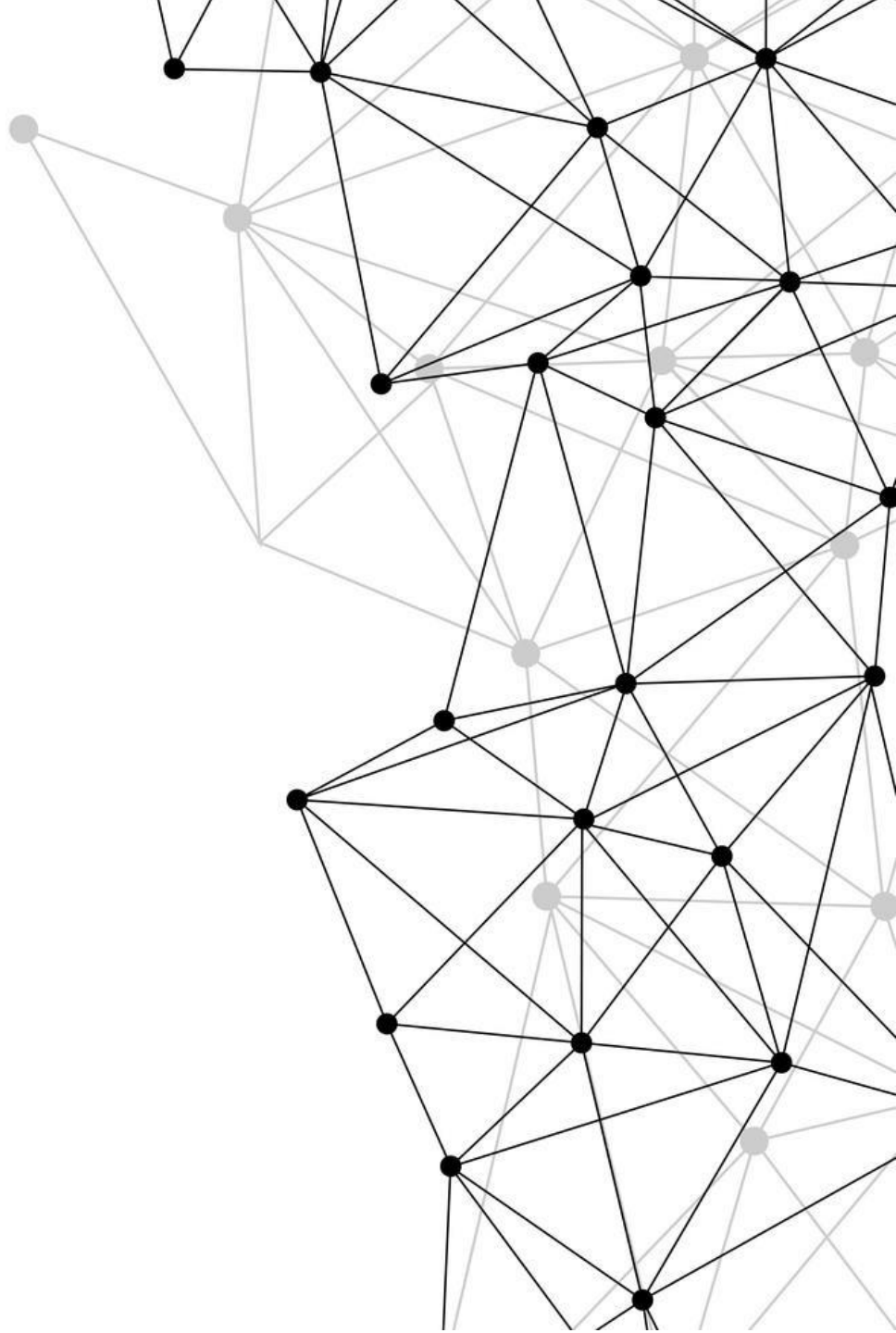
