Introduction to Machine Learning

Instructor: Warasinee Chaisangmongkon, PhD

Day 3-4

- Day 3 Morning : Nearest-Neighbor Methods, Feature Selection
- Day 3 Afternoon : Recommender System, Unsupervised Learning
- Day 4 Morning : Neural Network
- Day 4 Afternoon : Advanced Concepts in Machine Learning

Neural Network

Regression and Logistic Regression Recap

Get everyone on the same page with calculus and logistic regression.

Introduction to Neural Network

Understand the principles behind the best machine learning model on earth (currently).

Single Neuron and Simple Network

Understand the how single neurons make decision.

Training Neural Network

Understand how neural network is trained with gradient descent algorithms.

Neural Network Advance

Learn more complex neural network, with code example.

Scikit Learn Neural Network and Lab

Master Scikit Learn neural network library and complete some coding exercise.

Machine Learning Recap

i	Size (m^2)	Price (Mbaht)
1	50	1.4
2	128	2.6
3	24	0.8
4	78	1.2
i	…	…

x: feature

y: target or label

i: sample index

Machine Learning Recap

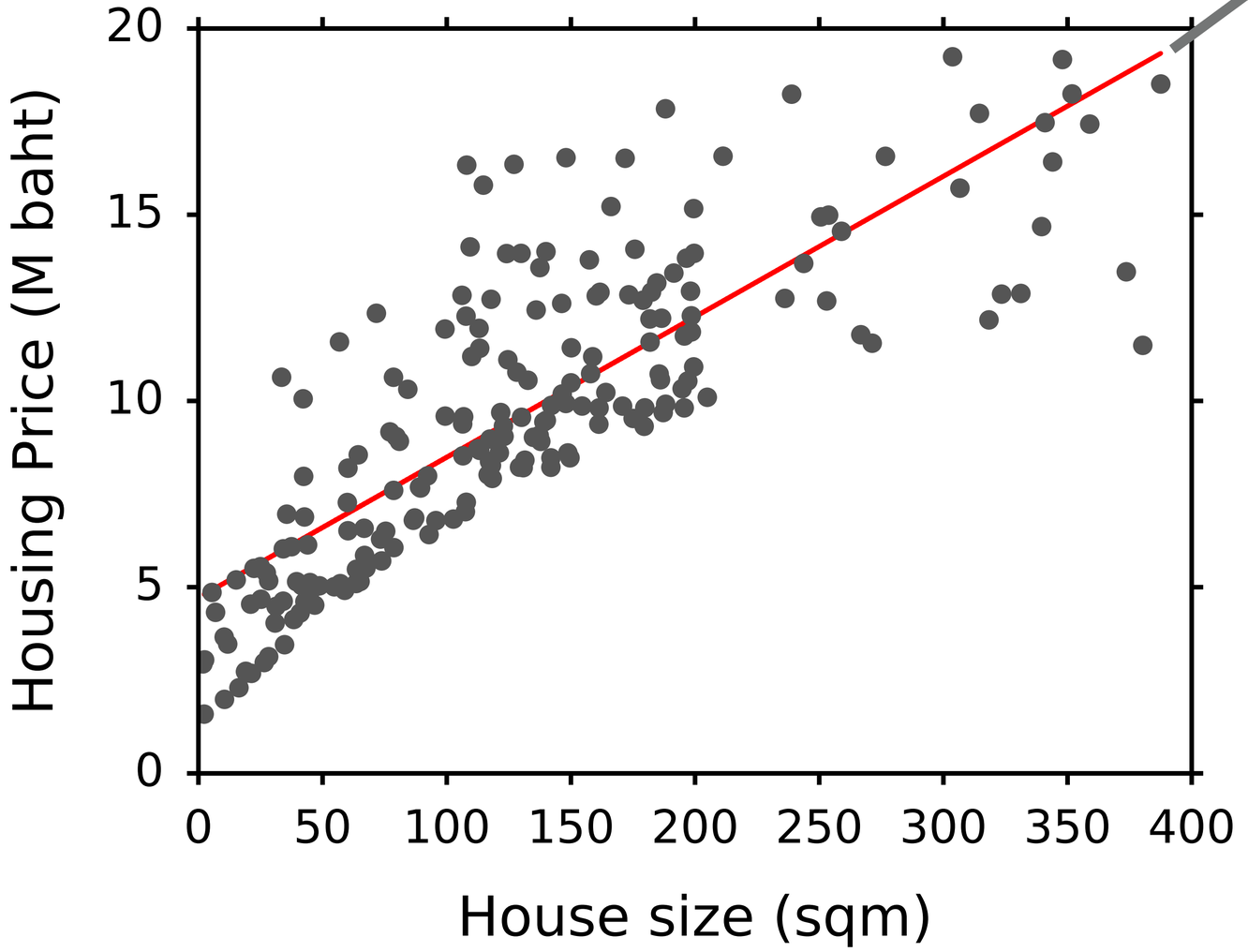
i	Size x	Price y
1	50 $= x_1$	1.4 $= y_1$
2	128 $= x_2$	2.6 $= y_2$
3	24 $= x_3$	0.8 $= y_3$
4	78 $= x_4$	1.2 $= y_4$
i	... $= x_i$	... $= y_i$

x: feature

y: target or label

i: sample index

Linear Regression Model



$$h(x) = \theta_0 + \theta_1 x$$

parameters of the model

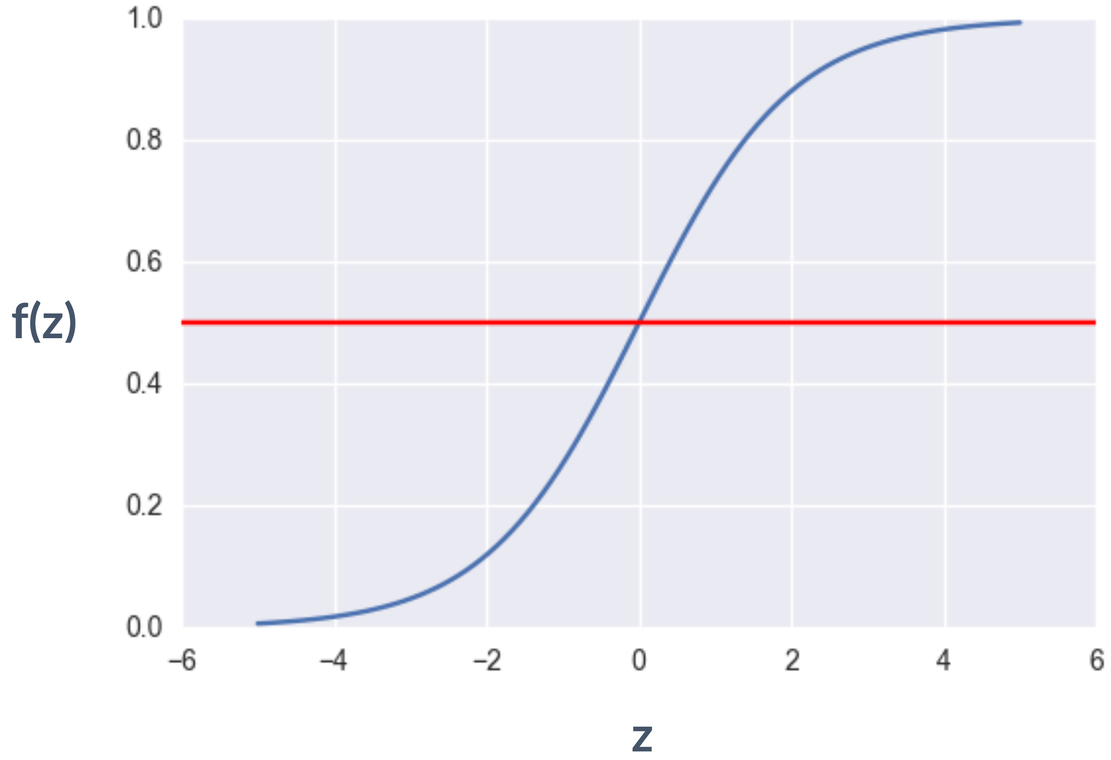
$$\theta_0 = 0.038$$

$$\theta_1 = 4.717$$

Logistic Regression Model

The Model

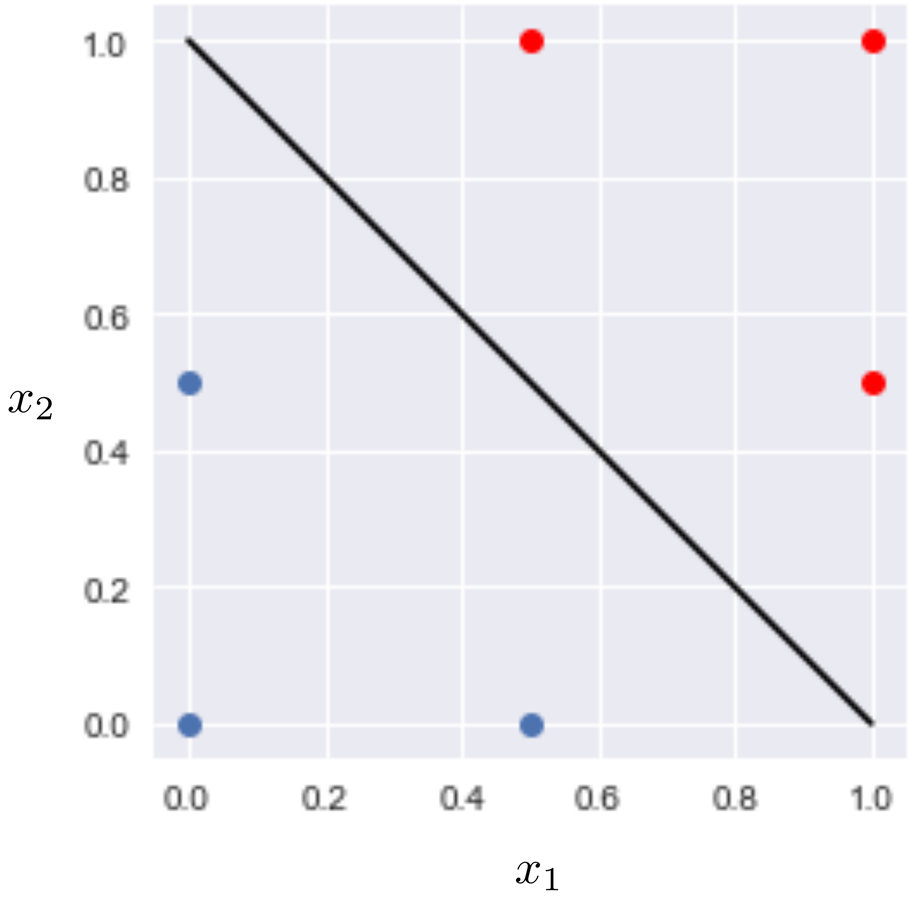
$$h(x) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...)$$



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

y' = 1, if f(z) > 0.5 or z > 0
y' = 0, if f(z) < 0.5 or z < 0

Logistic Regression Model



model: $h(x) = f(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = f(z)$

parameter: $\theta_0 = -1$

$\theta_1 = 1$

$\theta_2 = 1$

Logistic regression model draws a boundary at z=0

$$1 * x_1 + 1 * x_2 - 1 = 0$$

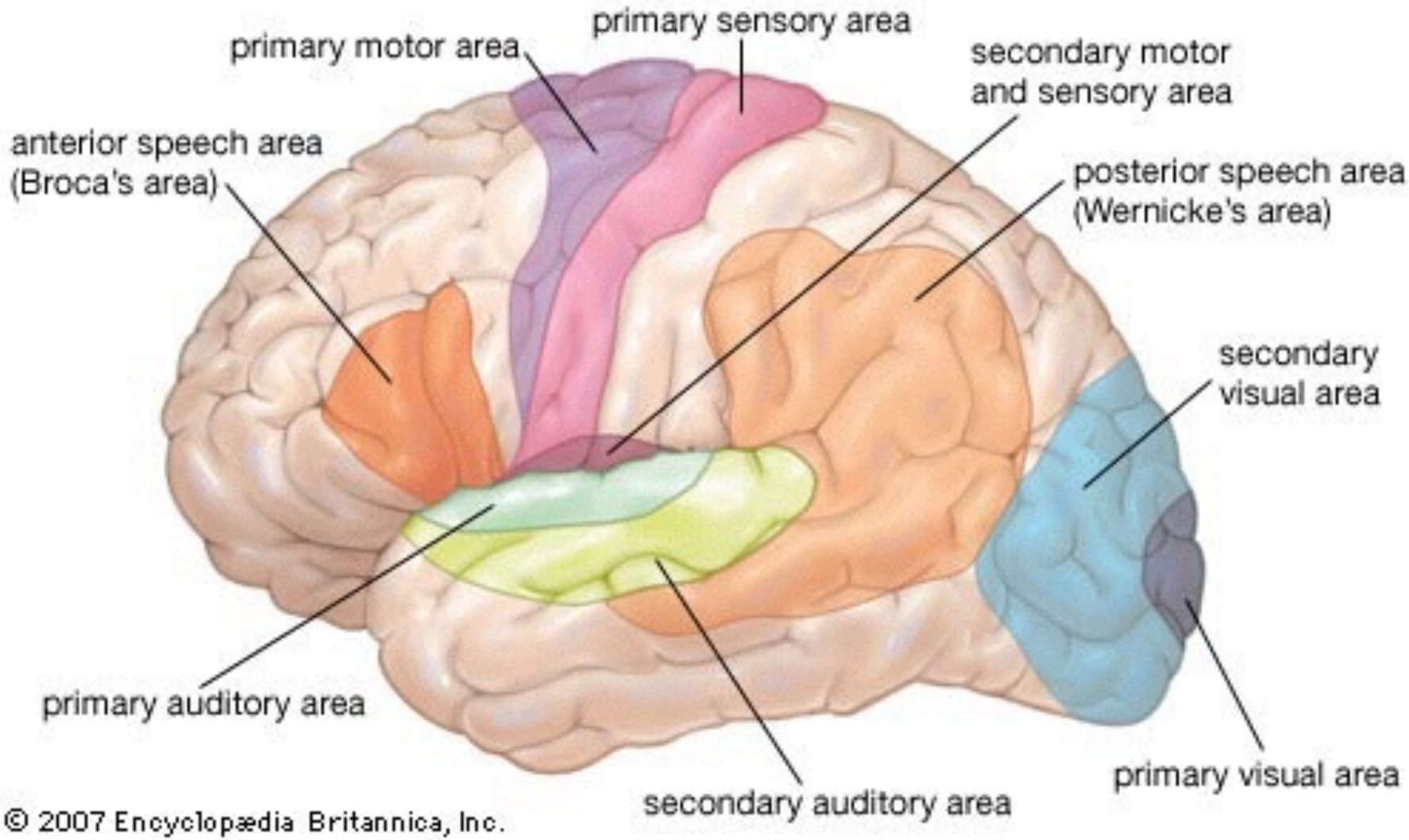
$$x_2 = -x_1 + 1$$

Introduction to Neural Network

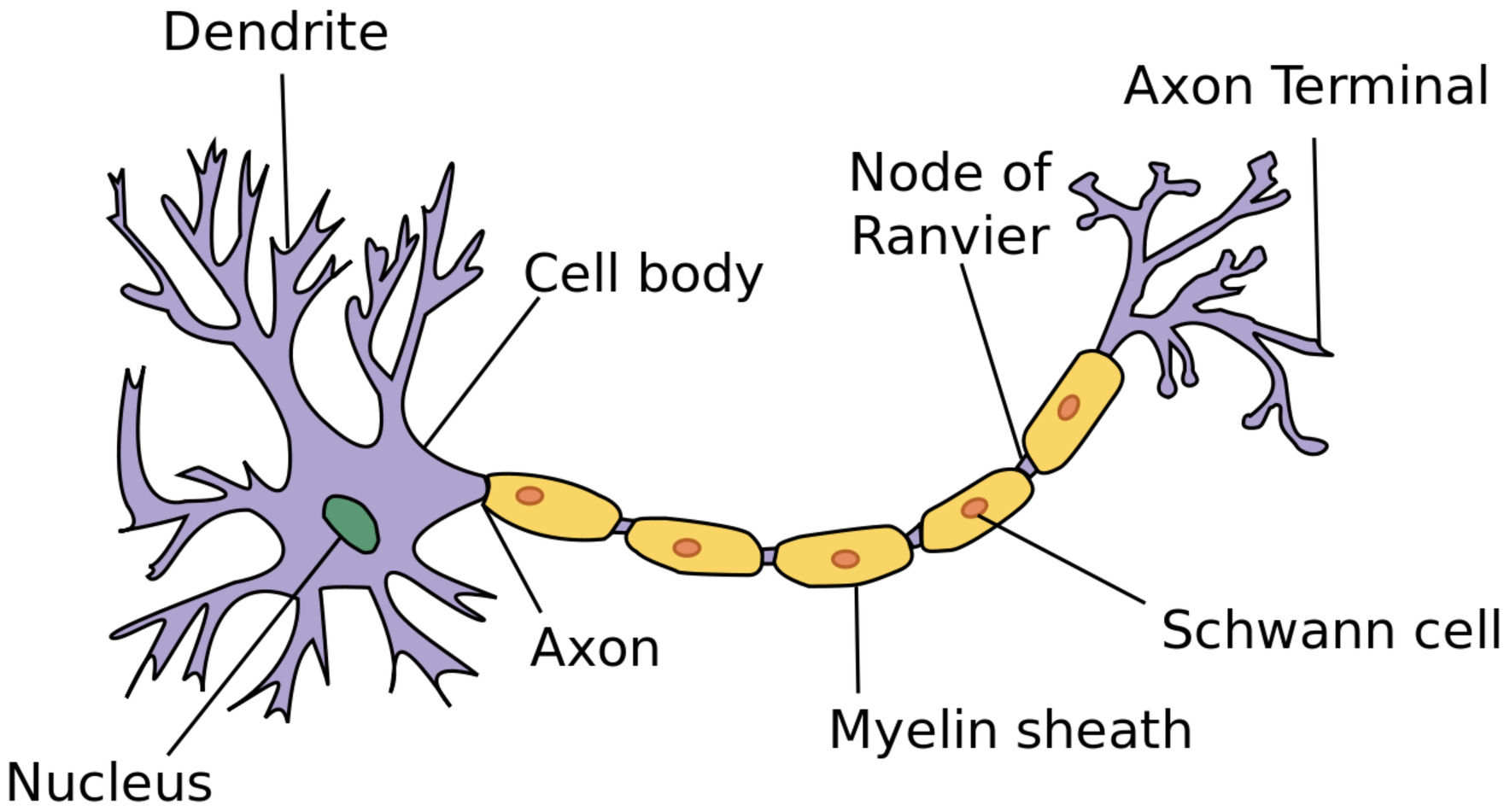
- Real neuron
- Artificial neuron
- Two-layer example
- Multi-layer demo
- Training NN
- Gradient descent

