Neural Network: Deeper Understanding

Example: track brand mentions more accurately with images

- Companies use social media analysis tools to track and analyze how people are talking about their brand.

- When collecting image mentions of your brand, you gain the ability to see who is using your products. Not just when they talk about using your products, but in their everyday lives. What is the setting of the photo? What else are they doing alongside using your product? Are they using other products as well?

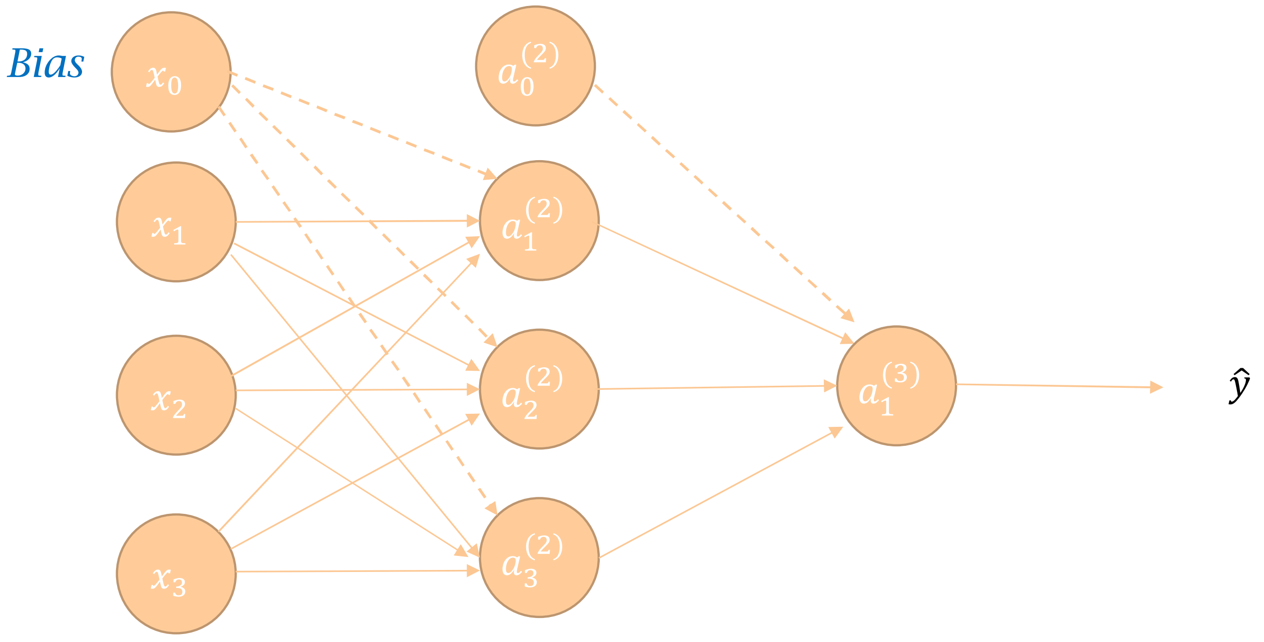
- In the past, the only way to find mentions of your brand or product within social posts was to look for text-based mentions or direct tags of your brand.

- Over three billion photos are shared daily on social media according to Mary Meeker. Many of those photos contain brands' products and logos, but 85% of them don't include a text reference to the brand (LogoGrab). Without image analytics, brands are missing out on a huge chunk of the social conversations about their brand, products, customers, and competitors.

- When a big brand like Nike wants to see how many people are talking about their products, one of their first steps may be to analyze their share of voice on social. But, as we've discussed, looking at these direct mentions is only one piece of the puzzle. What about the conversation that doesn't involve text? This is where "share of eye" is important.

An example

- Some studies done using image analysis found that Pabst Blue Ribbon beer logo was popular in and around the Portland area. What those audience researchers found was that this was happening because of beer hipsters. Of course!
- The beer brand had previously targeted older people and those living in northern or more rural areas.
- By applying this learning, the CMO managed to help bring PBR back nationally, they also improved their messaging by saying “look, it’s actually cool to drink PBR, people in Portland are doing it!”. So that’s a great example of some of the capabilities brands could have leveraging this technology.



1st Layer: Input layer with $n = 3$ input units (not counting bias)

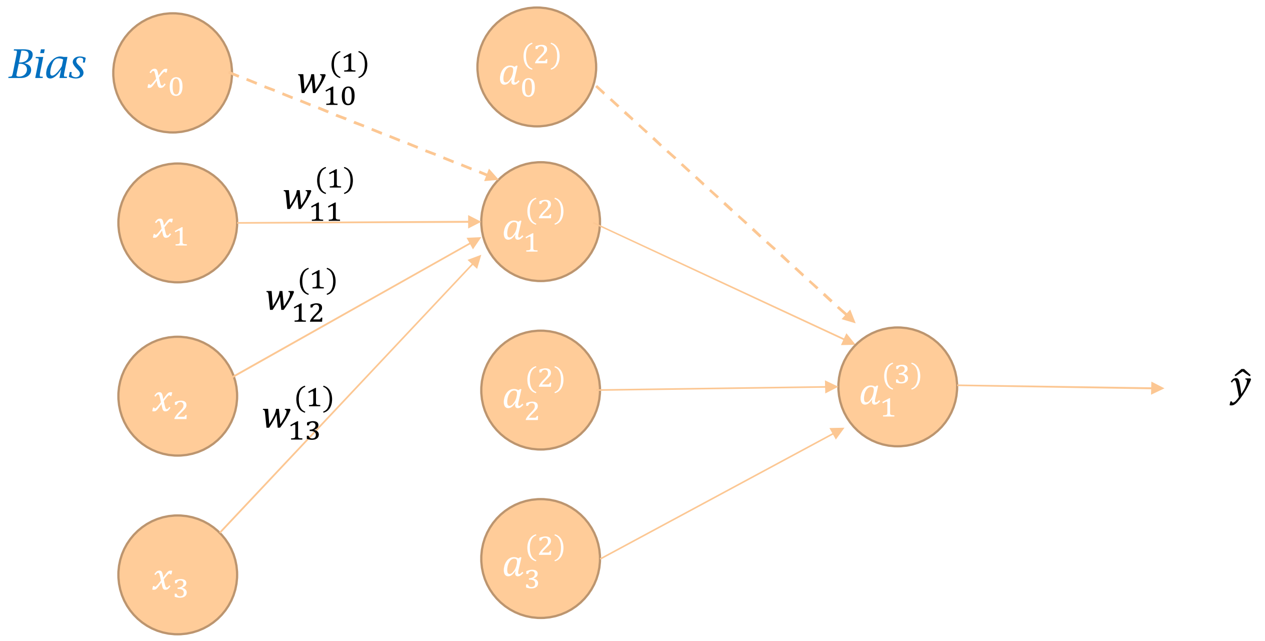
2nd Layer: Hidden layer with $h = 3$ hidden units, not counting bias

3rd Layer: Output layer

Neural Network: some notation

- n: number of features (inputs)
- h: number of hidden units
- $a_i^{(l)}$: “activation” of neuron i in layer l
- $w_{jk}^{(l)}$: weights controlling function mapping from the k^{th} neuron in layer l to the j^{th} neuron in layer $l+1$
- $\mathrm{W}^{(1)} \in R^{h\times(n+1)}$: the weight matrix that connects the input and hidden layer

Neural Network



1st Layer: Input layer with $n = 3$ input units (not counting bias)

2nd Layer: Hidden layer with $h = 3$ hidden units, not counting bias

3rd Layer: Output layer

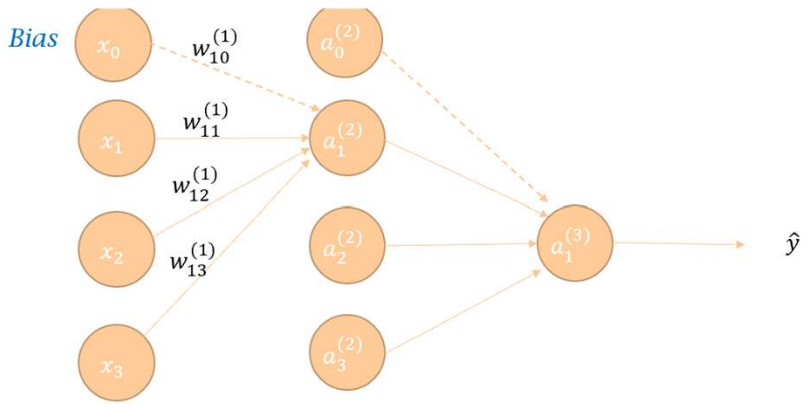
Neural Network

- Activations:

