

WORKSHOP SERIES

Deep Learning

with TensorFlow



DEEP LEARNING 101

Ta Virot Chiraphadhanakul, PhD

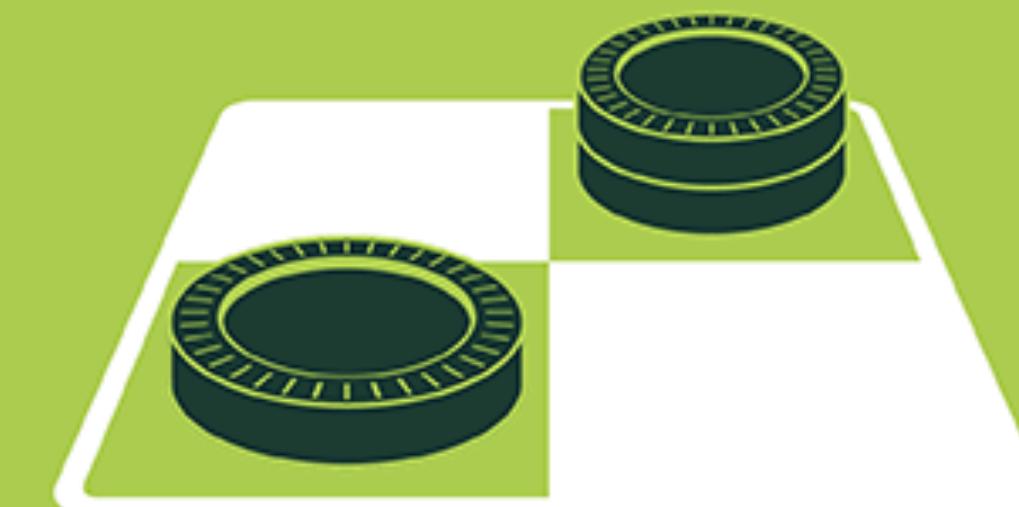
“ Artificial Intelligence is the
new electricity.”

— Andrew Ng

Deep Learning

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



1950's

1960's

1970's

1980's

1990's

2000's

2010's

MACHINE LEARNING

Machine learning begins to flourish.



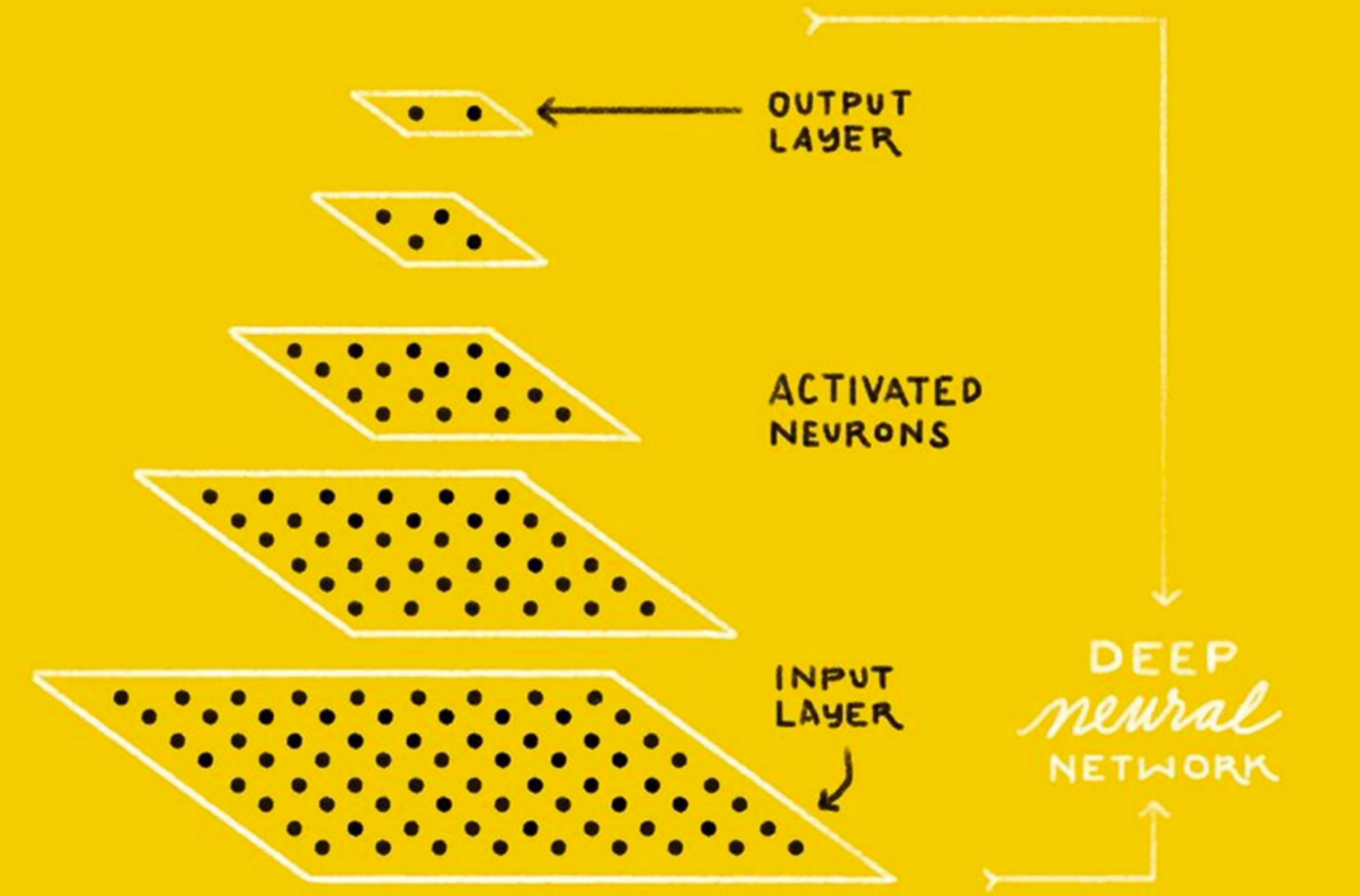
DEEP LEARNING

Deep learning breakthroughs drive AI boom.

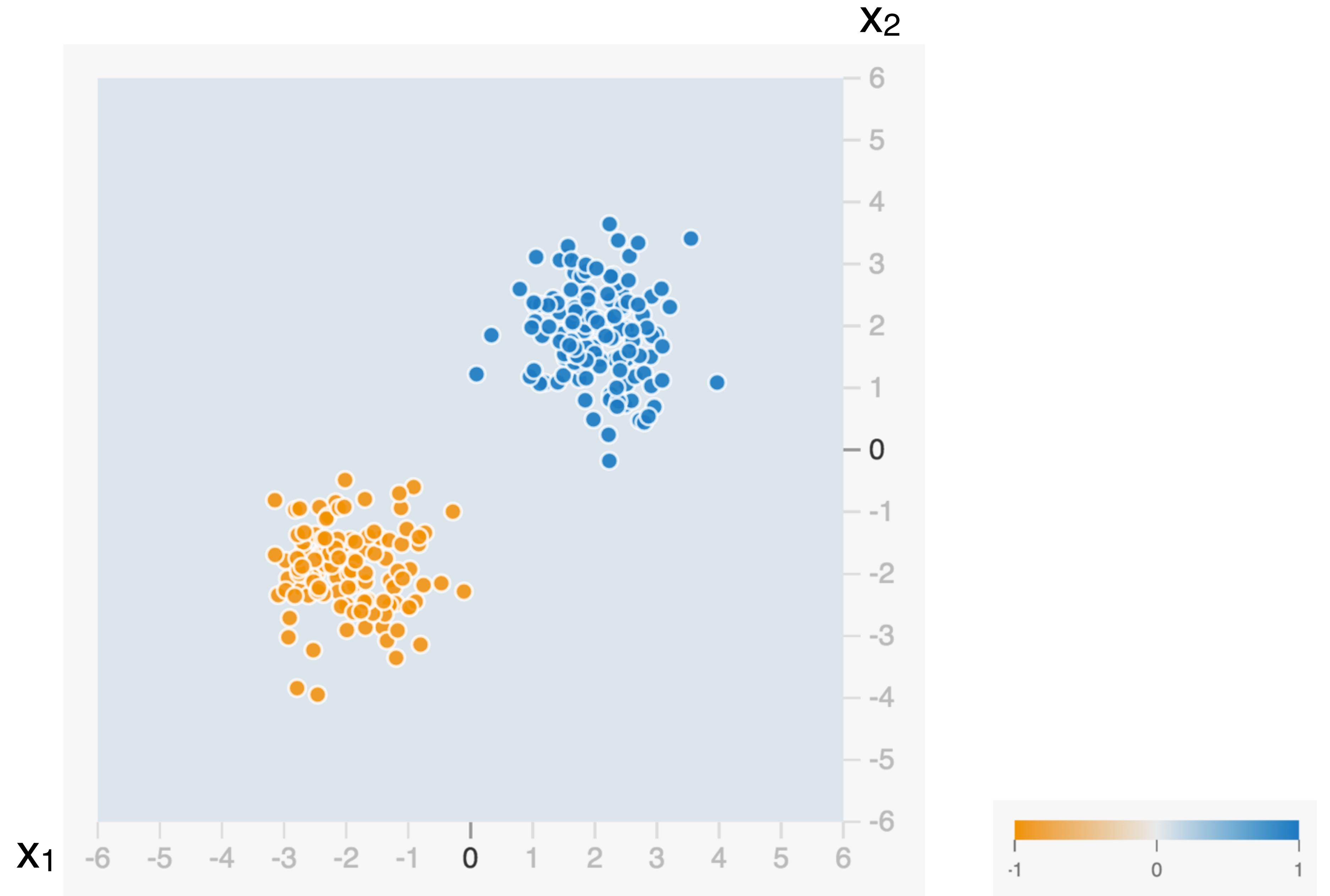


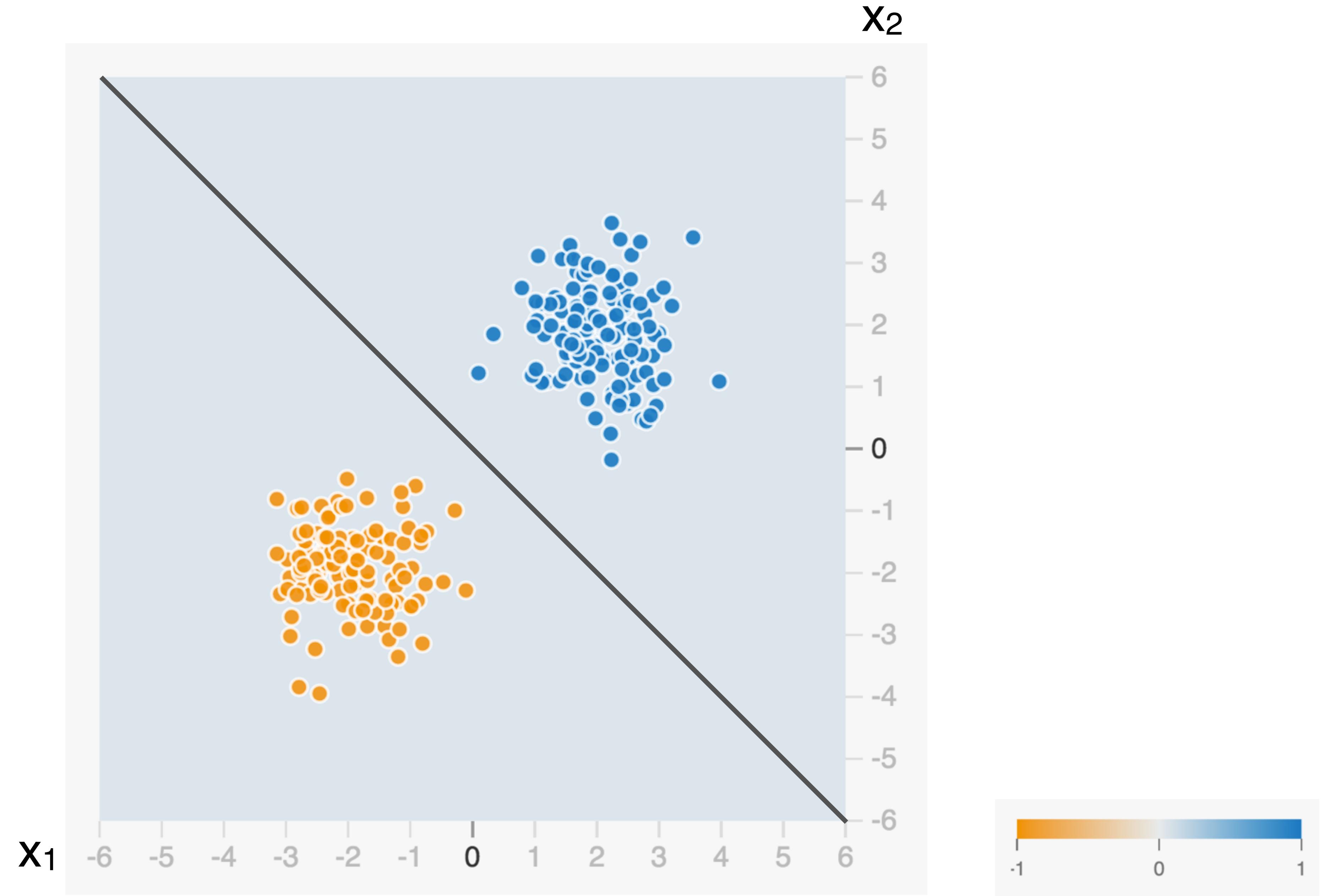
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

IS THIS A
CAT or DOG?

