A Gentle Introduction to Neural Networks

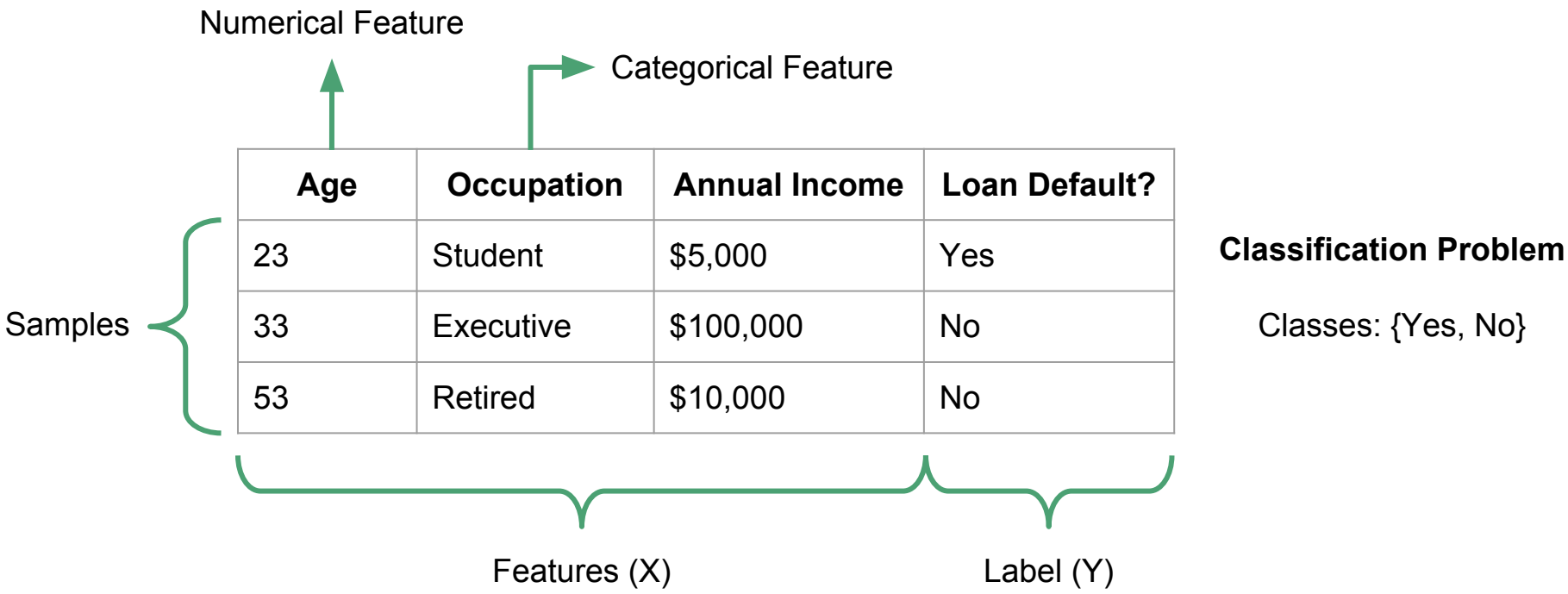
Suyash Lakhotia
NTUOSS TGIFHacks #79

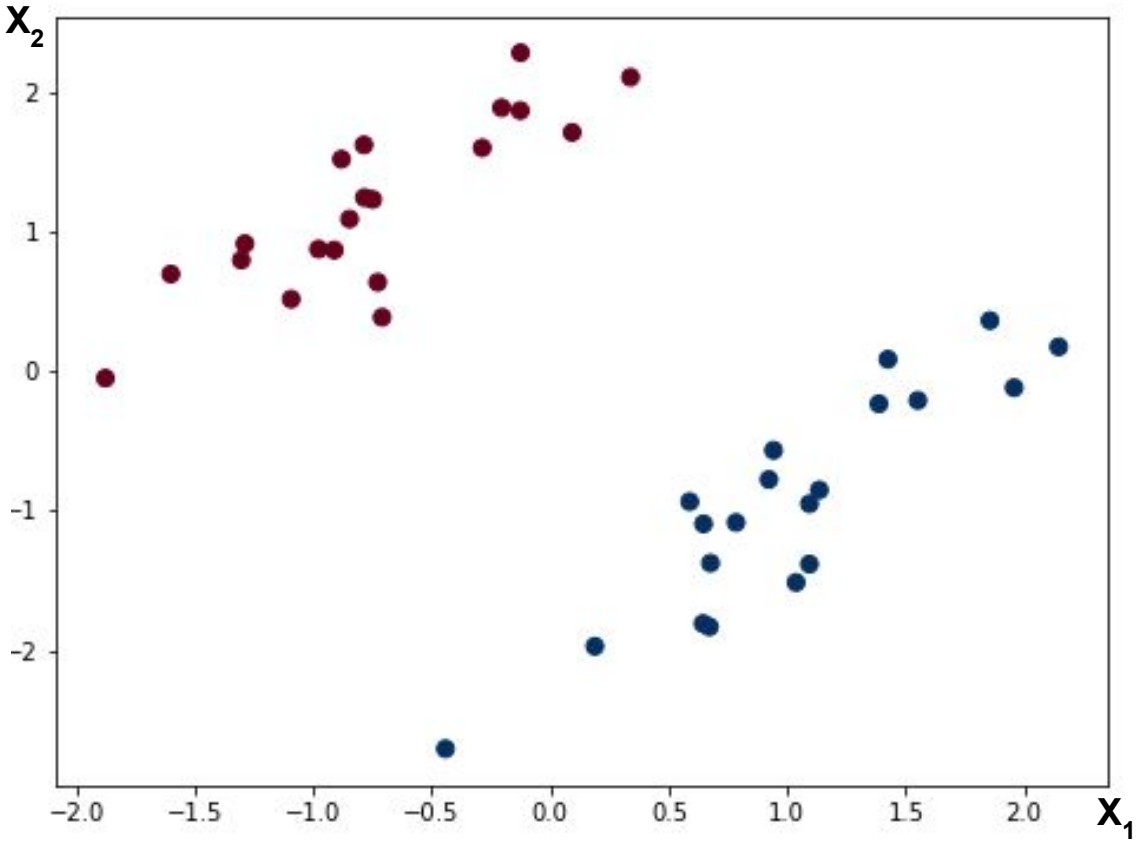
Machine Learning

What is machine learning?

