Machine Learning with TensorFlow & Keras
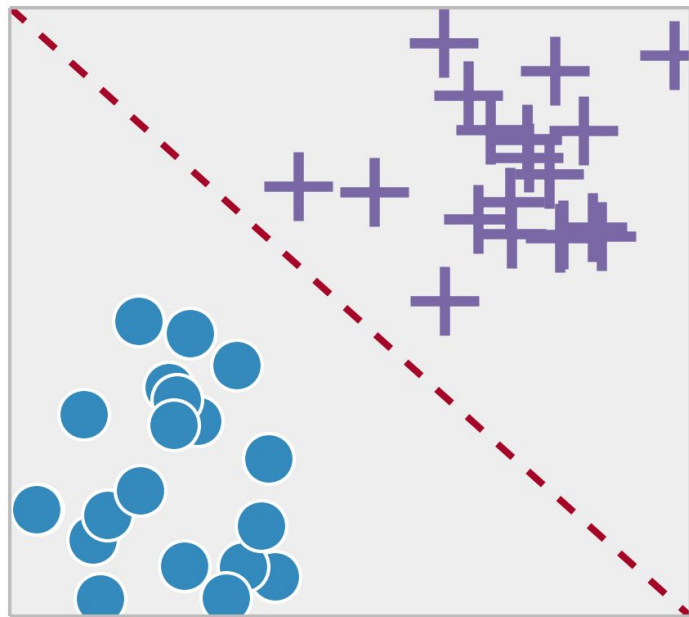
Ganesh Katrapati – PhD Candidate @ IIITH, EC Member – Swecha
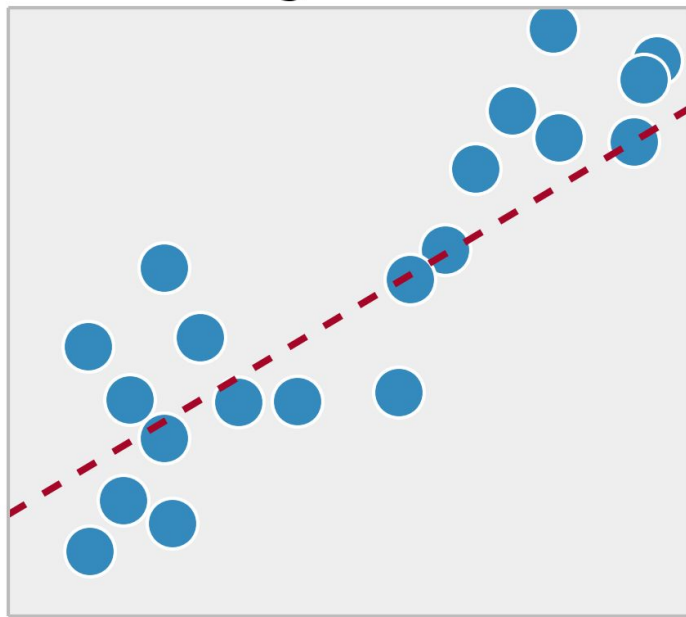
ML ?

Look at the past to predict **a possible** future.

Two Problem Classes (mostly)

Classification

Regression

Regression

- Age vs Height - I have data for ages 1-20, 40-60. Can I form a pattern for 20-40 ?
- Experience vs Salary
- Global Temperature vs FF Emission.
- Price Rise vs Demand.
- Change vs Risk (degree)

Classification

- Apples vs Oranges : I have a *labeled dataset* for Apples vs Oranges. Given a fruit, can I decide ?
- Risk vs Not-A-Risk
- Carcinogenic or Not
- Disease Classification

The General Idea

- So, in general, both Classification and Regression are about learning a **line.**
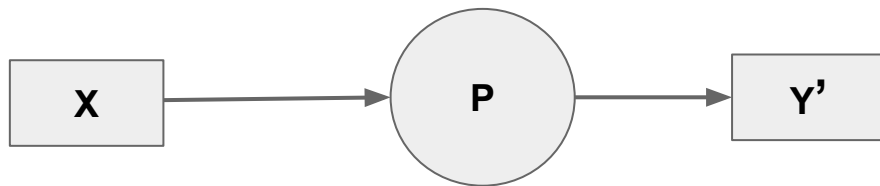- Lets go back to math. What is a line ?

$$Y = MX + C$$

- We know : X (input), Y(output)
- We don't know : M & C (Slope & Intercept)
- Lets call these : **weights**

