

Deep Learning for Design and Verification Engineers

John Aynsley



Deep Learning for Design and Verification Engineers



What is Deep Learning?

Neural Networks

How a Network Learns

Getting Started

Machine Learning – Definition

"Giving computers the ability to learn without being explicitly programmed"

Machine Learning Algorithms

Classification

Regression

Neural networks

Support vector machines

Dimensionality reduction

Bayesian statistics

Hebbian learning

Markov models

Decision tree learning

Random forests

Reinforcement learning

Evolutionary algorithms

Used in statistics and data science

Deep Learning

Deep learning is very simple algorithms
applied very intensively to massive datasets
and often applied end-to-end

This is not a definition, just an intuitive description!

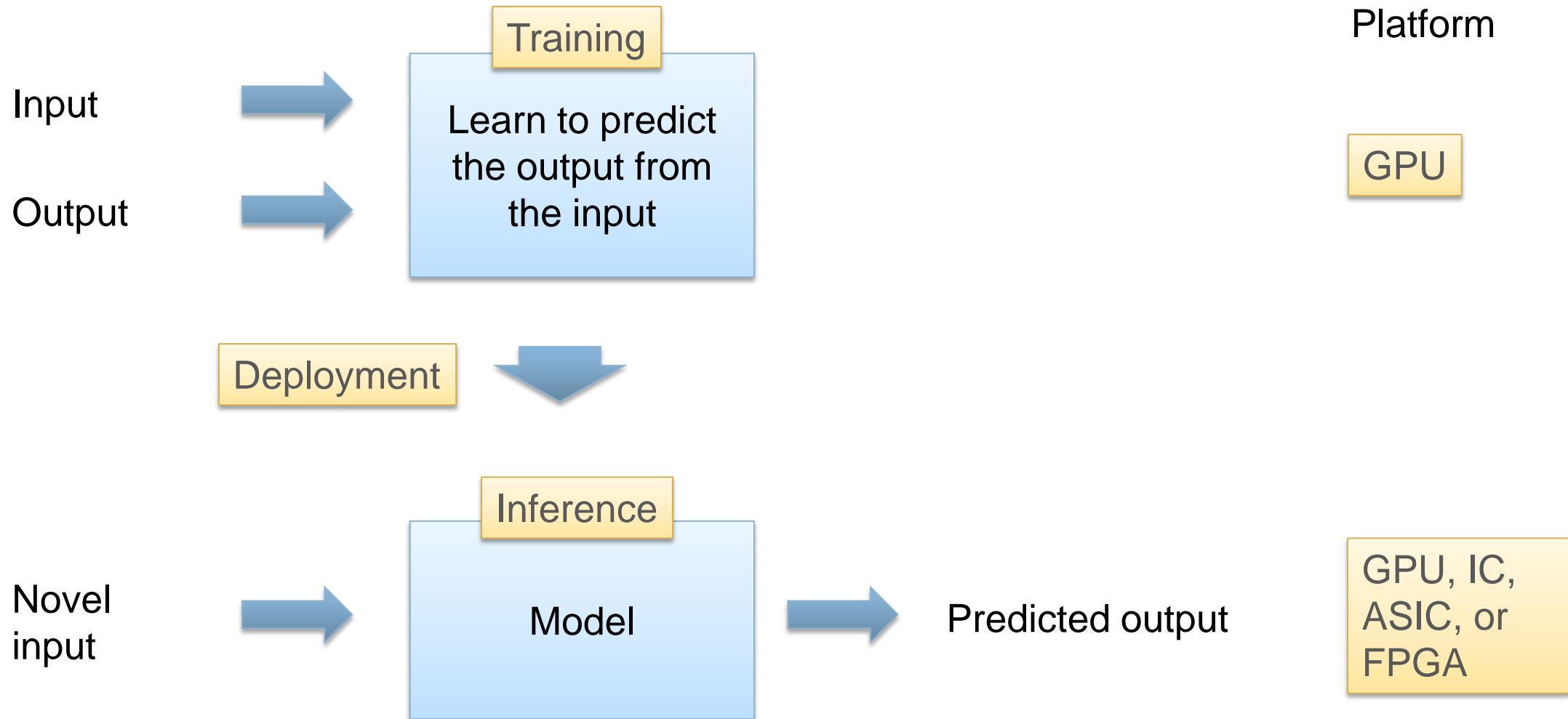
Machine Learning and Deep Learning

Machine learning is not new

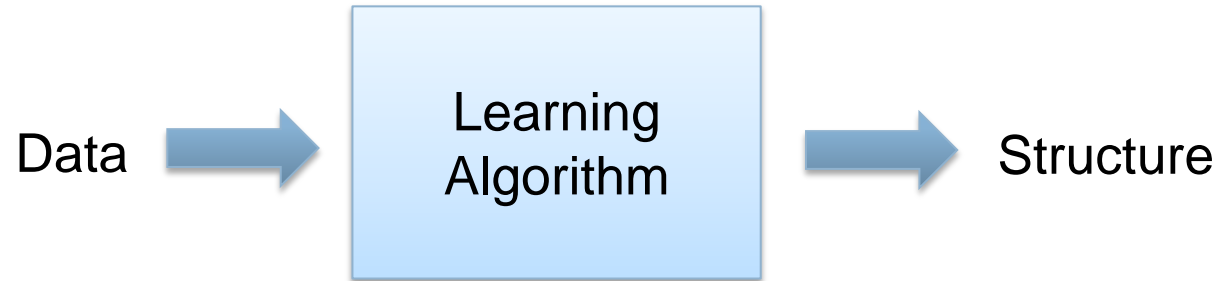
Deep learning is new

Deep learning is growing fast because it really works!

Supervised Learning



Unsupervised Learning



Examples

Partition the data into clusters based on similarity

Find hidden patterns in the data

Pick out anomalous data samples

Supervised Learning with a Neural Network

Labels

Persian
Abyssian
Siamese
British Shorthair
Scottish Fold
Burmese

Training data



Ground Truth

Compare

Cost
Function

Neural Network

Prediction

Deep Learning: >1 hidden layer

Why Neural Networks Now?

2012

Bigger datasets

Faster computers

Since 2012

Improved neural network architectures

Neural networks often outperforming previous state-of-the-art

Image Classification


← → ↻ Secure | https://cloud.google.com/vision/ ☆

Google Cloud Platform

Search CONSOLE SIGN IN

Why Google Products Solutions Launcher Pricing Customers Documentation > TRY IT FREE CONTACT SALES

Labels Web Text Document Properties Safe Search JSON



IMG_0352.JPG

Telephony	84%
Corded Phone	79%
Product	77%
Technology	74%
Electronics	70%
Electronic Instrument	68%
Product Design	63%
Electronic Device	62%

The ImageNet Challenge (ILSVRC)

ImageNet Large Scale Visual Recognition Challenge: 1.2M images in 1000 categories

*Depends what
you measure!*

Year	Network	#Layers	Top-5 Error Rate
2012 winner	AlexNet (CNN)	8	15.3%
2013 winner	ZFNet (CNN)	8	14.8%
2014	VGGNet (CNN)	19	7.3%
2014 winner	GoogLeNet (Inception)	22	6.7%
2015 winner	ResNet (residual)	152	3.6%
2016 winner	CUIImage (ensemble)	-	3.0%

← *Dramatic improvement*

← *Human error rate ~ 5%*

← *3% bad labels*

Training typically takes a few weeks on a few GPUs

Natural Language Translation

The screenshot shows the Google Cloud Platform Translate API interface. At the top, the browser address bar displays the URL <https://cloud.google.com/translate/>. The Google Cloud Platform header includes the logo, a search bar, and links for [CONSOLE](#) and [SIGN IN](#). A navigation menu below the header contains links for [Why Google](#), [Products](#) (which is underlined), [Solutions](#), [Launcher](#), [Pricing](#), [Customers](#), and [Documentation](#), followed by a right arrow and two buttons: [TRY IT FREE](#) and [CONTACT SALES](#).

The main content area features a blue banner that says "TRY THE API". Below this banner is a translation interface with two columns. The left column is labeled "Source Language" and has a dropdown menu set to "English (en)". The right column is labeled "Target Language" and has a dropdown menu set to "French (fr)". A double-headed arrow icon is positioned between the two language labels. The source text in the left column is "It was raining cats and dogs and I had forgotten my umbrella." The target text in the right column is "Il y avait des chiens et des chiens et j'avais l'idée d'être un monstre."

Other Exciting Applications

Image segmentation

Adding captions to images

Adding color to black-and-white images

Generating images that mimic other artists or styles

Spotting significant events in videos

Human pose estimation to analyze behavior

Handwriting recognition

Voice recognition and voice generation

Predicting patterns in natural phenomena

Playing games

Cool Stuff

<http://www.yaronhadad.com/deep-learning-most-amazing-applications/>

<https://deepdreamgenerator.com>

Deep Learning for Design and Verification Engineers

What is Deep Learning?



Neural Networks

How a Network Learns

Getting Started

artificial intelligence

human brain

machine learning

data mining

image processing

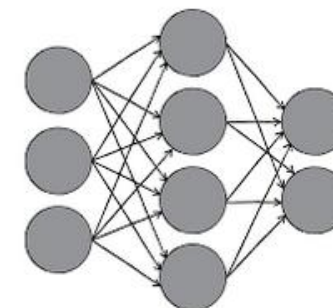
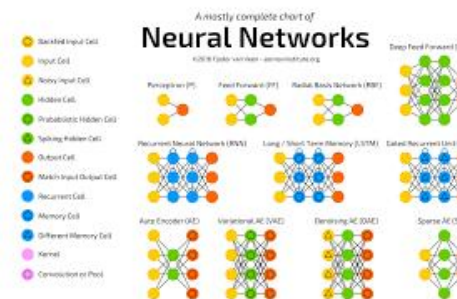
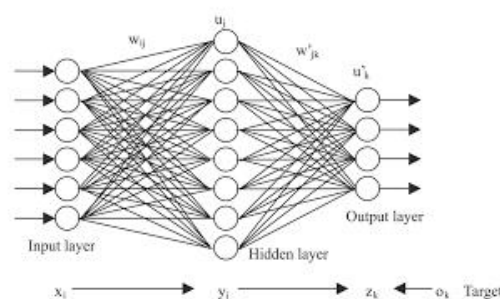
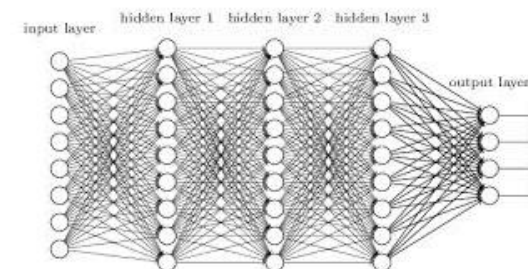
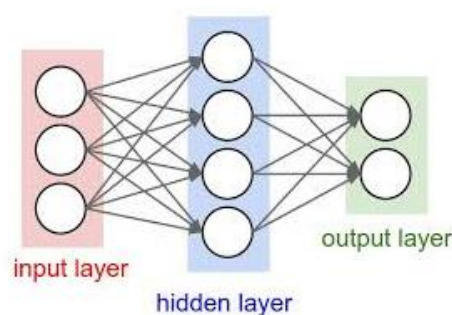
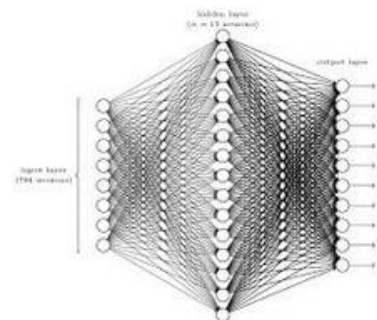
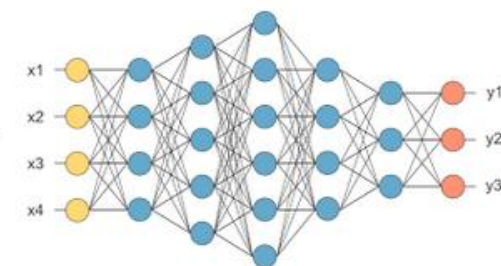
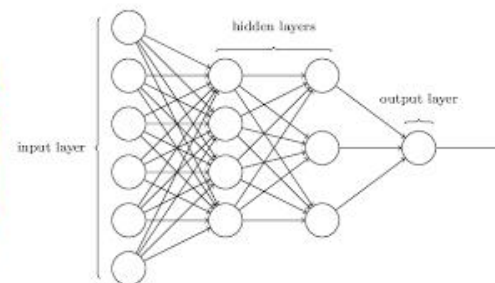
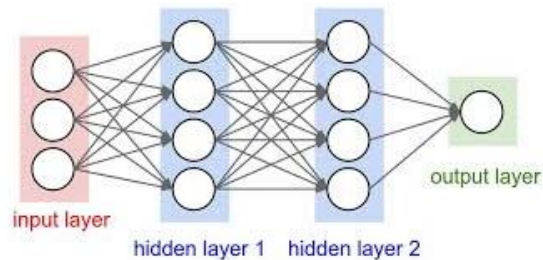
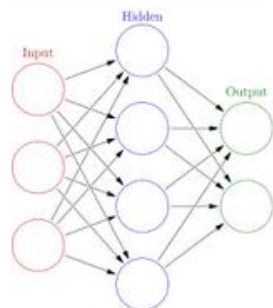
computer vision

distributed

pattern recognition

xor

deep

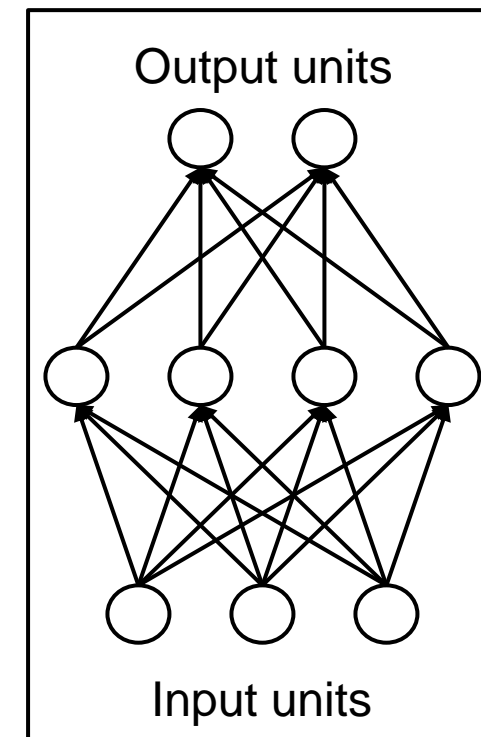
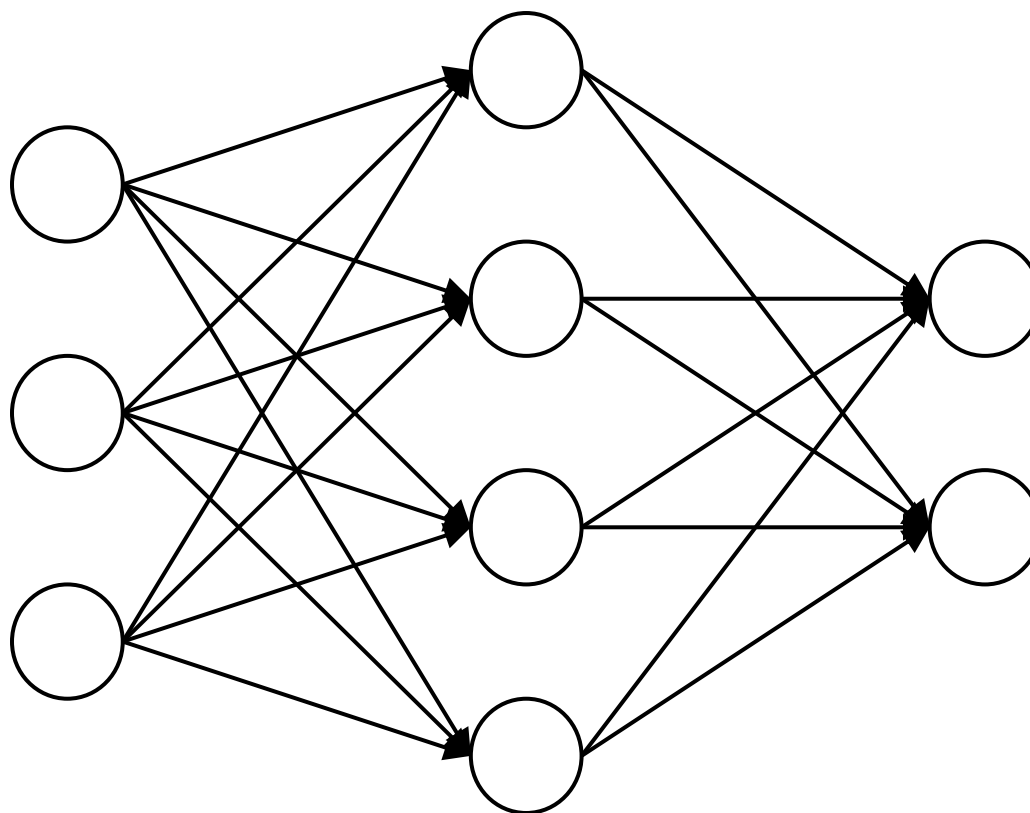


A Neural Network?