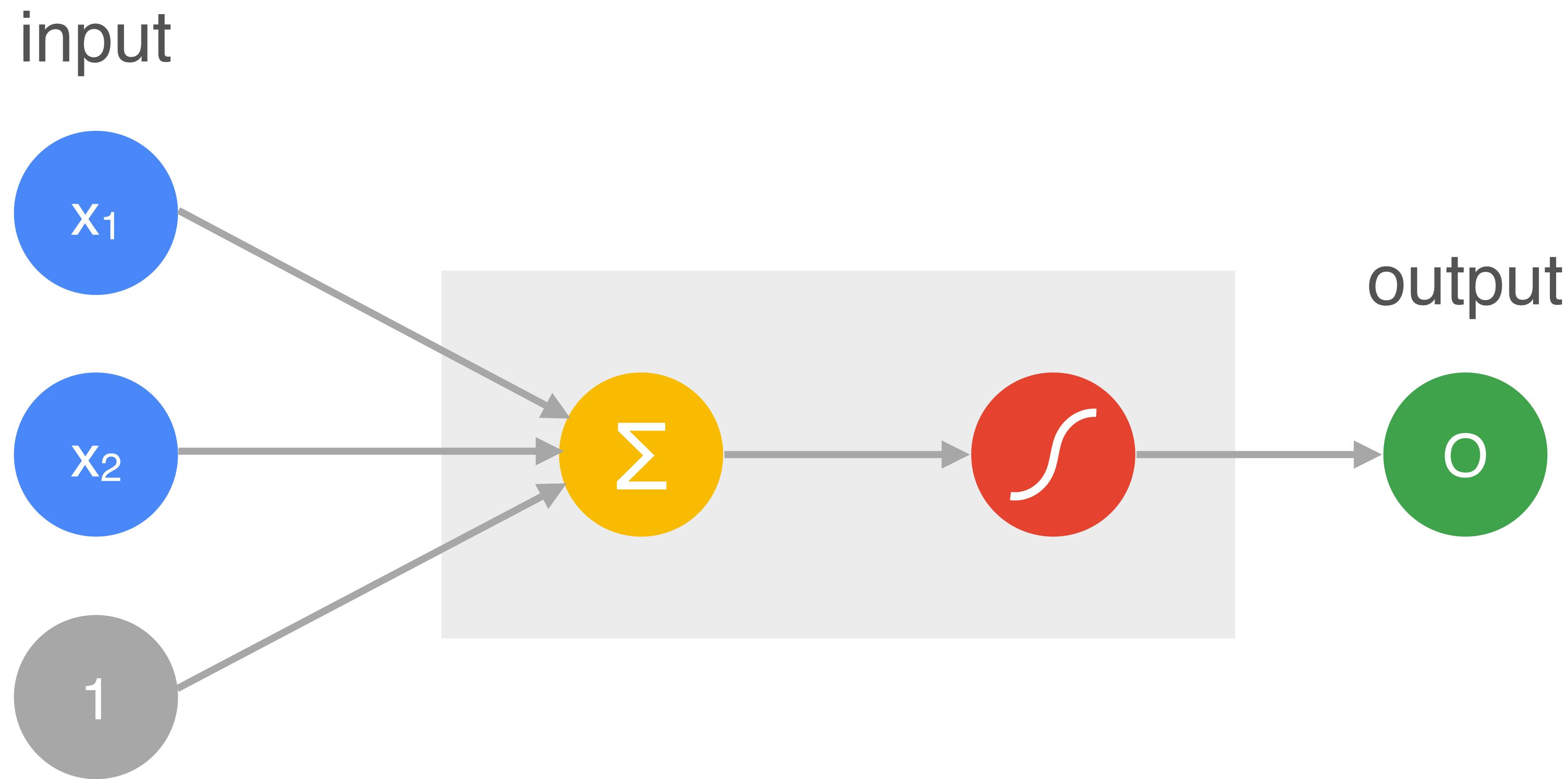


Source: Large-Scale Deep Learning for Intelligent Computer Systems, Jeff Dean, WSDM 2016