Points scored
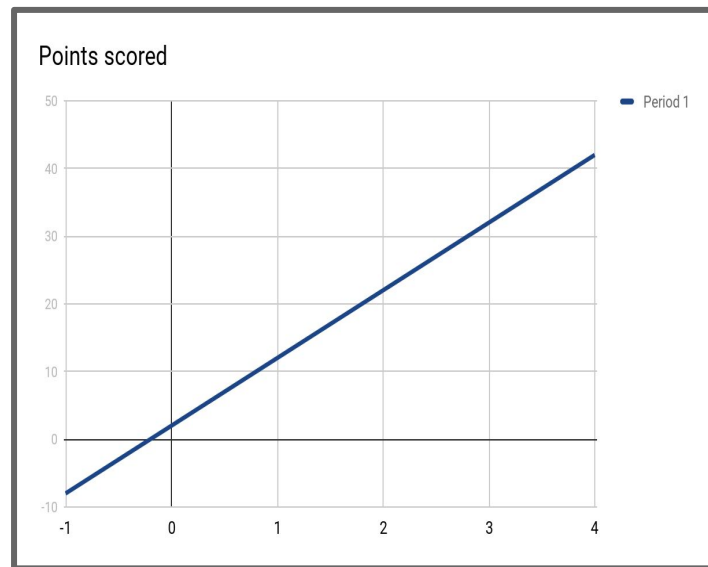
The Machine

$$Y = MX + C$$

- Algorithm
  - For every input **X**, we run it through the Perceptron (**P**)
  - **P** outputs some output **Y'**
 - Our goal is to make **Y'** as similar as possible to **Y.**

The Algorithm

- The central goal : minimise loss function (J)
  - **J = Y - Y'**
- What is learning ?
  - Learn the required weights which minimise J.
  - For each turn, adjust weights proportional to the loss encountered.
  - Usually, learning is not one-shot. So, learning is according to a *learning rate.*

Lets Complicate it a little bit :)

- Input can have more than one property
  - For apples and oranges, it can be color & shape for example
  - $Y = W_1X_1 + W_2X_2 + C$

- Consider a case where
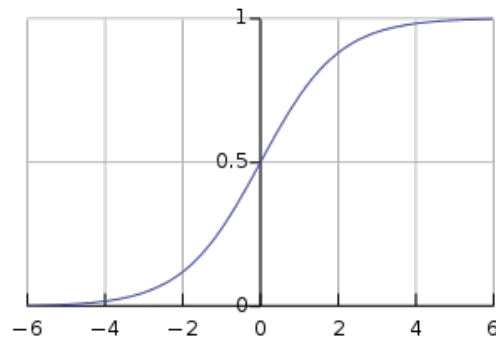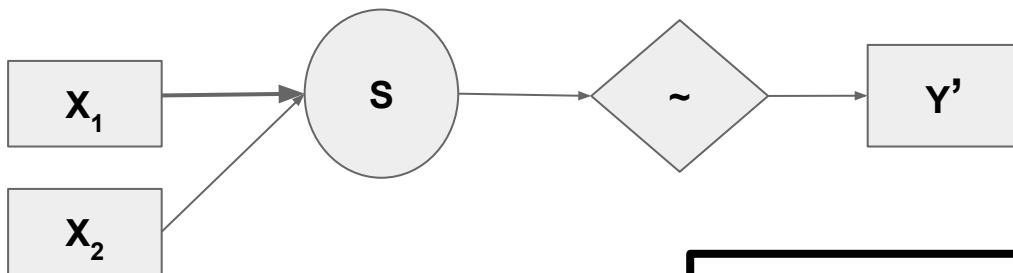Drawing a line is useless.