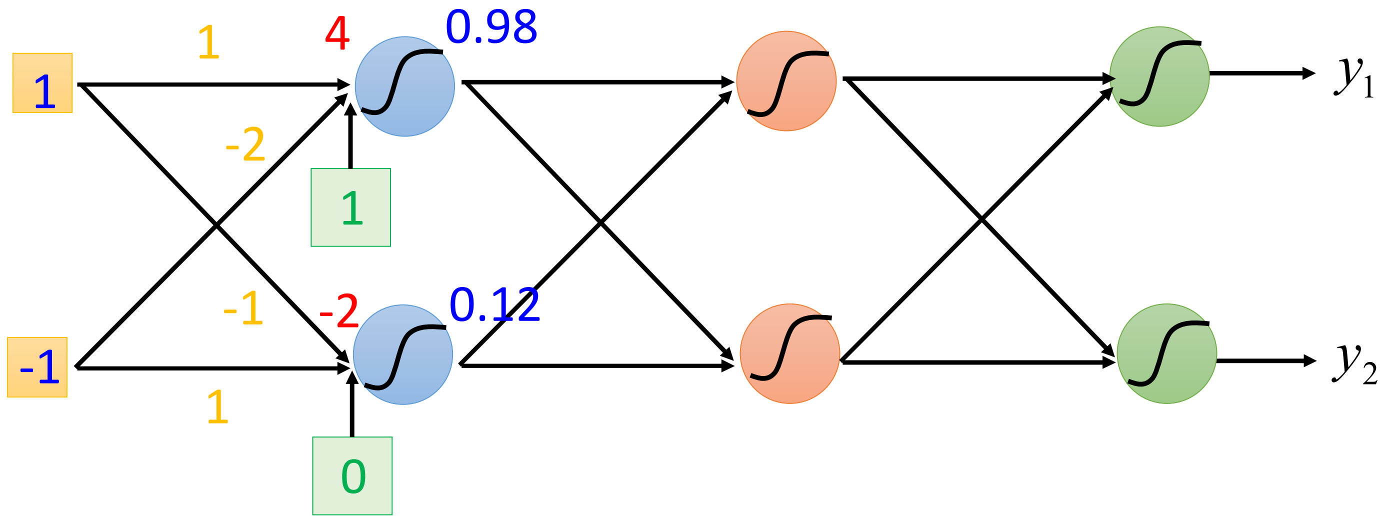
$$\boldsymbol{a_1^{(2)} = \sigma(w_{10}^{(1)} x_0 + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3)}$$

$$a_2^{(2)} = \boldsymbol{\sigma}(w_{20}^{(1)} x_0 + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3)$$

$$a_3^{(2)} = \boldsymbol{\sigma}(w_{30}^{(1)} x_0 + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3)$$

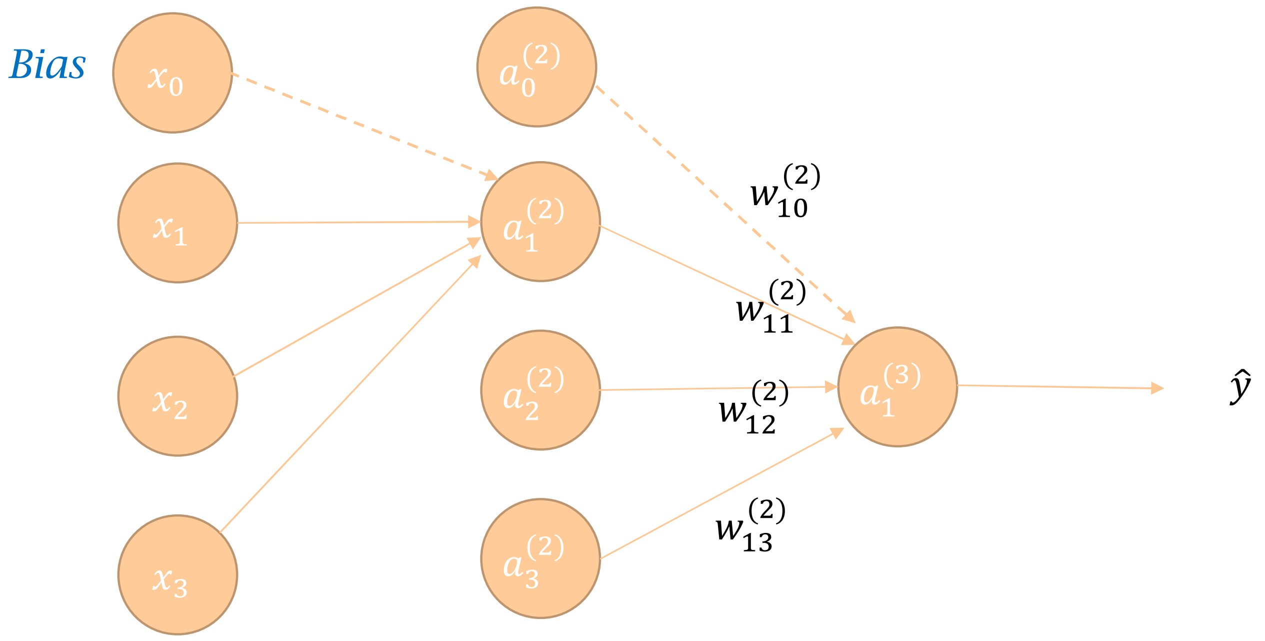


Neural Network: matrix operation



$$\underbrace{\sigma\left(\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}} = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



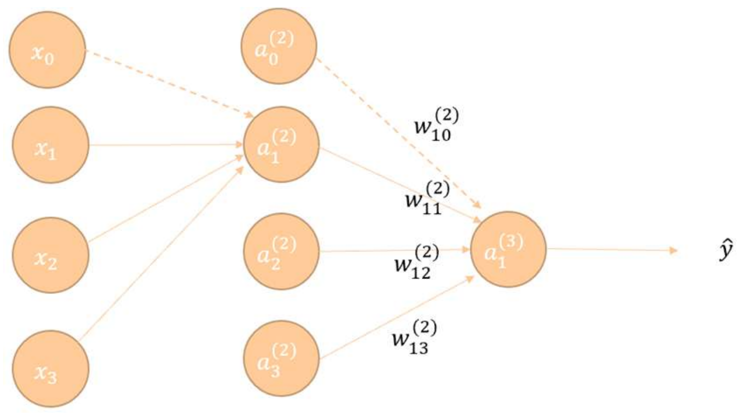
1st Layer:
Input layer with $n = 3$ input units (not counting bias)

2nd Layer: Hidden layer with $h = 3$ hidden units, not counting bias

3rd Layer:
Output layer

- Output activation:

$$\boldsymbol{a_1^{(3)} = \sigma(w_{10}^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)})}$$



Neural Network: in matrix notation

- The intermediate quantity in the activations of the hidden layers:

$$z_1^{(2)} = w_{10}^{(1)} x_0 + w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3$$
$$z_2^{(2)} = w_{20}^{(1)} x_0 + w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3$$
$$z_3^{(2)} = w_{30}^{(1)} x_0 + w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3$$

$$\underbrace{\begin{pmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{pmatrix}}_{z^{(2)}} = \underbrace{\begin{pmatrix} w_{10}^{(1)} & w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{30}^{(1)} & w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{pmatrix}}_{\mathrm{W}^{(1)}} \underbrace{\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{x}$$

