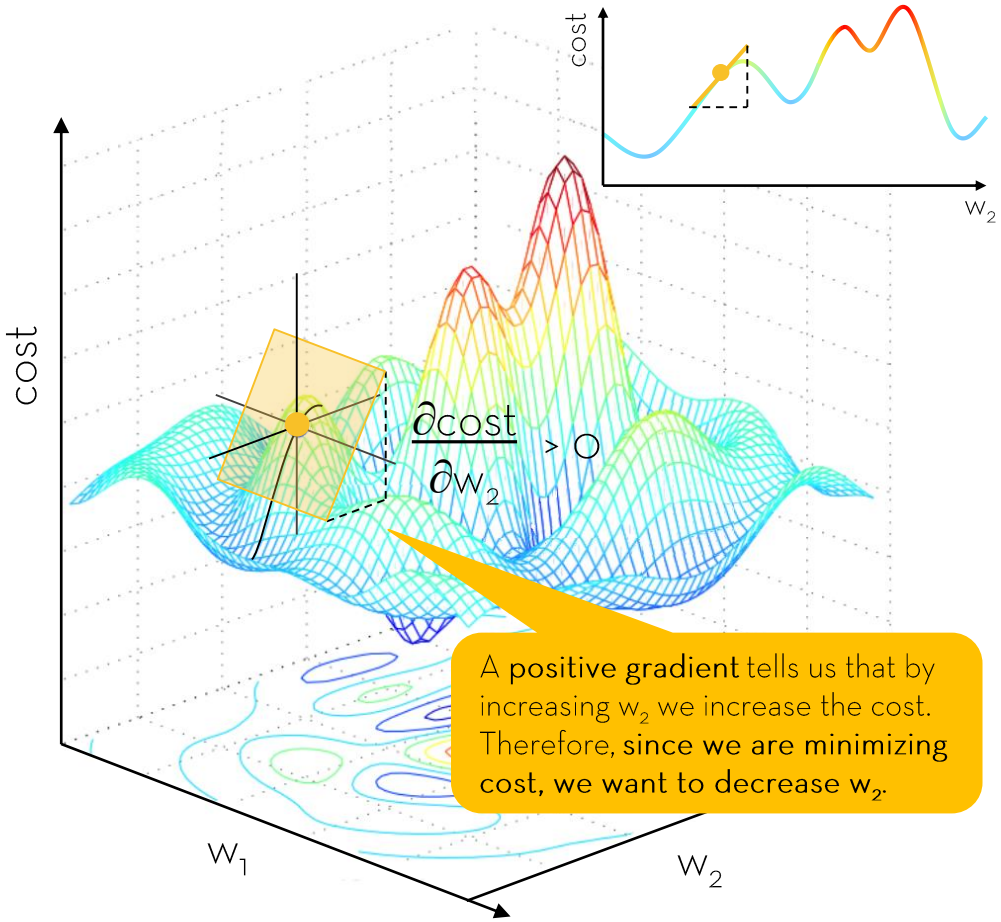
$$\frac{\partial cost}{\partial w_1} \quad \frac{\partial cost}{\partial w_2}$$



How to update the weights using gradients

- If $\partial cost/\partial w_2 > 0$, decreasing the w_2 will decrease the cost.
- If $\partial cost/\partial w_2 < 0$, increasing the w_2 will decrease the cost.

$$\frac{\partial cost}{\partial w_1} \quad \frac{\partial cost}{\partial w_2}$$

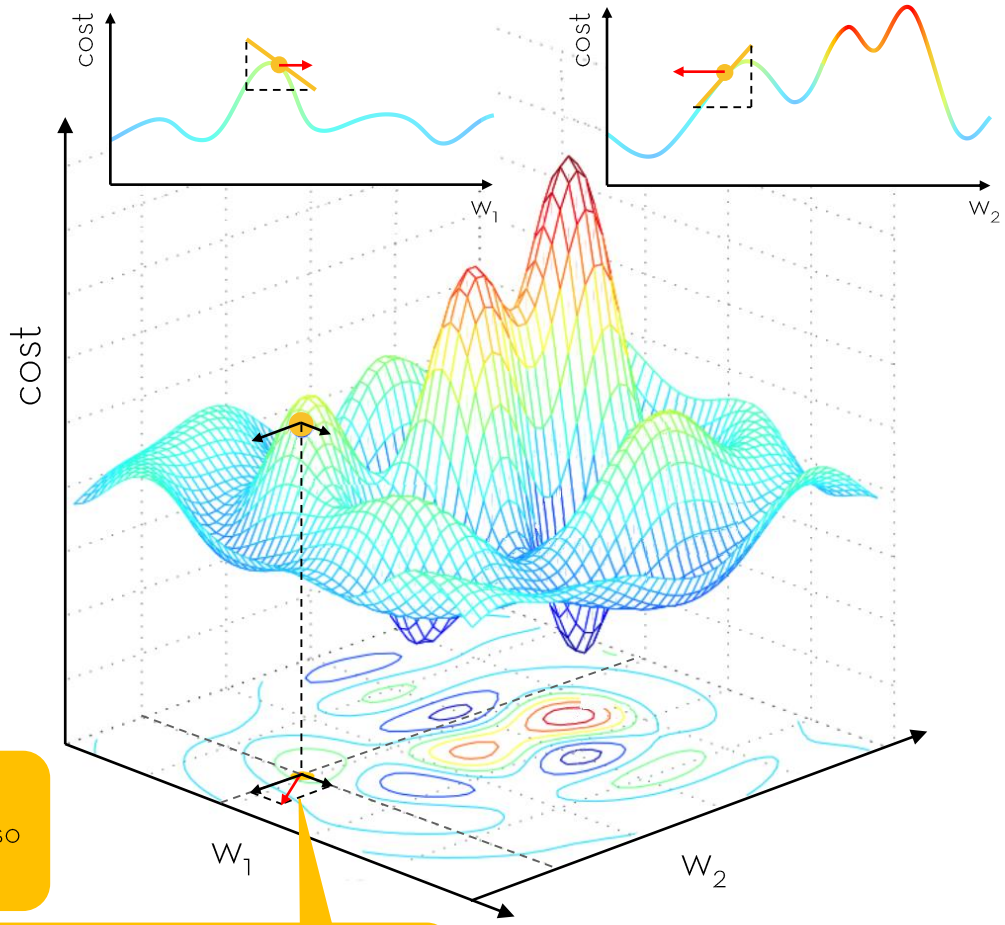


How to update the weights using gradients

Since $\partial cost/\partial w1$ is negative we want to increase w_1.

Since $\partial cost/\partial w2$ is positive we want to decrease w_2.

$$\frac{\partial cost}{\partial w_1} \quad \frac{\partial cost}{\partial w_2}$$

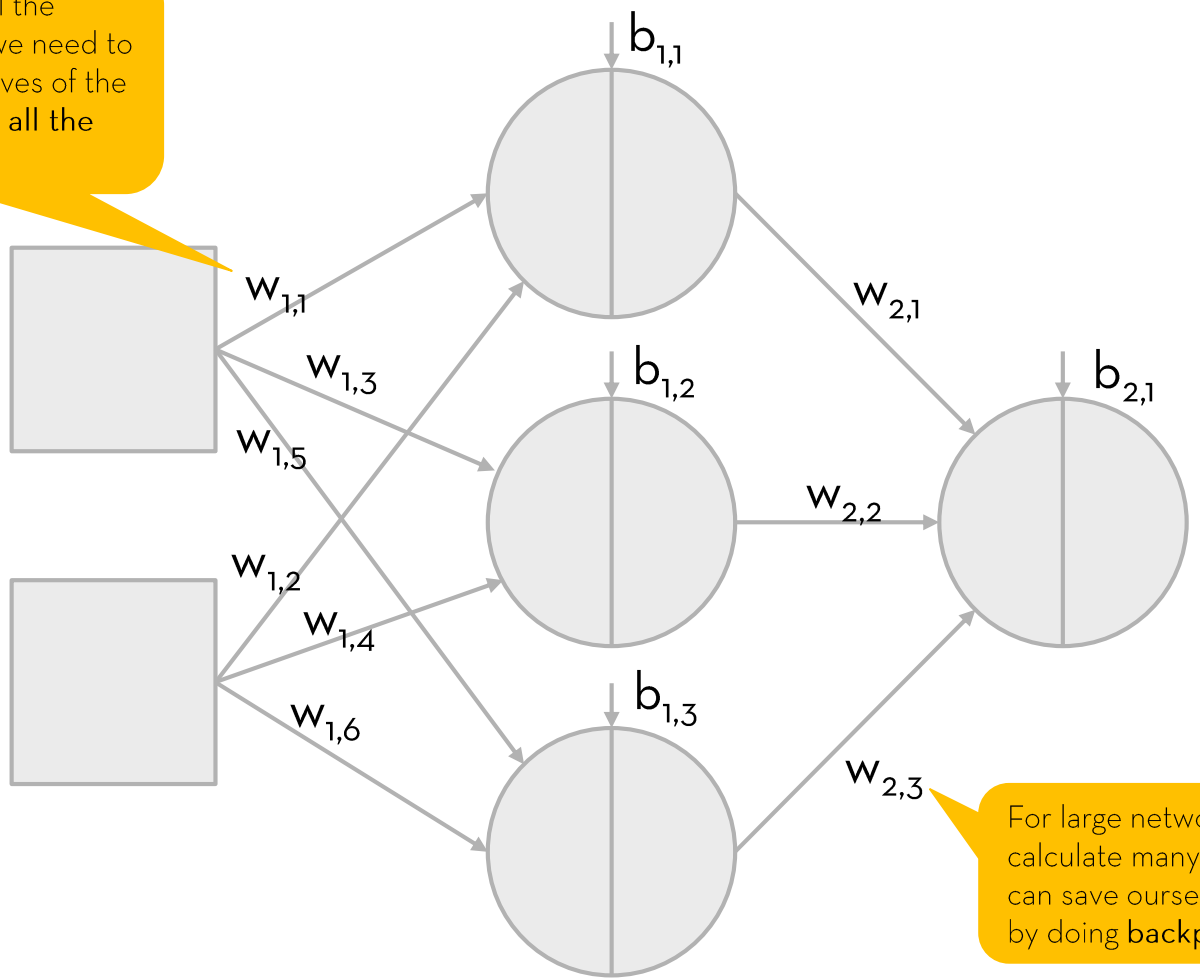


The gradient not only tells us which direction to travel but also **how far we need to travel.**

Since $\partial cost/\partial w2$ is steeper than $\partial cost/\partial w1$ we prefer to descend along w_2 since it will have the larger effect on the cost.

How to calculate the gradients

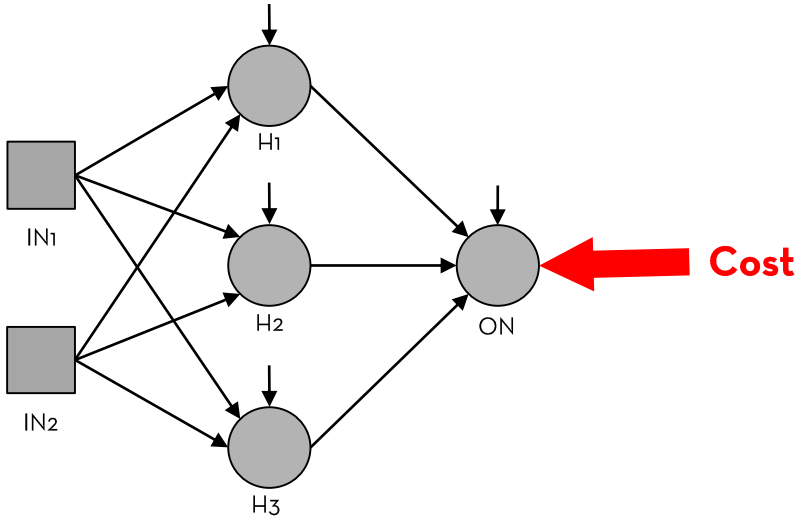
In order to update all the weights and biases, we need to calculate the derivatives of the cost **with respect to all the weights and biases.**



For large networks, we need to calculate many gradients! We can save ourselves some time by doing **backpropagation.**

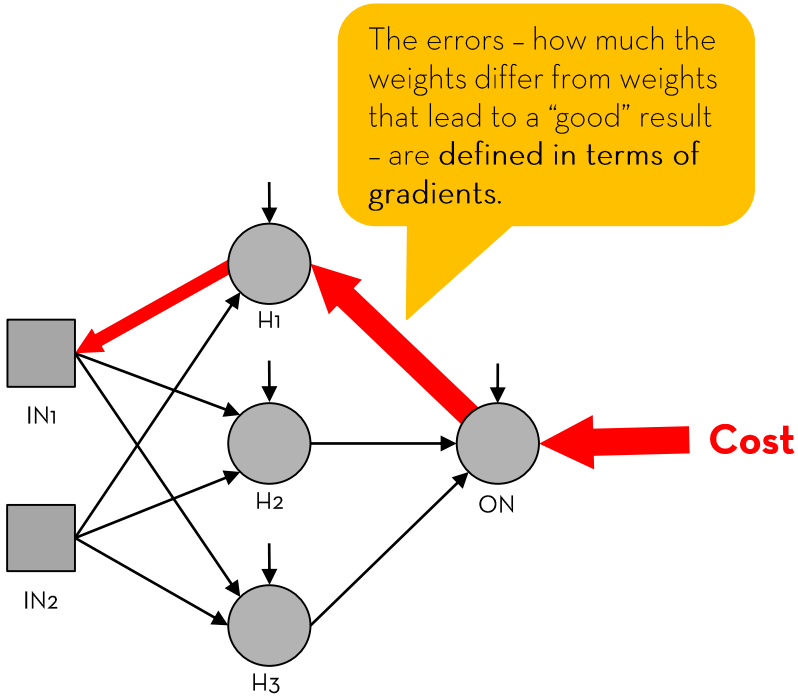
We use backpropagation to get the gradients of the network: Why back?

- We start at the **end of the network** – i.e., at the back.
- Then, we send the gradients backwards through the network.



We use backpropagation to get the gradients of the network: Why propagation?

- To calculate the gradient of earlier layers we **propagate the gradients from later layers backwards** through the network.
- Backpropagation saves us from having to recalculate the full gradient from scratch at each step.
- Formally, backpropagation is just the repeated application of the **chain rule**.



A quick refresher on the chain rule

A gradient can be decomposed using the **chain rule**.

We rephrase the question from
- How much does y change if we change x?

to
1) How much does y change if we change z? and
2) How much does z change if we change x?

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z}\frac{\partial z}{\partial x}$$

If y = f(x) and z = g(x).

$$= f'(z)\, g'(x)$$

See a primer in backpropagation for a more in-depth explanation.

We "pass the gradients backwards" by calculating the gradient of the cost with respect to earlier values

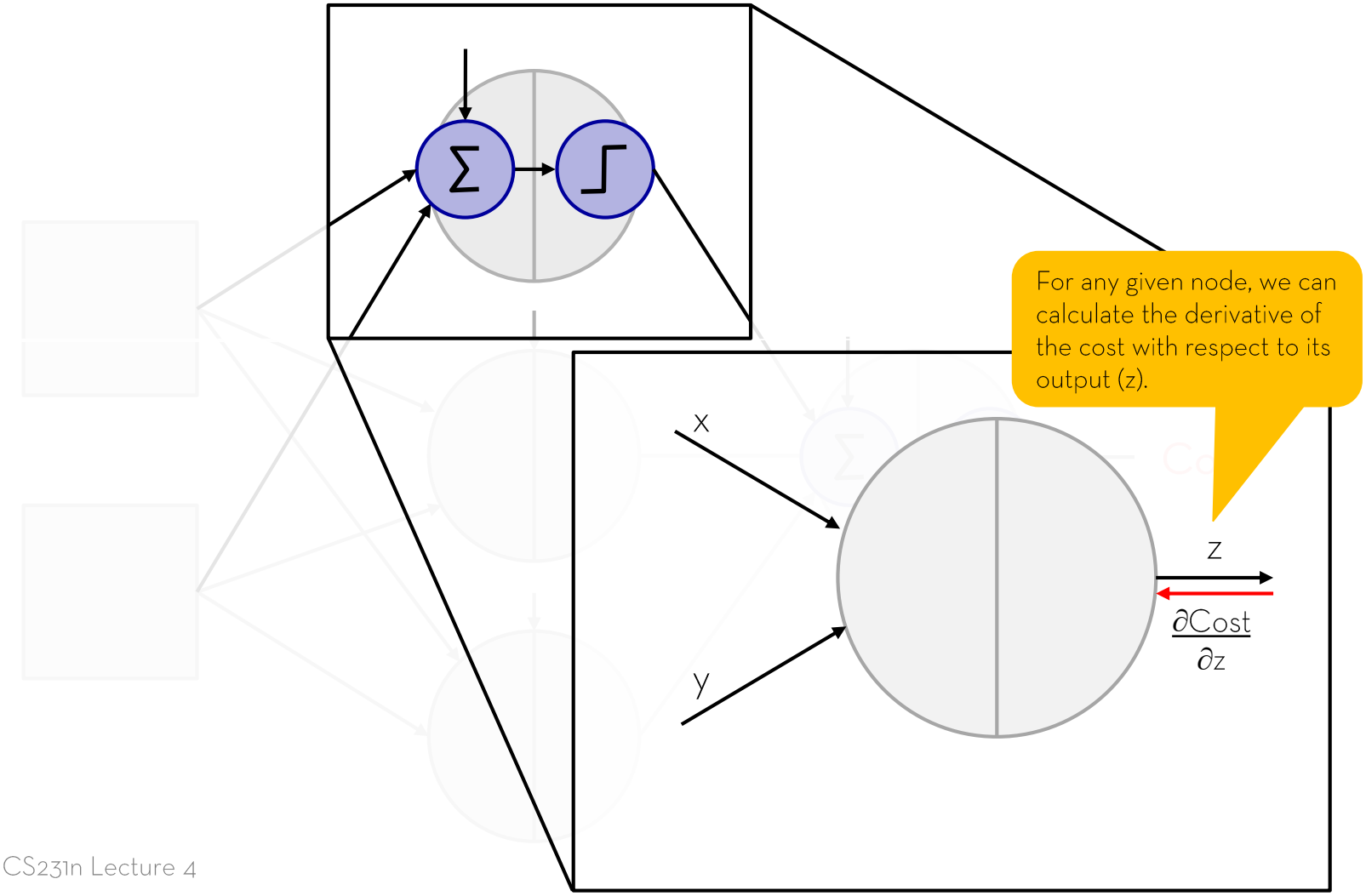


Figure adapted from CS231n Lecture 4

We "pass the gradients backwards" by calculating the gradient of the cost with respect to earlier values