Input units

Hidden units

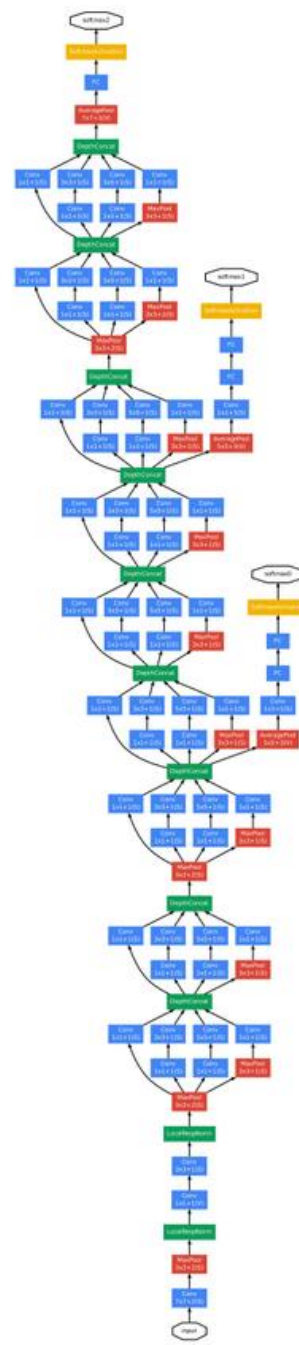
Output units



In the academic literature

E.g. GoogLeNet

Inception network

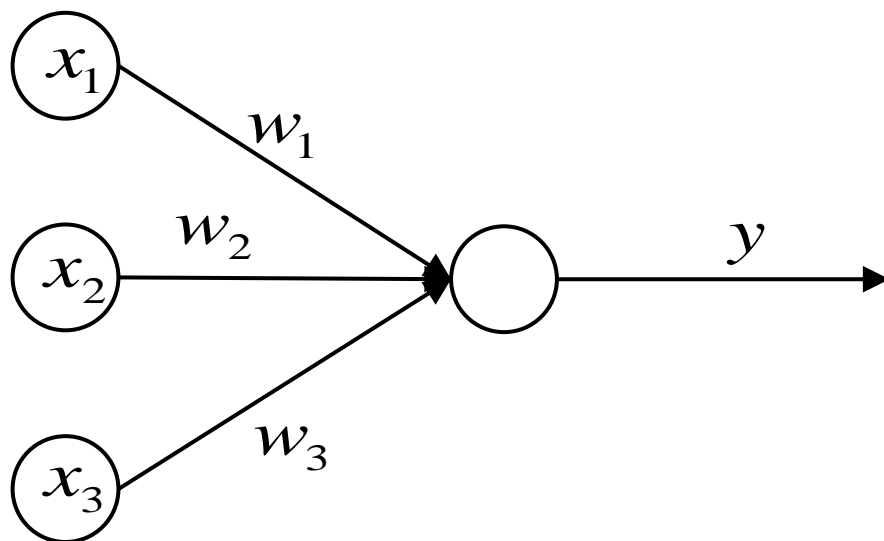


Auxiliary outputs

An Artificial Neuron

Input units

Hidden unit



$$u = \sum_{i=1}^3 w_i x_i + b$$

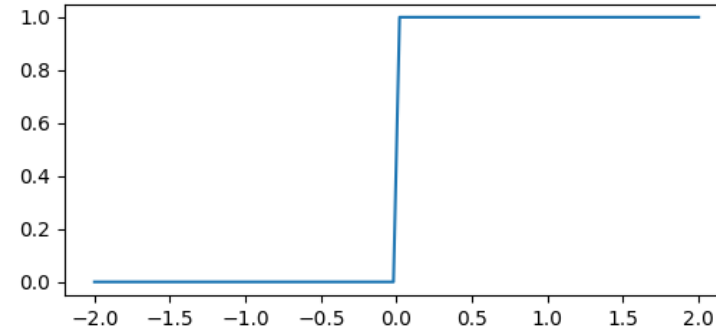
Linear function

$$y = \text{activation}(u)$$

Non-linear function

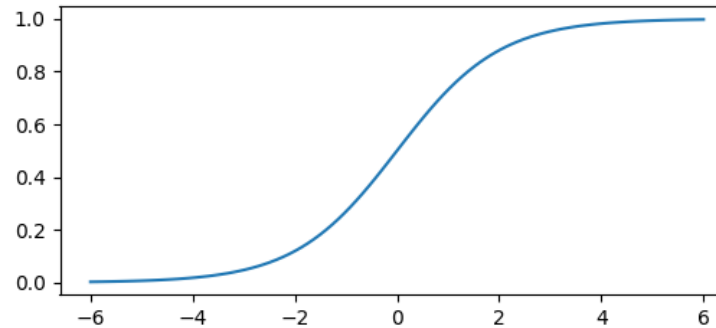
Common Activation Functions

Step



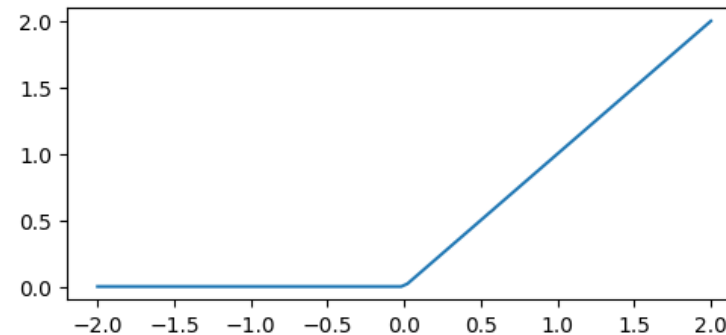
The Perceptron

Sigmoid
aka
logistic



Early neural networks
and still used today

ReLU



Popular today

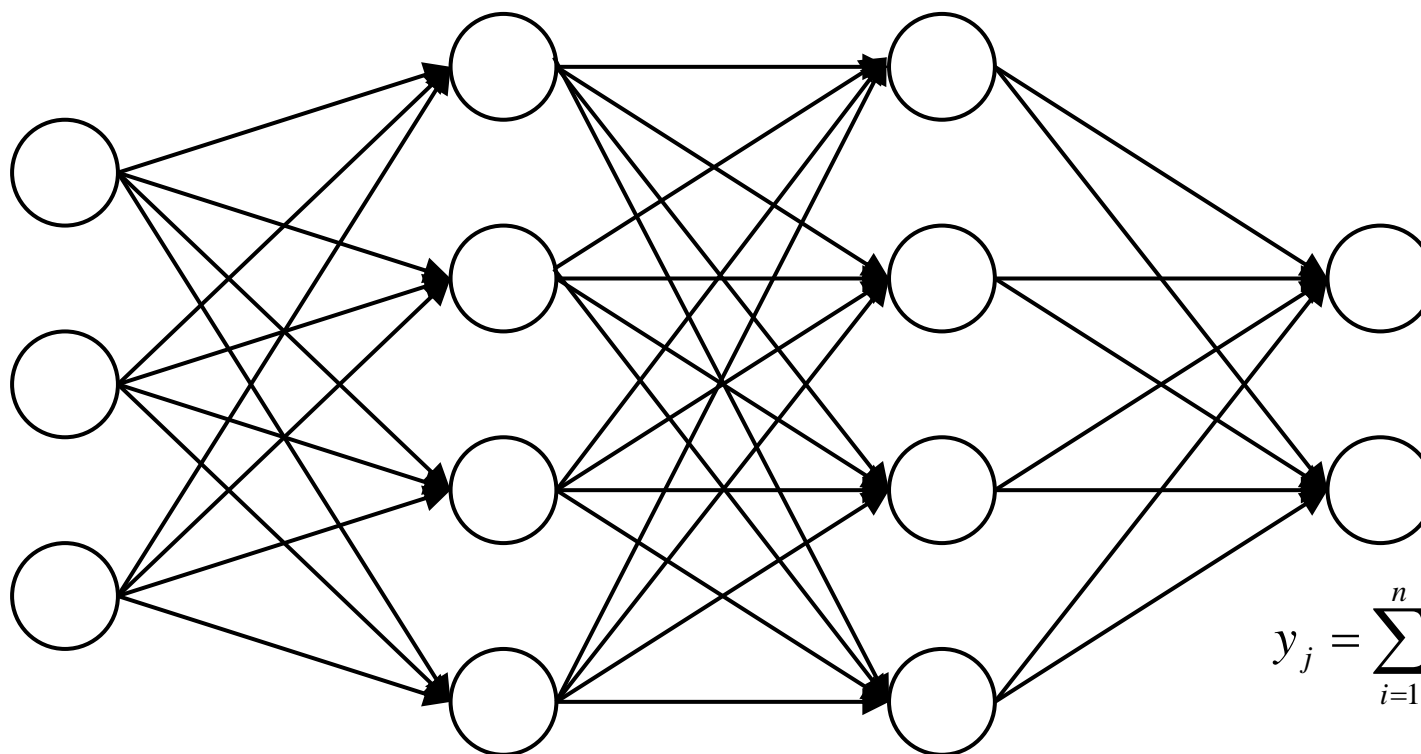
A Deep Neural Network

Input units

Hidden units

Hidden units

Output units



$$y_j = \sum_{i=1}^n w_{ji} x_i + b_j$$

$$y_j = \text{RELU} \left(\sum_{i=1}^3 w_{ji} x_i + b_j \right)$$

$$y_j = \text{RELU} \left(\sum_{i=1}^n w_{ji} x_i + b_j \right)$$

Kinds of Neural Network

ANN – Artificial Neural Network

CNN – Convolutional Neural Network (e.g. image processing)

R-CNN – Regional CNN (image segmentation)

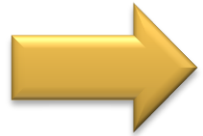
RNN – Recursive Neural Network (e.g. natural language processing)

GAN – Generative Adversarial Network

Deep Learning for Design and Verification Engineers

What is Deep Learning?

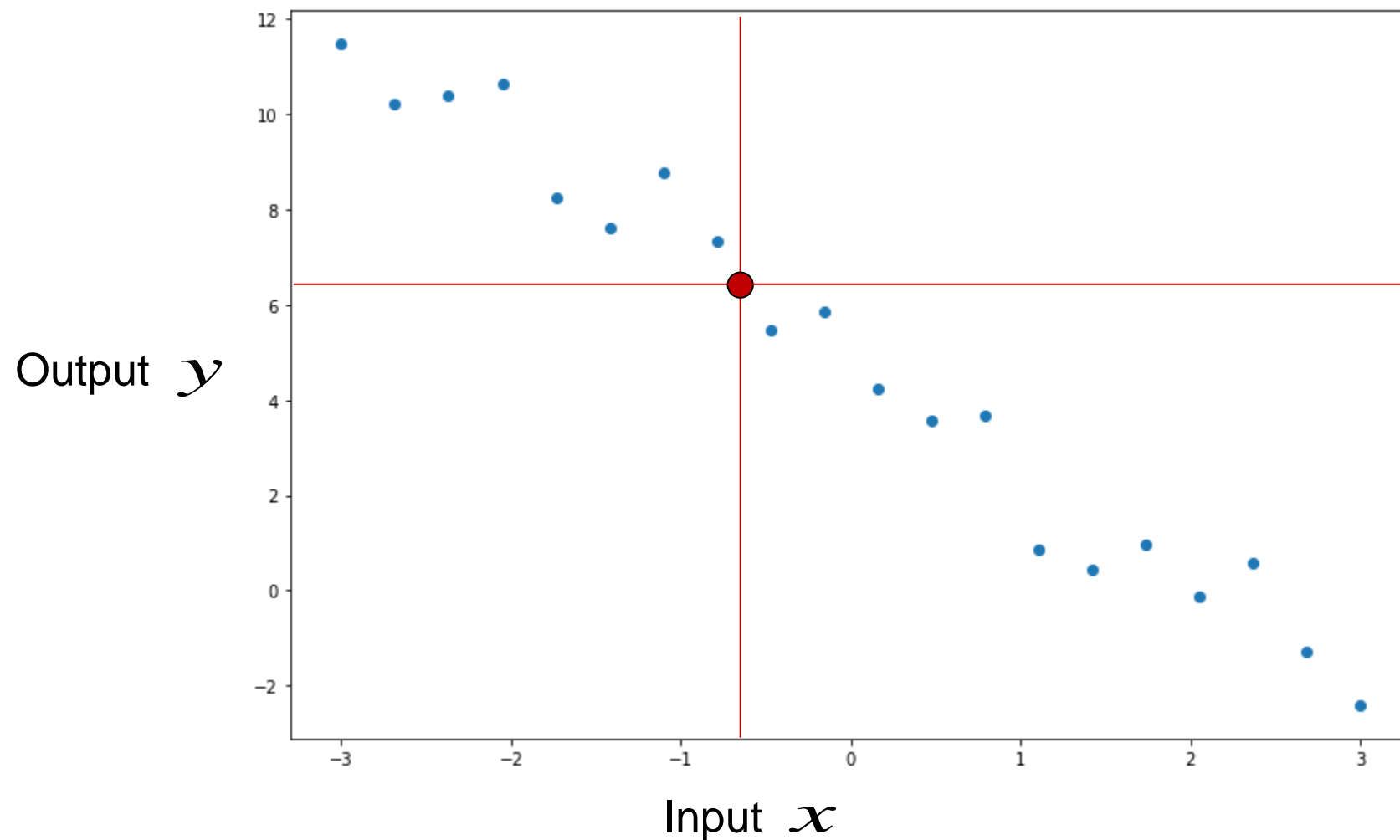
Neural Networks



How a Network Learns

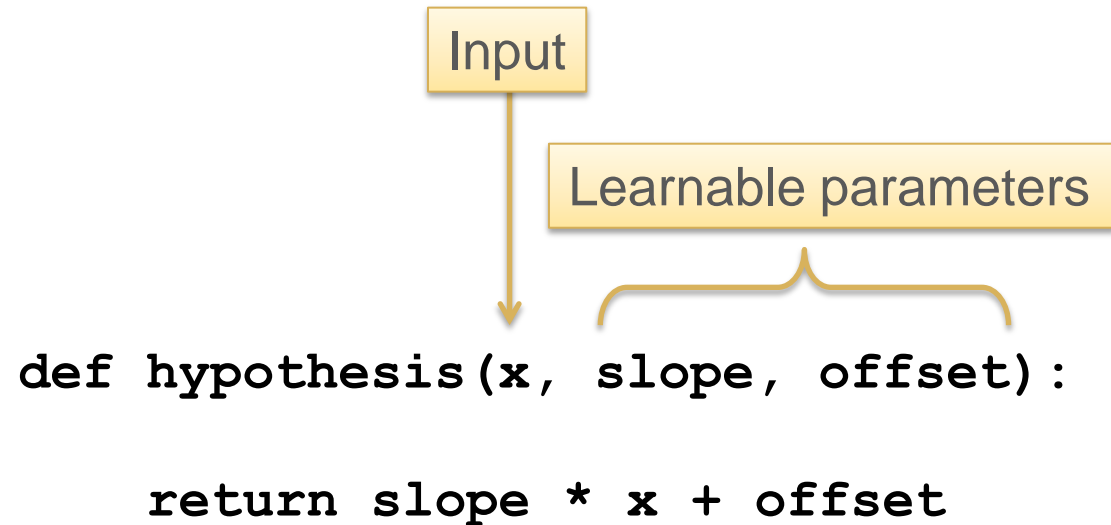
Getting Started

Regression Task

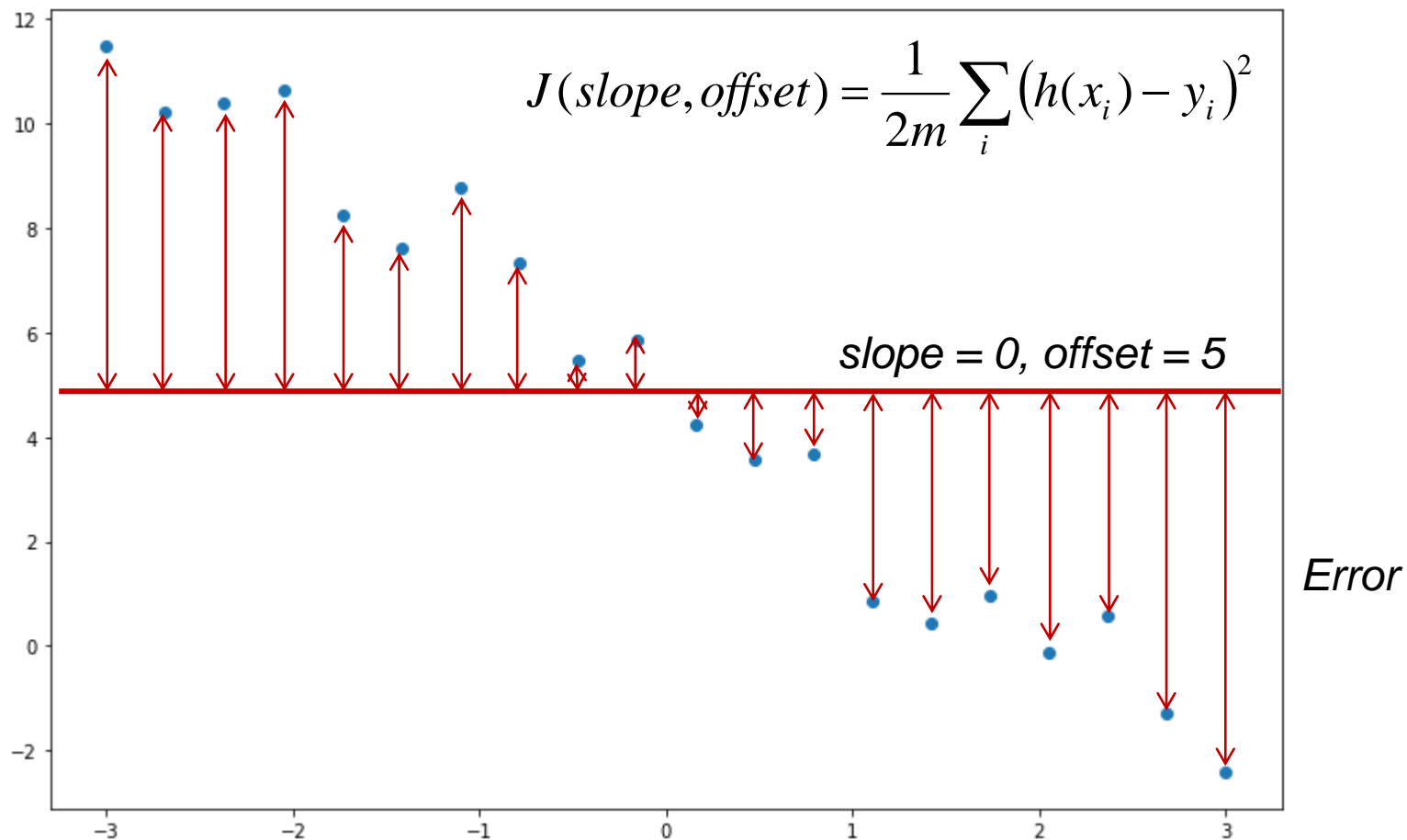


Define a Hypothesis / Model / Network

$$h_{slope,offset}(x) = slope \cdot x + offset$$

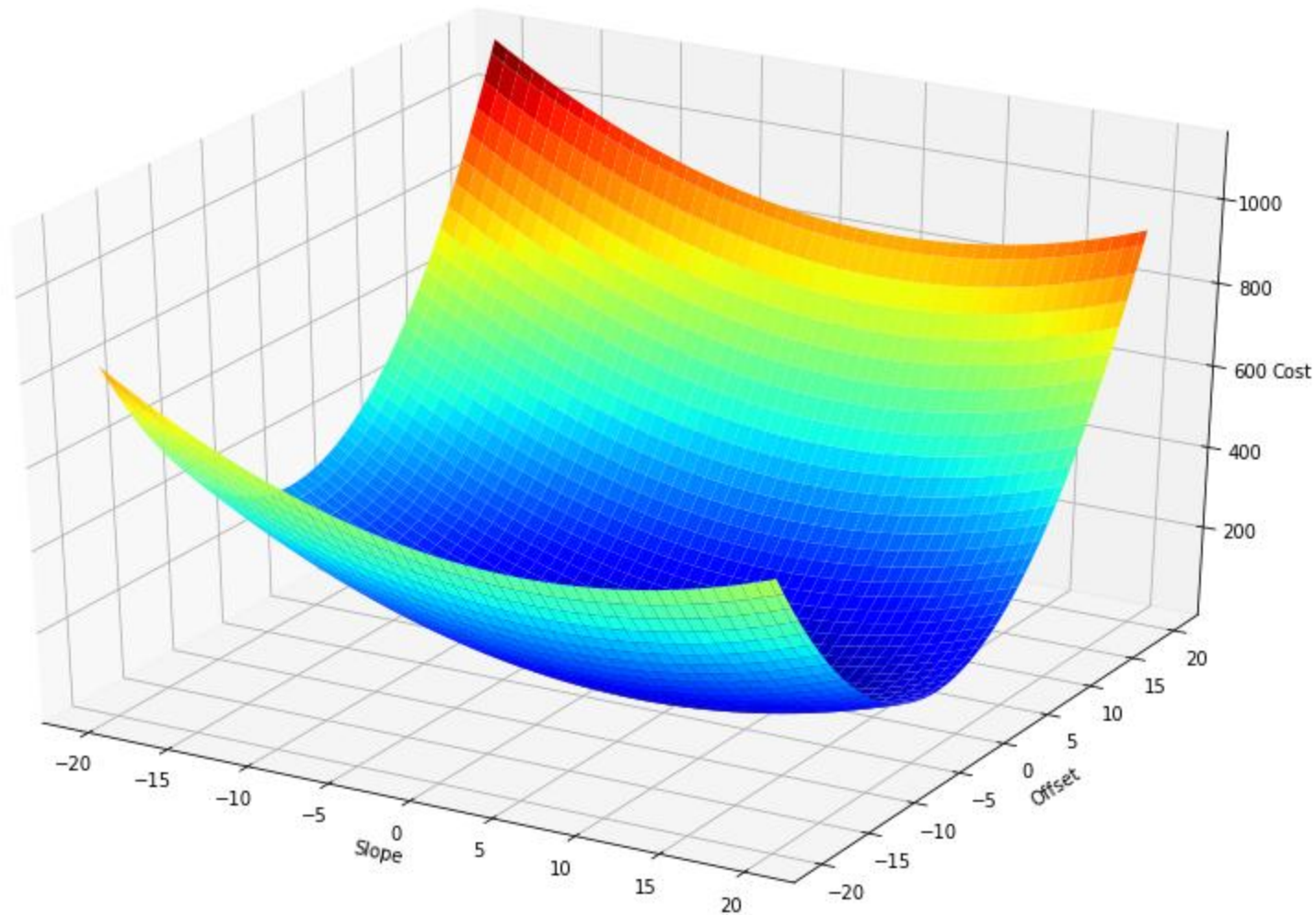


Cost Function

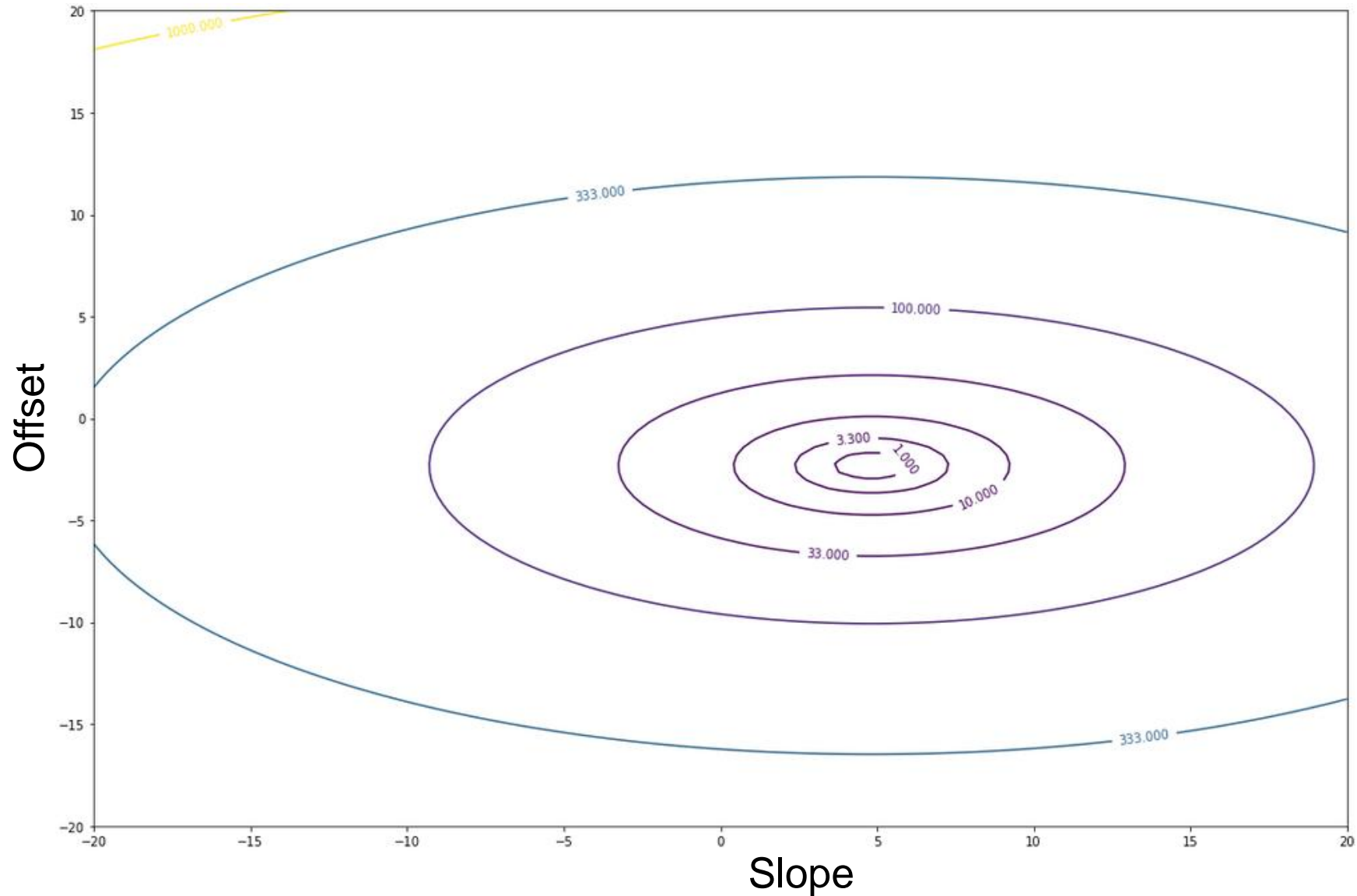


```
def cost(x, slope, offset, y):
    return np.mean(np.square(hypothesis(x, slope, offset) - y)) / 2.0
```