NEURAL NETWORK BASICS

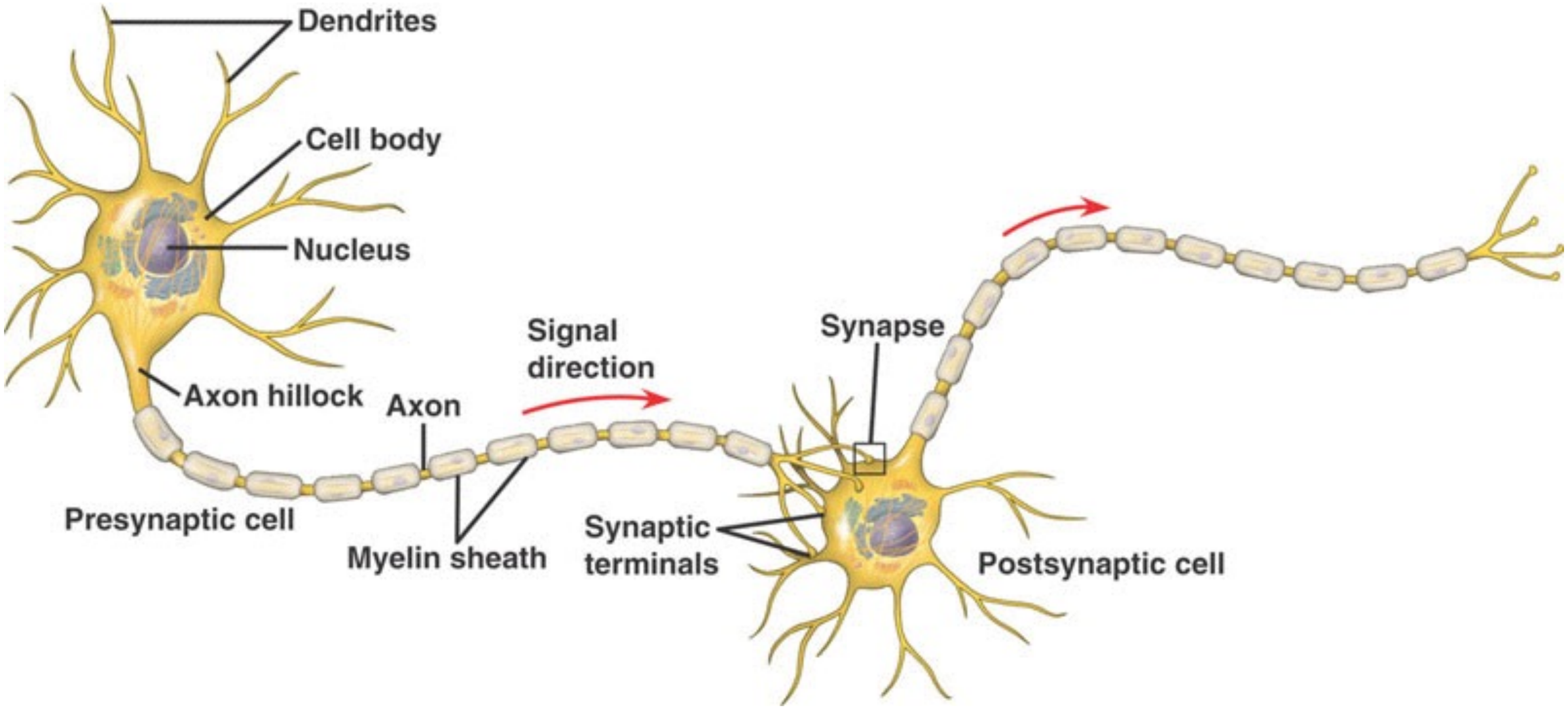
Natural Intelligence = Neural Network



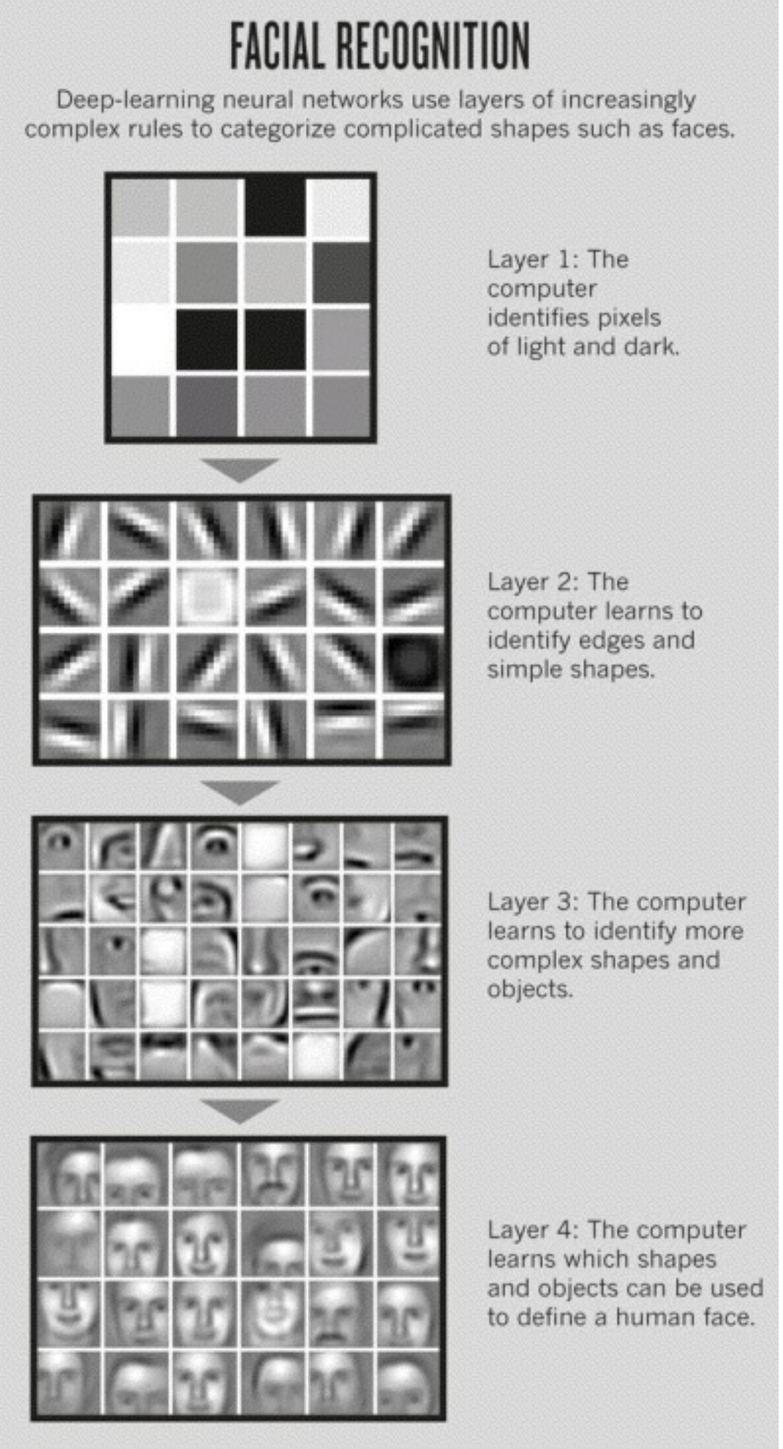
Human Neuron Cell



Human Neuron Interaction



Network of neurons can learn anything



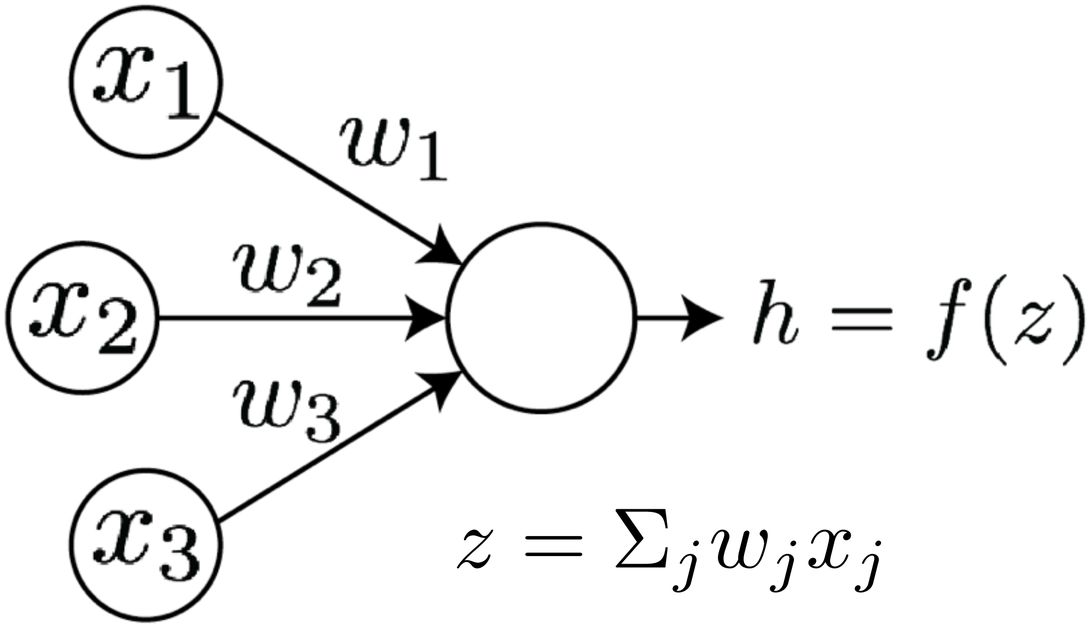
Network of Neurons can Understand Anything

Computer scientist achieved state-of-the-art face recognition machine with neural network model. The network mimics how human brains see images. Human brain break down visual perception into several steps, for example:

- Edge detection
- Shape detection
- Object recognition
- Face identity recognition

Neural network ML models process images successively, in the same fashion as human neural networks.

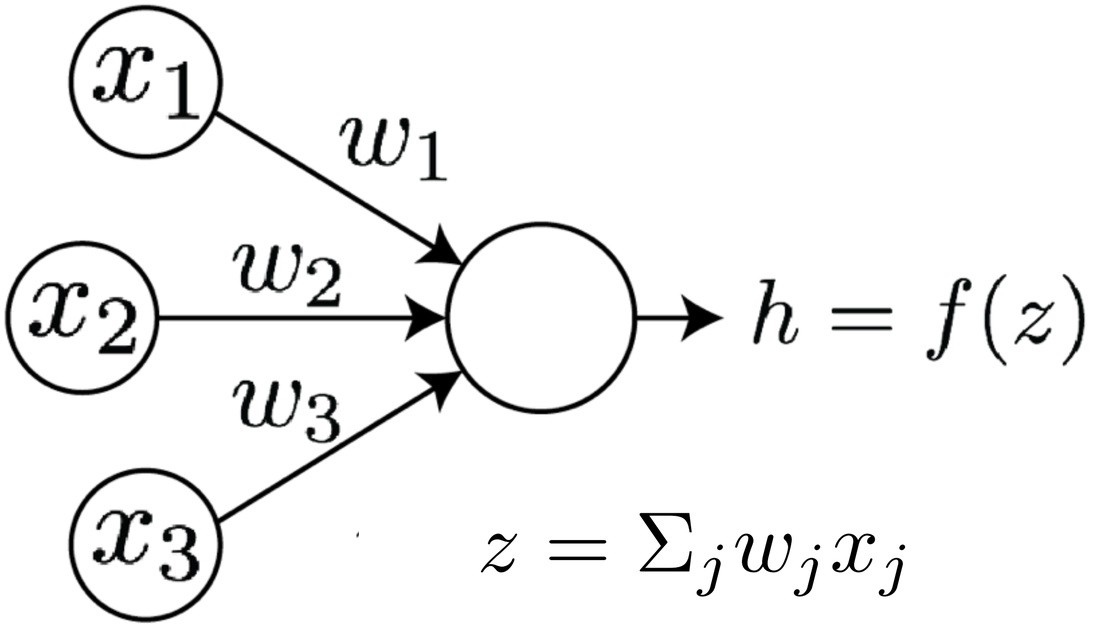
An Artificial Neuron

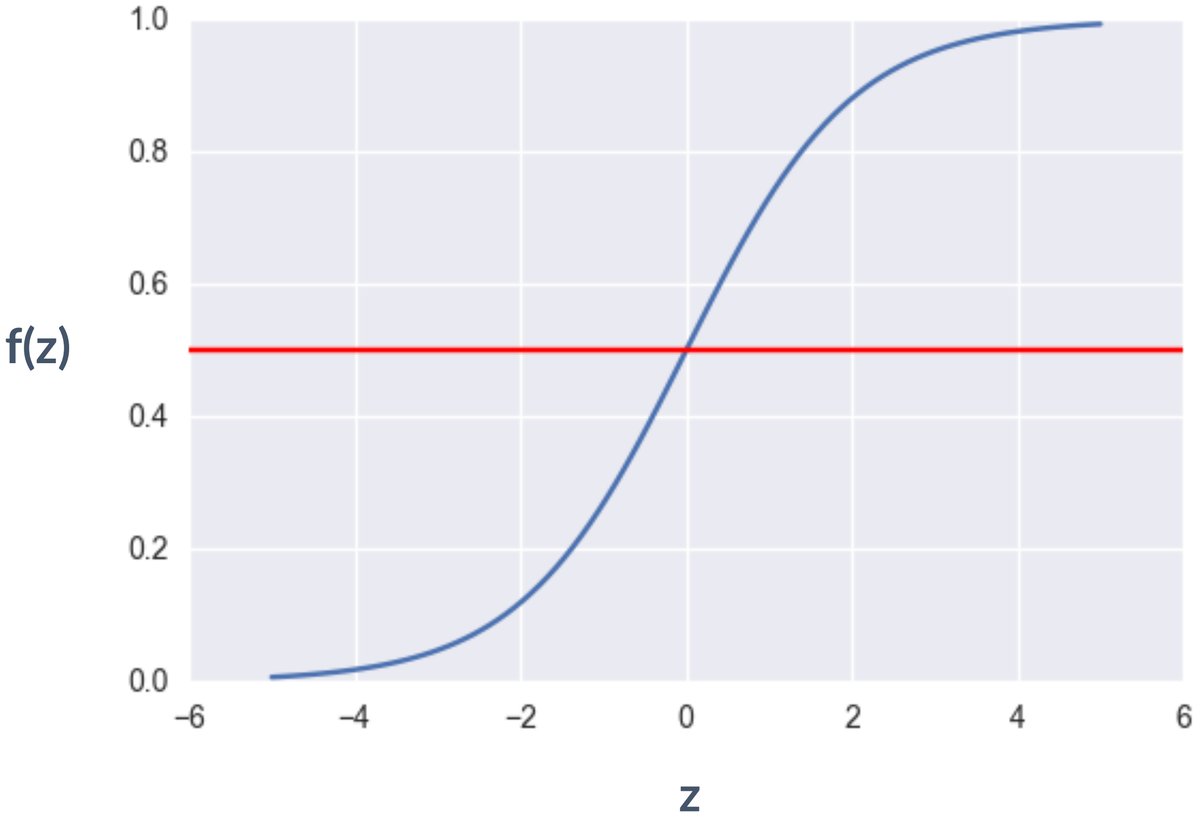


- **x**: features
- **w**: weights (parameters)
- **z**: sum of weights * features (neuron's current)
- **f**: activation function
- **h**: the model

The Activation Function

Signal received by a neuron (z) is passed to an "**Activation Function**" to get output (neural activity).

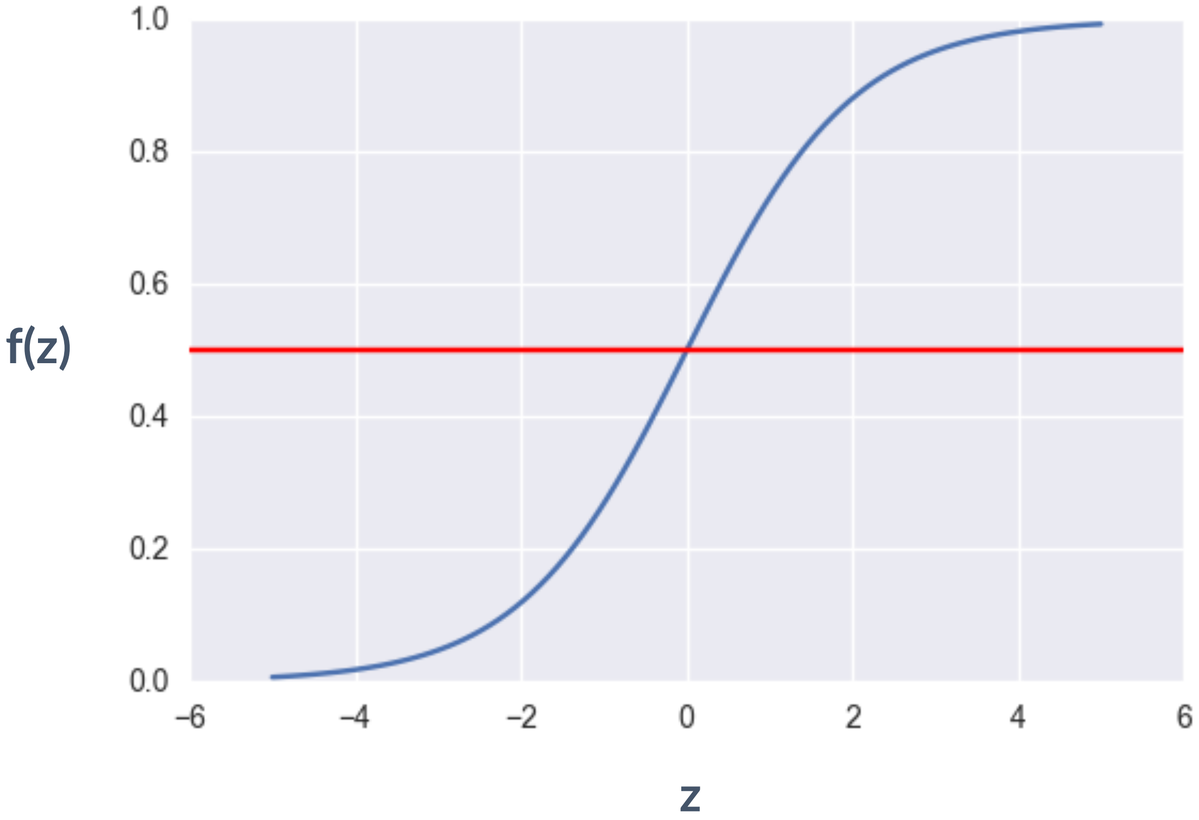




A common activation is **sigmoid** function.