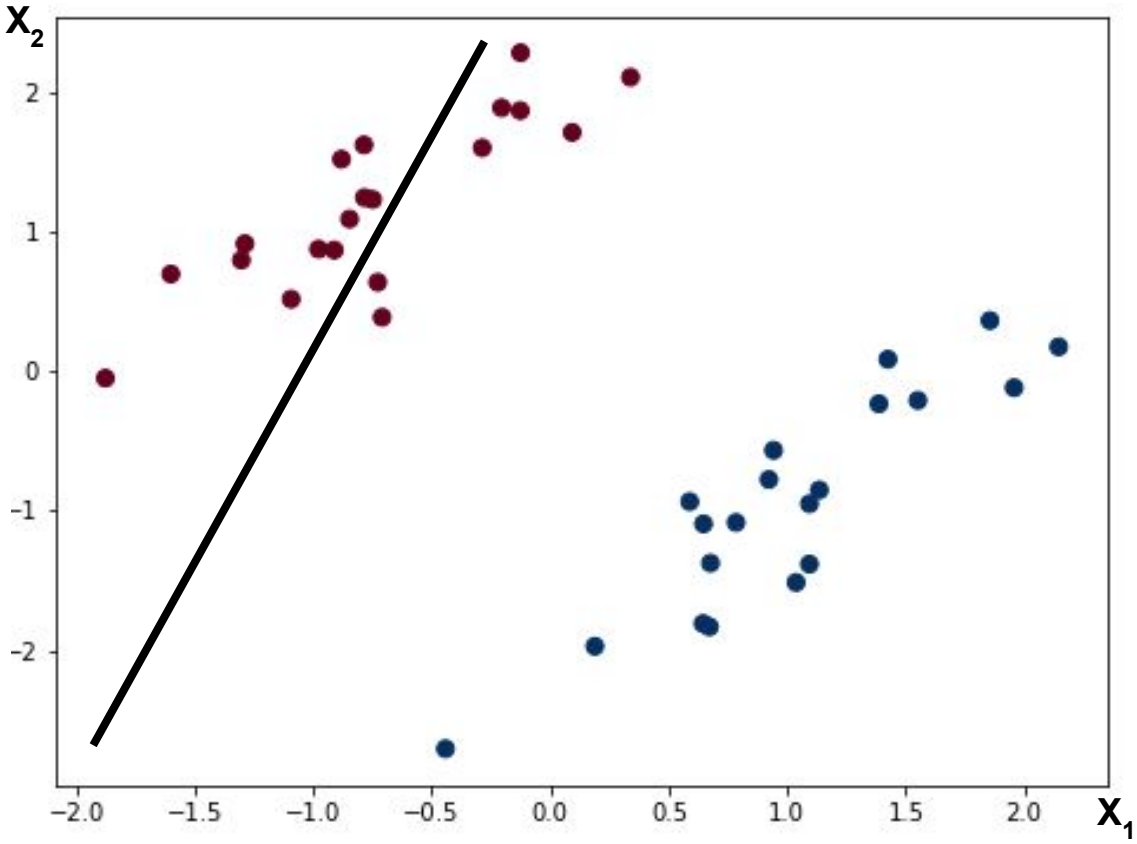
- Algorithms that use **large real-world datasets** to **understand patterns and relationships in data** in order to **predict** the outcome of **unseen data**.
- Computers are **not** provided with **hard-coded rules** to make decisions but rather ***learn* the "rules" of the data** by repeatedly looking through the dataset.
- Learning algorithms produce **machine learning models** by **training** on a (large) **training dataset** and are tested on an **independent test dataset**.
- Broadly split into two categories — **supervised** and **unsupervised** learning.
- In supervised learning, the model is given the **input features** as well as the **corresponding labels**. In unsupervised learning, the model is only provided with the **input features** and learns **logical clusterings / classifications**.