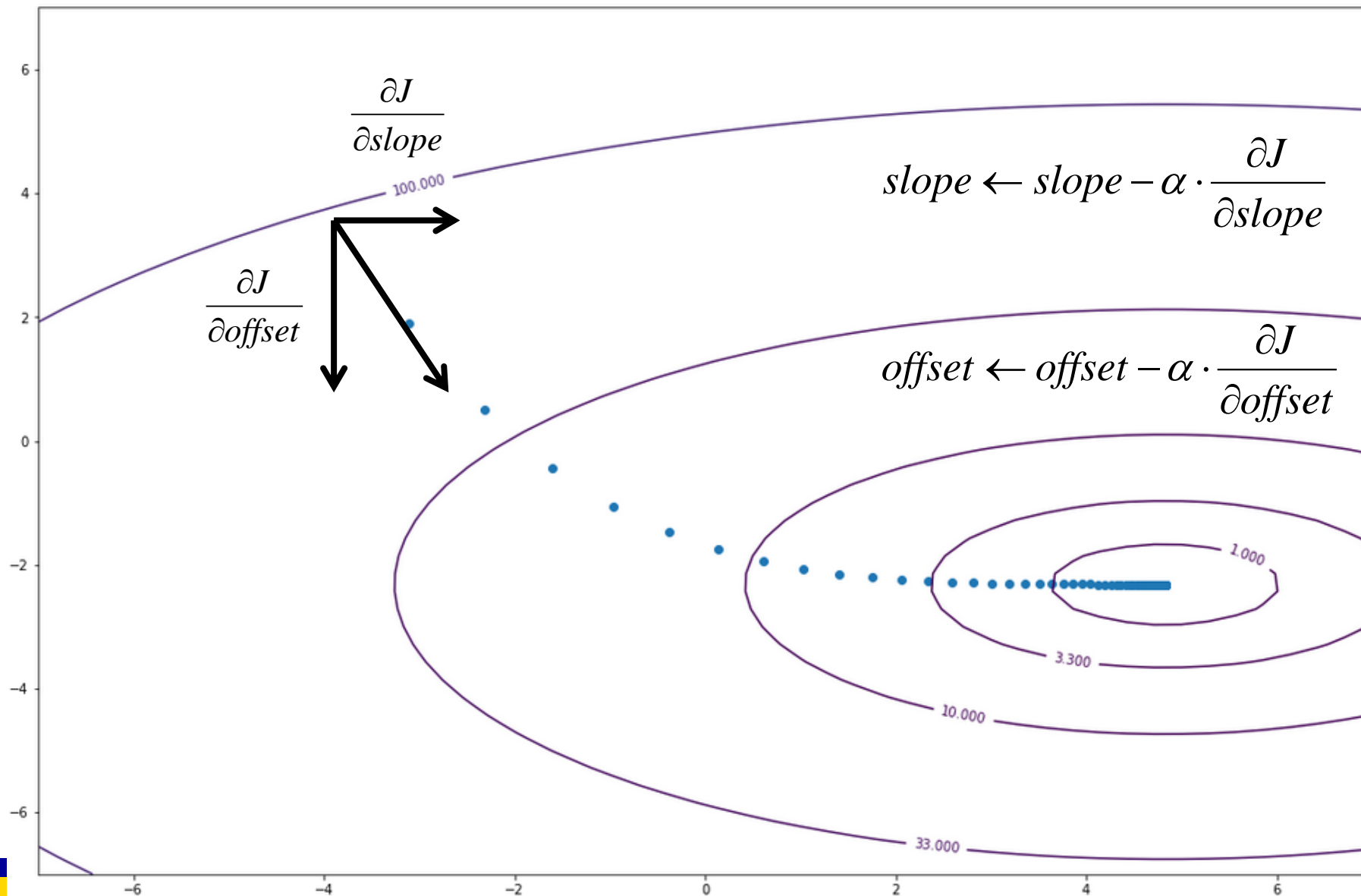
Cost as a Function of Slope and Offset



Contour Plot of Cost Function



Gradient Descent



Gradient Descent Algorithm

```
num_steps      = 3000  
learning_rate  = 0.001
```

```
slope  = 4.0  
offset = -4.0
```

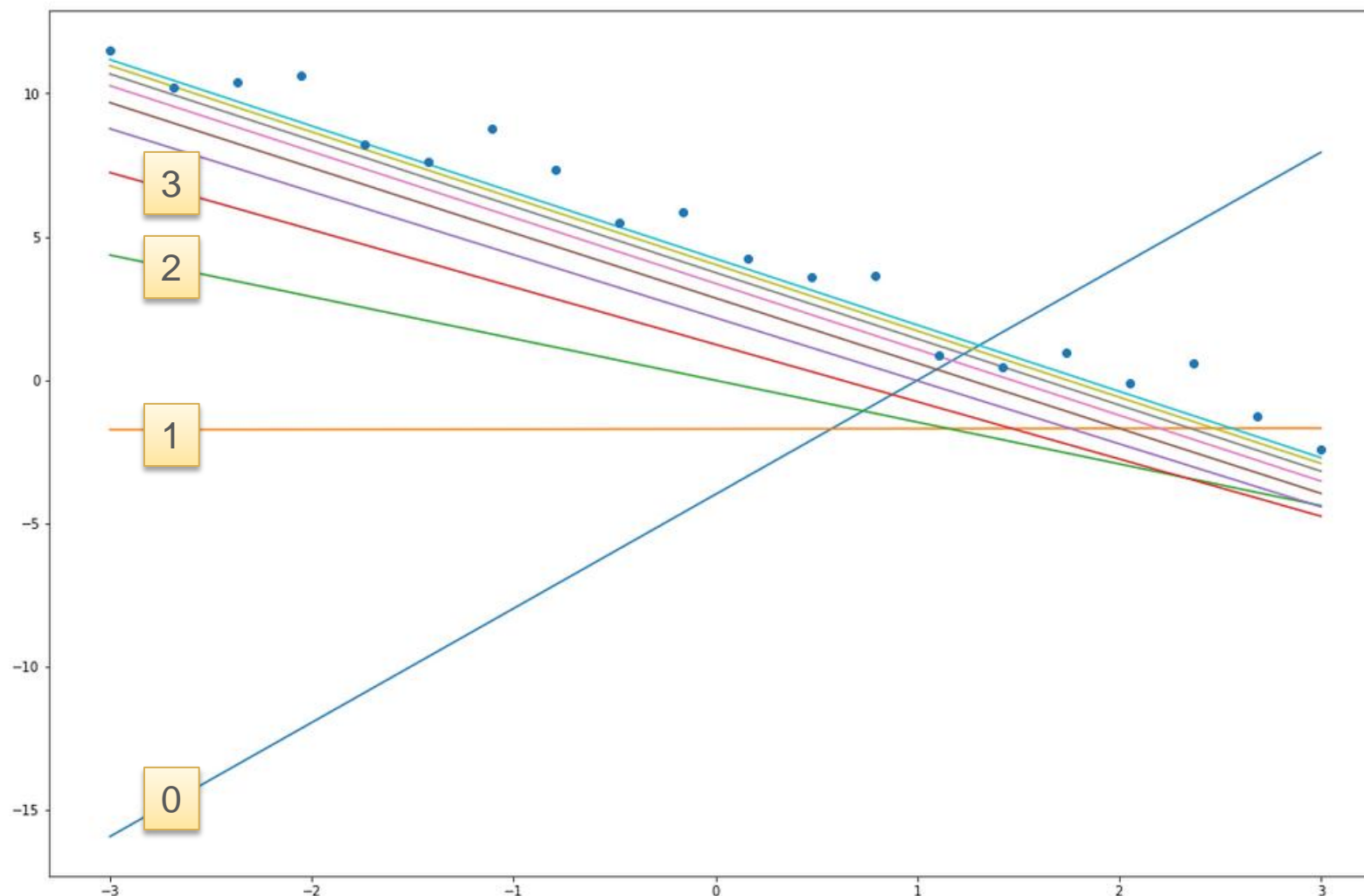
```
def derivative_wrt_slope():  
    return np.mean((hypothesis(x, slope, offset) - y) * x)
```

```
def derivative_wrt_offset():  
    return np.mean((hypothesis(x, slope, offset) - y))
```

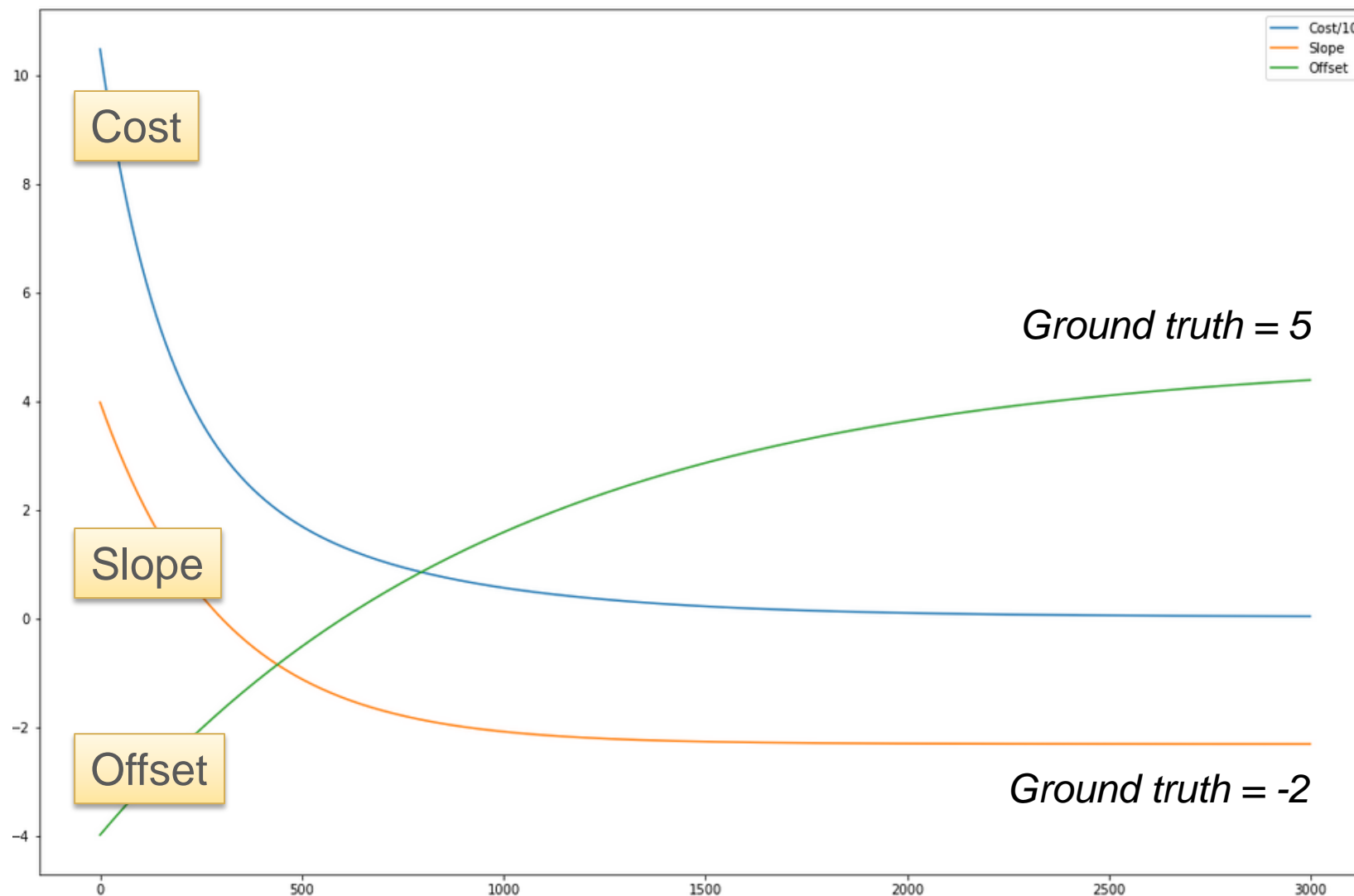
```
for step in range(num_steps):  
    new_slope  = slope  - learning_rate * derivative_wrt_slope()  
    new_offset = offset - learning_rate * derivative_wrt_offset()  
    slope      = new_slope  
    offset     = new_offset
```

Converging on the Minimum

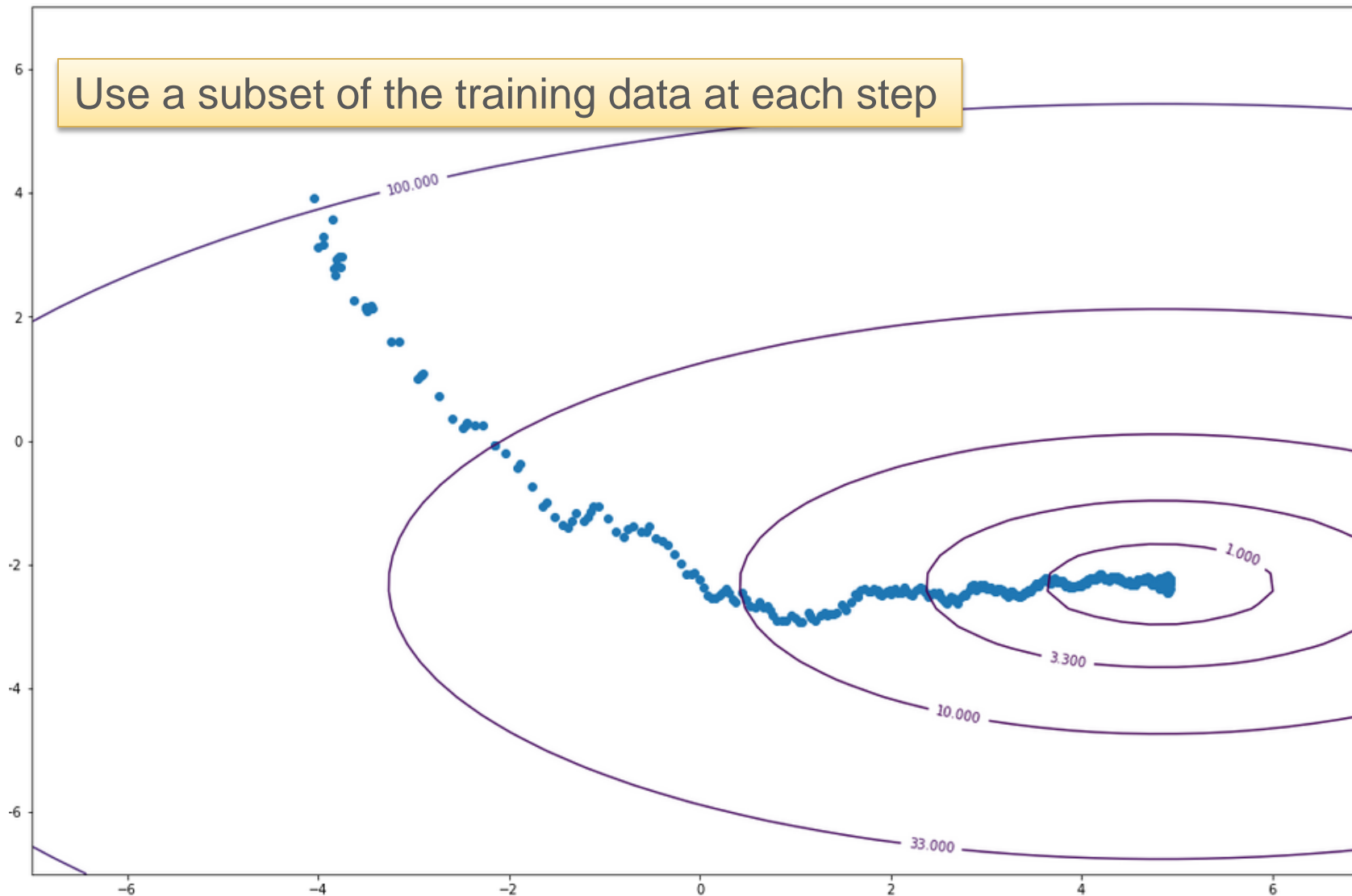
Final slope = -2.31499114425 offset = 4.38980555415