Linear vs. nonlinear problems
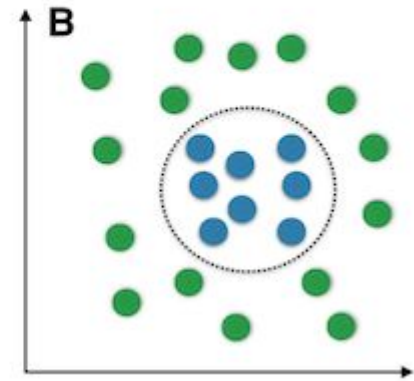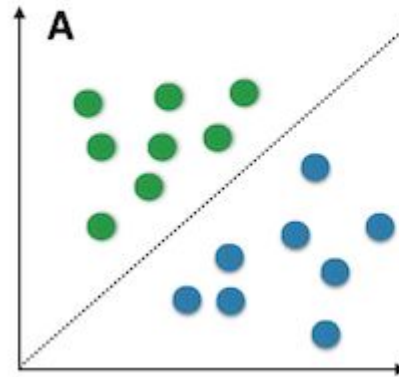
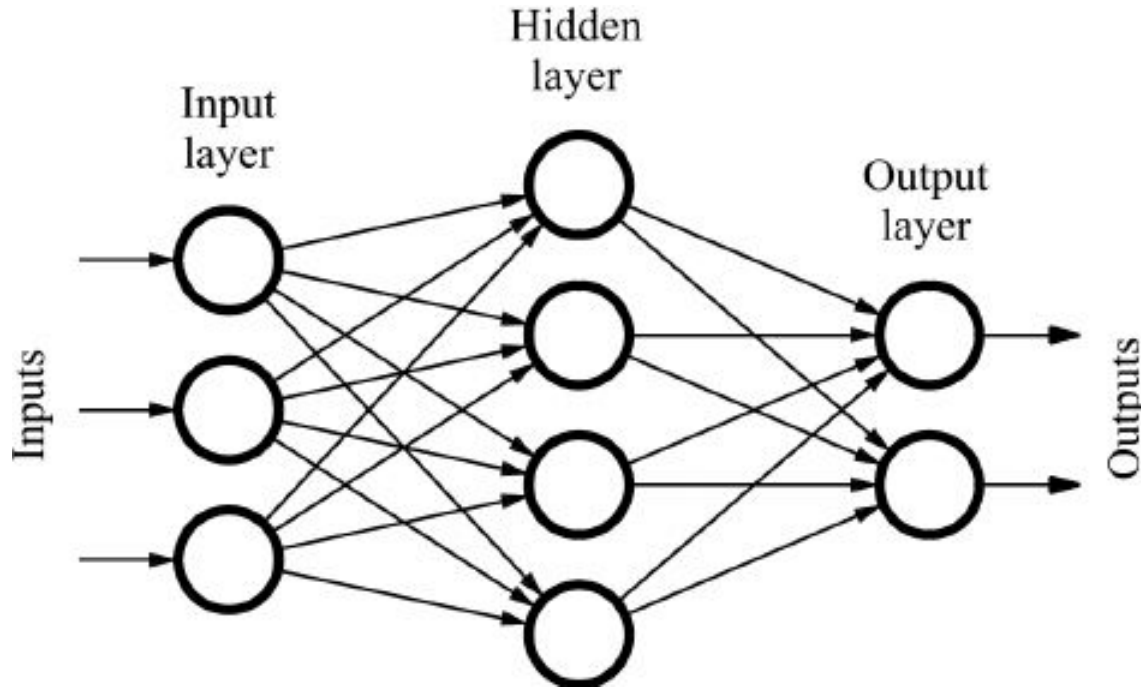# A Super Power : Non-Linearity

- We have to send our output (which is a sum) through a non-linear function (eg:sigmoid)

$$\mathbf{Y} = F\tilde{\ }(W_1X_1 + W_2X_2 + C)$$


```
X₁ → S → ~ → Y'
X₂ ↗
```

ML is a generic function approximation algorithm

Put it all together = NN

- At each 'neuron', we determine its contribution to the loss function.
- When **units** are stacked in the shown manner (**layers).** Complex functions can be learnt
- When we do this for all the units, we approach a desired state
- This is called **Backpropagation.**

Lets get started

- TensorFlow -> A library which is very customisable to the extent of individual layers and units to learn and use neural nets
- **Keras** : TensorFlow for humans (kind-of).

Setting Up

- Python
- pip install tensorflow
- pip install keras
- Windows ?
  - Install Anaconda 3.6
  - Conda install mingw libpython
  - Conda install tensorflow
  - Pip install keras

Our First Neural Net

- Problem : Let us make our NN *learn* **addition.**
- **Inputs : X_1, X_2**
- **Output : Y (where Y = X_1+ X_2)**
- The data will be linearly separable.
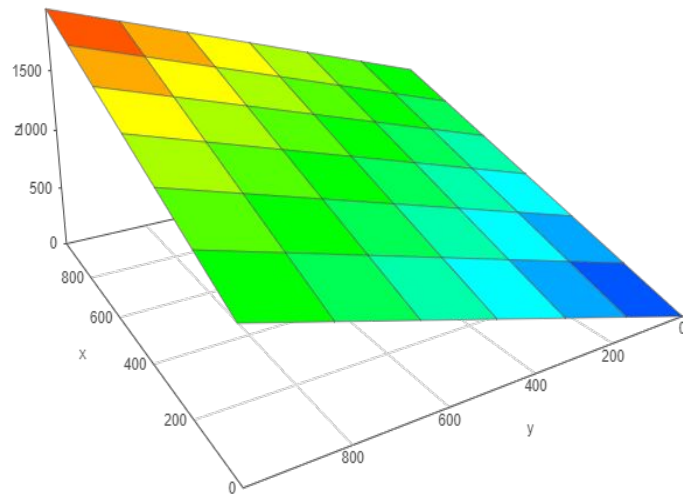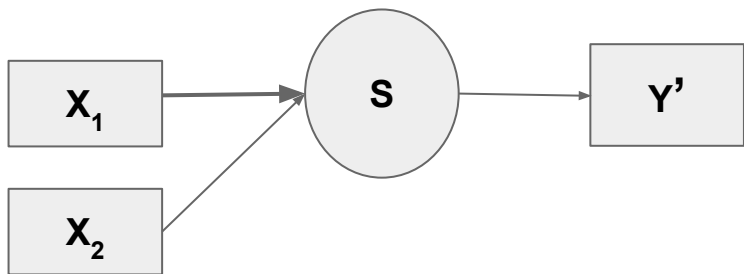
Anatomy of a Keras Script

```python
# Imports
from keras.models import Sequential
from keras.layers import Dense, Activation
```