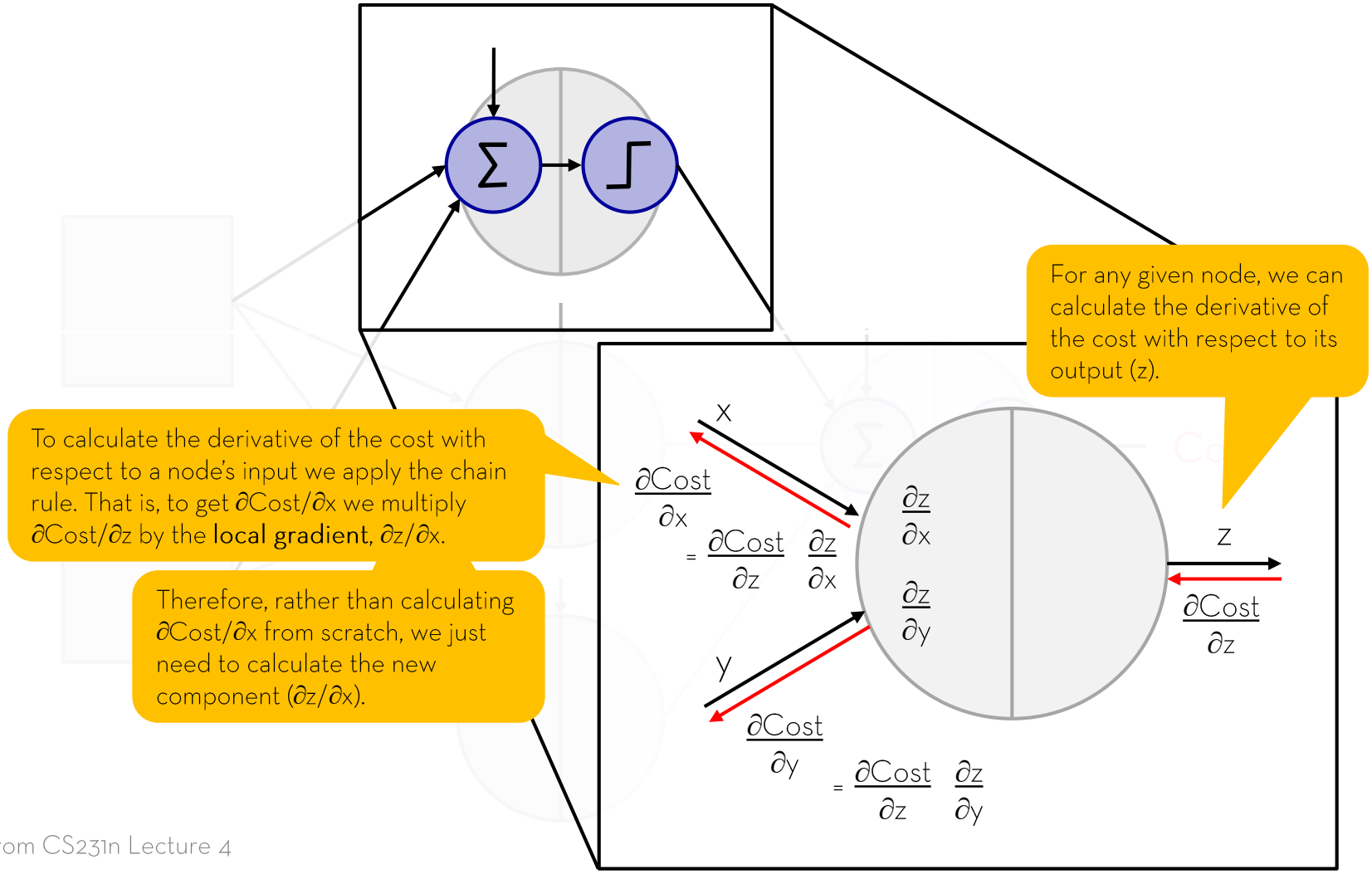
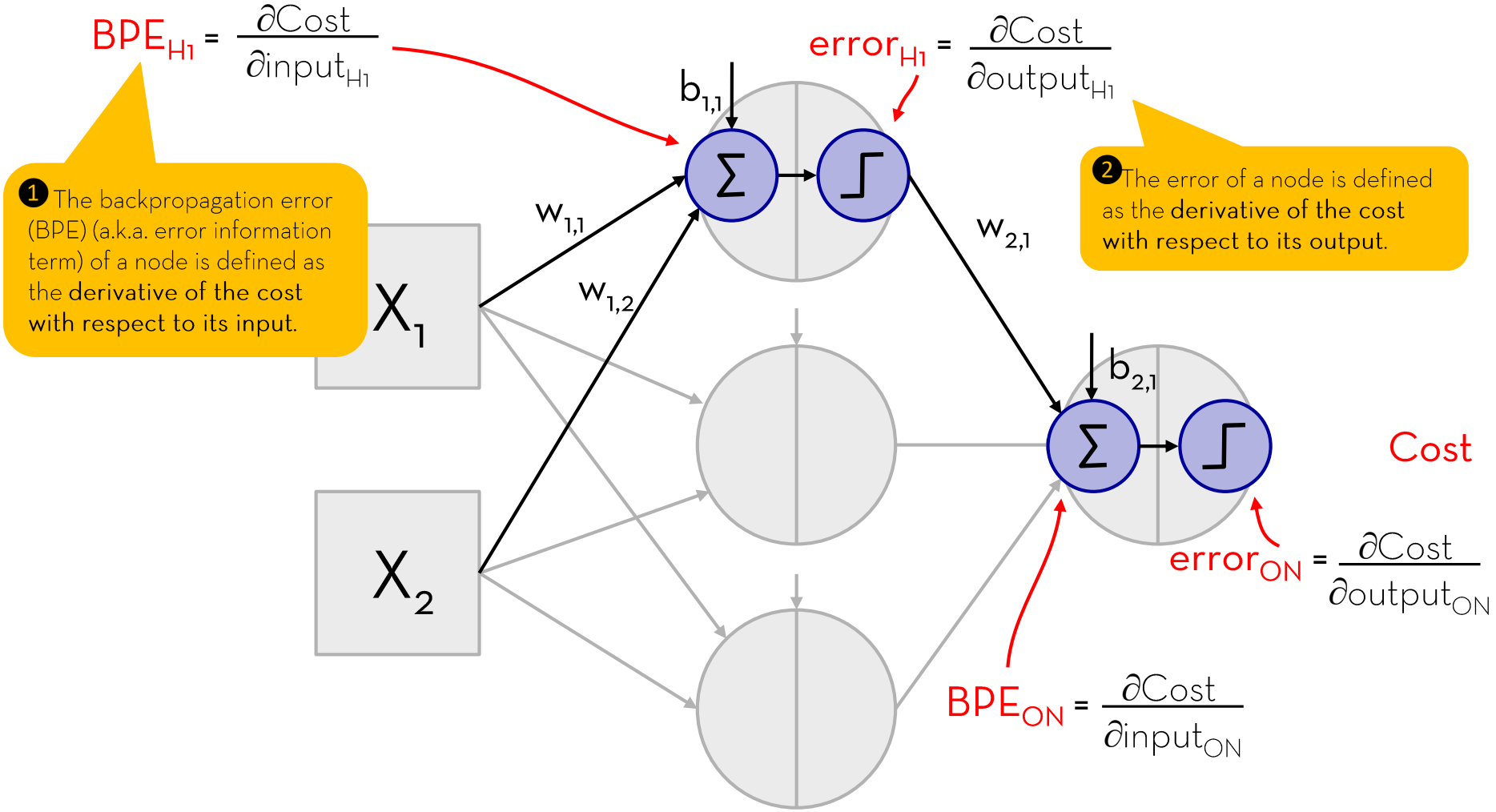
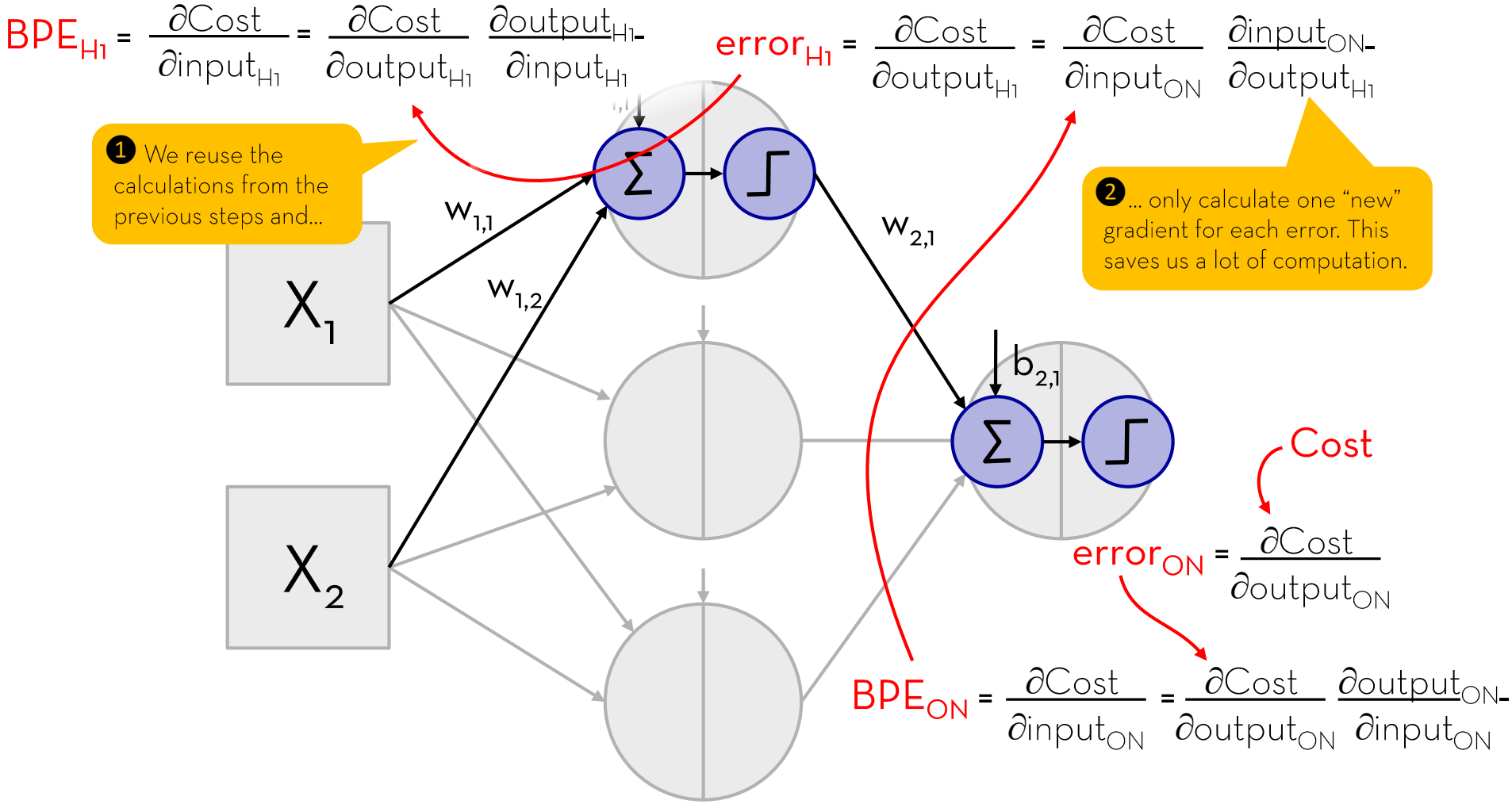


Figure adapted from CS231n Lecture 4

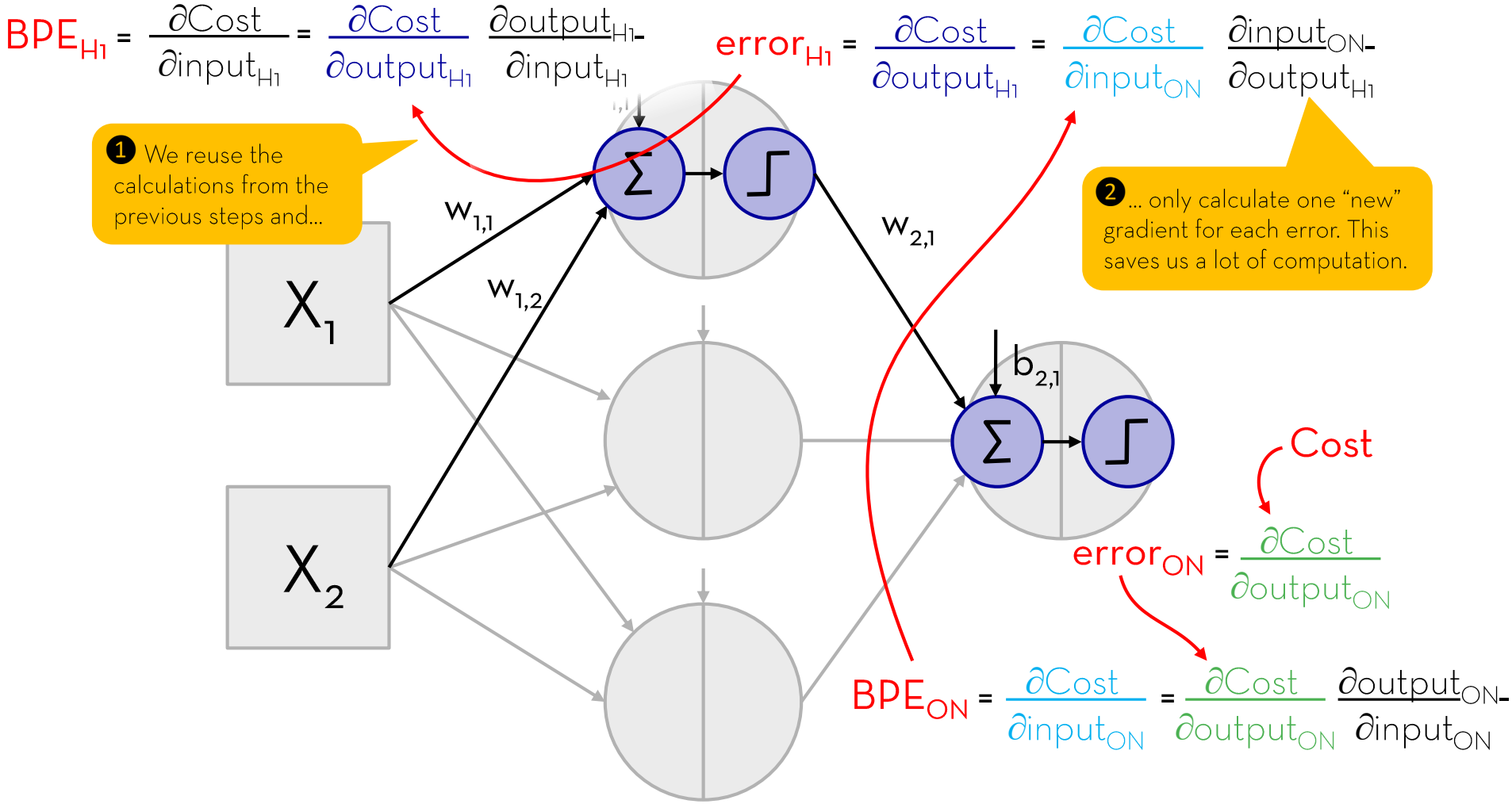
We can define the gradients in terms of errors



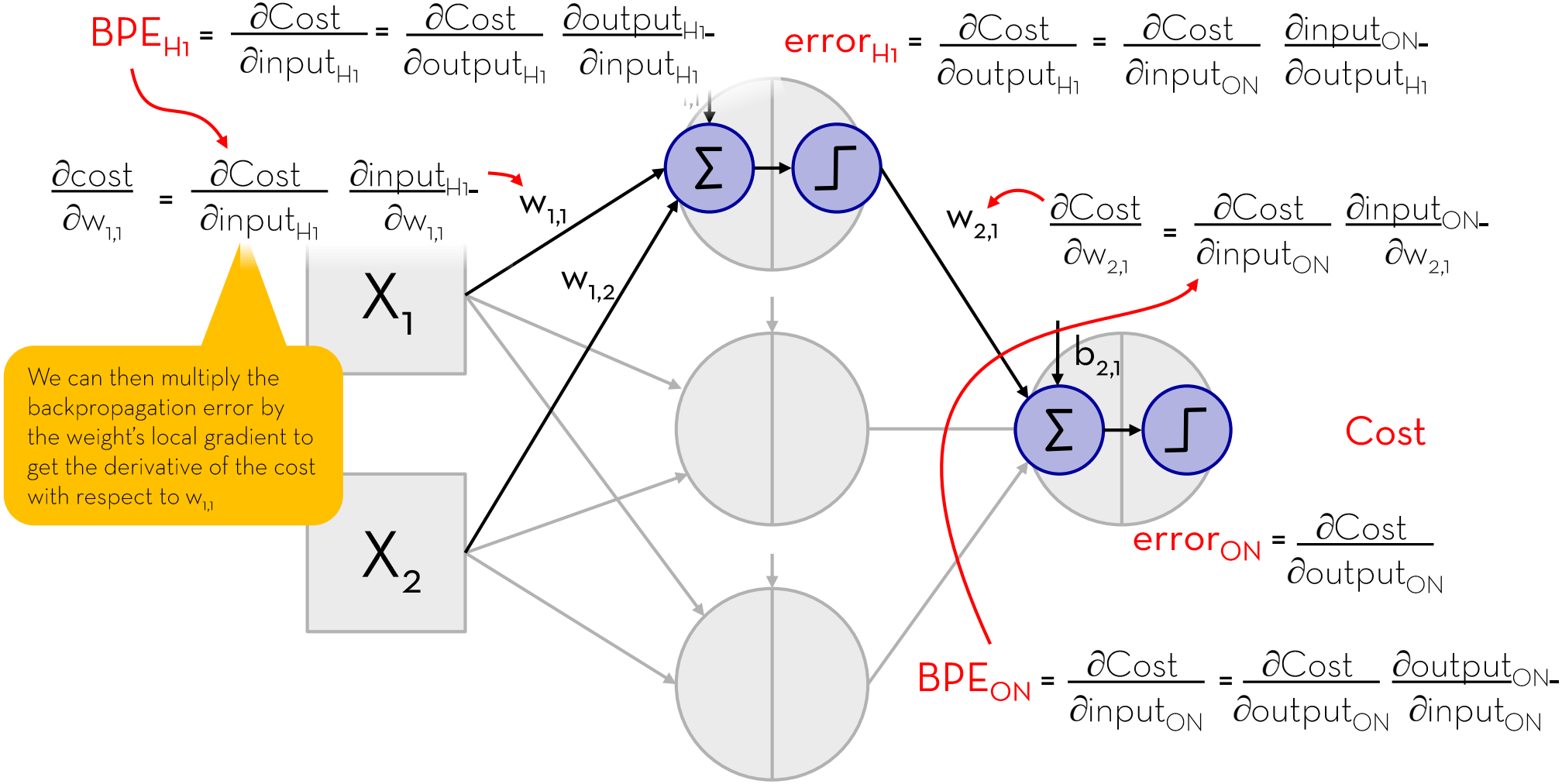
We can calculate all the important gradients using the chain rule



We can calculate all the important gradients using the chain rule



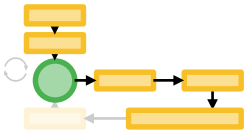
We can calculate all the important gradients using the chain rule



Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. **Backpropagation.**
6. Update weights.

Repeat 3. – 6. until convergence criteria is met.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

Step 5: Backpropagation
How to calculate the gradients?

a) Output layer:

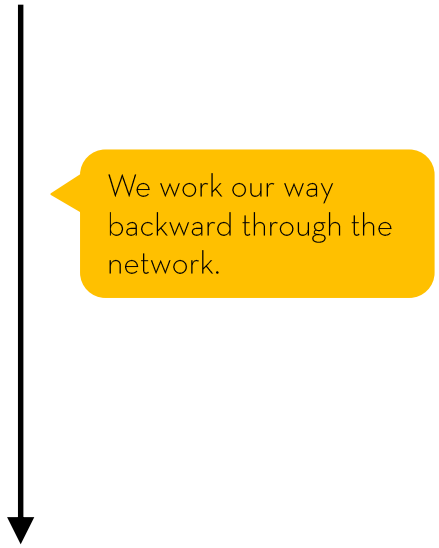
   1. Calculate error.