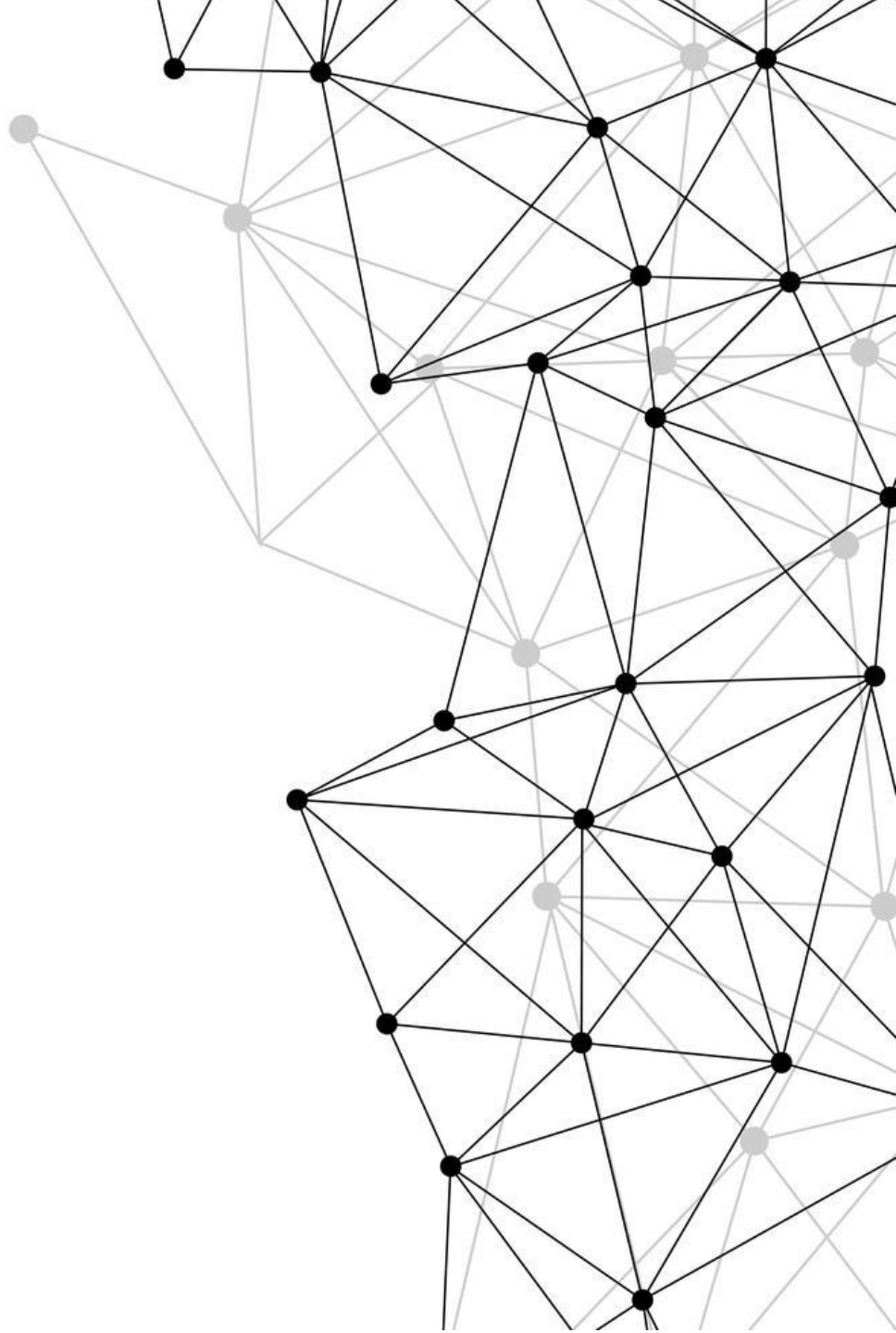
Neural Networks