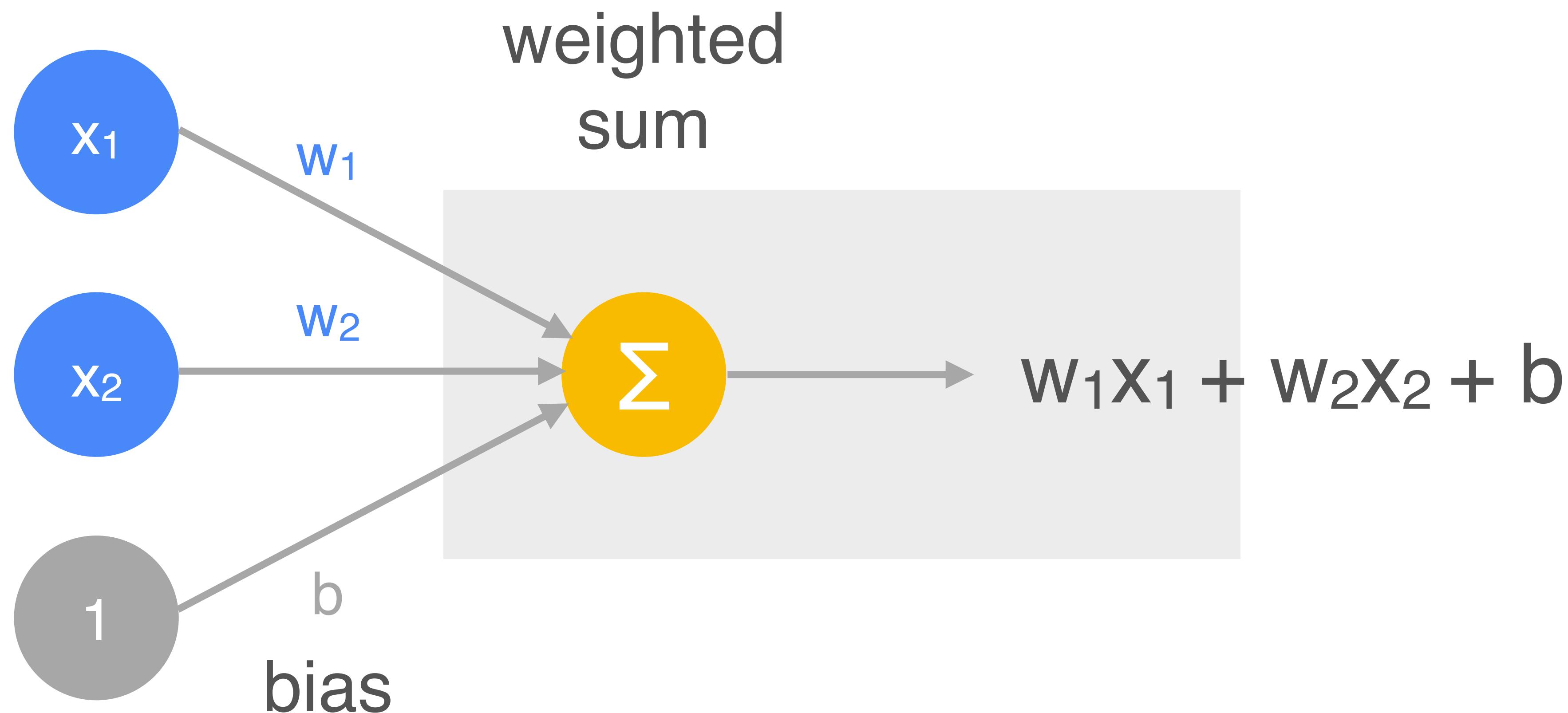




A Perceptron

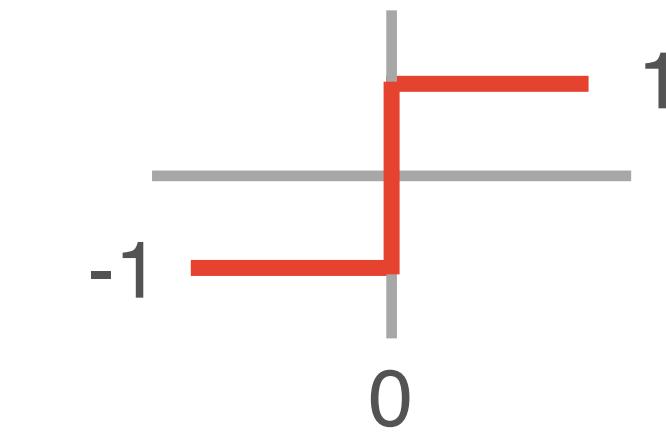


A Perceptron

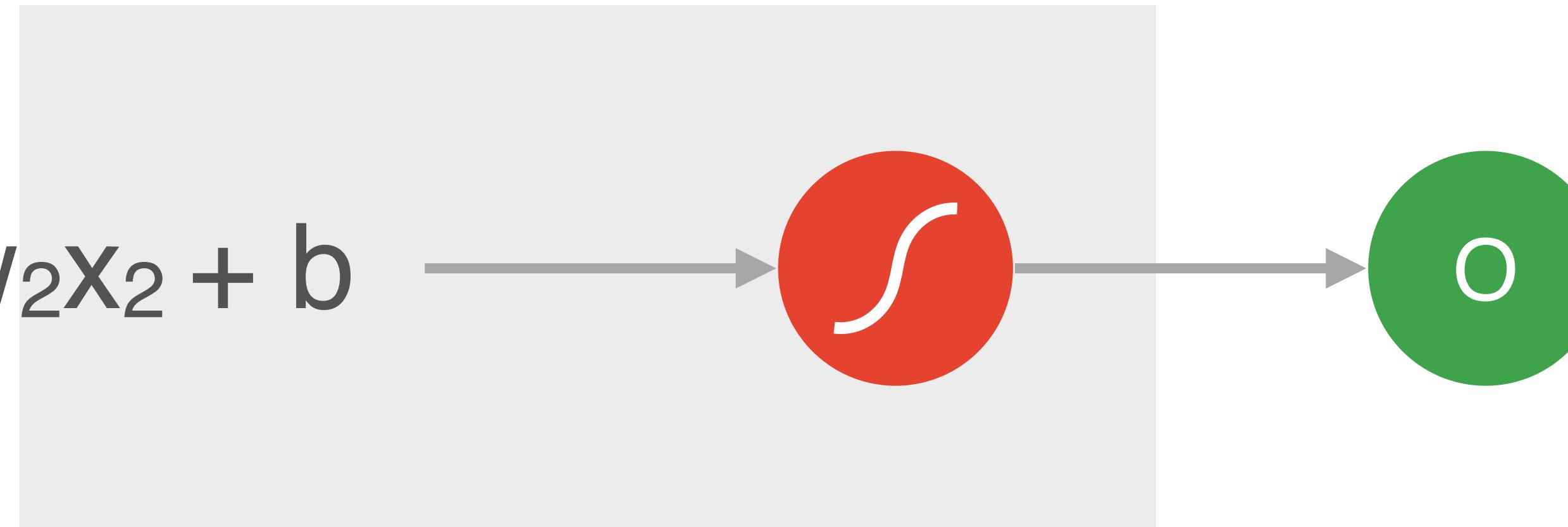


A Perceptron

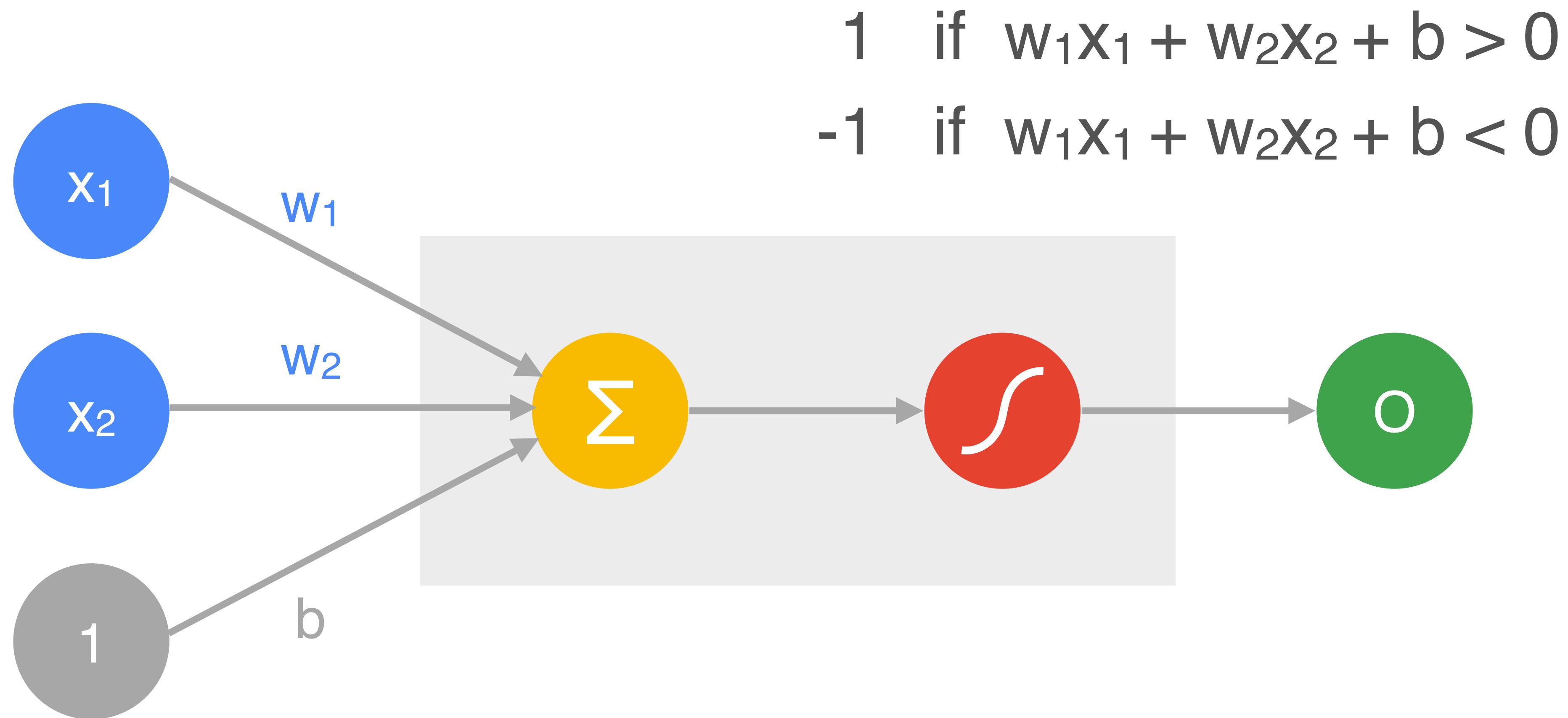
Activation Function

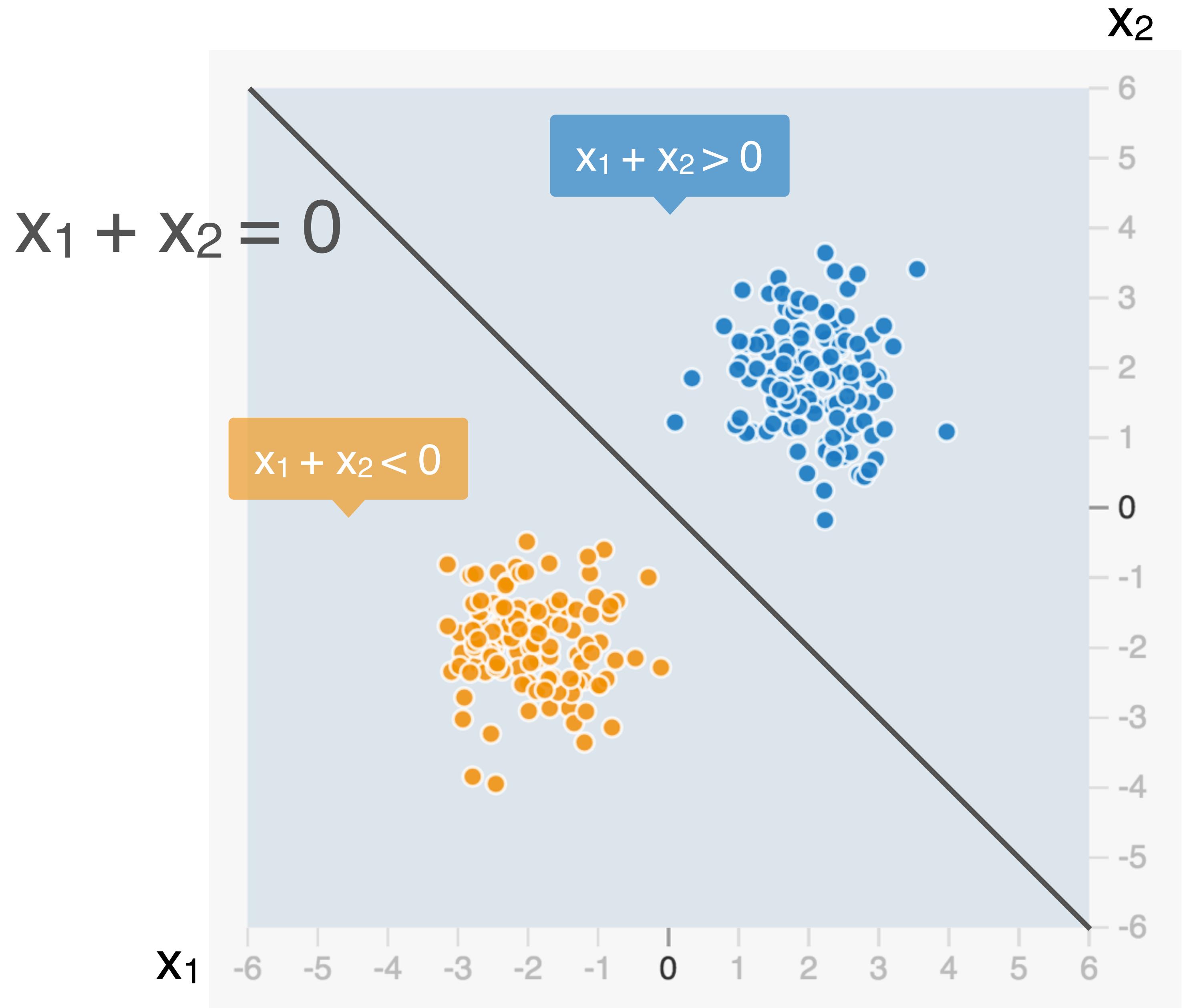


$w_1x_1 + w_2x_2 + b$



A Perceptron



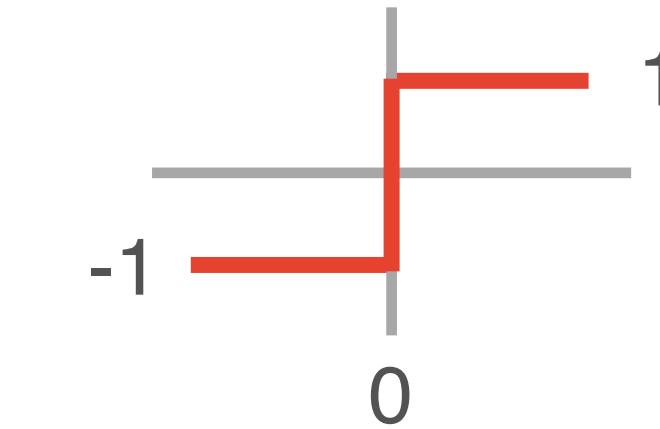


Demo

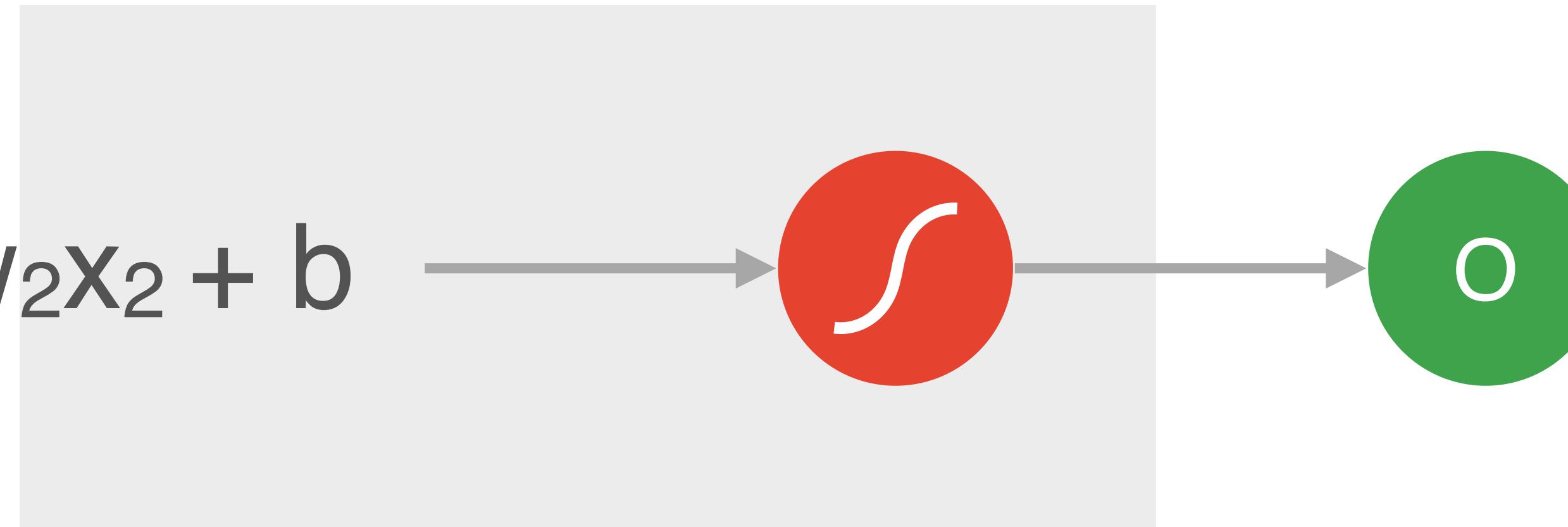