Some Basic Definitions

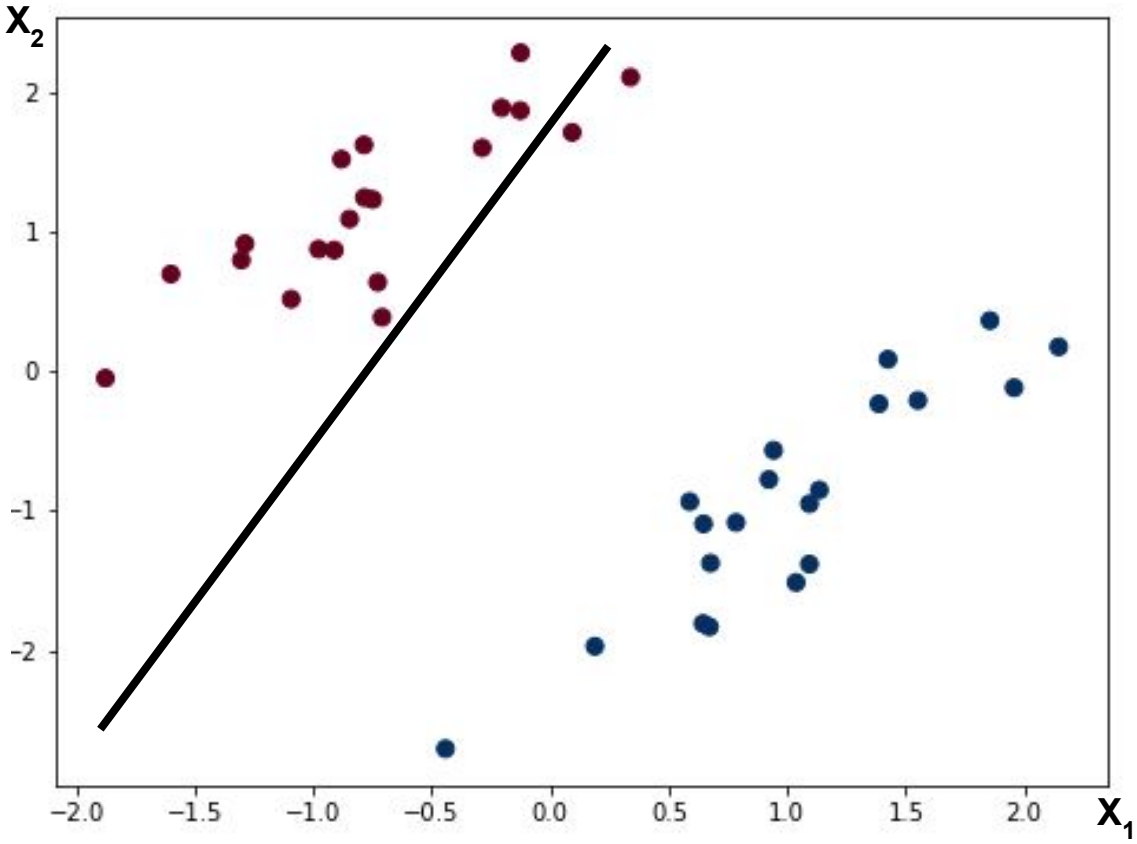




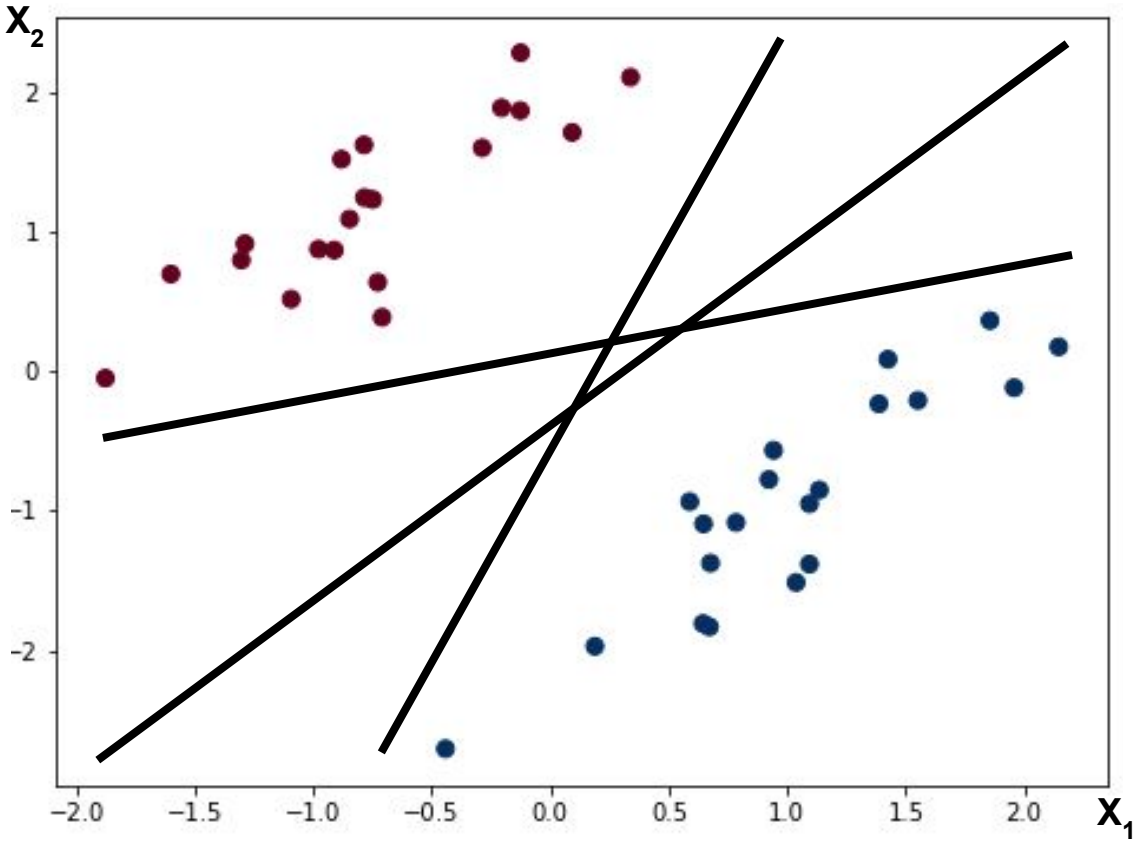
Two-Class Classification Problem: Dataset



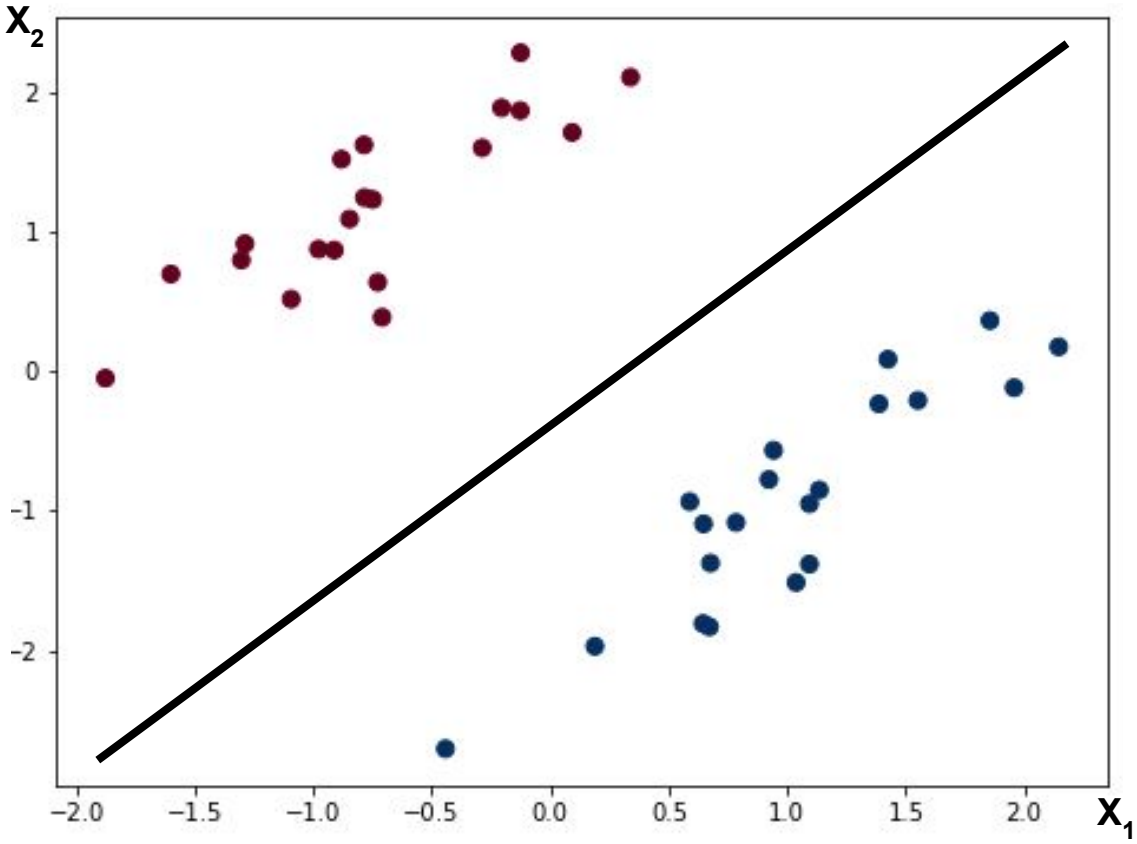
Two-Class Classification Problem: Training Iteration A



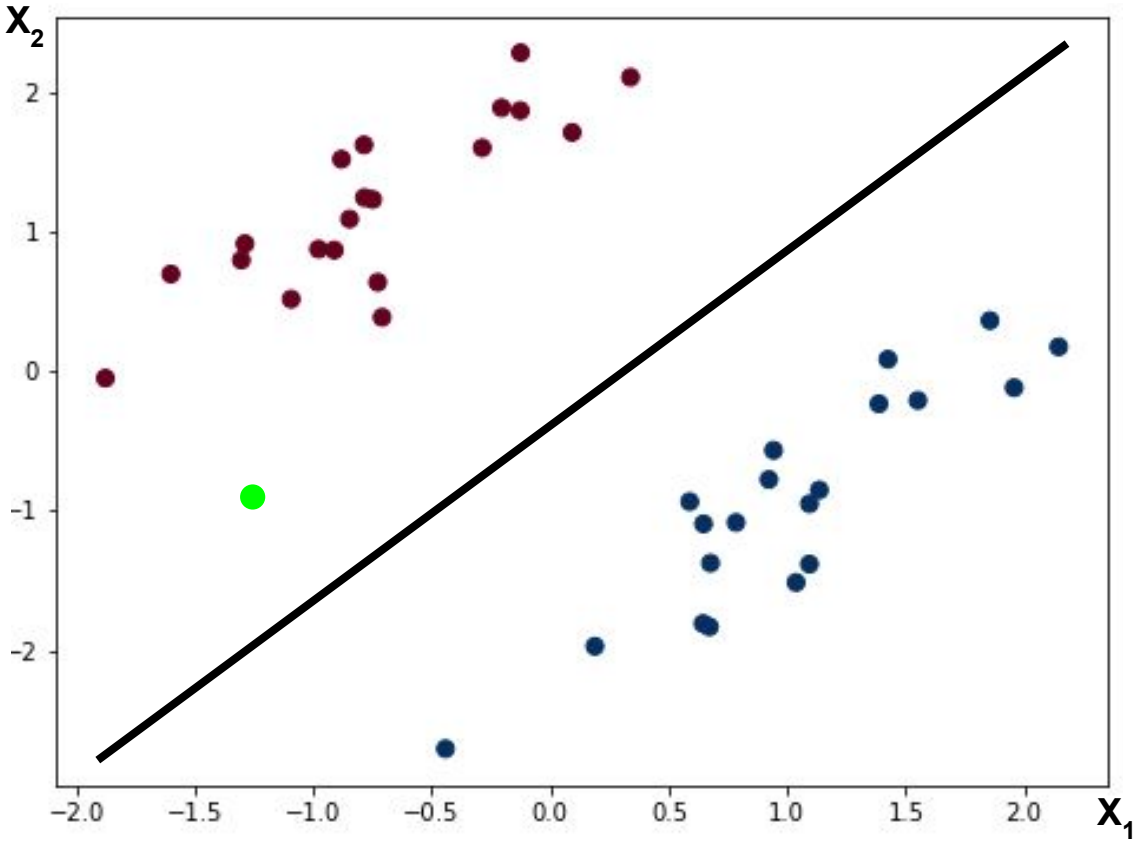
Two-Class Classification Problem: Training Iteration A + 1