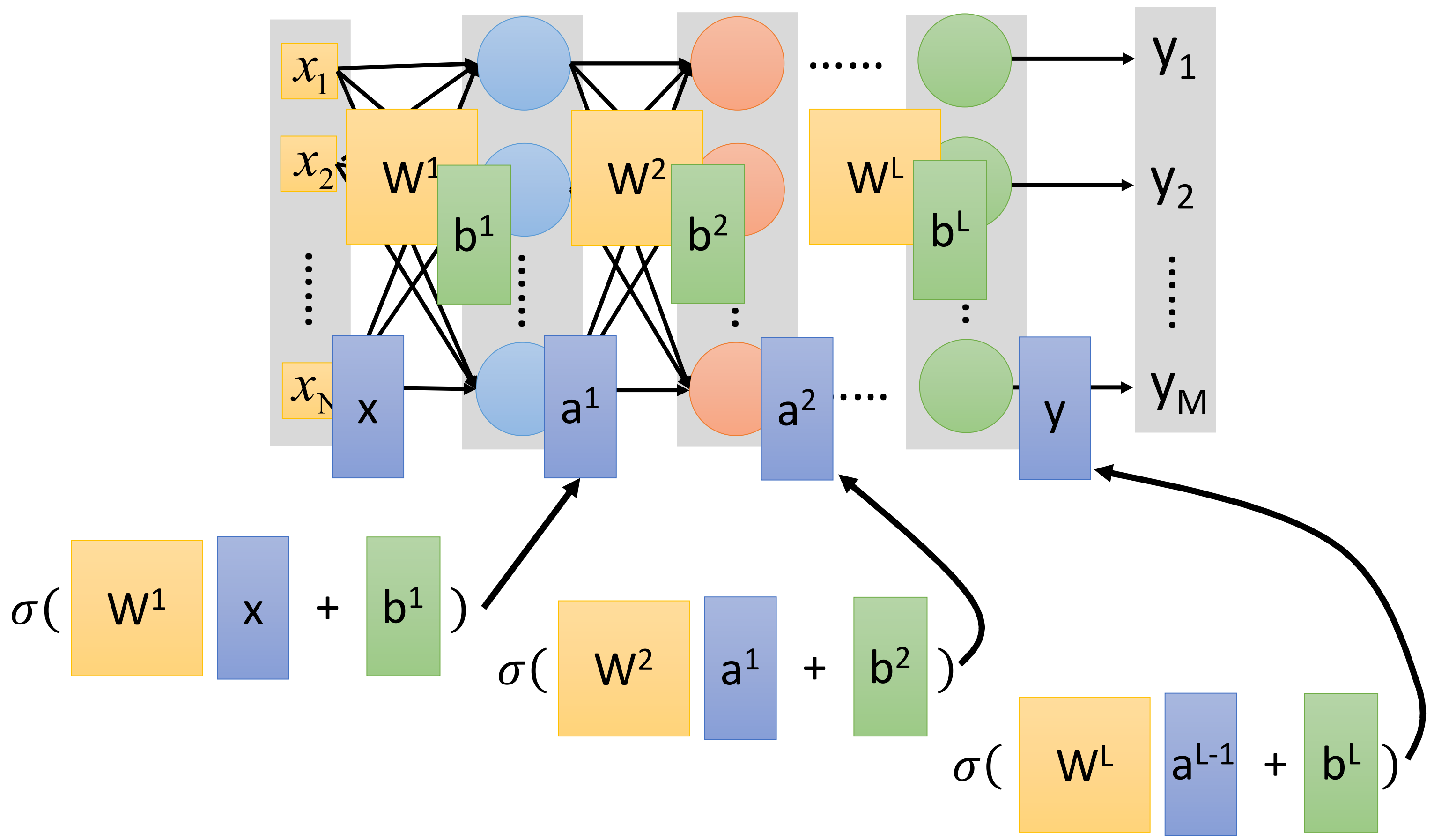
Neural Network

- Activations:

$$a_1^{(2)} = \boldsymbol{\sigma}(z_1^{(2)})$$
$$a_2^{(2)} = \boldsymbol{\sigma}(z_2^{(2)})$$
$$a_3^{(2)} = \boldsymbol{\sigma}(z_3^{(2)})$$

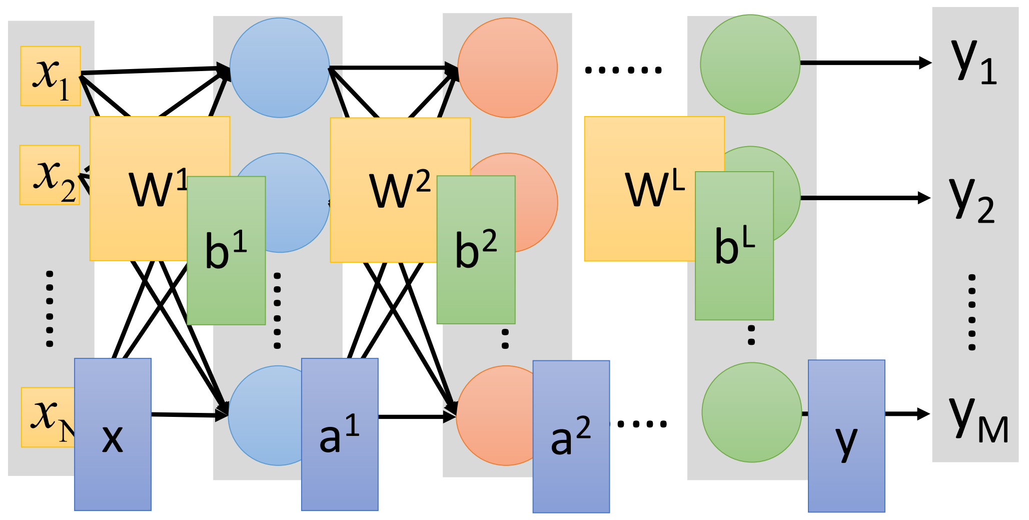
$$a_1^{(3)} = \boldsymbol{\sigma}(w_{10}^{(2)} a_0^{(2)} + w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)})$$

Neural Network: matrix operation (optional)



x_1
x_2
x_N
x
W^1
b^1
a^1
W^2
b^2
a^2
W^L
b^L
y
y_1
y_2
y_M
$\sigma($ W^1 x + b^1 $)$
$\sigma($ W^2 a^1 + b^2 $)$
$\sigma($ W^L a^{L-1} + b^L $)$

Neural Network: matrix operation (optional)

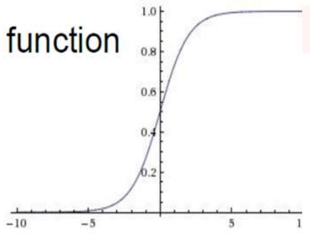


$$y = f(x)$$

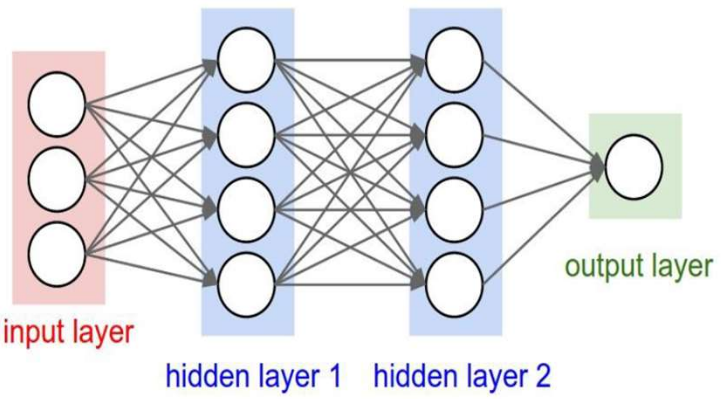
Using parallel computing techniques to speed up matrix operation

$$= \sigma\left(W^L \cdots \sigma\left(W^2 \, \sigma\left(W^1 x + b^1\right) + b^2\right) \cdots + b^L\right)$$