<http://bit.ly/2sn7jrY>

A Perceptron

Activation Function

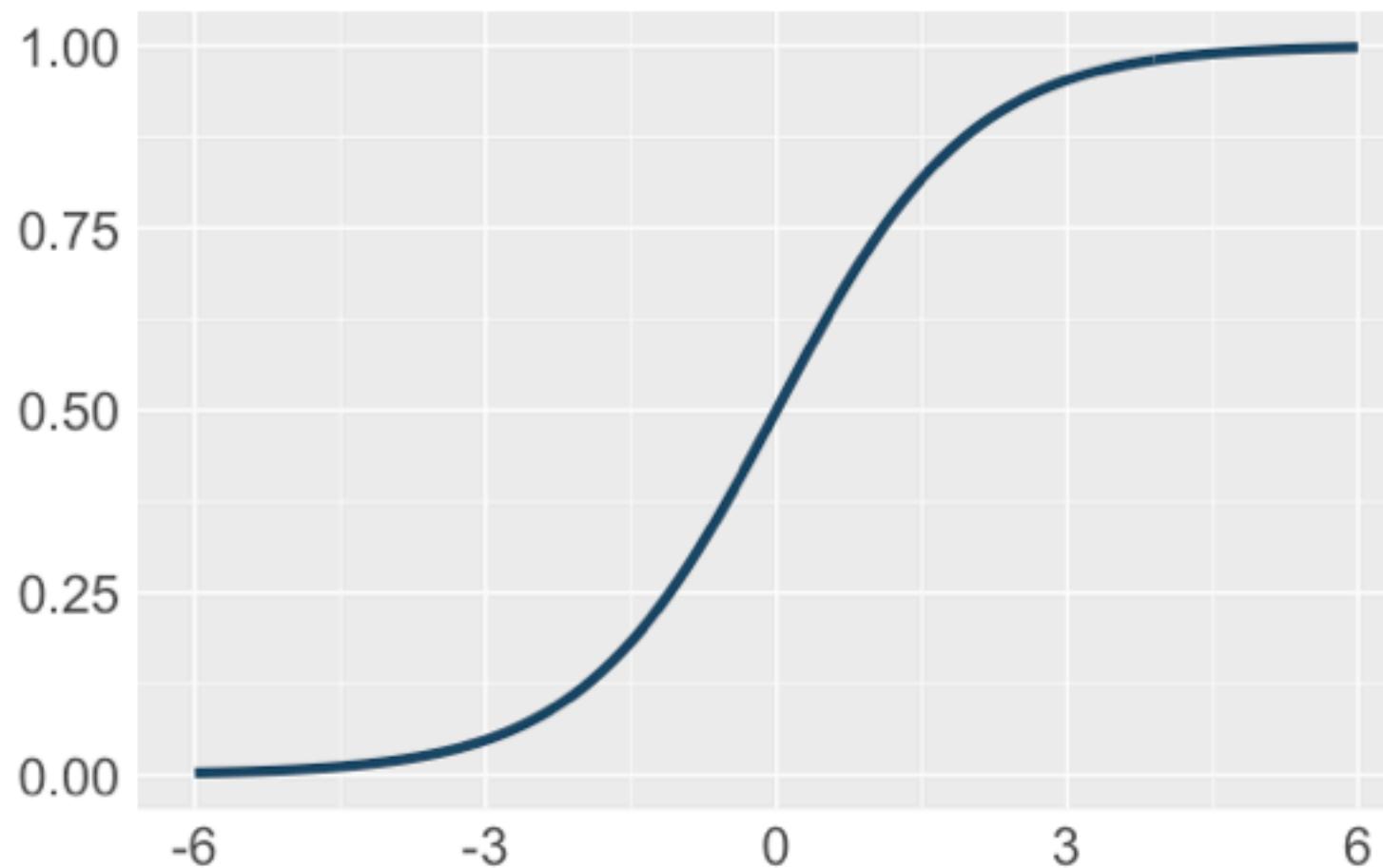


$w_1x_1 + w_2x_2 + b$



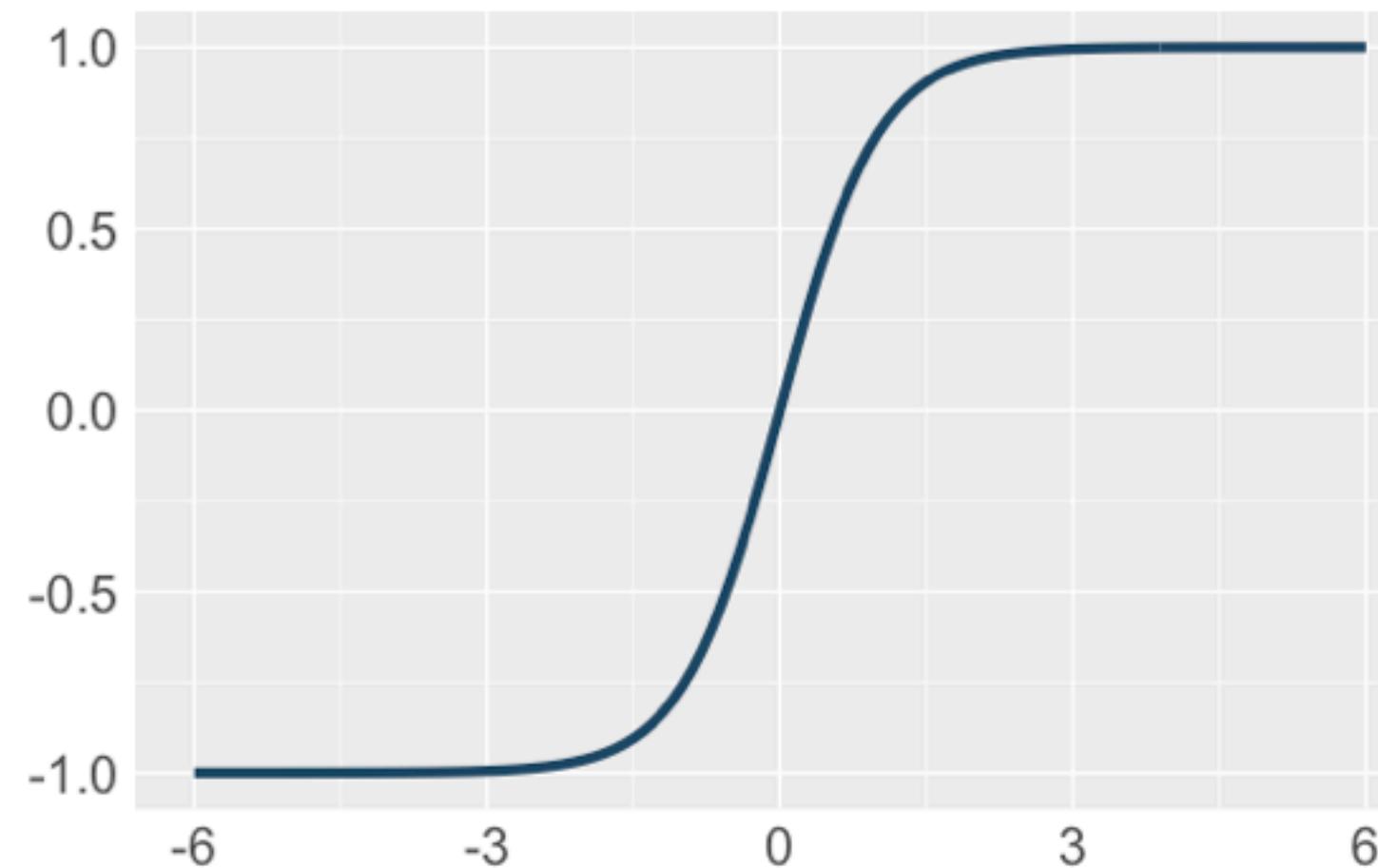
Other Activation Functions

Sigmoid



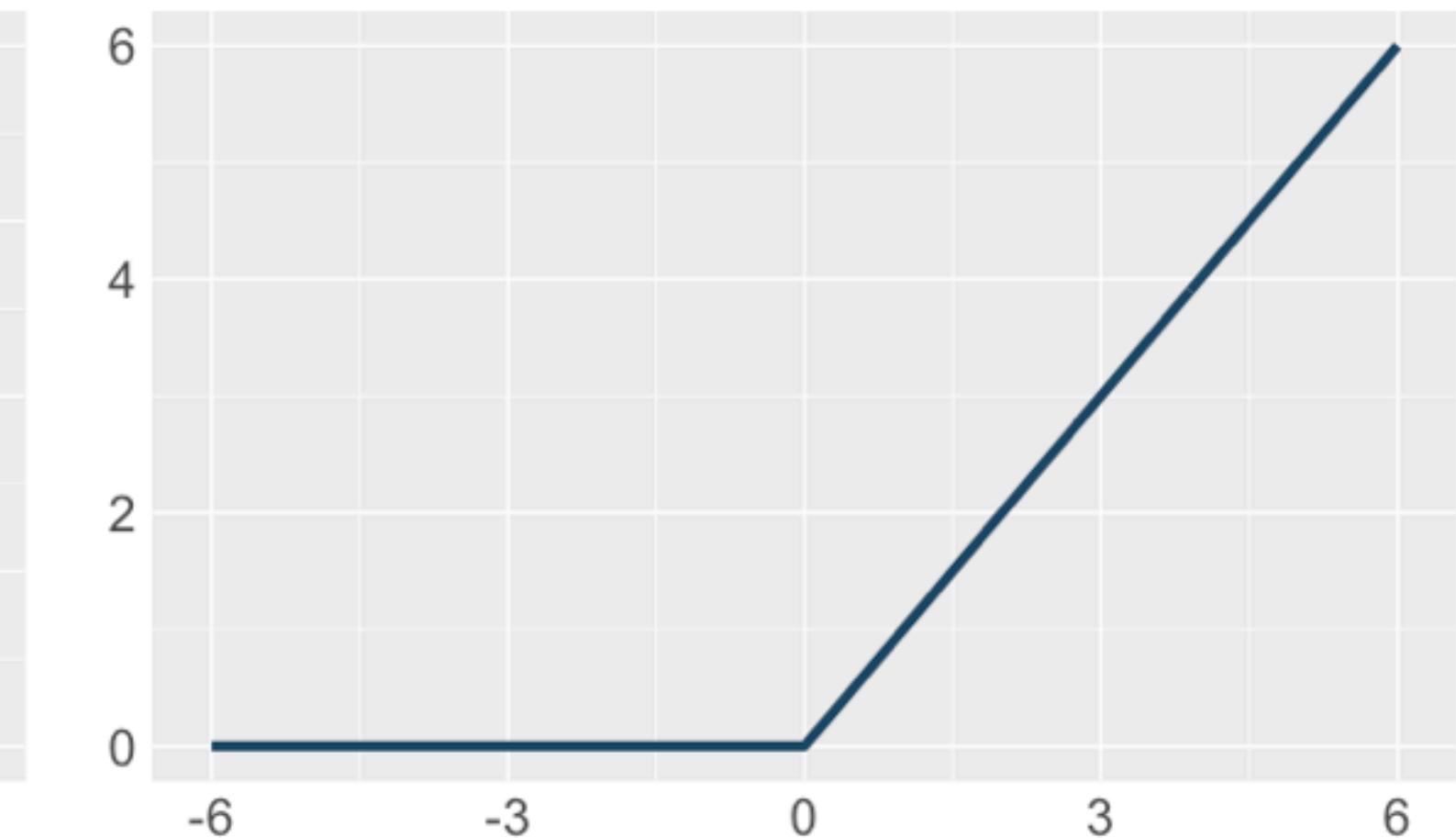
$$f(x) = \frac{1}{1 + e^{-x}}$$

TanH



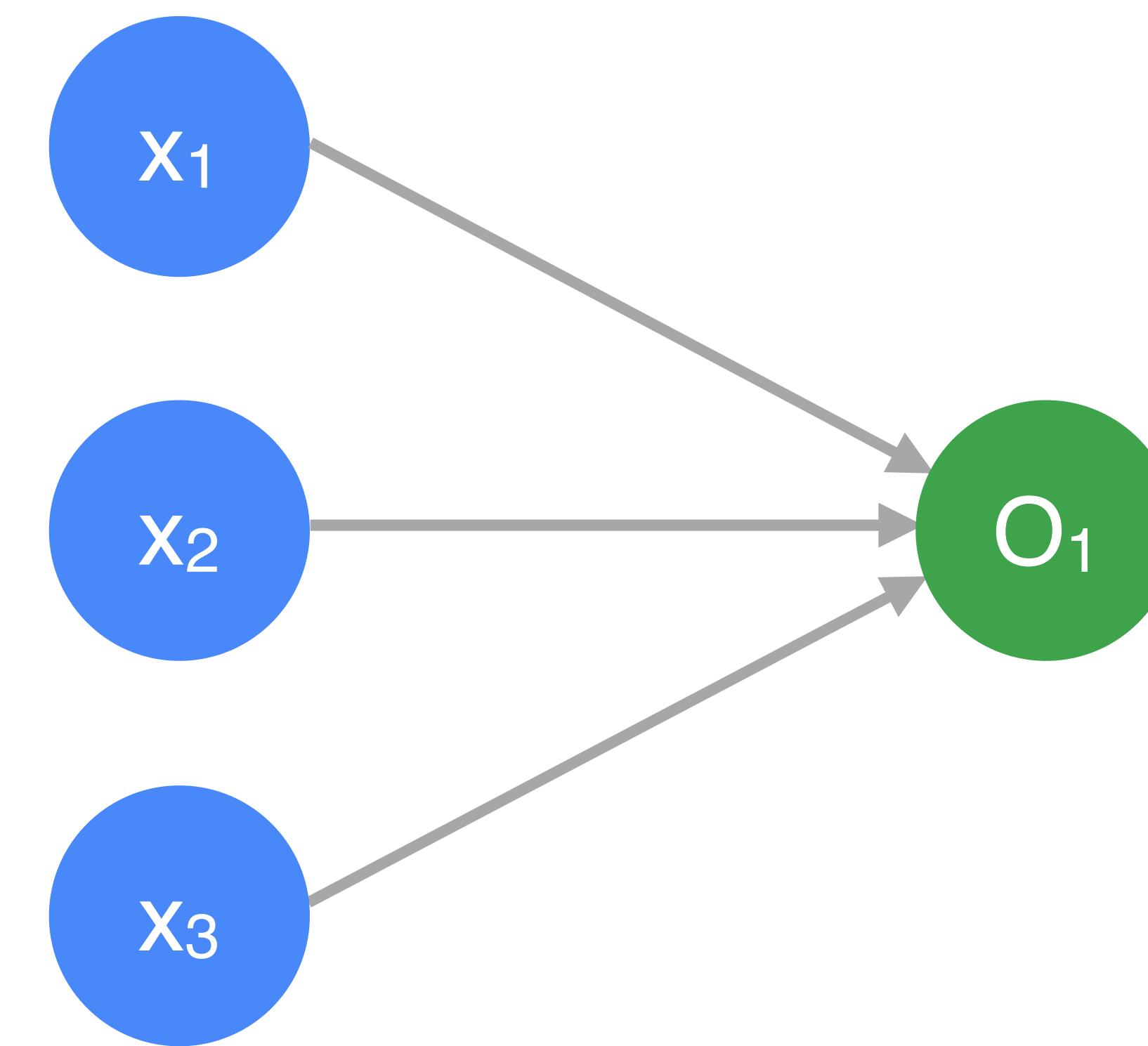
$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