Two-Class Classification Problem: Generalisation



Two-Class Classification Problem: Final Model

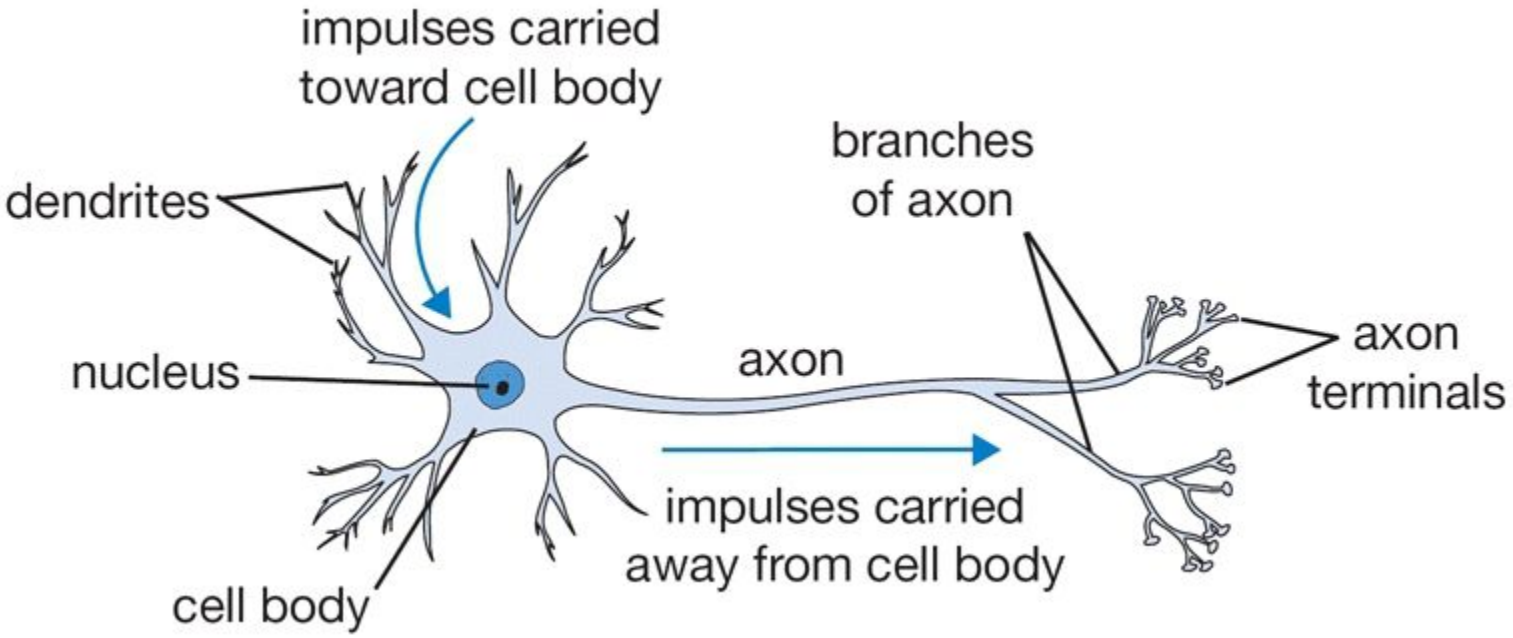


Two-Class Classification Problem - Inference / Testing

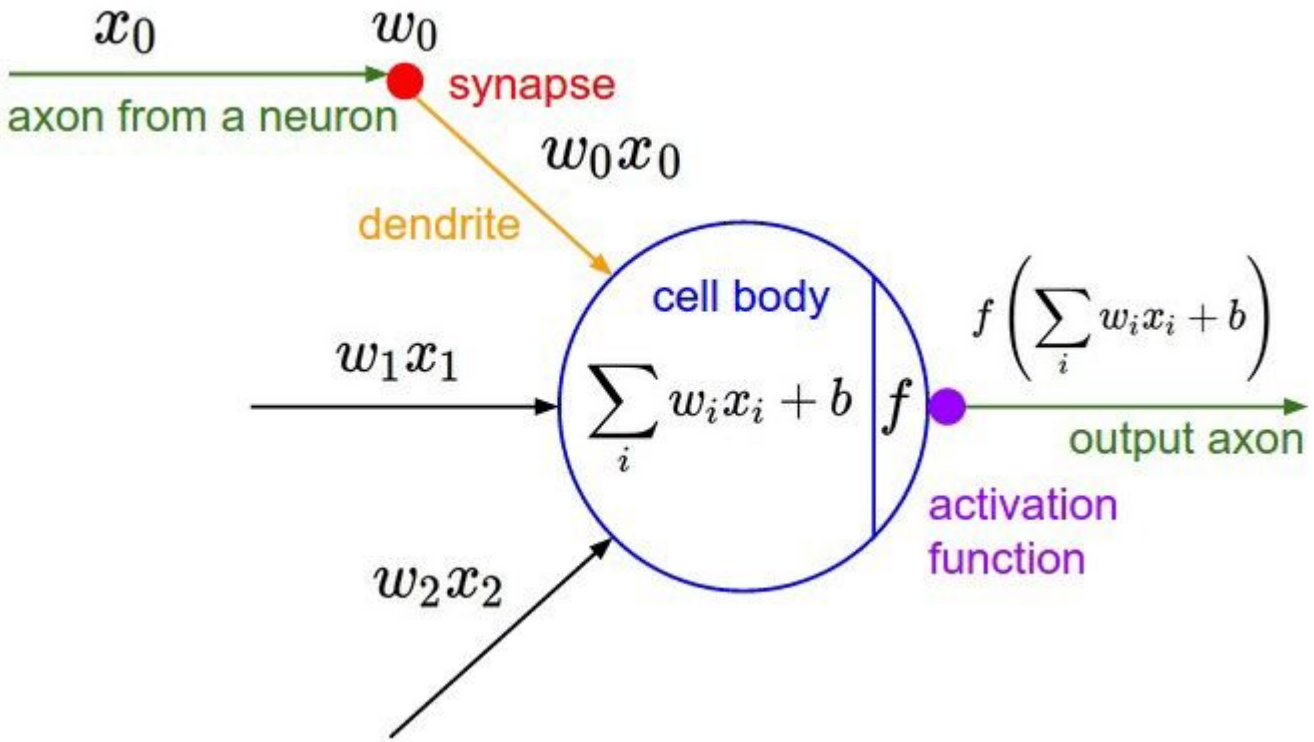
High-Level Intuition

- Educated trial-and-error with a well-defined **loss function** (think: error metric), which allows the learning algorithm to understand whether it is getting closer or further away from the optimum model.
- The model is usually trained over **several epochs** (i.e. runs through the training dataset), where the order of samples may be shuffled at every epoch.
- Large number of **training samples** allows the model to learn a more **general** solution that is unbiased and robust to outliers.
- An **independent test set** is required to benchmark the performance of the model since the model has already *seen* the data it has trained on. Imagine if all our exam questions were questions from tutorials.