Sigmoid function



A Simple Neural Network

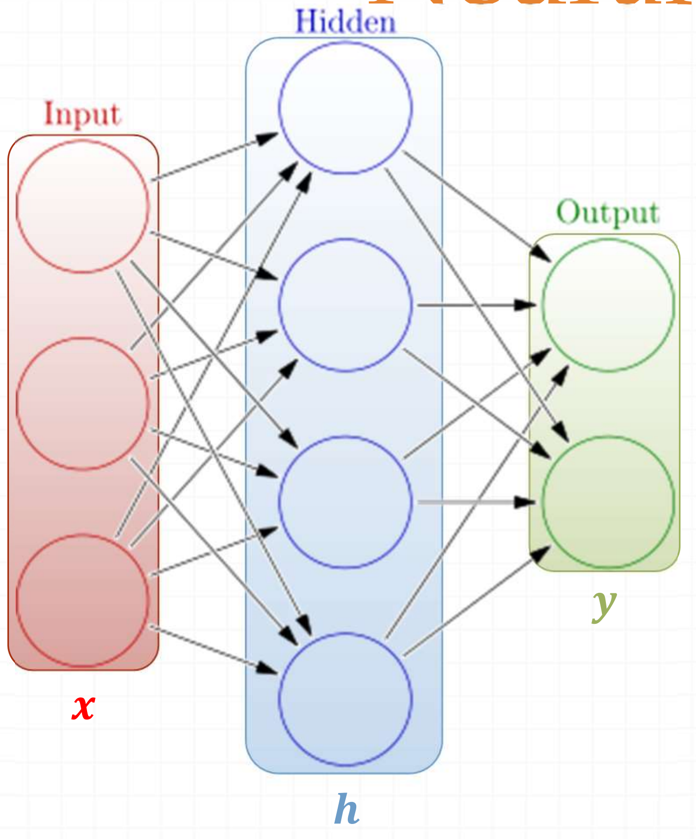


W1, b1, W2, b2, W3, b3 are network parameters that need to be learned.

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

(Python codes on this slide are optional!)

Parameters of Artificial Neural Network



Weights

$$h = \boldsymbol{\sigma}(W_1 x + b_1)$$

$$y = \boldsymbol{\sigma}(W_2 h + b_2)$$

Activation functions

How do we train?

4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases

26 learnable **parameters**

Demo

Basic idea of training: learning by trial-and-error

Continuous process of:

- **Trial:**

  Processing an input to produce an output (In terms of ANN: Compute the output function of a given input)

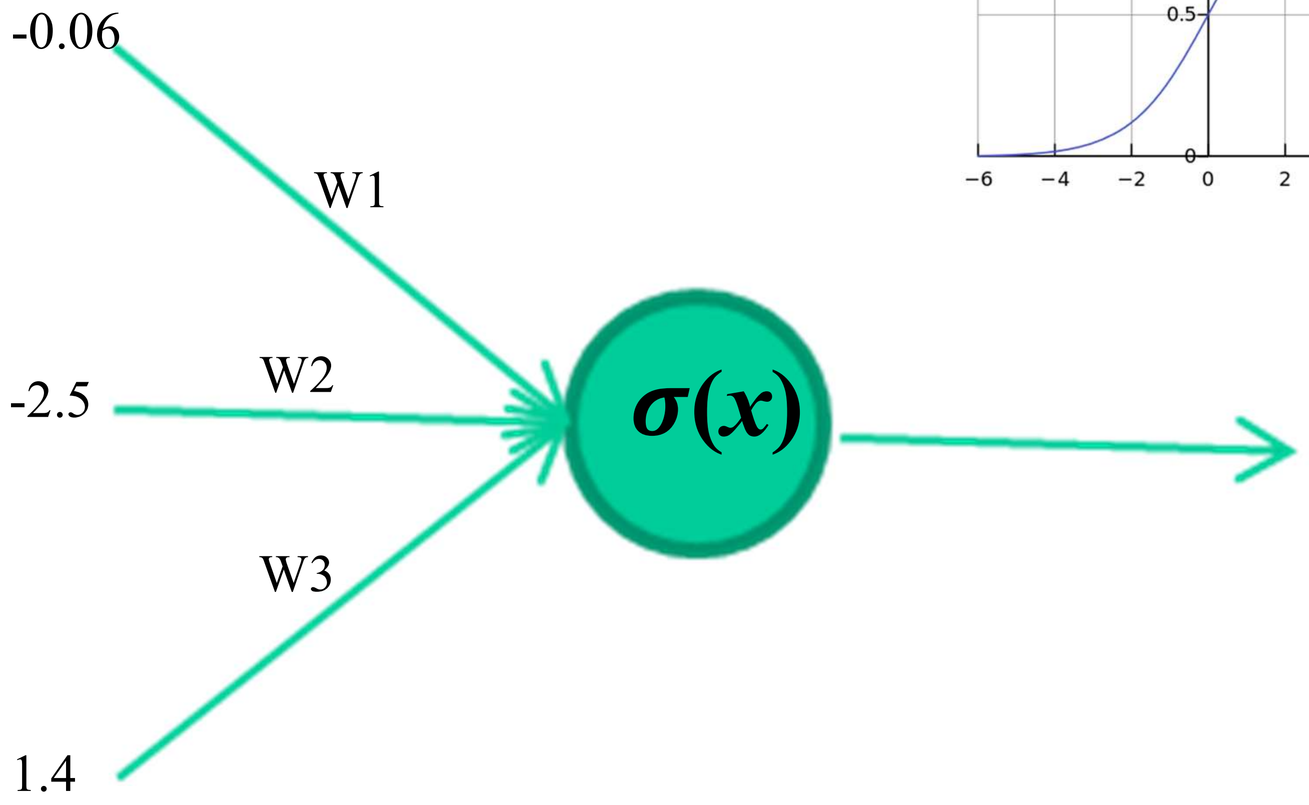
- **Evaluate:**

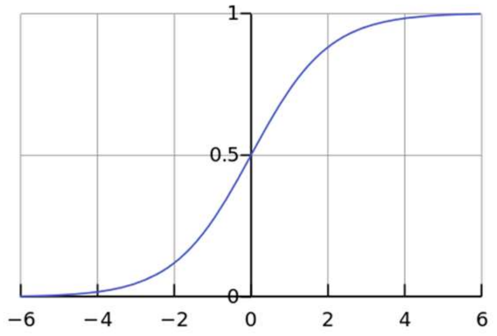
  Evaluating this output by comparing the actual output with the expected output.

- **Adjust:**

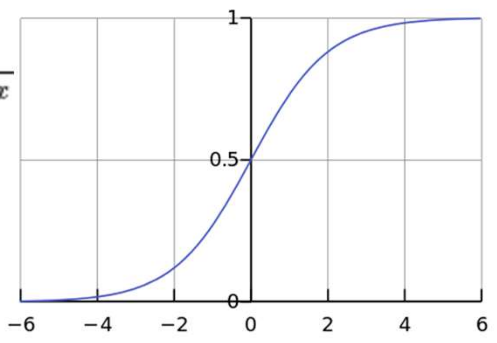
  Adjust the *weights*.

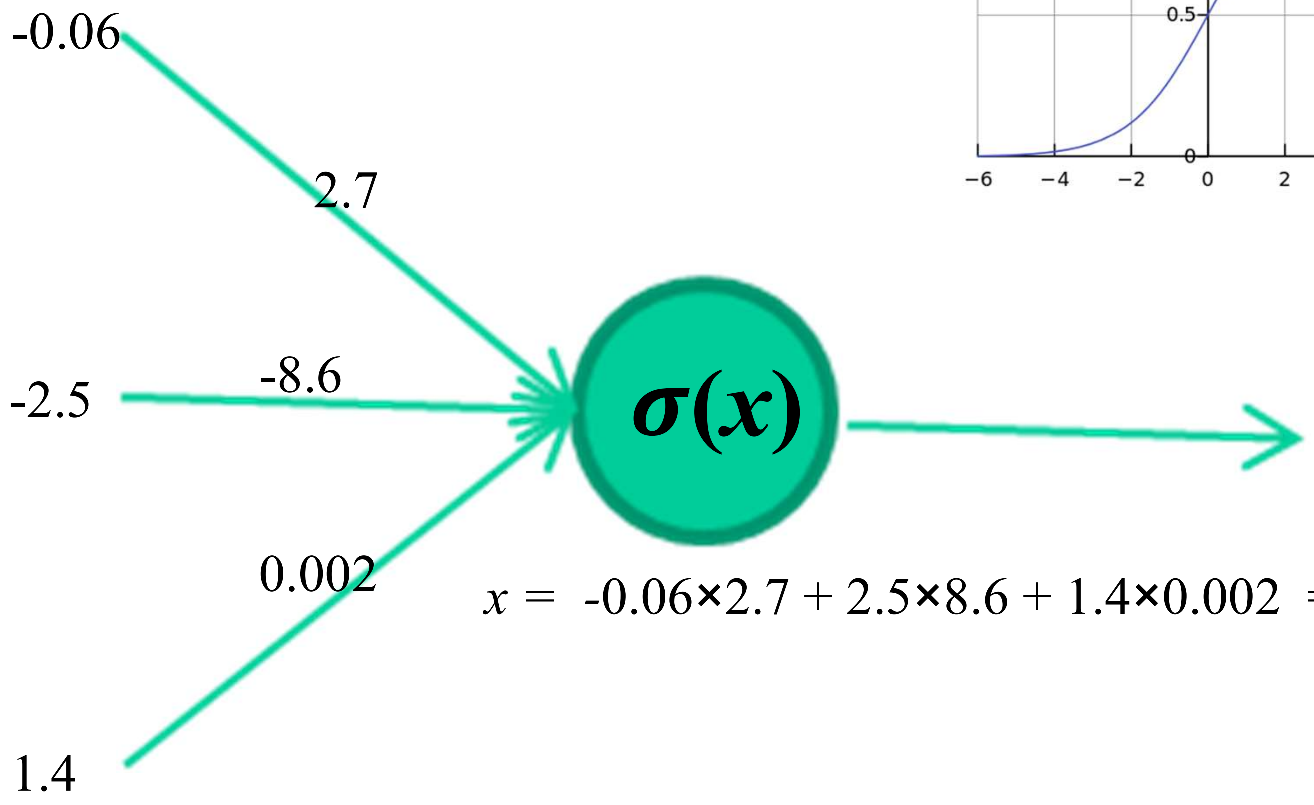
Some insights on weights updating for NN