Cost, Slope, Offset against Step

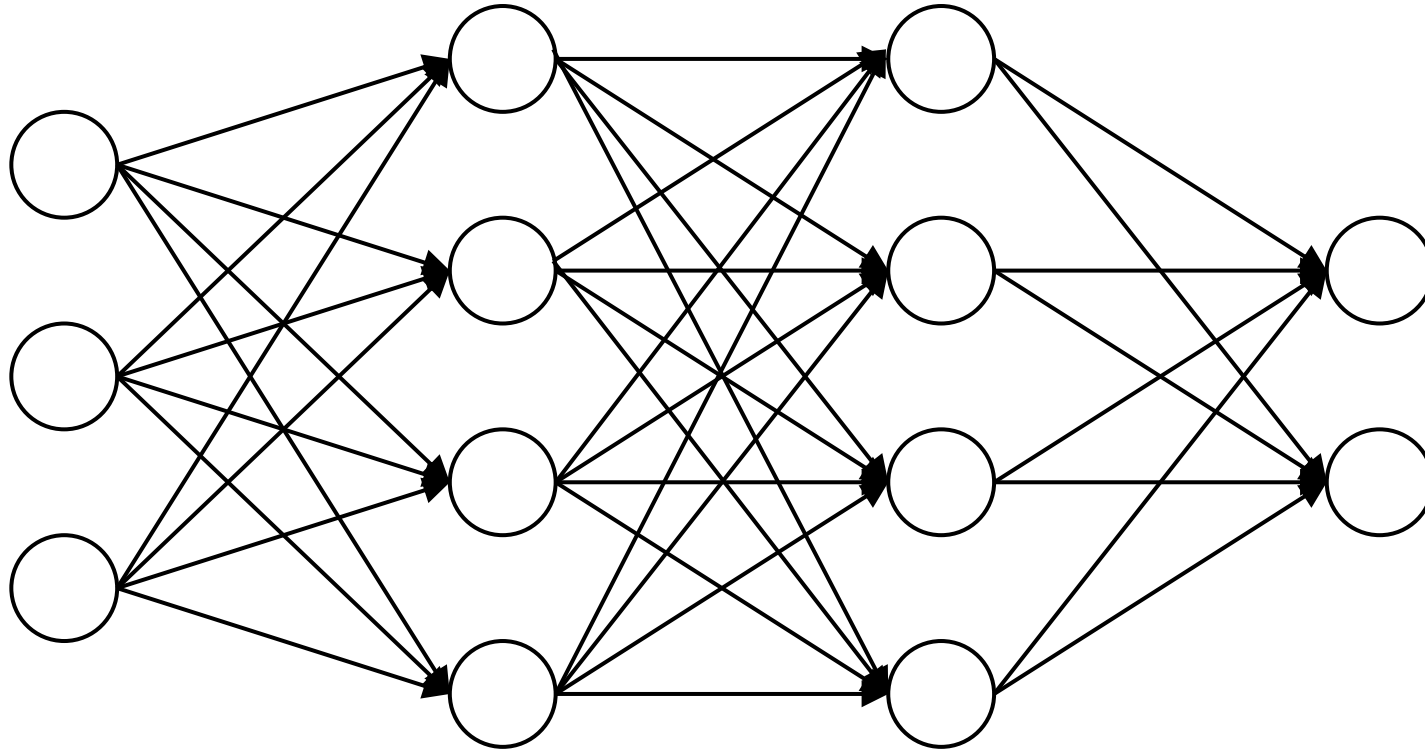


Stochastic Gradient Descent



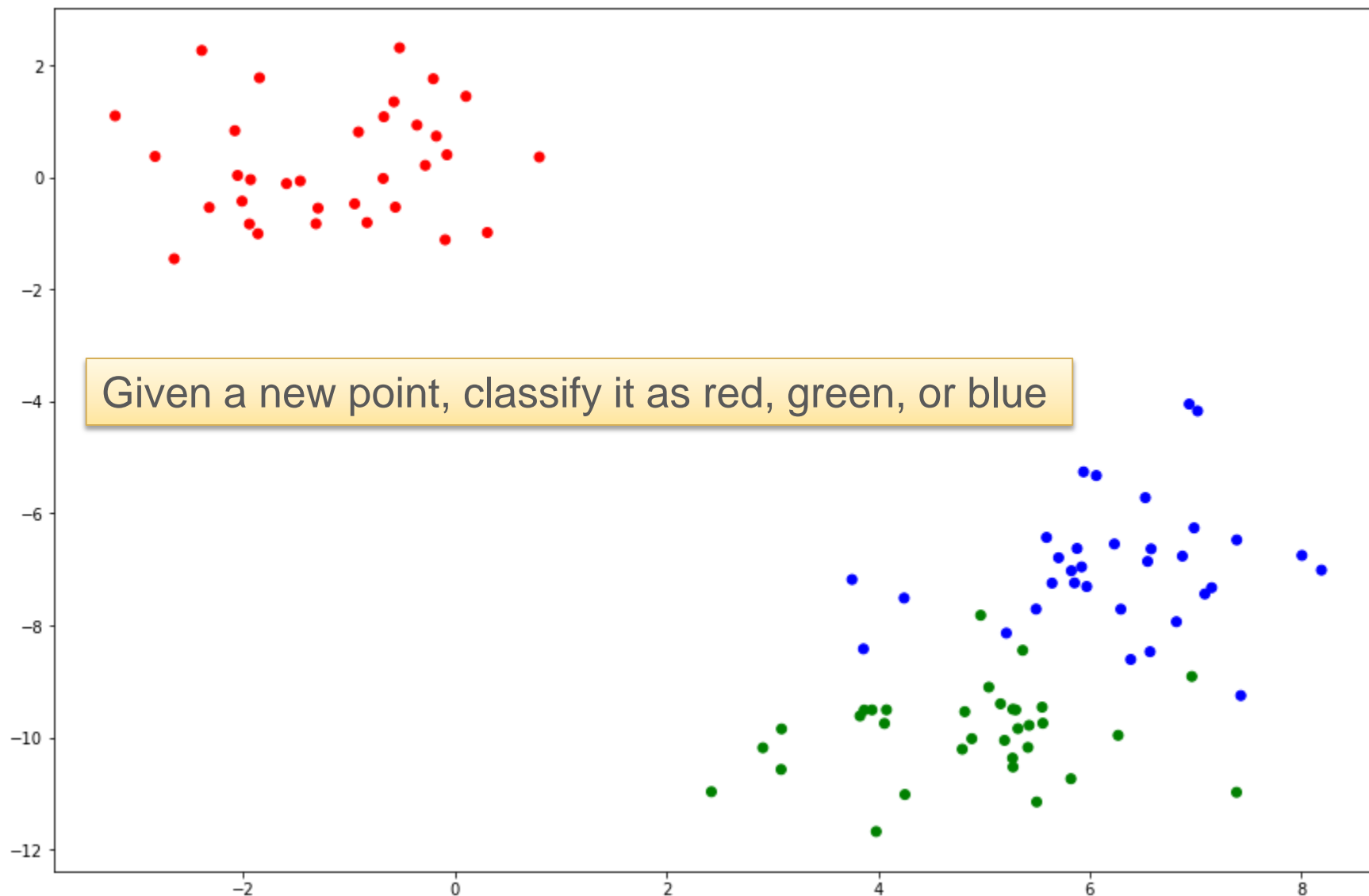
Forward and Back-Propagation

Forward propagation calculates weighted sums and activation function



Back propagation calculates gradients

Classification Task



One-Hot Labels

Category: Red Green Blue

Numeric label
(single output):

1

2

3

Would introduce a bias

One-hot label
(3 outputs):

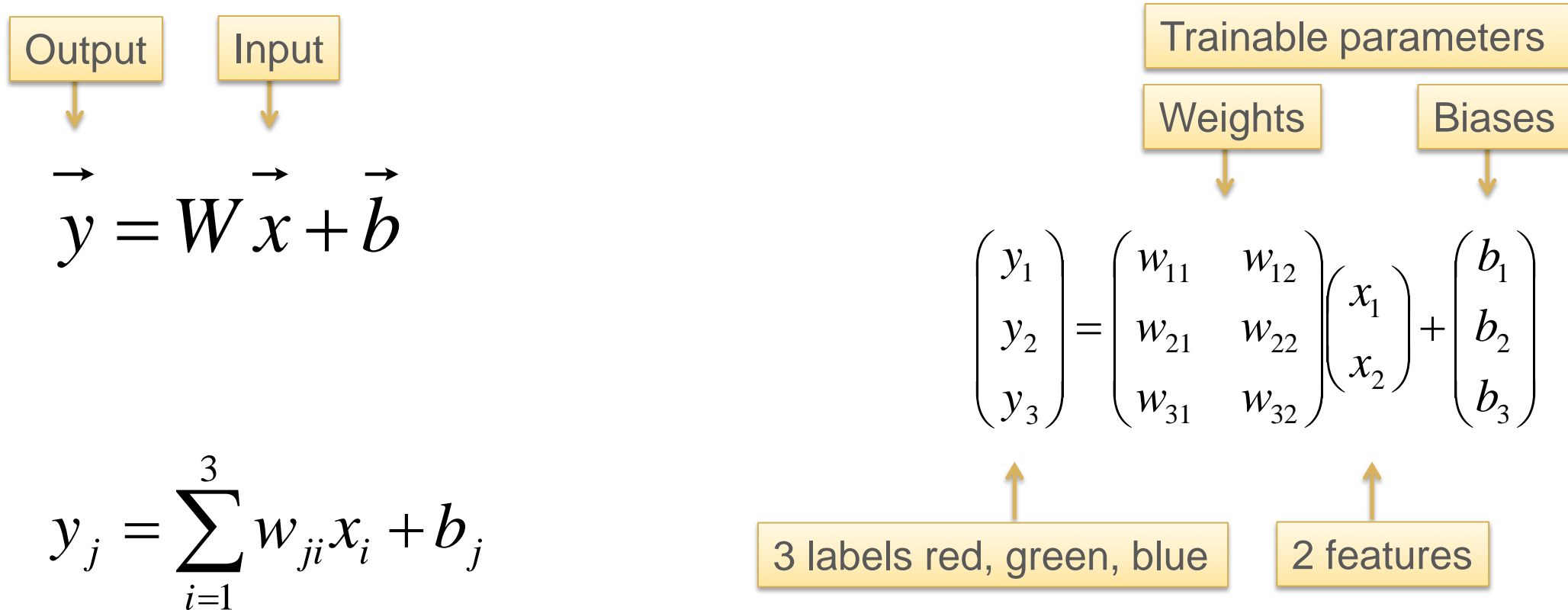
$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

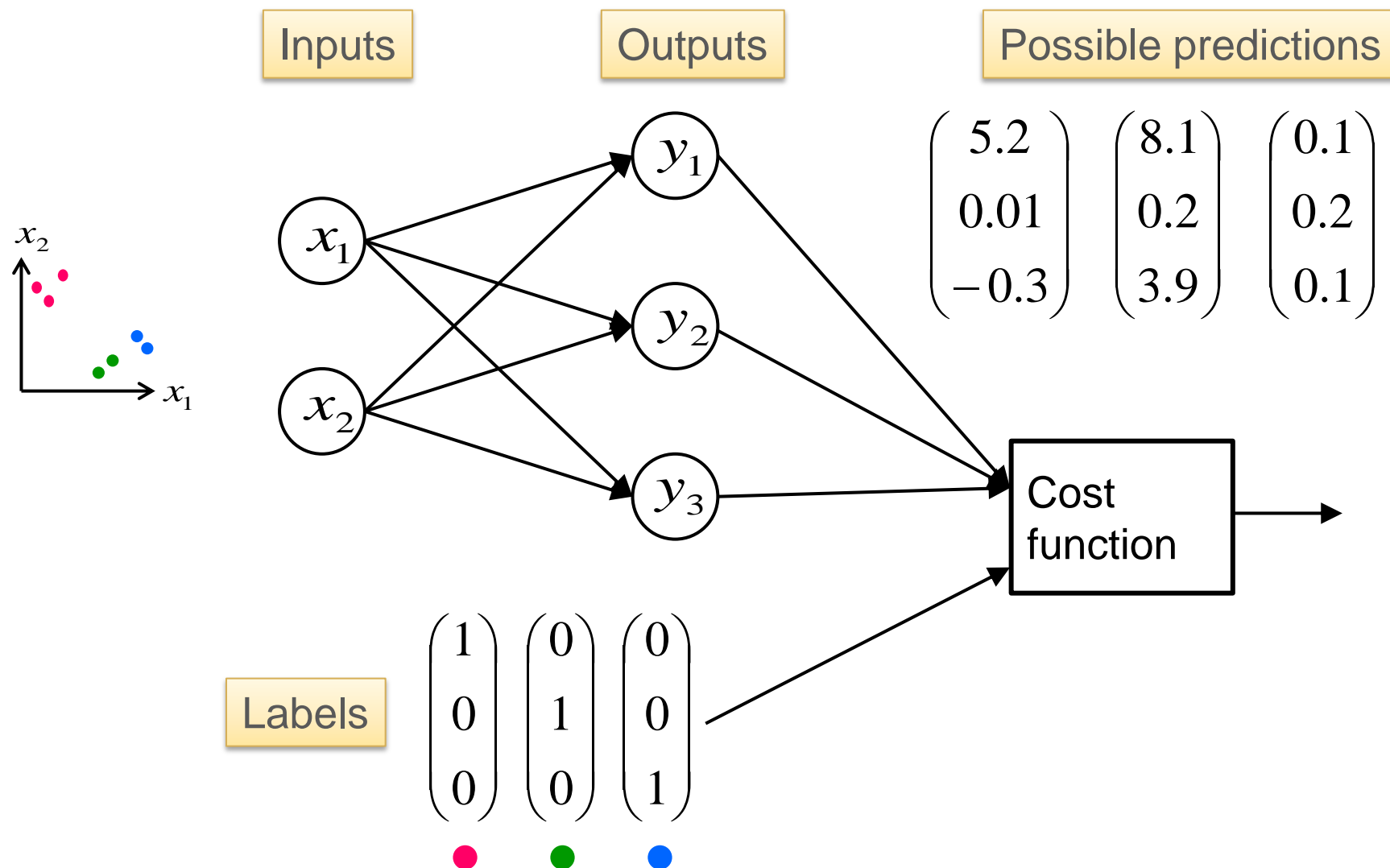
$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

No bias

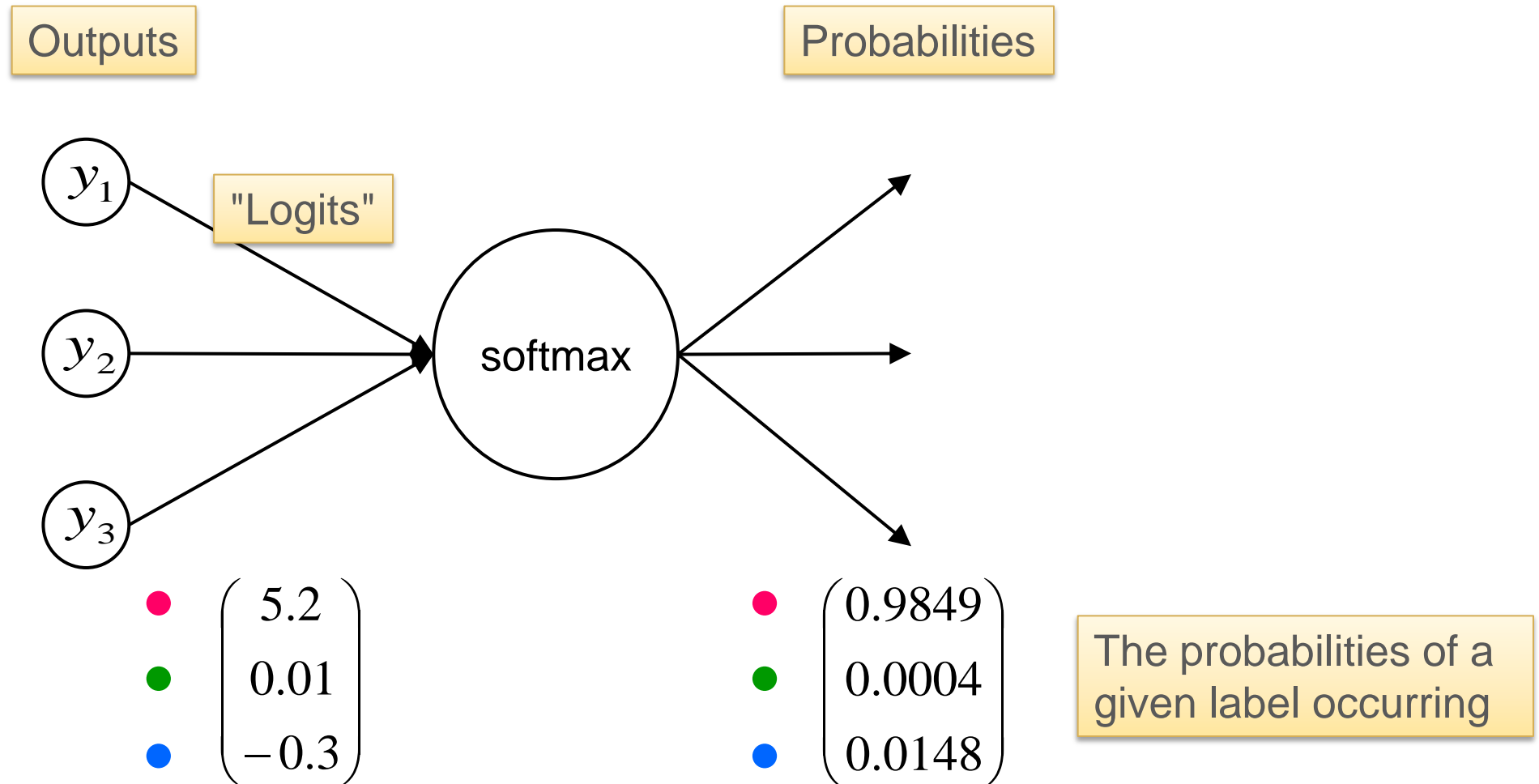
The Hypothesis or Model



Calculating the Cost Function



Converting Scores to Probabilities



The Softmax Function

$$\sigma(y)_j = \frac{e^{y_j}}{\sum_k e^{y_k}}$$

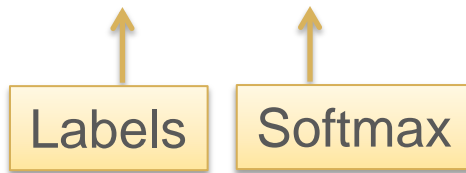
$$\begin{pmatrix} -0.1 \\ 0.0 \\ +0.1 \end{pmatrix} \rightarrow \begin{pmatrix} 0.301 \\ 0.332 \\ 0.367 \end{pmatrix} \xleftrightarrow{\text{compare}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1.0 \\ 2.0 \\ 3.0 \end{pmatrix} \rightarrow \begin{pmatrix} 0.090 \\ 0.245 \\ 0.665 \end{pmatrix} \xleftrightarrow{\text{compare}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \rightarrow \begin{pmatrix} 2 \times 10^{-9} \\ 5 \times 10^{-5} \\ 9.999 \end{pmatrix} \xleftrightarrow{\text{compare}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

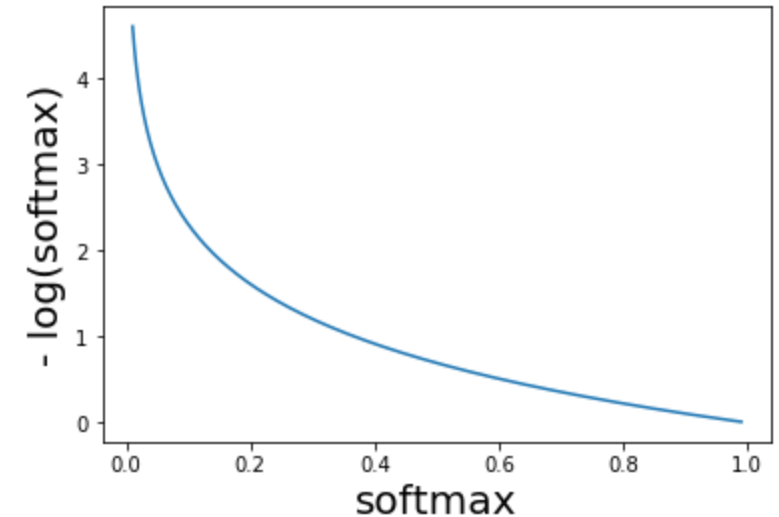
Compare using Cross-Entropy

$$H(\vec{\sigma}, \vec{\ell}) = - \sum_i \ell_i \log(\sigma_i)$$



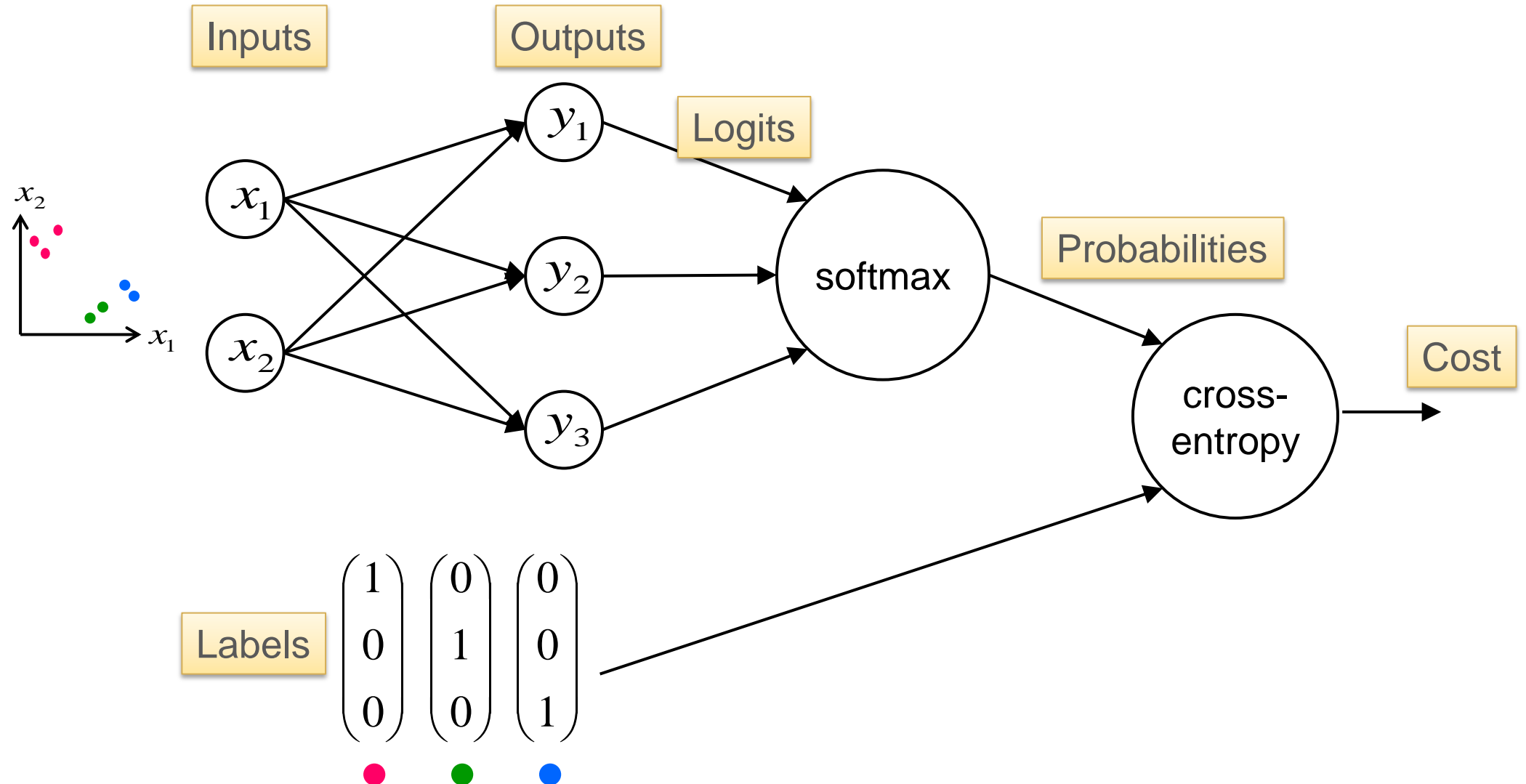
$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0.1 \\ 0.2 \\ 0.7 \end{pmatrix} \sigma_i \in (0,1)$$

The third column vector $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ and the third row vector $\begin{pmatrix} 0.1 \\ 0.2 \\ 0.7 \end{pmatrix}$ are highlighted with a red box, indicating the calculation for the third class.

