WORKSHOP SERIES

Deep Learning
with TensorFlow

Google Developers Experts

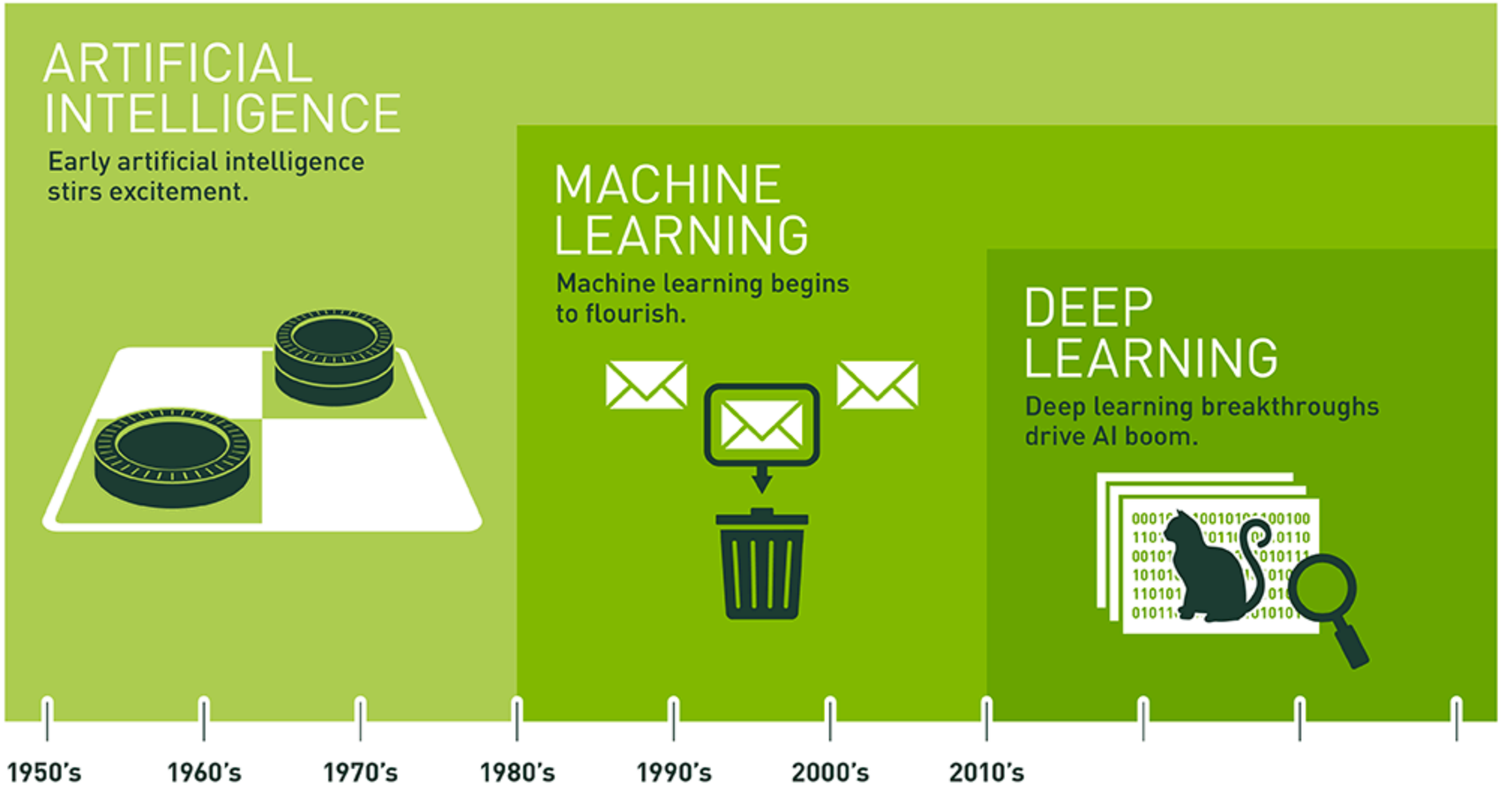
DEEP LEARNING 101

Ta Virot Chiraphadhanakul, PhD

“Artificial Intelligence is the new electricity.”