The basic keras api.
Dense = Summing unit
Activation = A function upon dense.

```python
# Create a model
model = Sequential()
model.add(Dense(1, input_dim=2))
```

Input Dimensions = 2 (X1, X2)
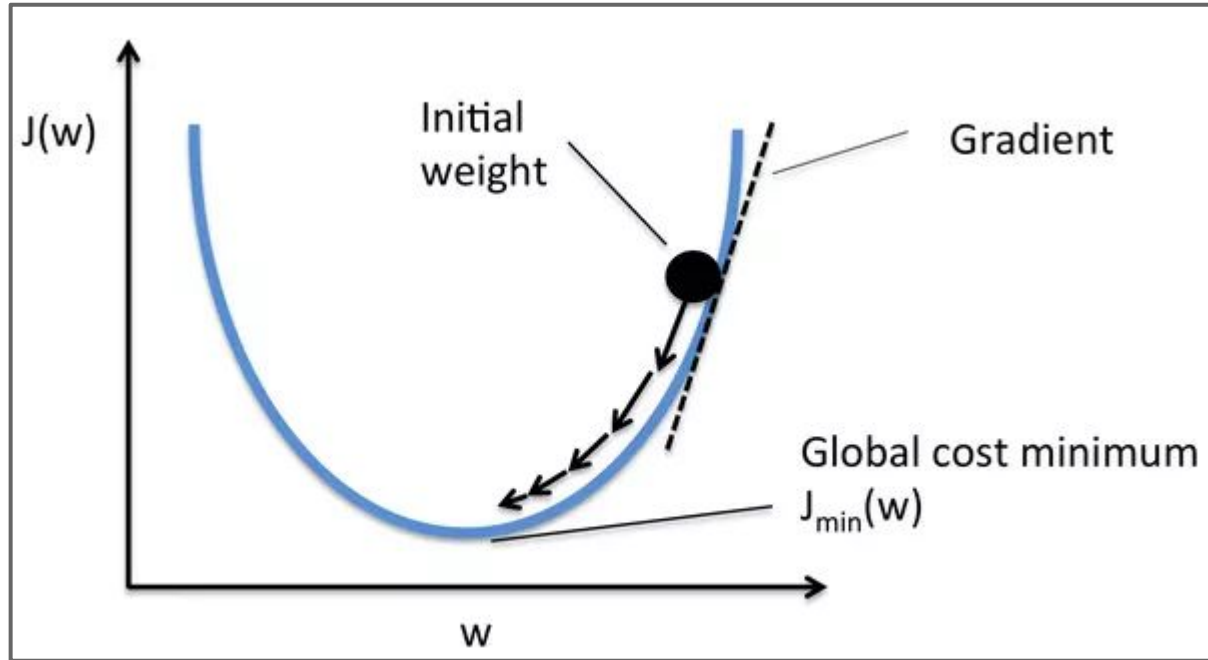Output Dimensions = 1 (Y)

```python
# Compile the model
model.compile(optimizer='sgd',
        loss='mse')
```

SGD = Stochastic Gradient Descent
MSE = Mean Squared Error

A Side Note : SGD & MSE

- Error (J) will be better if it is squared $(Y - Y')^2$ because it will be smoother.

- Gradient Descent is a process of slowly crawling down to the lowest J possible.
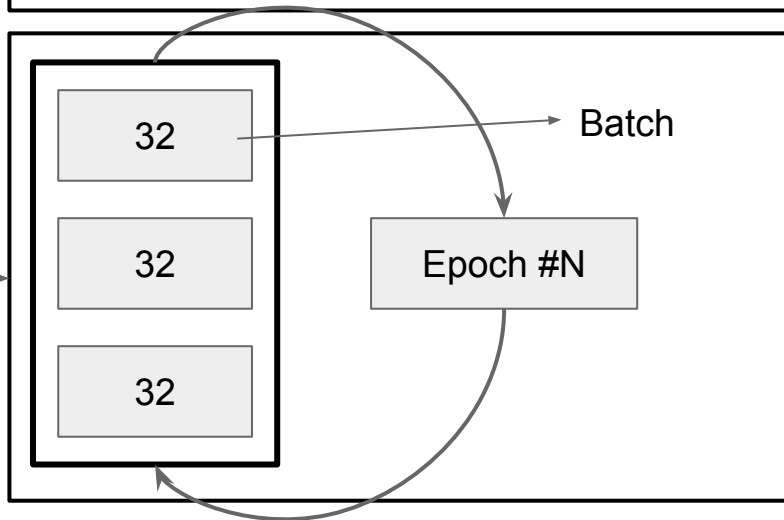
Anatomy of a Keras Script - Training

```
# Generate Training data
import numpy as np
TRAIN_SIZE = 1000
X = np.random.random((TRAIN_SIZE, 2))
Y = X.sum(axis = 1)
```

```
Import numpy
Generate X (1000x2) matrix
Sum columns to get Y (X1 + X2)
```

```
# Train
model.fit(X, Y, epochs=10, batch_size=32)
```