-0.06
-2.5
1.4
W1
W2
W3
$\sigma(x)$


1
0.5
0
−6
−4
−2
0
2
4
6

$$f(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

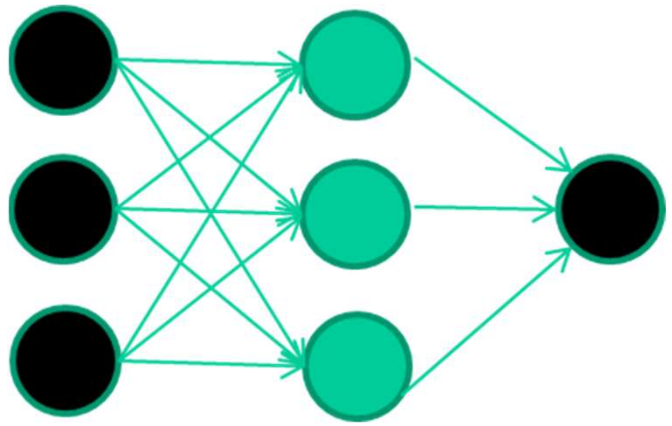




$x = -0.06 \times 2.7 + 2.5 \times 8.6 + 1.4 \times 0.002 = 21.34$

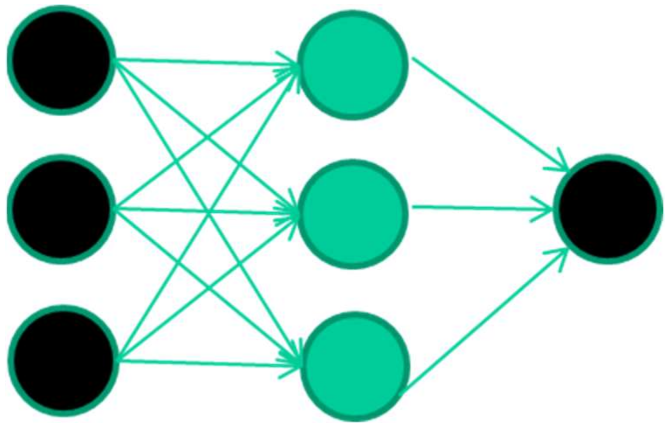
A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Training the neural network

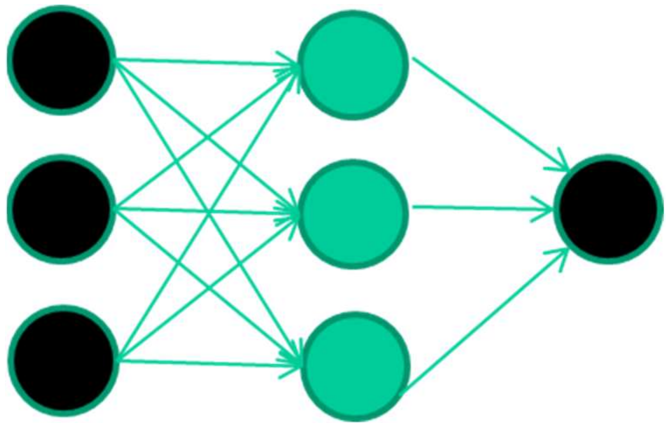
Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