   2. Calculate the backpropagation error.

b) Hidden layer:

   1. Calculate error.

   2. Calculate the backpropagation error.



Step 5: Backpropagation
How to calculate the gradients?

a) Output layer:

   1. Calculate error.

   2. Calculate the backpropagation error.

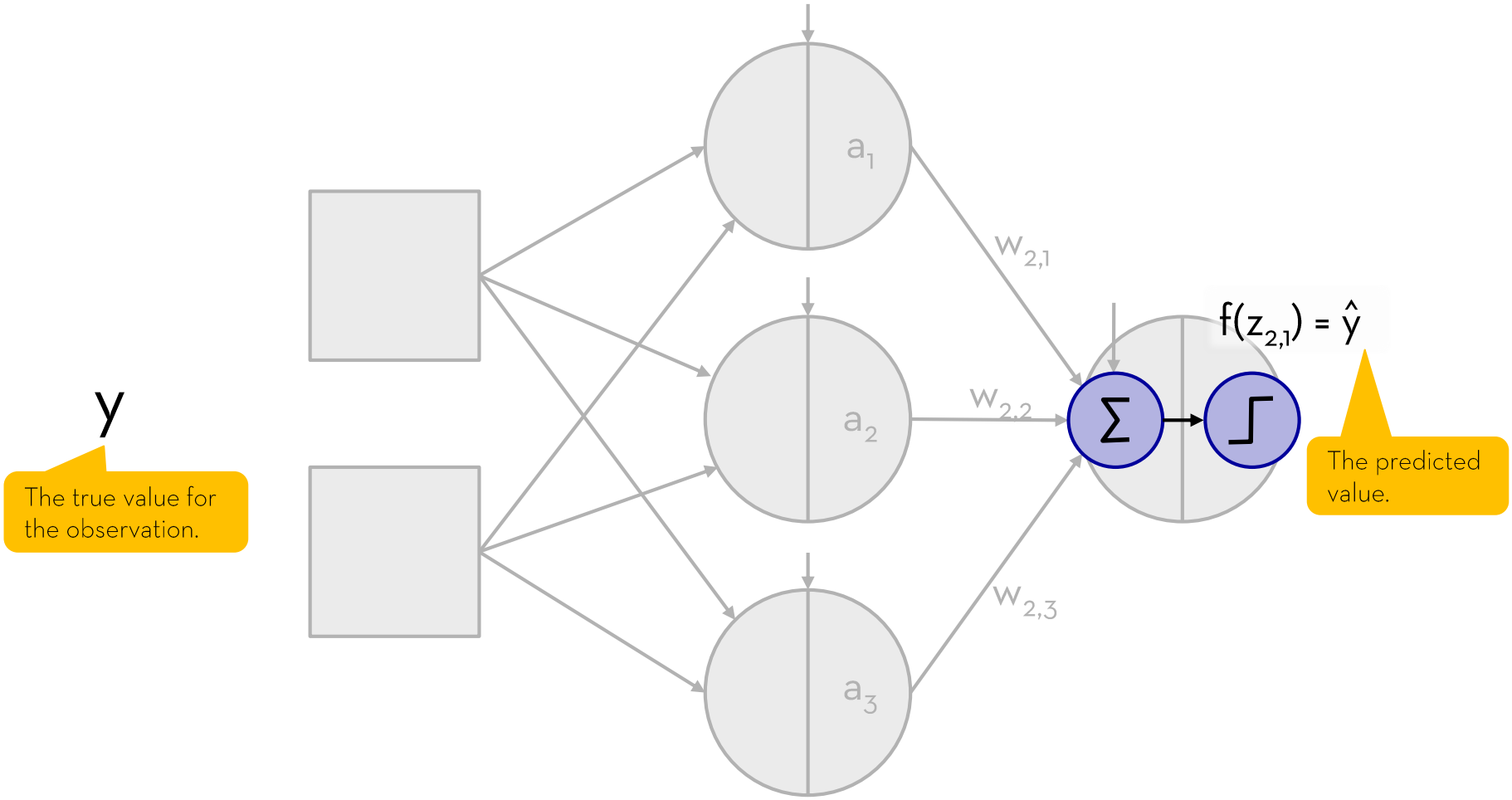
b) Hidden layer:

   1. Calculate error.

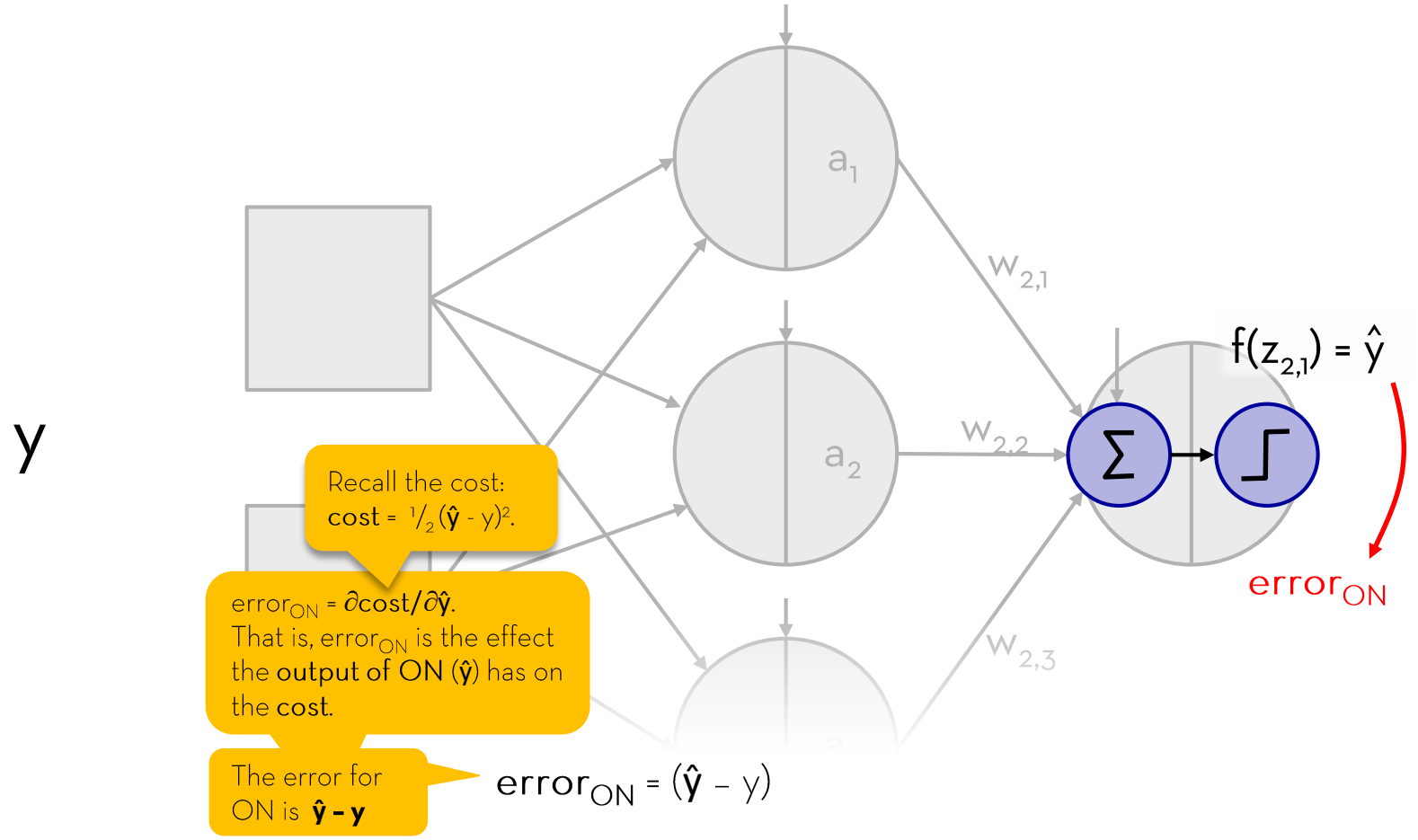
   2. Calculate the backpropagation error.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

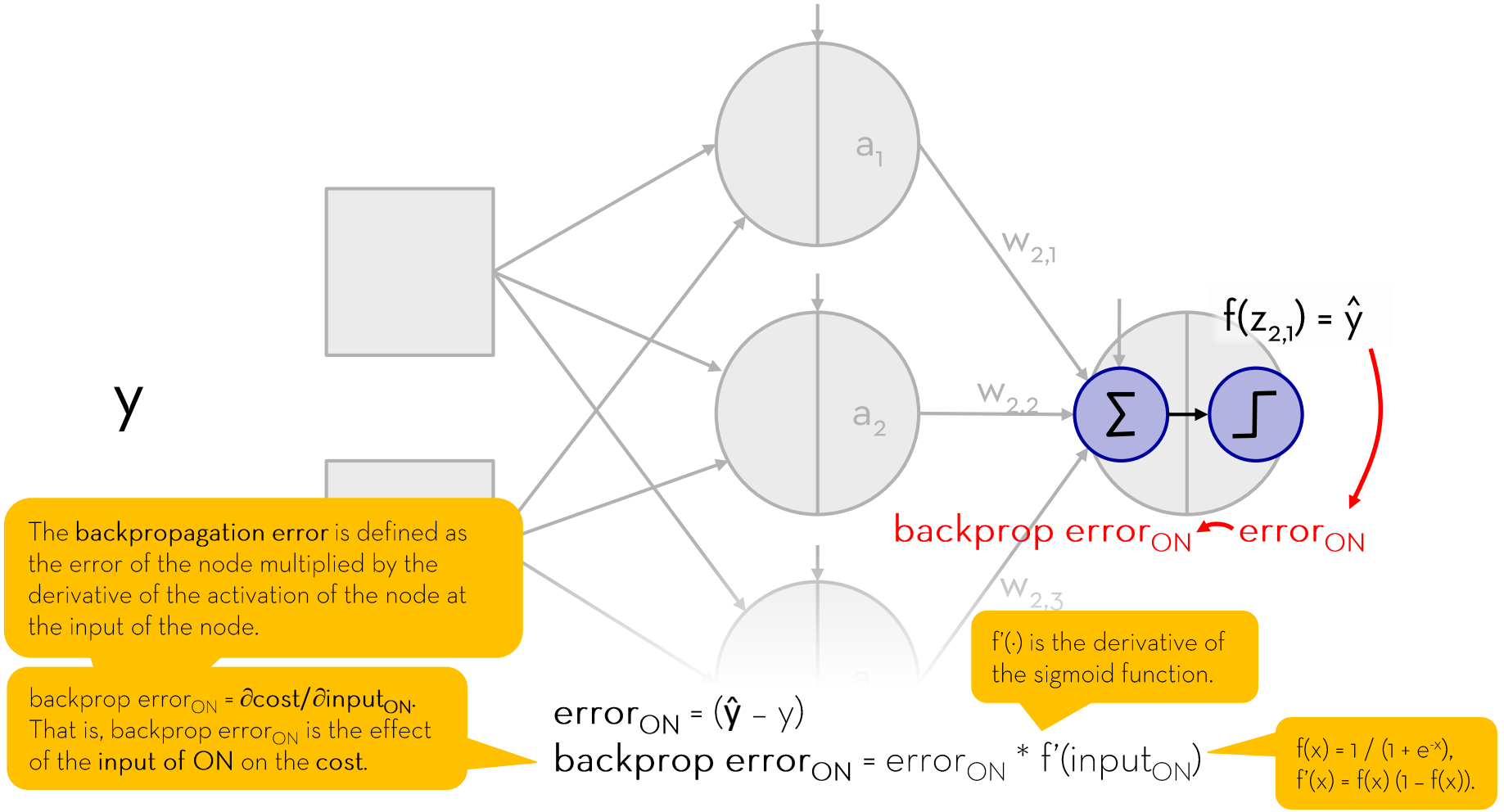
Step 5: Backpropagation
a1) Calculate the error for the output layer (1/2)



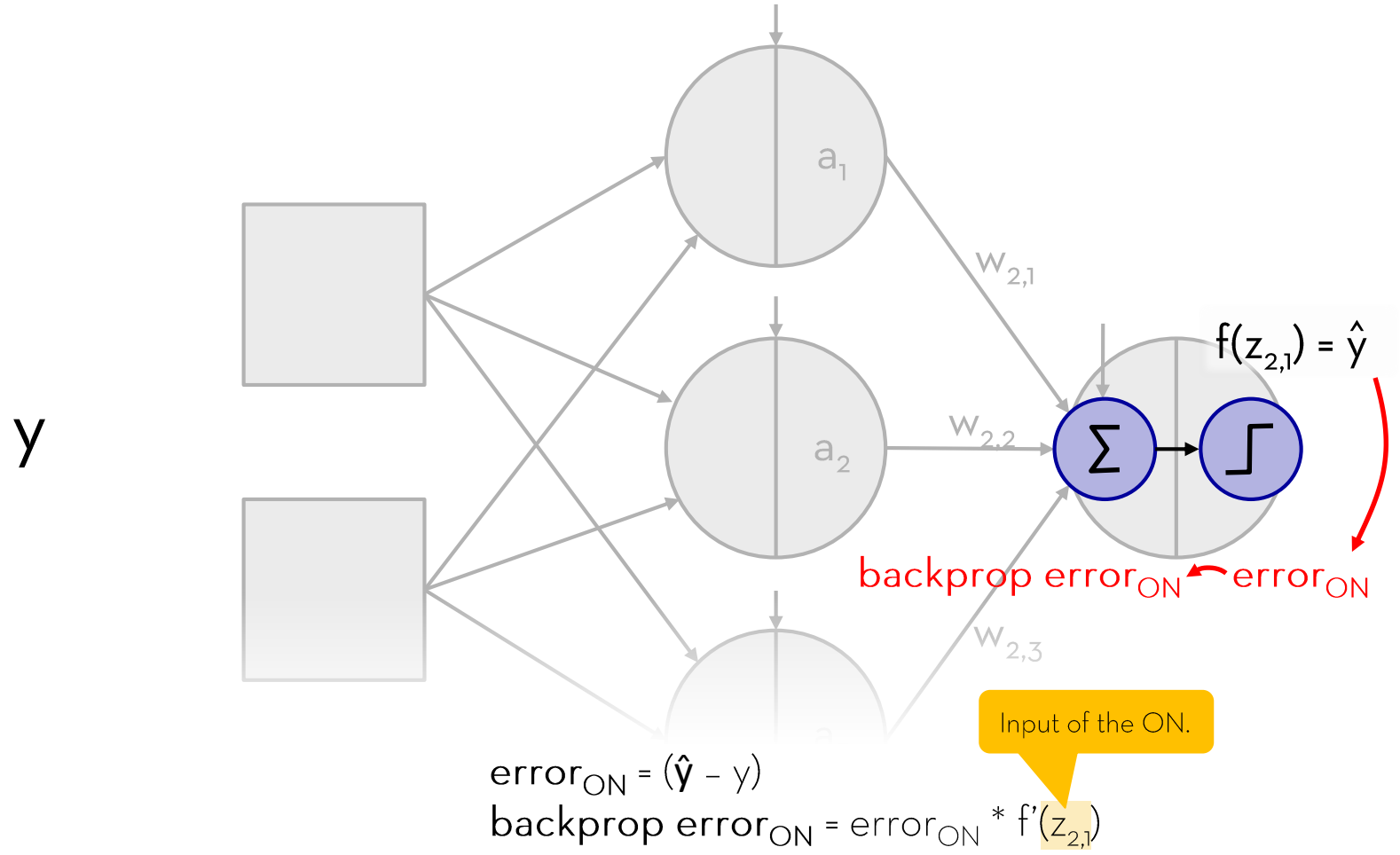
Step 5: Backpropagation
a1) Calculate the error for the output layer (2/2)



Step 5: Backpropagation
a2) Calculate the backpropagation error for ON (1/2)



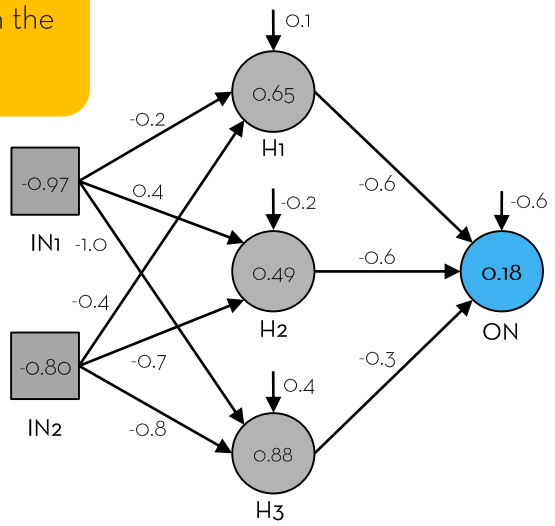
Step 5: Backpropagation
a2) Calculate the backpropagation error for ON (2/2)



Step 5: Backpropagation
a1) Calculate the error for output node (1/2)

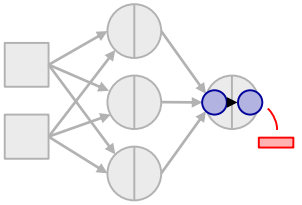
Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

The **error** is the difference between the estimated and true value: $\hat{y} - y$.



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

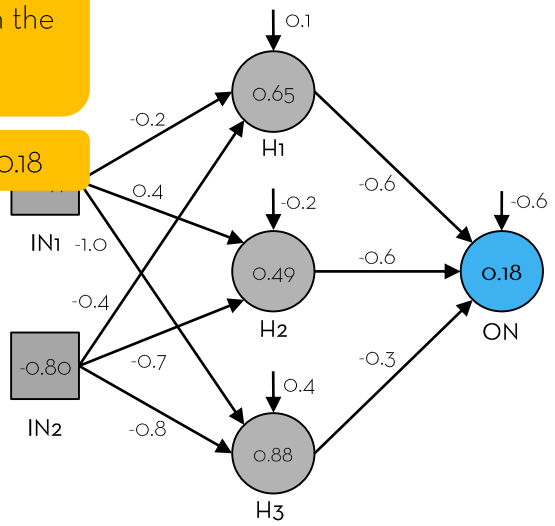


Step 5: Backpropagation
a1) Calculate the error for output node (2/2)

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

The **error** is the difference between the estimated and true value: $\hat{y} - y$.

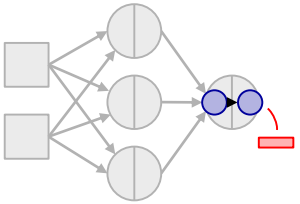
$error_{ON} = 0.18 - 0 = 0.18$



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error				0.18

$error_{ON} = \partial cost / \partial \hat{y}$.



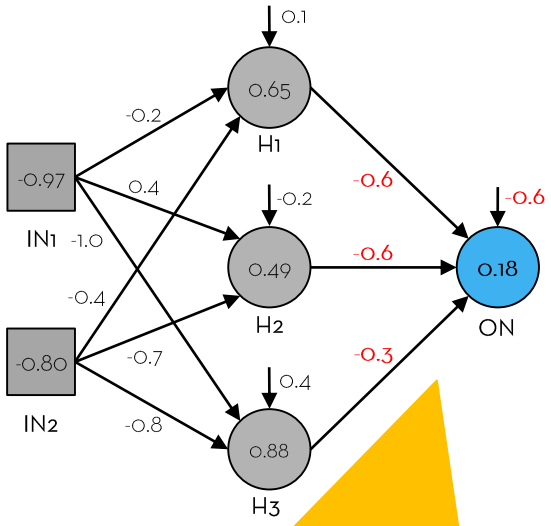
Step 5: Backpropagation
a2) Calculate the backpropagation error for ON

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error				0.18
Backprop. Err				0.025

backprop $error_{ON}$ = $\partial cost/\partial input_{ON}$.



The **backpropagation error for ON**:

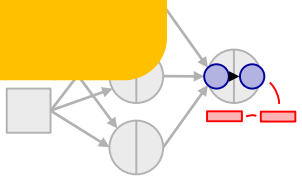
backprop $error_{ON}$

= $error_{ON}$ * f'($a_1w_1 + a_2w_2 + a_3w_3 + b$)

= $error_{ON}$ * f'(-0.6*0.65 + -0.6*0.49 + -0.3*0.88 + -0.6)

= 0.18 * f'(-1.55)

= 0.025



Step 5: Backpropagation
How to calculate the gradients?

a) Output layer:

   1. Calculate error.

   2. Calculate the backpropagation error.

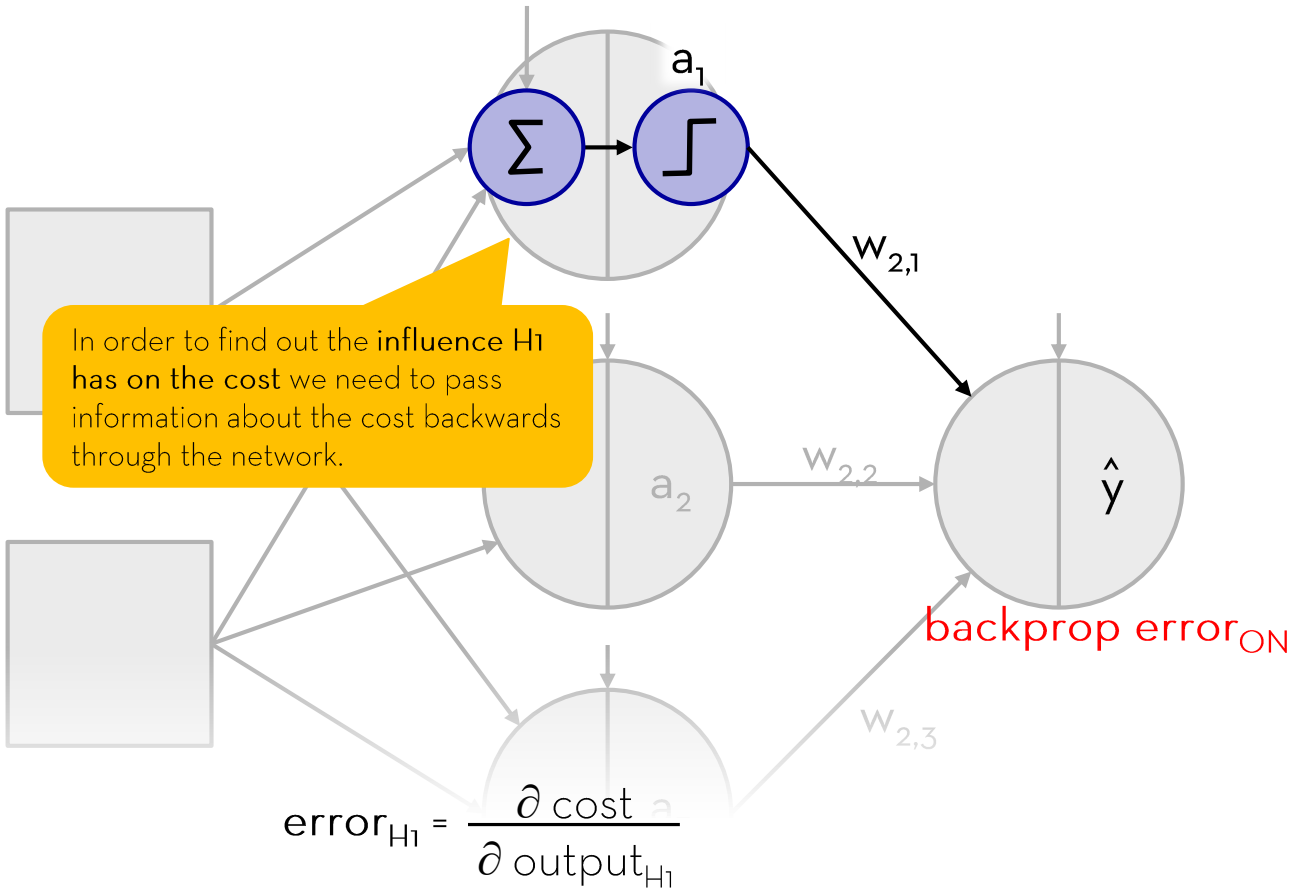
b) Hidden layer:

   1. Calculate error.

   2. Calculate the backpropagation error.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

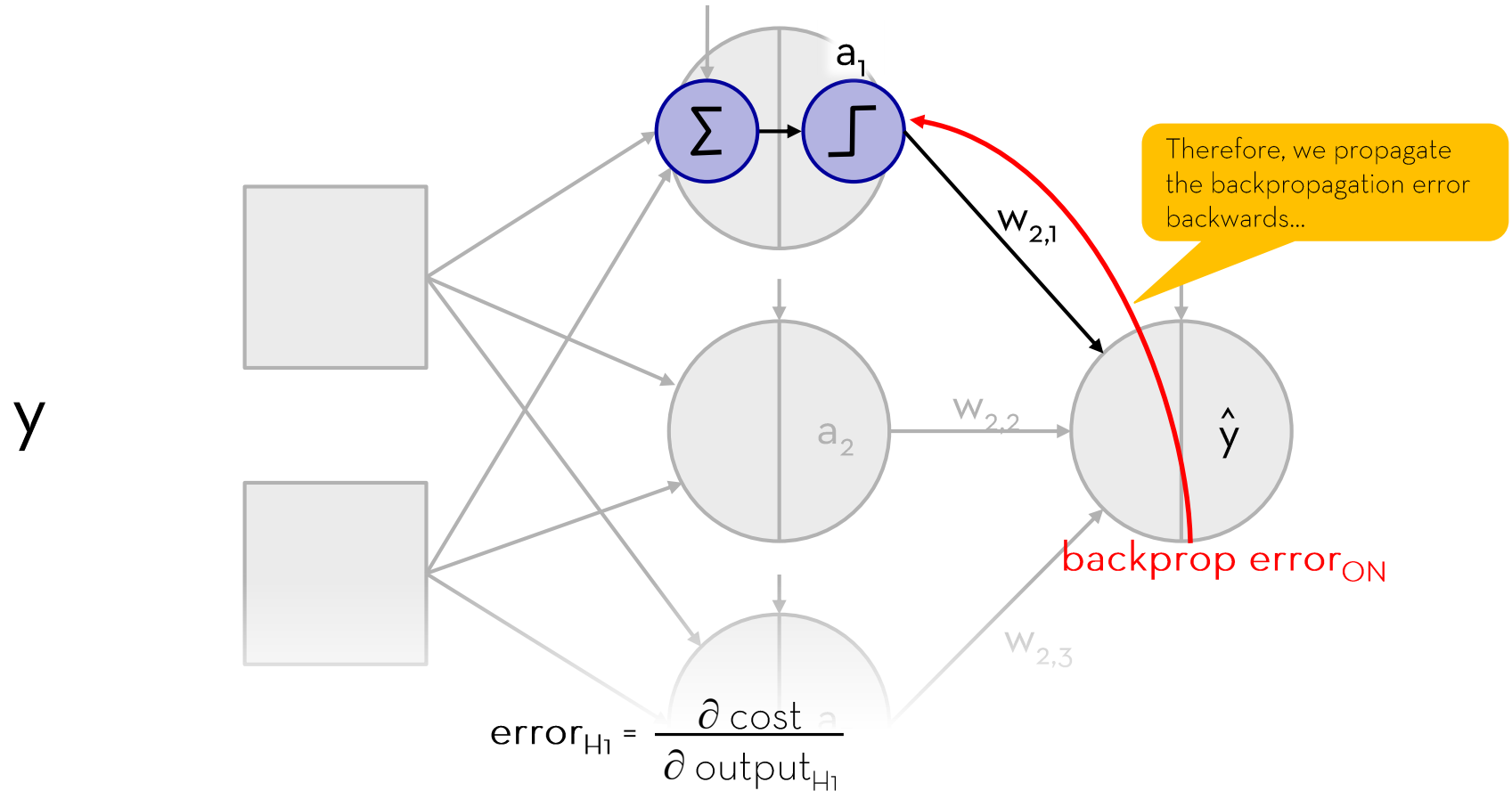
Step 5: Backpropagation
b1) Calculate the error for the hidden layer (H1,1) (1/4)



y

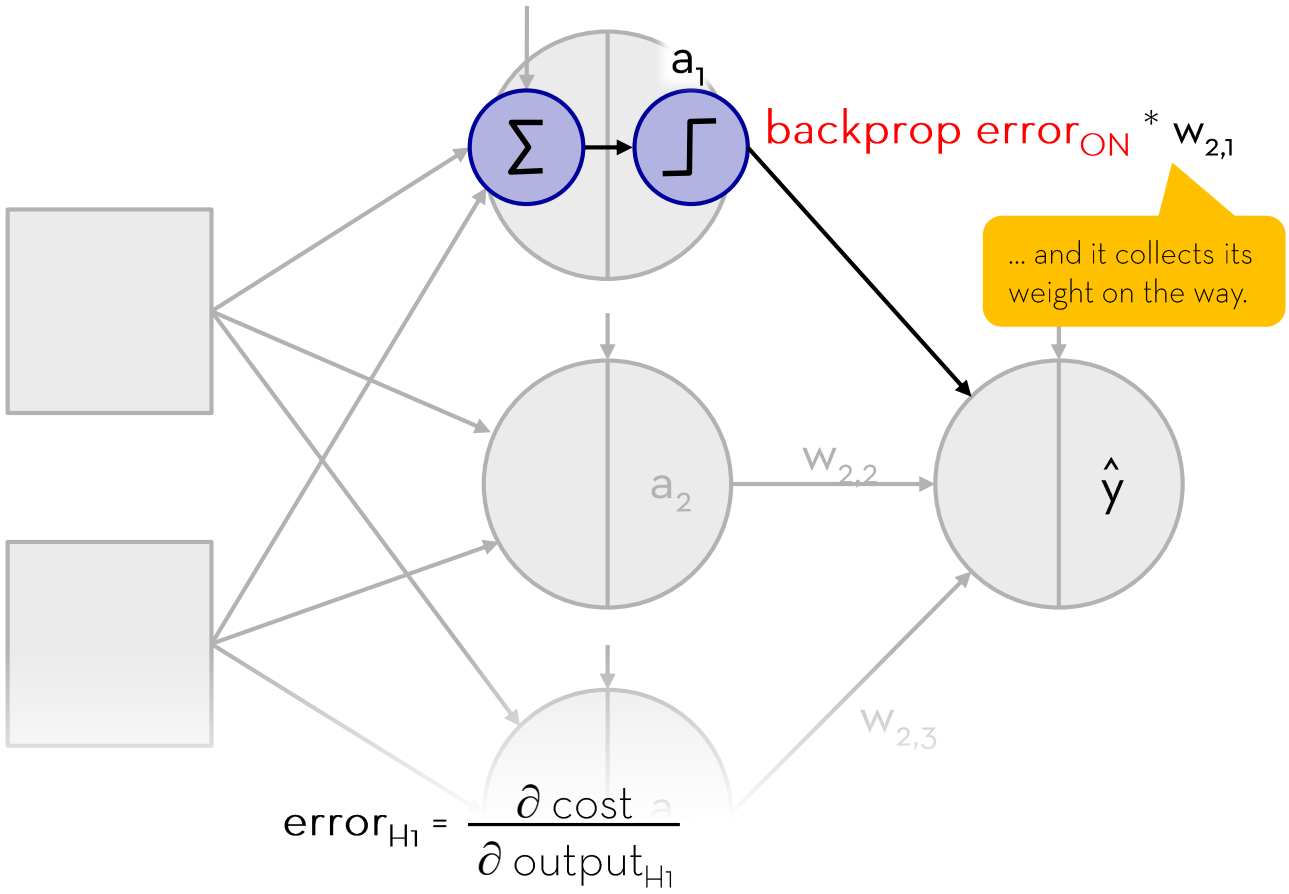
Label | Input layer | Hidden layer | Output layer

Step 5: Backpropagation
b1) Calculate the error for the hidden layer (H1,1) (2/4)



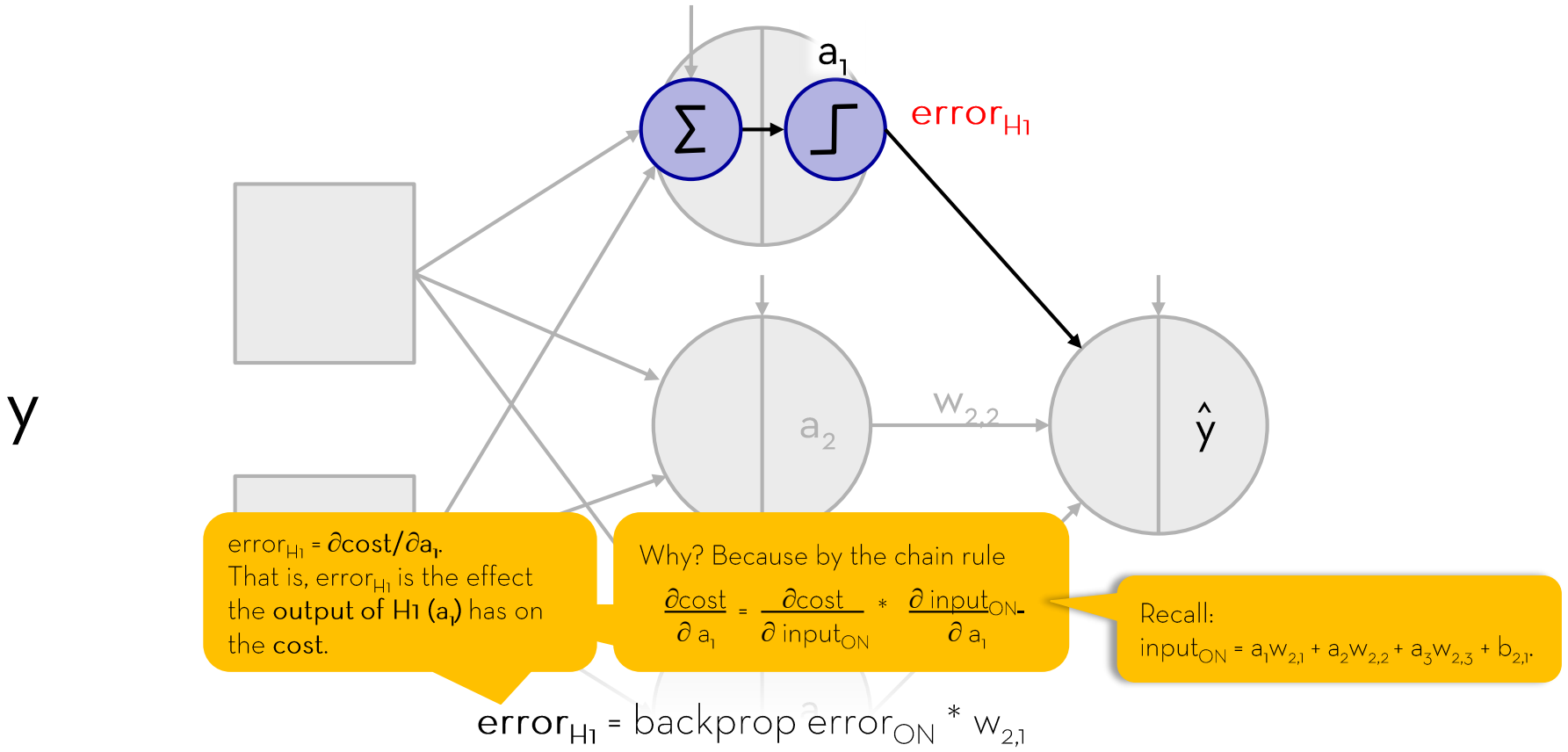
Label Input layer Hidden layer Output layer

Step 5: Backpropagation
b1) Calculate the error for the hidden layer (H1,1) (3/4)



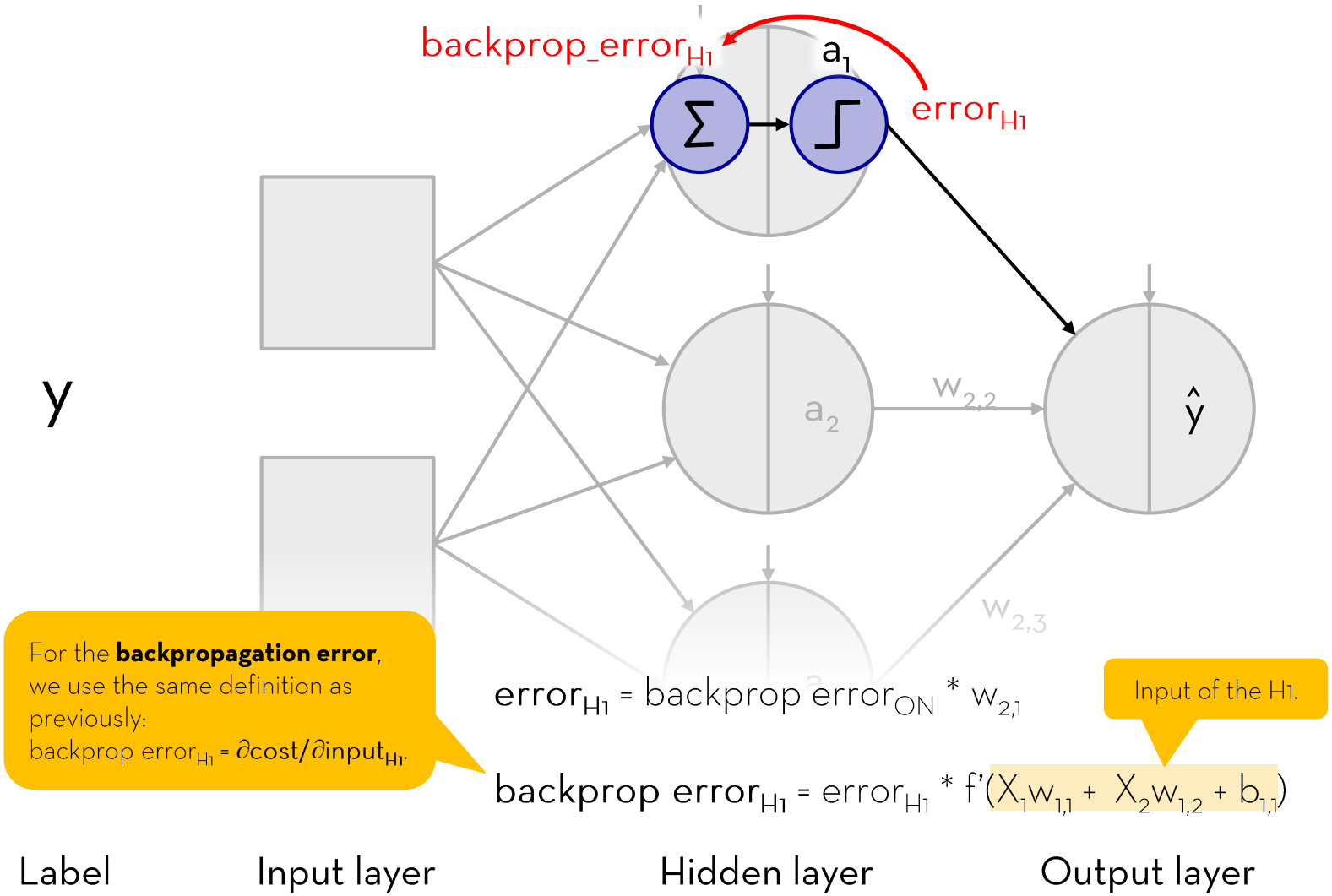
Label Input layer Hidden layer Output layer

Step 5: Backpropagation
b1) Calculate the error for the hidden layer (H1,1) (4/4)



See a primer in backpropagation for a more in-depth explanation.

Step 5: Backpropagation
b2) Calculate the backpropagation error for the HL



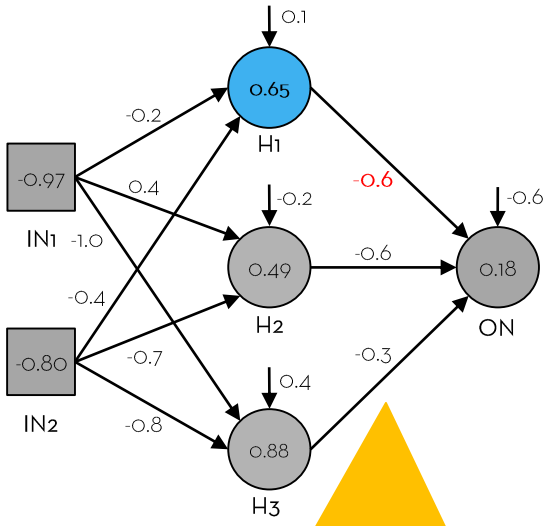
Step 5: Backpropagation
b1) Calculate error for the HL nodes (1/3)

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015			0.18
Backprop. Err				0.025

$error_{H1} = \partial cost / \partial output_{H1}$.

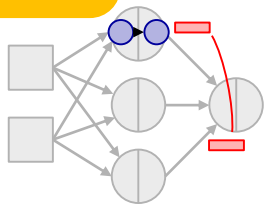


Compute the error:

$error_{H1}$ = backprop $error_{ON}$ * $w_{H1 \to ON}$.

= 0.025 * -0.6

= -0.015.



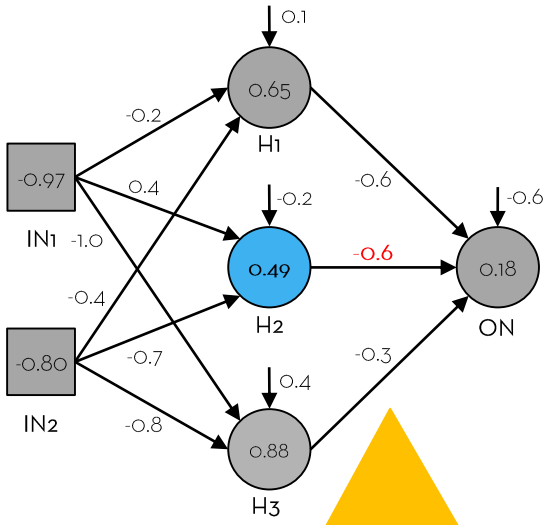
Step 5: Backpropagation
b1) Calculate error for the HL nodes (2/3)

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

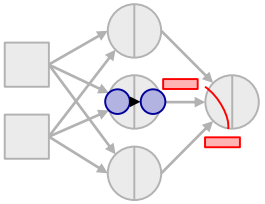
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015		0.18
Backprop. Err				0.025

$error_{H_2} = \partial cost / \partial output_{H_2}$.



$error_{H_2}$ = backprop $error_{ON}$ * $w_{H_2 \to ON}$.
= 0.025 * -0.6
= -0.015.



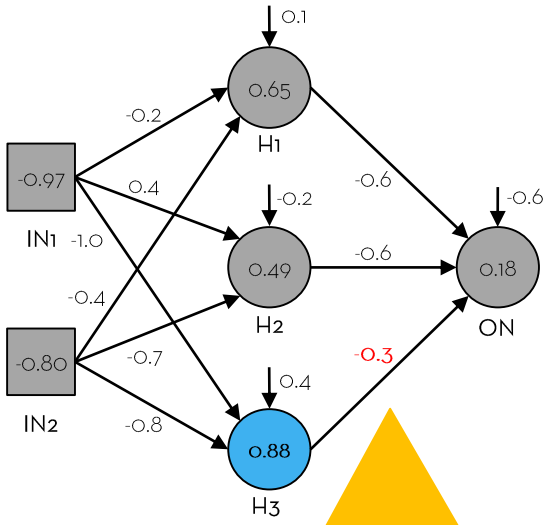
Step 5: Backpropagation
b1) Calculate error for the HL nodes (3/3)

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

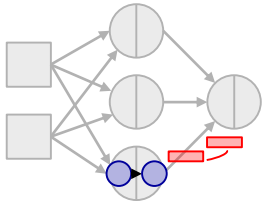
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err				0.025

$error_{H3} = \partial cost/\partial output_{H3}$.