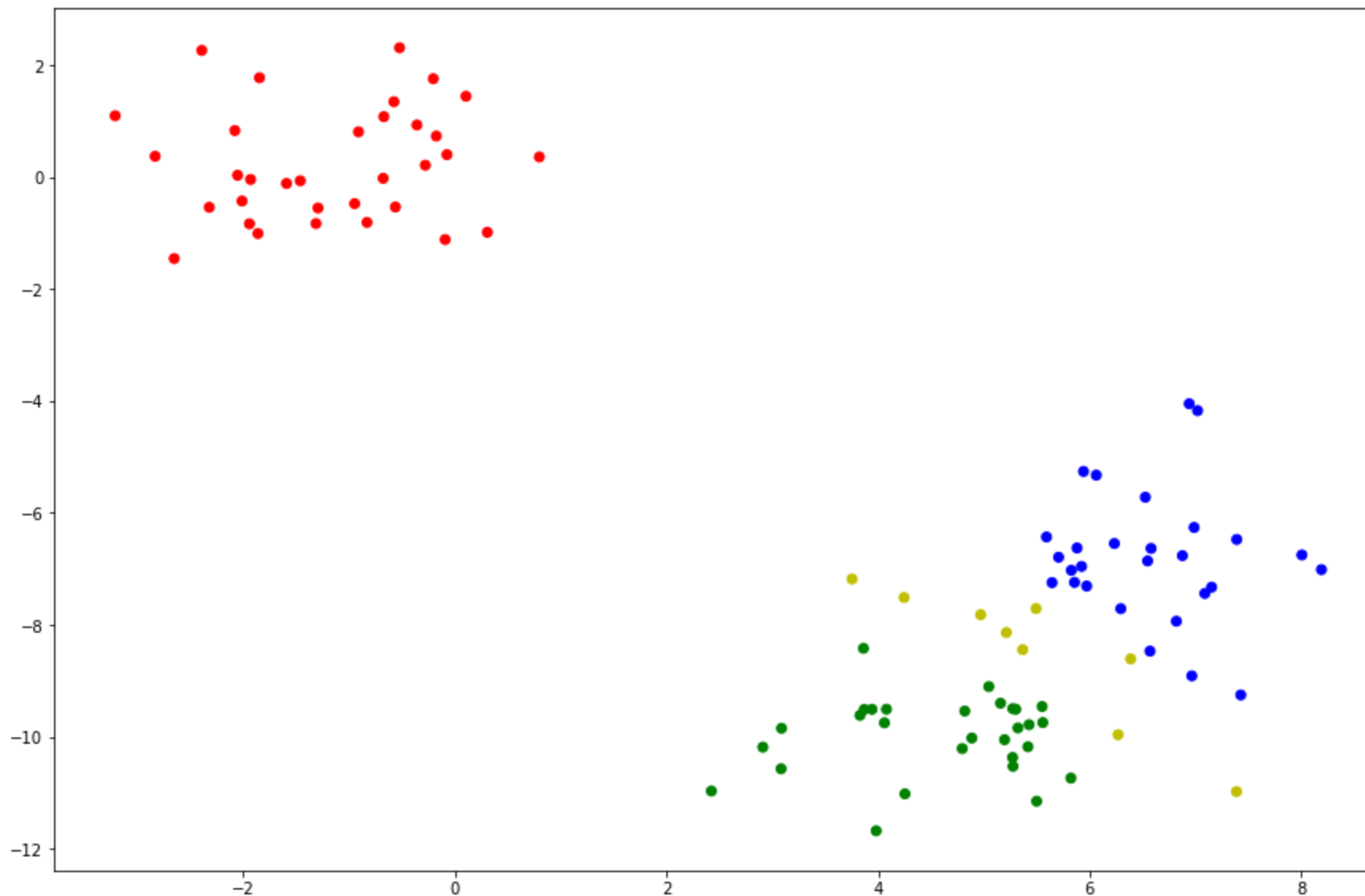


The closer the softmax value corresponding to the given label is to 1, the closer the cost is to 0.