Sina K. Maram, M.Sc., P.Eng.



Neural Networks

Sina K. Maram, M.Sc., P.Eng.



By the end of this lecture you'd be able to:

- Discuss the concept of Machine Learning
- Discuss the concept of Supervised and Unsupervised Learning
- Understand the concept of Overfitting and Underfitting
- Understand the concept of Bias and Variance
- Build your first Neural Network model

Agenda:

01

Supervised Vs. Unsupervised
Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

02

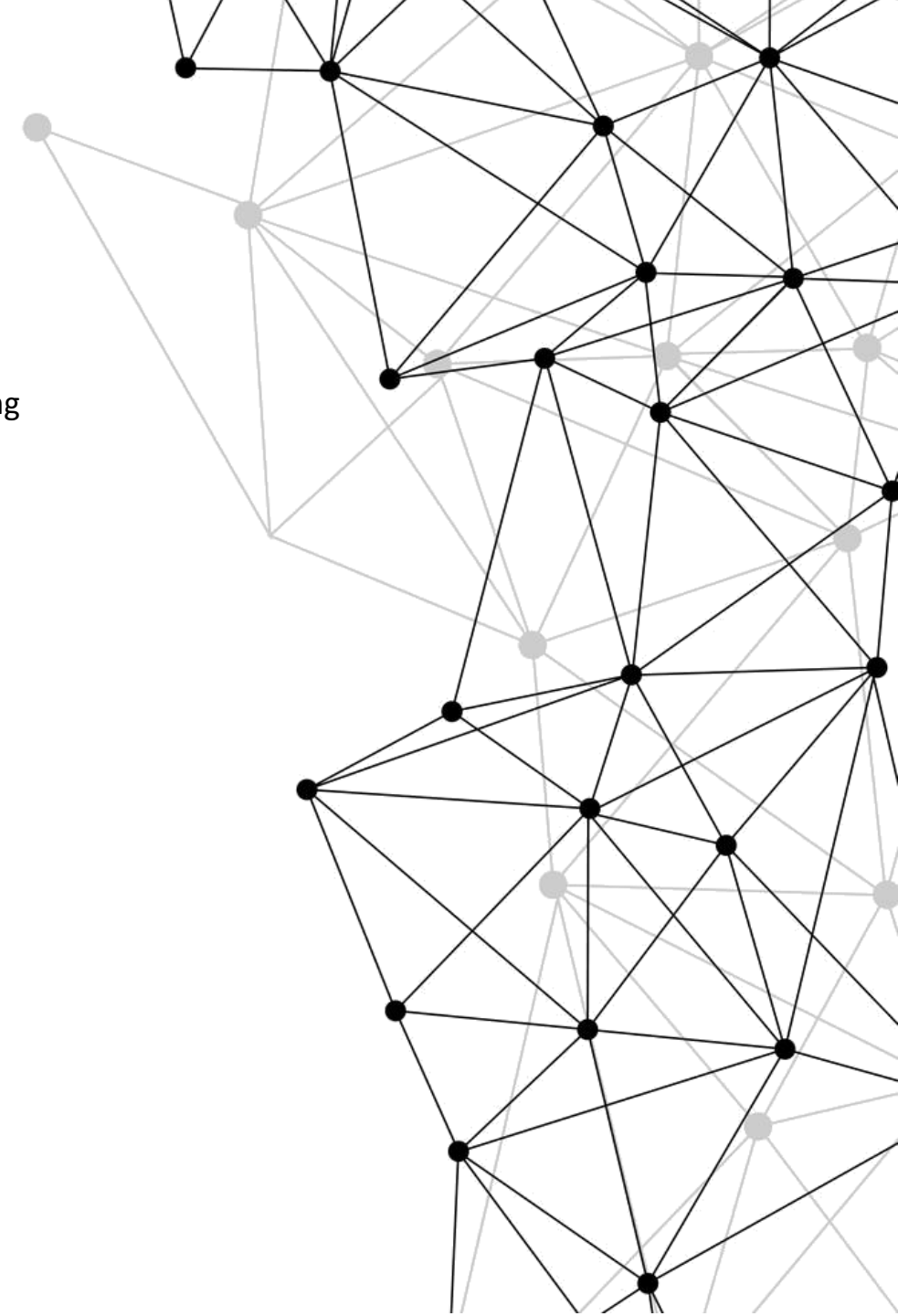
Model Accuracy
Overfitting, Underfitting, Bias & Variance

03

Neural Networks Mathematics
Back Propagation Calculations

04

Building your first Neural Network
Cell Tower Anomalies example



01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

What is machine learning?