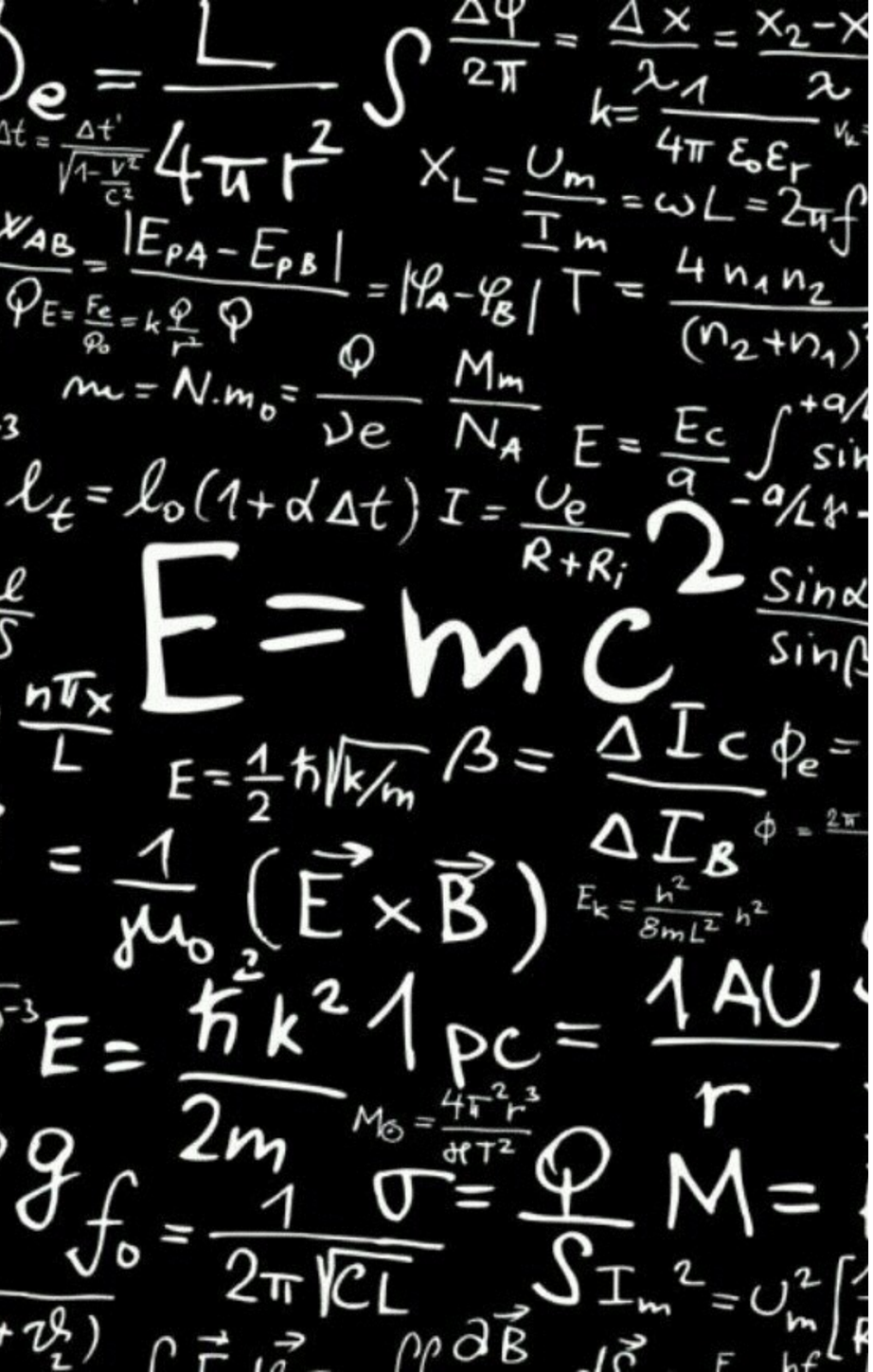
The Activation Function

A common activation function is **sigmoid** or **logistic** function.



A **sigmoid** or **logistic** functional form:

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



SINGLE NEURON QUIZ

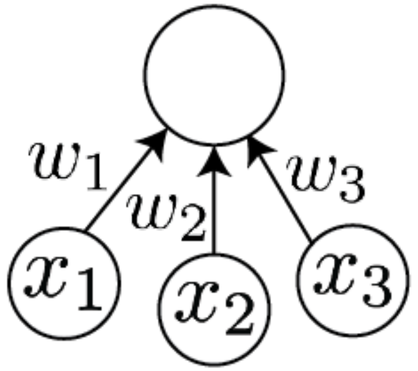
Single Neuron Quiz

Find the current input (z) into the top neuron. What class (y') would the neuron predicts for each sample?

$$h = f(\Sigma_j w_j x_j)$$

x1	x2	x3	z	y'
1	0	0		
1	1	0		
1	0	1		
1	1	1		

w1	-1.5
w2	1
w3	1

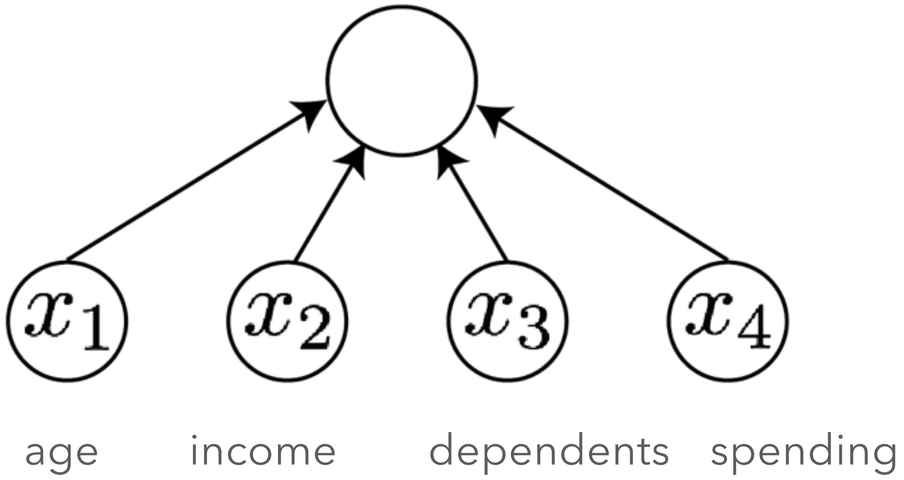


A simple example

Let's look at a simple 2-layer neural network for example:

Suppose we want to predict if a given bank customer will be good or bad loan taker.

$$h = f(\Sigma_j w_j x_j)$$



x1	x2	x3	x4	history
40	50	0	30	1
25	40	2	35	1
18	10	0	12	0
34	22	1	10	1

A simple example

We first need to do preprocessing, such as normalization and standardization.

x1	x2	x3	x4	history
40	50	0	30	1
25	40	2	35	1
18	10	0	12	0
34	22	1	10	1

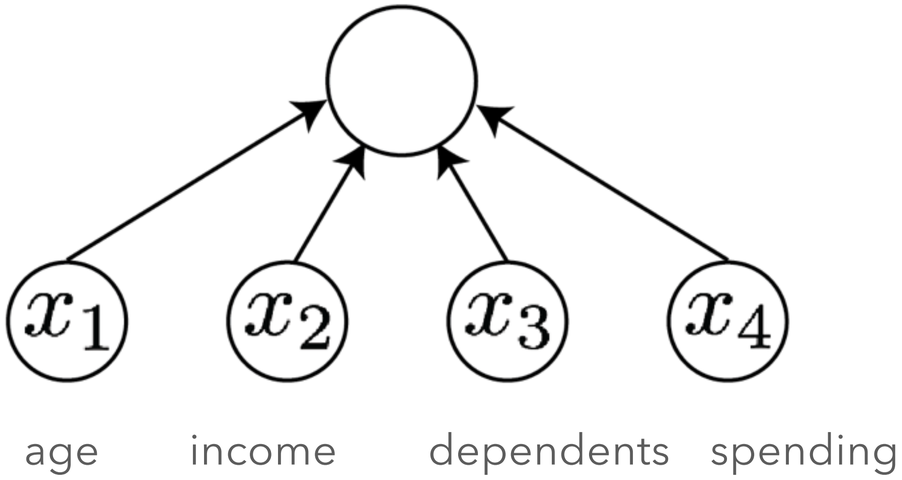
x1	x2	x3	x4	history
0.44	0.63	0	0.6	1
0.28	0.50	0.5	0.7	1
0.20	0.13	0	0.24	0
0.38	0.28	0.25	0.2	1

A simple example

Then fit the neural network to the data.

Suppose after fitting, here are the weight numbers.

$$h = f(\Sigma_j w_j x_j)$$

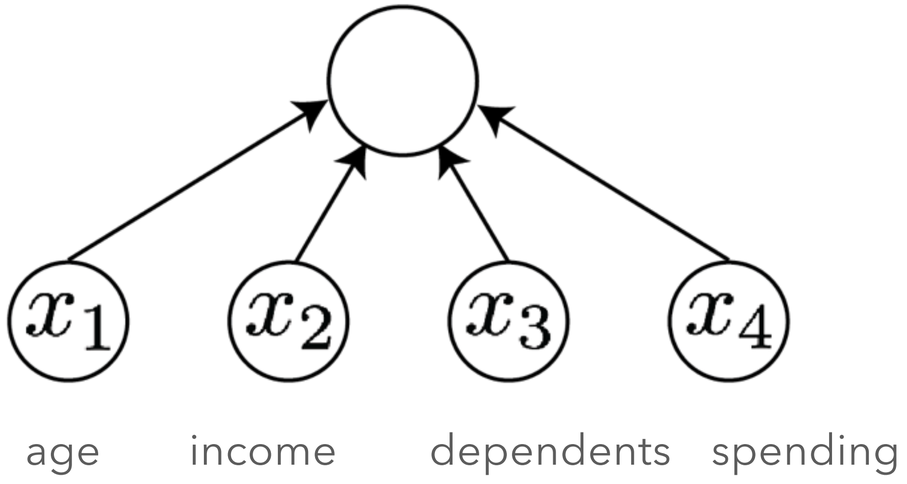


w1	0.7
w2	0.6
w3	-0.1
w4	-0.2

A simple example

Let us make prediction for a single customer…

$$h = f(\Sigma_j w_j x_j)$$



	X	W	X*W
age	0.44	0.7	0.31
income	0.63	0.6	0.38
dependent	0.00	-0.1	0.00
spending	0.60	-0.2	-0.12
		sum:	0.57
		h:	0.64

What the output means

Neural network classification

	X	W	X*W
age	0.44	0.7	0.31
income	0.63	0.6	0.38
dependent	0.00	-0.1	0.00
spending	0.60	-0.2	-0.12
		sum:	0.57
		h:	0.64

h indicates the probability of customer being good

h = 0.64

64% chance that he will be good

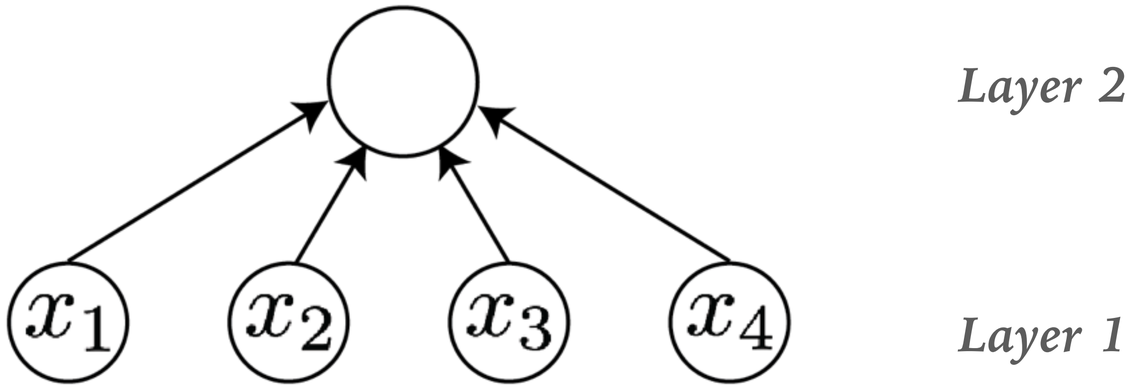
36% chance that he will be bad

Two layer neural network is almost equivalent to logistic regression.

2-Layer Perceptron

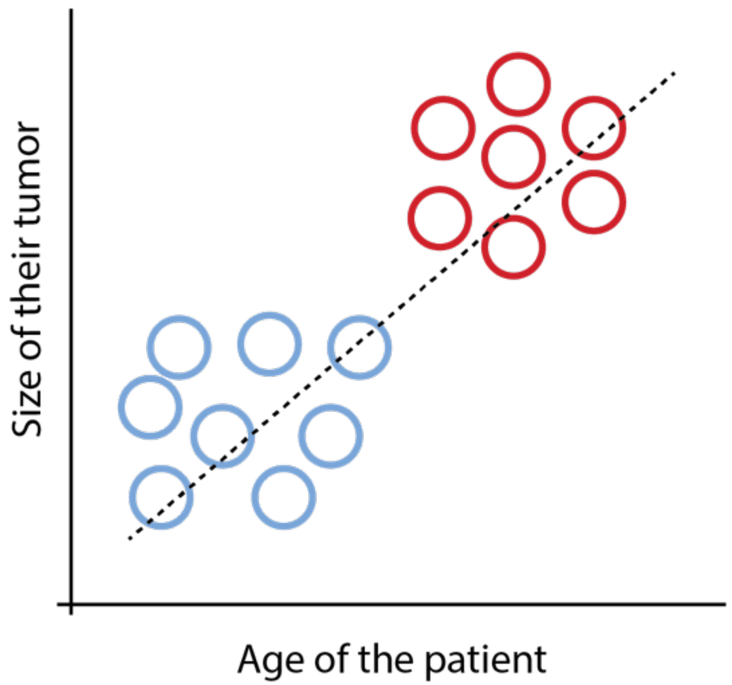
So far we have seen only neural network with two layers, so called multilayer perceptron.

$$h = f(\Sigma_j w_j x_j)$$



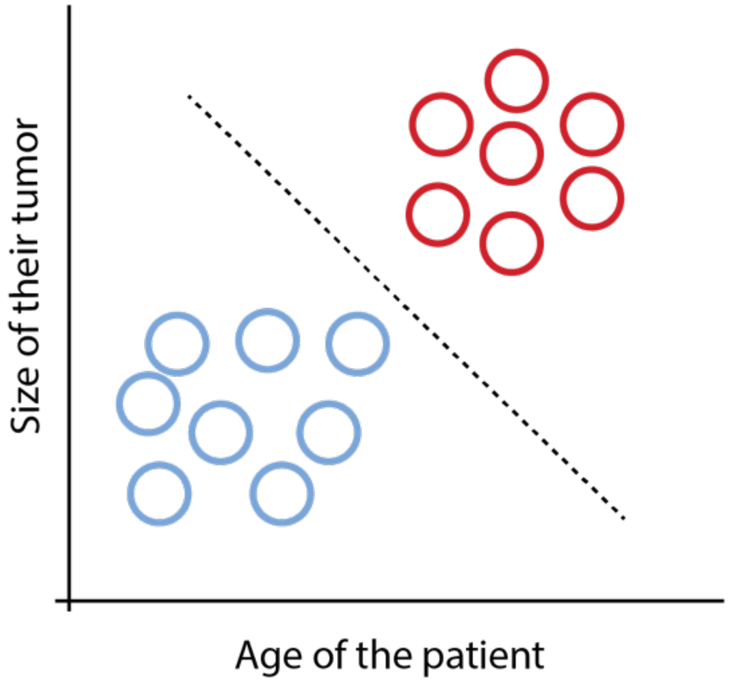
Linear Decision Boundary

2-layer neural network fits a linear decision boundary. It is the so-called “Linear Classifier.” For high-dimension, you might think of it as a decision hyperplane.



$$w_1 = 0, w_2 = 1$$

cost function high



$$w_1 = 1, w_2 = -1$$

cost function low

Live Demo

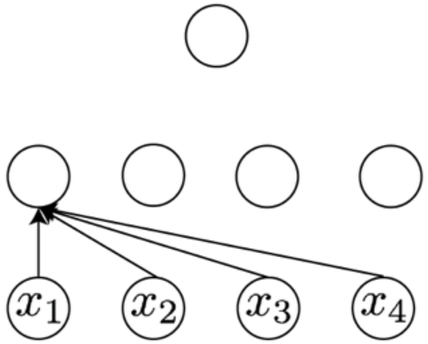
2-layer perceptron cannot fit nonlinear decision boundary.