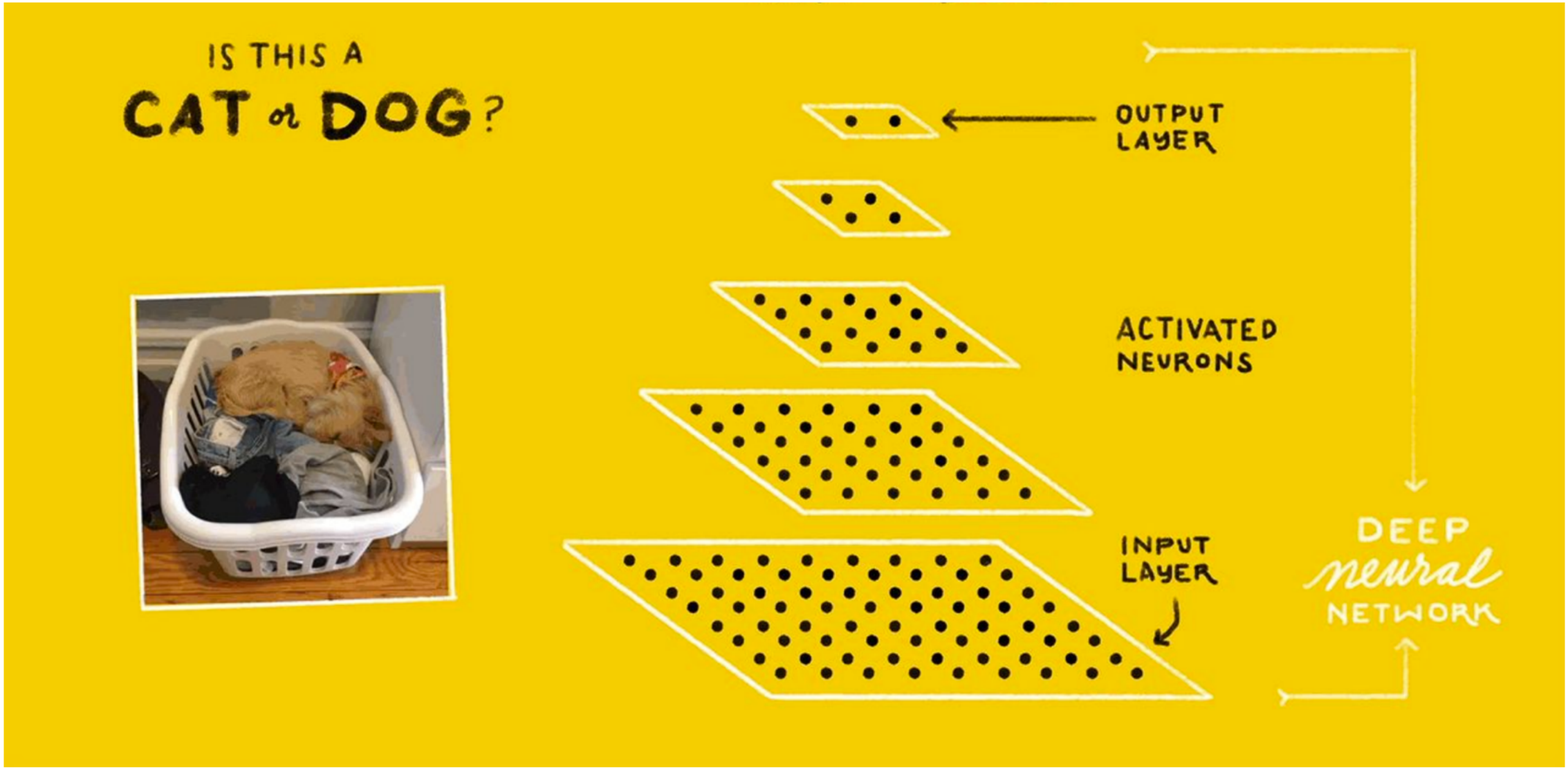
— Andrew Ng

Deep Learning

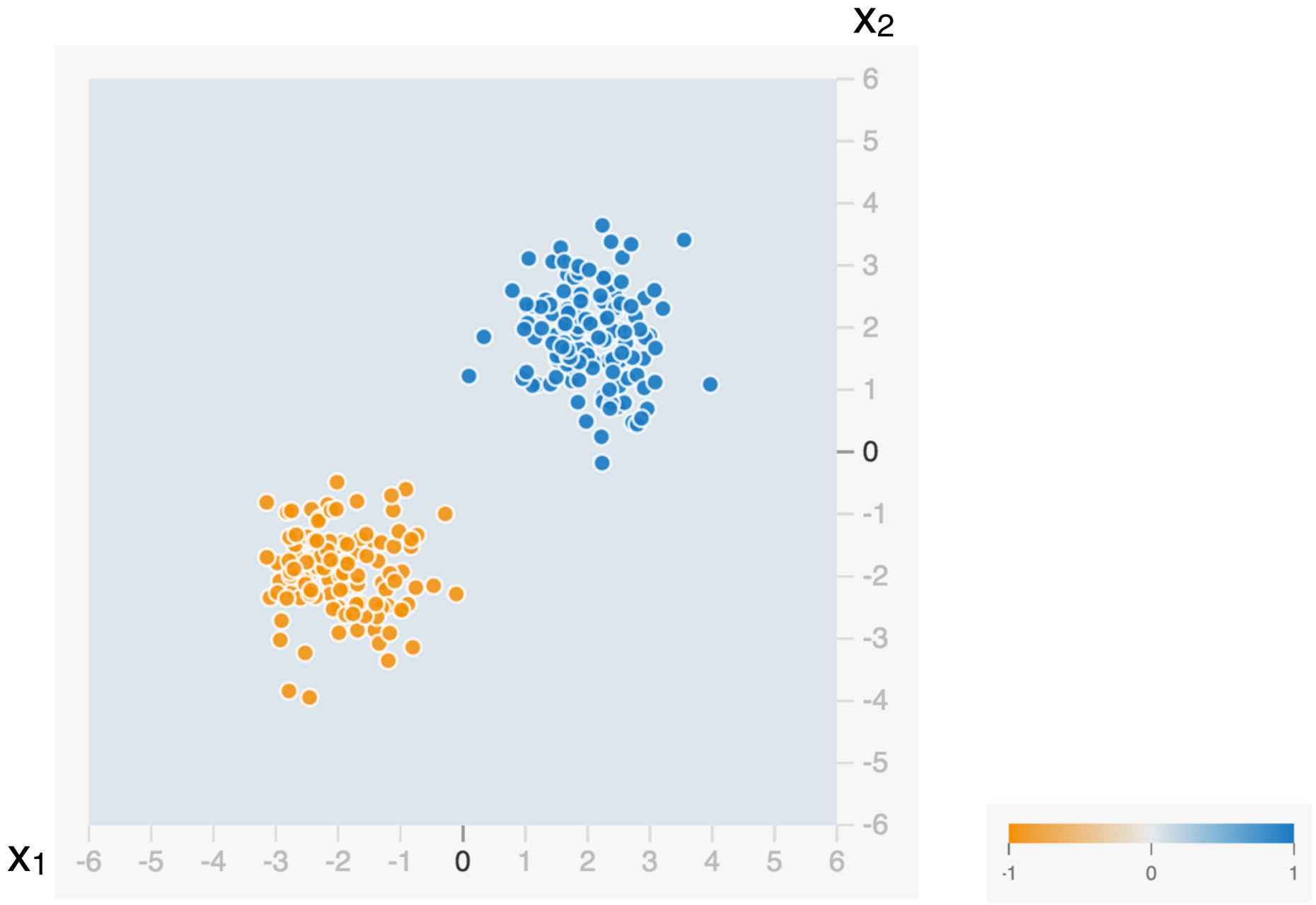


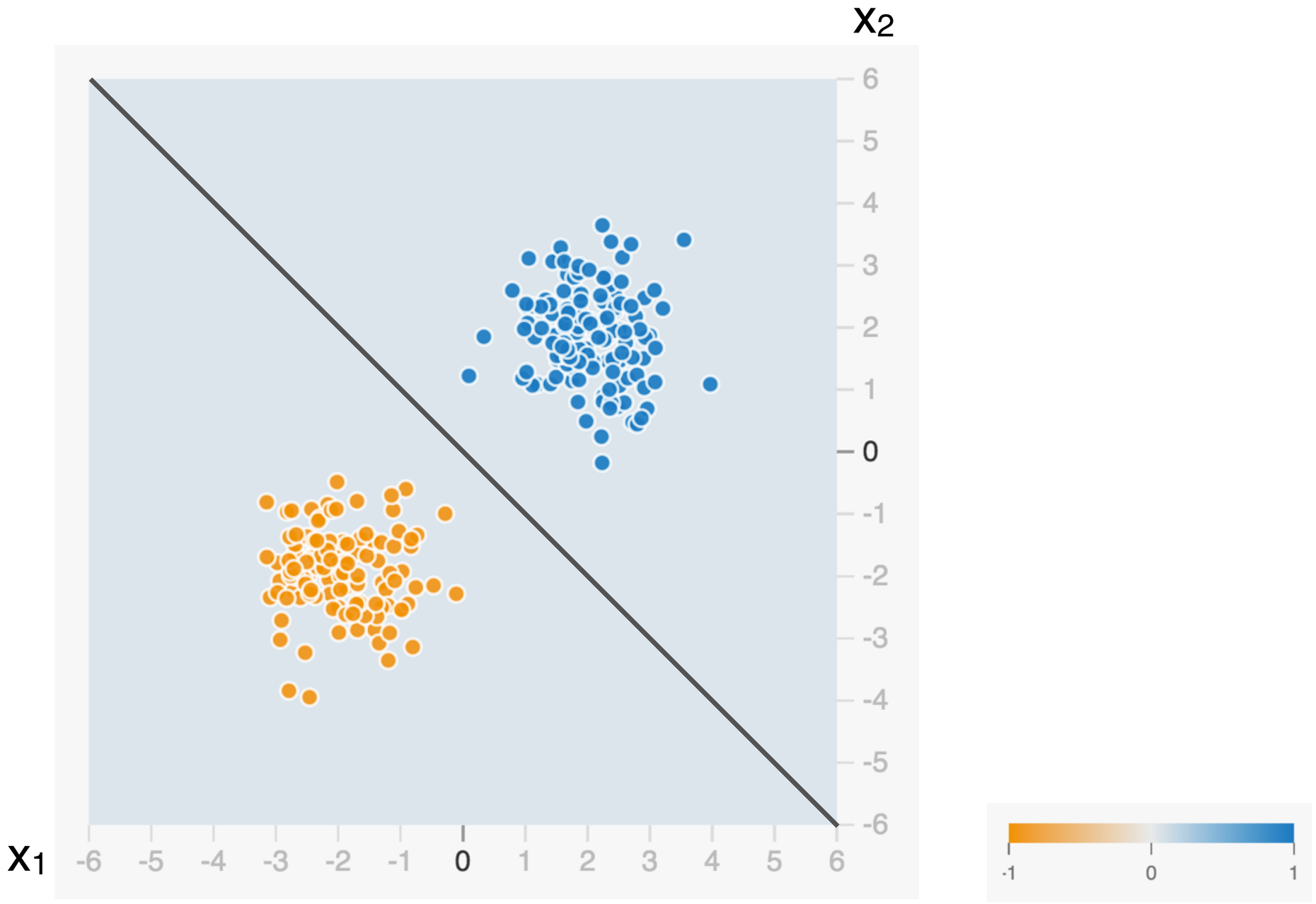
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Photo: NVIDIA Blog



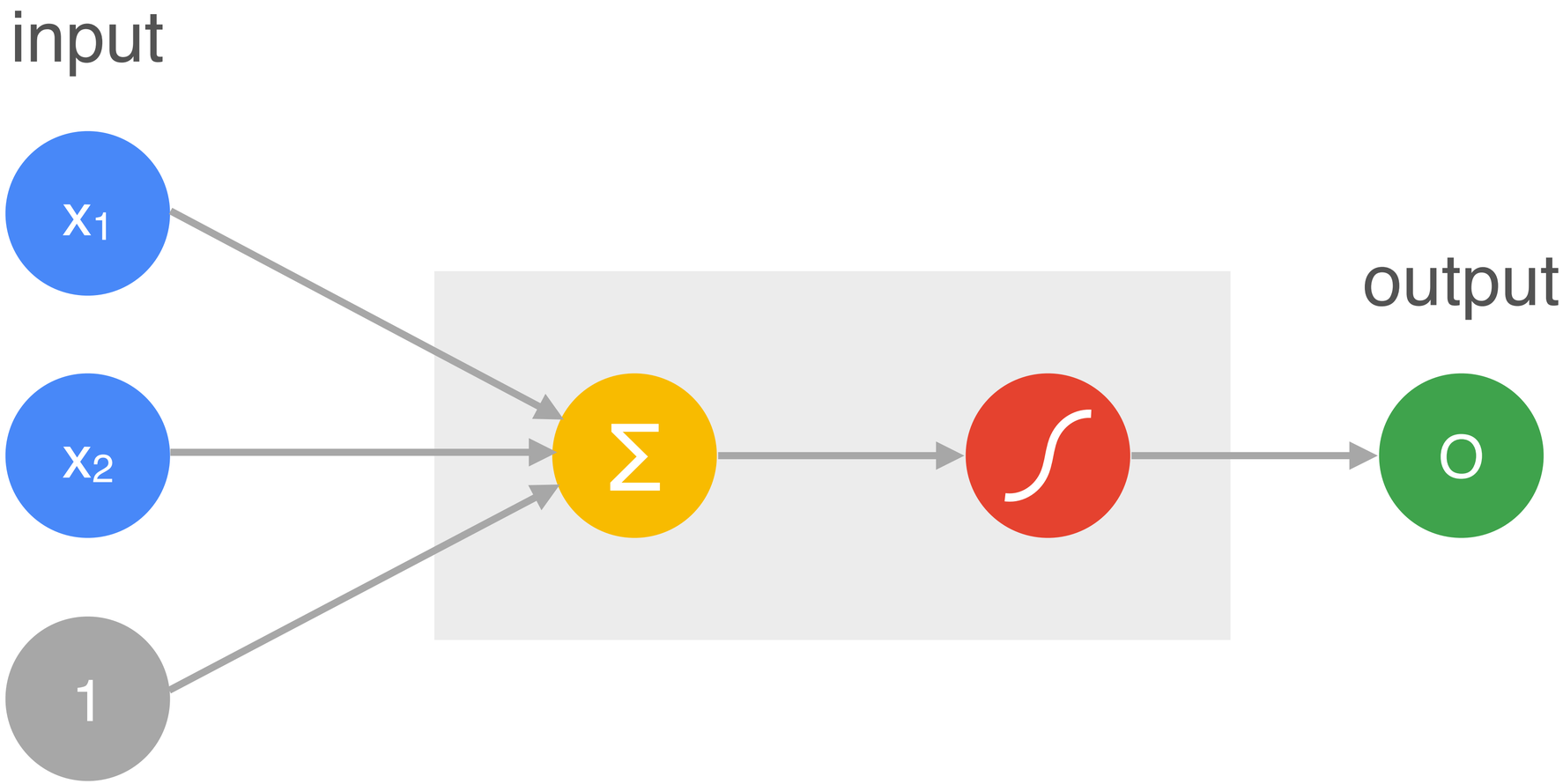
Source: Large-Scale Deep Learning for Intelligent Computer Systems, Jeff Dean, WSDM 2016


x_2
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
x_1
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
-1
0
1