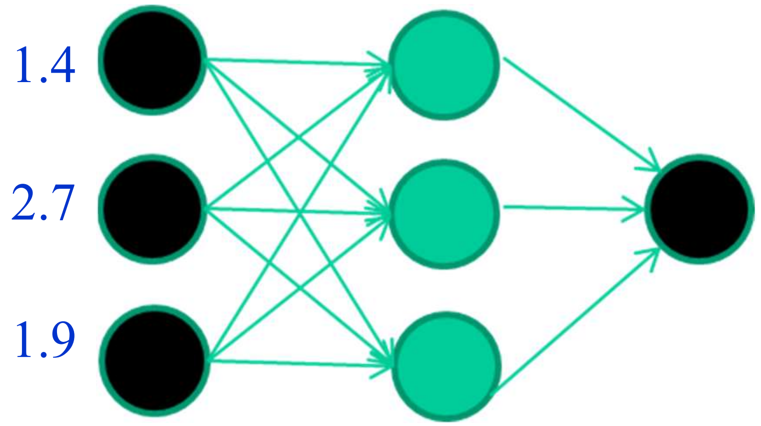
Initialise with random weights



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

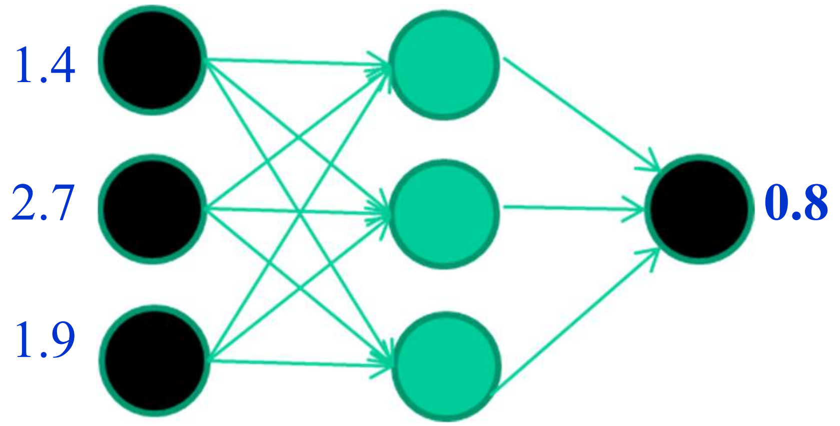
Present a training pattern



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

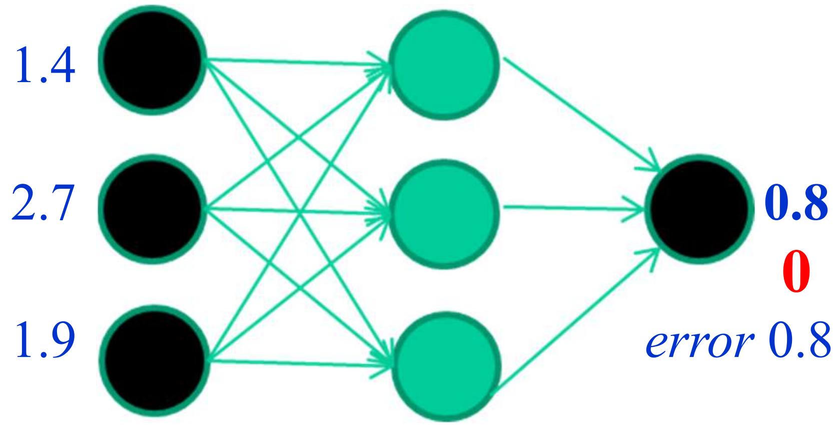
Feed it through to get output



Compare with target output

Training data

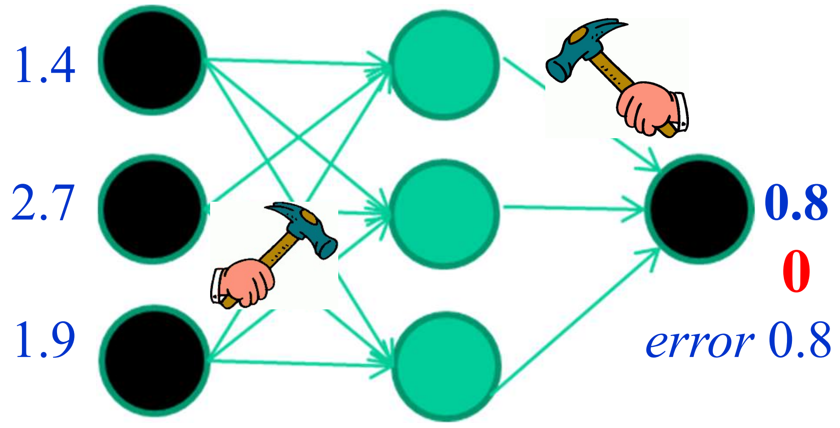
Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

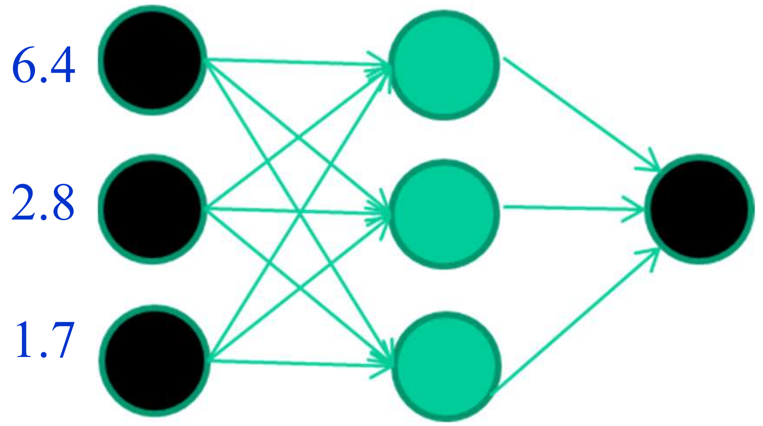
Adjust weights based on error



Present a training pattern

Training data

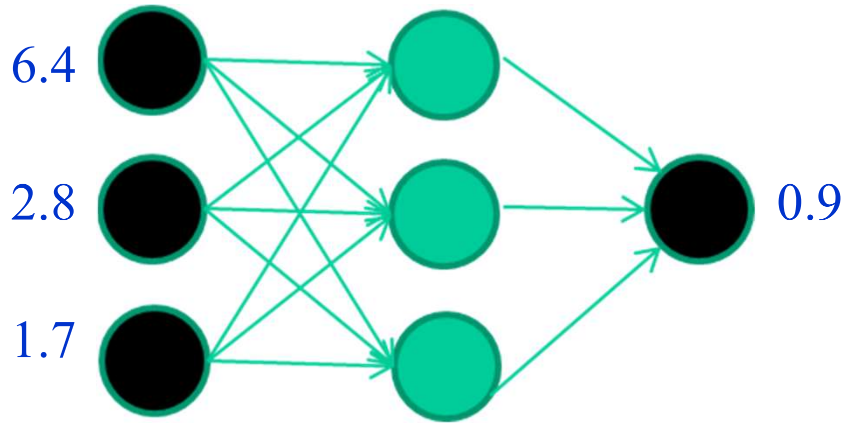
Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

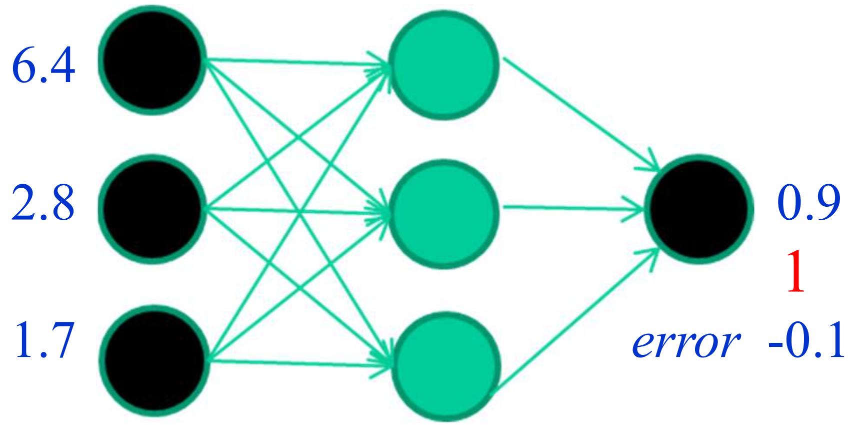
Feed it through to get output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

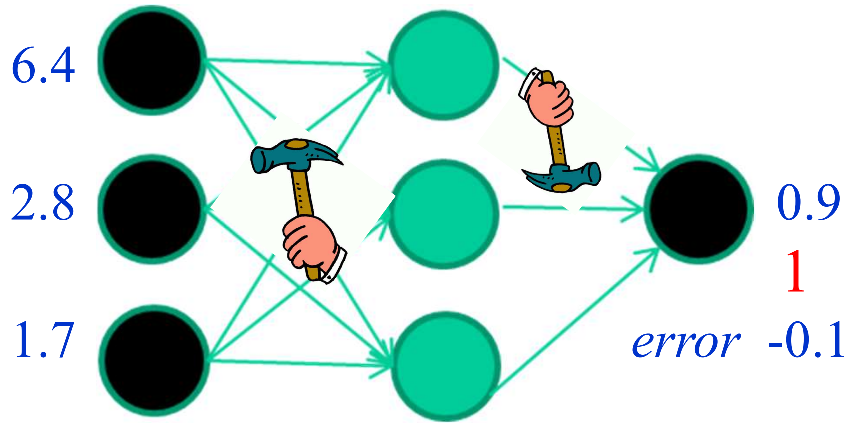
Compare with target output



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

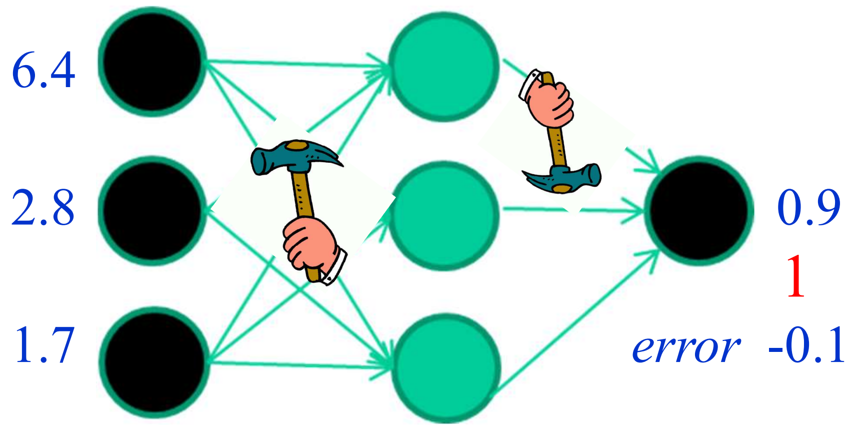
Adjust weights based on error



Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

And so on ….