$error_{H3}$ = backprop $error_{ON}$ * $w_{H3 \rightarrow ON}$.
= 0.025 * -0.3
= -0.008

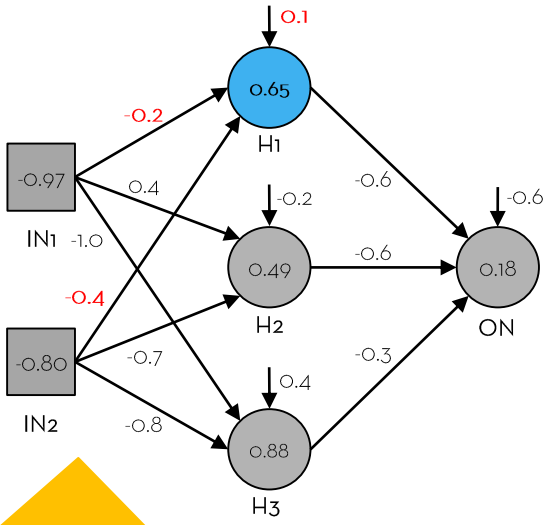


Step 5: Backpropagation
b2) Calculate backpropagation error for HL nodes

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

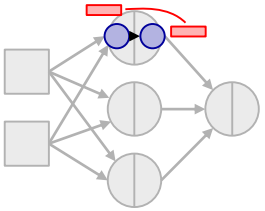
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003			0.025



Then, calculate the **backpropagation error**:

backprop $error_{H1}$
= $error_{H1}$ * f'($input_{H1}$)
= $error_{H1}$ * f'(-0.97*-0.2 + -0.80*-0.4 + 0.1)
= -0.015 * f'(0.614)
= -0.003

backprop $error_{H1}$ = $\partial cost/\partial input_{H1}$

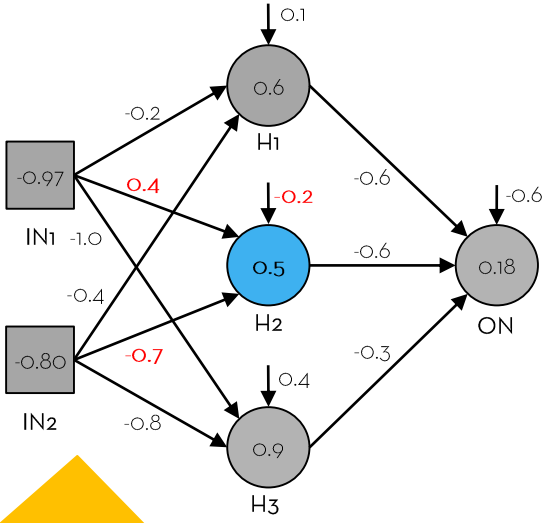


Step 5: Backpropagation
b2) Calculate backpropagation error for HL nodes

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

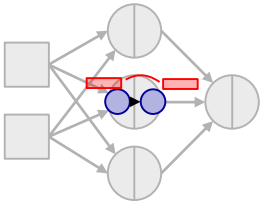
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004		0.025



Then, calculate the **backpropagation error**:

backprop $error_{H_2}$
= $error_{H_2}$ * f'($input_{H_2}$)
= $error_{H_2}$ * f'(-0.97*0.4 + -0.80*-0.7 + -0.2)
= -0.015 * f'(-0.026)
= -0.004

backprop $error_{H_2}$ = $\partial cost / \partial input_{H_2}$.

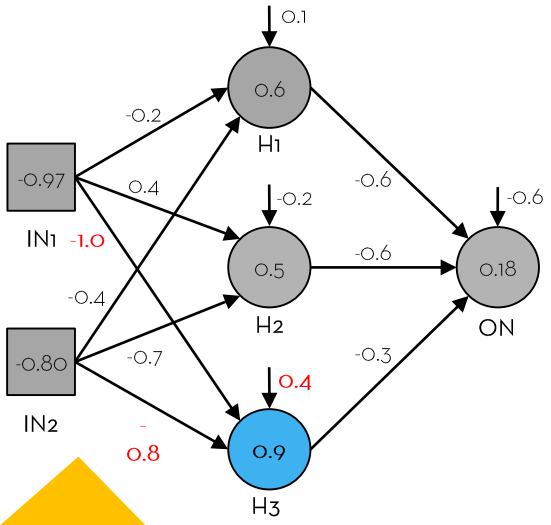


Step 5: Backpropagation
b2) Calculate backpropagation error for HL nodes

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

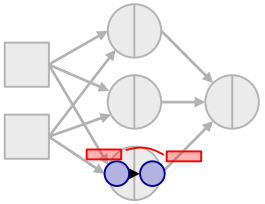
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025



Then, calculate the **backpropagation error**:

backprop $error_{H3}$
= $error_{H3}$ * f'($input_{H3}$)
= $error_{H3}$ * f'(-0.97*-1.0 + -0.80*-0.7 + 0.4)
= -0.008 * f'(2.01)
= -0.001

backprop $error_{H3}$ = $\partial cost/\partial input_{H3}$.



Step 5: Backpropagation
Repeat steps and a) and b) for all observations

Y	X1	X2	H1	H2	H3	Output	Cost	Err H1	Err H2	Err H3	Err ON	BPE H1	BPE H2	BPE H3	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.015	-0.015	-0.008	0.18	-0.003	-0.004	-0.001	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03								
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03								
...	...	...	...	...	...	...	...								
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67								
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05								
							Σ = 1.55								

So far, we have calculated the errors and backpropagation errors for the output and hidden layer for the first observation.

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

Step 5: Backpropagation
Repeat steps and a) and b) for all observations

Y	X1	X2	H1	H2	H3	Output	Cost	Err H1	Err H2	Err H3	Err ON	BPE H1	BPE H2	BPE H3	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.015	-0.015	-0.008	0.18	-0.003	-0.004	-0.001	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.014	-0.014	-0.007	0.17	-0.003	-0.004	-0.001	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.014	-0.014	-0.007	0.17	-0.003	-0.004	-0.001	0.023
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.073	0.073	0.037	-0.82	0.017	0.018	0.005	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.023	-0.023	-0.012	0.22	-0.006	-0.006	-0.002	0.039
							Σ = 1.55								

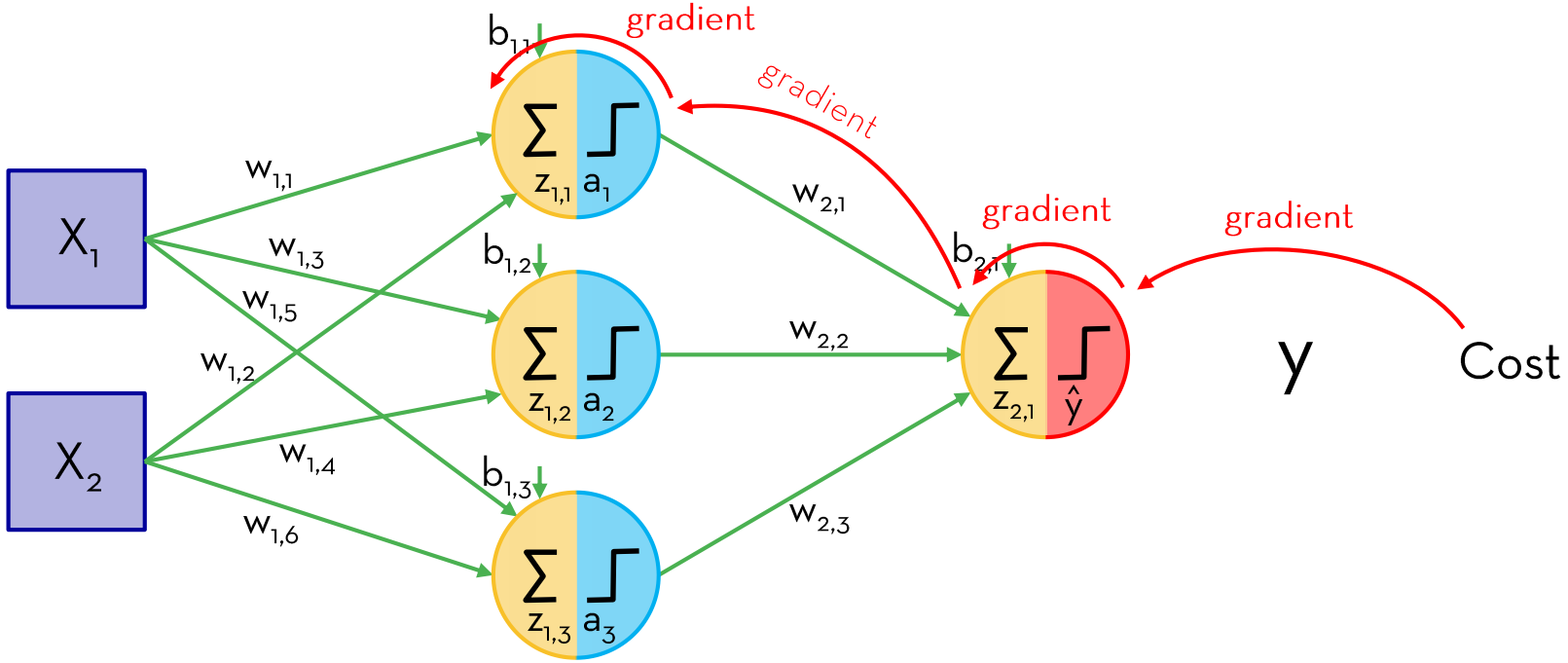
We repeat the same procedure for all observations.

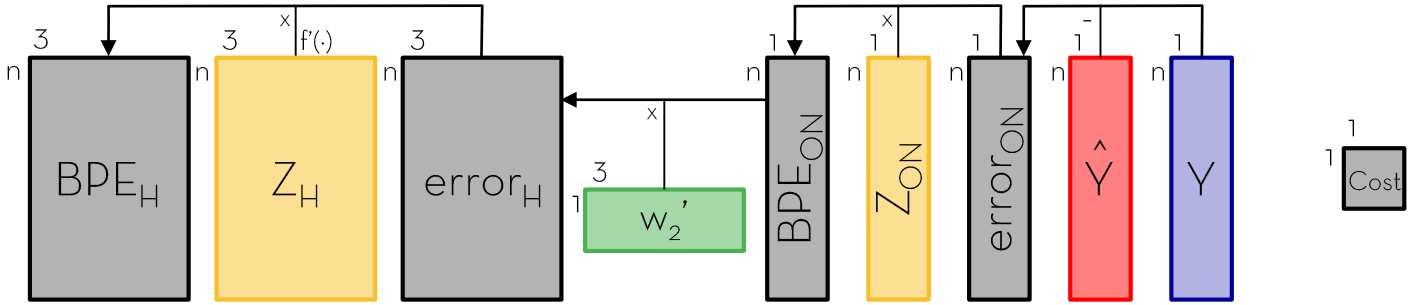
We will need these values to update the weights.

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6

A summary of the general principle of backpropagation



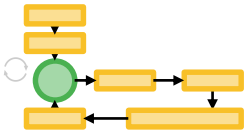


http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Steps of estimating neural networks

1. Design the neural network.
2. Initialize the neural network.
3. Forward propagation.
4. Calculate the cost.
5. Backpropagation.
6. Update weights.

Repeat 3. – 6. until convergence criteria is met.



http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

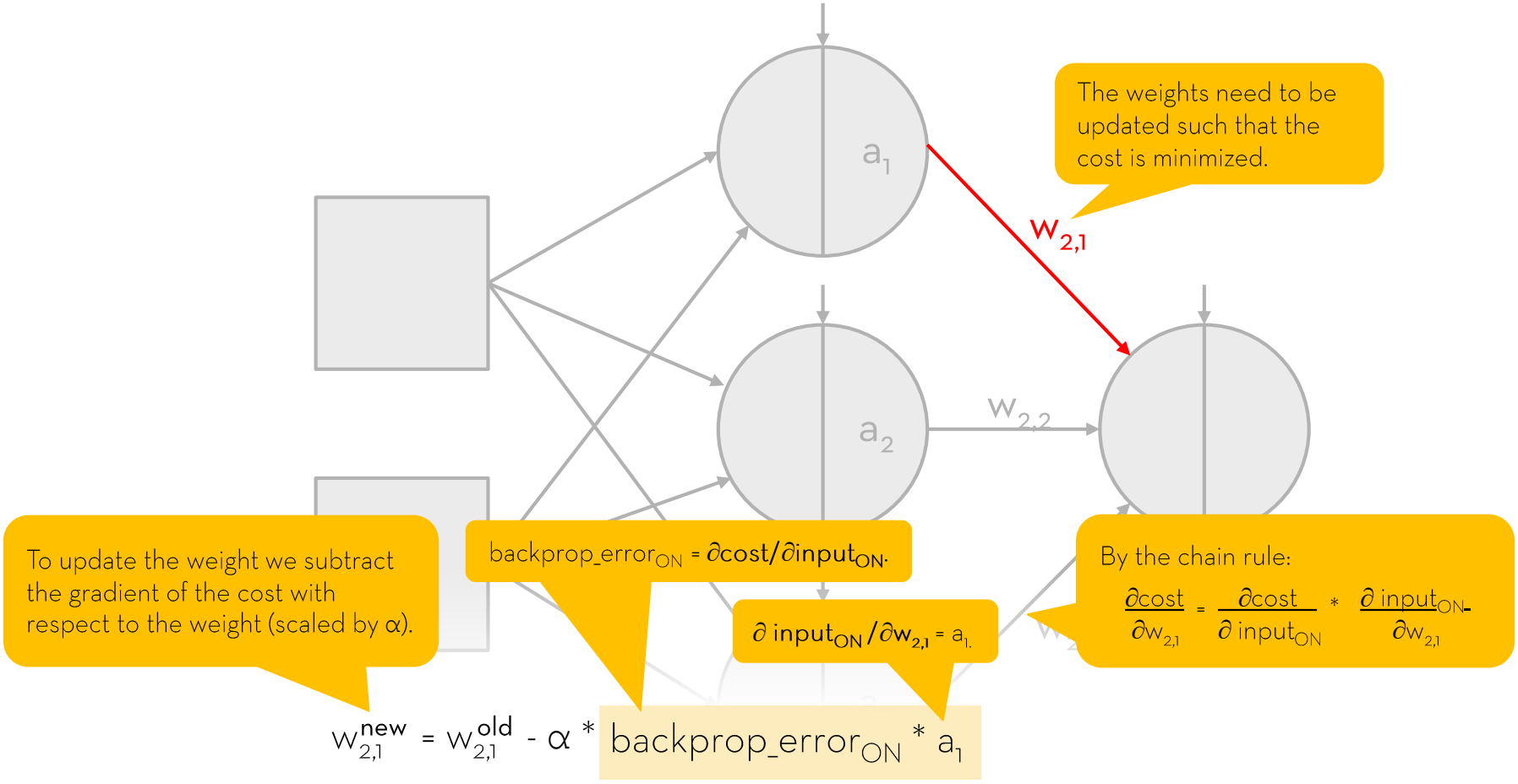
Step 6: Update weights
How to optimize the weights?

a) Update the weights and biases leading to the output node.

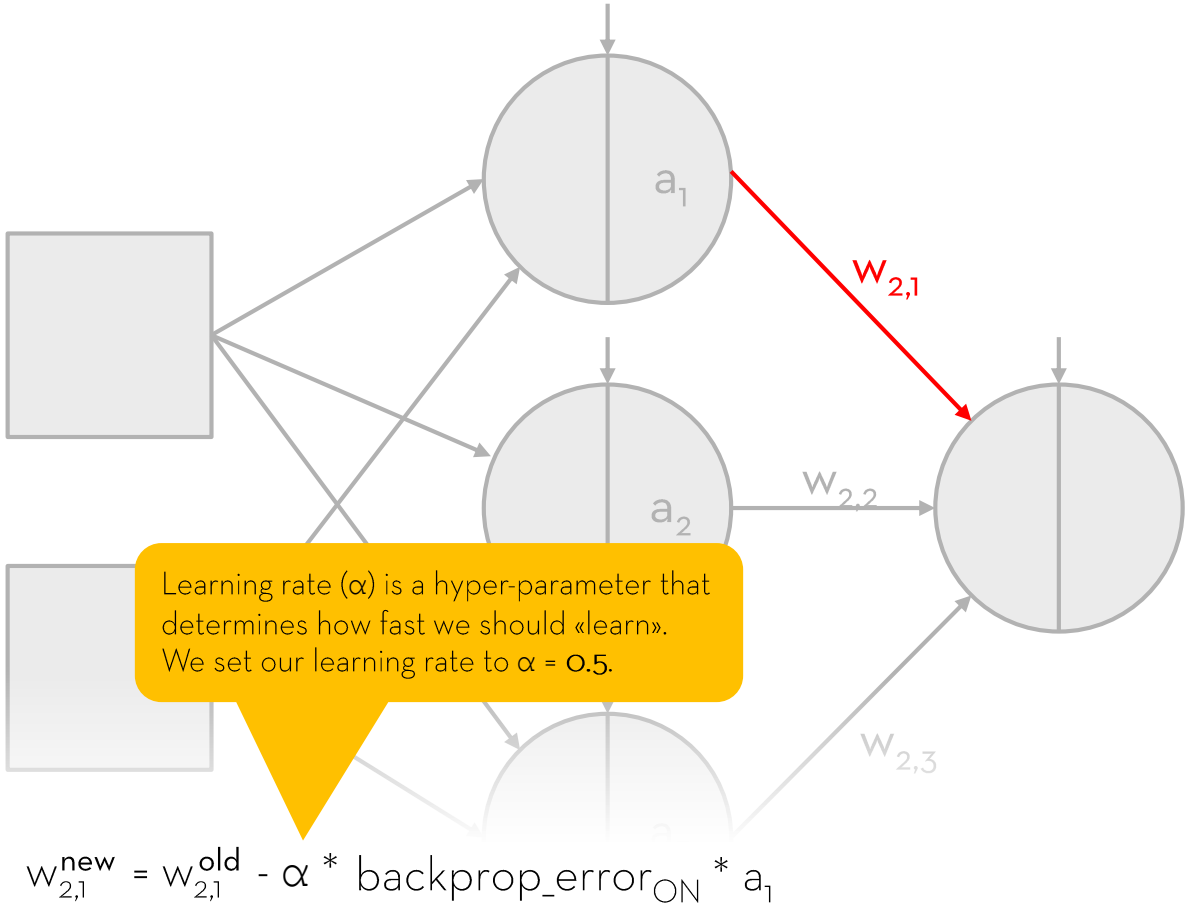
b) Update the weights and biases leading to the hidden layer.

http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

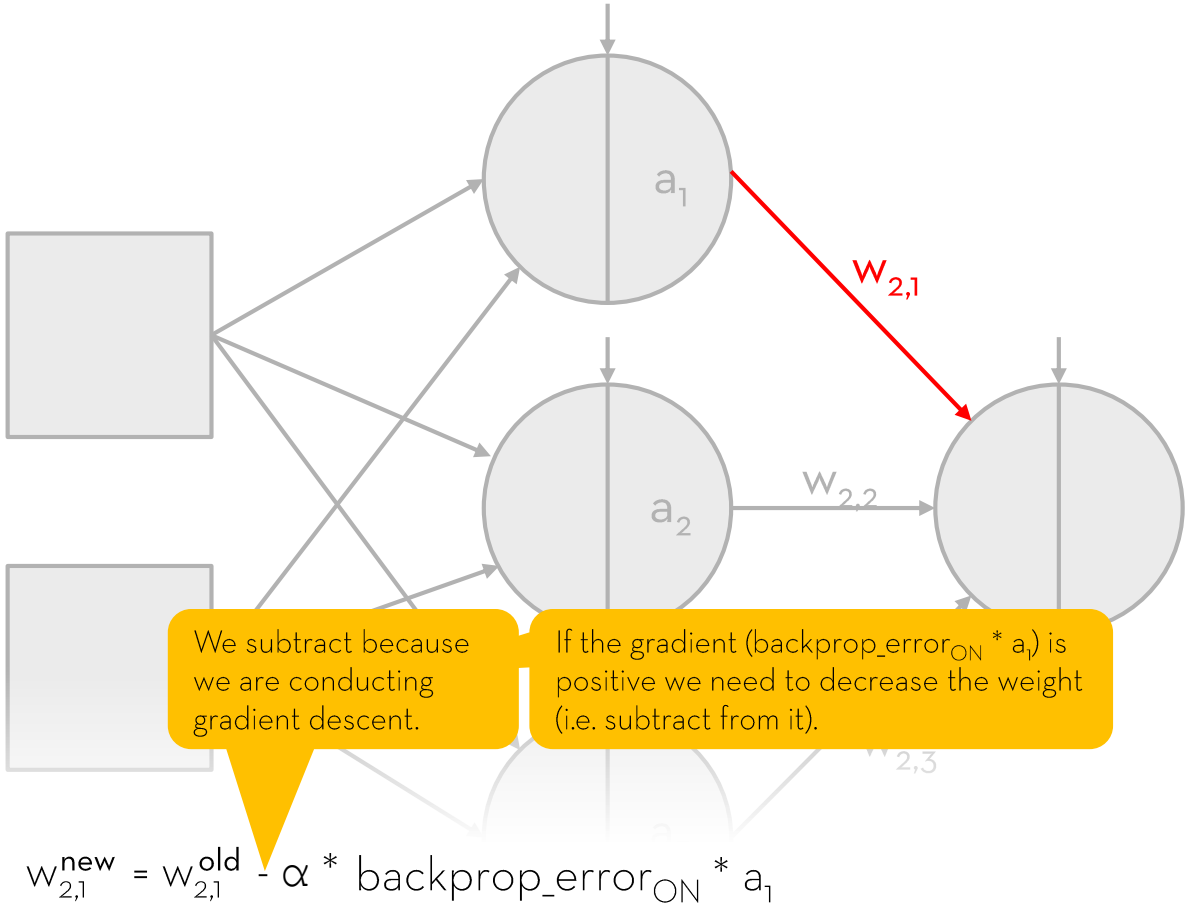
Step 6: Update weights
How to update the weights and biases? (1/4)