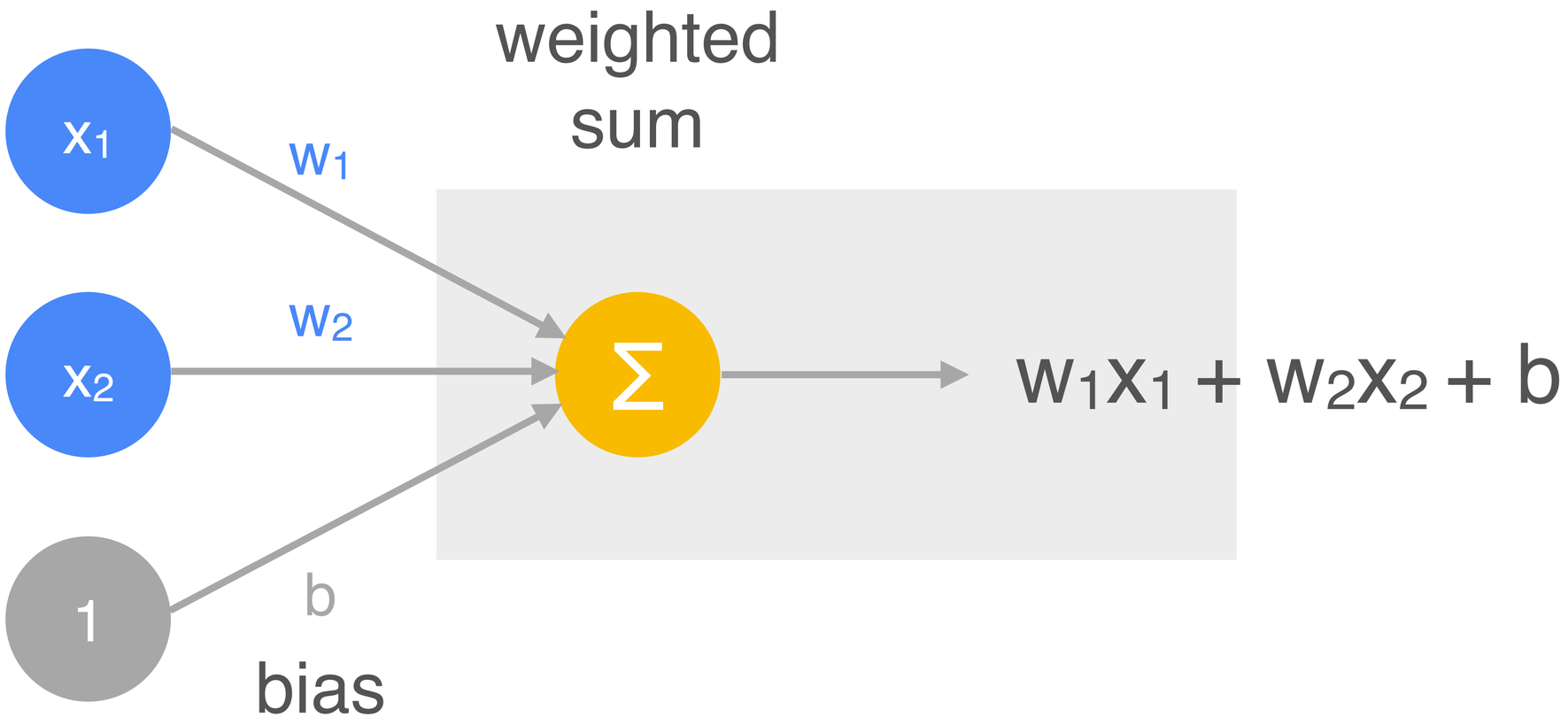

x_2
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
x_1
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
-1
0
1

A Perceptron

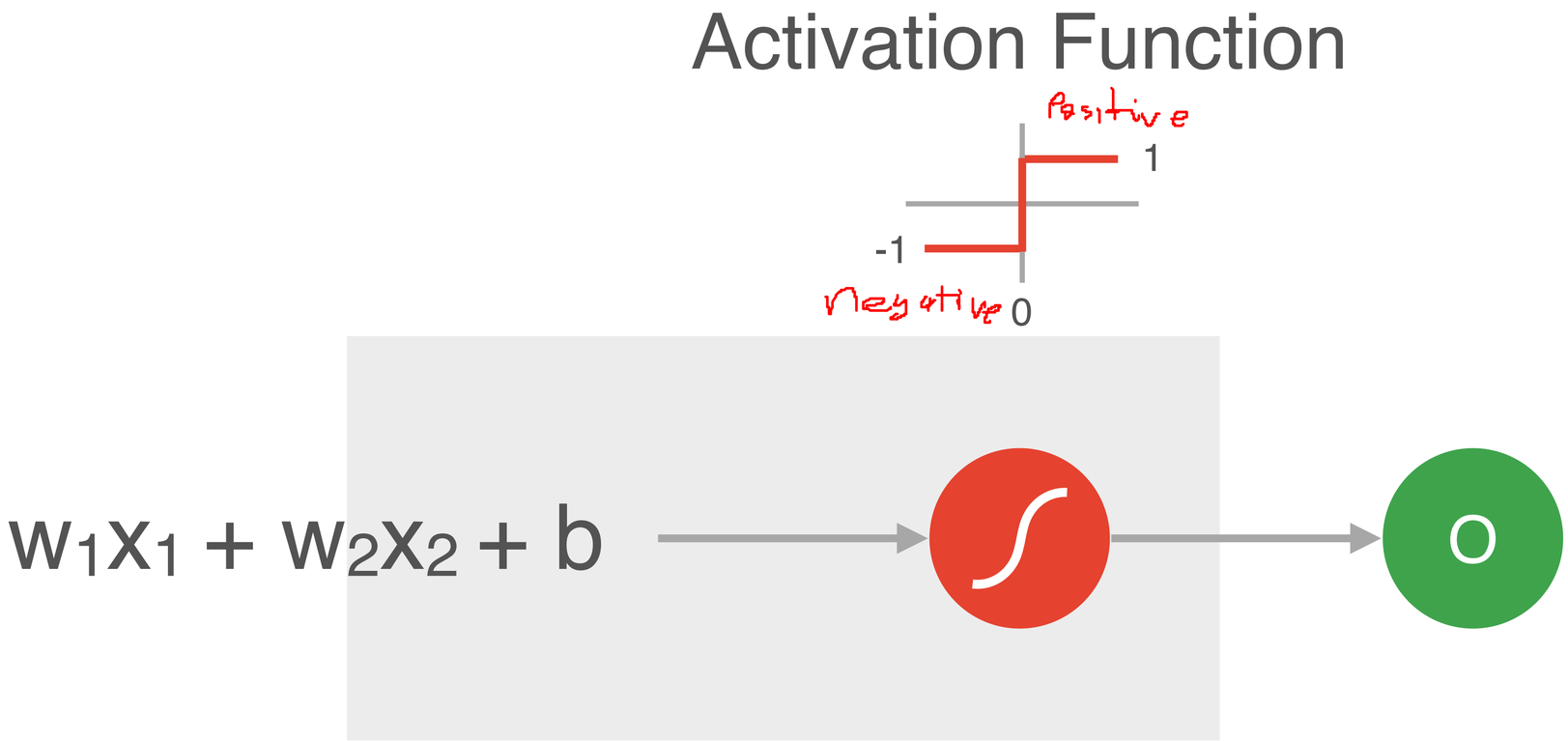
no hidden layer (1 layer)