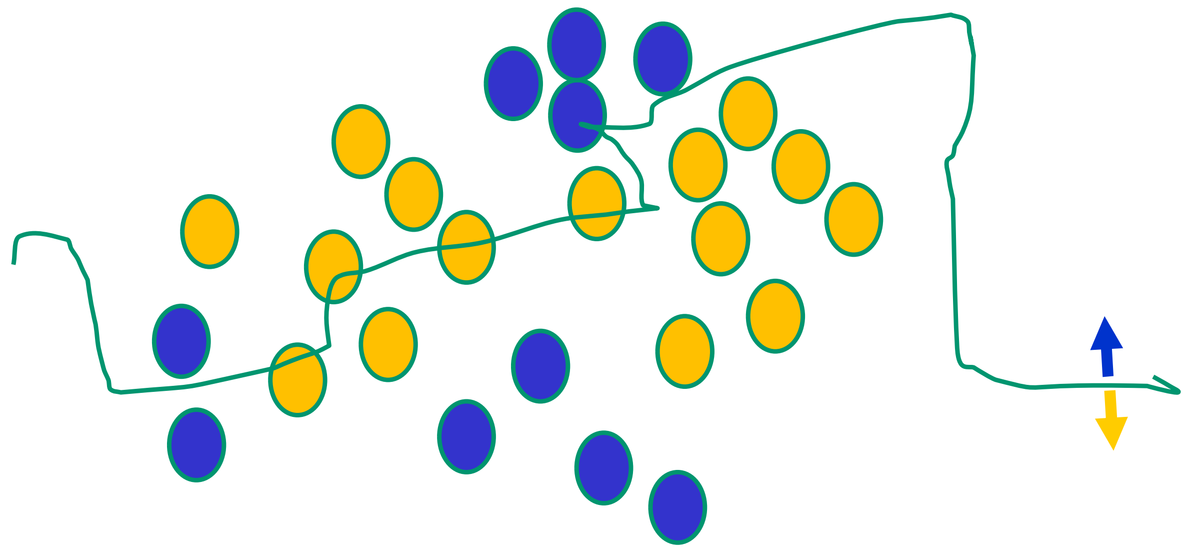


Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

Algorithms for weight adjustment are designed to make changes that will reduce the error