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

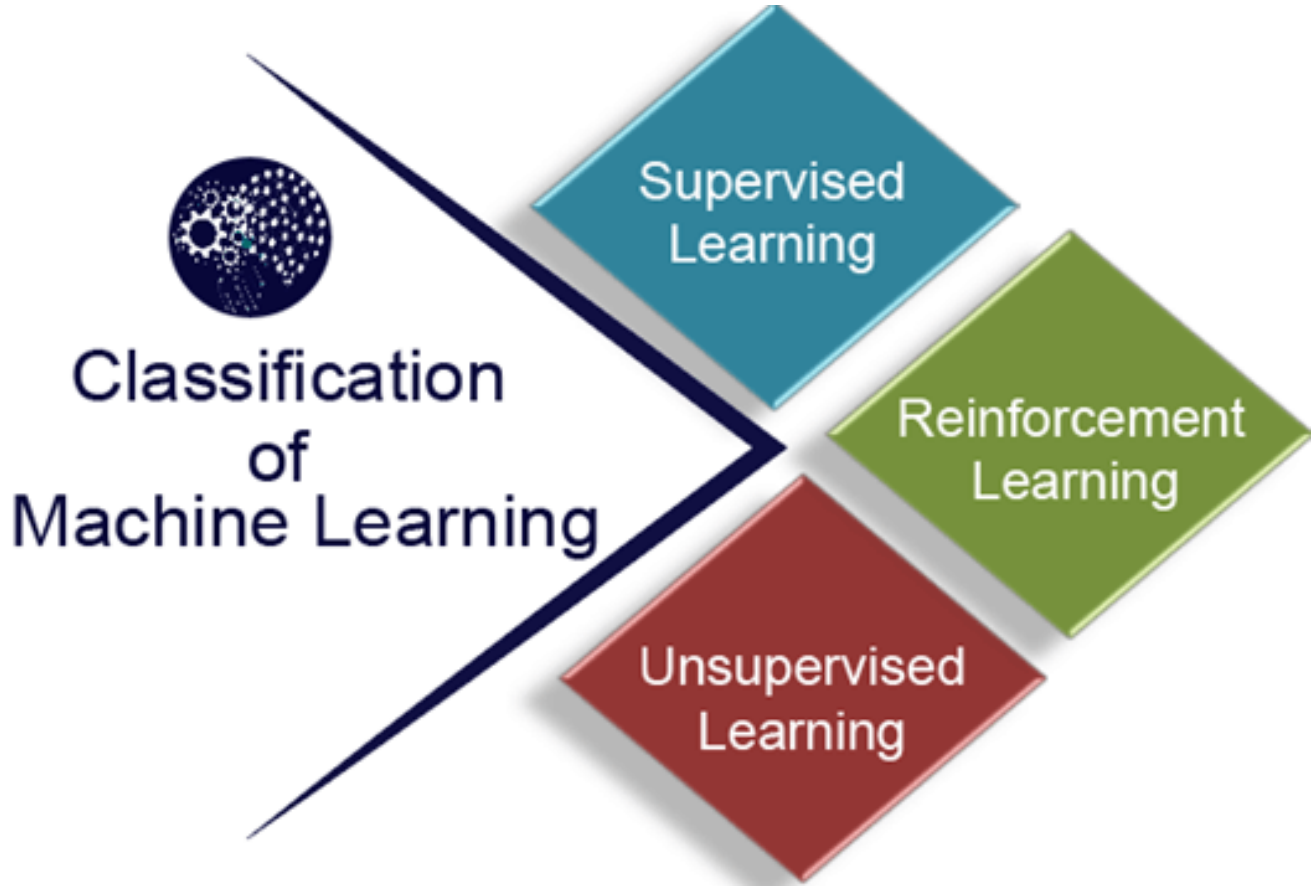
Why is machine learning important?