A Perceptron



A Perceptron

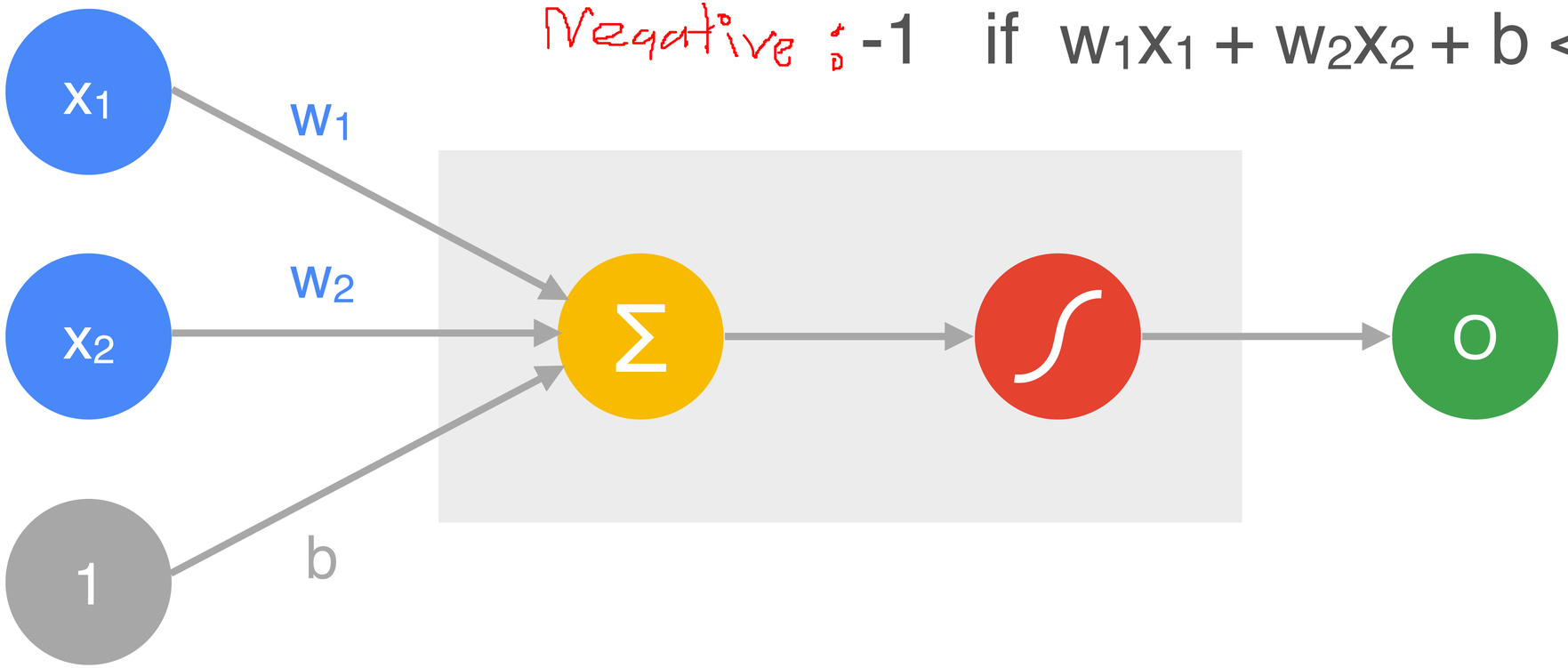


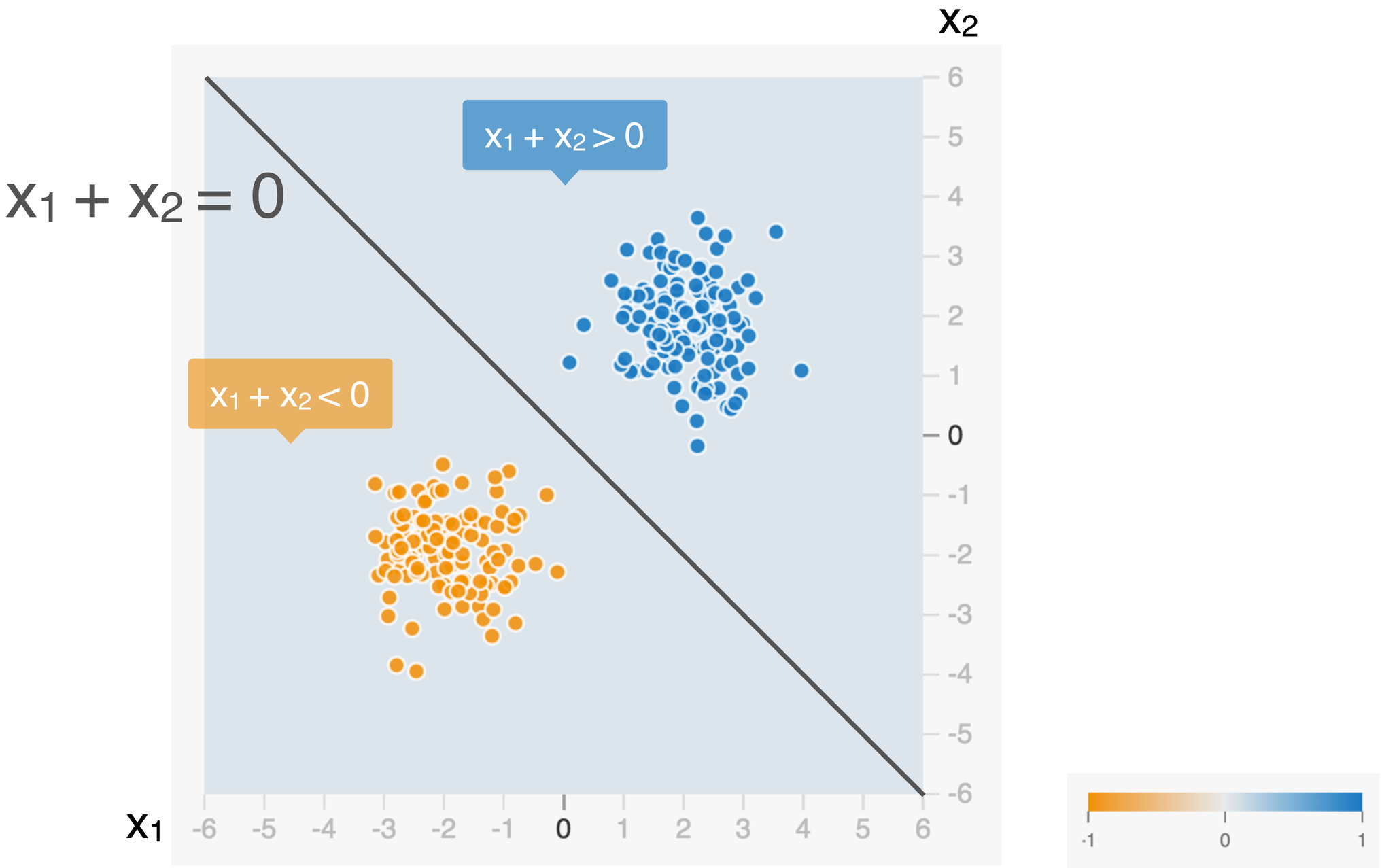
A Perceptron

Activation fn → other Activation
- Linear
- Sigmoid
- tanh
- ...