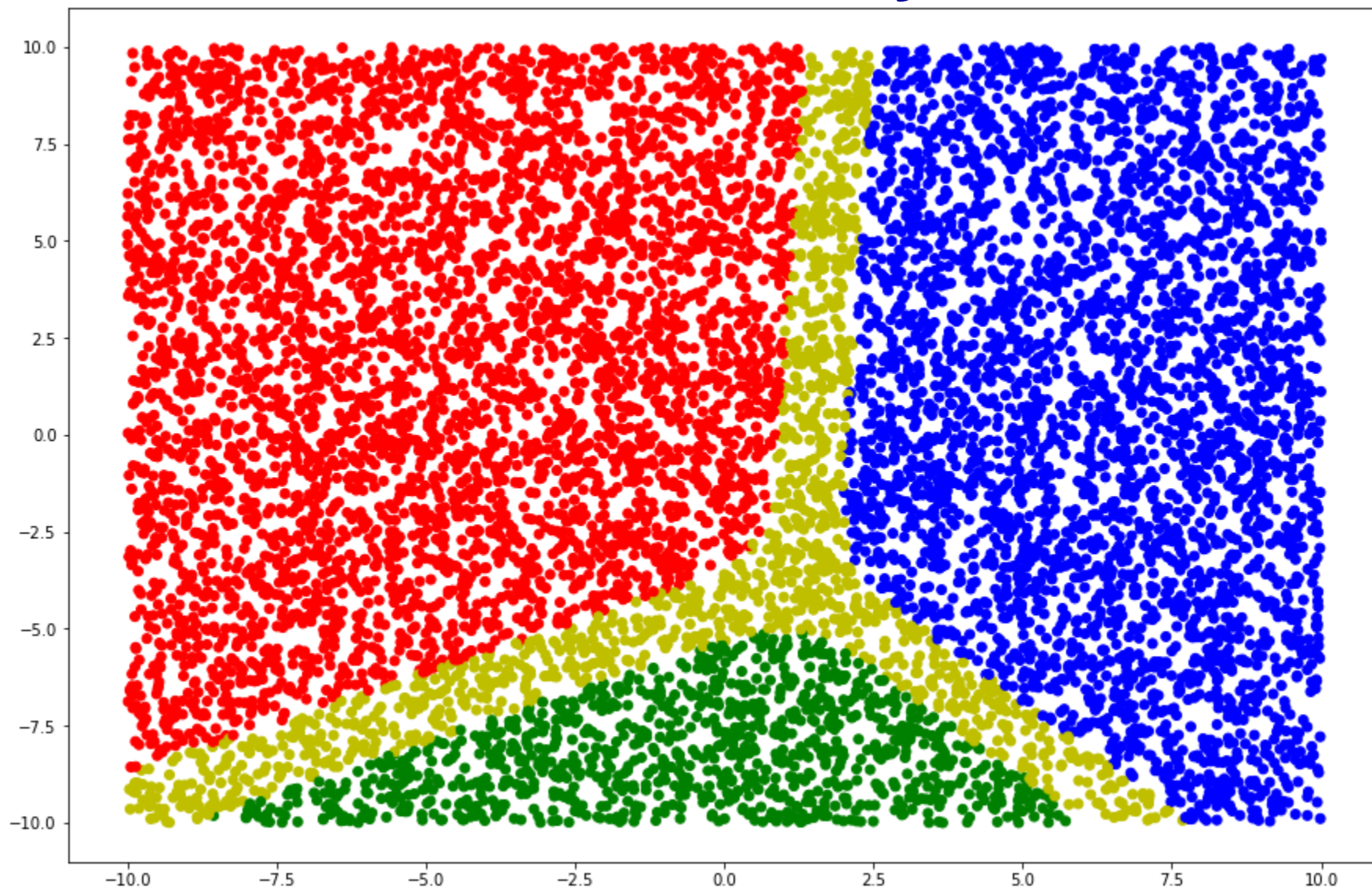
A Neural Network for Classification



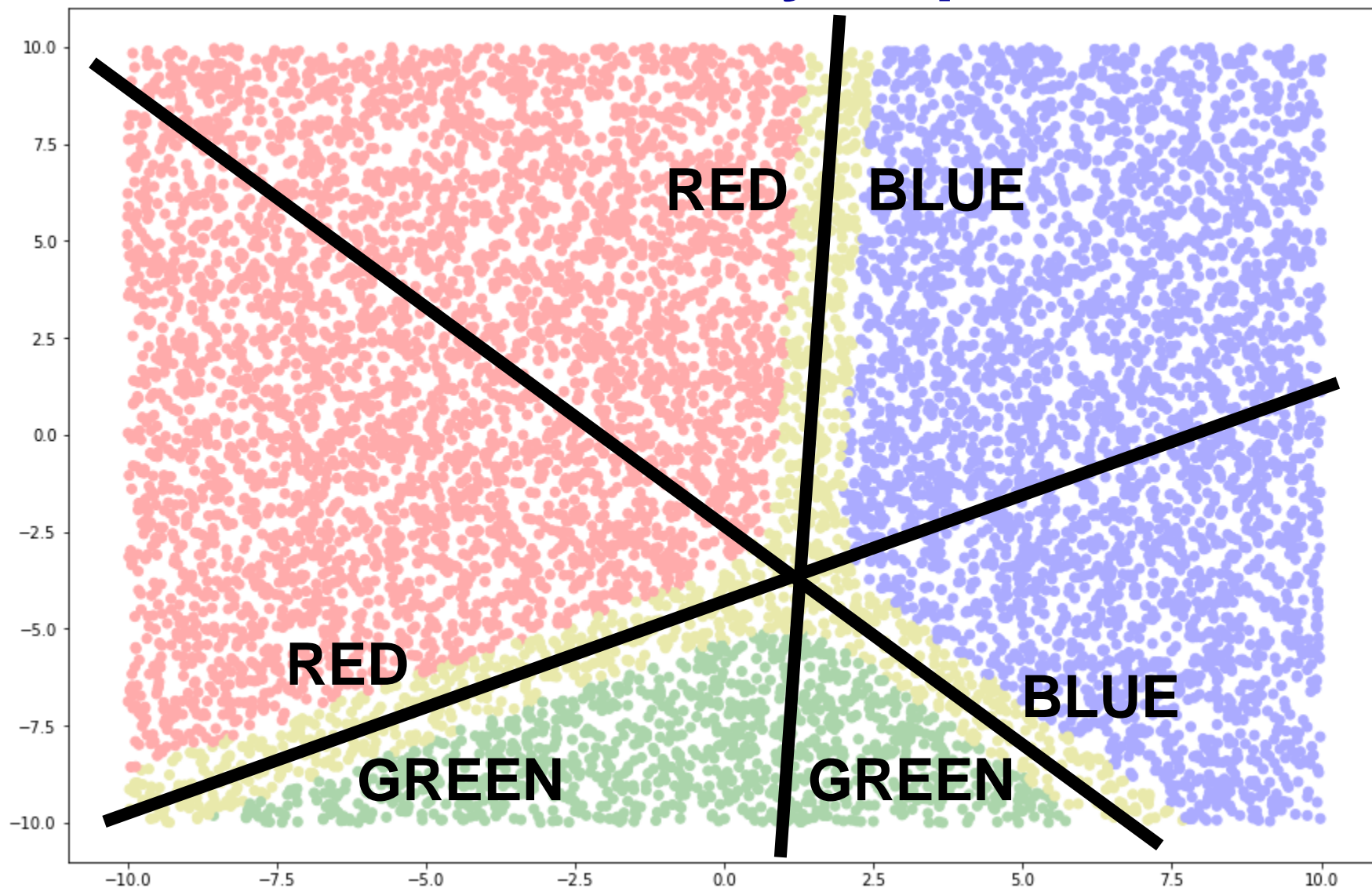
Predictions, Confidence > 0.8



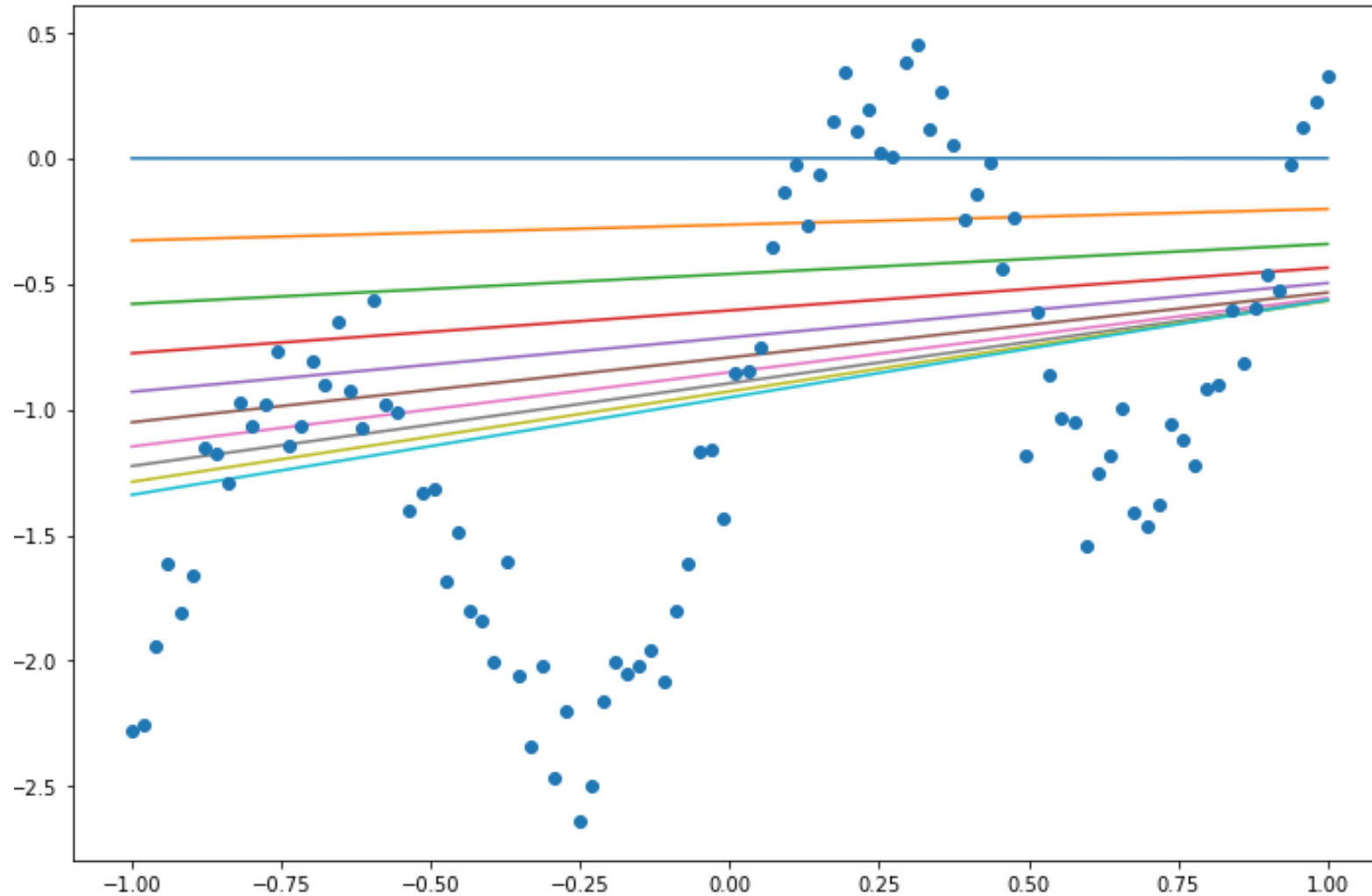
The Decision Boundary



Classes are Linearly Separable



Non-Linear Regression and Classification

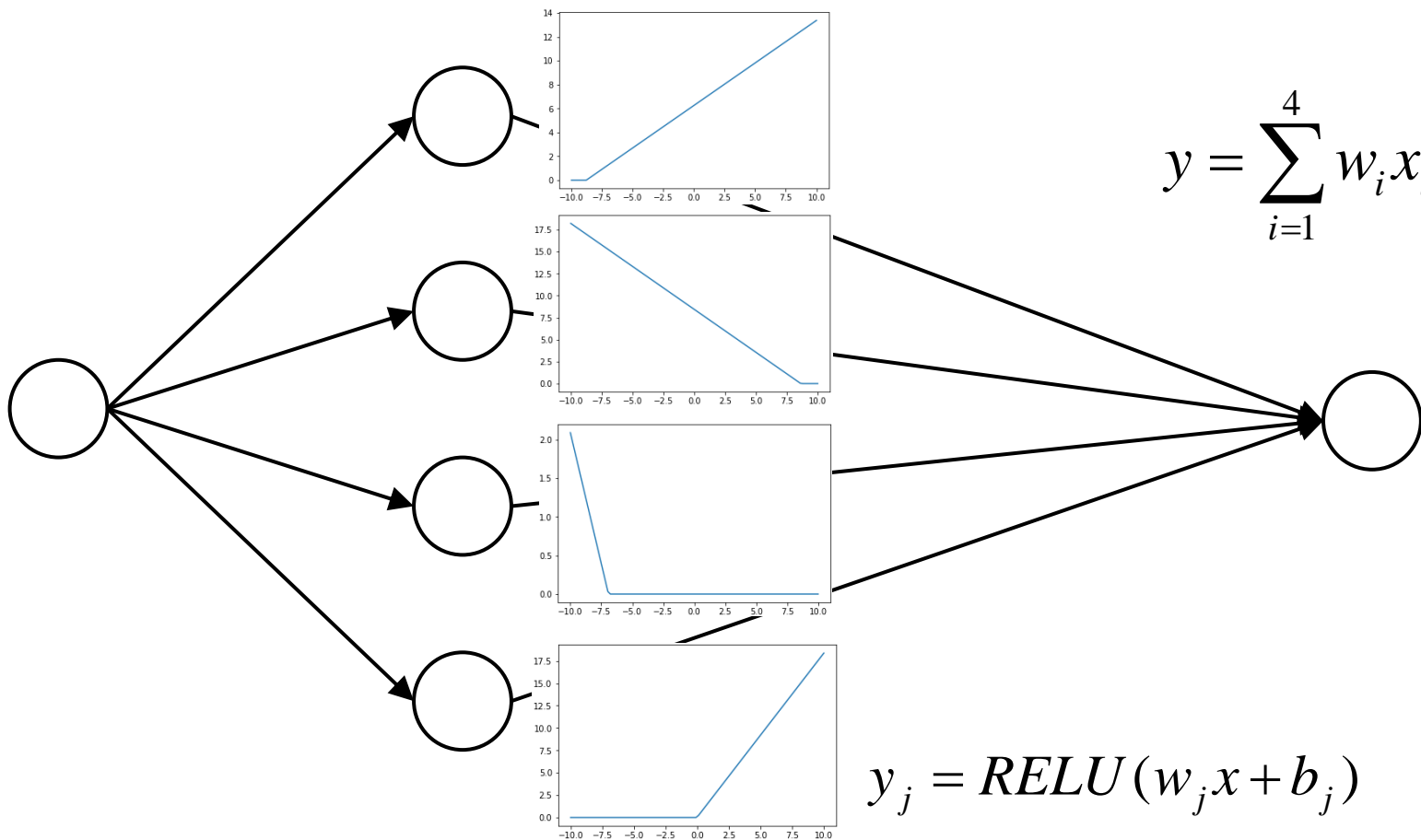


Piecewise Linear Approximation

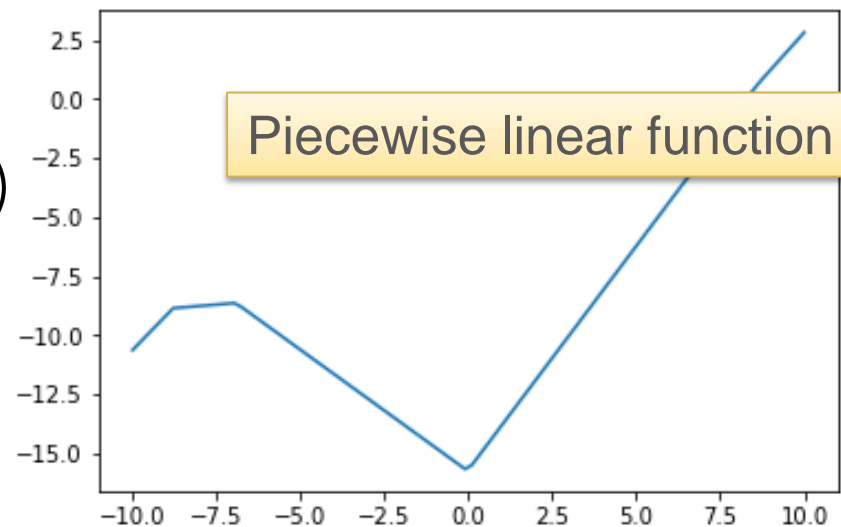
Input unit

Hidden units

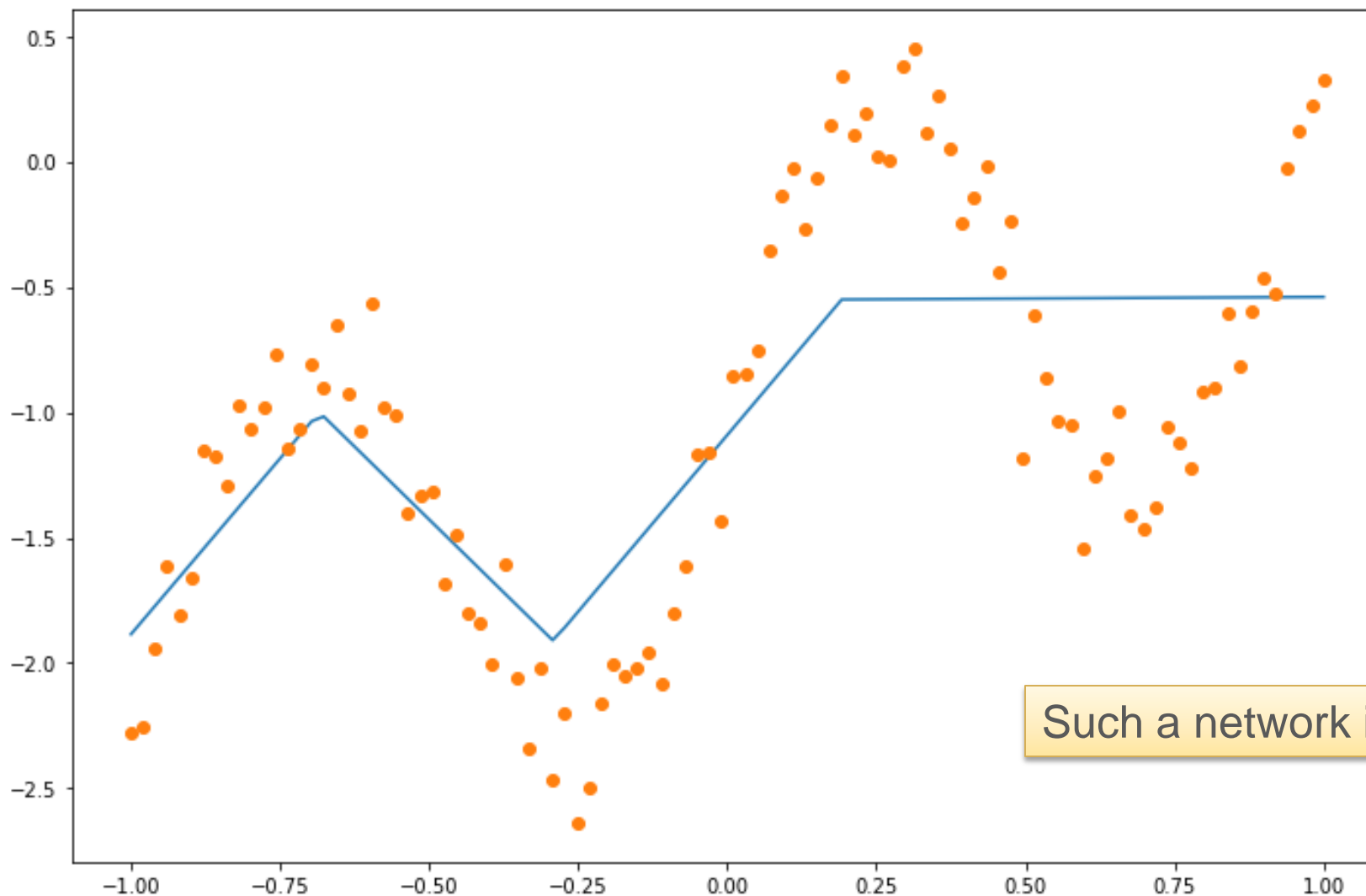
Output unit



$$y_j = \text{RELU}(w_j x + b_j)$$

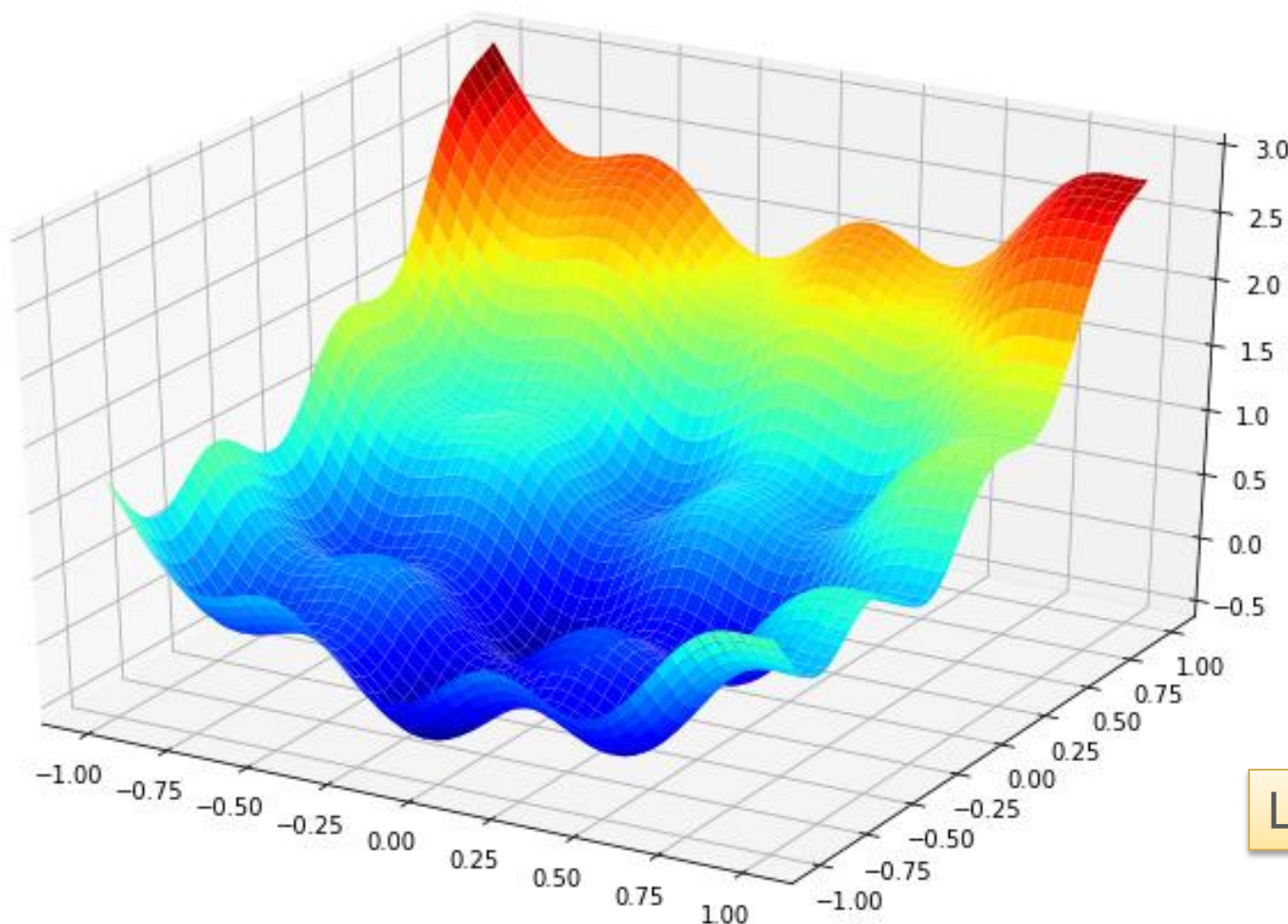


The Predicted Output



Such a network is hard to train

The Landscape of the Cost Function



Local and global minima

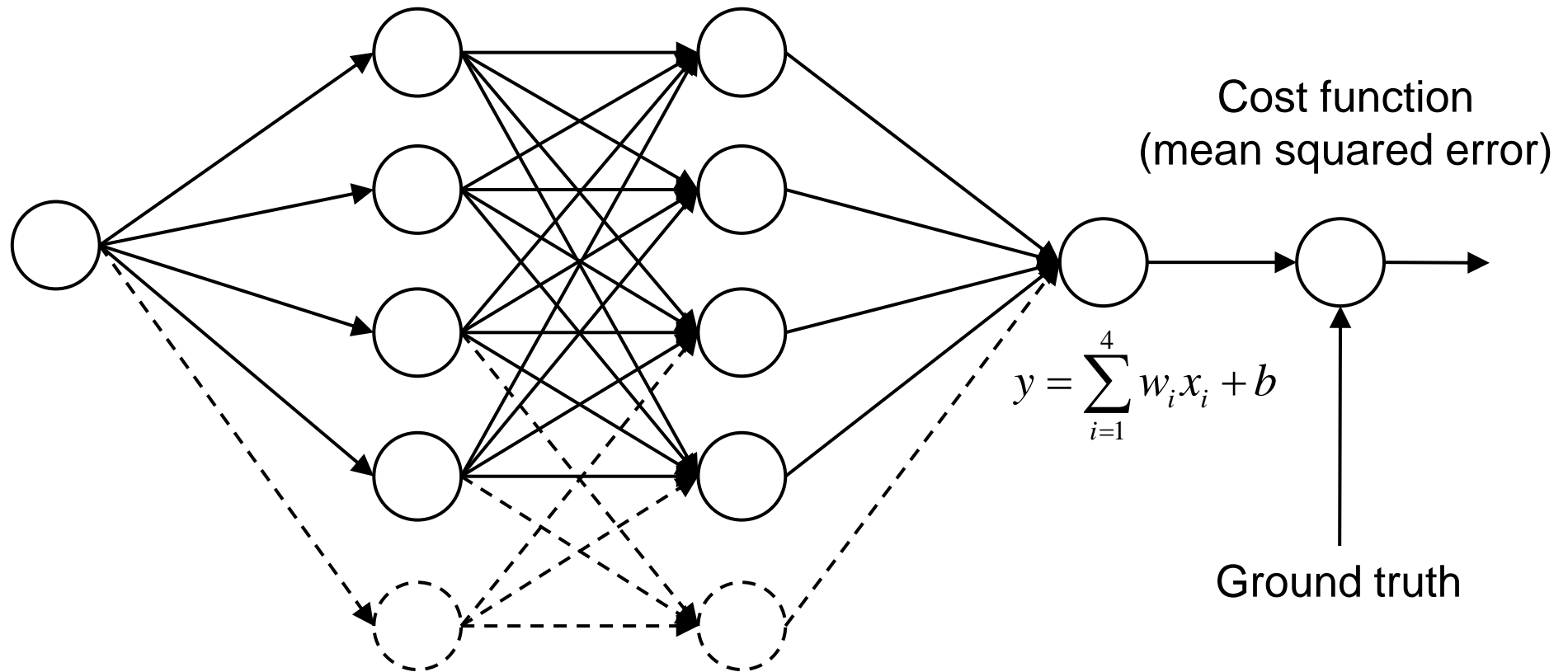
A Deep Neural Network

Input unit

16 hidden units

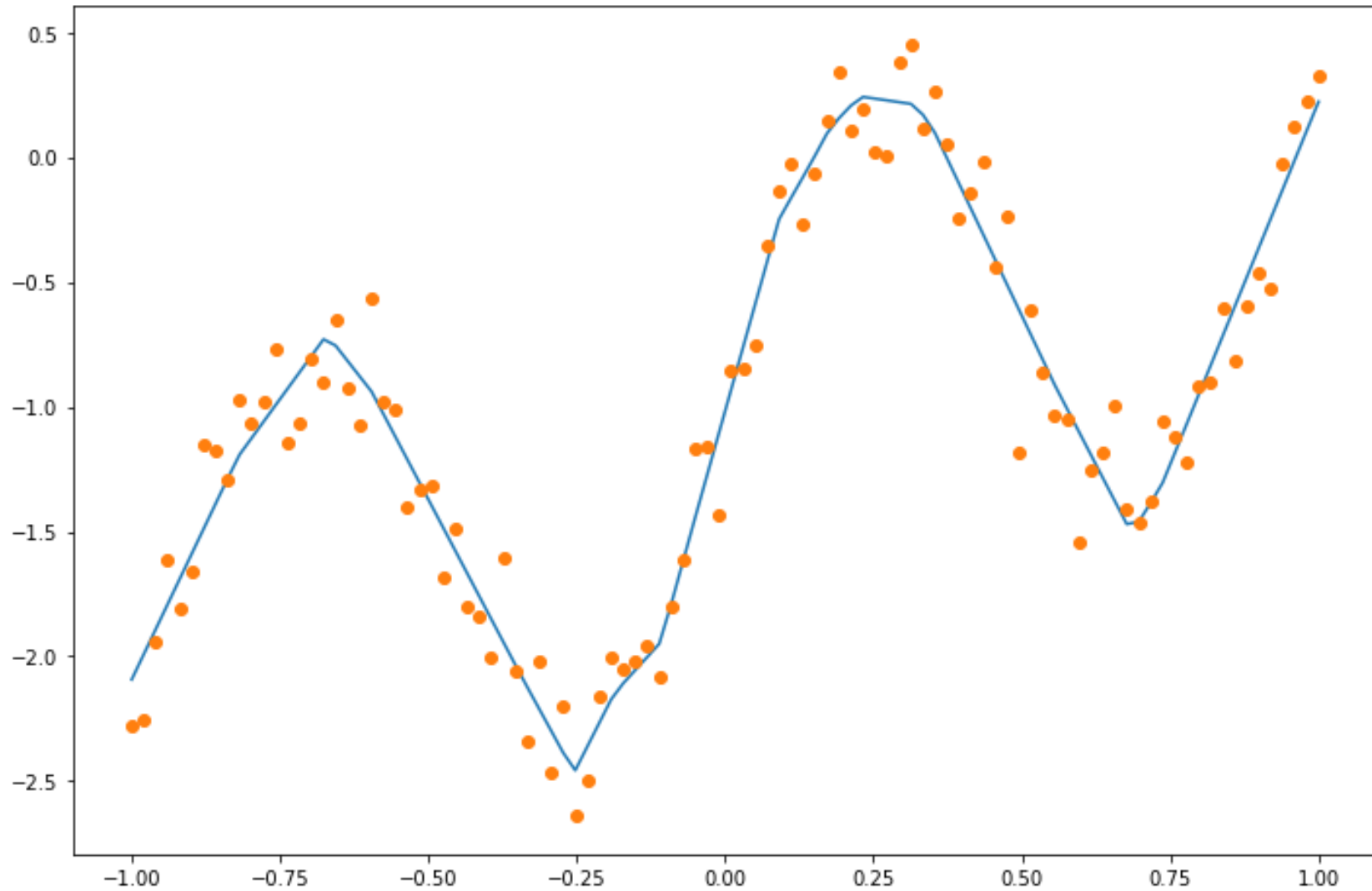
16 hidden units

Output unit



$$y_j = \text{RELU}(w_j x + b_j) \quad y_j = \text{RELU}(w_j x + b_j)$$

The Predicted Output



The Magic of Deep Neural Networks

Deep neural networks have many degrees of freedom / degenerate / redundant

Gradient descent tends not to get stuck in local minima

Gradient descent tends to find a good global minimum

Why?

Most stationary points are saddle points, not local minima?

Deep Learning for Design and Verification Engineers

What is Deep Learning?

Neural Networks

How a Network Learns



Getting Started

Libraries and Frameworks for Training

	Scikit-learn	Not deep learning
University of Montreal	Theano	Runs on GPU
	Pylearn2	(Theano)
	Lasagne	(Theano)
Berkeley AI Research	Caffe	Runs on GPU
Facebook, Twitter, ...	Torch, PyTorch	Runs on GPU
Google	TensorFlow	Runs on GPU
Microsoft	CNTK	Cognitive Tool Kit
	Keras	(Theano, TensorFlow, CNTK)
Skyminid	DL4J	Deep Learning for Java
Apache	MXNet	
Intel	Neon	Optimized for Intel CPUs
MathWorks	MATLAB	Various toolboxes

Deep Learning Platforms and Toolkits

Amazon Deep Learning AMIs

Au-Zone DeepView

Google Cloud Machine Learning Engine

IBM Watson

Intel Nervana Cloud

Microsoft Azure Machine Learning Studio

MVTec HALCON

NVIDIA TensorRT

Qualcomm Snapdragon NPE SDK

Xilinx reVISION

And many more! All trademarks acknowledged

Deep Learning IP and Chips

BrainChip

Cadence Tensilica Vision DSP

Google TPU

Graphcore

IBM TrueNorth

Intel Loihi

Intel Movidius

KAIST MVLSI Laboratory

KALRAY MPPA

Synopsys DesignWare EV6x

And many more! All trademarks acknowledged

Python and Jupyter Notebook

Simple Example of Non-Linear Regression using Keras

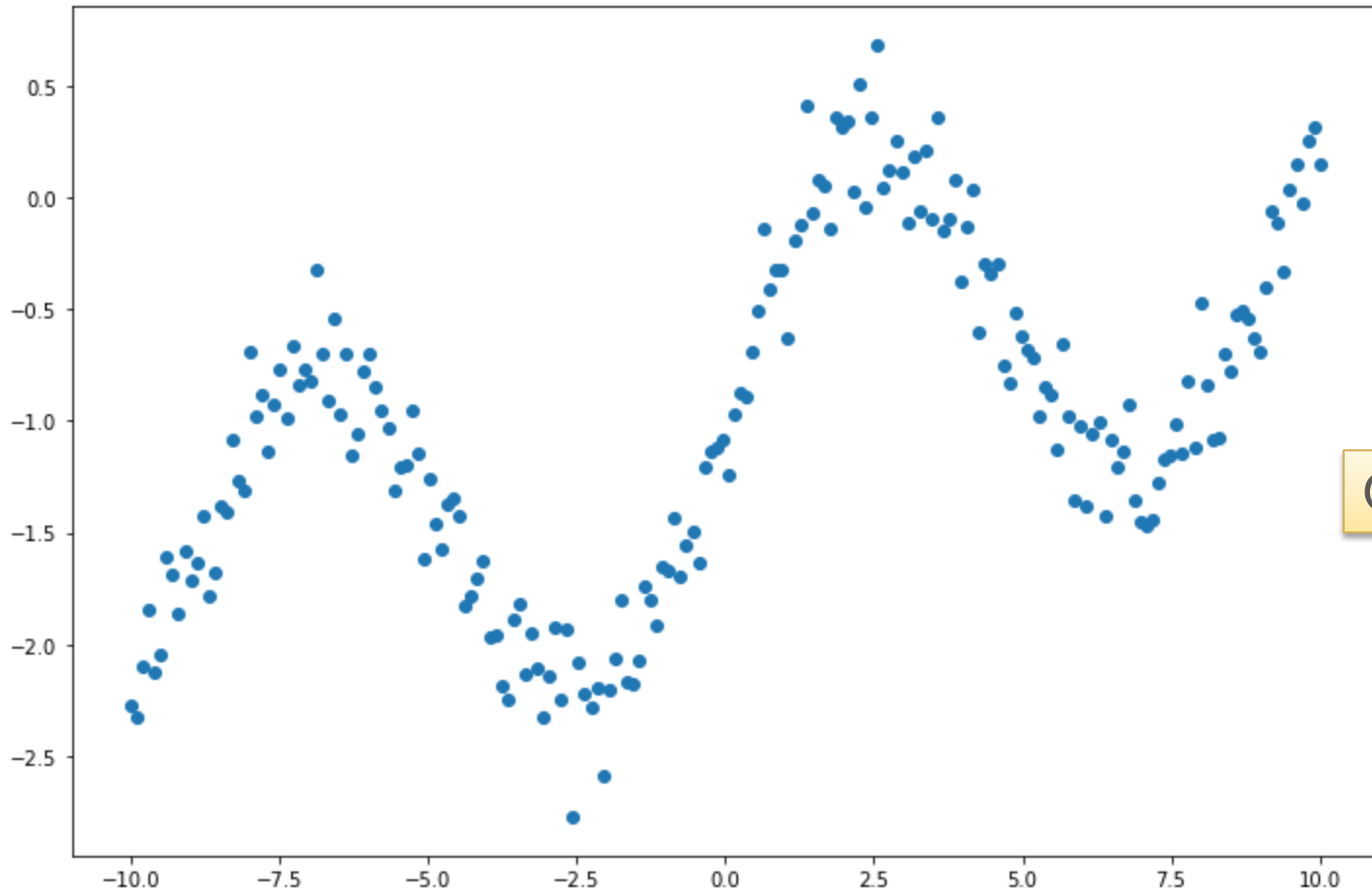
Keras is an API on top of TensorFlow (or Theano, another ML library) that hides and abstracts a lot of the detail when building deep neural network models. Keras models are a lot more compact and readable than low-level TensorFlow models, although because a lot of the detail is obscured, it might not be so clear to beginners what is going on under-the-hood.