Live Demo: *https://goo.gl/lwf8Tf*

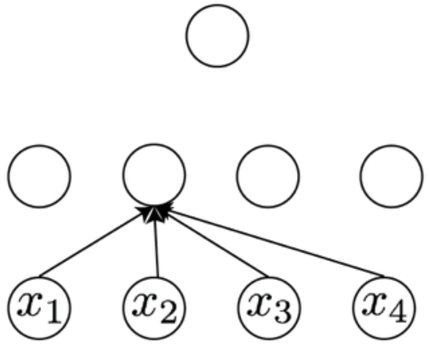
EXERCISE

- **try picking nonlinear model**
- **try adding an extra layer with 2 neurons in the hidden layer**
- **make hidden layer with 5 neurons**

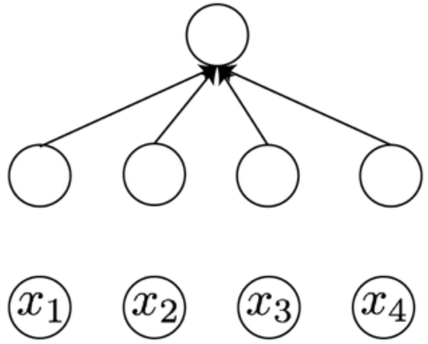
Multilayer Perceptron



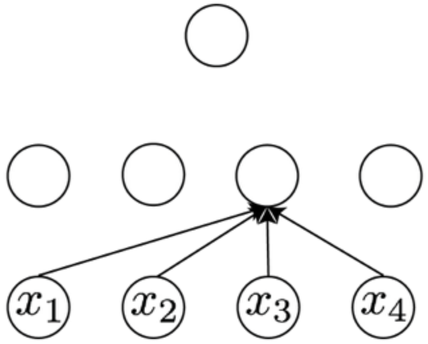
classification 1



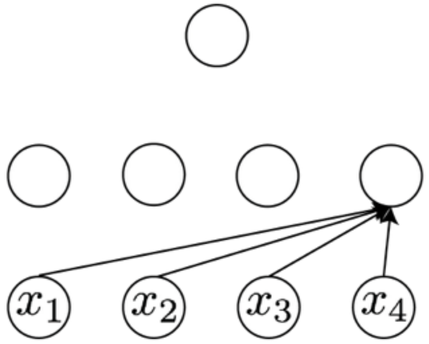
classification 2



final classification



classification 3



classification 4

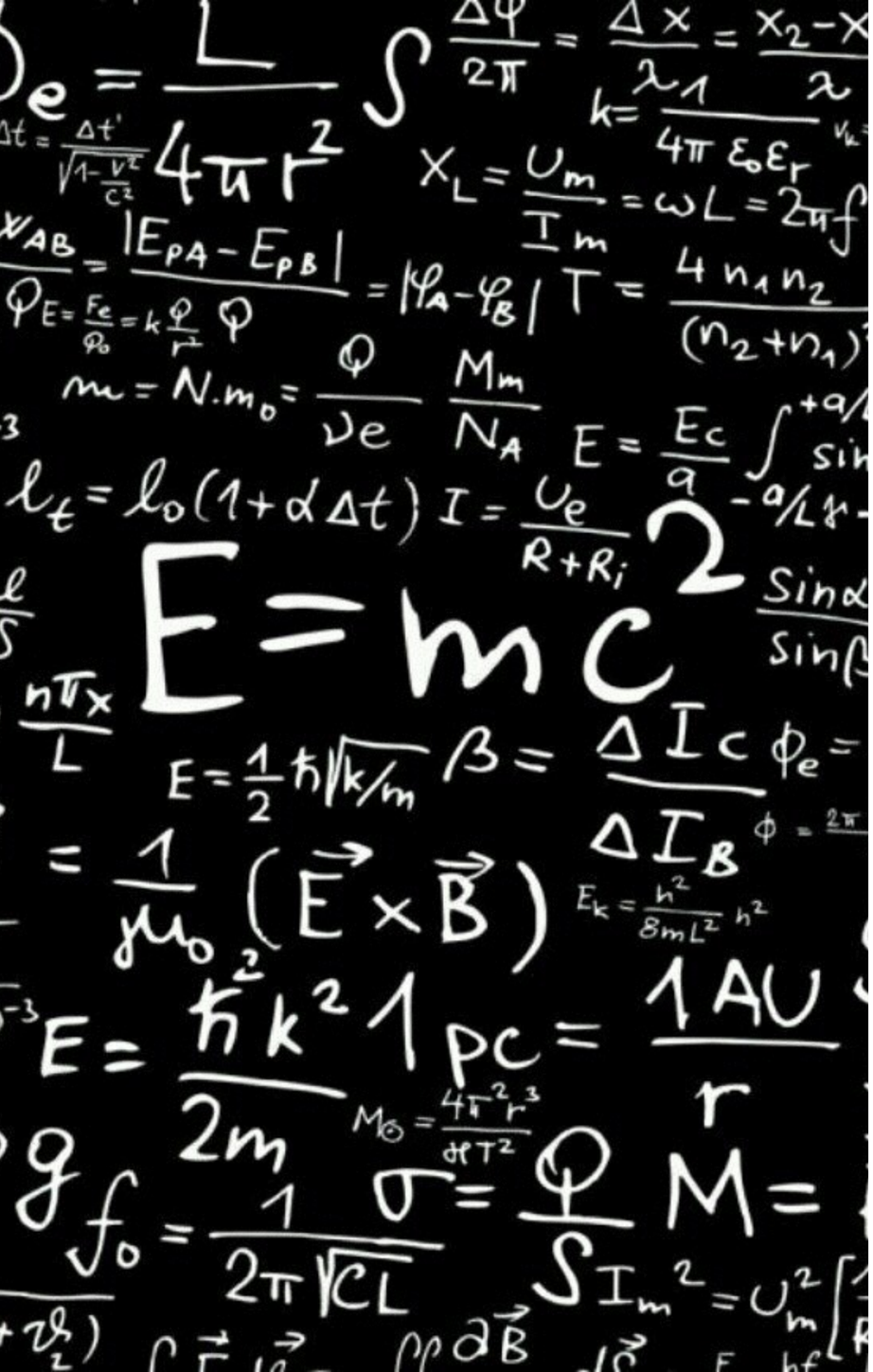
Live Demo

The more complex the boundary, the larger network you need.

Live Demo: *https://goo.gl/lwf8Tf*

EXERCISE

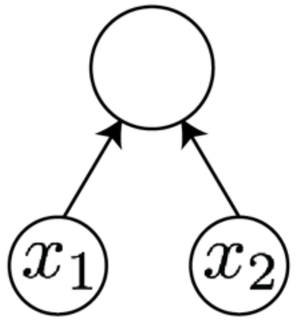
- **Try picking spiral dataset and configure network until your neural network can make accurate decision.**



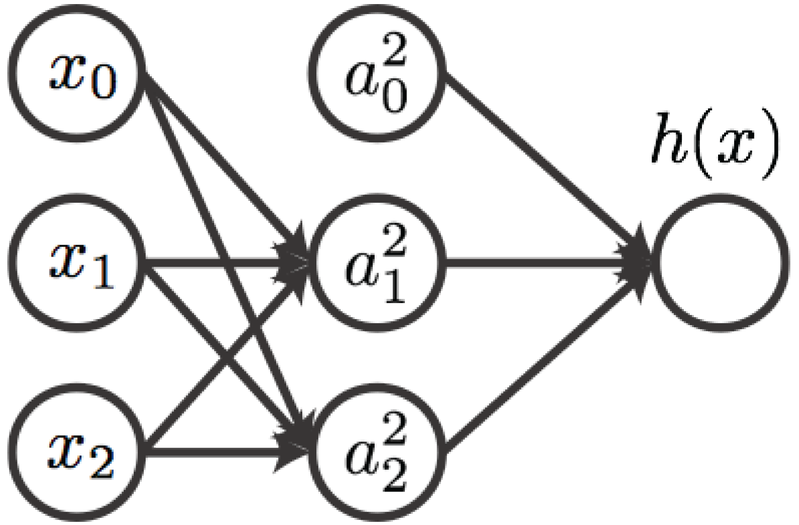
MULTILAYER PERCEPTRON QUIZ

MLP Quiz

How many layers, units, and weights we have in these two neural networks?



Network 1



Network 2

Training Neural Network

- Each neuron receives input from their friends and pass those inputs through a logistic function. Each neuron performs classification.
- It sends the vote (answer) to friends in the next layer.
- Before training, each neuron is not accurate about its classification. The purpose of training is to adjust the weight to make these classifications more accurate.

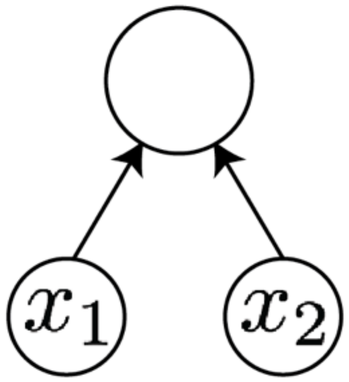
Training Neural Network

- Initialize neural network with number of layers and units we desire.
- Initialize the network weights to be random numbers.
- Measure the goodness of our neural network with a cost function (at first cost function should be high).
- Adjust the weights with a learning algorithm to minimize cost function.

Training Neural Network Example

We will start simple by training our neural network to perform regression task with 2D input.

$$h = f(\Sigma_j w_j x_j)$$



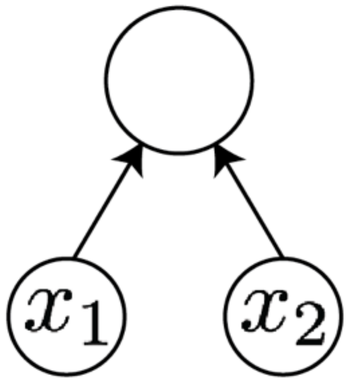
Linear Activation Function

$$f(z) = z$$

Training Neural Network Example

The purpose of regression is to adjust **W** to minimize regression cost function (sum of square error).

$$h = f(\Sigma_j w_j x_j)$$

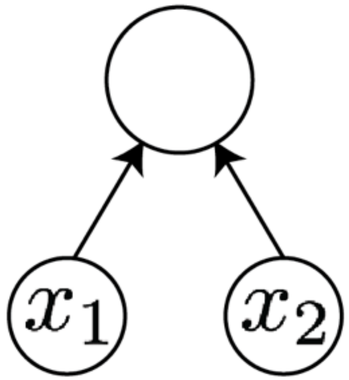


Cost Function

$$\begin{aligned} E(W) &= \Sigma_i E^i \\ &= \Sigma_i (h^i - y^i)^2 \\ &= \Sigma_i ((w_1 x_1^i - w_2 x_2^i) - y^i)^2 \end{aligned}$$

Weight Initialization

$$h = f(\Sigma_j w_j x_j)$$



$$\vec{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix}$$

- We will pick random numbers for the weights.
- Note that the weights cannot start at zeros.
- Often times, these random initial conditions are constrained to be small.

Weight Adjustment Algorithm

In this example, we will use **gradient descent algorithm** to adjust weights.

$$w_{ij} := w_{ij} - \alpha * \frac{\partial\, Cost}{\partial\, w_{ij}}$$

new weight

old weight

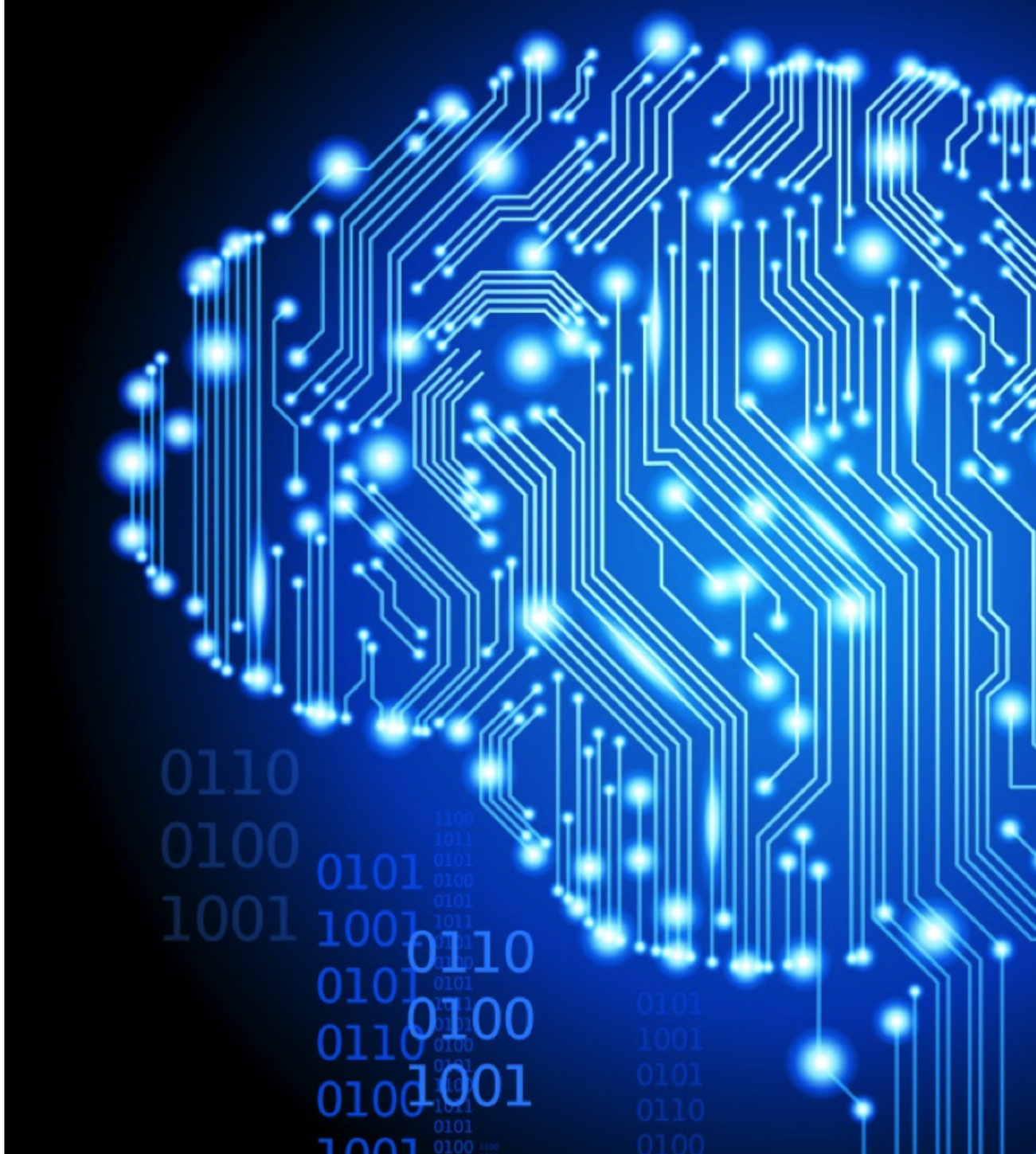
learning rate

gradient of cost function

Coding Example

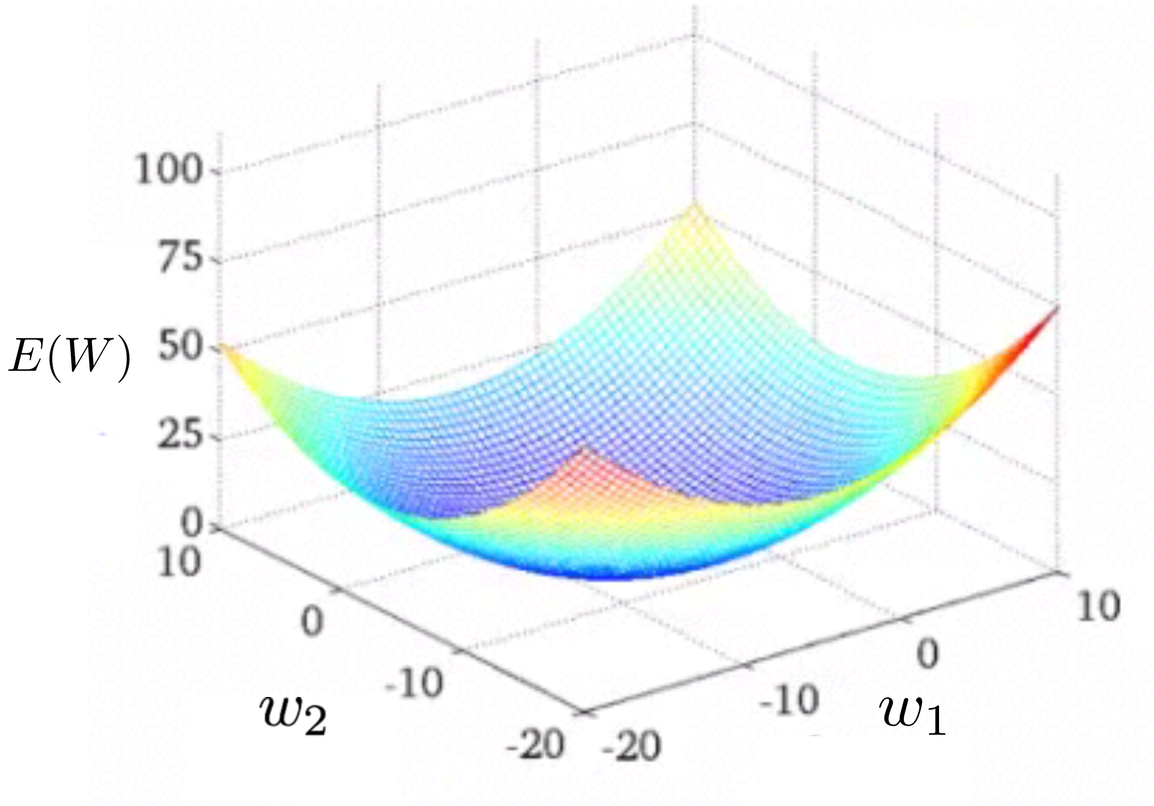
Simple Neural Network Training

https://goo.gl/31Umt2



Understanding GD

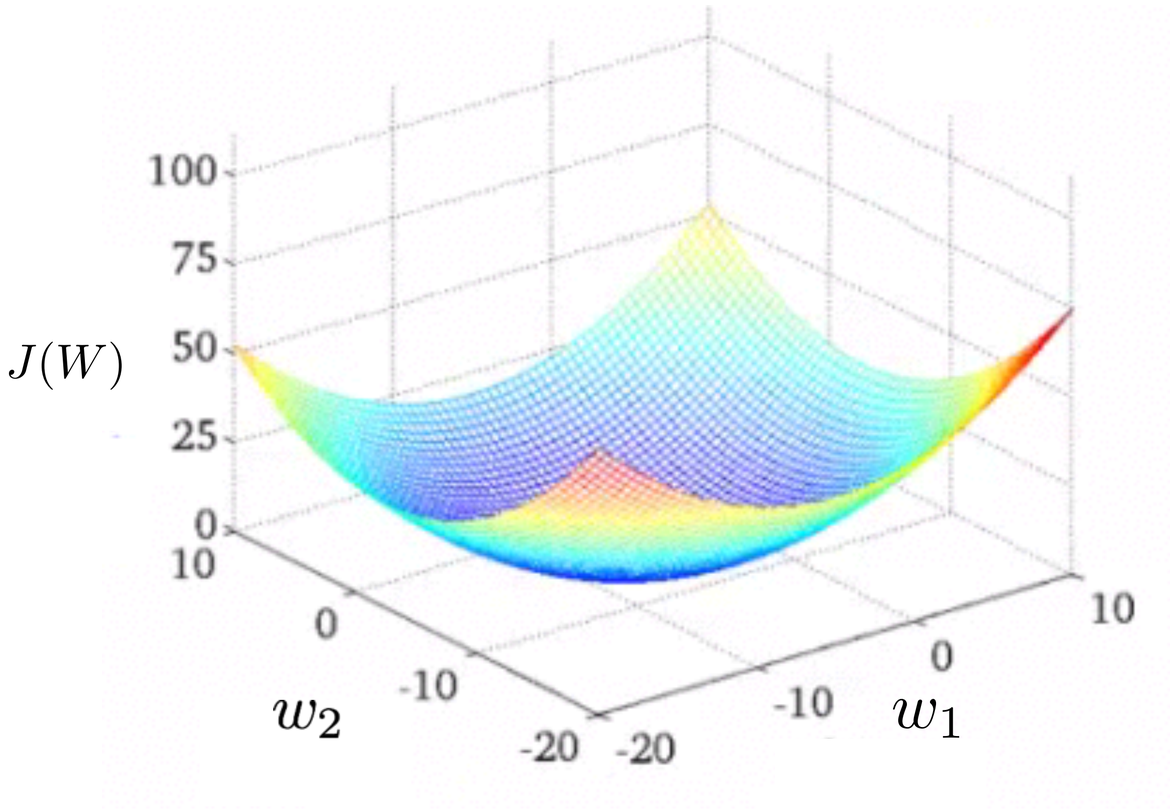
- In this example, we use gradient descent algorithm to adjust neural network weights.