Positive : 1 if $w_1x_1 + w_2x_2 + b > 0$

Negative : -1 if $w_1x_1 + w_2x_2 + b < 0$



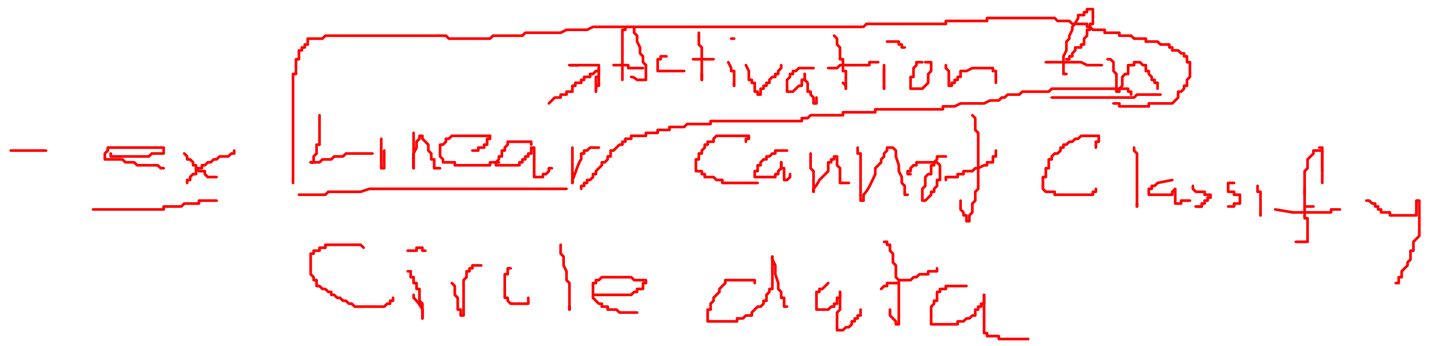


Demo

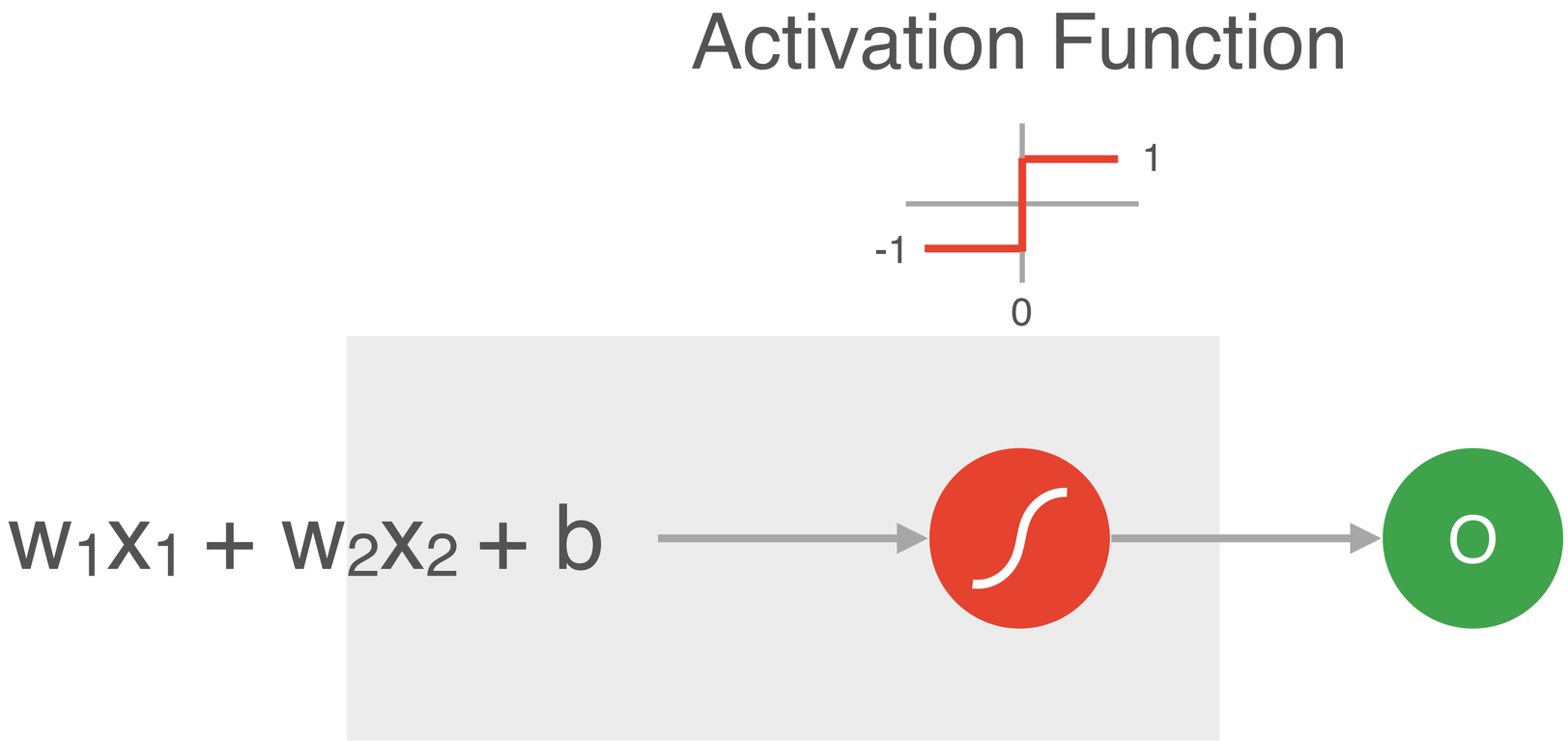
tensorflow playground

http://bit.ly/2sn7jrY

- ex Linear (activation fn) cannot classify Circle data

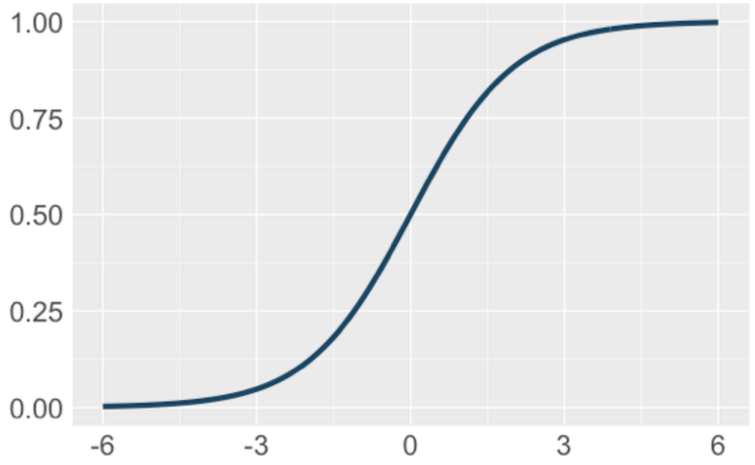


A Perceptron



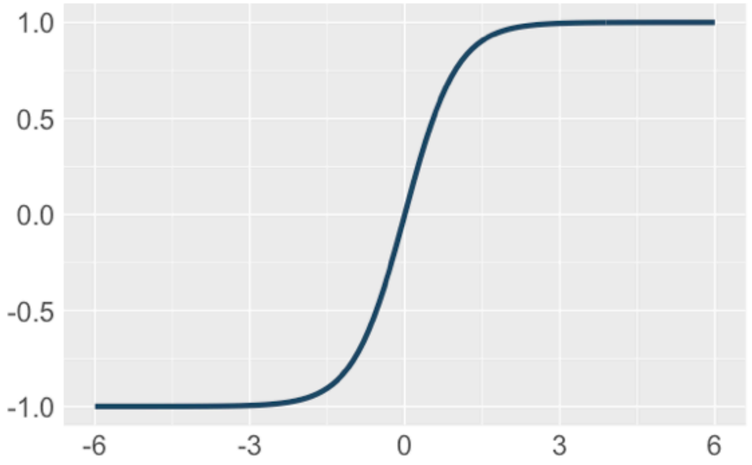
Other Activation Functions

Sigmoid



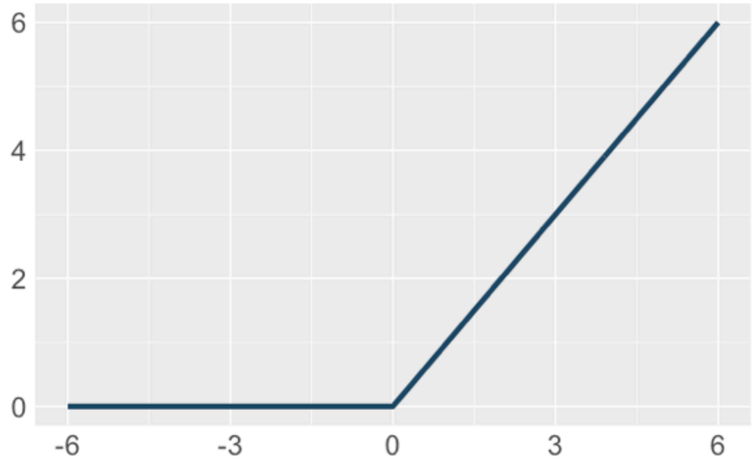
$$f(x) = \frac{1}{1 + e^{-x}}$$

TanH



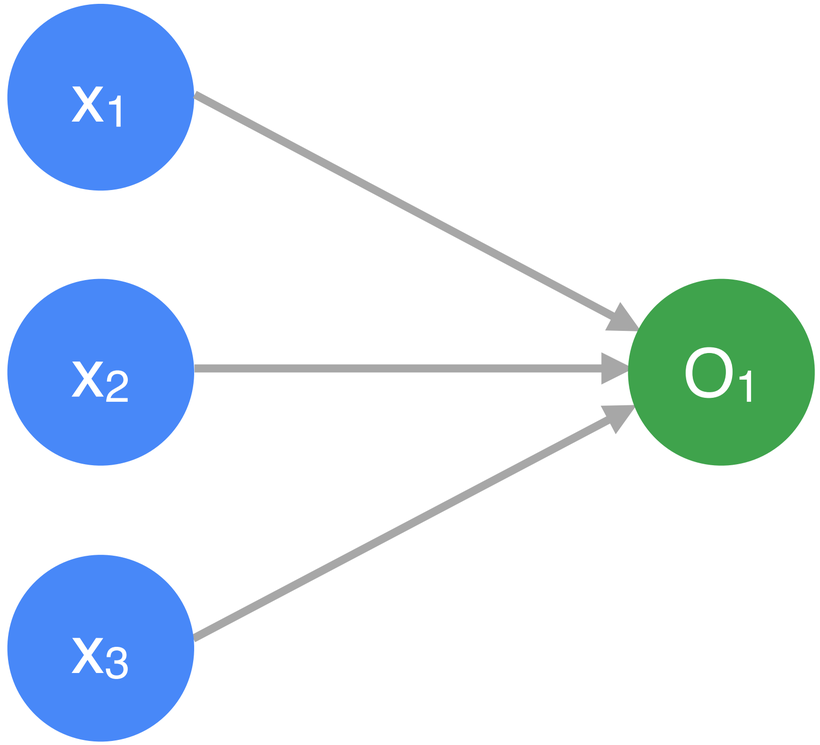
$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