32

32

32

Batch

Epoch #N

Anatomy of a Keras Script - Testing

```
# Generate Training data
TEST_SIZE = 100
X_test = np.random.random((TRAIN_SIZE,
2))
Y_test = X_test.sum(axis = 1)
```

```
# Evaluate(Optional)
score = model.evaluate(X_test, Y_test)
print("Error is", score * 100, "%")
```
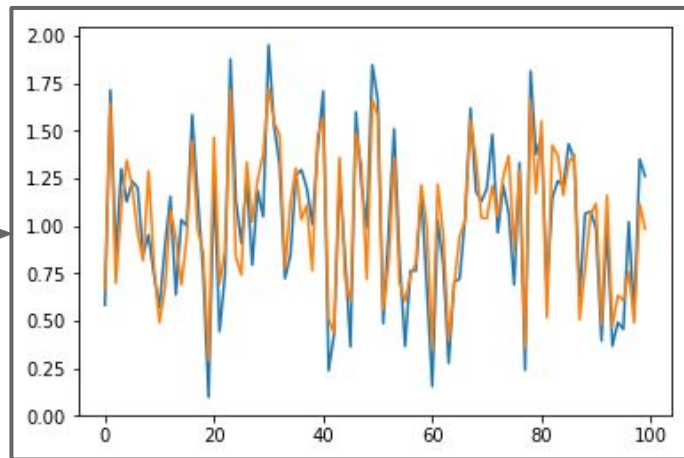
Score is the Mean Squared Error over all the samples

```
# Get the predicted Y's
Y_pred = model.predict(X_test)
print(Y_test, Y_pred)
```
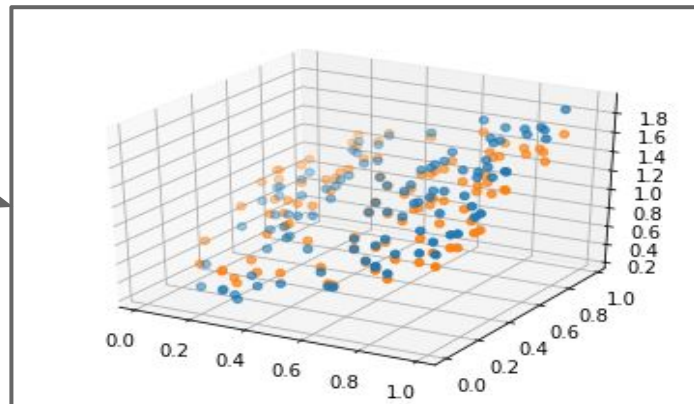
Bonus : Plot