(Artificial) Neural Networks



Biological Motivations: Why “artificial” neural networks?



Translating Biology to Math: A basic (artificial) neuron.

$$u = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$u = \boldsymbol{x}^T \boldsymbol{w} + b$$

$$f(u) = u$$

$$\boldsymbol{x}^T \boldsymbol{w} + b = 0 \longleftarrow \text{Linear Hyperplane}$$

Linear Neuron

$$u = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$u = \boldsymbol{x}^T \boldsymbol{w} + b$$

$$f(u) = f(u)$$

Perceptron

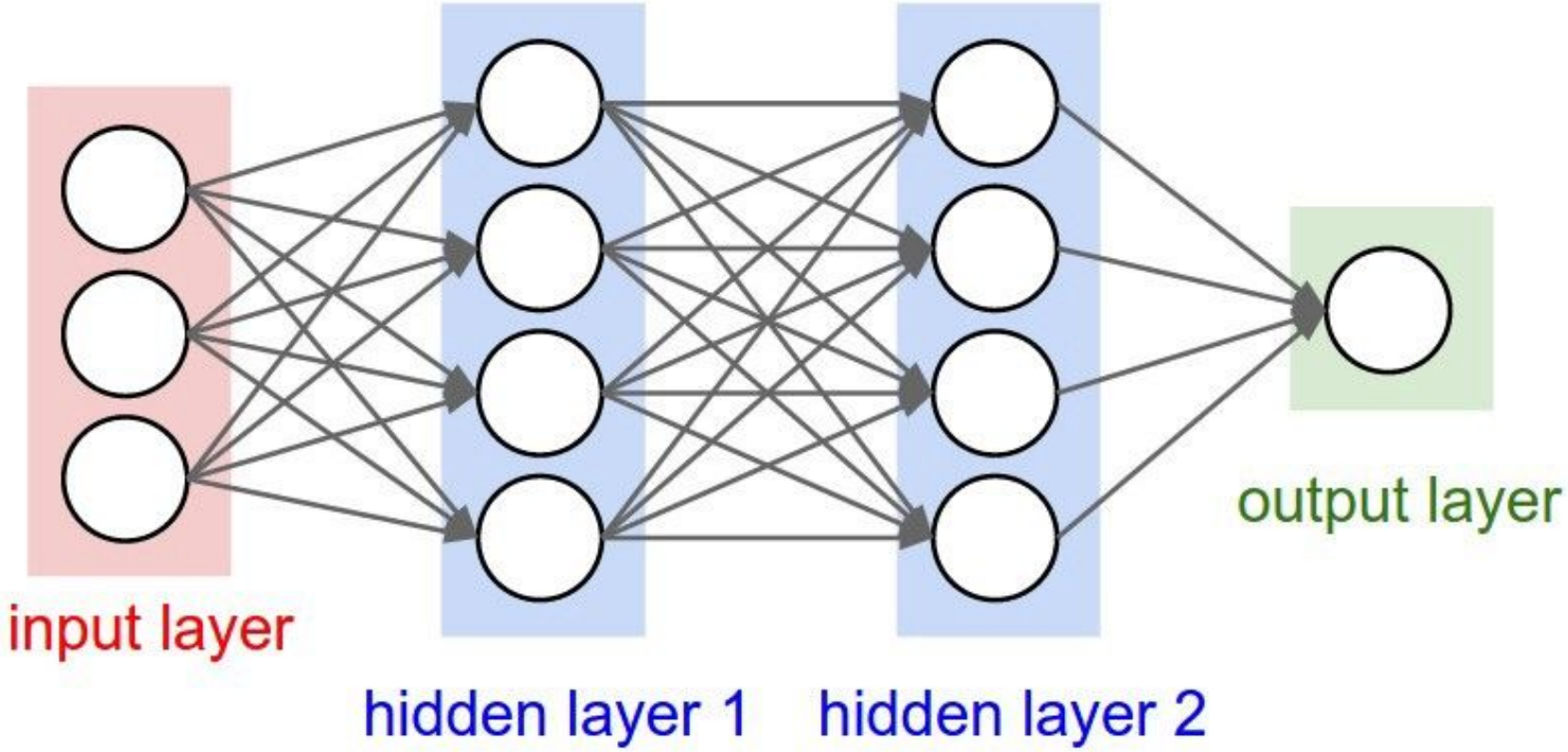
$$f(u) = \frac{a}{1 + e^{-bu}}$$

$$f(u) = \max\{0, u\}$$

$$f(u_k) = \frac{e^{(u_k)}}{\sum_{k=1}^{K} e^{(u_k)}}$$

Activation Functions: Sigmoid, ReLU & Softmax

Neuron Layers

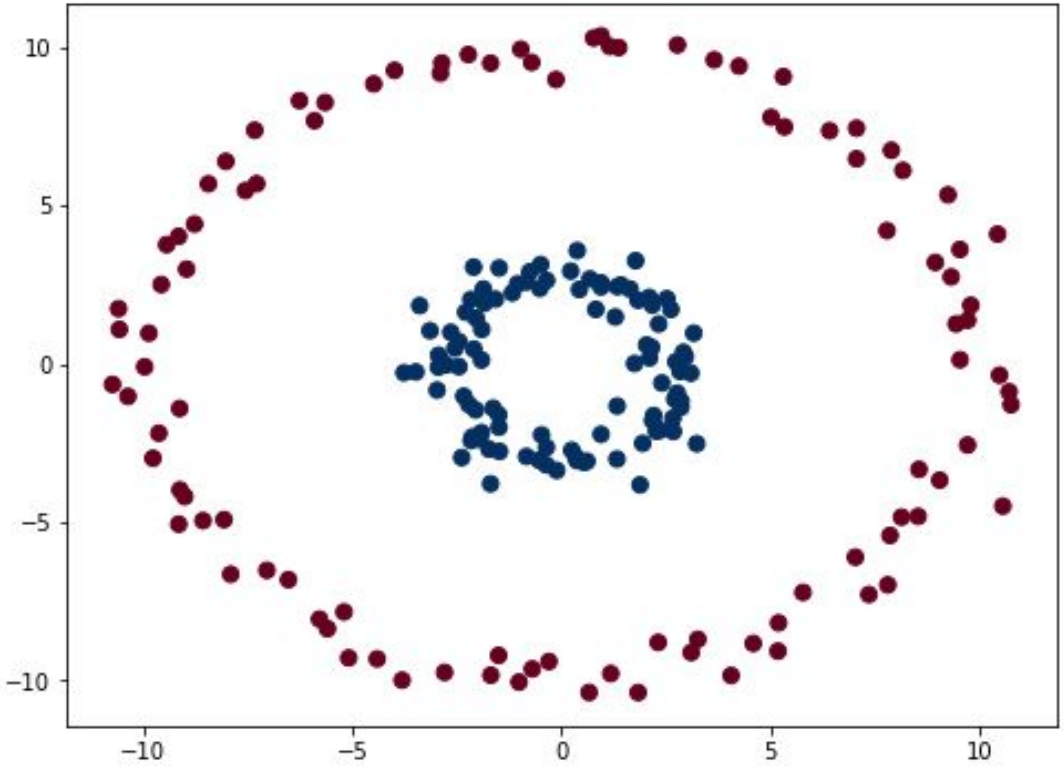


Multi-Layer Perceptron

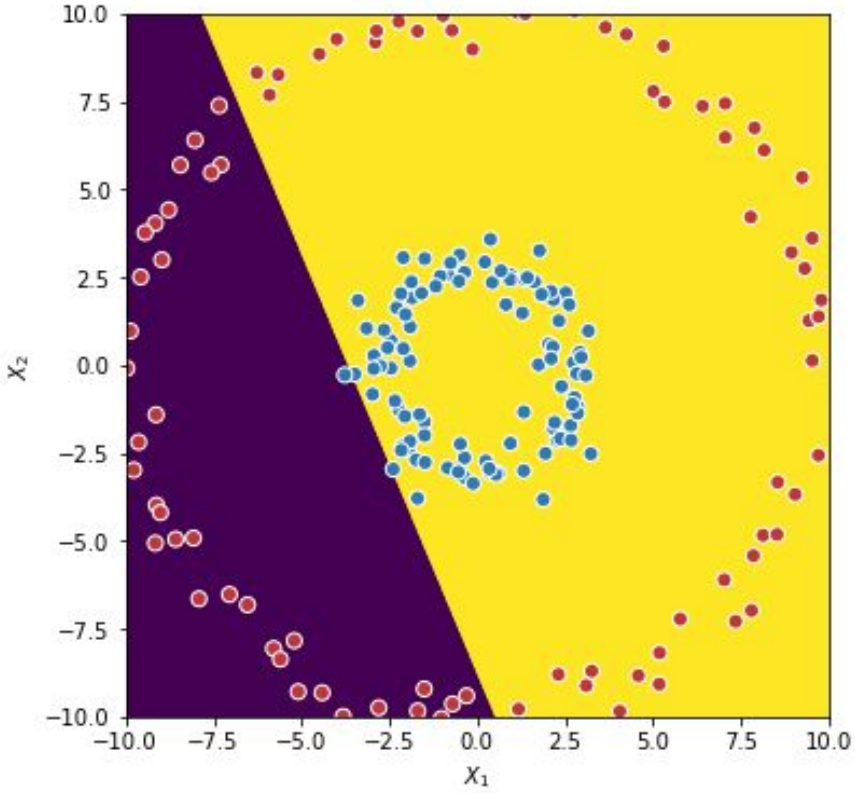
Neuron Layers

- **Input Layer:** Receives the input features and passes them to the first hidden layer (or directly to the output layer). Number of neurons is equal to the number of input features.
- **Hidden Layer(s):** There can be >= 0 hidden layers composed of varying number of neurons between the input layer and the output layer, which increase the learning capacity of the network.
- **Output Layer**: This layer generates the final output of the model. In the case of regression problems, there is one output neuron and in the case of classification, there are as many neurons as number of classes.