ReLU

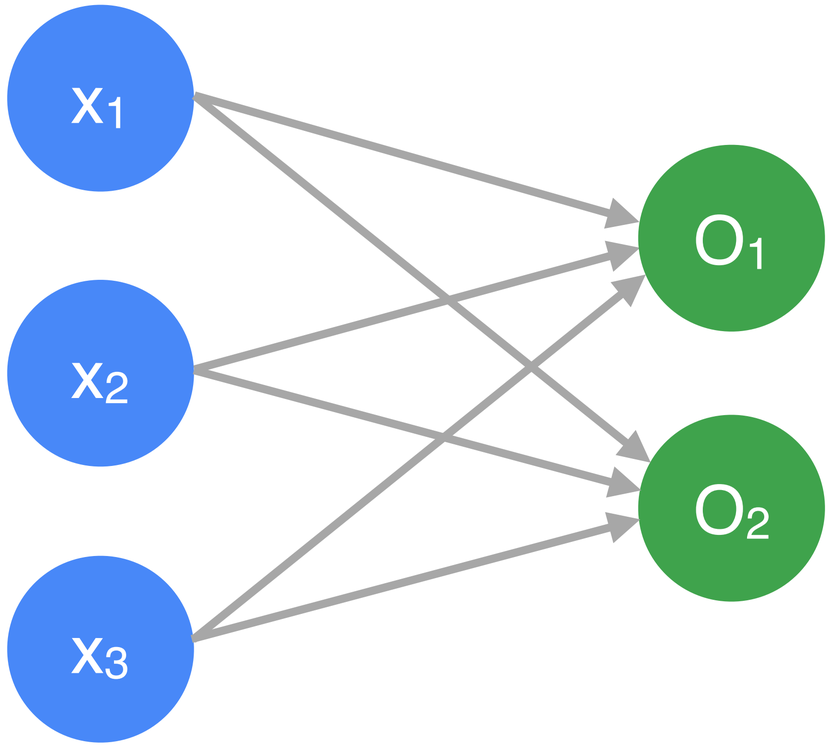


$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

From a Perceptron

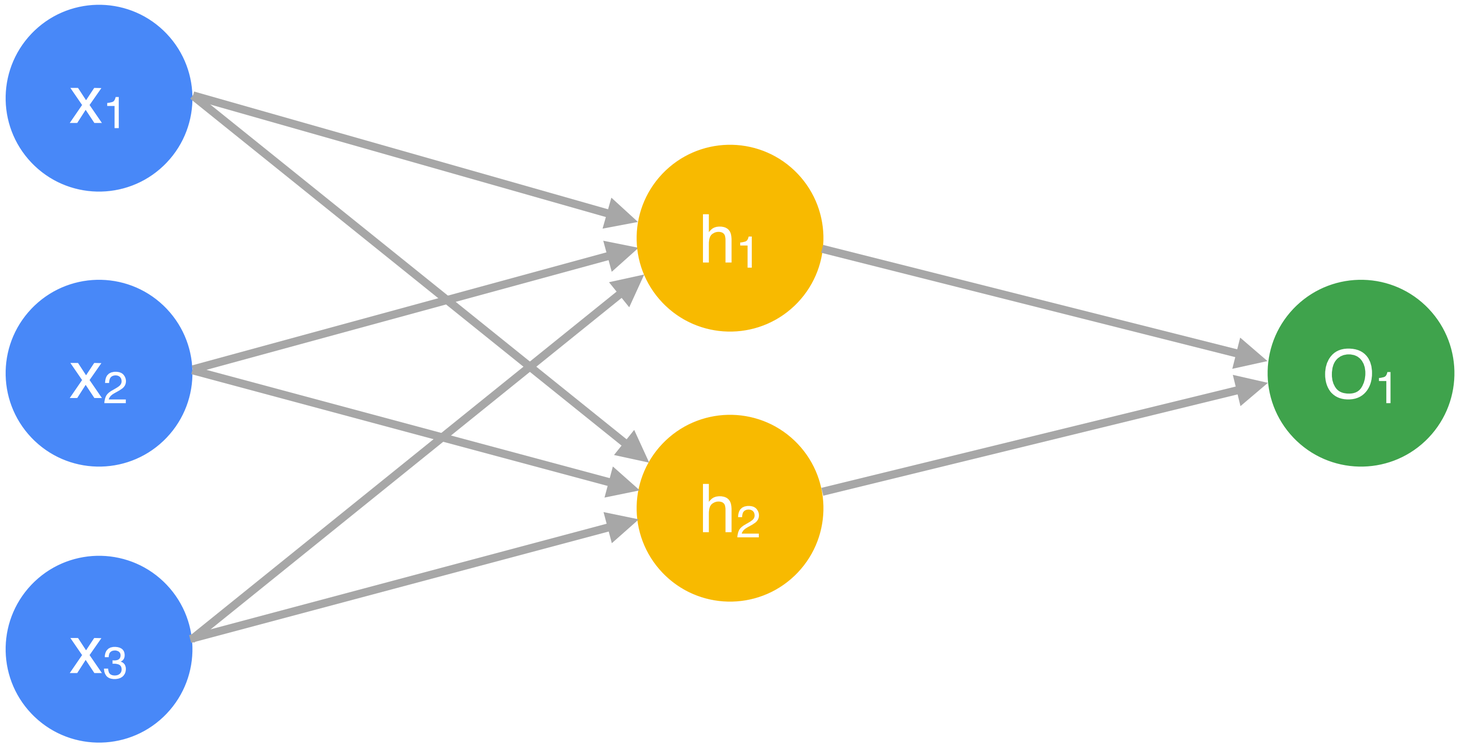


Multi-Output Perceptron

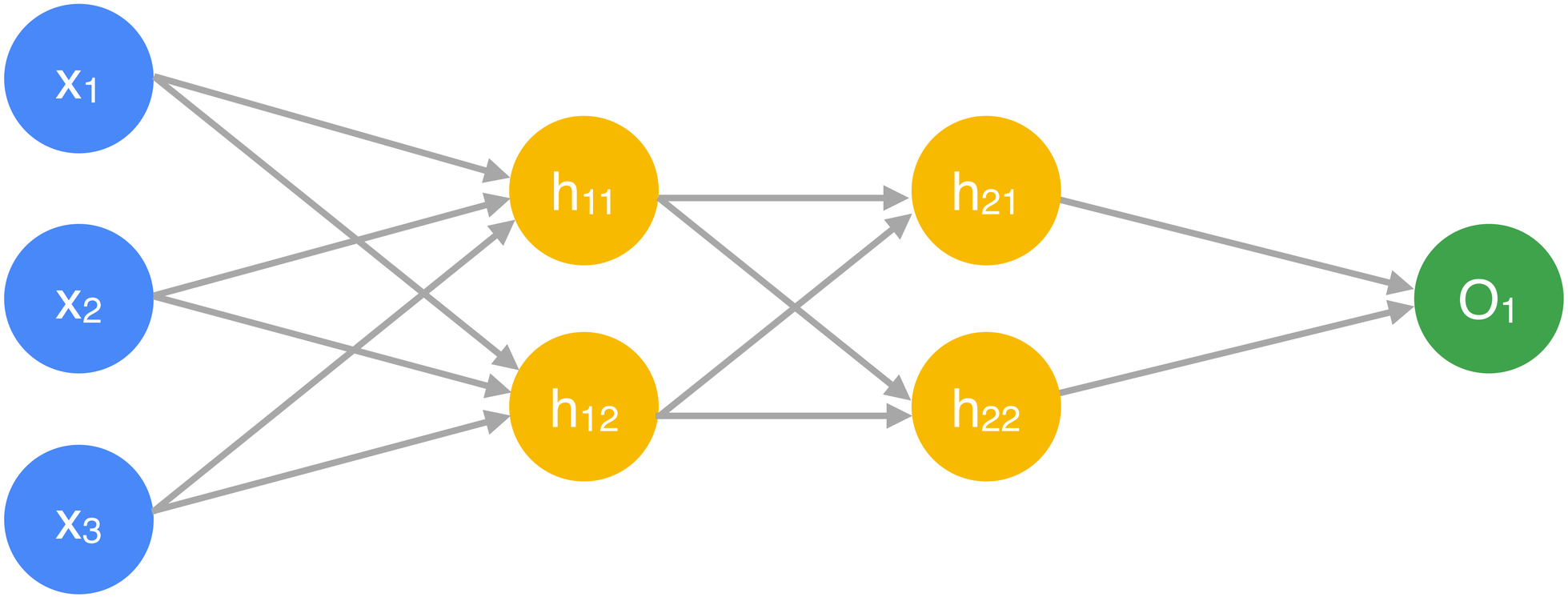


input layer output layer

Multi-Layer Perceptron



Deep Neural Network



input layer hidden layers output layer

Demo

http://bit.ly/2scieF6

Epoch
000,335

Learning rate: 0.03
Activation: Sigmoid
Regularization: None
Regularization rate: 0
Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$

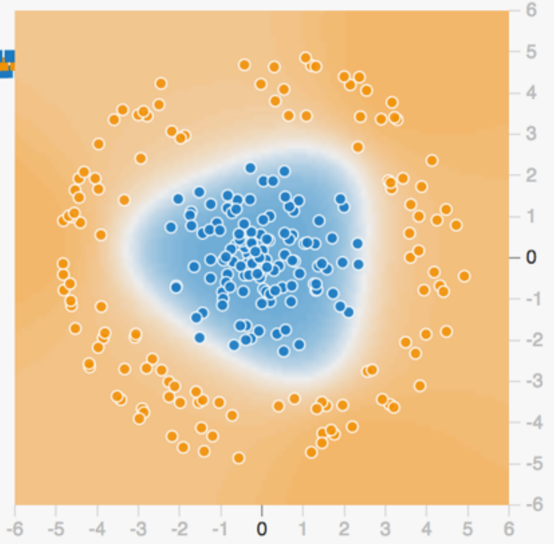
1 HIDDEN LAYER

3 neurons

*This is the output from one **neuron**. Hover to see it larger.*

OUTPUT

Test loss 0.040
Training loss 0.038



Colors shows data, neuron and weight values.

-1 0 1

Show test data Discretize output

Epoch
000,691

Learning rate
0.03

Activation
Linear

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$\sin(X_1)$

$\sin(X_2)$

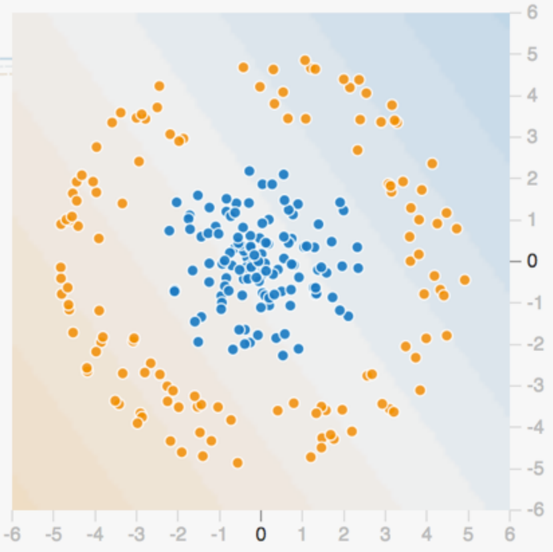
1 HIDDEN LAYER

3 neurons

*This is the output from one **neuron**. Hover to see it larger.*

OUTPUT

Test loss 0.510
Training loss 0.497



Colors shows data, neuron and weight values.