ReLU

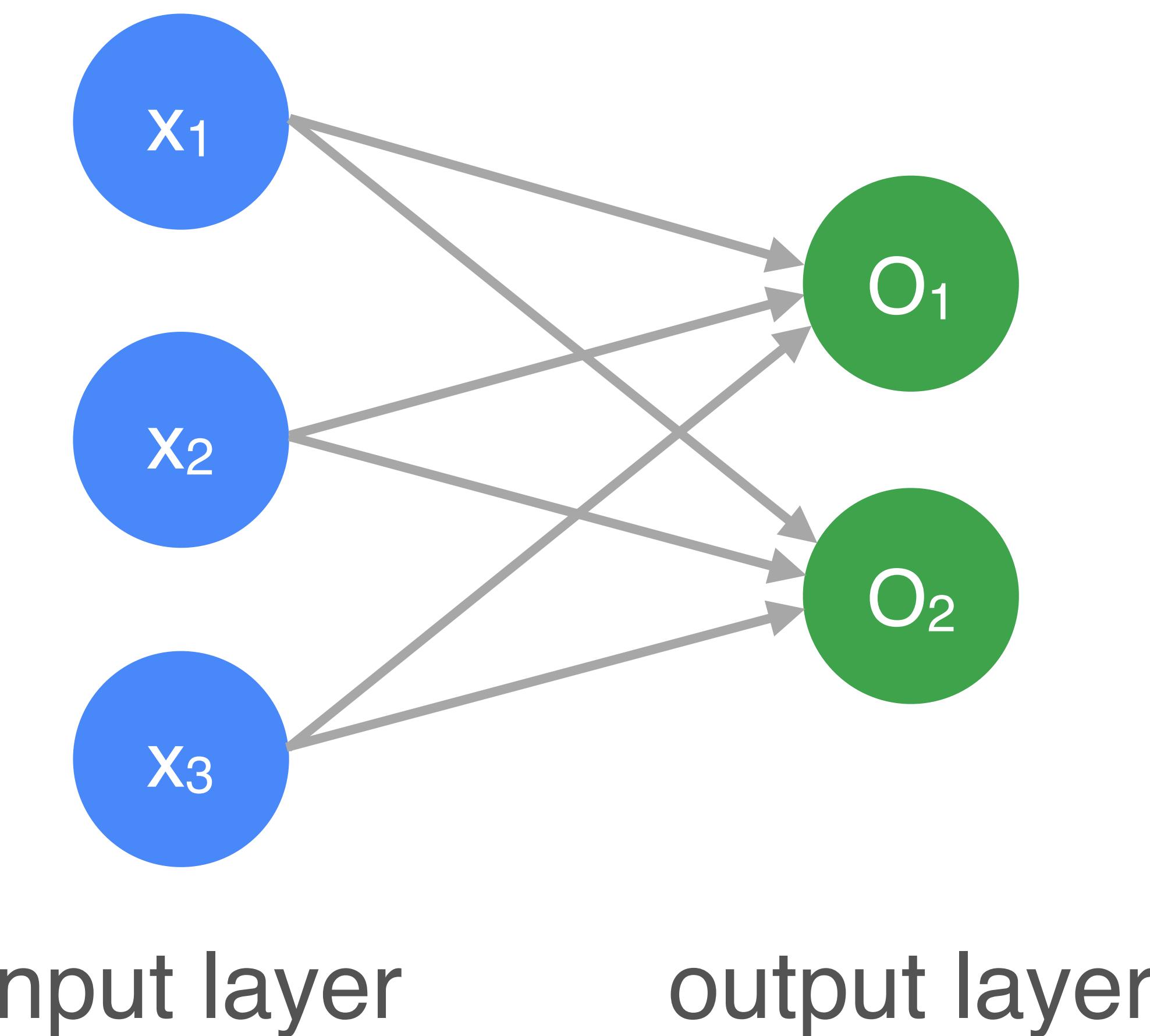


$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

From a Perceptron



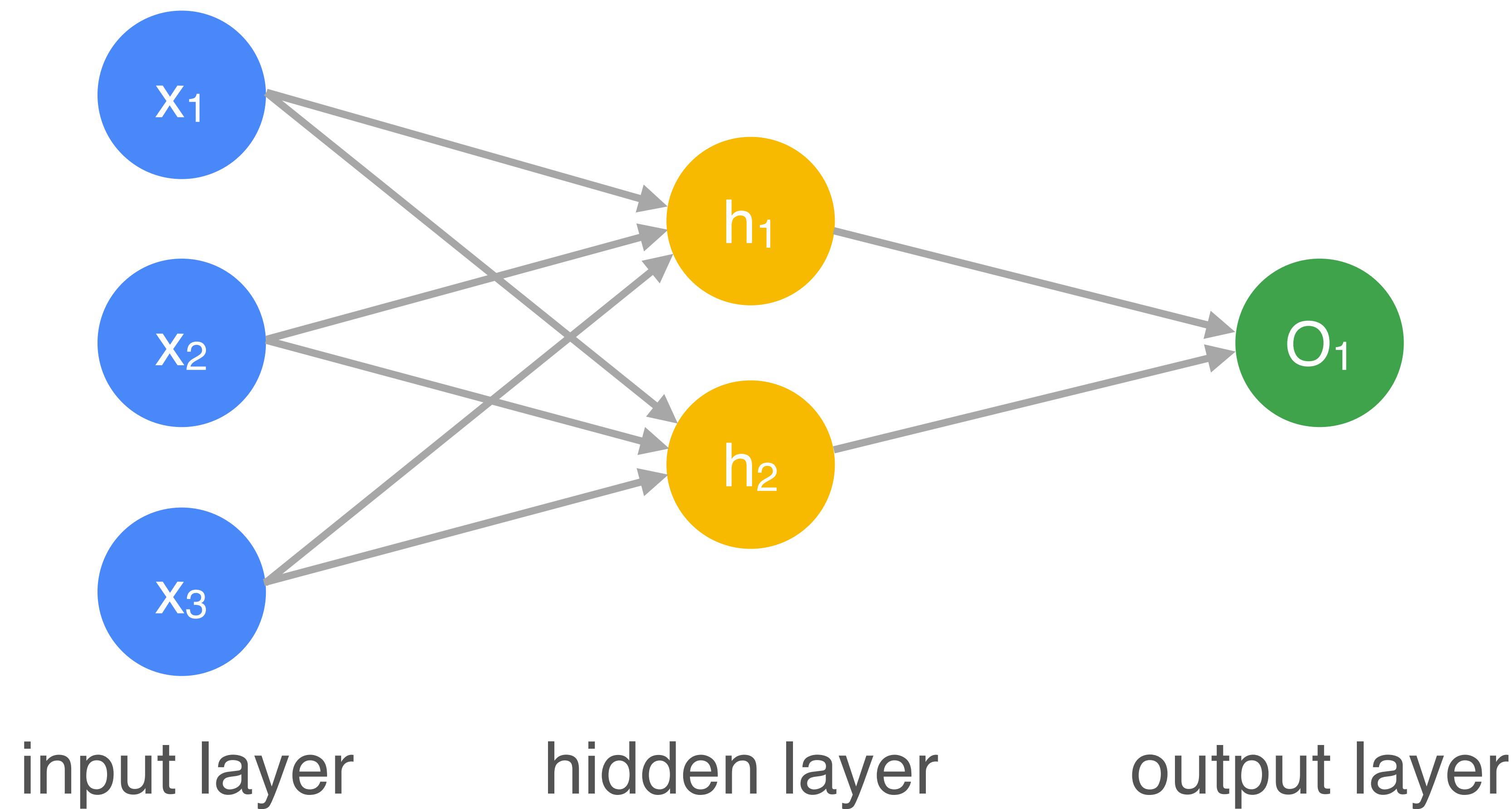
Multi-Output Perceptron



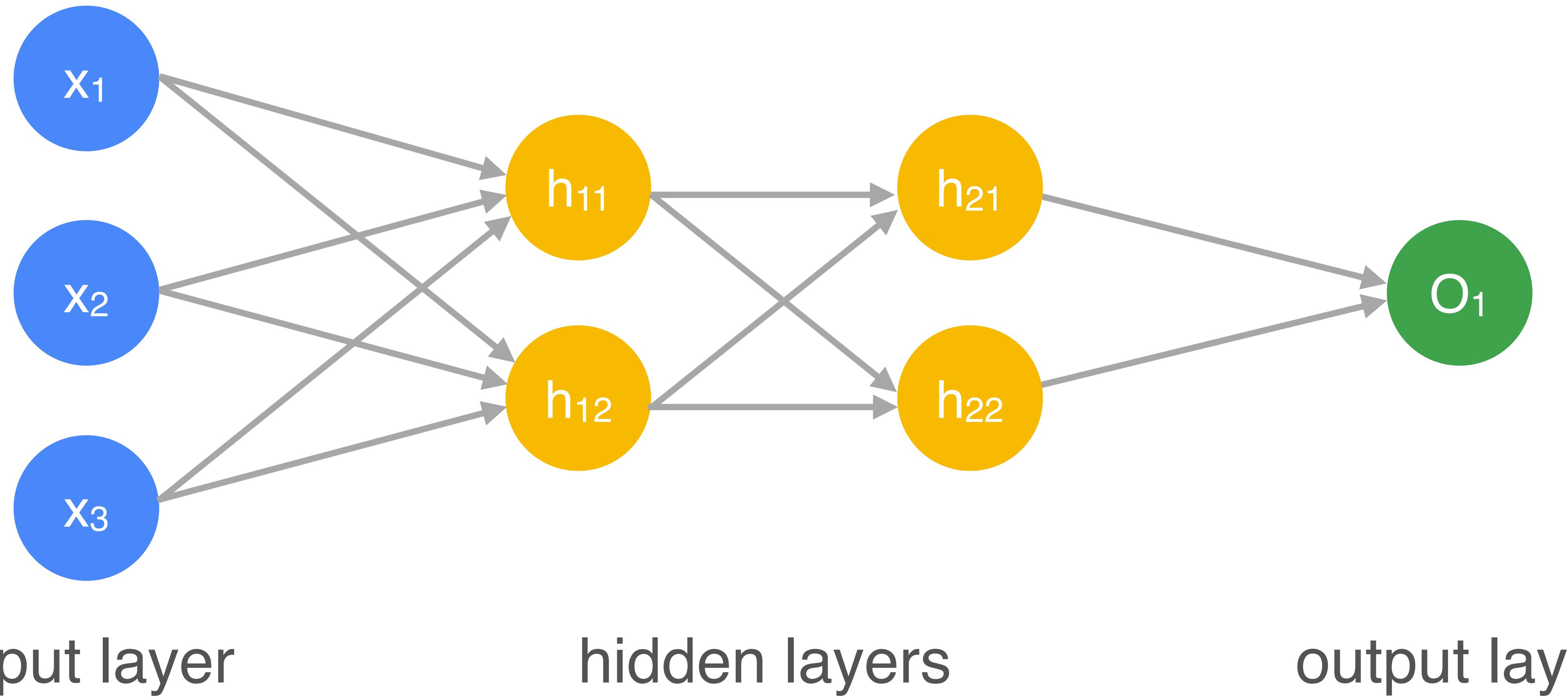
input layer

output layer

Multi-Layer Perceptron



Deep Neural Network



Demo

<http://bit.ly/2scieF6>

Epoch
000,335Learning rate
0.03Activation
SigmoidRegularization
NoneRegularization rate
0Problem 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?

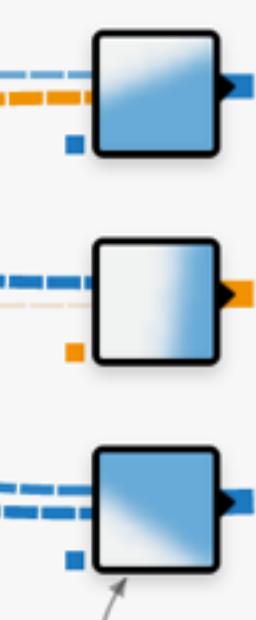
 X_1
 X_2
 X_1^2
 X_2^2
 $X_1 X_2$
 $\sin(X_1)$
 $\sin(X_2)$

1 HIDDEN LAYER

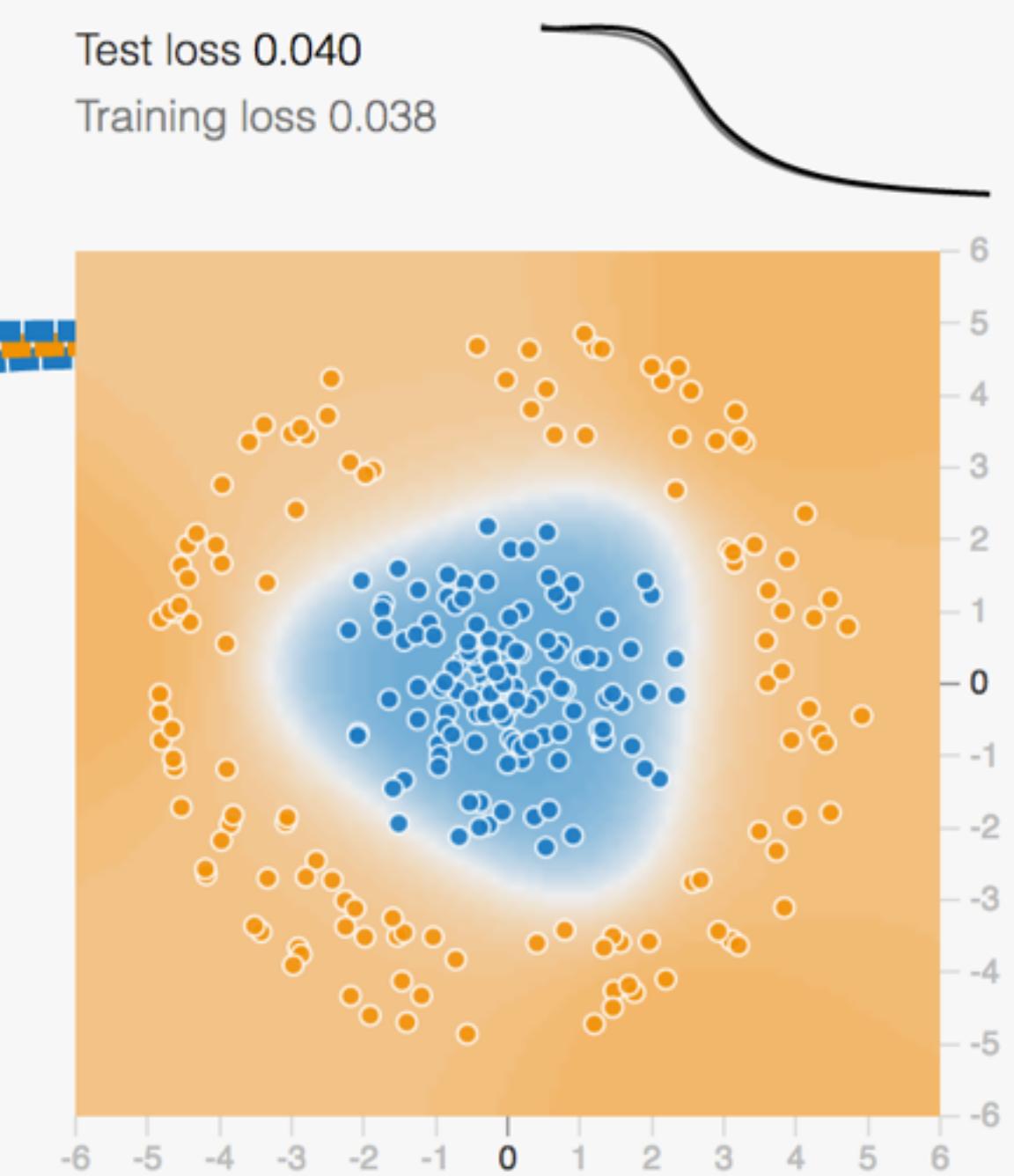
+

-

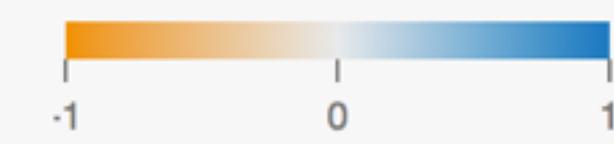
3 neurons



OUTPUT

Test loss 0.040
Training loss 0.038

Colors shows data, neuron and weight values.

 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%

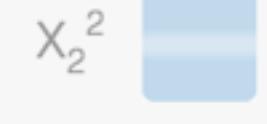
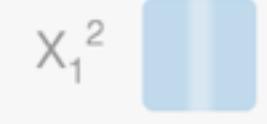
Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?