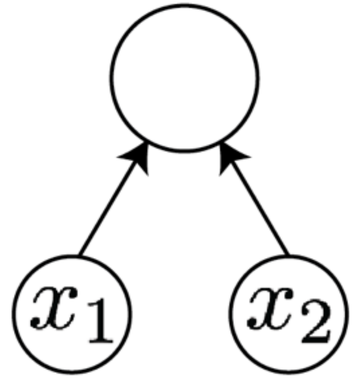


- To understand gradient descent, let's start with understanding the idea of cost function landscape.
- Given a set of x and y, we can plot cost function as a function of w1 and w2.

Cost Function Landscape

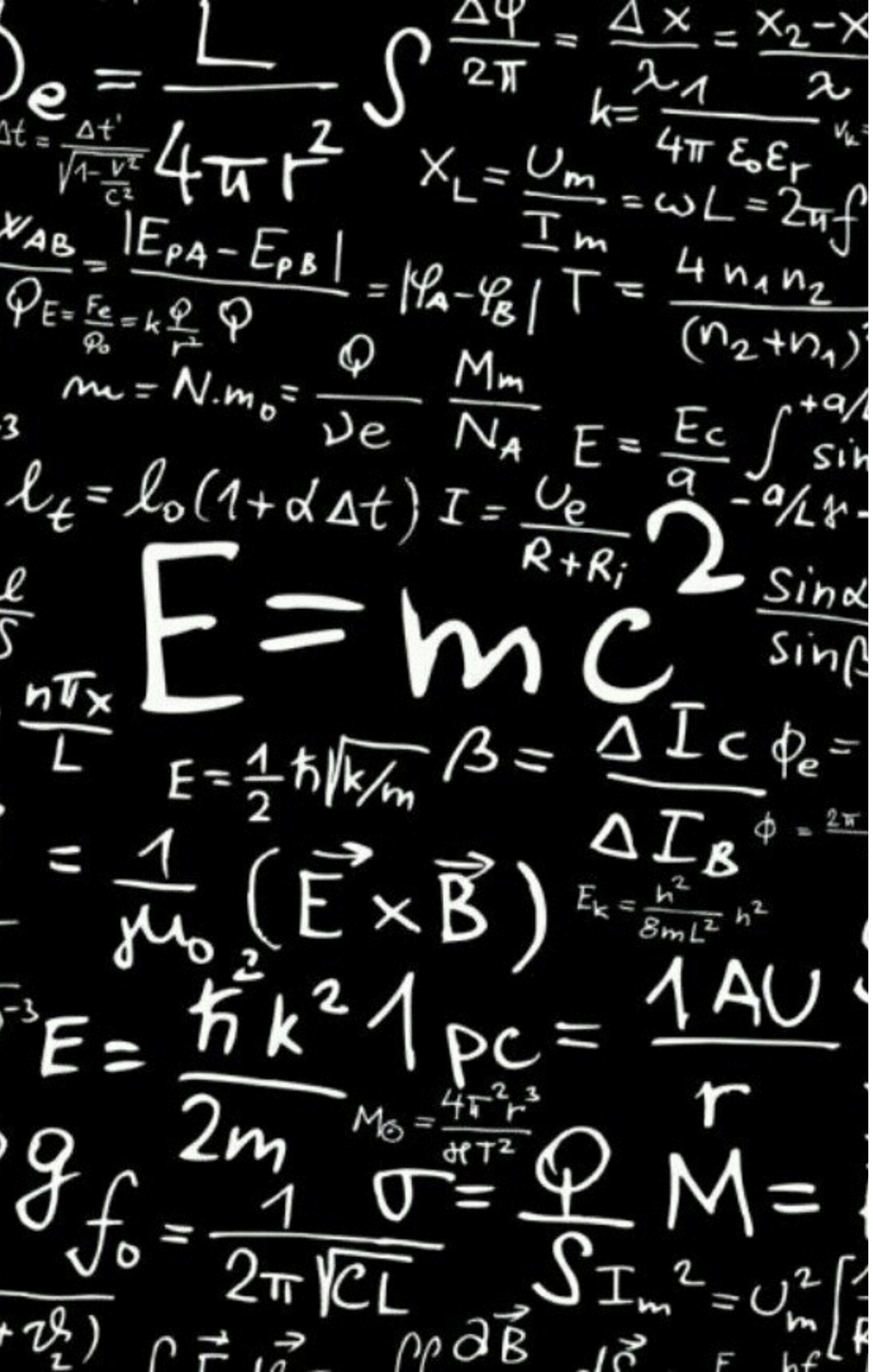
$$h = f(\Sigma_j w_j x_j)$$





Cost Function

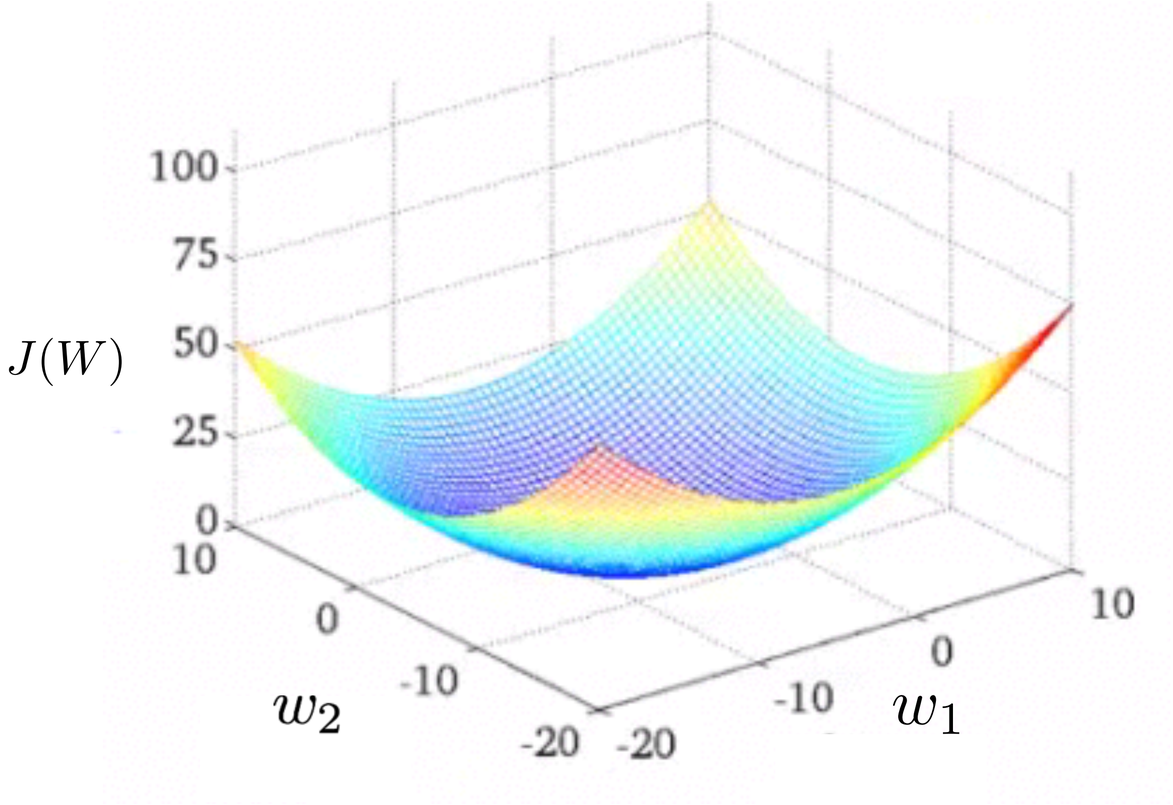
$$J(W) = \Sigma_i E^i$$
$$= \Sigma_i (h^i - y^i)^2$$
$$= \Sigma_i ((w_1 x_1^i - w_2 x_2^i) - y_i)^2$$



COST FUNCTION LANDSCAPE QUIZ

Cost Function Landscape Quiz

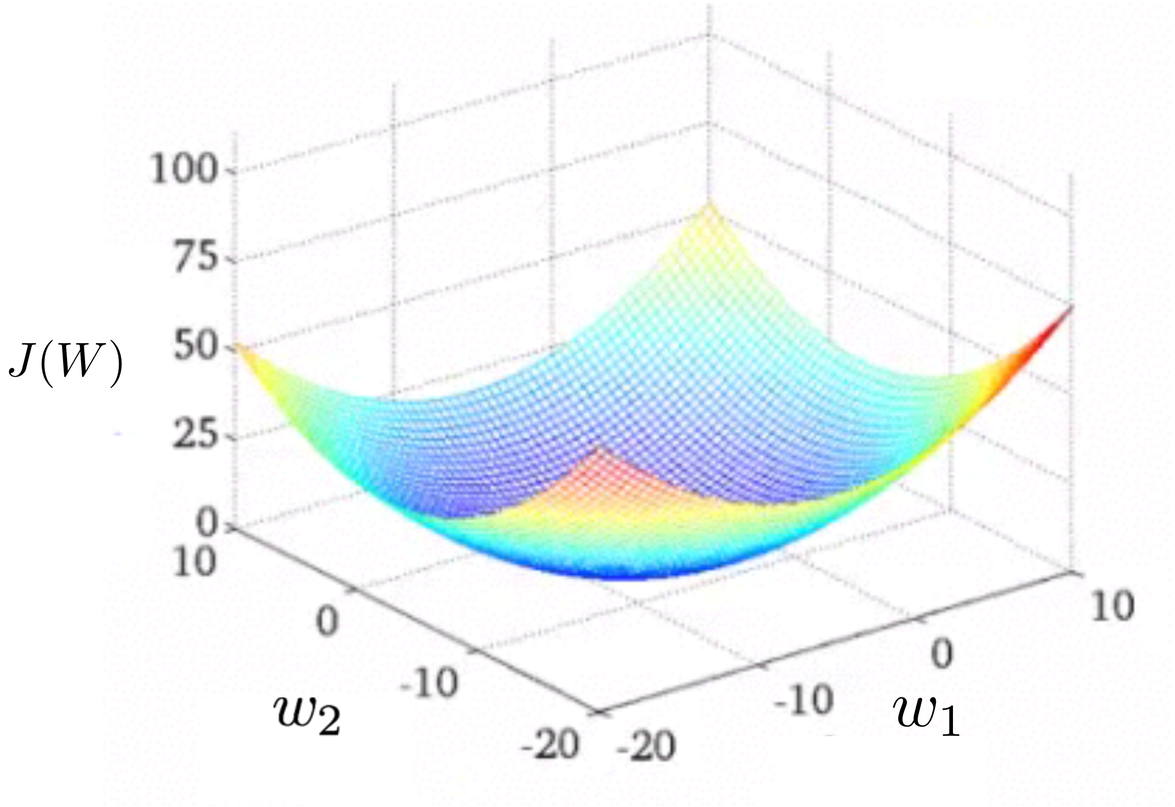
Answer the following questions about the graph. Just eyeball the graph and give a rough estimate answer.



- **What w1 and w2 correspond to the bottom of the bowl.**
- **What value of J will you get at the bottom of the bowl.**
- **Will the bowl look different if our training set (x and y) are different?**

Gradient Descent Algorithm

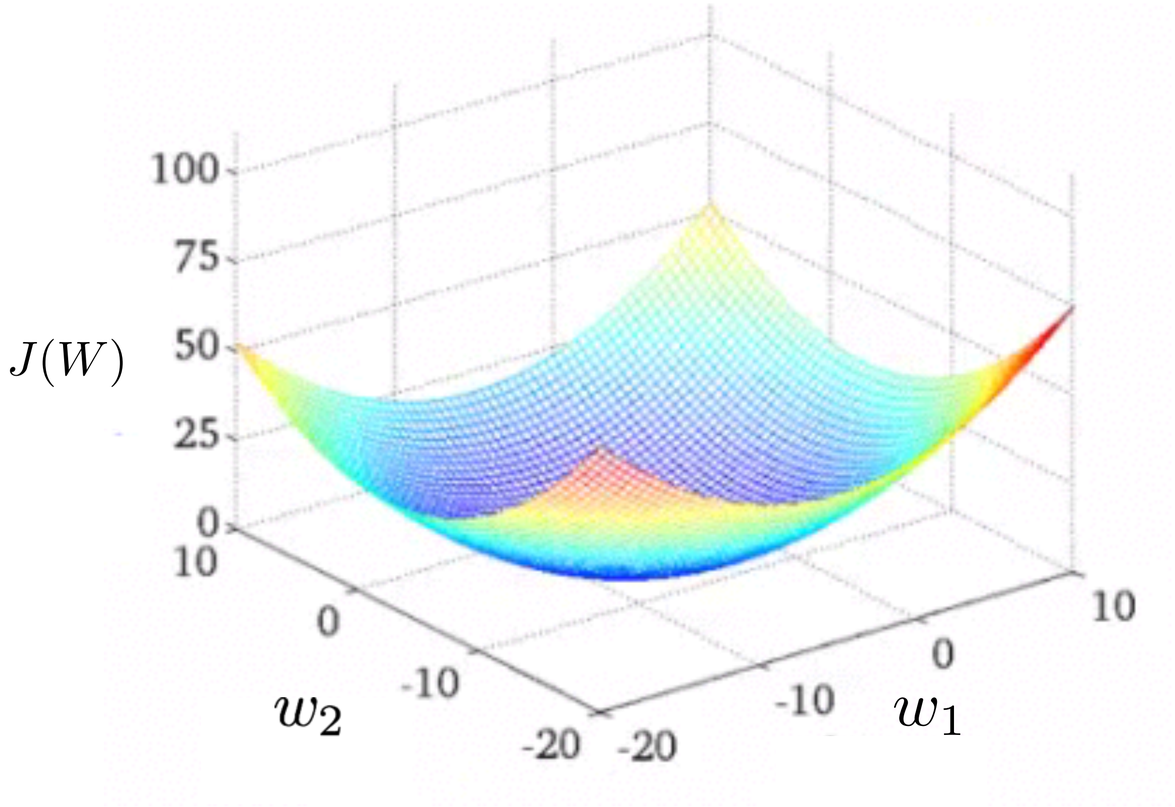
Getting to the bottom of the bowl with gradient descent.



- Pick any pair of w1 and w2, getting dropped at any point in the landscape.
- Find a gradient at that point.
- Gradient ($-\nabla J(\theta_0, \theta_1)$) will always point to the steepest direction down the bowl.

Gradient Descent Algorithm

Getting to the bottom of the bowl with gradient descent.