-1 0 1

Show test data

Discretize output

Epoch
000,279

Learning rate
0.03

Activation
ReLU

Regularization
None

Regularization rate
0

Problem type
Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$\sin(X_1)$

$\sin(X_2)$

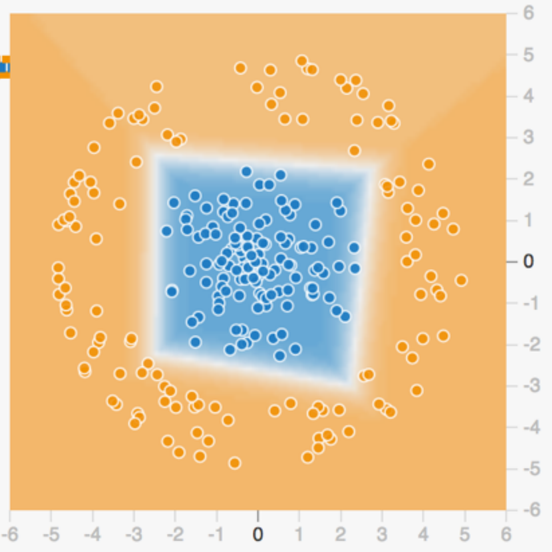
1 HIDDEN LAYER

3 neurons

*This is the output from one **neuron**. Hover to see it larger.*

OUTPUT

Test loss 0.013
Training loss 0.009

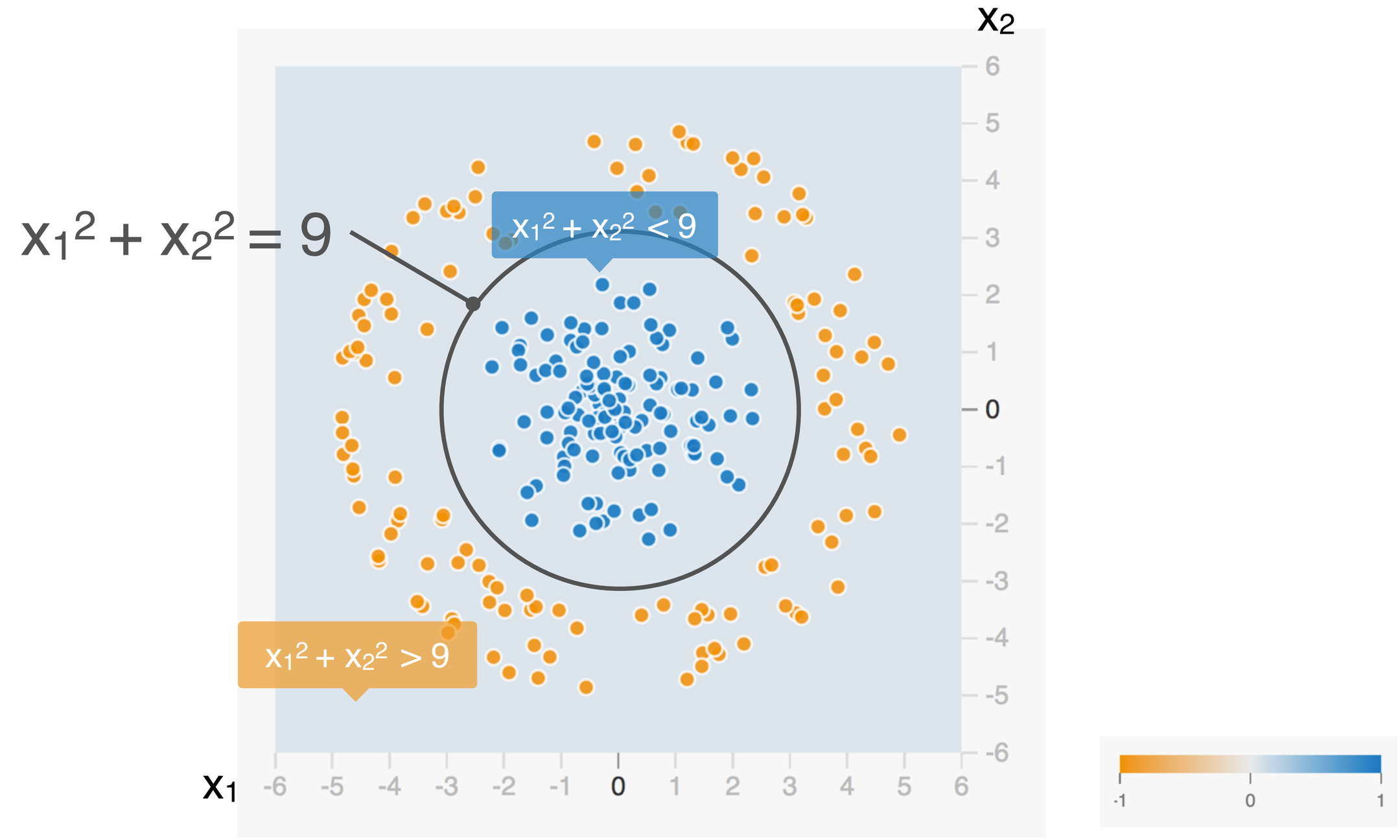


Colors shows data, neuron and weight values.

-1 0 1

Show test data

Discretize output



DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$\sin(X_1)$

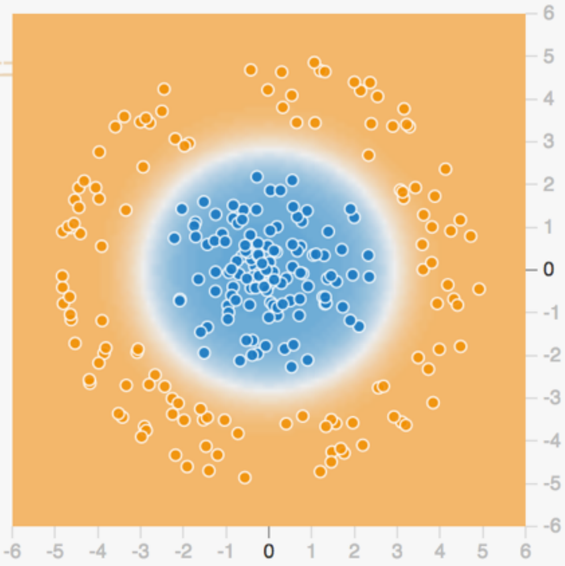
$\sin(X_2)$

+ − 0 HIDDEN LAYERS

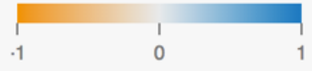
OUTPUT

Test loss 0.010

Training loss 0.009



Colors shows data, neuron and weight values.



☐ Show test data ☐ Discretize output

Demo

http://bit.ly/2rexPin

Epoch
001,297

Learning rate: 0.03

Activation: ReLU

Regularization: None

Regularization rate: 0

Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$\sin(X_1)$

$\sin(X_2)$

3 HIDDEN LAYERS

8 neurons

8 neurons

5 neurons

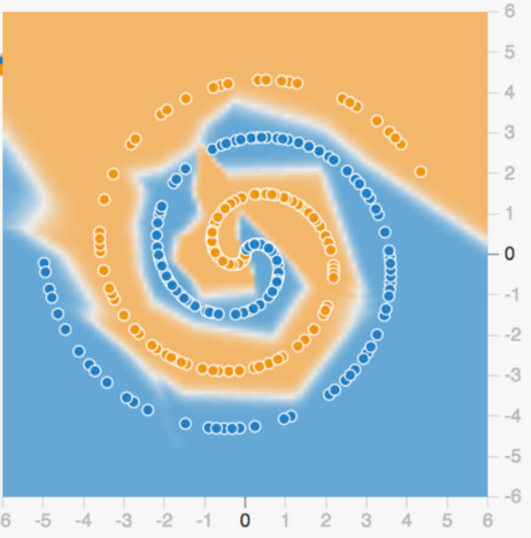
This is the output

The outputs are mixed with varying

OUTPUT

Test loss 0.066

Training loss 0.009



Colors shows data, neuron and weight values.

-1 0 1

Show test data

Discretize output

Why Deep Learning?

- Traditional machine learning requires feature engineering, and models may have simplifying assumptions
- Deep learning models are very flexible
- But…it requires a lot of data (and CPUs/GPUs)