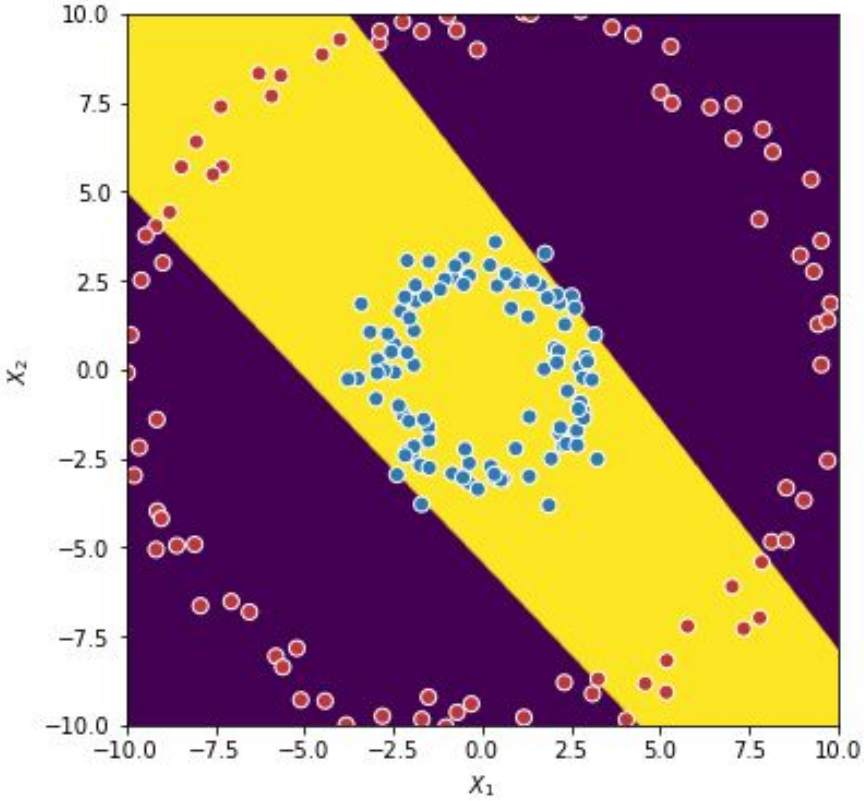
Importance of Non-Linearity



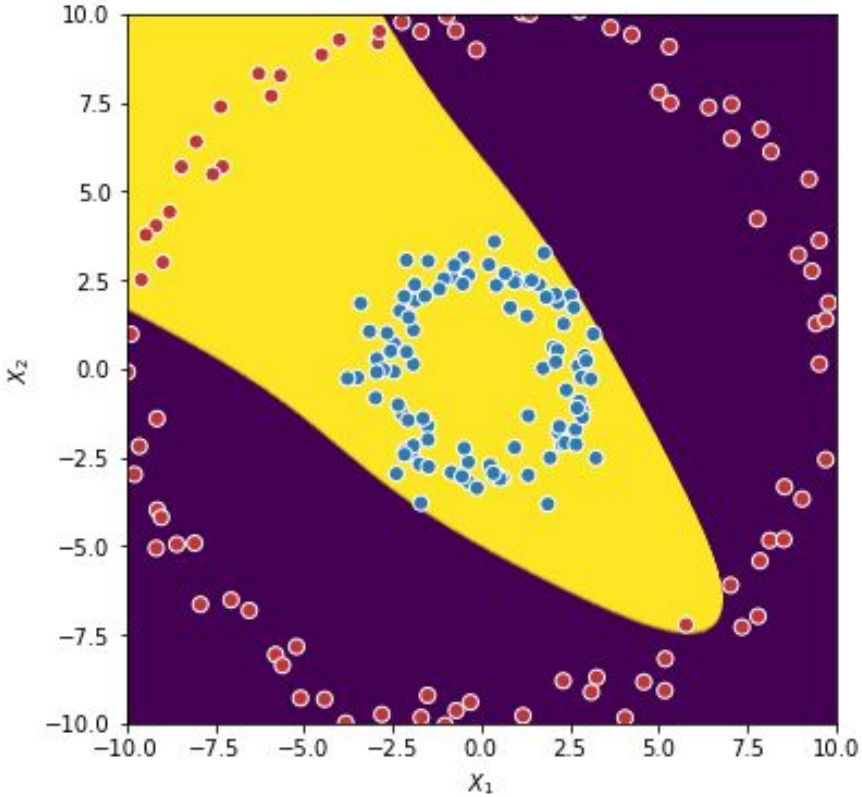
Non-Linear Classification Problem



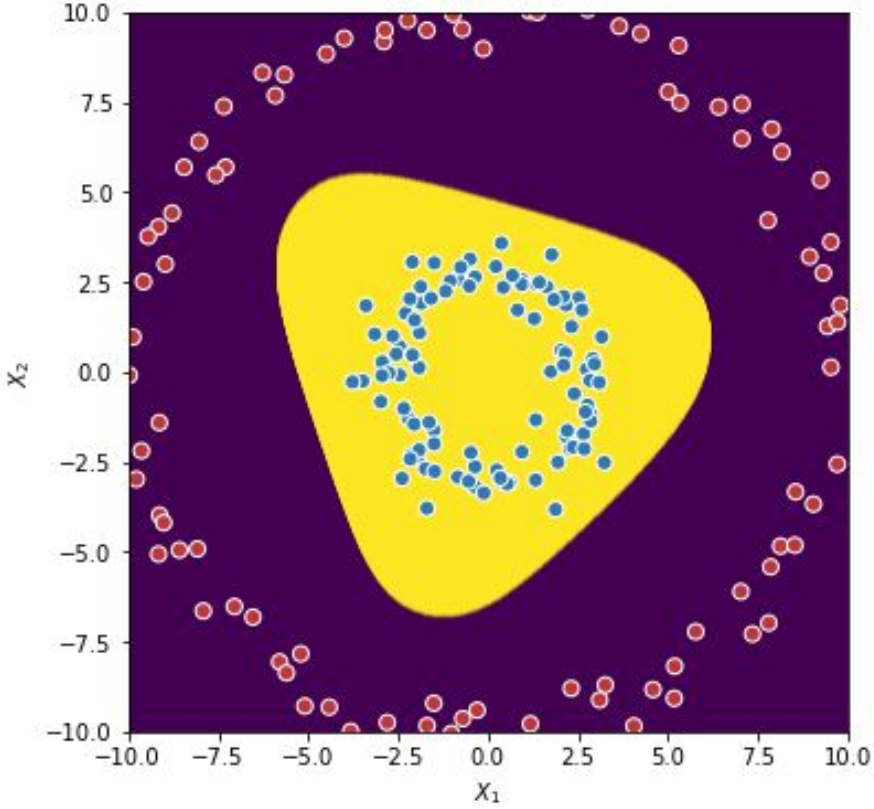
1 Neuron (Hidden Layer, ReLU Activation)