First run the code to generate and plot the dataset.

```
In [43]: 1 import numpy as np
          2 import matplotlib.pyplot as plt
          3
          4 m = 200
          5 train_slope = 0.1
          6 train_offset = -1.0
```

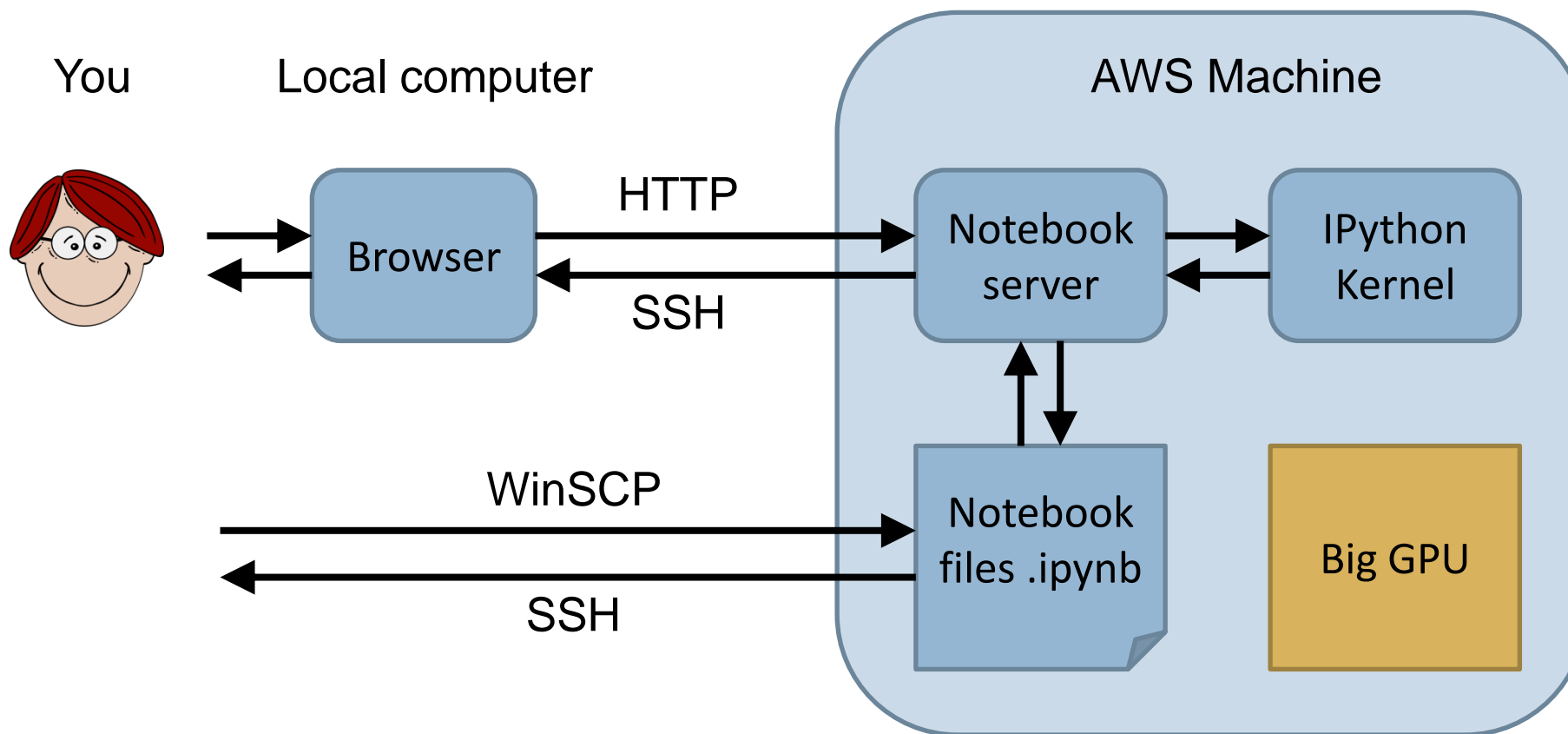
```
5 train_slope = 0.1
6 train_offset = -1.0
7
8 train_x = np.linspace(-10, 10, m).astype(np.float32)
9 rng = np.random.RandomState(seed=42)
10 train_y = (train_slope * train_x + np.sin(train_x / 1.5) + train_offset +
11           rng.normal(0.0, 0.2, size=len(train_x))).astype(np.float32)
12
13 plt.rcParams["figure.figsize"] = (12, 8)
14 plt.plot(train_x, train_y, 'o');
15 plt.show()
```

Python code

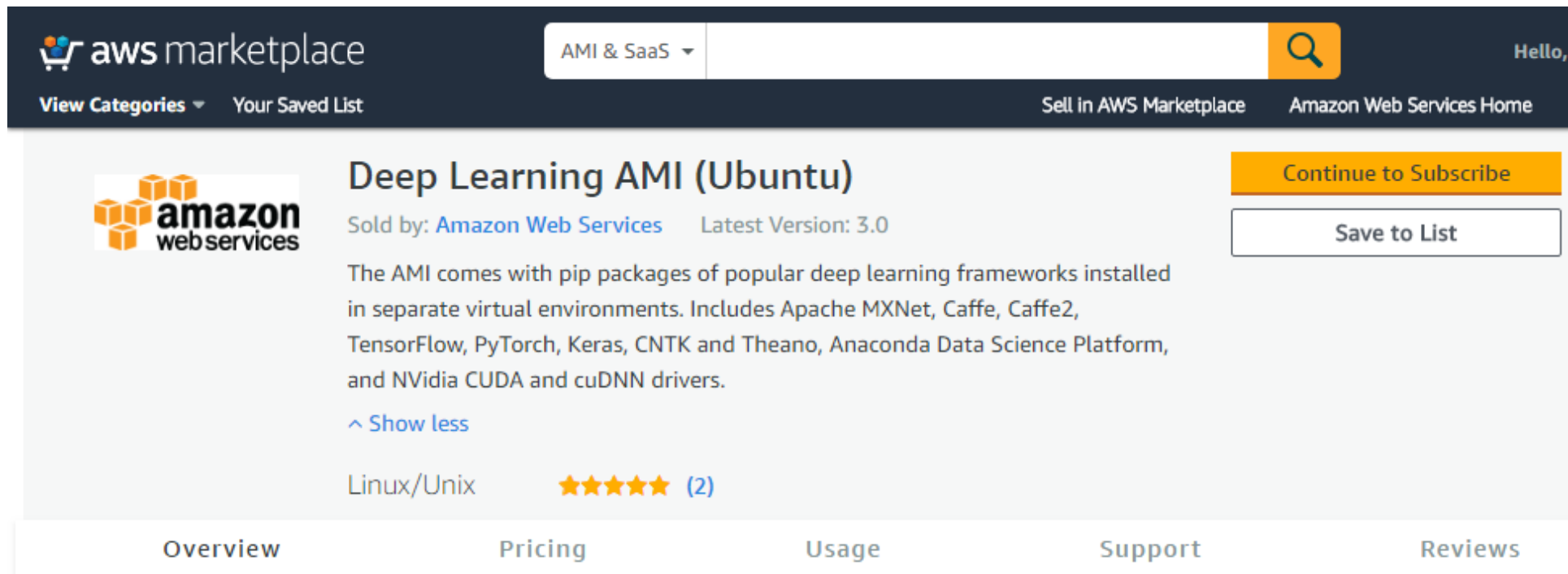


Output

Jupyter Notebook Architecture



AWS Deep Learning AMI




aws marketplace

AMI & SaaS

Hello,

View Categories ▾ Your Saved List

Sell in AWS Marketplace Amazon Web Services Home

 **Deep Learning AMI (Ubuntu)**

Sold by: [Amazon Web Services](#) Latest Version: 3.0

The AMI comes with pip packages of popular deep learning frameworks installed in separate virtual environments. Includes Apache MXNet, Caffe, Caffe2, TensorFlow, PyTorch, Keras, CNTK and Theano, Anaconda Data Science Platform, and NVidia CUDA and cuDNN drivers.

[^ Show less](#)

Linux/Unix ★★★★★ (2)

[Continue to Subscribe](#)

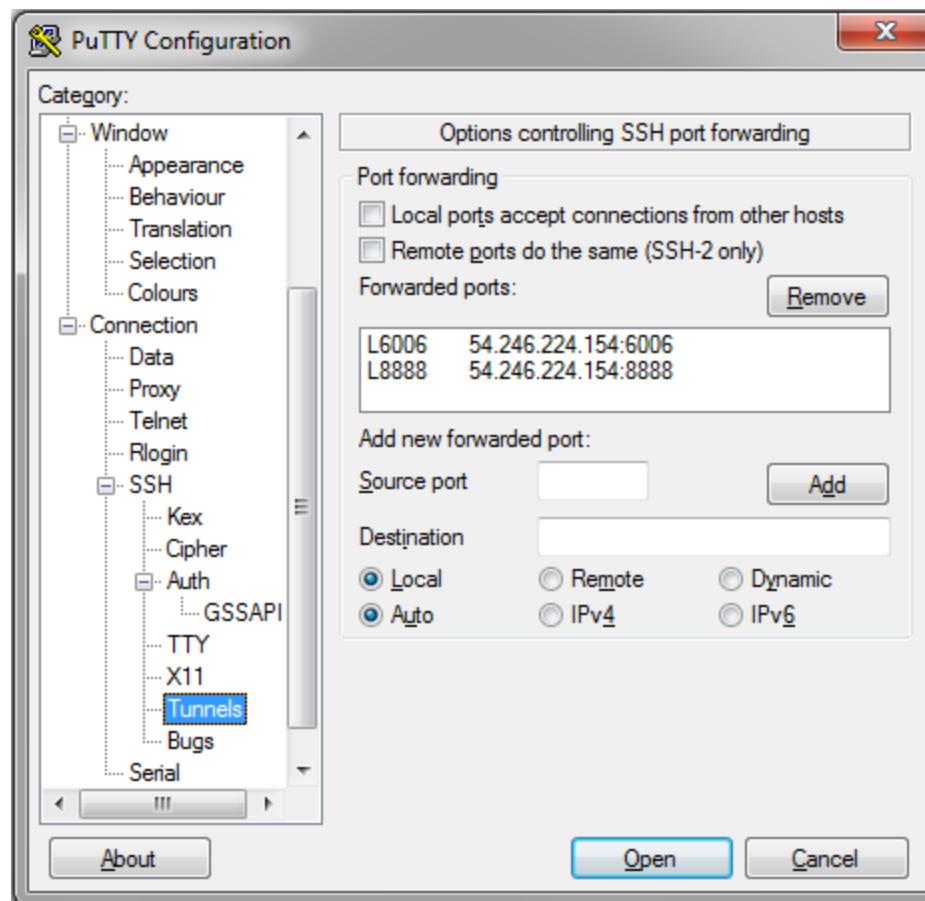
[Save to List](#)

[Overview](#) [Pricing](#) [Usage](#) [Support](#) [Reviews](#)

Apache MXNet, Caffe2, TensorFlow, PyTorch, Keras, CNTK, Theano

SSH Port Forwarding

Jupyter Notebook in
local web browser



Amazon EC2 instance
with NVIDIA Tesla GPU

Build a neural network model with two hidden layers using the Keras API.

In [47]:

```
1 import keras
2 from keras.models import Sequential
3 from keras.layers import Dense
4
5 keras.backend.clear_session()
6 n_hidden = 32
7
8 model = Sequential()
9 model.add(Dense(input_dim=1, units=n_hidden, activation='relu'))
10 model.add(Dense(units=n_hidden, activation='relu'))
11 model.add(Dense(units=1))
12 model.summary()
13 model.compile(loss='mean_squared_error', optimizer='sgd', metrics=['accuracy'])
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 32)	64
dense_2 (Dense)	(None, 32)	1056
dense_3 (Dense)	(None, 1)	33

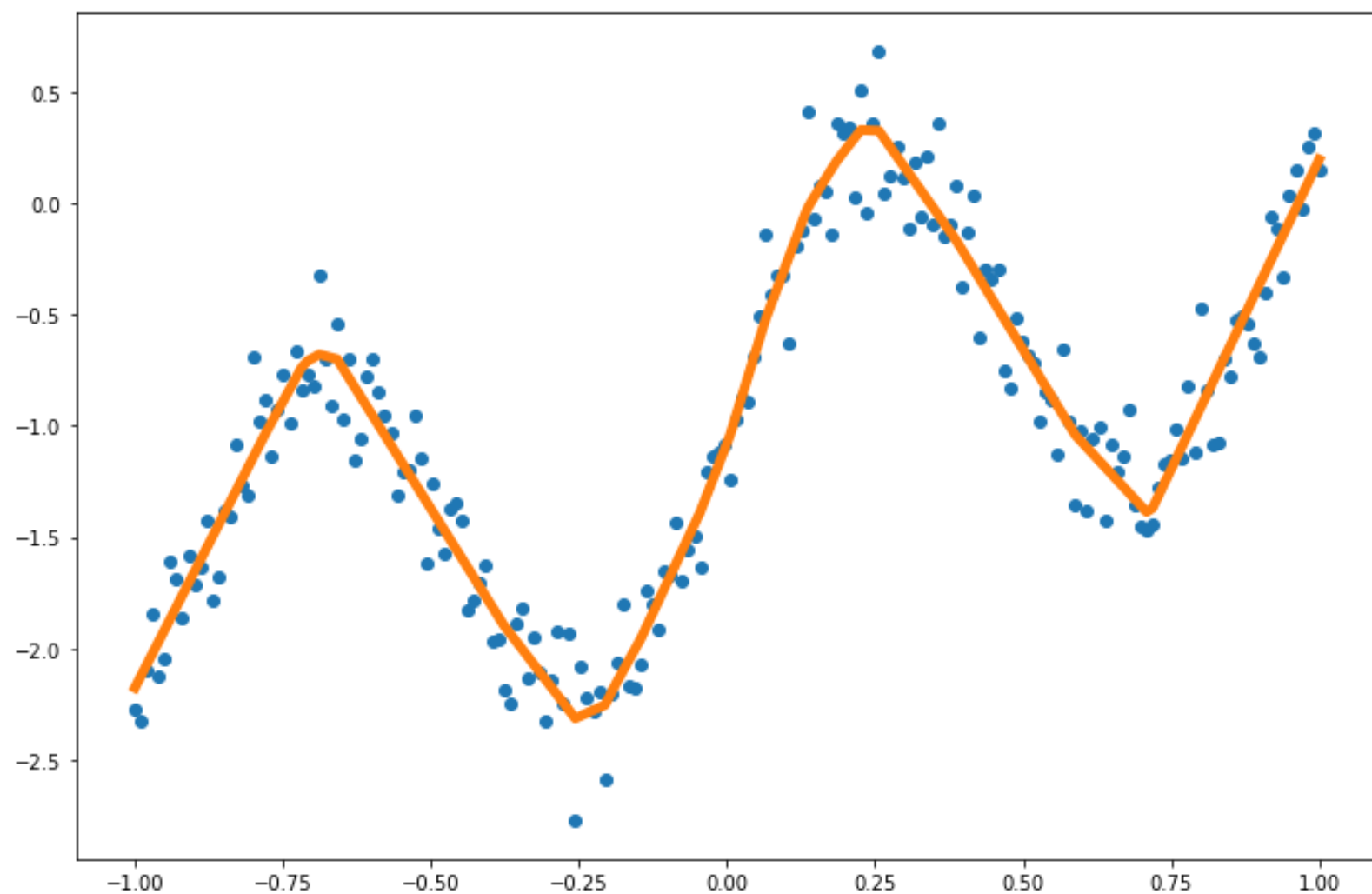
=====
Total params: 1,153
Trainable params: 1,153
Non-trainable params: 0
=====

In [44]:

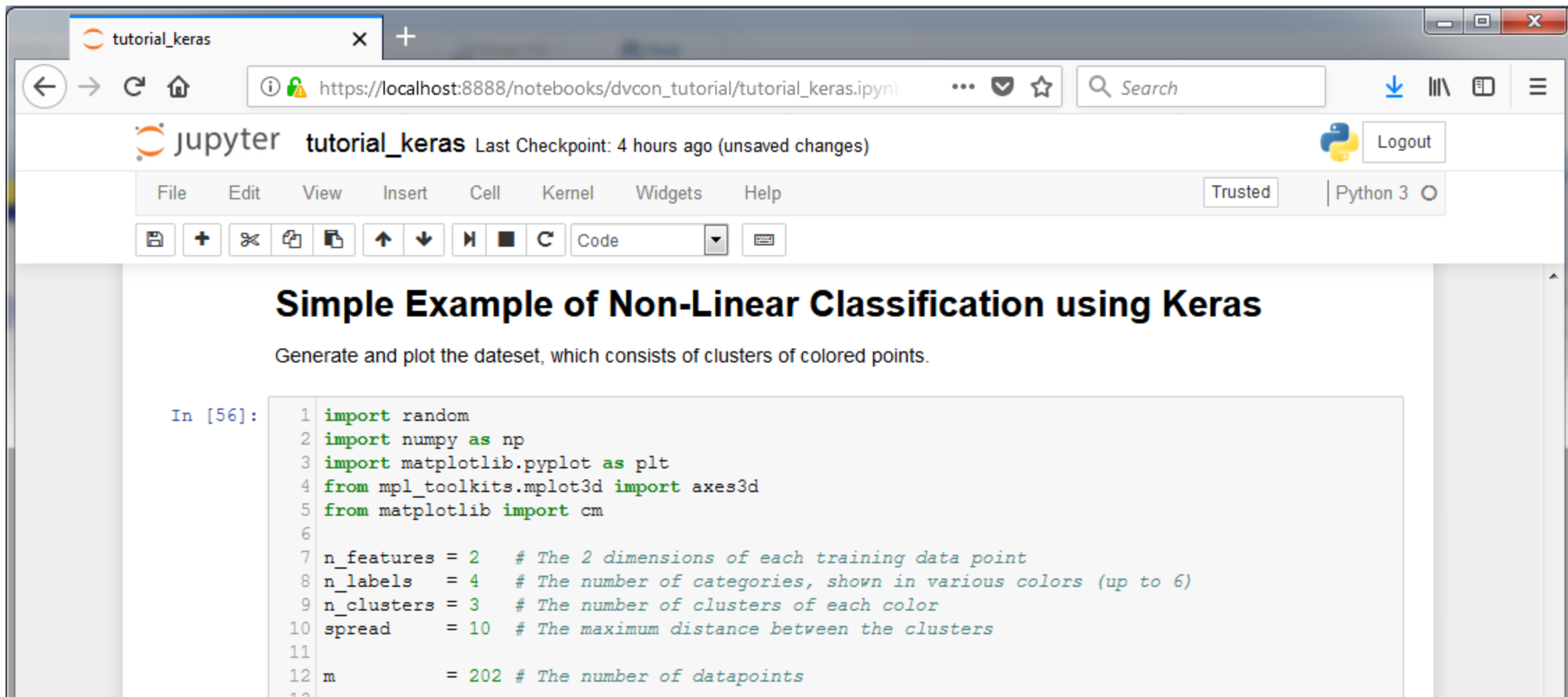
```
1 model.fit(train_x, train_y, epochs=10000, batch_size=m, verbose=0)
```

Plot the predicted curve learnt by the model.