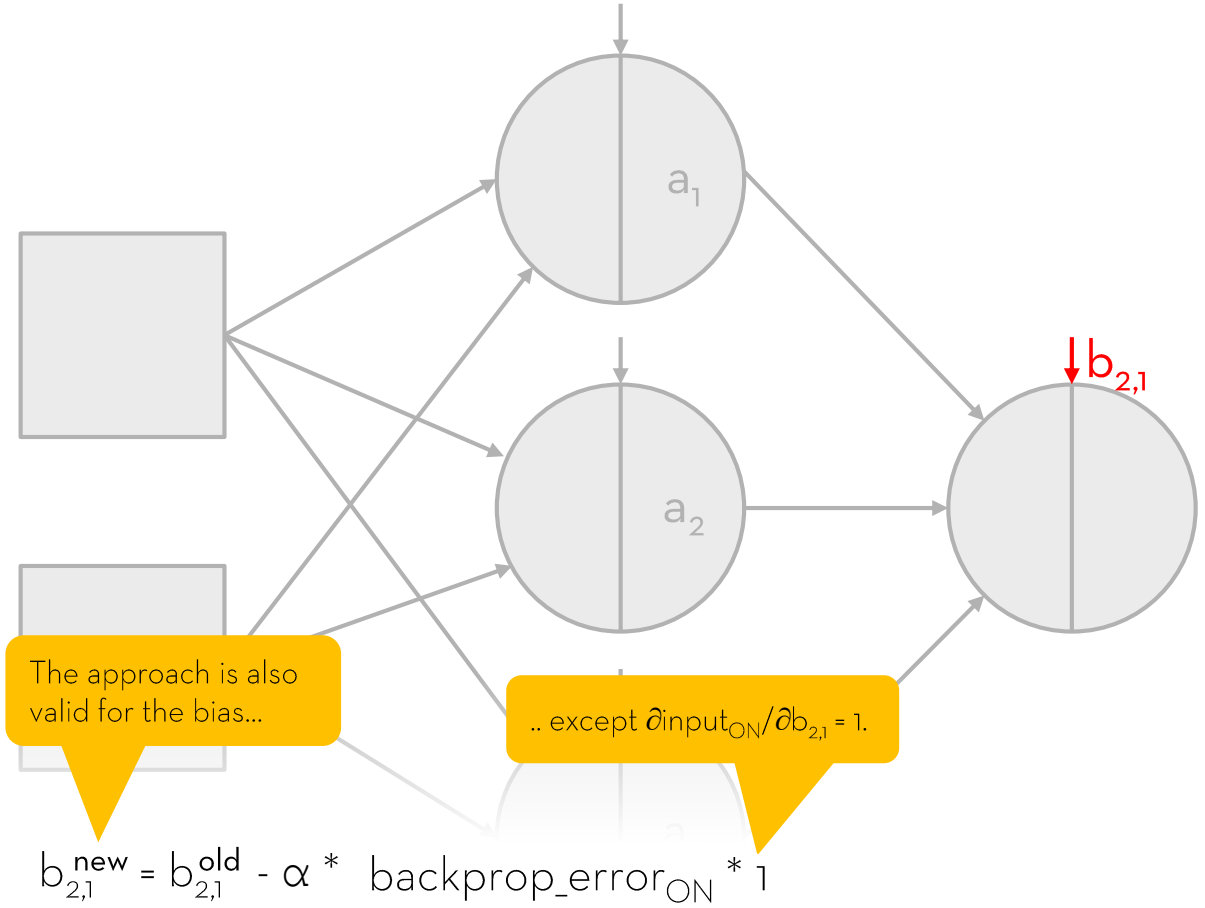
Step 6: Update weights
How to update the weights and biases? (2/4)



Step 6: Update weights
How to update the weights and biases? (3/4)



Step 6: Update weights
How to update the weights and biases? (4/4)

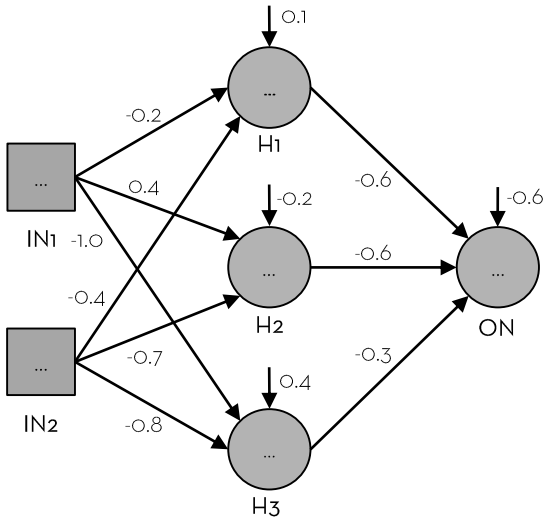


Step 6: Update weights
a) Update the weights and biases leading to the ON

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

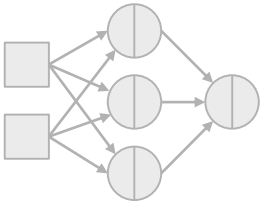
	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

We import the output BPEs for every observation from the previous section to facilitate understanding.



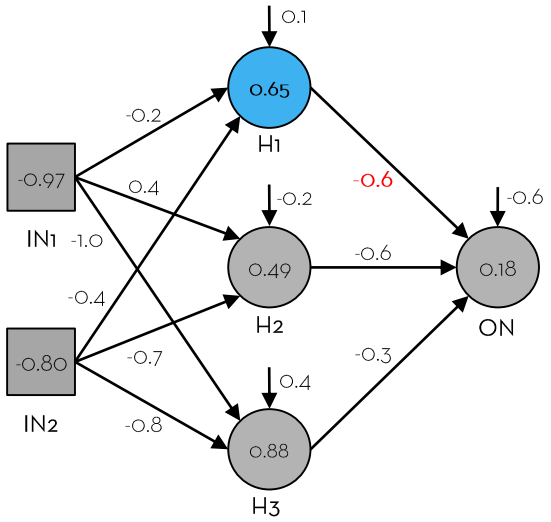
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

For familiarity, we will keep the backpropagation here for the first observation.



Step 6: Update weights
a) Update the weights leading to the ON (1/9)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

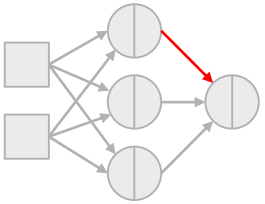


	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.6	-0.6	-0.3

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

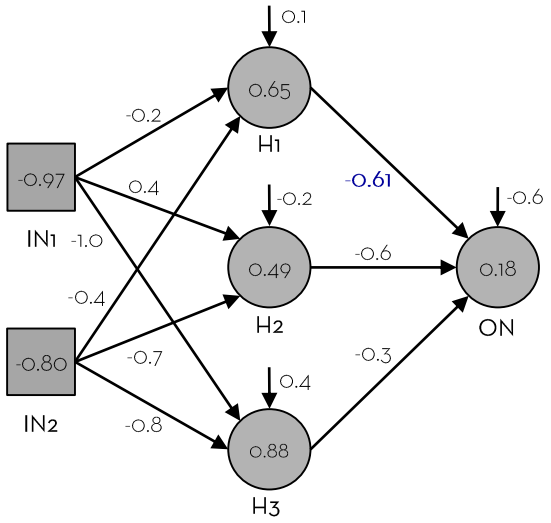
To get the **new weight** we take the old weight, subtract the backpropagation error multiplied by the **learning rate 0.5** and the value of the **H1**.

$$\text{weight}_{H1\rightarrow ON} = \text{weight}_{H1\rightarrow ON} - \alpha * \text{backprop\_error}_{ON} * a_{H1,1}$$
$$= -0.6 - 0.5 * 0.025 * 0.65$$
$$= -0.608$$



Step 6: Update weights
a) Update the weights leading to the ON (2/9)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

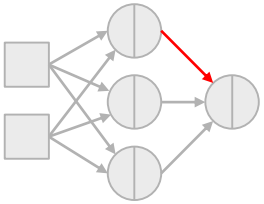


	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.608	-0.6	-0.3

The weight changes from -0.6 to -0.608.

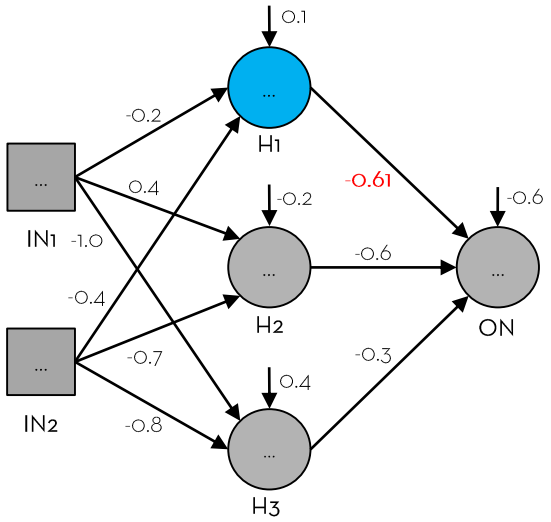
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

Notice: the weight decreased from the original value since the gradient is positive.



Step 6: Update weights
a) Update the weights leading to the ON (3/9)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.608	-0.6	-0.3

	H1	H2	H3	
Bias	0.1	-0.2	0.4	
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

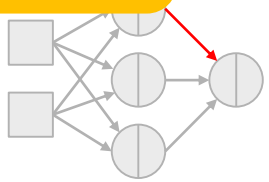
We repeat this for every observation:

$weight_{H1 \rightarrow ON} = weight_{H1 \rightarrow ON} - \alpha * (backprop\_error_{ON2} * a_{H1,2} + ... + backprop\_error_{ON10} * a_{H1,10})$

$= -0.608 - 0.5 * (0.024 * 0.65 + ... + 0.039 * 0.44)$

$= -0.496$

Since updating the weight is linear, you can update the weight one observation at a time, or all together.



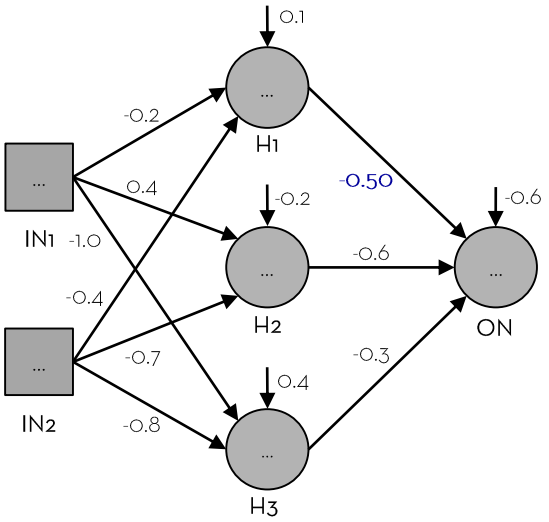
Step 6: Update weights
a) Update the weights leading to the ON (4/9)

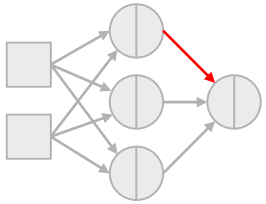
Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.6	

The final weight after updating the weight with every observation is -0.496.

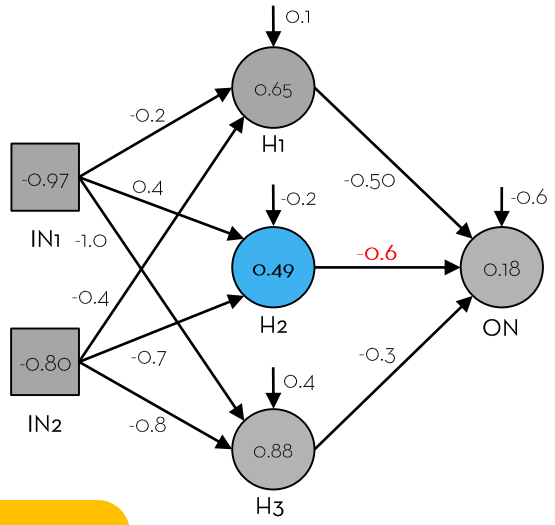
	H1	H2	H3	
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025





Step 6: Update weights
a) Update the weights leading to the ON (5/9)

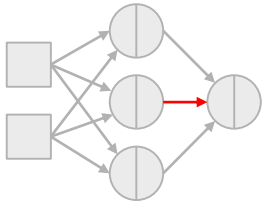
Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	0.496	-0.6	-0.3

$$weight_{H_2 \rightarrow ON} = weight_{H_2 \rightarrow ON} - \alpha * backprop\_error_{ON} * a_{H_{2,1}}$$
$$= -0.6 - 0.5 * 0.025 * 0.49$$
$$= -0.606$$

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025



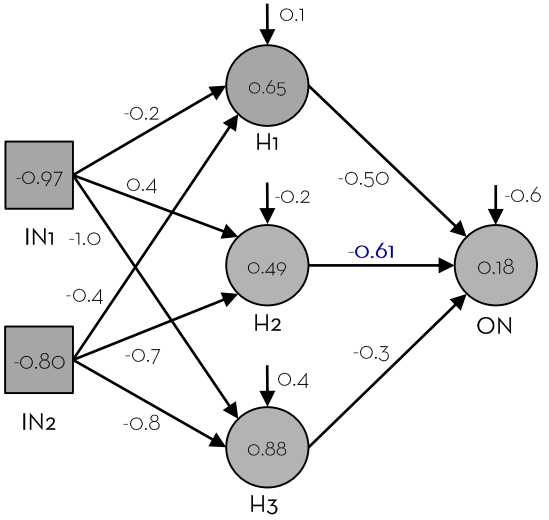
Step 6: Update weights
a) Update the weights leading to the ON (6/9)

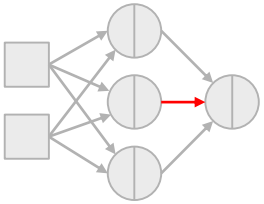
Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.606	-0.7

The weight changes from -0.6 to -0.606.

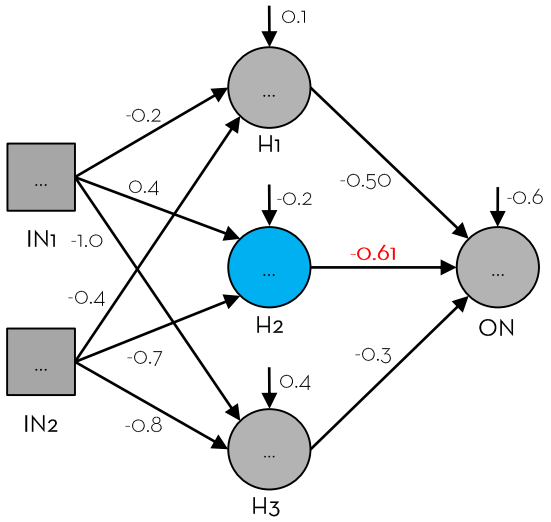
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025





Step 6: Update weights
a) Update the weights leading to the ON (7/9)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.606	-0.3

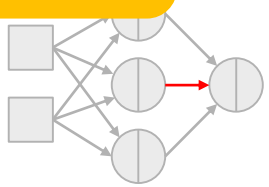
	H1	H2	H3	
Bias	0.1	-0.2	0.4	
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

We repeat this for every observation:

$weight_{H2 \rightarrow ON} = weight_{H2 \rightarrow ON} - \alpha * (backprop\_error_{ON2} * a_{H1,2} + ... + backprop\_error_{ON10} * a_{H1,10})$

$= -0.606 - 0.5 * (0.024 * 0.55 + ... + 0.039 * 0.49)$

$= -0.506$



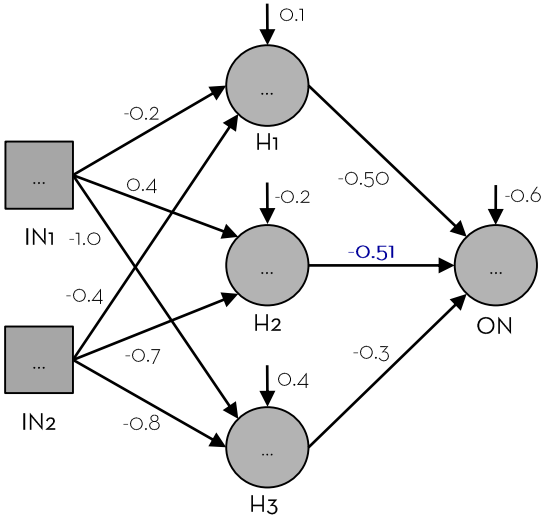
Step 6: Update weights
a) Update the weights leading to the ON (8/9)

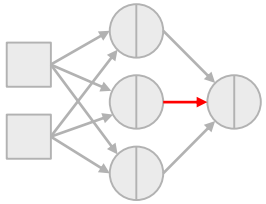
Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.3

The final weight after updating the weight with every observation is -0.506.

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025





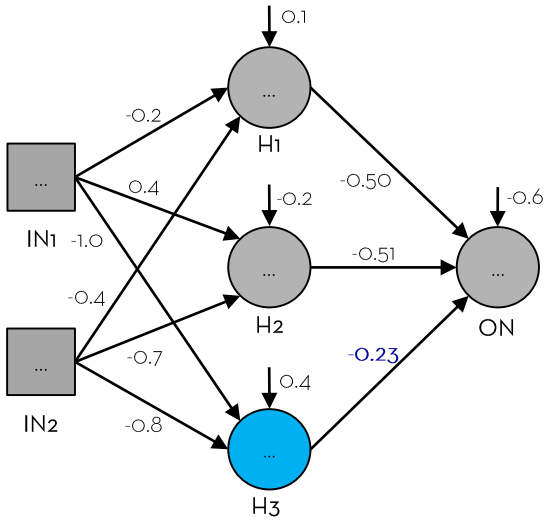
Step 6: Update weights
a) Update the weights leading to the ON (9/9)

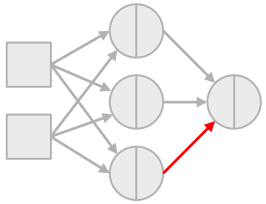
Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

Repeat for the final weight.

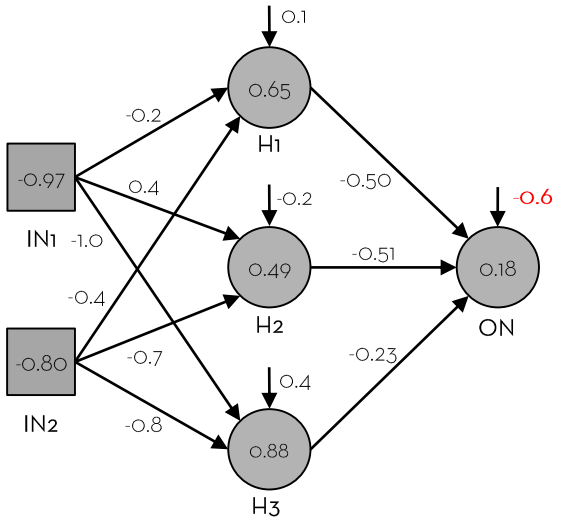
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025





Step 6: Update weights
a) Update the bias of the ON (1/3)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

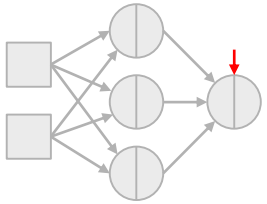
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.6
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

We **update the bias** as follows:

$bias_{ON} = bias_{ON} - \alpha * backprop\_error_{ON1}$

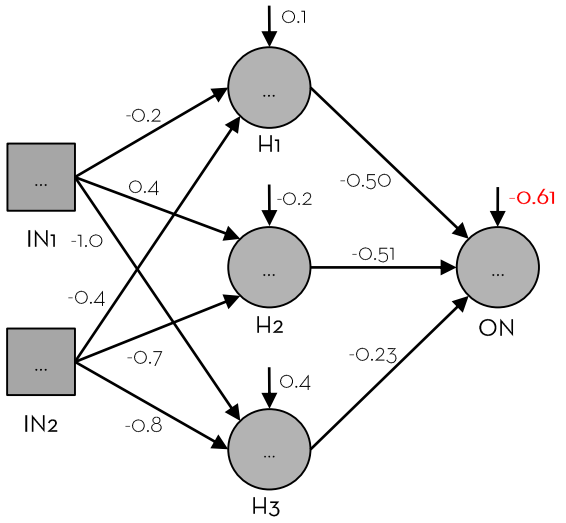
$= -0.6 - 0.5 * 0.025$

$= -0.613$



Step 6: Update weights
a) Update the bias of the ON (2/3)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	



	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

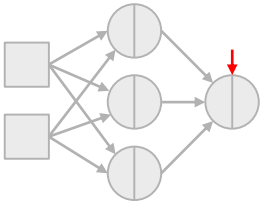
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.613
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

And repeat for every observation.