```
# Y_test vs Y_pred
import matplotlib.pyplot as plt
plt.plot(range(0,len(Y_test)), Y_test)
plt.plot(range(0,len(Y_pred)), Y_pred)
plt.show()
```



```
# Get the predicted Y's
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_test[:,0], X_test[:,1], Y_test)
ax.scatter(X_test[:,0], X_test[:,1], Y_pred)
plt.show()
```
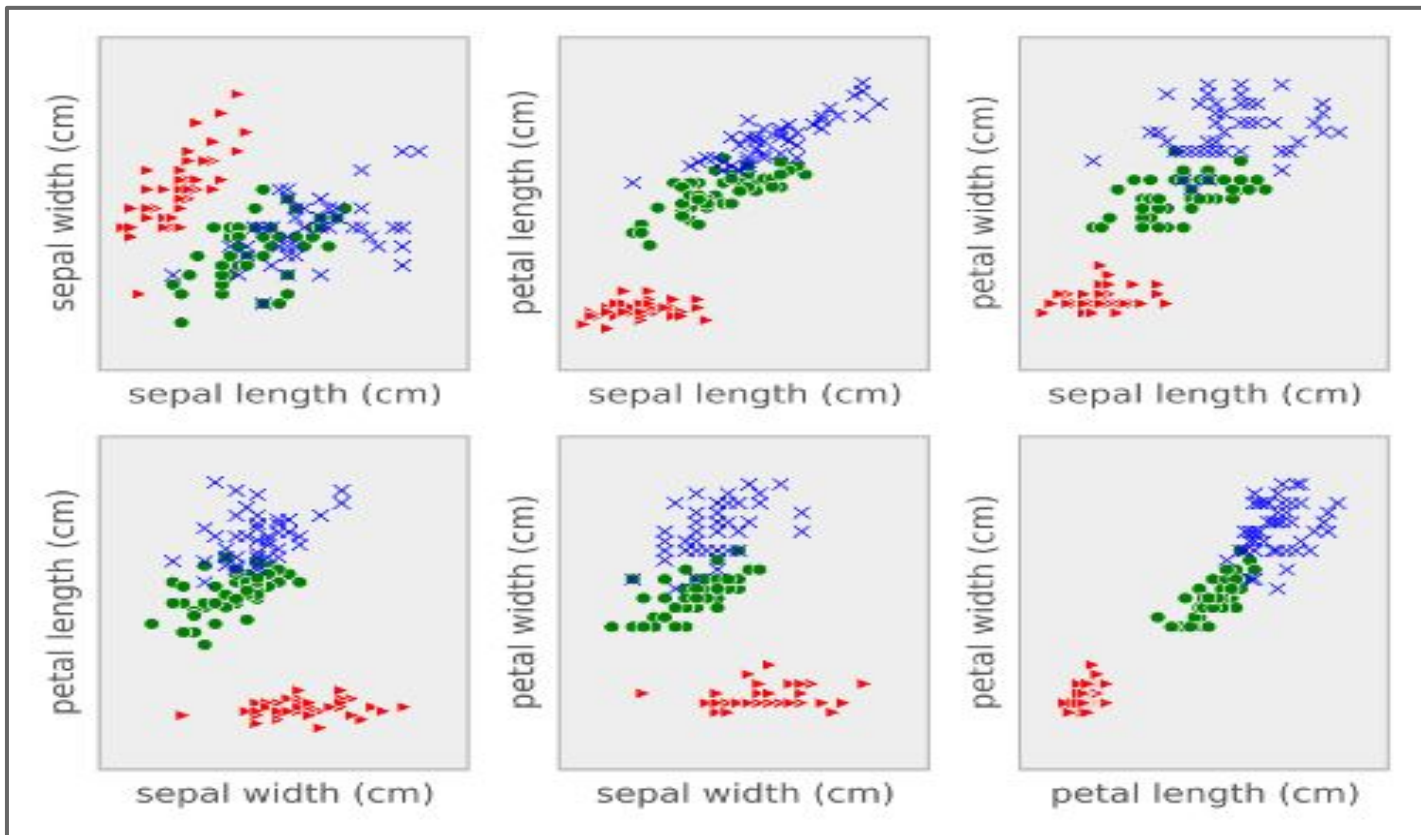
Kudos !. You have built the great-great-great-great------grandmother of Skynet.
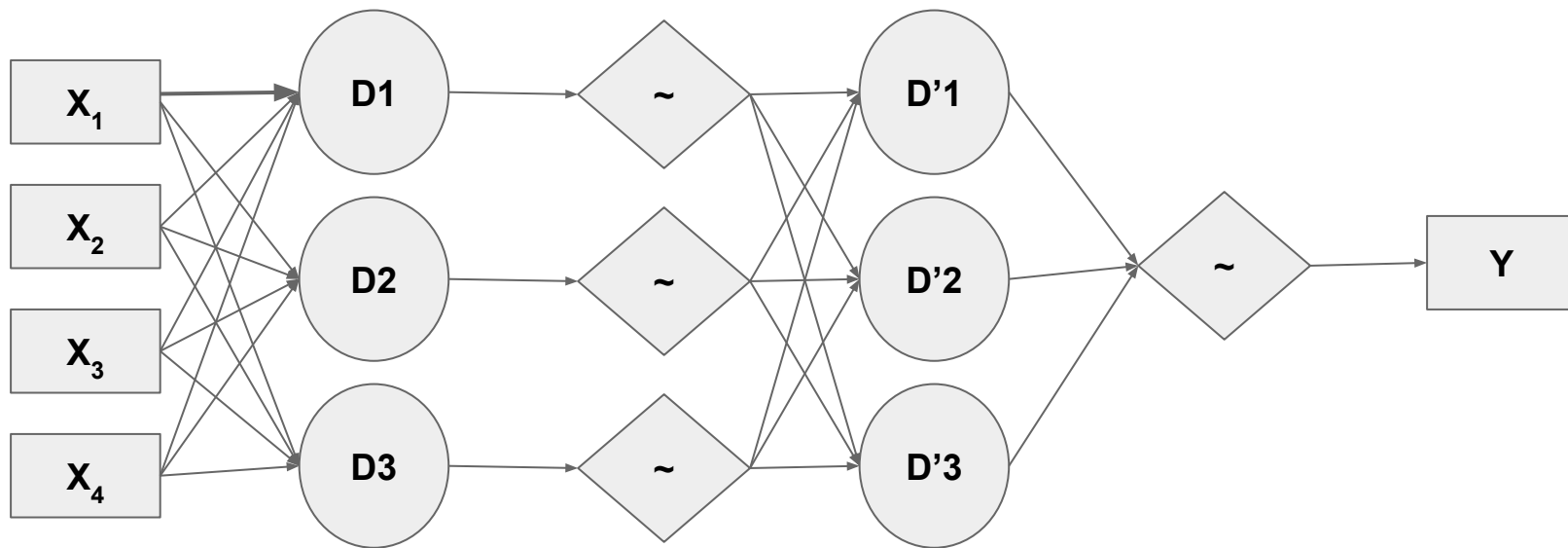
Take 2 : Multi-Class Classification