```
In [42]: 1 y = model.predict(train_x)
          2 plt.plot(train_x, train_y, 'o')
          3 plt.plot(train_x, y, linewidth=5)
          4 plt.show()
```

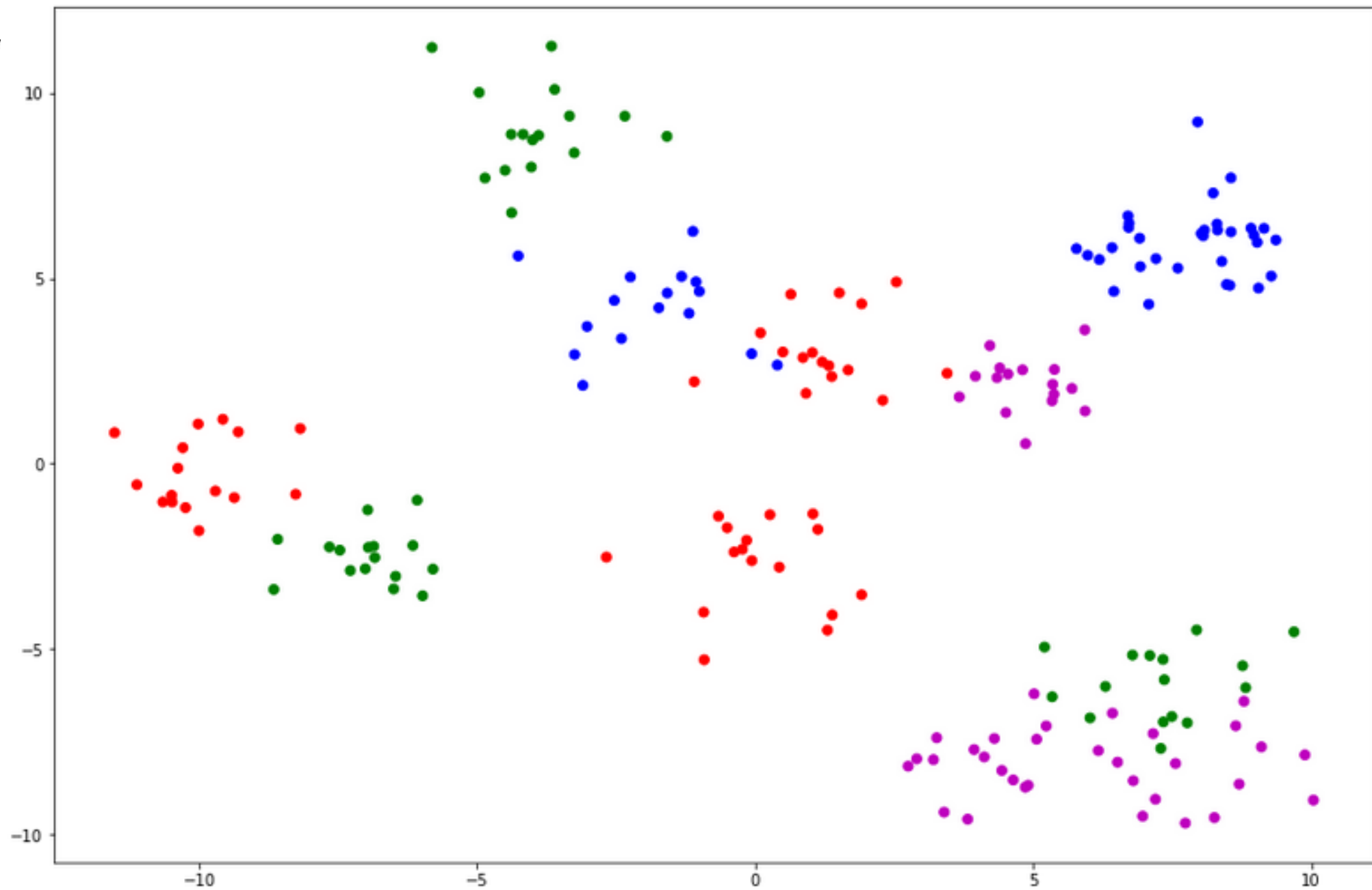


Classification



The screenshot shows a Jupyter Notebook window titled 'tutorial_keras'. The browser address bar indicates the URL is `https://localhost:8888/notebooks/dvcon_tutorial/tutorial_keras.ipynb`. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, and running code. The notebook content displays the title 'Simple Example of Non-Linear Classification using Keras' followed by a description: 'Generate and plot the dataset, which consists of clusters of colored points.' Below this, a code cell labeled 'In [56]:' contains the following Python code:

```
1 import random
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from mpl_toolkits.mplot3d import axes3d
5 from matplotlib import cm
6
7 n_features = 2 # The 2 dimensions of each training data point
8 n_labels = 4 # The number of categories, shown in various colors (up to 6)
9 n_clusters = 3 # The number of clusters of each color
10 spread = 10 # The maximum distance between the clusters
11
12 m = 202 # The number of datapoints
13
```



In [98]:

```
1 import keras
2 from keras.models import Sequential
3 from keras.layers import Dense
4 from keras.optimizers import sgd
5
6 keras.backend.clear_session()
7 n_hidden = 16
8
9 model = Sequential()
10 model.add(Dense(input_dim=n_features, units=n_hidden, activation='relu'))
11 model.add(Dense(units=n_hidden, activation='relu'))
12 model.add(Dense(units=n_hidden, activation='relu'))
13 model.add(Dense(units=n_labels, activation='softmax'))
14 model.summary()
15 model.compile(loss='categorical_crossentropy', optimizer=sgd(lr=0.05), metrics=['accuracy'])
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 16)	48
dense_2 (Dense)	(None, 16)	272
dense_3 (Dense)	(None, 16)	272
dense_4 (Dense)	(None, 4)	68
Total params: 660		

Run and Evaluate the Model

Run gradient descent.

```
In [101]: 1 model.fit(train_x, train_y, epochs=5000, batch_size=m, shuffle=False, verbose=0)
```

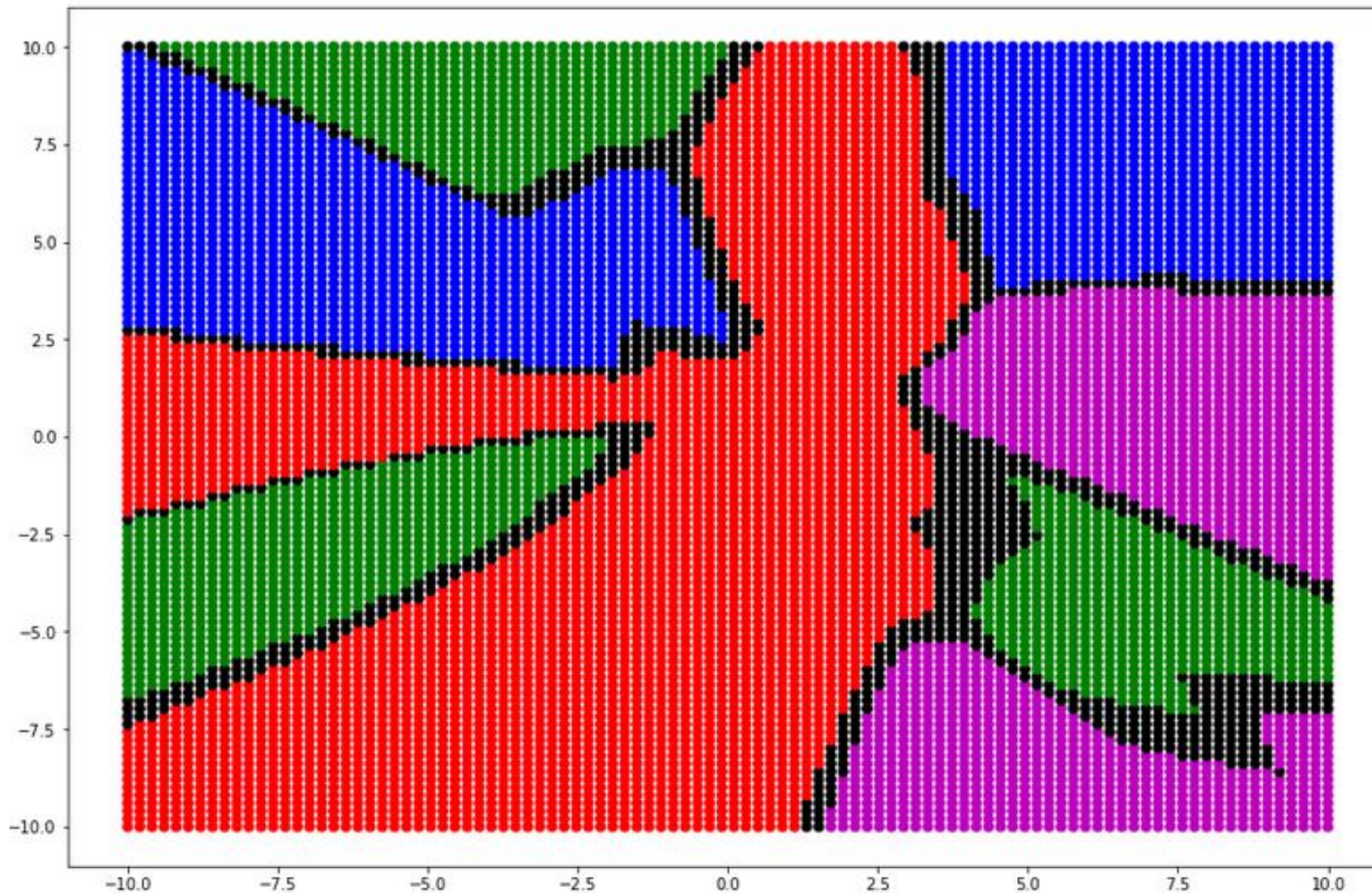
```
Out[101]: <keras.callbacks.History at 0x7f416fe2d390>
```

Evaluate the trained model on the training data.

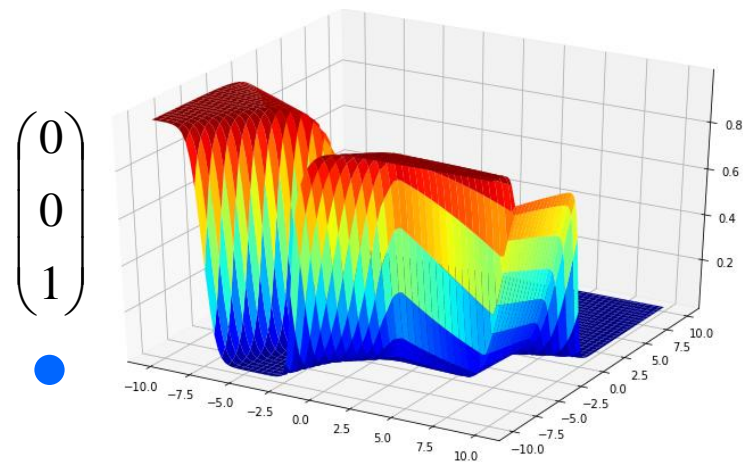
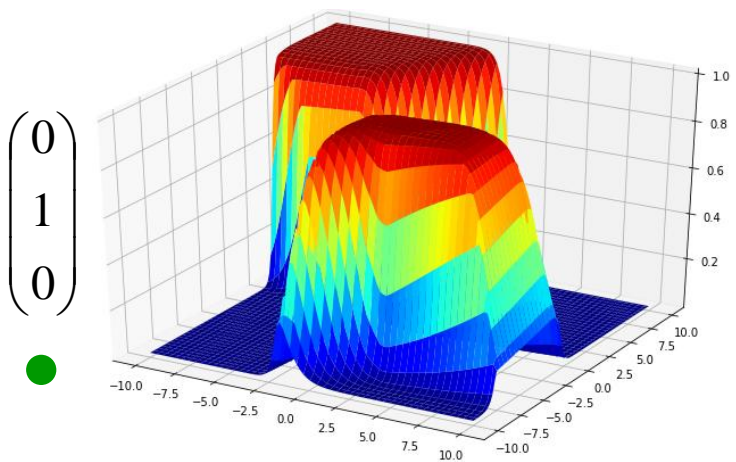
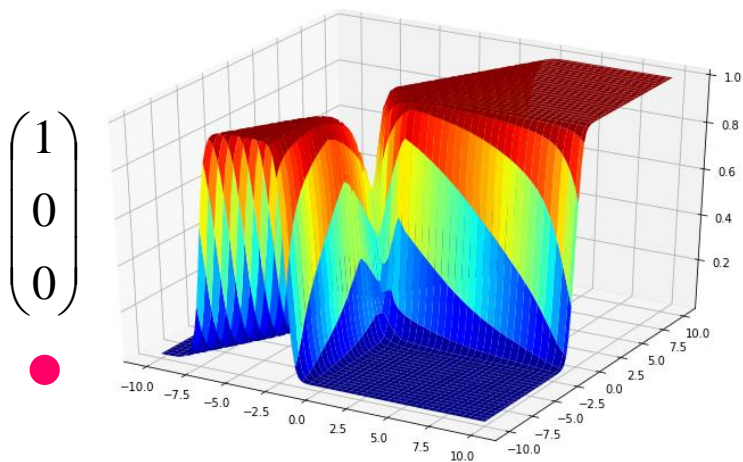
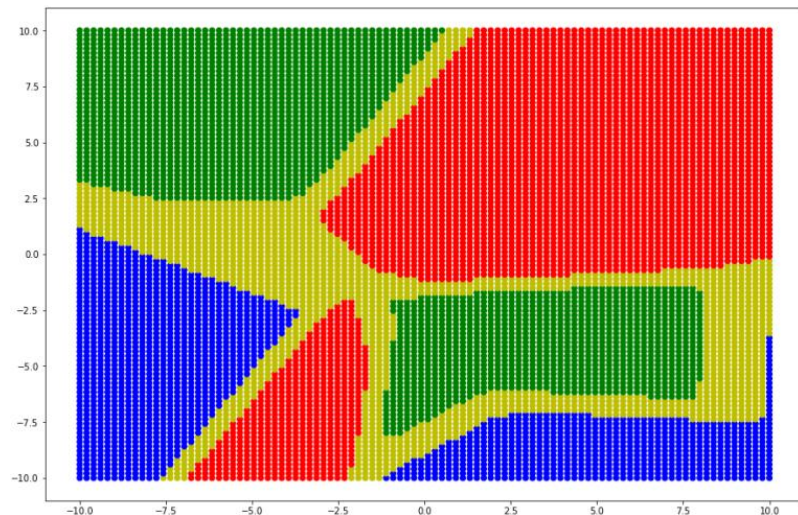
```
In [102]: 1 loss_and_acc = model.evaluate(train_x, train_y, batch_size=m, verbose=0)
          2 print('Accuracy = {:.4.2f}'.format(loss_and_acc[1]))
```

```
Accuracy = 0.98
```

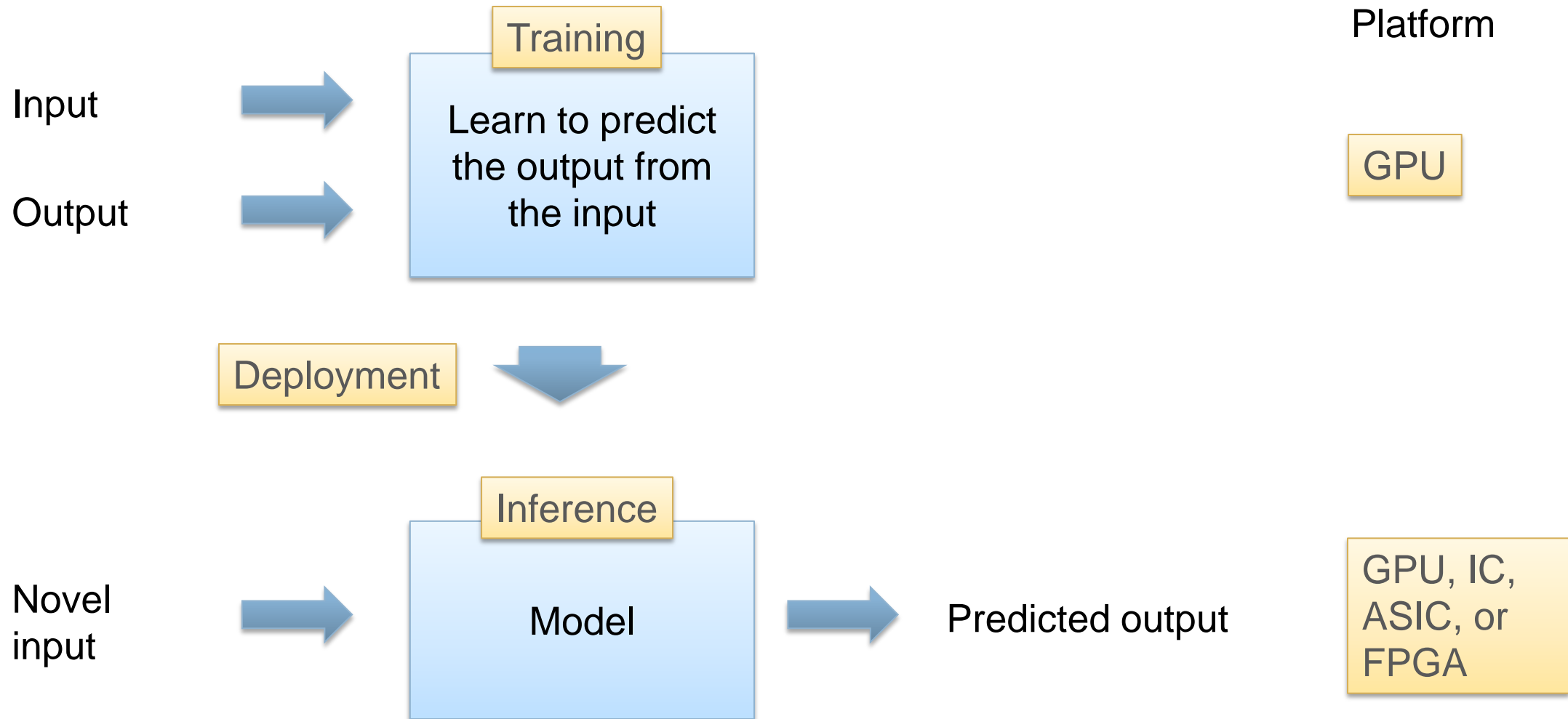
The Decision Boundary



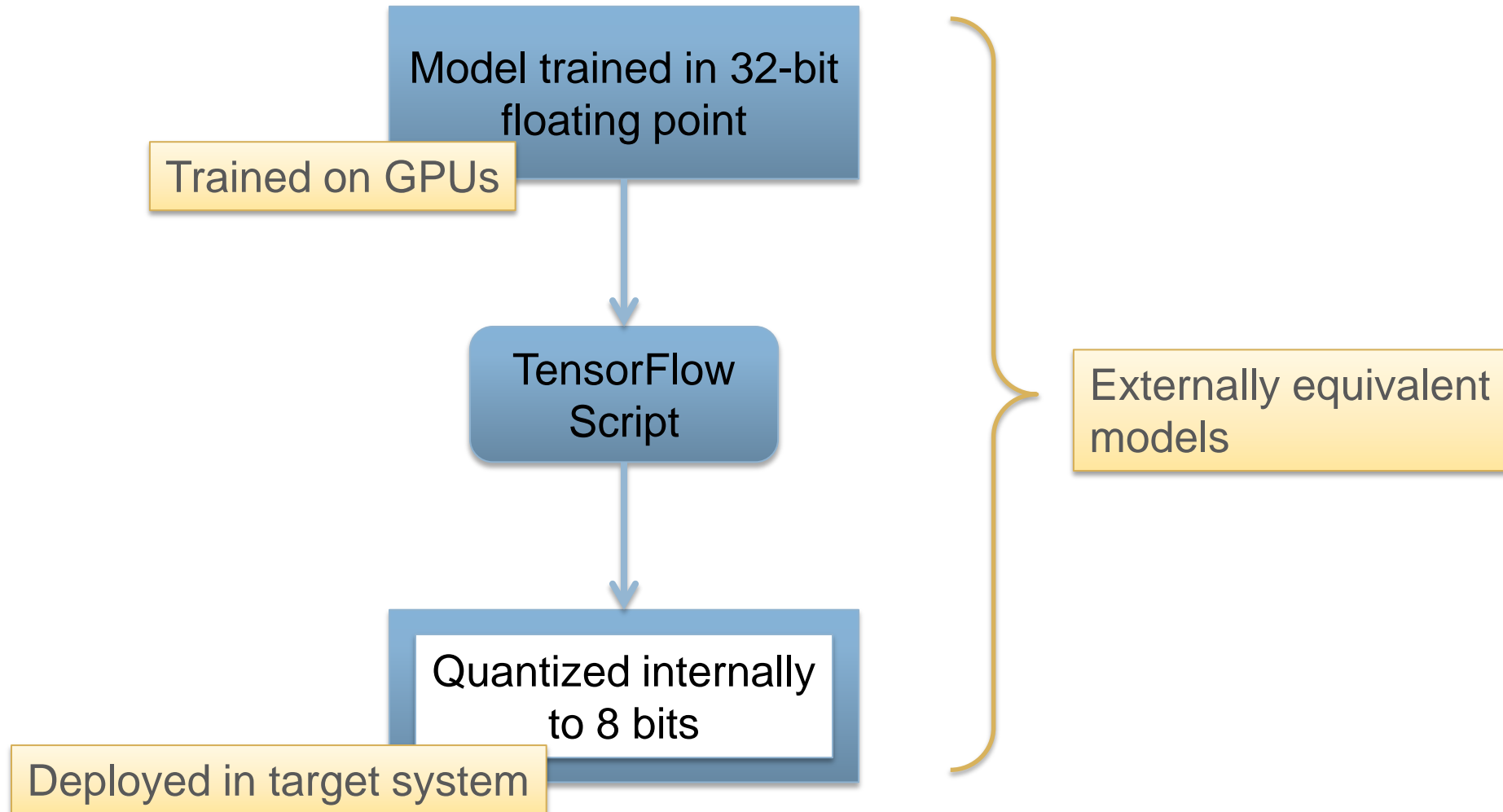
Softmax Probability for each Label



Training versus Inference



Weight Quantization



For More Information

Example Jupyter Notebook:

www.doulos.com/downloads/dvcon_dl.ipynb

john.aynsley@doulos.com