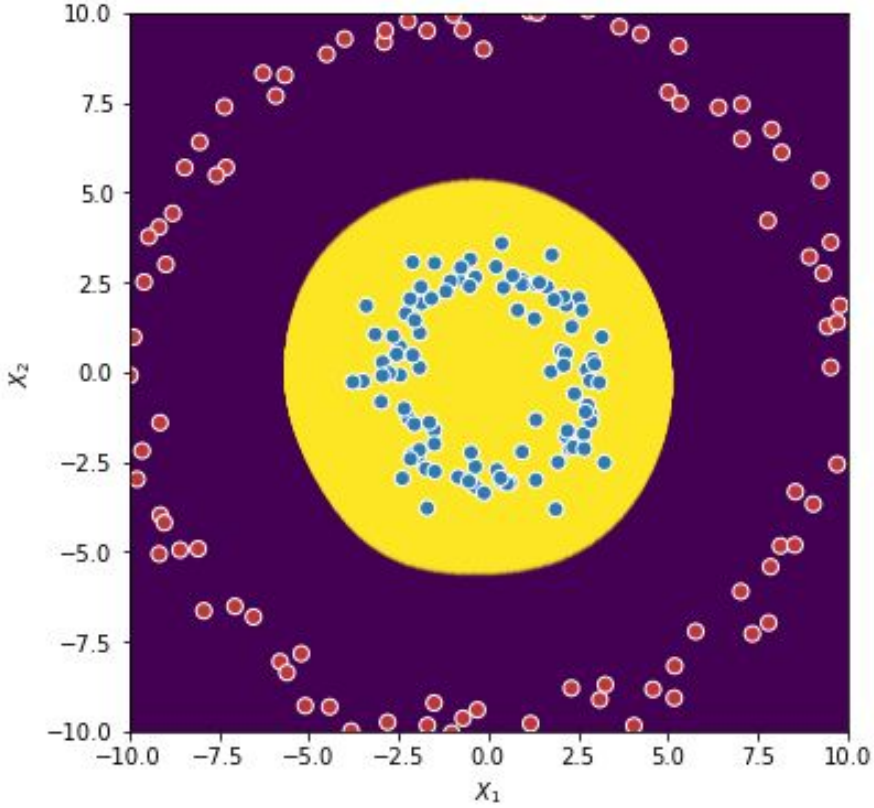
2 Neurons



3 Neurons



6 Neurons



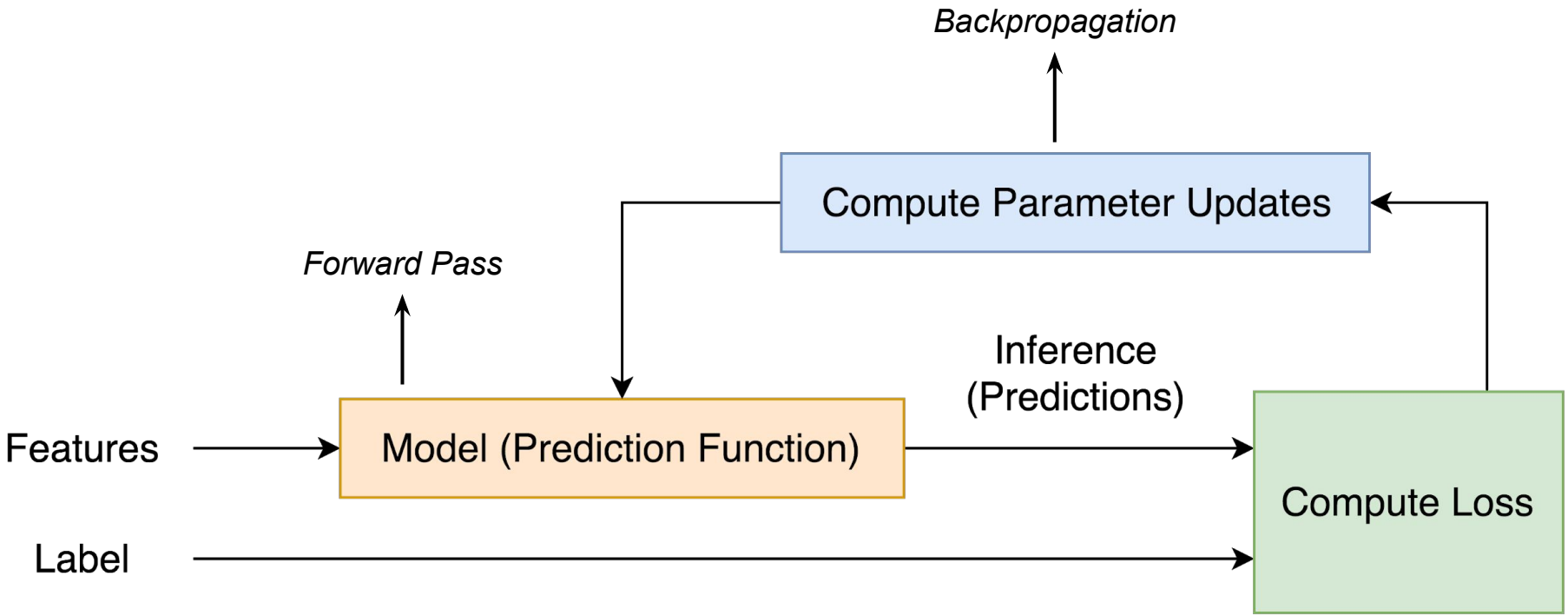
30 Neurons

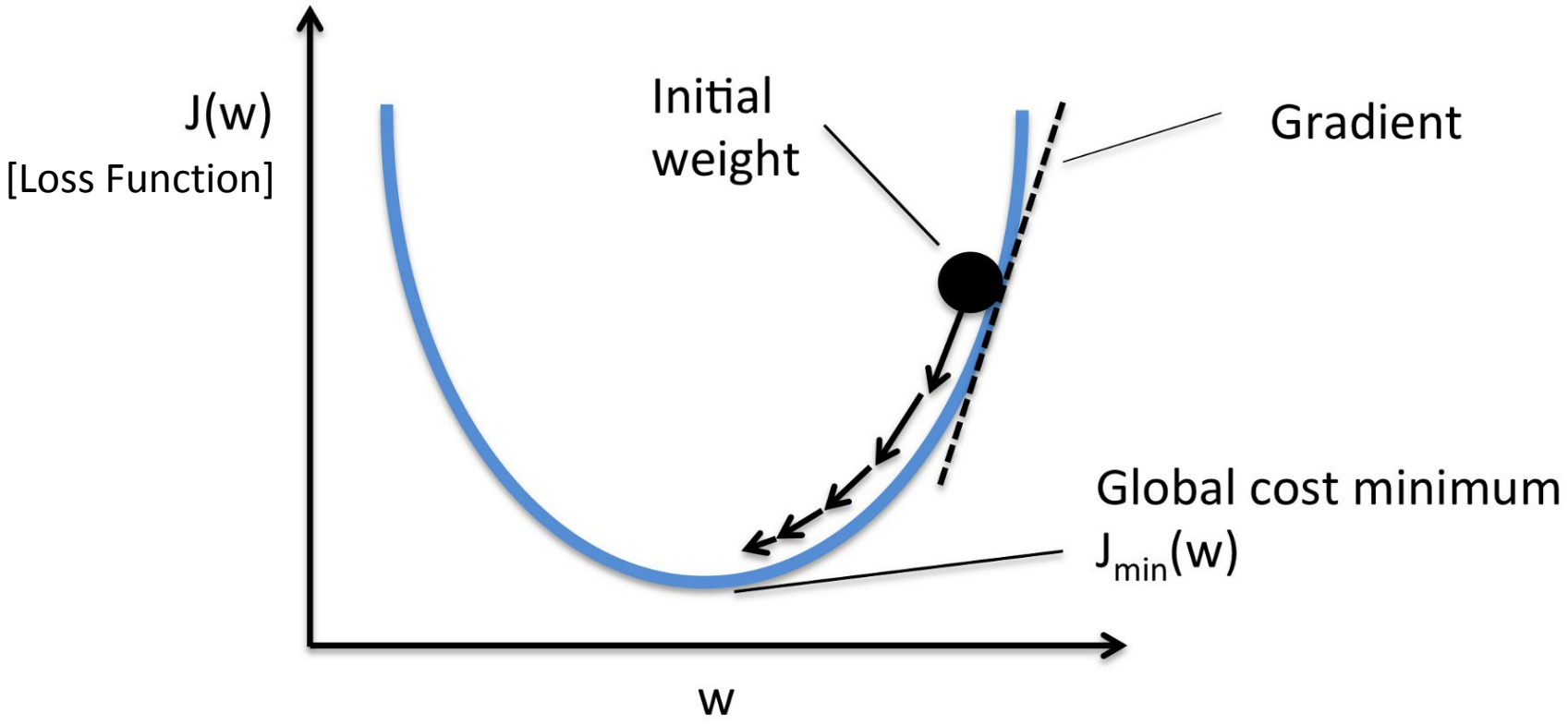
TensorFlow Playground:
http://playground.tensorflow.org/

Learning: Gradient Descent

Review

- Neuron inputs are transformed to the neuron's output by multiplying the respective weights and adding a bias term (i.e. forming the synaptic input) and passing it through a (non-linear) activation function.
- *Learning* is an iterative trial-and-error process based on a well-defined loss function / cost function (think: error metric).
- Weights and biases (i.e. model parameters) are randomly initialized, usually from a (modified) uniform / normal distribution.
- The goal is to adjust the weights and biases during training such that they achieve a low error rate (i.e. low loss / high accuracy).


Backpropagation
Compute Parameter Updates
Forward Pass
Inference
(Predictions)
Features
Model (Prediction Function)
Compute Loss
Label



Gradient Descent Intuition

Implementation

Implementation Details

- **Language:** Python 3
- **Libraries:**
  - Keras (Neural Network, high-level wrapper for TensorFlow)
  - scikit-learn (Dataset, Preprocessing)
- **Dataset:** UCI ML Wine Dataset
  - Toy multi-class classification dataset released by UCI.
  - 178 Samples, 13 Features, 3 Classes
  - https://archive.ics.uci.edu/ml/machine-learning-databases/wine/
- **Environment:** Colaboratory (Online Research Tool by Google)
  - https://colab.research.google.com/notebook

Loading the Dataset

```
from sklearn.datasets import load_wine

# Load wine dataset
dataset = load_wine()

# Print description of dataset
print(dataset.DESCR)
```

Splitting the Dataset

```
from sklearn.model_selection import train_test_split

# Shuffle and split the dataset
X_train, X_test, y_train, y_test = train_test_split(dataset.data,
                                                    dataset.target,
                                                    test_size=0.30,
                                                    random_state=42)
```

Shuffle and split the dataset into a train set (70%) and test set (30%)

Preparing the Dataset

```