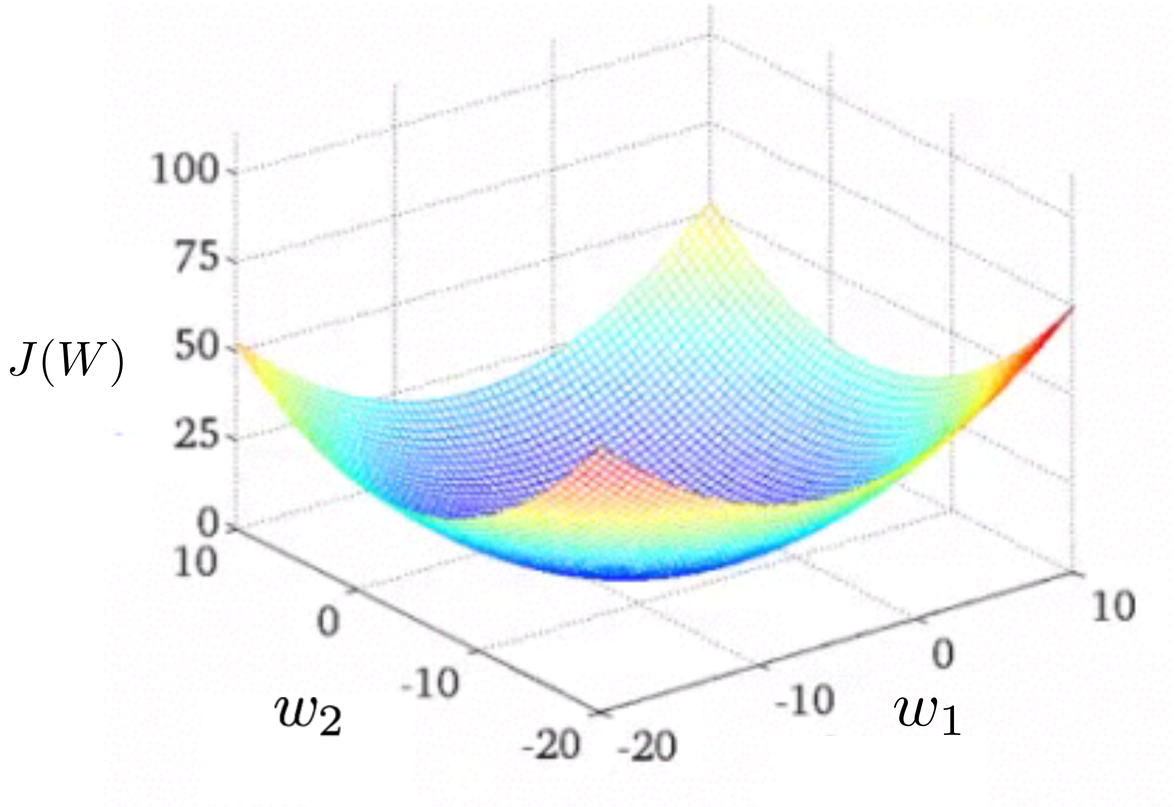


- Take a step down the bowl with the length of the footstep = α
- Each step, you will move from one point to another:

$$\begin{bmatrix}\theta_0\\ \theta_1\end{bmatrix} \rightarrow \begin{bmatrix}\theta_0\\ \theta_1\end{bmatrix} - \alpha\nabla J(\theta_0, \theta_1)$$

Gradient Descent Algorithm

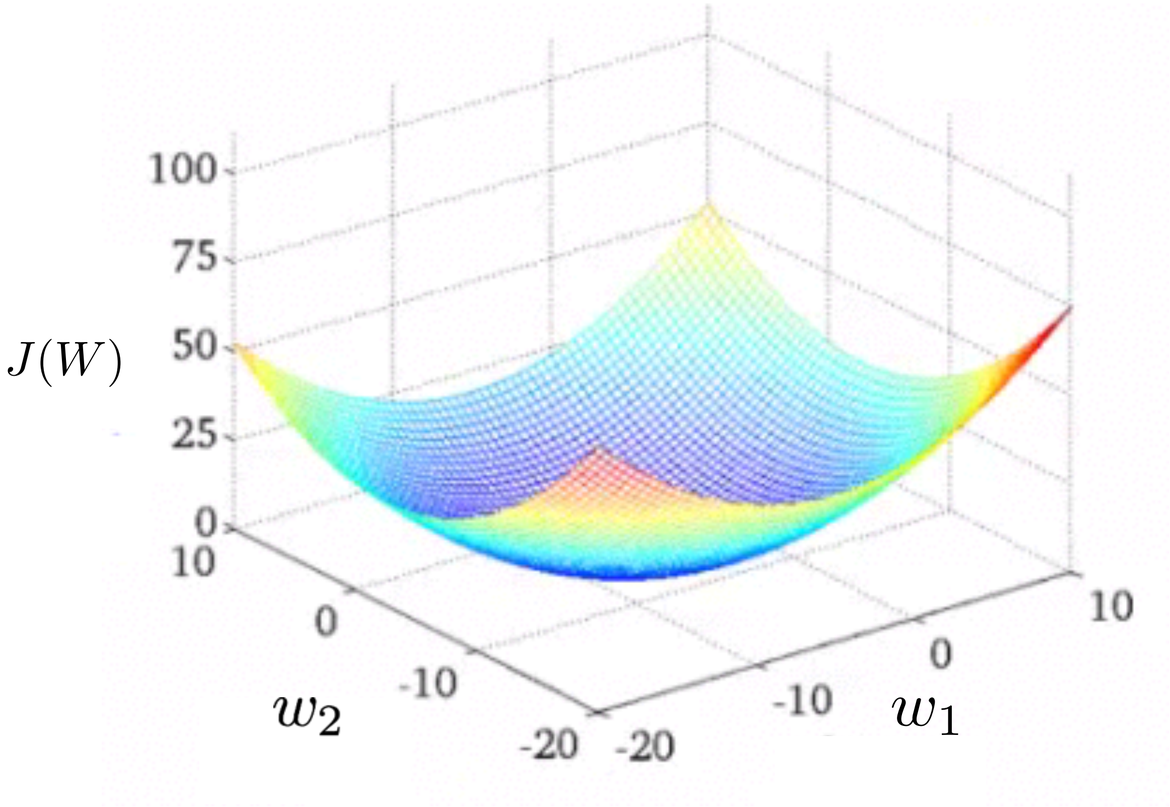
Getting to the bottom of the bowl with gradient descent.



- Continue walking with the same rule, over and over.
- Eventually, gradient will be around zero, and your step is tiny
- No matter where you step at the bottom, no lower points can be found
- At that point, you have reached the solution!

Gradient Descent Algorithm

Summary of GD algorithm



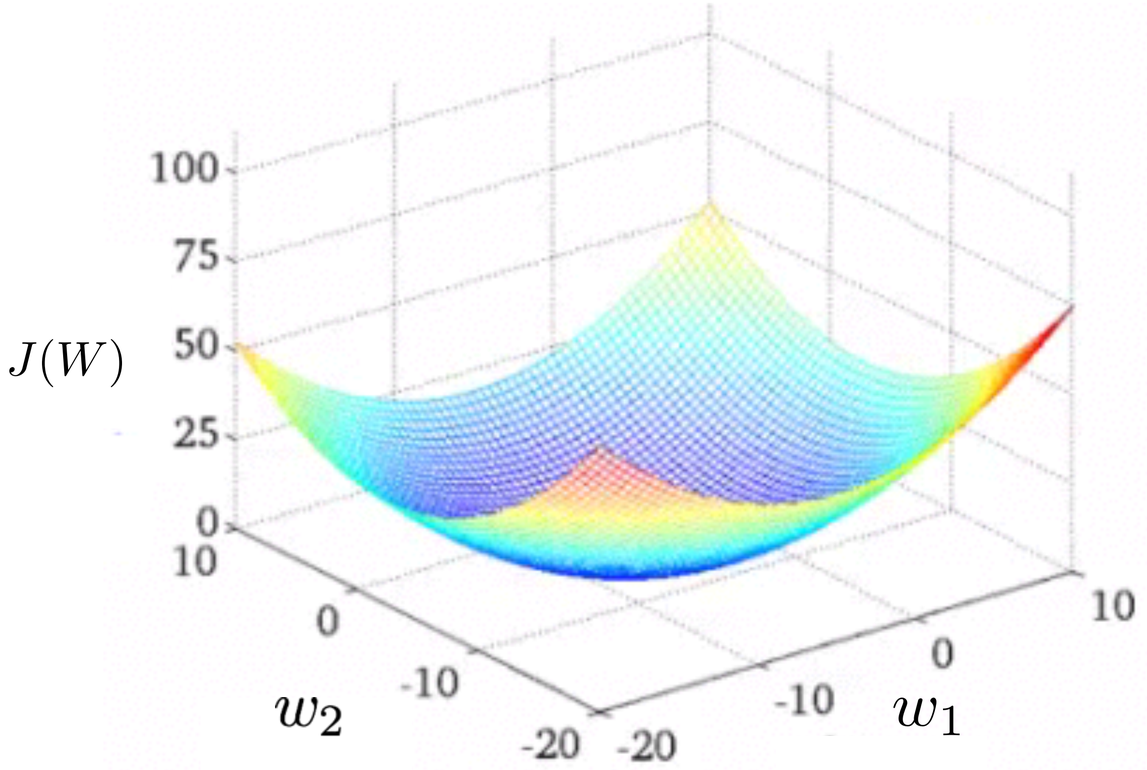
- Continue walking with the same rule, over and over.
- Eventually, gradient will be around zero, and your step is tiny
- No matter where you step at the bottom, no lower points can be found
- At that point, you have reached the solution!

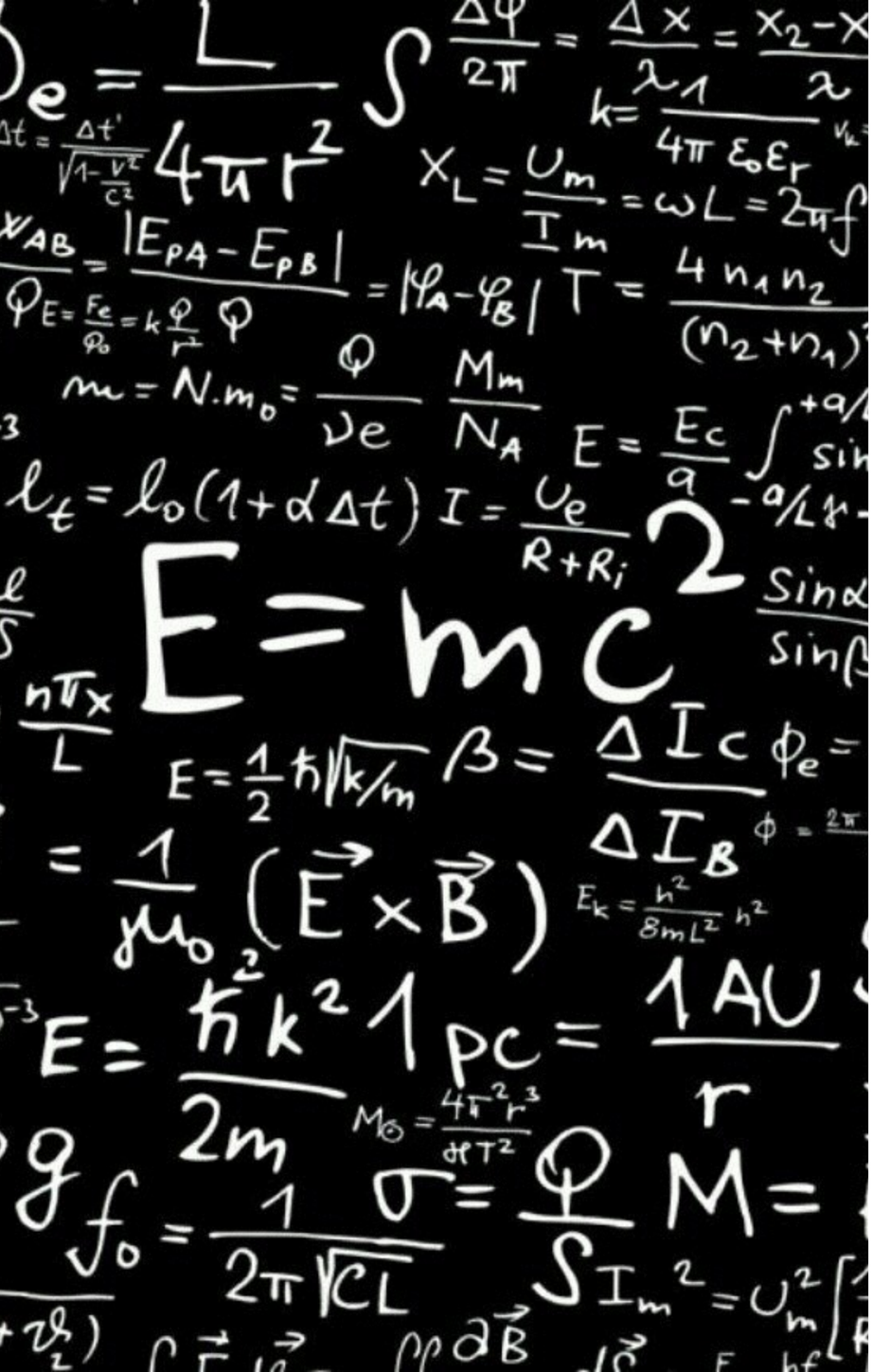
GD SUMMARY

- Randomly initialize w1 and w2
- Calculate the gradient
- Update the algorithm with the following formula:

$$\begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix} \rightarrow \begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix} - \alpha \nabla J(\theta_0, \theta_1)$$

- Monitor the cost function. Cost should be lower any time you update alpha.
- When cost function is low enough, stop updating.

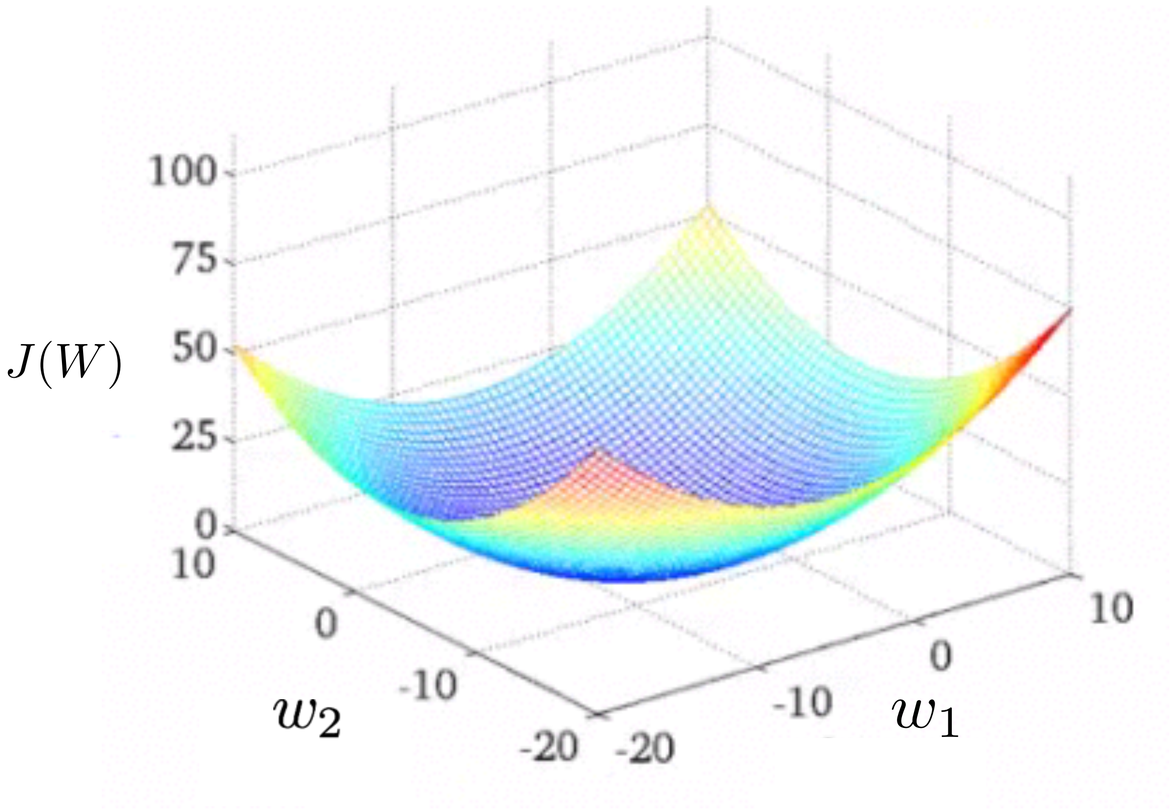




GRADIENT DESCENT QUIZ

Gradient Descent Quiz 1

Answer the following question.



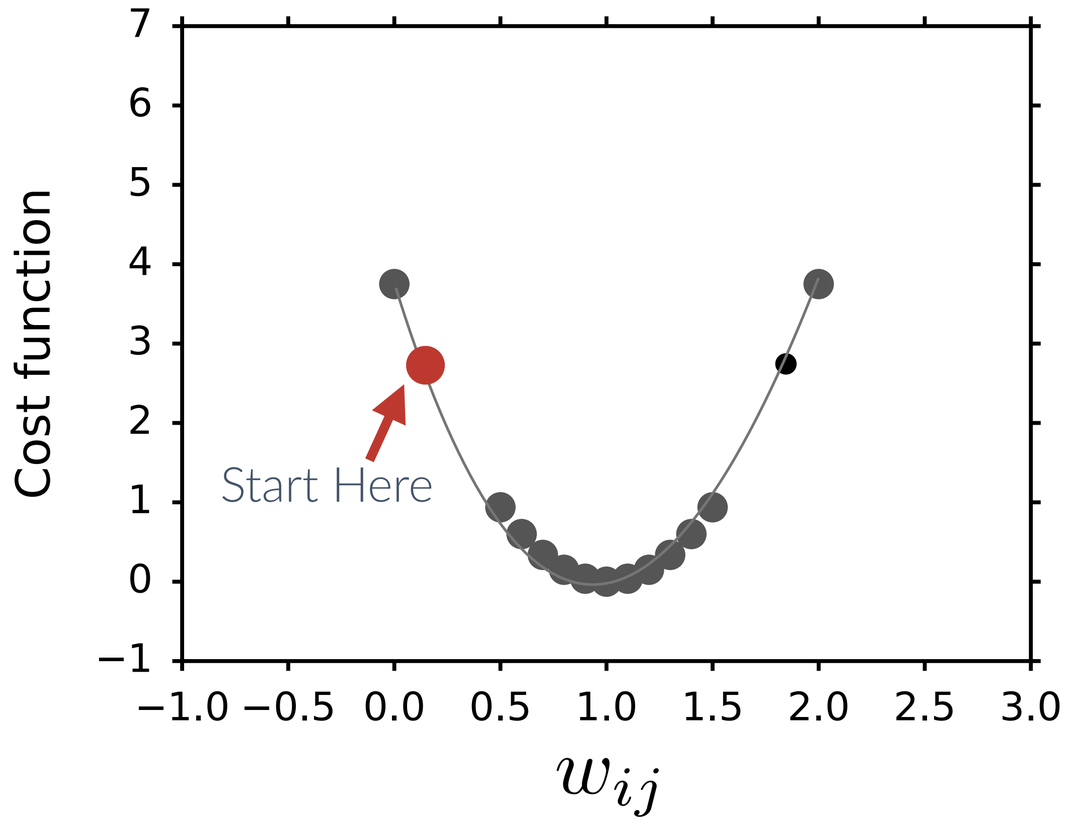
$$\begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix} := \begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix} - \alpha \nabla J(\theta_0, \theta_1)$$

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_j}, \text{ for j=0 and j=1}$$

- What value is controlled by $\nabla J(\theta_0, \theta_1)$
  - Direction of step
  - Length of step
  - Both length and direction

Gradient Descent Quiz 2

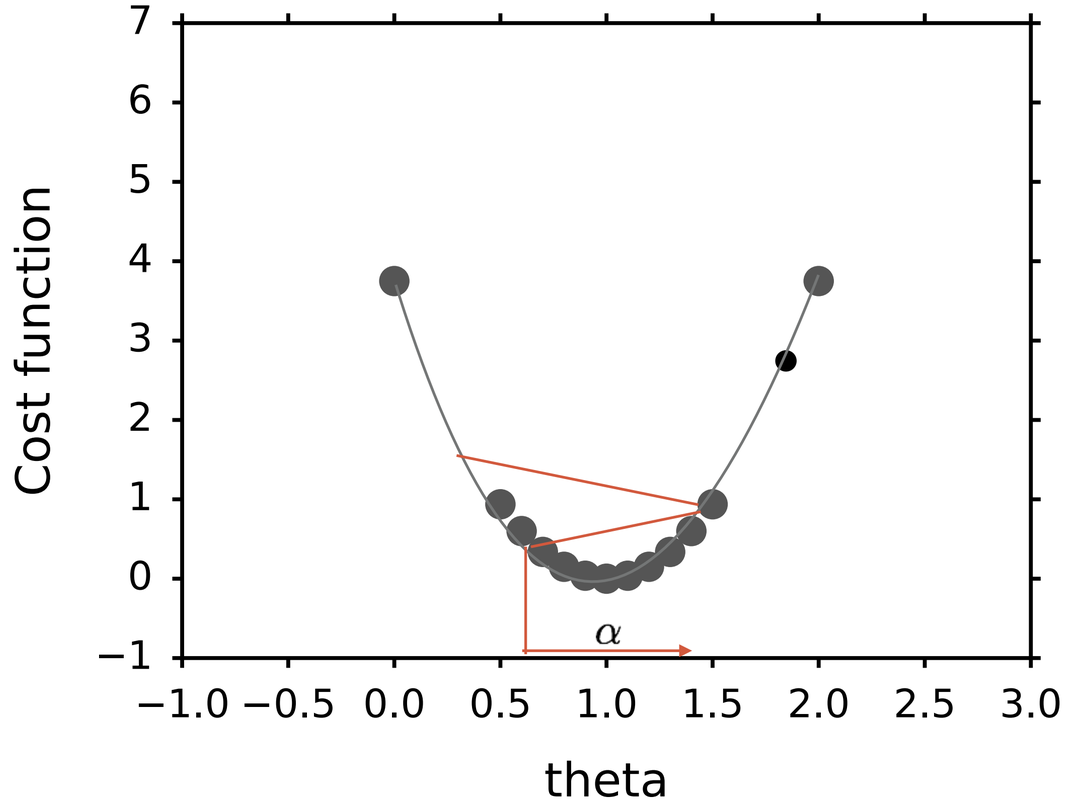
$$w_{ij} := w_{ij} - \alpha * \frac{\partial\, Cost}{\partial\, w_{ij}}$$



- You are trying to update w by the gradient descent equation above. Your initial condition is at the red point.
  - Will the gradient be positive or negative?
  - Will the algorithm suggests you to increase or decrease the value of w?
  - Will your algorithm walks to the right or the left?