$bias_{ON} = bias_{ON} - \alpha * (backprop\_error_{ON_2} + ... + backprop\_error_{ON_{10}})$.

$= -0.613 - 0.5 * (0.024 + ... + 0.039)$

$= -0.327$



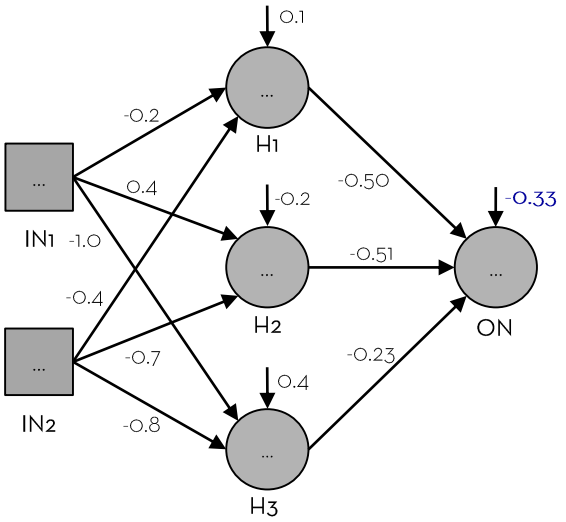
Step 6: Update weights
a) Update the bias of the ON (3/3)

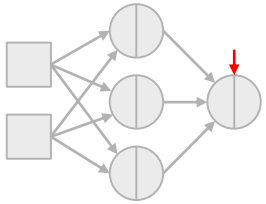
Y	X1	X2	H1	H2	H3	Output	Cost	BPE ON
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	0.025
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	0.024
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	0.023
...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	-0.122
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	0.039
							Σ = 1.55	

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

The bias of ON changes to -0.327.





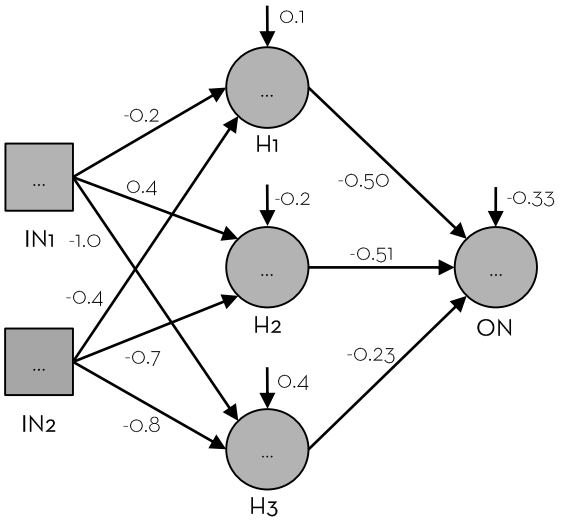
Step 6: Update weights
b) Update the weights and biases leading to the HL

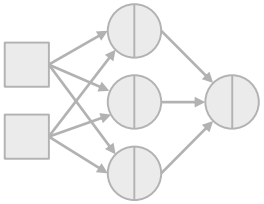
Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			

	W1	W2	W3
IN1 → HL	-0.2	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

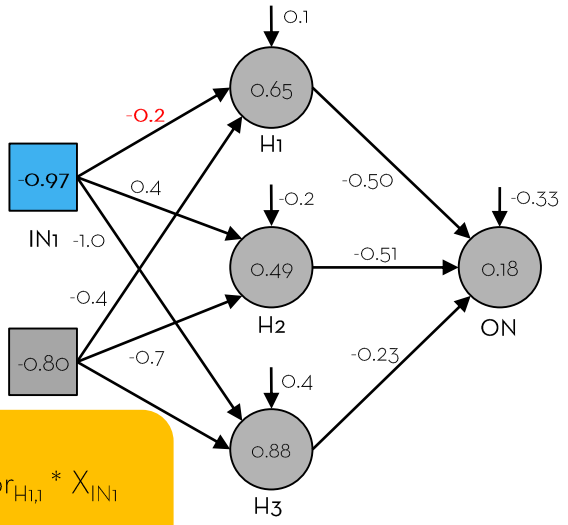
Now, we import the hidden layer BPEs for every observation from the previous section to facilitate understanding.





Step 6: Update weights
b) Update the weights leading to the HL nodes (1/5)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			



	W1	W2	W3
IN1 → HL	-0.2		
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

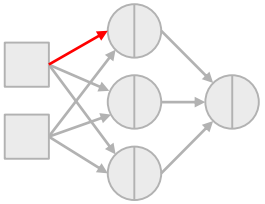
Update the weight from IN1 to H1:

$weight_{IN \rightarrow H1} = weight_{IN \rightarrow H1} - \alpha * backprop\_error_{H1,1} * X_{IN1}$

$= -0.2 - 0.5 * -0.003 * -0.97$

$= -0.202$

	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

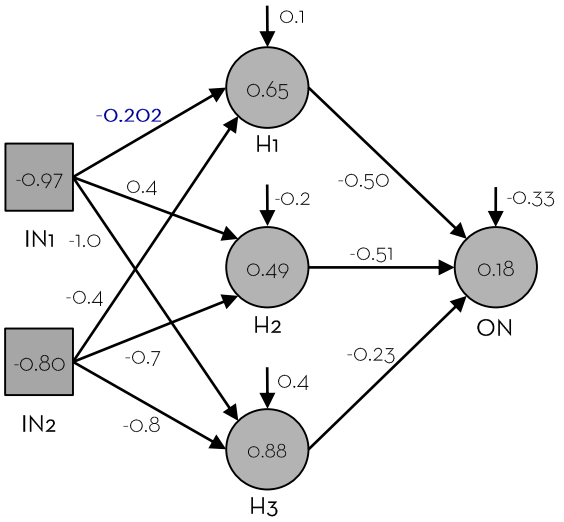


Step 6: Update weights
b) Update the weights leading to the HL nodes (2/5)

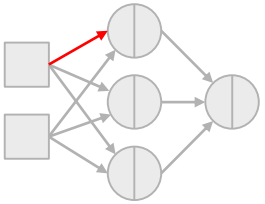
Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			

	W1	W2	W3
IN1 → HL	-0.202	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

The weight changes from -0.2 to -0.202.

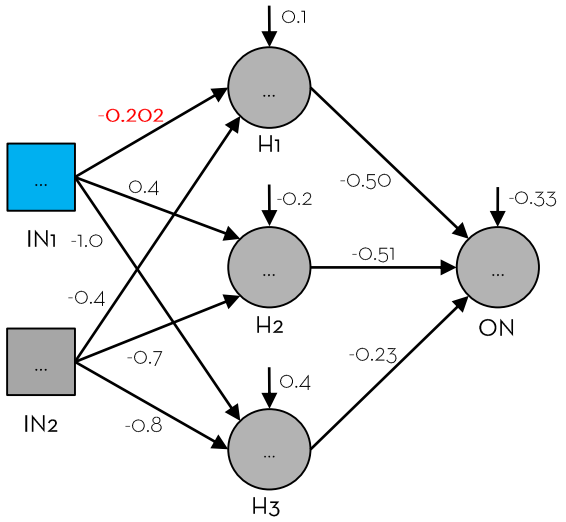


	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025



Step 6: Update weights
b) Update the weights leading to the HL nodes (3/5)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			



	W1	W2	W3
IN1 → HL	-0.202	0.4	-1.0
IN2 → HL	-0.4	-0.7	-0.8
HL → ON	-0.496	-0.506	

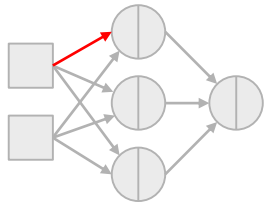
	H1
Bias	0.1
Error	-0.015
Backprop. Err	-0.003	-0.004	-0.001	0.025

We repeat this for every observation:

$weight_{IN \rightarrow H1} = weight_{IN \rightarrow H1} - \alpha * (backprop\_error_{HL1,2} * X_{IN2} + ... + backprop\_error_{HL1,10} * X_{IN10})$

$= -0.202 - 0.5 * (-0.003 * -0.66 + ... + -0.006 * 1.04)$

$= -0.230$

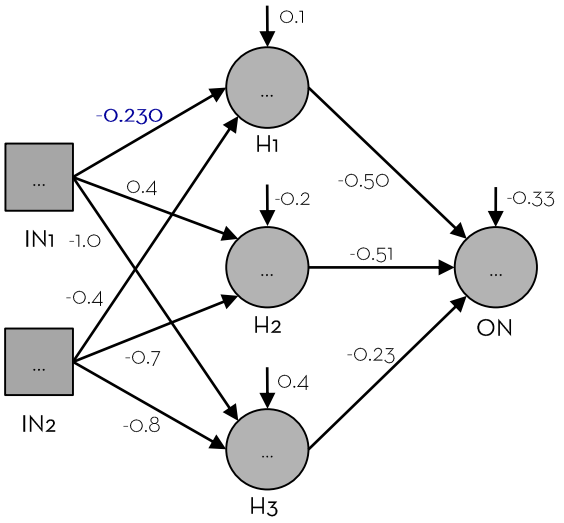


Step 6: Update weights
b) Update the weights leading to the HL nodes (4/5)

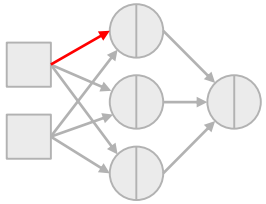
Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			

	W1	W2	W3
IN1 → HL	-0.230	0.4	
IN2 → HL	-0.4	-0.7	-0.
HL → ON	-0.496	-0.506	-0.2

The final weight after updating the weight with every observation is -0.230.



	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025



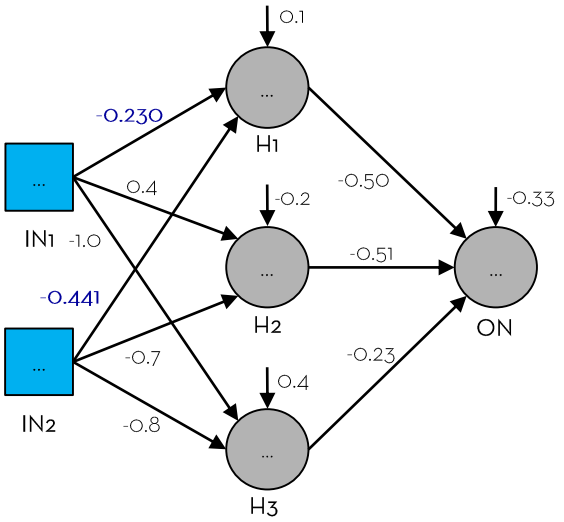
Step 6: Update weights
b) Update the weights leading to the HL nodes (5/5)

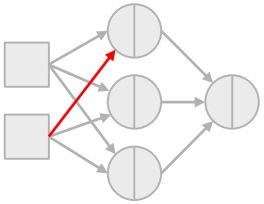
Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			

	W1	W2	W3
IN1 → HL	-0.230	0.4	-1.0
IN2 → HL	-0.441	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

We update both weights of this hidden layer.

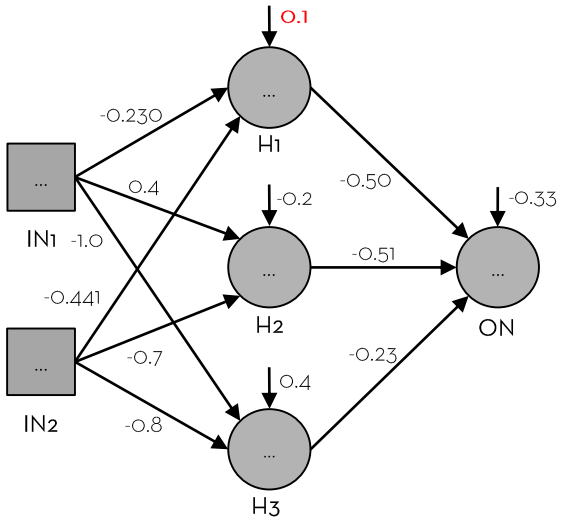
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025





Step 6: Update weights
b) Update the bias of the HL nodes (1/2)

Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			



	W1	W2	W3
IN1 → HL	-0.230	0.4	-1.0
IN2 → HL	-0.441	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

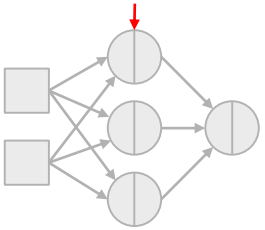
	H1	H2	H3	ON
Bias	0.1	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

We **update the bias** as follows:

$bias_{H1} = bias_{H1} - \alpha * (backprop\_error_{HL1,1} + ... + backprop\_error_{HL1,10})$

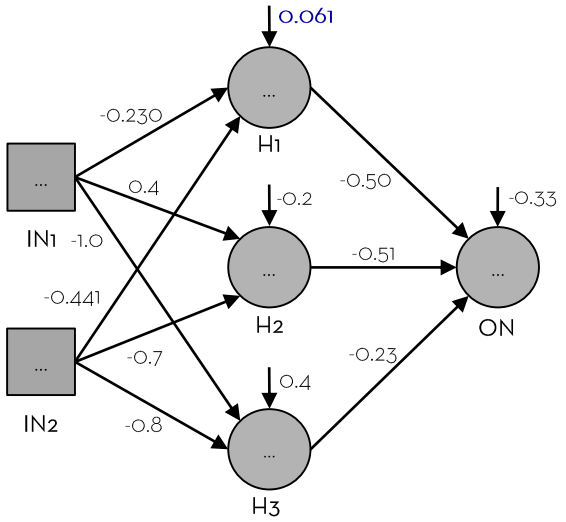
$= 0.1 - 0.5 * (-0.003 + ... + -0.006)$

$= 0.061$



Step 6: Update weights
b) Update the bias of the HL nodes (2/2)

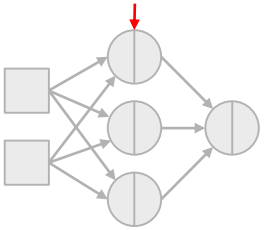
Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			



	W1	W2	W3
IN1 → HL	-0.230	0.4	-1.0
IN2 → HL	-0.441	-0.7	-0.8
HL → ON	-0.496	-0.506	-0.233

	H1	H2	H3	ON
Bias	0.061	-0.2	0.4	-0.327
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

The bias changes from 0.1 to 0.061.



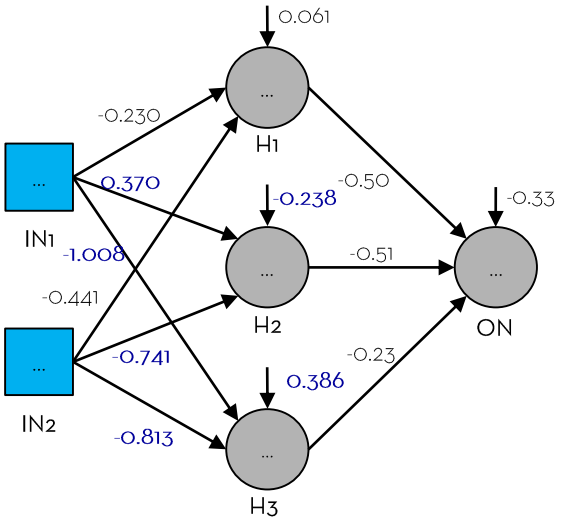
Step 6: Update weights
b) Update the rest of the weights and biases

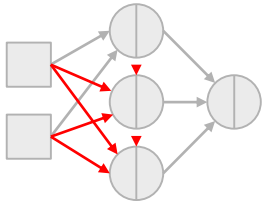
Y	X1	X2	H1	H2	H3	Output	Cost	BPE H1	BPE H2	BPE H3
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03	-0.003	-0.004	-0.001
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03	-0.003	-0.004	-0.001
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03	-0.003	-0.004	-0.001
...	...	...	...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67	0.017	0.018	0.005
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05	-0.006	-0.006	-0.002
							Σ = 1.55			

	W1	W2	W3
IN1 → HL	-0.230	0.370	-1.008
IN2 → HL	-0.441	-0.741	-0.813
HL → ON	-0.496	-0.506	-0.233

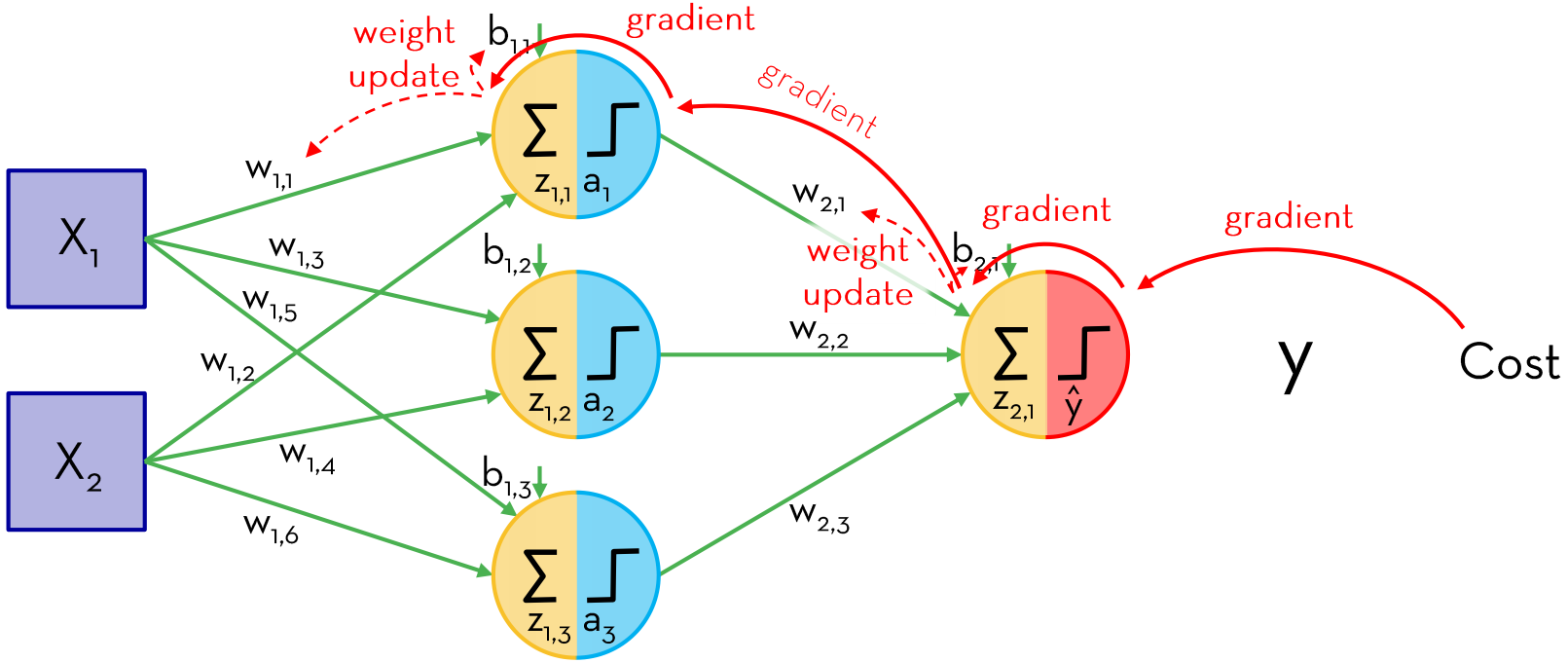
We proceed in the **same way** for the rest of the HL and update all weights and biases.