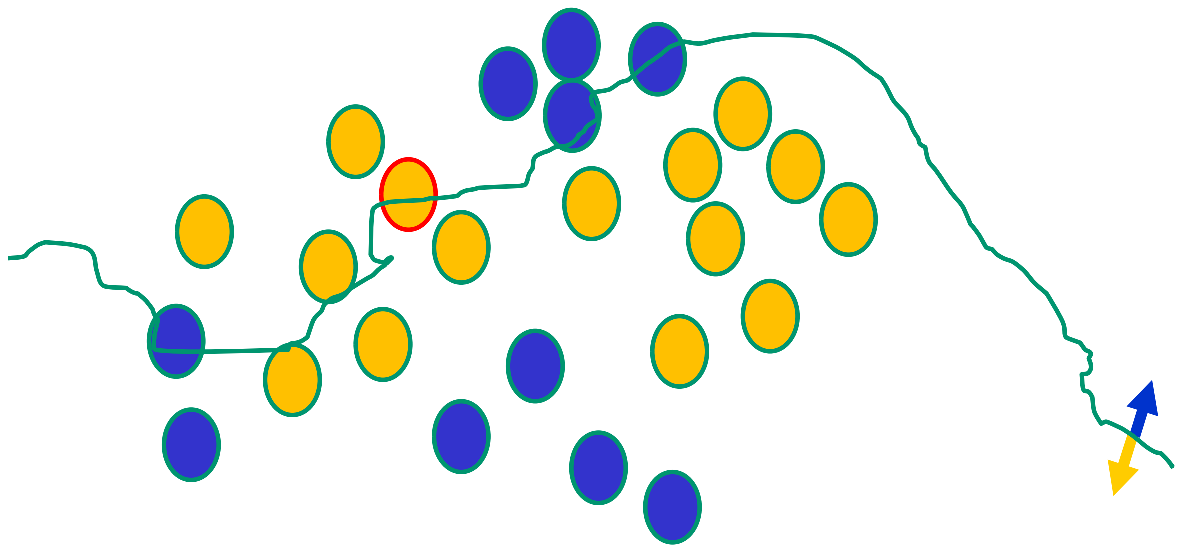
The decision boundary perspective…

Initial random weights



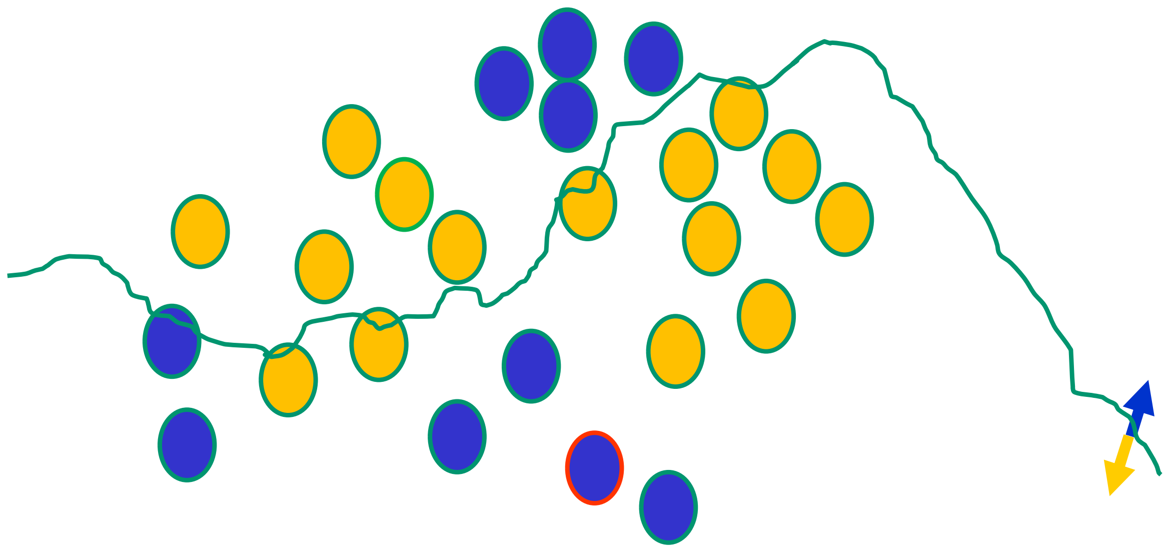
The decision boundary perspective…

Present a training instance / adjust the weights



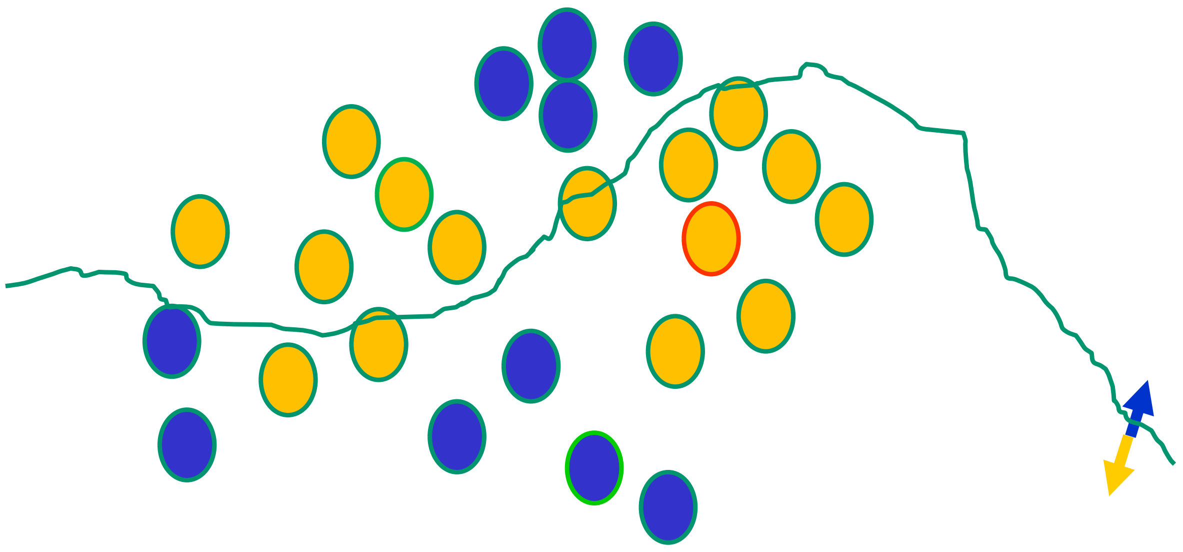
The decision boundary perspective…

Present a training instance / adjust the weights



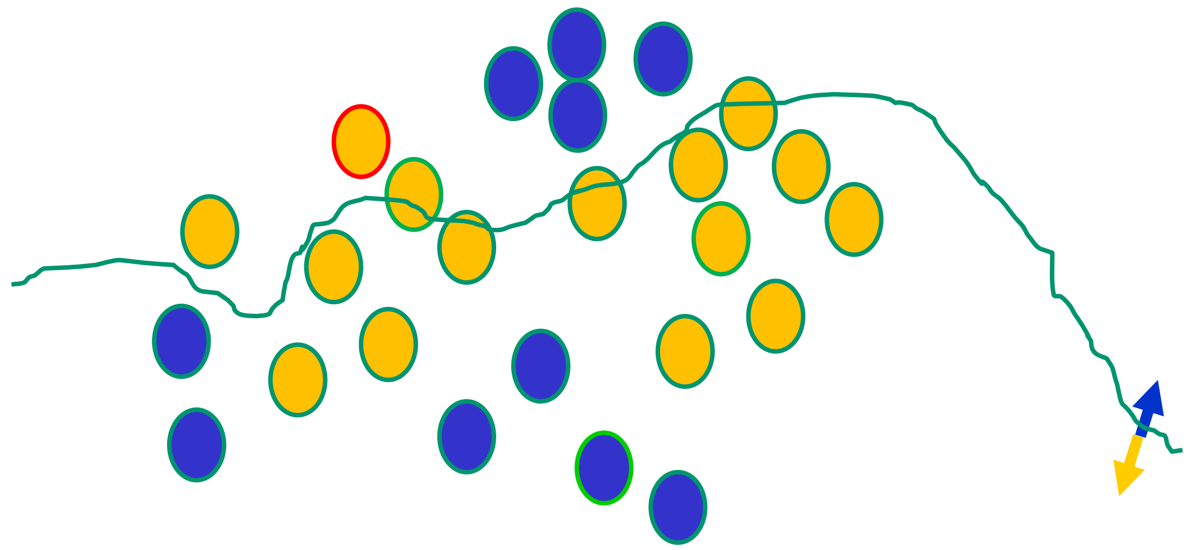
The decision boundary perspective…

Present a training instance / adjust the weights



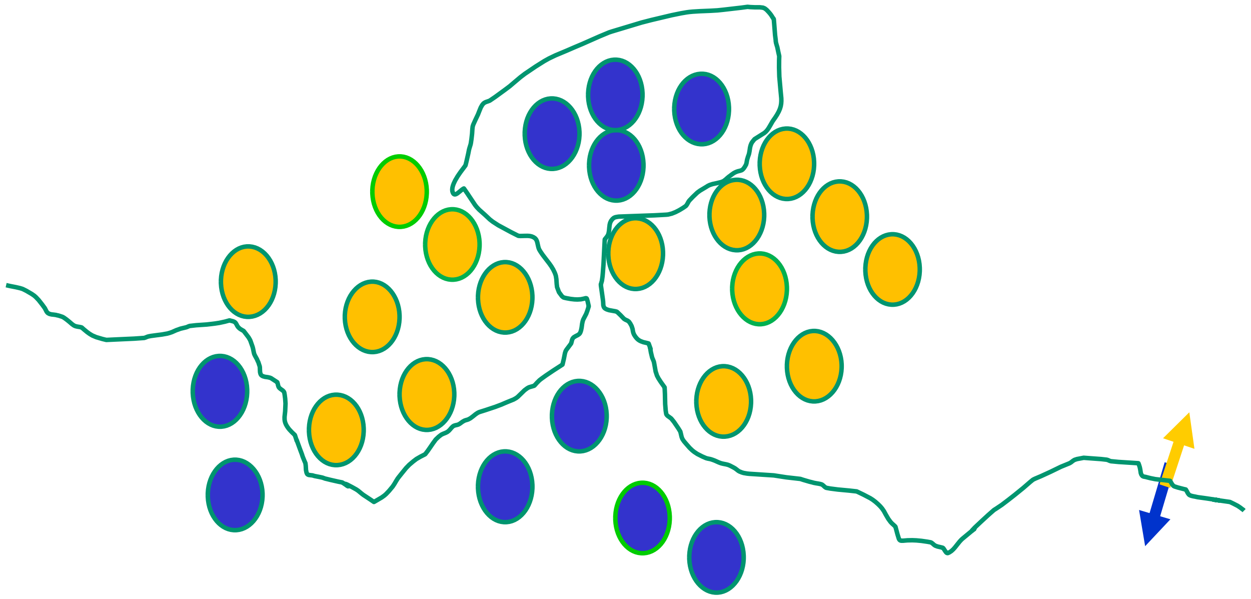
The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

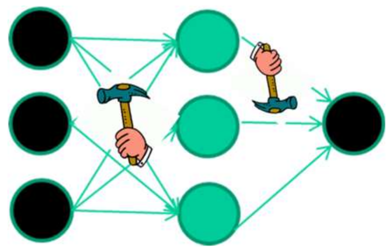
Eventually ….



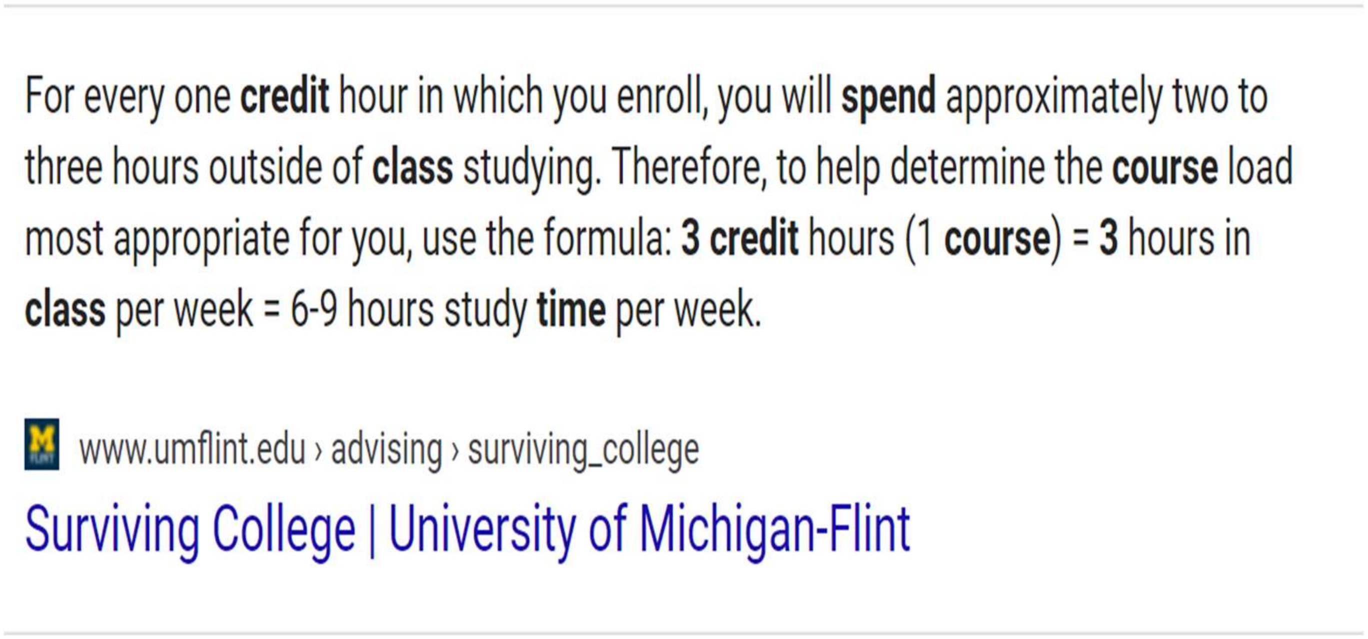
Summary of Insights

- Weight-learning works by making thousands and thousands of tiny adjustments, each making the network do better at the most recent pattern, but perhaps a little worse on many others

- Eventually this tends to be good enough to learn effective classifiers for many real applications



Tips for studying


For every one **credit** hour in which you enroll, you will **spend** approximately two to three hours outside of **class** studying. Therefore, to help determine the **course** load most appropriate for you, use the formula: **3 credit** hours (1 **course**) = **3** hours in **class** per week = 6-9 hours study **time** per week.

www.umflint.edu › advising › surviving_college

Surviving College | University of Michigan-Flint