Training Neural Networks

Pick an objective / loss function

Cross entropy → Classification

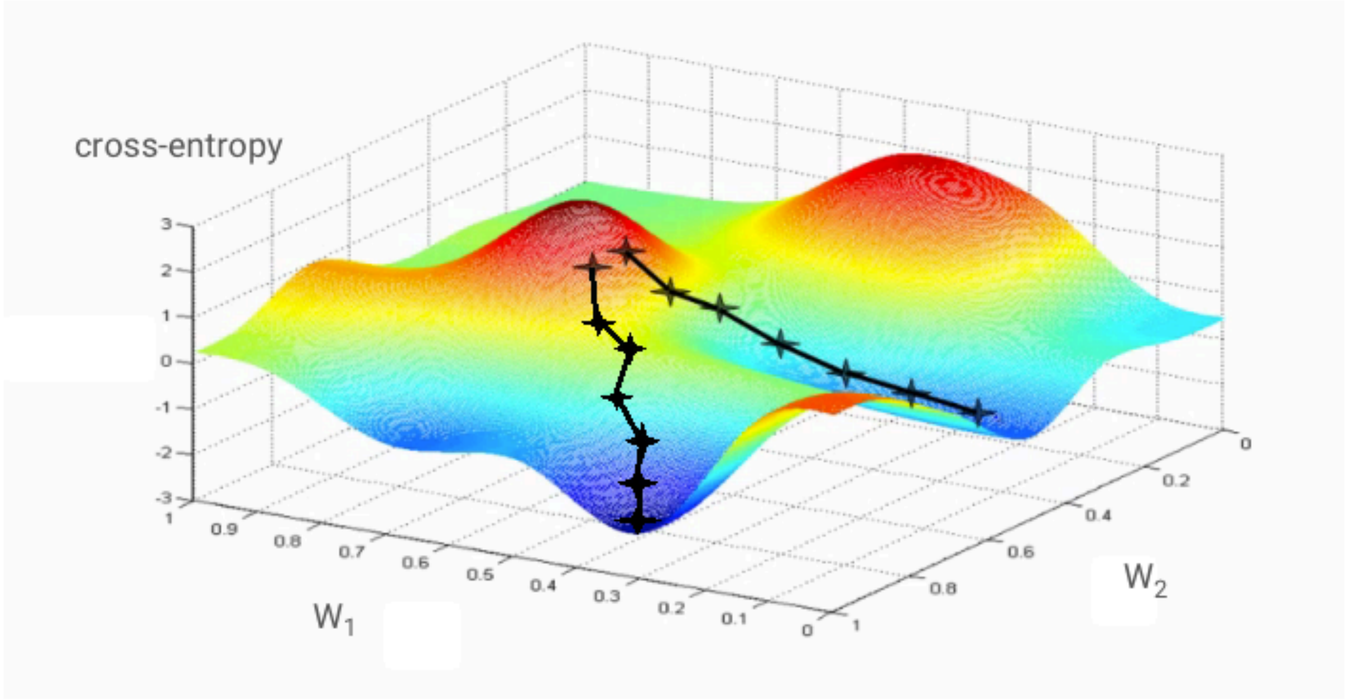
$$-\sum Y_i' . log(Y_i)$$

Mean Squared Error (MSE) → Regression

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Training Neural Networks

- Minimize the objective function using Gradient Descent



Source: Google Codelabs

Back-propagation

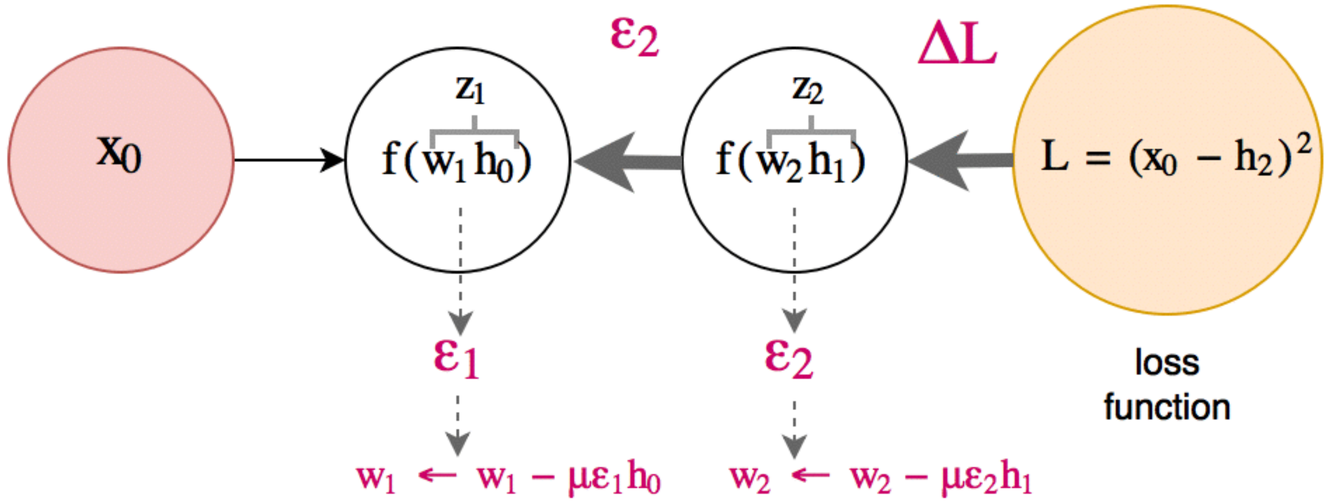
- Use backpropagation to compute gradient

Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial W_2}$$

Source: http://introtodeeplearning.com/ (Lecture 1)

Back-propagation

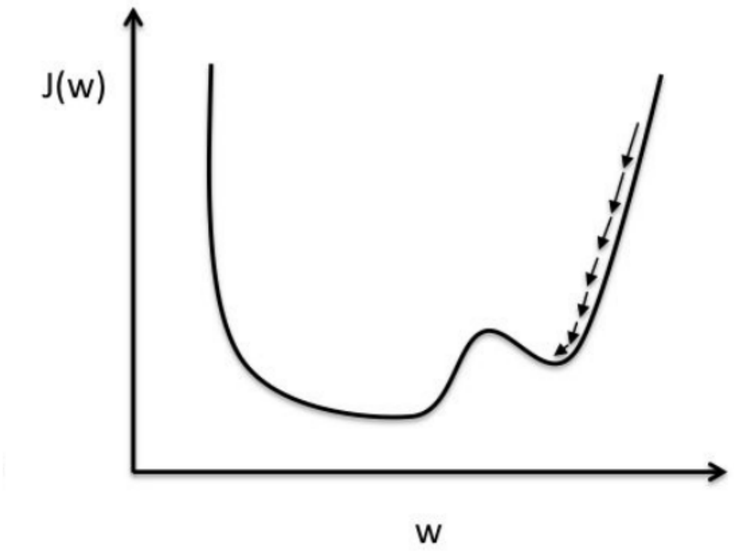


Source: https://alonalj.github.io/2016/12/10/What-is-Backpropagation/

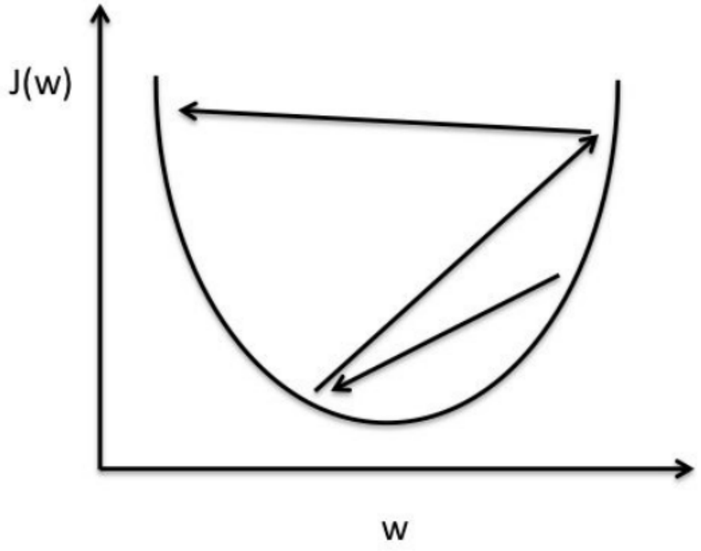
Challenges

- Learning rate
- Minibatches vs. Stochastic Gradient Descent
- Avoid overfitting (regularization): Dropout, Early stopping, etc.

Learning Rate



Small learning rate: Many iterations until convergence and trapping in local minima.

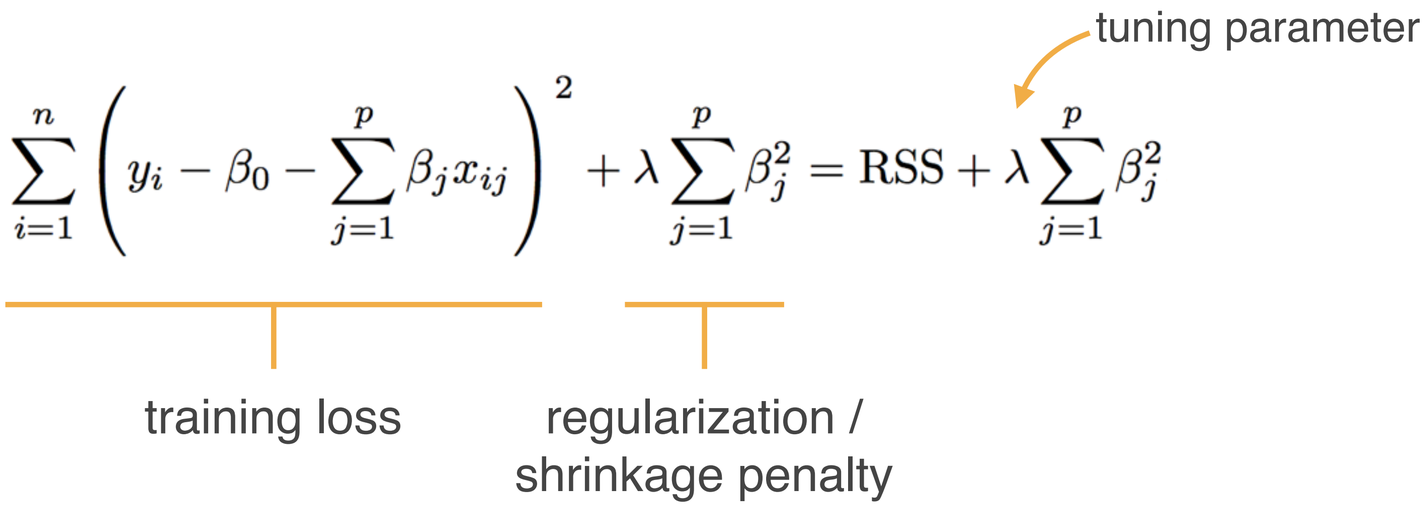


Large learning rate: Overshooting.

Source: http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

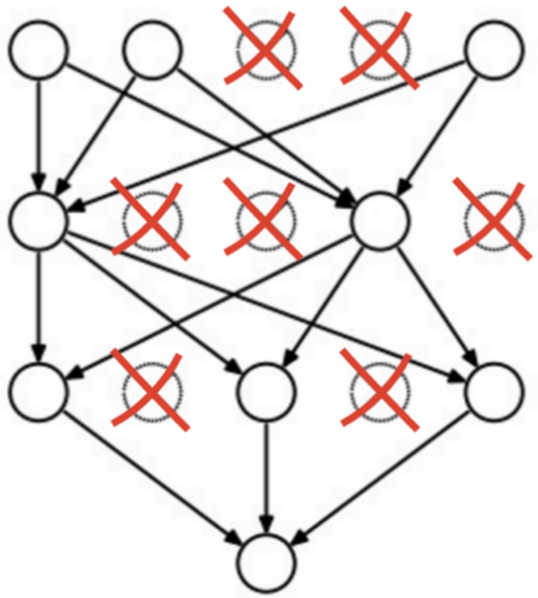
Regularization

- Shrink the coefficient estimates towards zero to reduce the variance
- Penalize model complexity
- Example: Ridge Regression

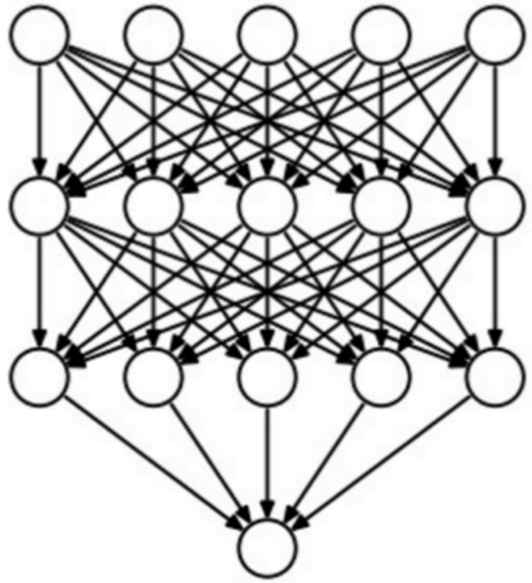


Dropout

Random dropout node and compare with test to reduce overfit and check error



Training
pkeep = 0.75



Test
pkeep = 1.0

Source: Google Codelabs