- Problem : Classify IRIS flowers into subtypes.
- **Inputs :** Sepal Length, Sepal Width, Petal Length, Petal Width
- **Output : Class Name**
  - Iris Setosa
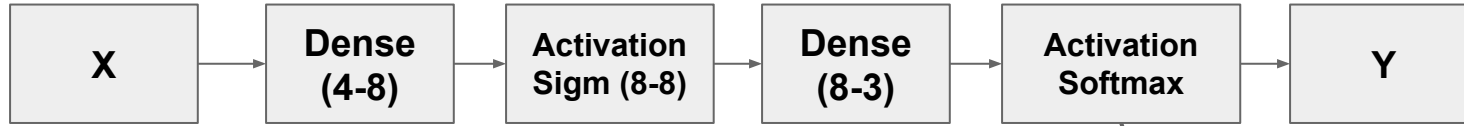 - Iris Versicolor
  - Iris Virginica

IRIS

IRIS - Network Design

IRIS - Network Design

X	→	Dense (4-8)	→	Activation Sigm (8-8)	→	Dense (8-3)	→	Activation Softmax	→	Y

```
# New Model
model = Sequential()
model.add(Dense(units = 8, input_dim=4))
model.add(Activation('sigmoid'))
model.add(Dense(units = 3))
model.add(Activation('softmax'))
```

Softmax function gives relative likelihood of which class a sample belongs to. For example : X might be (75%, 15%, 10%) if Iris (Setosa, Virginica, Versicolor)

Why More Layers ?

- For more complex problems (data is not simply separable), more layers allow learning these complex decision boundaries

- Any NN with #L > 2 is called a Deep Neural Net. aka "Deep Learning"

- Usually, each layer learns a level of generalisation.
  - Say for face recognition,
  - Layer 1 learns edges & lines
  - Layer 2 learns facial components
  - Layer 3 learns to distinguish faces themselves.
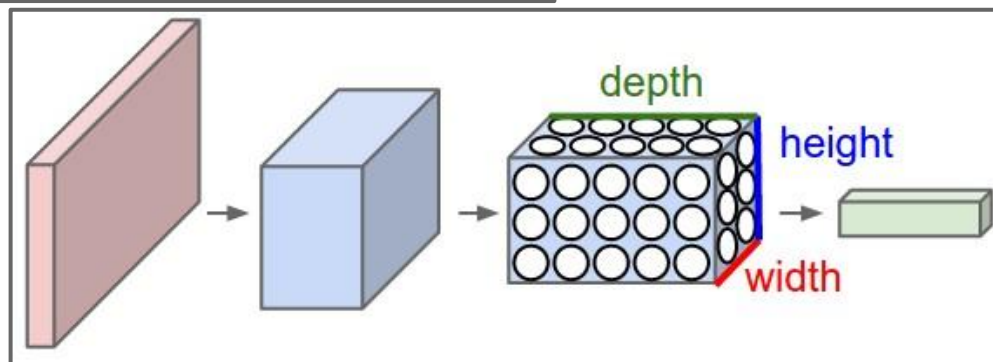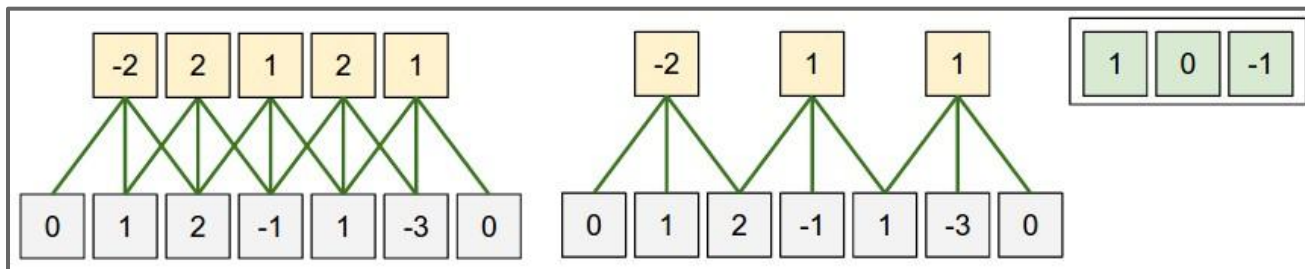
Example 3: Image Recognition