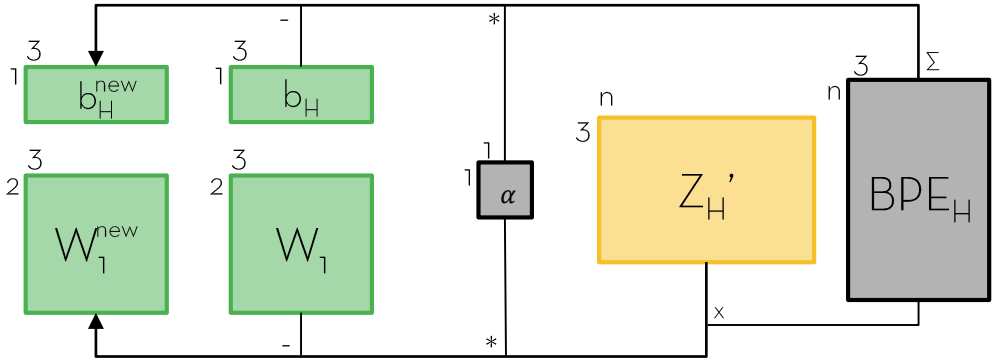
	H1	H2	H3	ON
Bias	0.061	-0.238	0.386	0.386
Error	-0.015	-0.015	-0.008	0.18
Backprop. Err	-0.003	-0.004	-0.001	0.025

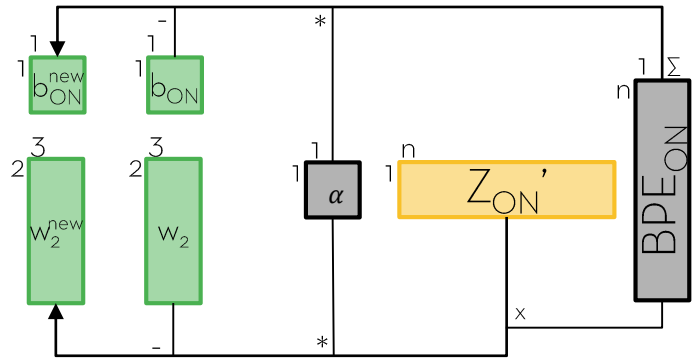




A summary of the general principle of updating the weights using gradient descent







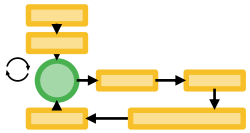
http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Repeat steps 3. – 6. until convergence criteria is met

- We now **repeat** the forward propagation and backpropagation process.
- One **iteration** (i.e., estimating once forward propagation and backpropagation for the entire dataset) **is called one cycle or epoch**.
- **Training of neural networks requires many cycles** (10'000s of cycles) until the neural network is good enough (the summed-up error doesn't change anymore).
- We set the following training goal: **summed-up error < 0.001.**

There are **many possible training goals**, for example:

1) Maximum number of epochs reached.
2) Maximum amount of time exceeded/performance minimized to a threshold.
3) Performance gradient falls below a minimum.
4) Validation performance has increased x times since the last time it decreased (when using validation).



https://ch.mathworks.com/help/nnet/ref/traingd.html

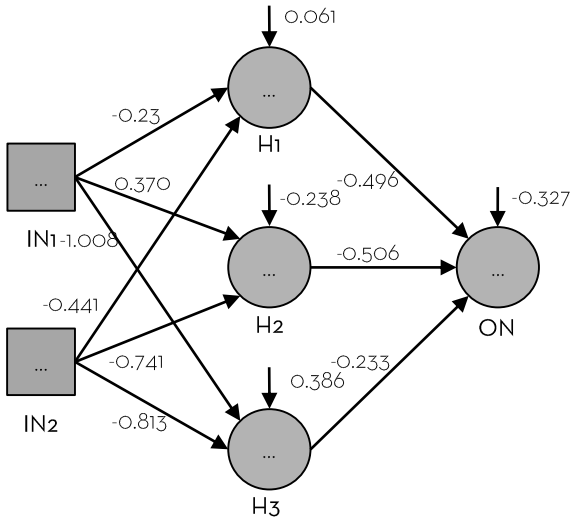
After 1 epoch

During iterative training of a neural network , an **epoch is a single pass through the entire training set**, followed by testing of the verification set. When training in mini-batches, then an epoch consist of multiple iterations.

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.65	0.49	0.88	0.18	0.03
0	-0.69	-0.97	0.65	0.55	0.87	0.17	0.03
0	-1.14	-1.09	0.68	0.53	0.92	0.17	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.61	0.46	0.83	0.18	0.67
0	1.04	0.34	0.44	0.49	0.29	0.22	0.05
							Σ = 1.55

	W1	W2	W3
IN1 → HL	-0.230	0.370	-1.008
IN2 → HL	-0.441	-0.741	-0.813
HL → ON	-0.496	-0.506	-0.233

	H1	H2	H3	ON
Bias	0.061	-0.238	0.386	-0.327



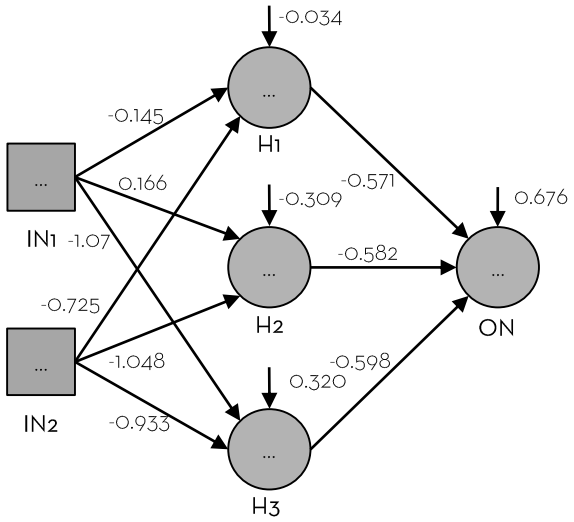
https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

After 10 epochs

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.71	0.58	0.89	0.36	0.13
0	-0.69	-0.97	0.72	0.63	0.88	0.36	0.13
0	-1.14	-1.09	0.77	0.64	0.93	0.34	0.12
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.66	0.52	0.84	0.38	0.38
0	1.04	0.34	0.34	0.39	0.25	0.52	0.27
							Σ = 0.96

	W1	W2	W3
IN1 → HL	-0.415	0.166	-1.071
IN2 → HL	-0.725	-1.048	-0.933
HL → ON	-0.571	-0.582	-0.598

	H1	H2	H3	ON
Bias	-0.034	-0.309	0.320	0.676

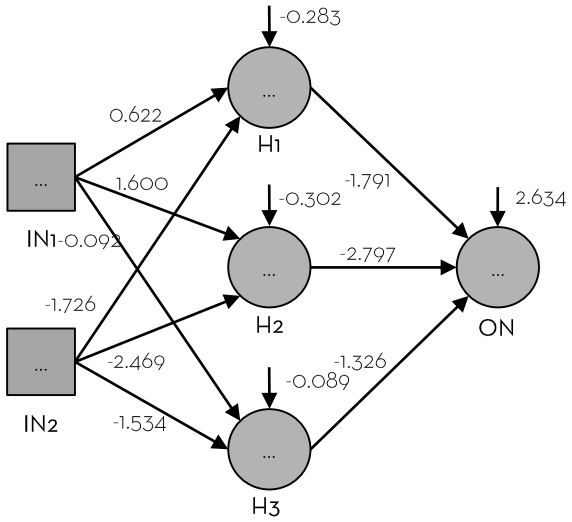


After 100 epochs

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.62	0.53	0.78	0.27	0.07
0	-0.69	-0.97	0.72	0.73	0.78	0.15	0.02
0	-1.14	-1.09	0.71	0.64	0.85	0.18	0.03
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.53	0.43	0.69	0.39	0.37
0	1.04	0.34	0.44	0.63	0.33	0.42	0.17
							Σ = 0.41

	W1	W2	W3
IN1 → HL	0.622	1.600	-0.092
IN2 → HL	-1.726	-2.469	-1.534
HL → ON	-1.791	-2.797	-1.326

	H1	H2	H3	ON
Bias	-0.283	-0.302	-0.089	2.634

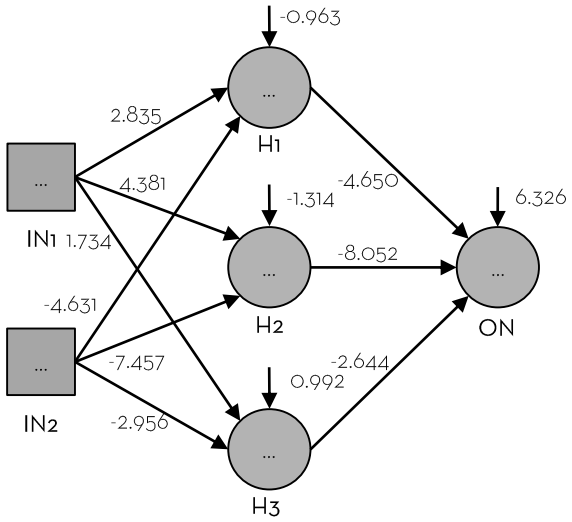


After 1'000 epochs

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.50	0.60	0.42	0.127	0.016
0	-0.69	-0.97	0.83	0.95	0.66	0.001	0.000
0	-1.14	-1.09	0.70	0.86	0.56	0.005	0.000
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.31	0.29	0.30	0.851	0.022
0	1.04	0.34	0.60	0.67	0.45	0.045	0.002
							Σ = 0.02

	W1	W2	W3
IN1 → HL	2.835	4.381	1.734
IN2 → HL	-4.631	-7.457	-2.956
HL → ON	-4.650	-8.052	-2.644

	H1	H2	H3	ON
Bias	-0.963	-1.314	0.992	6.326



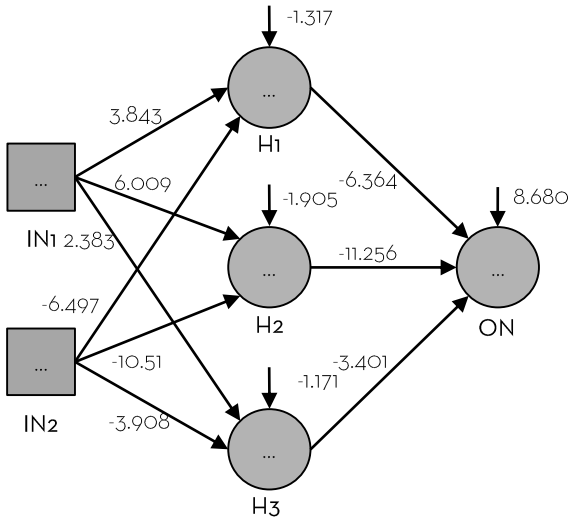
After 8'000 epochs

Y	X1	X2	H1	H2	H3	Output	Cost
0	-0.97	-0.80	0.54	0.66	0.41	0.027	0.001
0	-0.69	-0.97	0.91	0.98	0.73	0.000	0.000
0	-1.14	-1.09	0.80	0.94	0.59	0.000	0.000
...	...	...	...	...	...	...	...
1	-0.79	-0.52	0.27	0.23	0.26	0.968	0.001
0	1.04	0.34	0.62	0.68	0.49	0.010	0.000
							Σ < 0.001

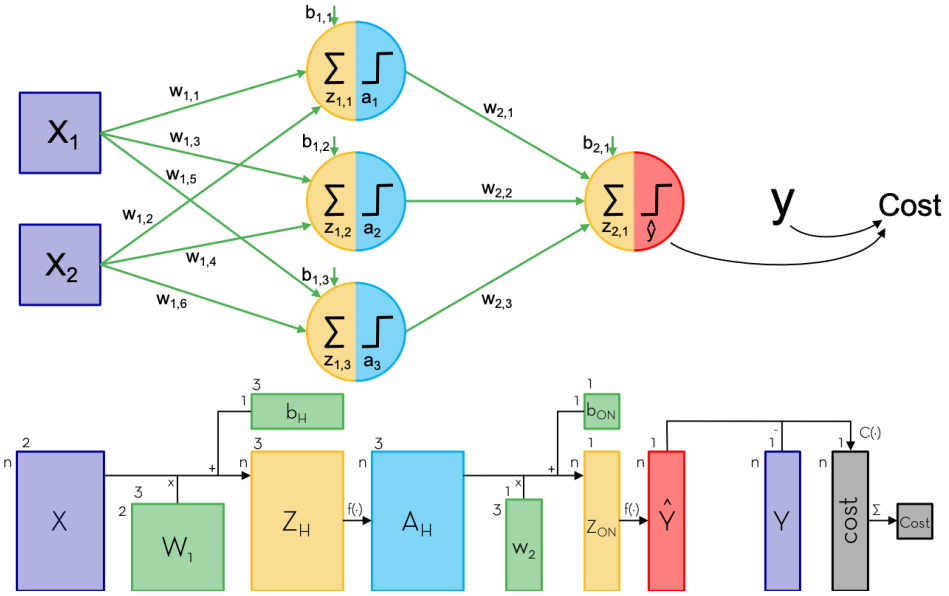
Goal reached!

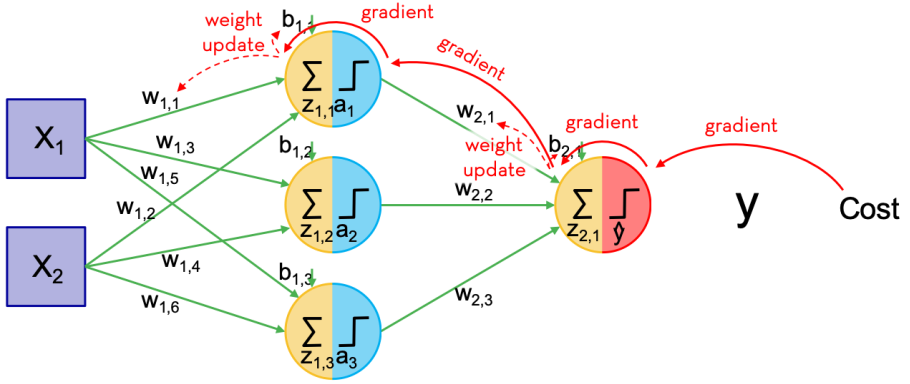
	W1	W2	W3
IN1 → HL	3.843	6.009	2.383
IN2 → HL	-6.497	-10.508	-3.908
HL → ON	-6.364	-11.256	-3.401

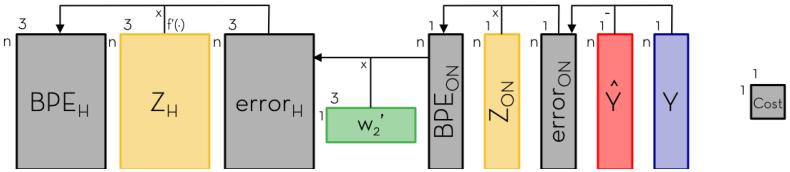
	H1	H2	H3	ON
Bias	-1.317	-1.905	-1.171	8.680



Summary of the neural network training algorithm



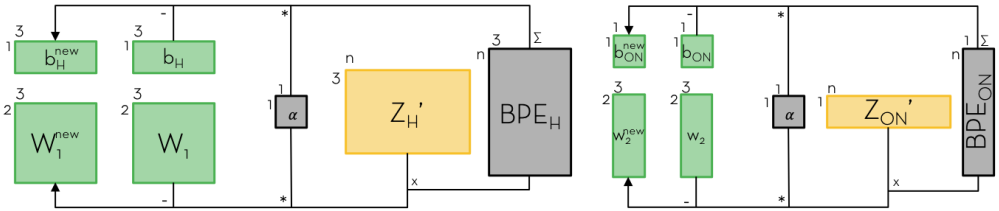




Forward pass.

Calculate cost.

Backpropagation.



Weight update.

http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Summary of the neural network training algorithm

3. Forward propagation:

   a) Calculate the values of the hidden layer: $z_H = f(\overbrace{\Sigma_{i \in IN} X_i w_{i \to H} + b_H}^{z_H})$ for all $H \in HL$.

   b) Calculate the values of the output layer: $\hat{y} = f(\underbrace{\Sigma_{i \in HL} a_i w_{i \to ON} + b_{ON}}_{z_{ON}})$.

IN: input nodes.
ON: output node.
HL: hidden layer nodes.
$f(\cdot)$: sigmoid function.

4. Backpropagation:

   a) For each layer (in reverse):

      1. Calculate error:

      2. Calculate the backpropagation error (BPE):

Output node (ON)	Hidden node $H \in HL$.
$(\hat{y} - y)$	$BPE_{ON} * w_{H \to ON}$
$error_{ON} * f'(z_{ON})$	$error_H * f'(z_H)$

5. Weight updates:

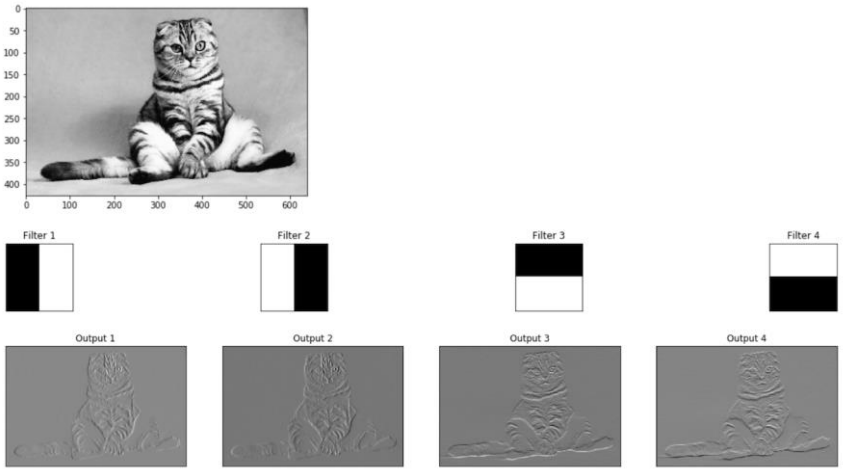
   a) Update the weights: $w^{new}_{from \to to} = w^{old}_{from \to to} - \alpha * bpe_{to} * a_{from}$.

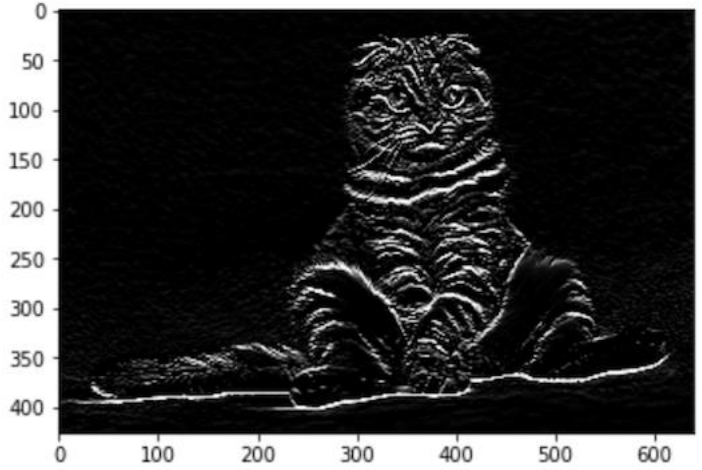
   b) Update the biases: $b^{new}_{node} = b^{old}_{node} - \alpha * bpe_{node}$.

Advantages and disadvantages of neural networks

Advantages:

- Relatively simple learning algorithm.
- Can approximate any function, regardless of its linearity.
- Great for complex / abstract problems like image recognition.





https://www.quora.com/What-are-the-advantages-disadvantages-of-Artificial-Neural-networks
http://neuralnetworksanddeeplearning.com/chap4.html
https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb

Advantages and disadvantages of neural networks

Disadvantages:

- Difficult to interpret what a neural network has learned and why.
- Cannot deal with uncertainties.
- Computationally intensive to train.
- Mimics the brain – machine cannot completely copy a complex organism.

Breaking Linear Classifiers on ImageNet

Mar 30, 2015

You've probably heard that Convolutional Networks work very well in practice and across a wide range of visual recognition problems. You may have also read articles and papers that claim to reach a near *"human-level performance"*. There are all kinds of caveats to that (e.g. see my G+ post on Human Accuracy is not a point, it lives on a tradeoff curve), but that is not the point of this post. I do think that these systems now work extremely well across many visual recognition tasks, especially ones that can be posed as simple classification.

Yet, a second group of seemingly baffling results has emerged that brings up an apparent contradiction. I'm referring to several people who have noticed that it is possible to take an image that a state-of-the-art Convolutional Network thinks is one class (e.g. "panda"), and it is possible to change it almost imperceptibly to the human eye in such a way that the Convolutional Network suddenly classifies the image as any other class of choice (e.g. "gibbon"). We say that we *break*, or *fool* ConvNets. See the image below for an illustration:

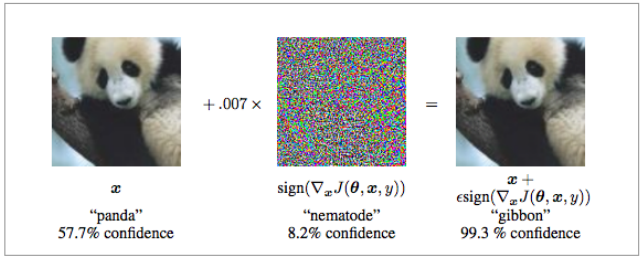


Figure from Explaining and Harnessing Adversarial Examples by Goodfellow et al.

http://karpathy.github.io/2015/03/30/breaking-convnets/

https://www.quora.com/What-are-the-advantages-disadvantages-of-Artificial-Neural-networks

Making machines think like humans.

How (Shallow) Neural Networks work

Dr. Markus Meierer - markus.meierer@business.uzh.ch | Luca Gaegauf - luca.gaegauf@bf.uzh.ch