Picking the Right Alpha

$$w_{ij} := w_{ij} - \alpha * \frac{\partial\ Cost}{\partial\ w_{ij}}$$



- Alpha is usually between 0 and 1
- Large alpha: big step downhill
- Small alpha: small step
- You don’t want too big or too small alpha

Picking the Right Alpha

- The gradient is big at the top of the bowl and scale smaller at the bottom of the bowl
- $\nabla J(\theta_0, \theta_1)$ controls both direction and step size.
- So at the bottom of the bowl, you don't need to scale down the learning rate, because the decreasing gradient will take care of it.
- If you overshoot the minimum though you will not see it again.

Neural Network Cost Function

Cost function: sum of errors from classifying sample i $\sum_i (E_i)$

$$E_i = -(y^i log(h(x^i)) + (1 - y^i) log(1 - h(x^i)))$$

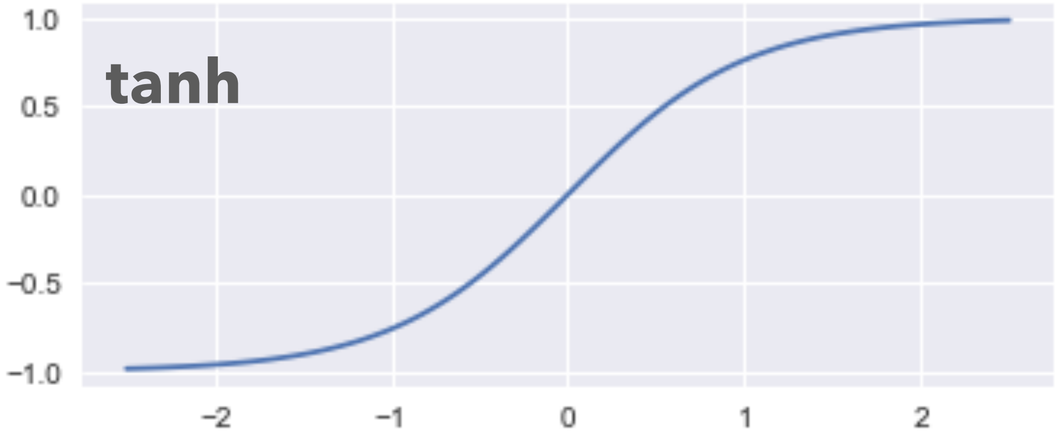
This cost function is ‘cross entropy’ error, defining how actual classes match your class predictions.

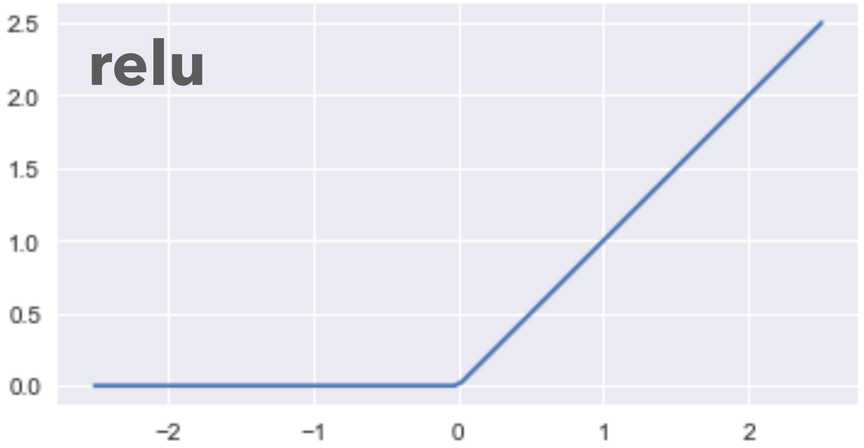
h	y	cost
0.8	1	0.10
0.1	1	1.00
0.1	0	0.05

- Activation Functions
- Bias units
- Cost functions
- Local minima
- Stop criteria
- Back propagation
- Code Examples

NEURAL NETWORK ADVANCE

The Activation Functions



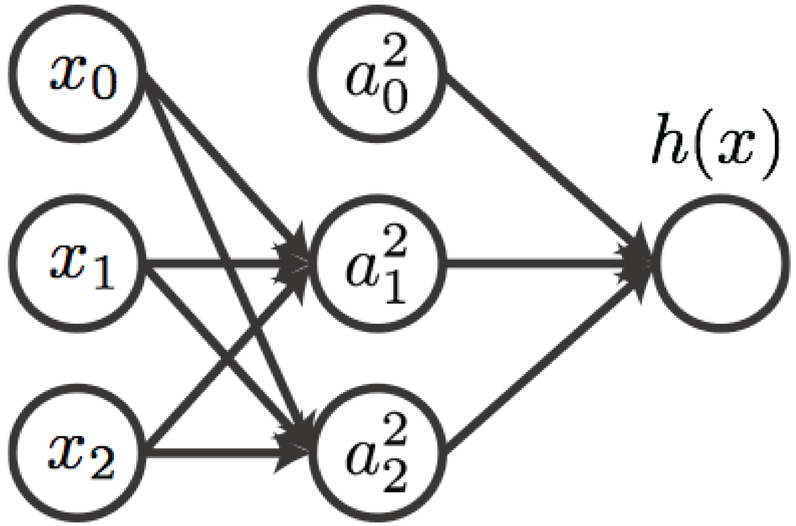


Bias Units

Bias units allows us to add any constant to the computation of each layer.

In regression, this is similar to the term θ_0

This provides a baseline for activity of neurons in each layer.



$$a_1^2 = g^2(W_{10}^1 x_0 + W_{11}^1 x_1 + W_{12}^1 x_2)$$

$$a_2^2 = g^2(W_{20}^1 x_0 + W_{21}^1 x_1 + W_{22}^1 x_2)$$

$$h(x) = g^3(W_{10}^2 a_0^2 + W_{11}^2 a_1^2 + W_{12}^2 a_2^2)$$

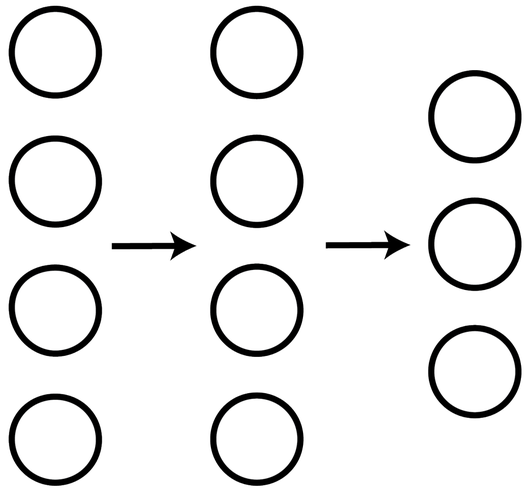
$$g^L(z) = z$$

Different Cost Functions

Classification problem with 2 classes : Binomial Log Loss

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m}[y^i log(h(x^i)) + (1-y^i)log(1-h(x^i))]$$

Classification problem with more than 2 classes : Multinomial Log Loss



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$E(W) = -\frac{1}{m}\sum_{c=1}^{k}\sum_{i=1}^{m}[y_c^i log(h(x^i)_c) + (1-y_c^i)log(1-h(x^i)_c)]$$

Different Cost Functions

Regression

For single output unit:

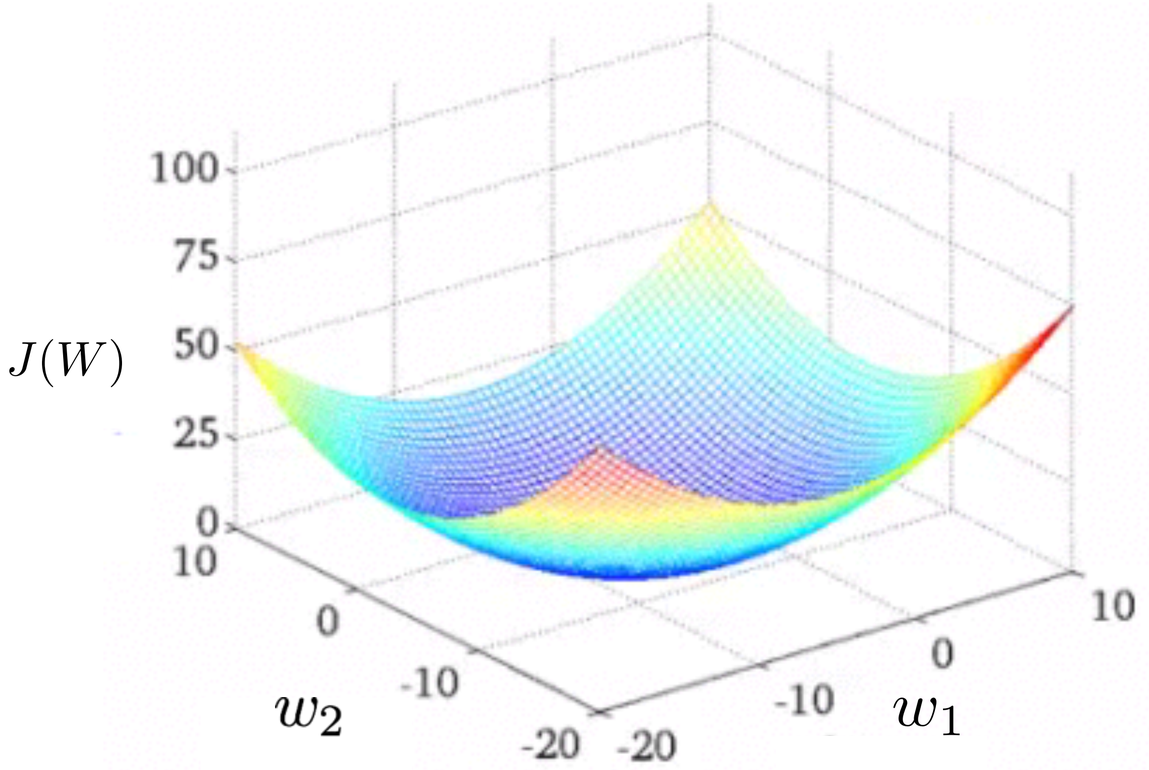
$$E(W) = \frac{1}{2m} \sum_{i=1}^{m} [y^i - h(x^i)]^2$$

For multiple output units:

$$E(W) = \frac{1}{2m} \sum_{c=1}^{k} \sum_{i=1}^{m} [y_c^i - h(x^i)_c]^2$$

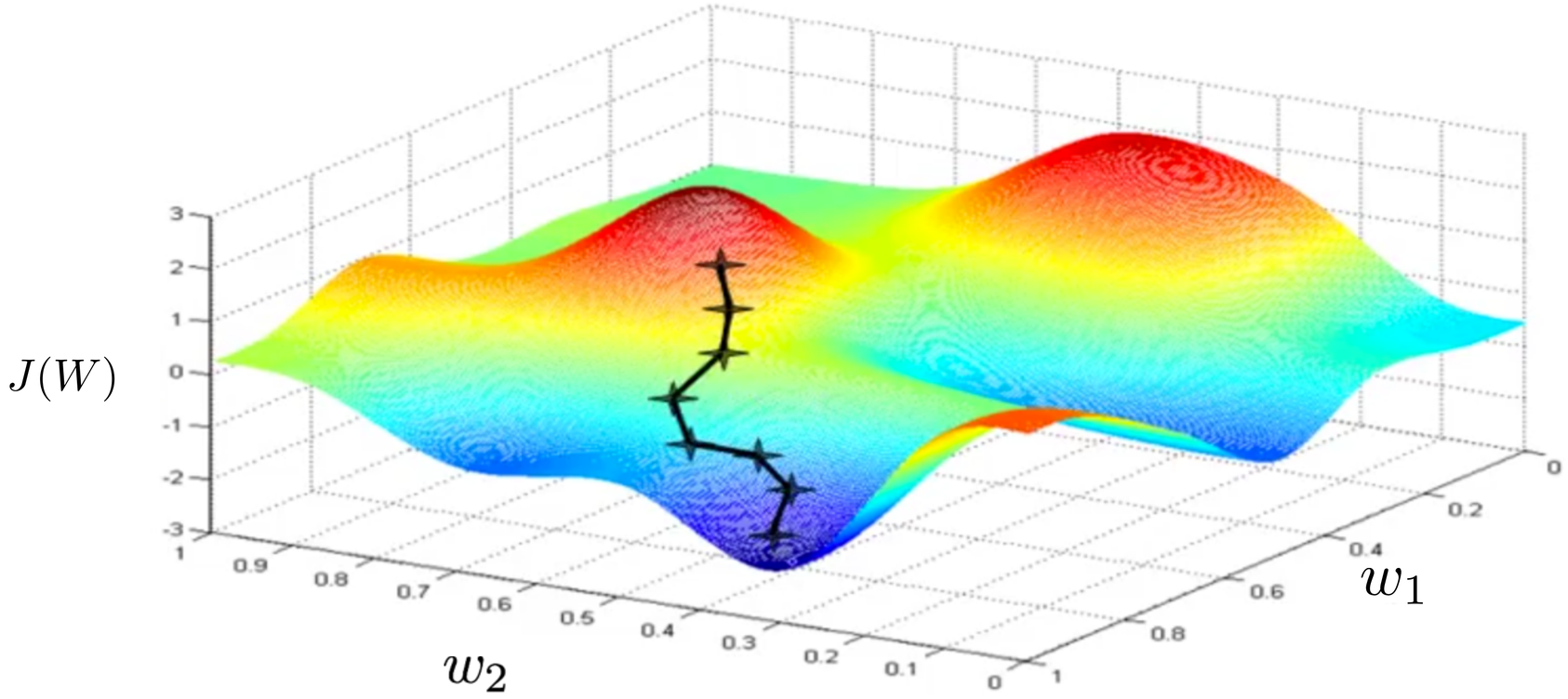
Single Minimum

- For linear regression, you always see a convex function like this, which means there's only one lowest point in the bowl.
- And no matter where you start, if you follow the gradient you will definitely reach the bottom!
- Initial conditions don't matter.



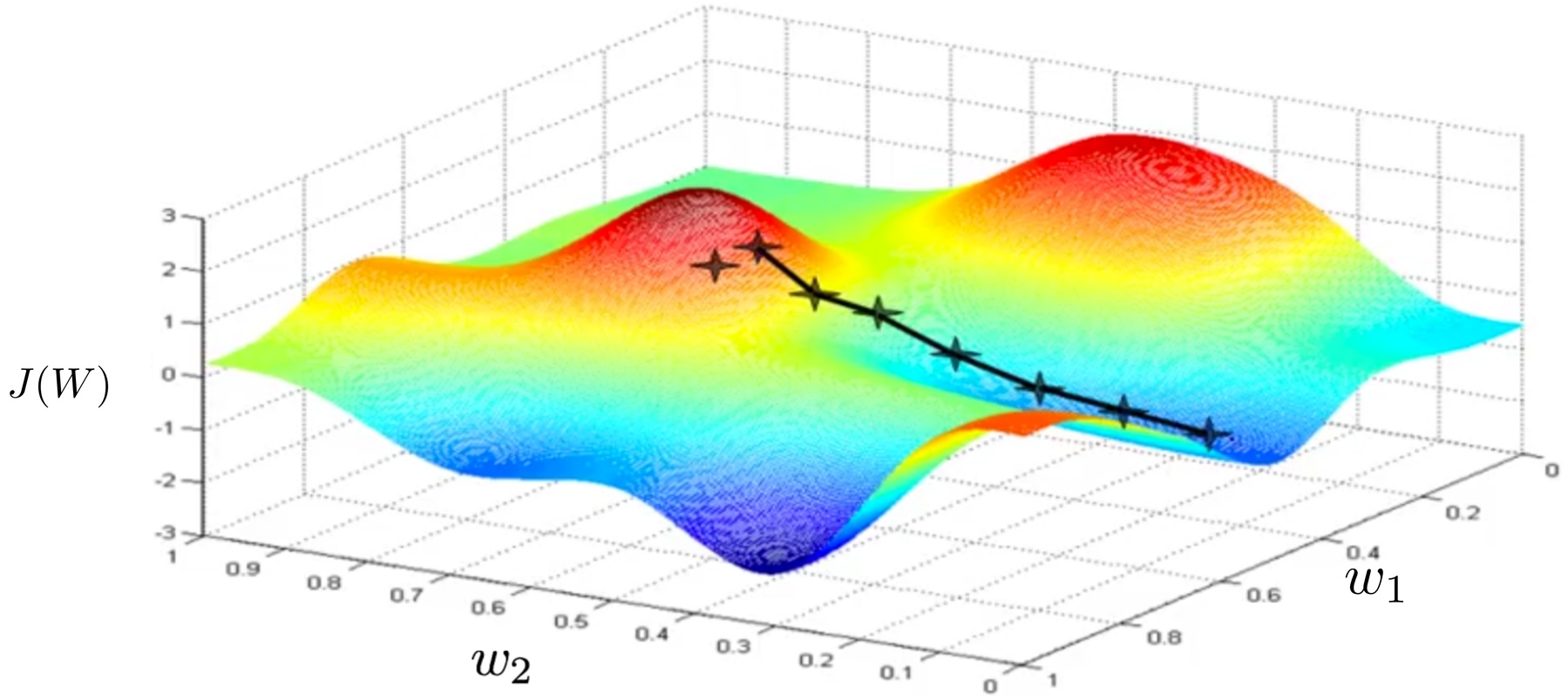
Multiple Local Minima

- In most machine learning scenarios, you will not get single-minimum problem, instead you experience getting stuck in a local minimum.