- MNIST is a dataset for handwritten digits
- It comes with keras preinstalled
- The architecture we will follow here is
  - Image -> Vector
  - Vector => Dense Layer 1
  - Dense Layer 1 => Relu
  - Dropout
  - Dense Layer 2 => Relu
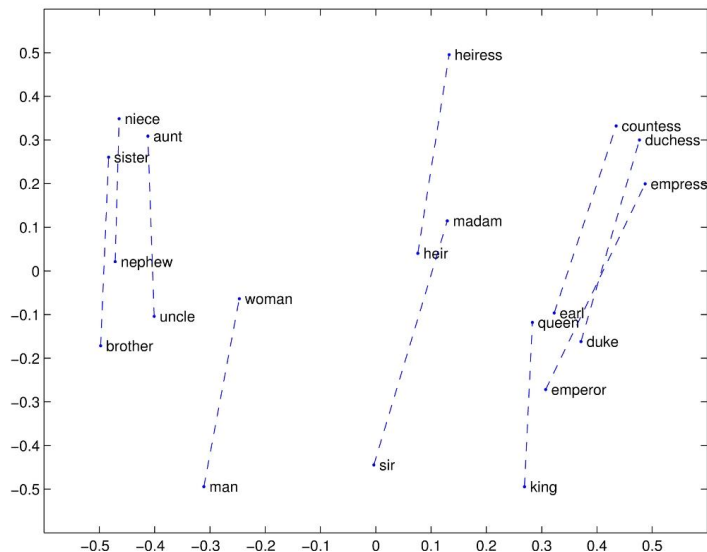 - Dropout
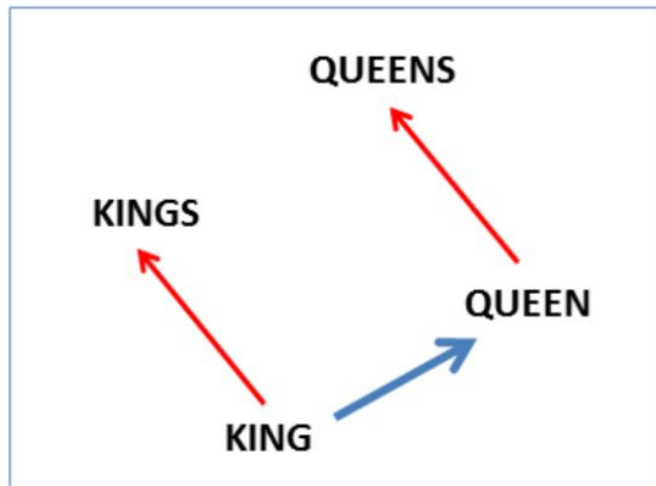 - Dense Layer 3 => Softmax

What's Next
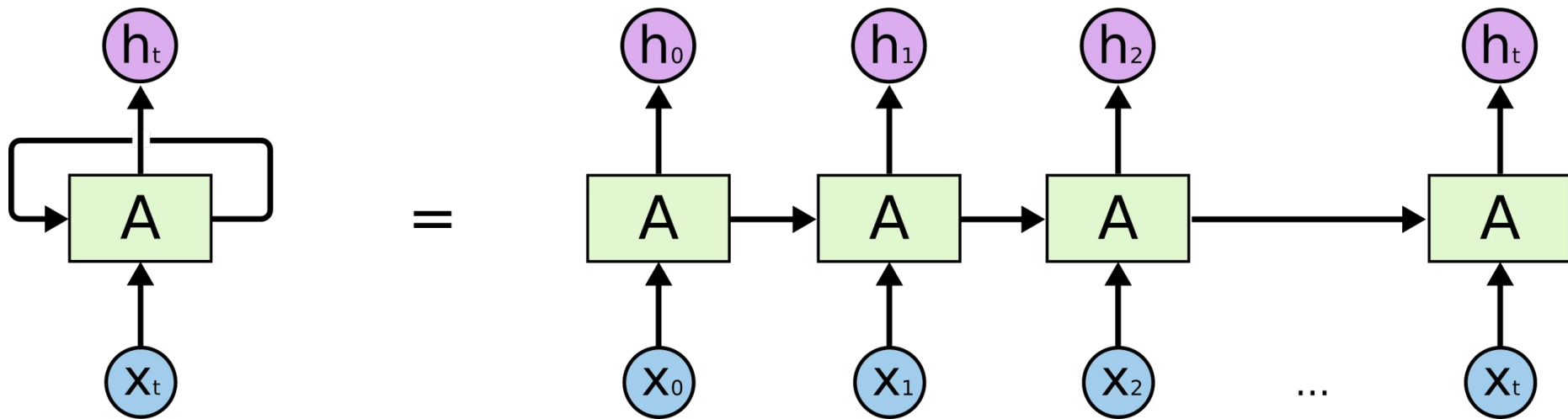
Recurrent Neural Network

What's Next

Convolutional Neural Network

What's Next

Embeddings in Word Space

Problems

- Training Bias
  - Tay the Bot
  - FaceApp
  - Court Judgements
  - Elections
  - Why does this matter so much to us ? :
  - https://www.ted.com/talks/joy_buolamwini_how_i_m_fighting_bias_in_algorithms#t-440352
- Job Loss
- Commoditization
  - Our digital footprint is the product that we sell.
- Privacy/Security