+ - 1 HIDDEN LAYER

+ -

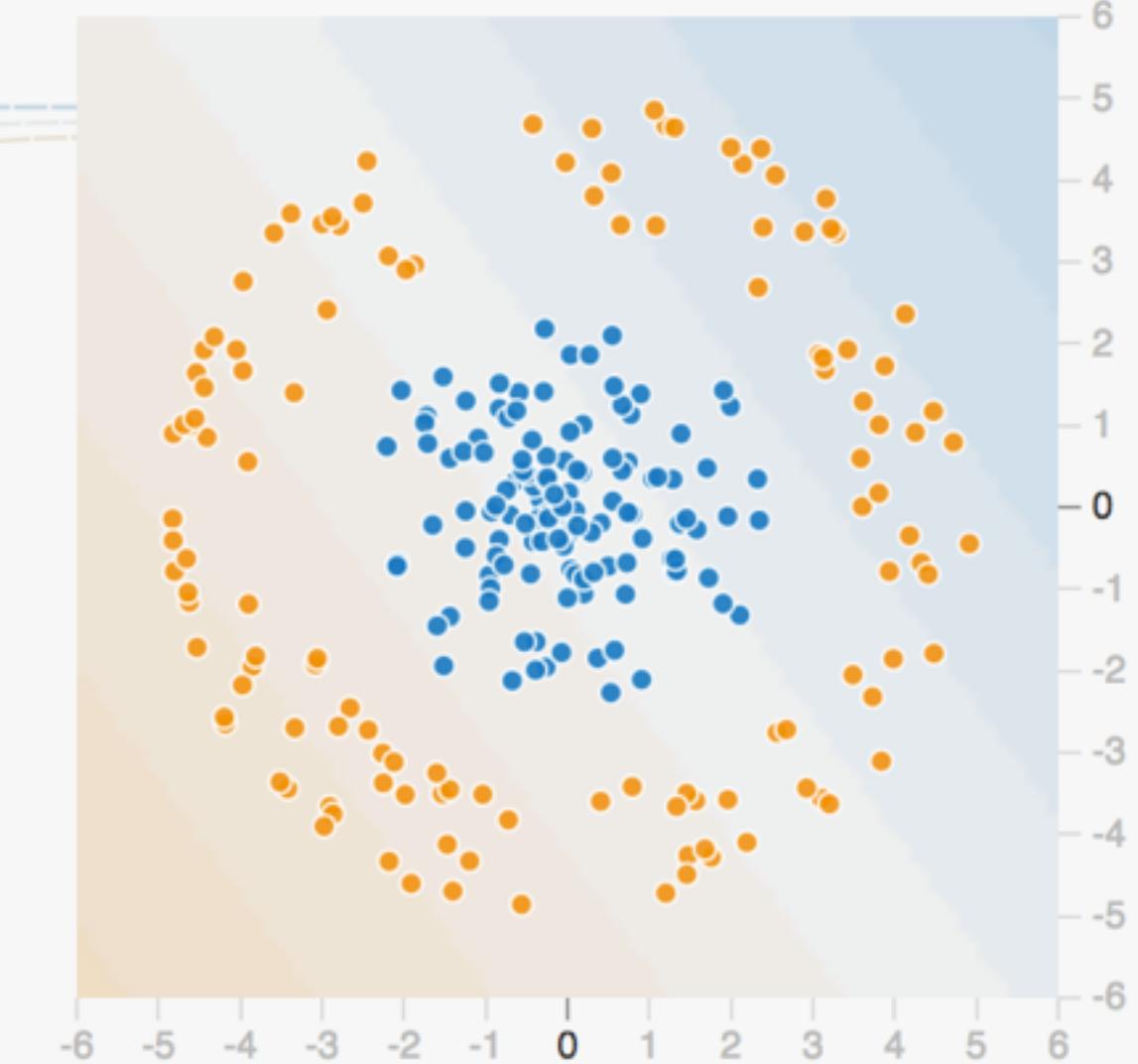
3 neurons



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

OUTPUT

Test loss 0.510
Training loss 0.497



-1 0 1
Show test data Discretize output



Epoch
000,279

Learning rate
0.03

Activation
ReLU

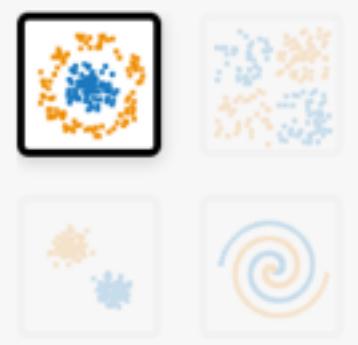
Regularization
None

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
- $X_1 X_2$

- $\sin(X_1)$
- $\sin(X_2)$

1 HIDDEN LAYER

+

-

3 neurons

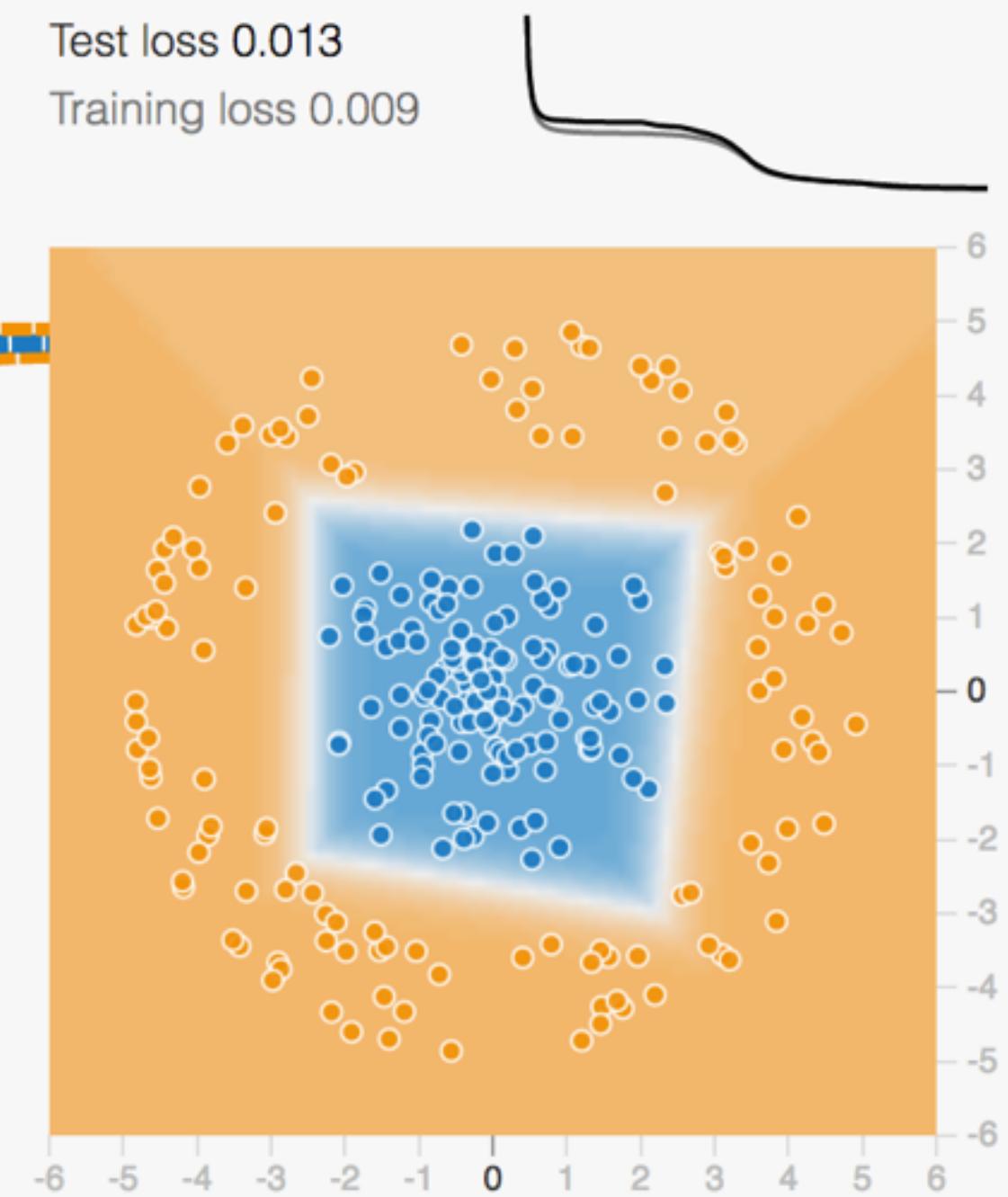
+

-

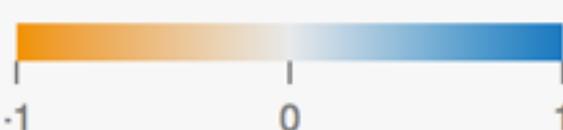
This is the output from one neuron.
Hover to see it larger.

OUTPUT

Test loss 0.013
Training loss 0.009

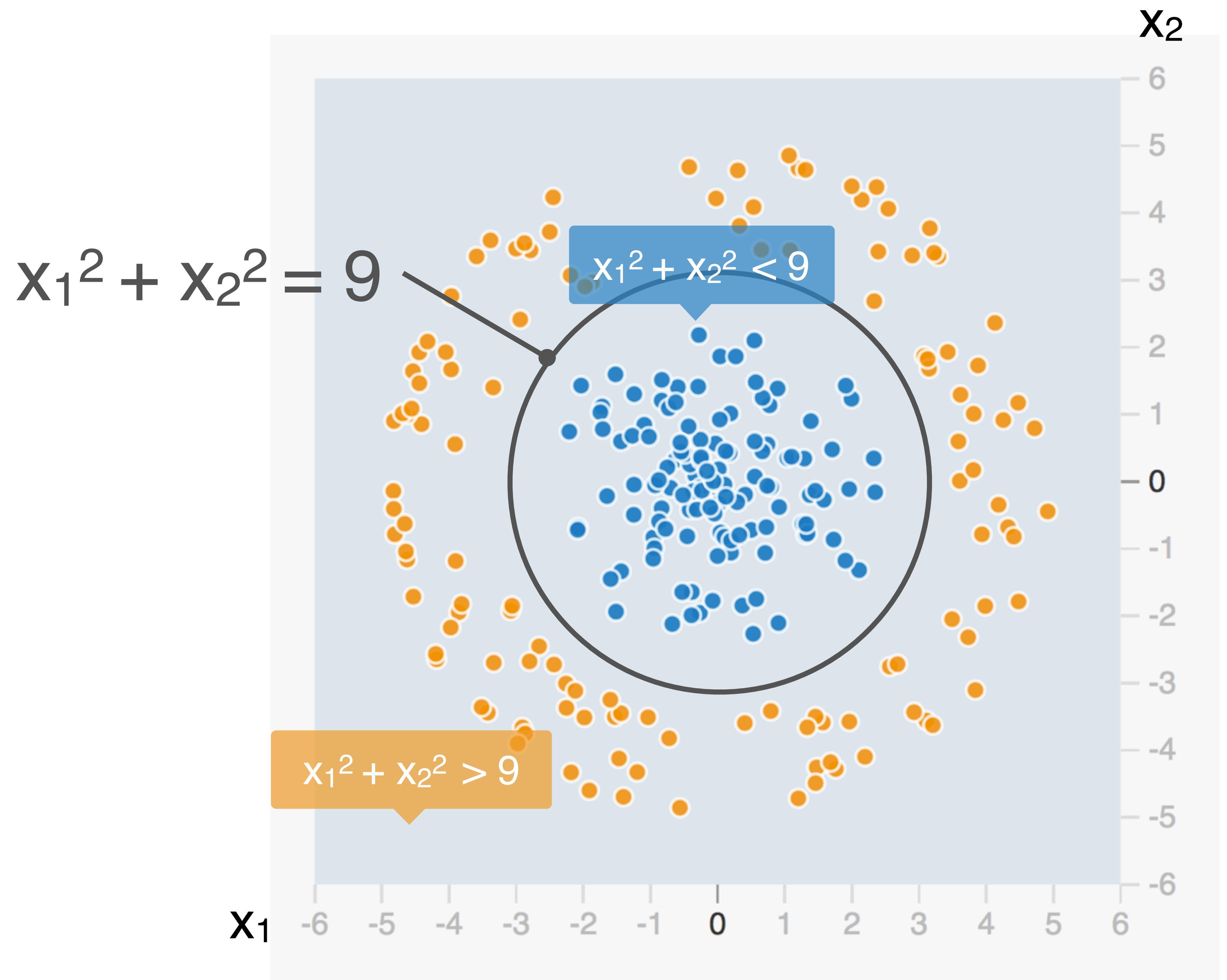


Colors shows data, neuron and weight values.



Show test data

Discretize output



DATA

Which dataset do you want to use?