from sklearn.preprocessing import StandardScaler, LabelBinarizer

# Scale the features of the dataset
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# One-hot encode the labels
binarizer = LabelBinarizer().fit(y_train)
y_train = binarizer.transform(y_train)
y_test = binarizer.transform(y_test)
```

Defining the Neural Network

```
from keras.models import Sequential
from keras.layers import Dense

# Initialize the constructor
model = Sequential()

# Add the input layer
model.add(Dense(13, activation='relu', input_shape=(13,)))

# Add one hidden layer
model.add(Dense(8, activation='relu'))

# Add the output layer
model.add(Dense(3, activation='softmax'))
```

Training the Model

```
# Compile the model with the appropriate loss, optimizer & metric(s)
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# Fit (i.e. train) the model on the train set
model.fit(X_train, y_train, epochs=20, batch_size=1, verbose=1)
```

Evaluating the Model

```
score = model.evaluate(X_test, y_test, verbose=1)
print(score)
```

`score` contains the values for the loss & accuracy of the model on the test set

In closing...

Closing Remarks

- We've barely scratched the surface on the topic of neural networks & deep learning and I've not gone beyond the very basic math involved.
- While the intuition is necessary to understand, I would suggest studying the mathematics to truly appreciate the power of neural networks.
- Keras is a great beginner-friendly library for implementing neural networks and has widespread community support and well-written documentation. However, if you're looking to go beyond the simple APIs available, I would recommend learning how to use TensorFlow.
- If you're interested in learning more, you can check out Google's Machine Learning Crash Course & Machine Learning Glossary.

THANK YOU

References

- Jupyter Notebook: https://goo.gl/yEh6xo
- CZ 4042 Neural Networks (Prof. Jagath Rajapakse)
- https://keras.io/
- http://scikit-learn.org/
- https://towardsdatascience.com/a-visual-introduction-to-neural-networks-68586b0b733b
- https://medium.freecodecamp.org/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076
- Code Screenshots: https://carbon.now.sh/