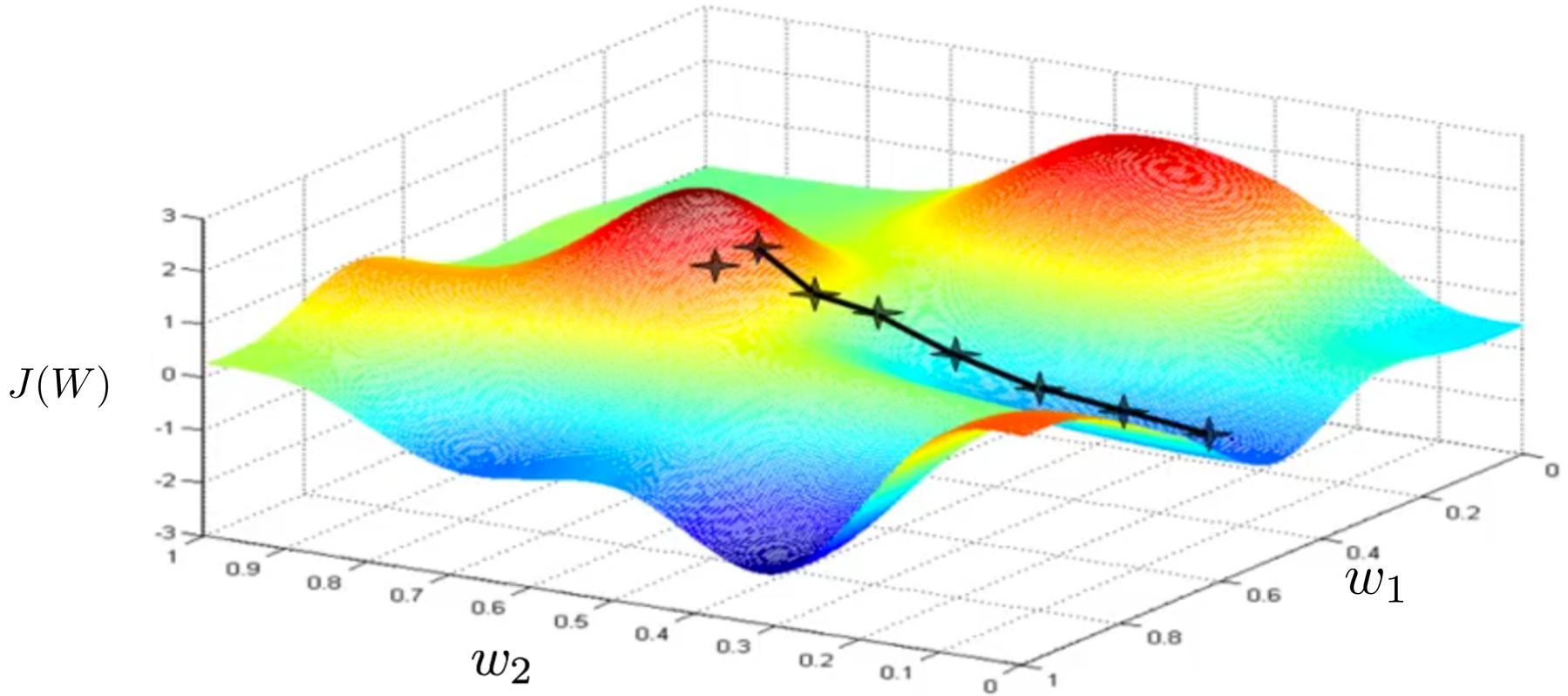
Multiple Local Minima

- When training neural network, you need to pick different initial conditions and run GD several times.



Multiple Local Minima

- When training neural network, you need to pick different initial conditions and run GD several times.



Stop Criteria

- Stop criteria tells the gradient descent algorithm when to stop updating.
- We can define stop criteria based on cost function. For example, stop when the difference between the cost in this iteration and last iteration is small.

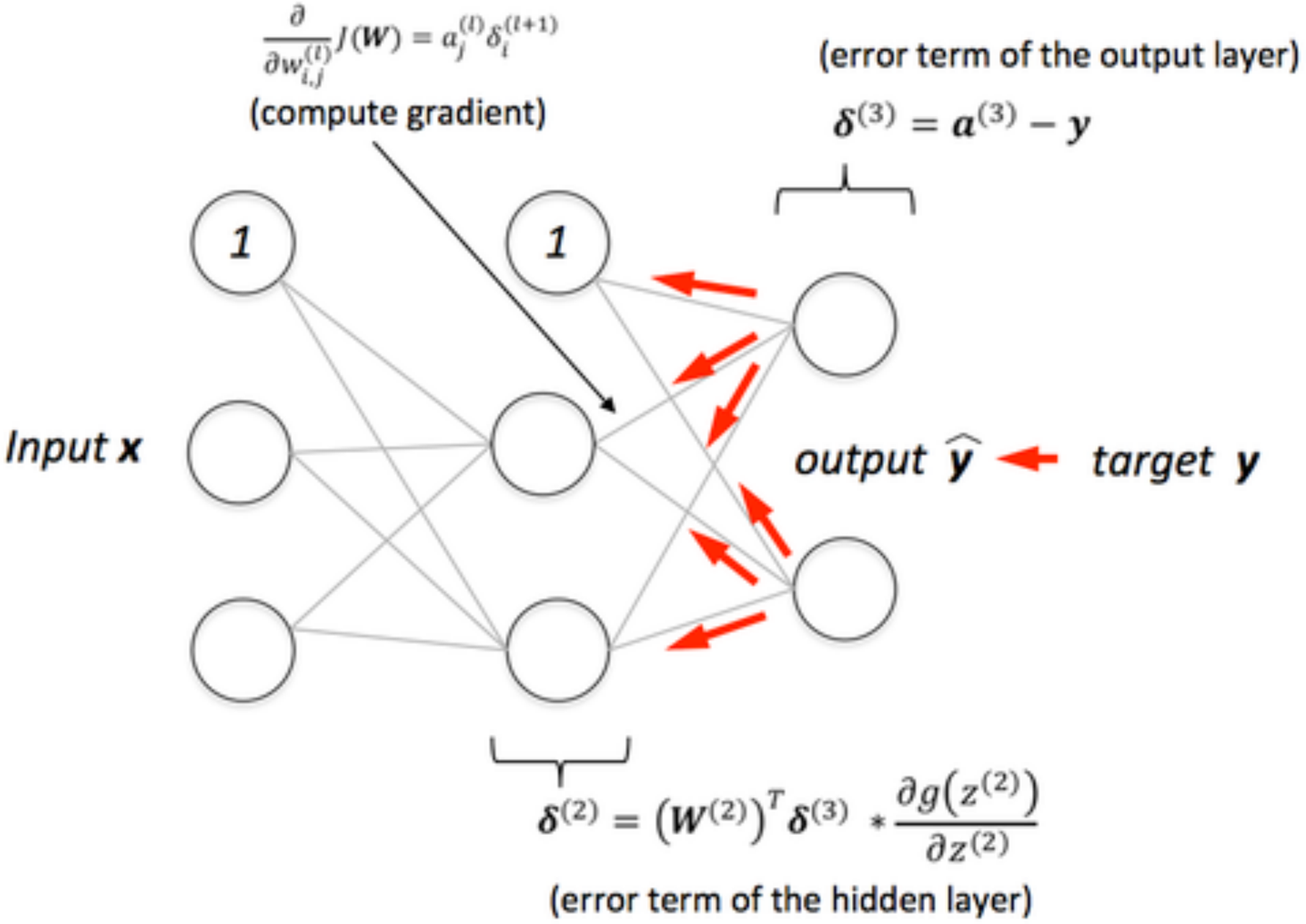
$$|cost(i-1) - cost(i)| < \epsilon$$

- The value of sigma is often called ‘tolerance’.
- Or we can stop when the gradient is very small.

Backpropagation Algorithm

- Back propagation algorithm is created to replace gradient calculation in neural network.
- When you have multiple layers of neural network, gradient calculation becomes super difficult.
- To solve this problem, mathematicians modified gradient calculation formula (chain rule) in the form of error passing algorithm.

Backpropagation Algorithm



Backpropagation Algorithm

- With Backpropagation it doesn’t matter how many layers you have. All you have to do is

  (1) back-propagate the error
  (2) calculate the gradient
  (3) update the weight accordingly

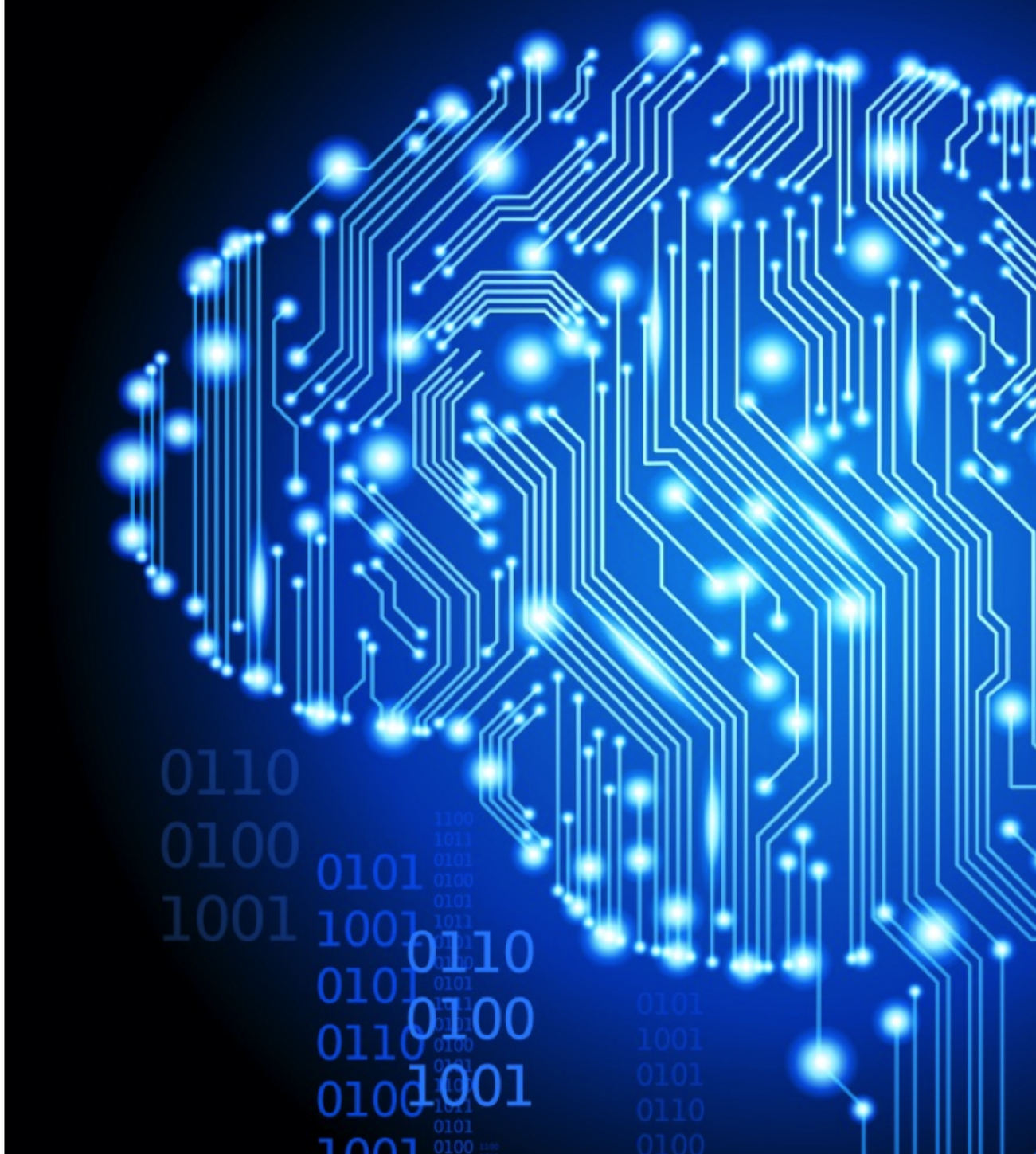
Tools for Neural Network in Python

- **Sklearn:** a module called MLPClassifier for basic and quick implementation of multilayer perceptron. The neural network part of sklearn is relatively immature but it will work for most simple tasks.
- **Tensorflow:** Google's machine learning library that can implement models from linear regression to deep convolutional neural network. Tensorflow optimizes various aspects of ML computation such as GPU compatibility.

Coding Examples

Example of Backpropagation Algorithm

https://goo.gl/0A9wX4



Sklearn MLPClassifier

DOCUMENTATION

http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Coding Examples

Example of Backpropagation Algorithm

https://goo.gl/0A9wX4

Neural Network Coding with Scikit Learn

https://goo.gl/BEyyPb

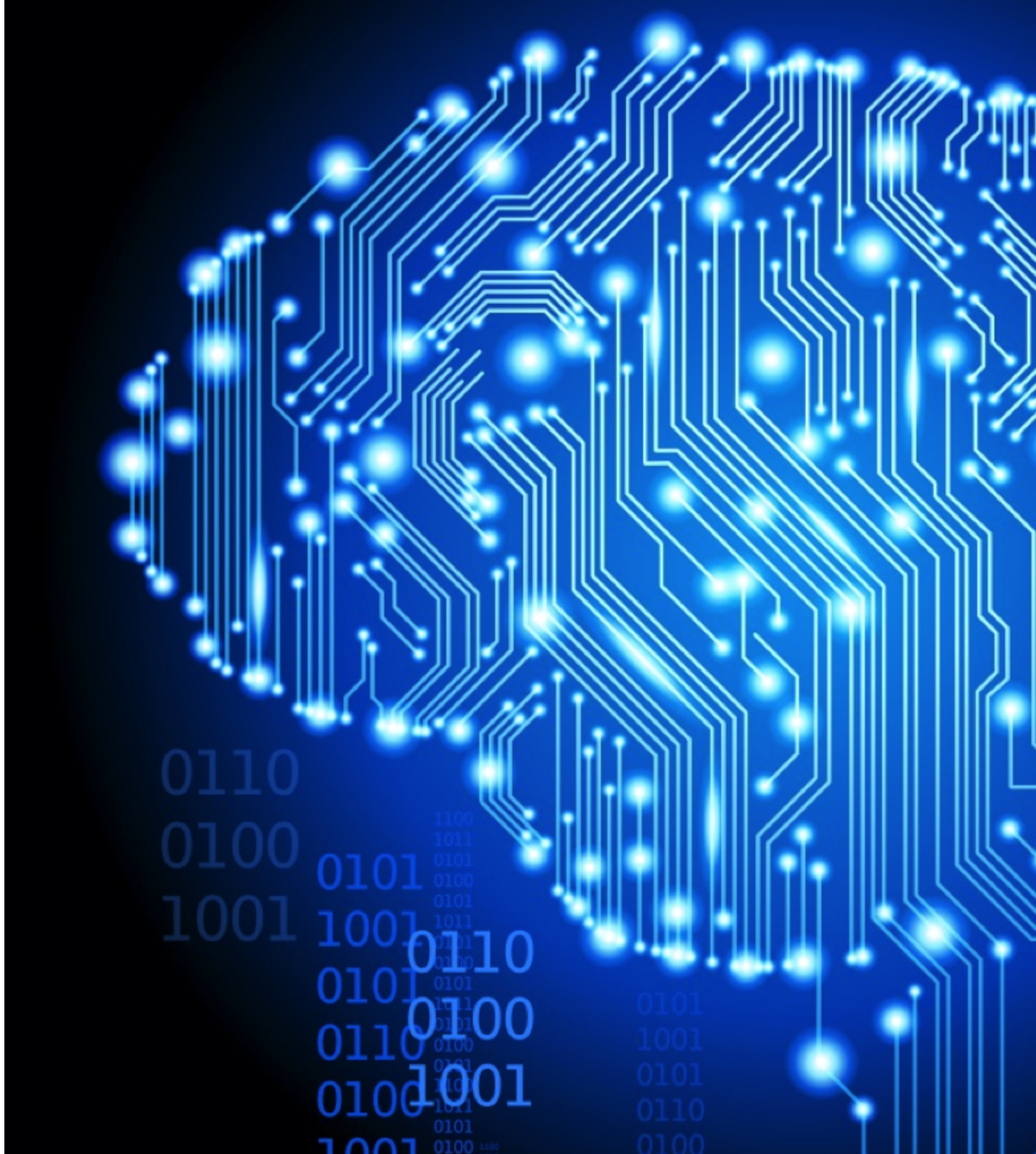


Image Classification
with Neural Network

NEURAL NETWORK CODING LAB

Coding Examples

Example of Backpropagation Algorithm

https://goo.gl/0A9wX4

Neural Network Coding with Scikit Learn

https://goo.gl/BEyyPb

Lab Exercise

https://goo.gl/PNYmdP