Ratio of training to test data: 50%

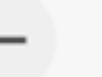
Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

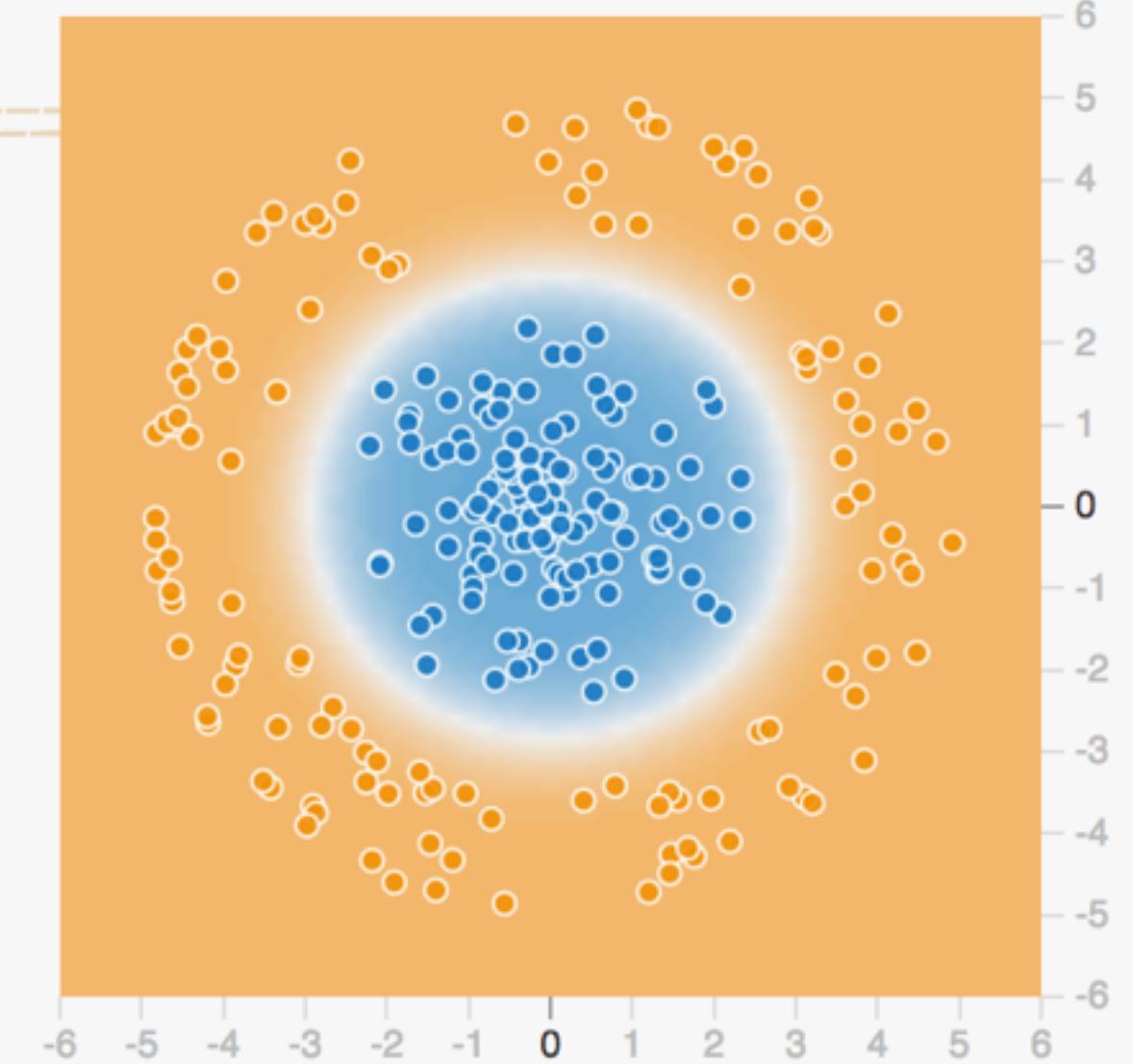


0 HIDDEN LAYERS

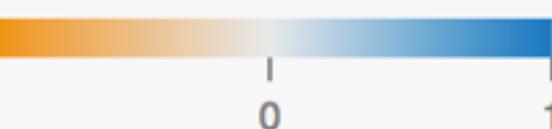
- x_1
- x_2
- x_1^2
- x_2^2
- x_1x_2
- $\sin(x_1)$
- $\sin(x_2)$

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

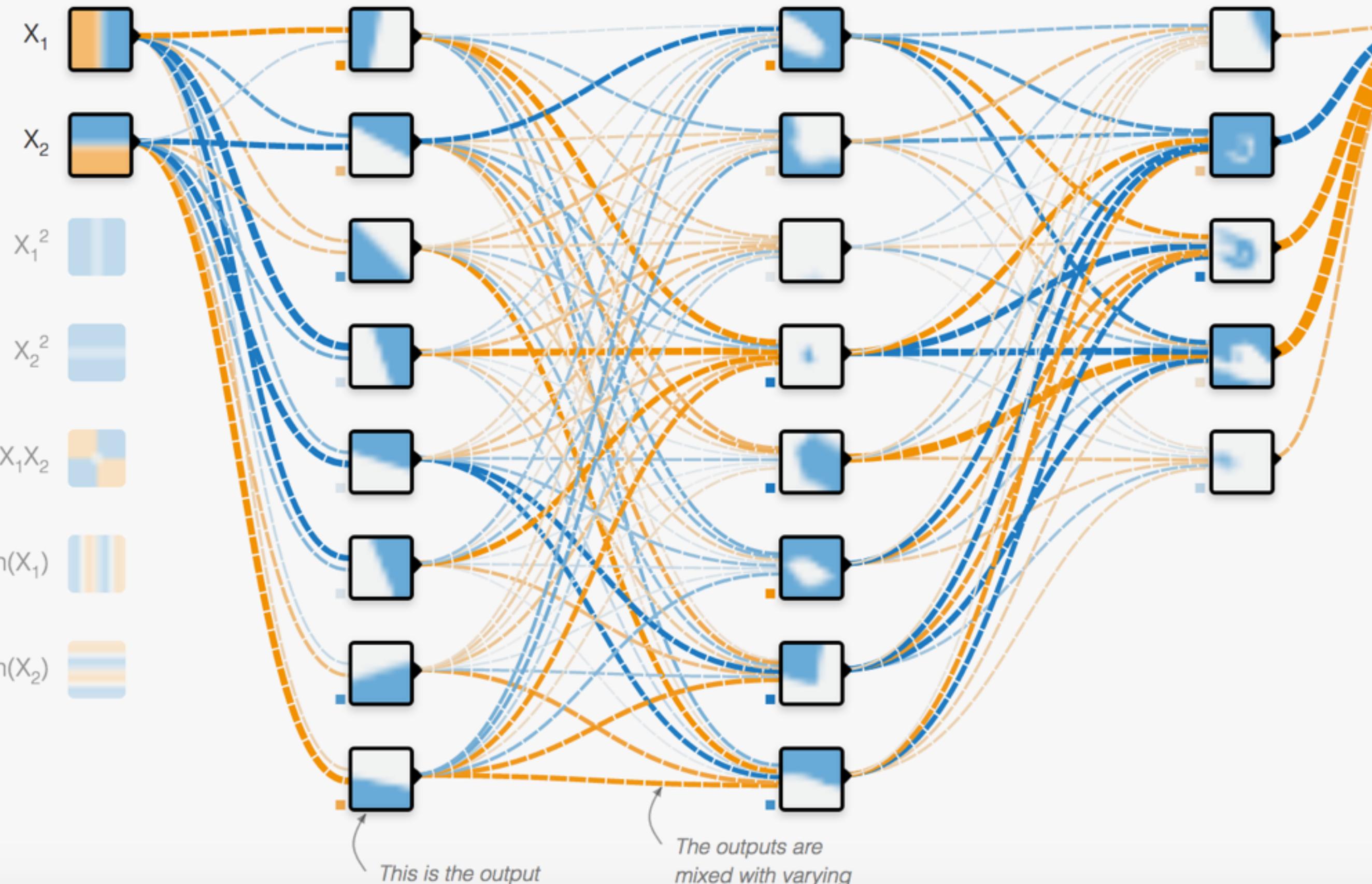
+ -
8 neurons

+ -

8 neurons

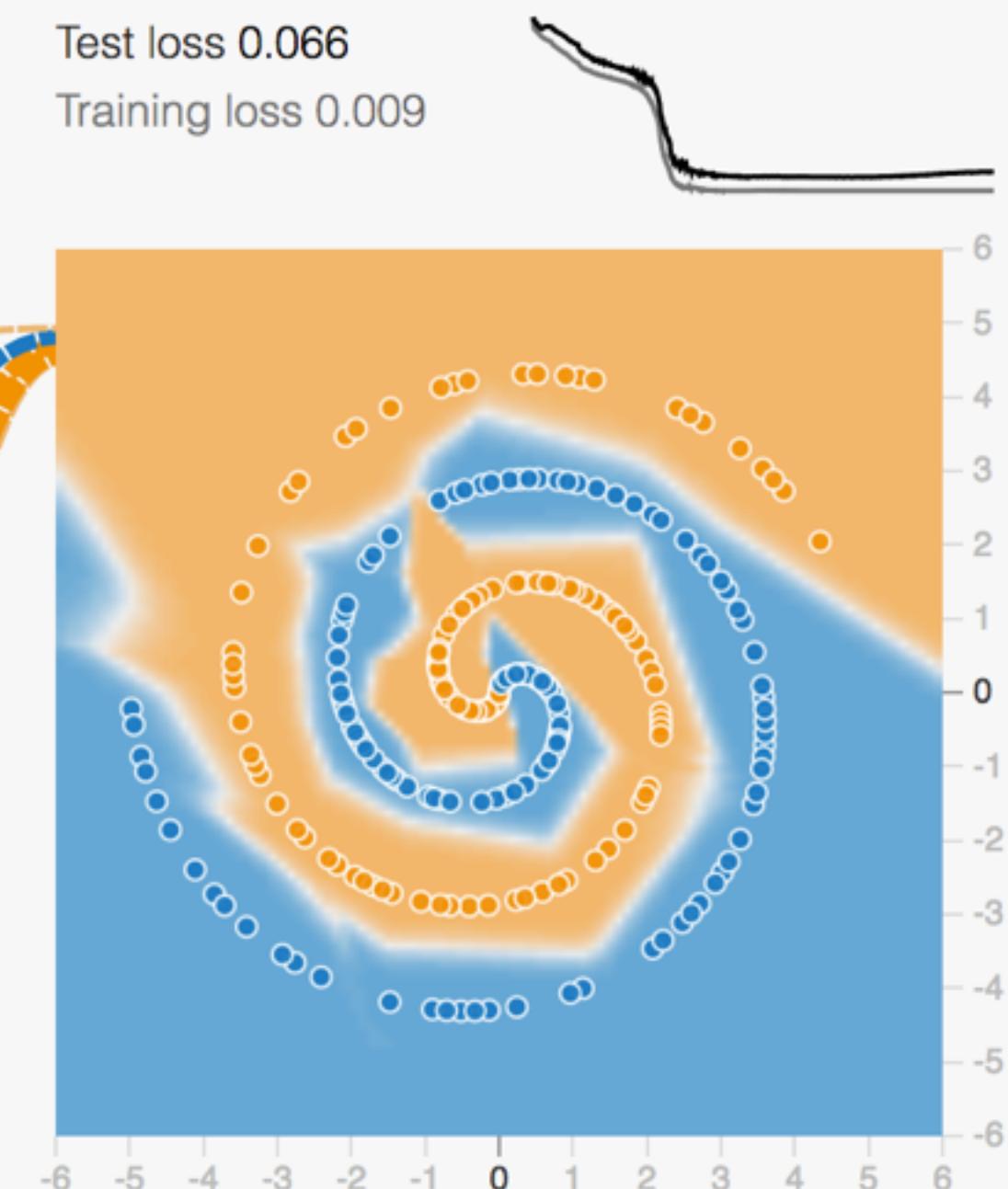
+ -

5 neurons

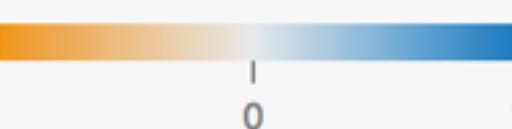


OUTPUT

Test loss 0.066
Training loss 0.009



Colors shows data, neuron and weight values.



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

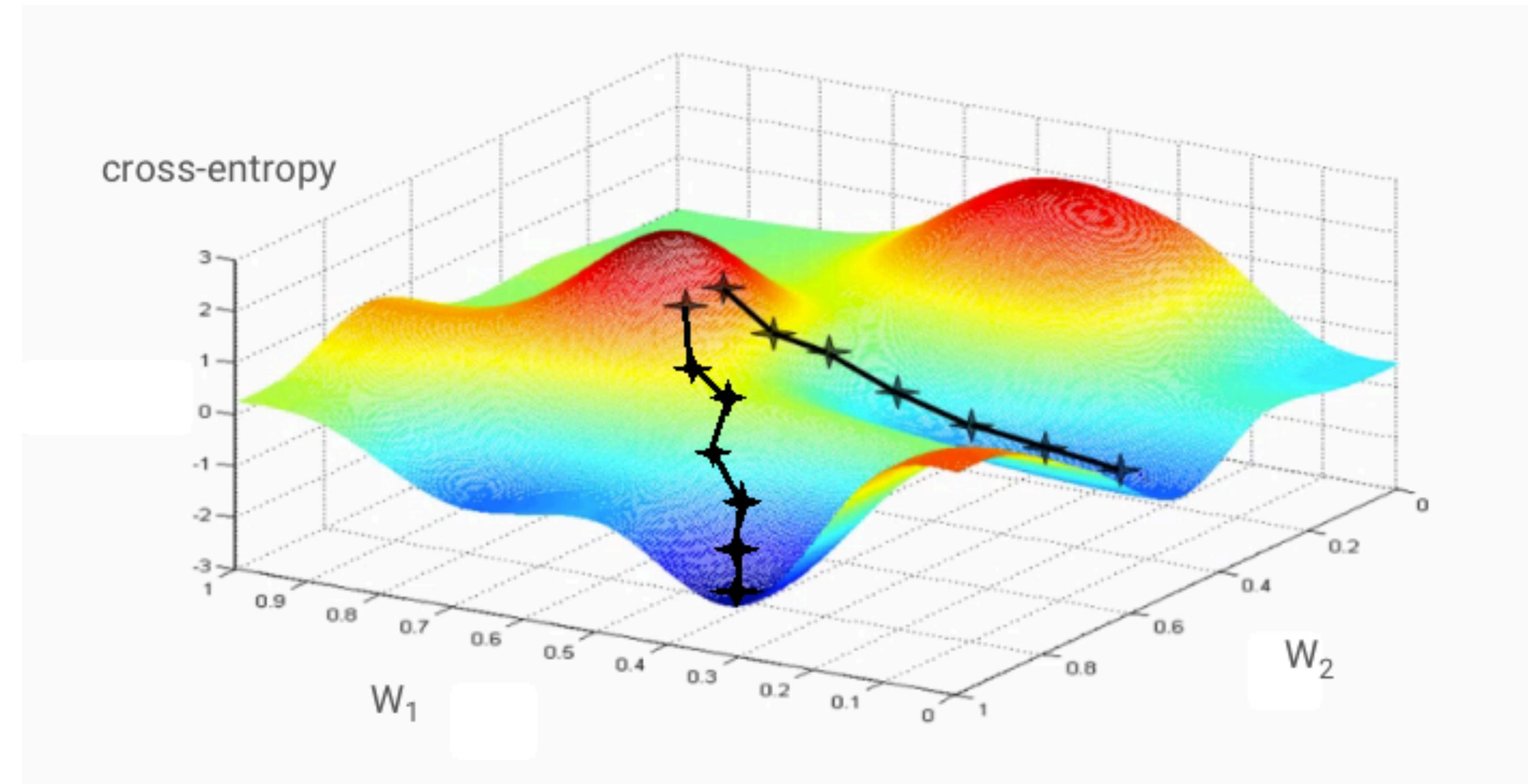
$$-\sum Y'_i \cdot \log(Y_i)$$

Mean Squared Error (MSE)

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

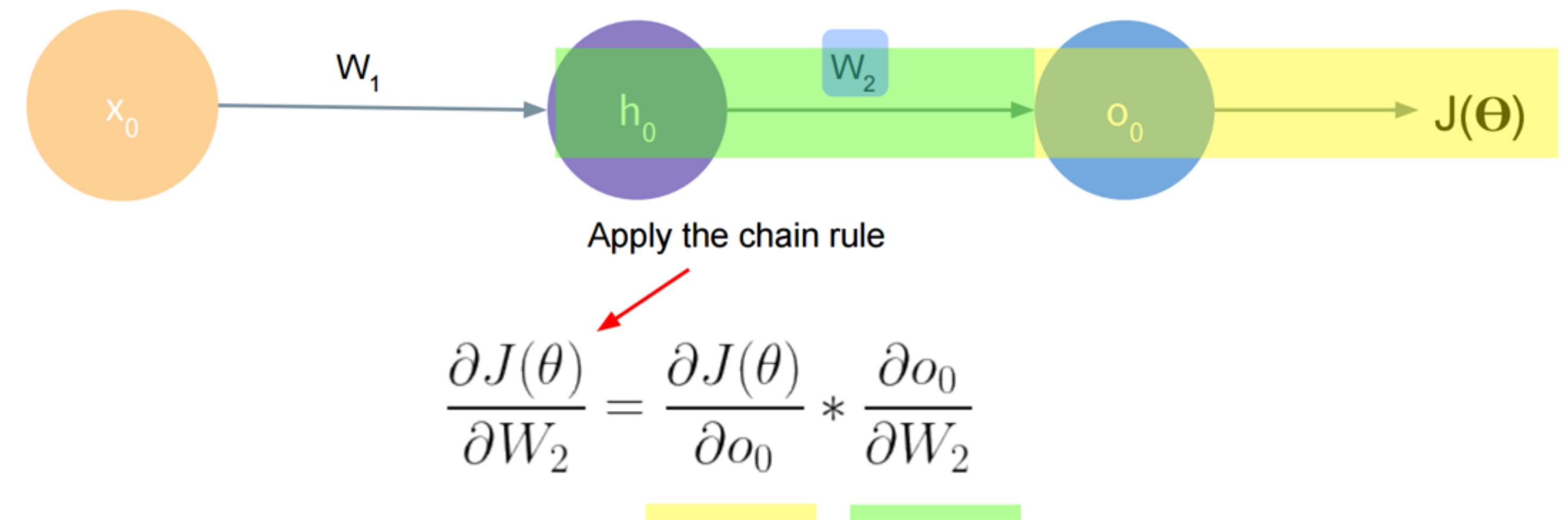
Training Neural Networks

- Minimize the objective function using Gradient Descent

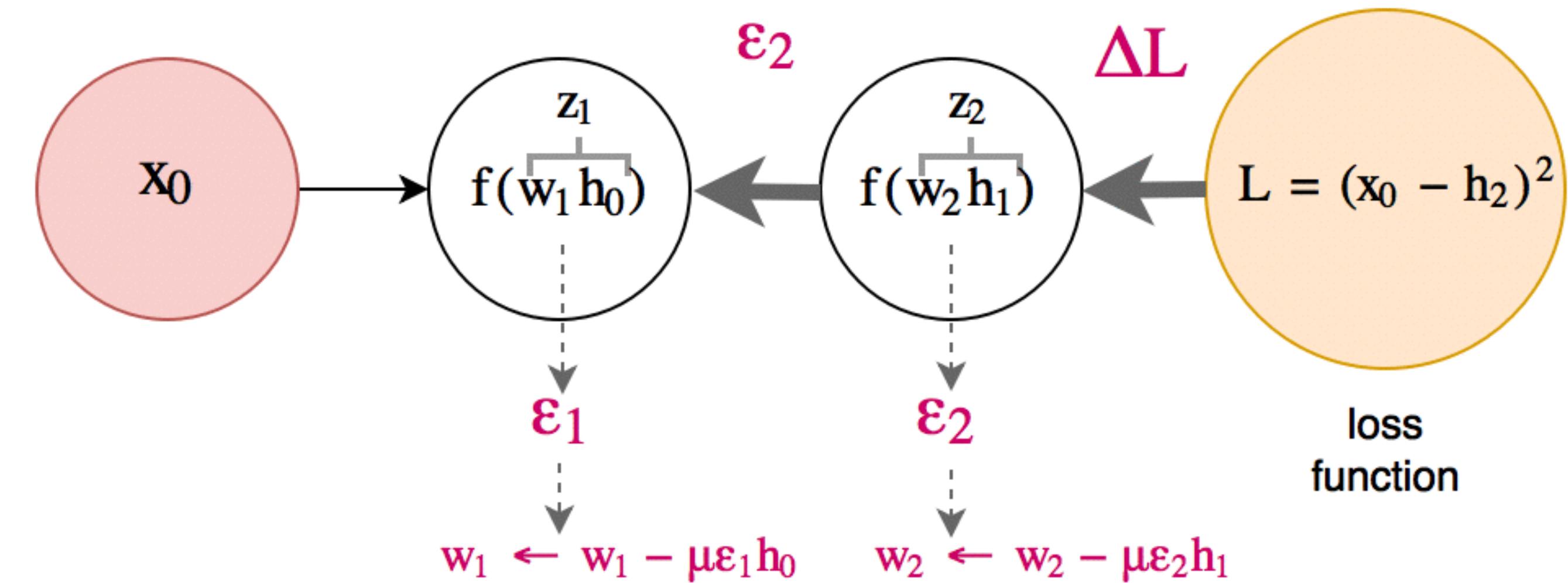


Back-propagation

- Use backpropagation to compute gradient



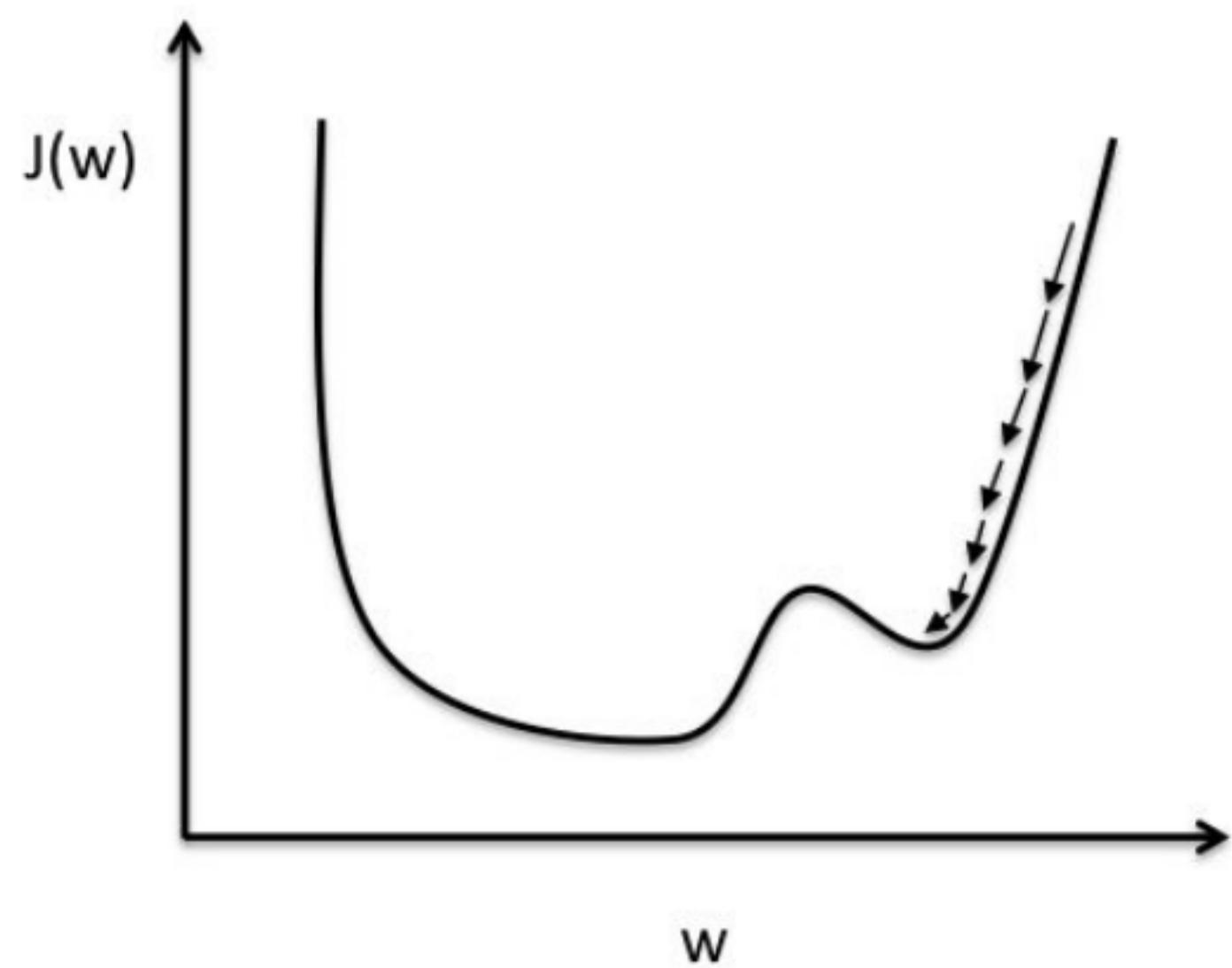
Back-propagation



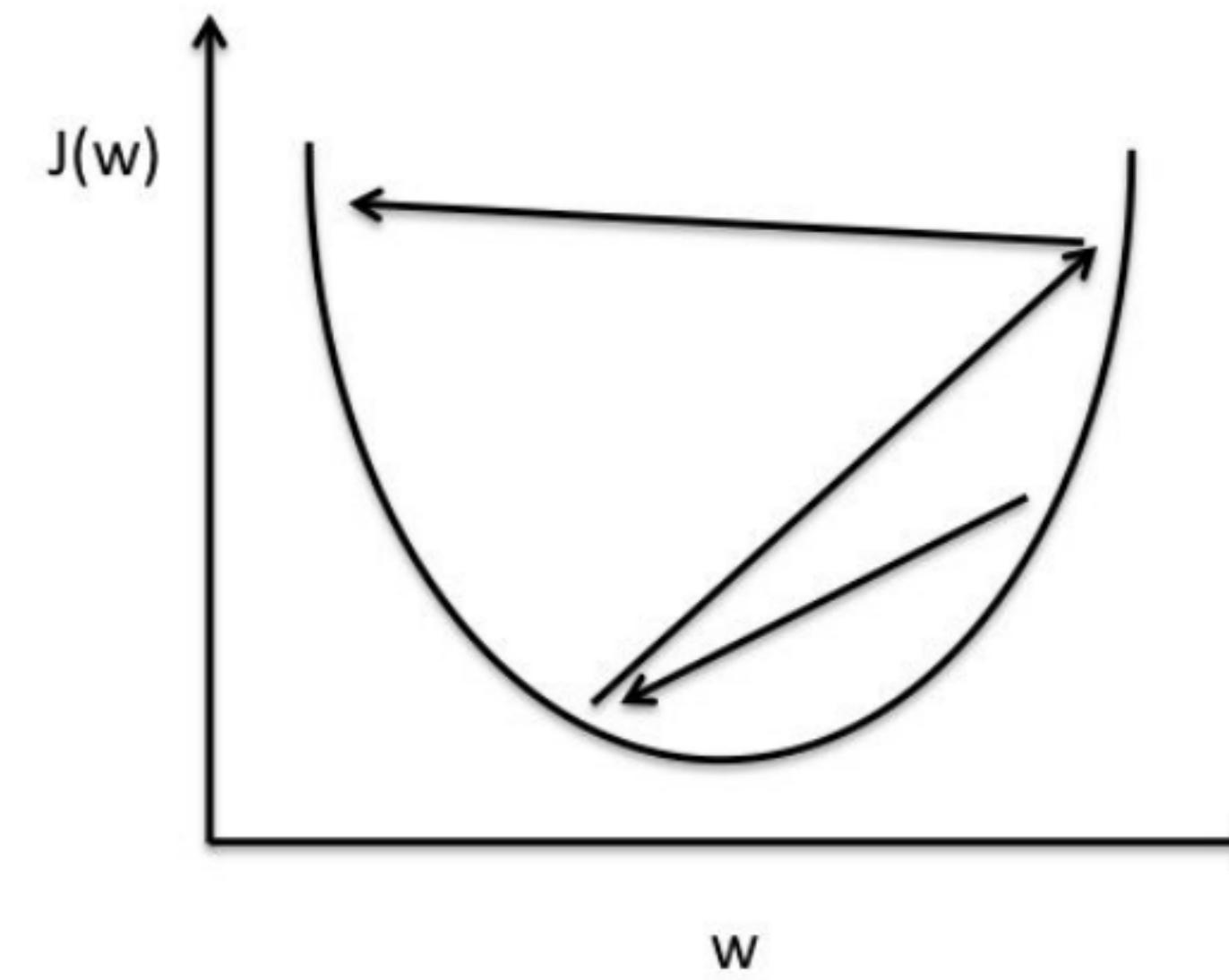
Challenges

- 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.

Regularization

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

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2$$



training loss regularization /
 shrinkage penalty

↑ tuning parameter

Dropout