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning



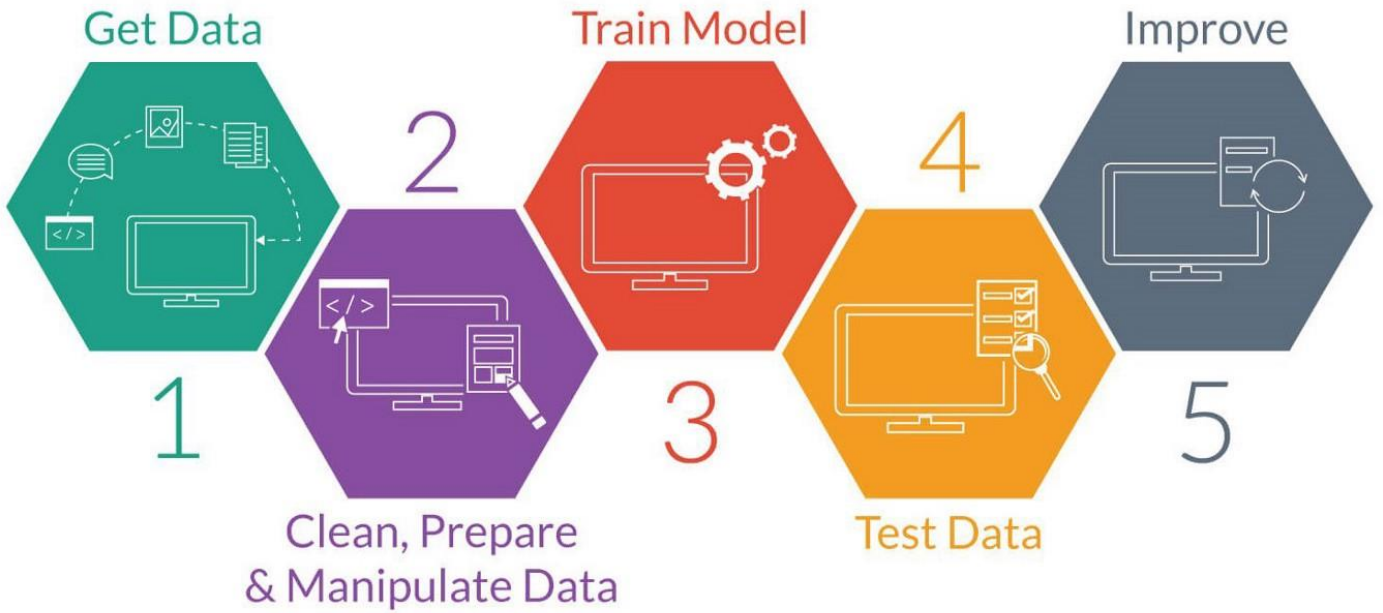
01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Supervised Learning

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately.



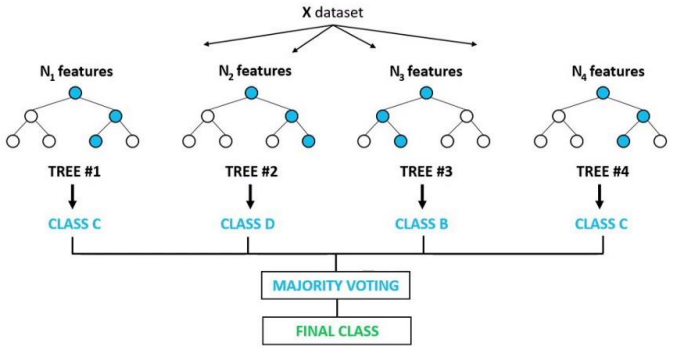
01

Supervised Vs. Unsupervised

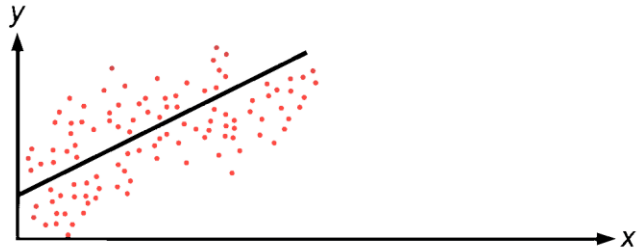
Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Supervised Learning - Examples

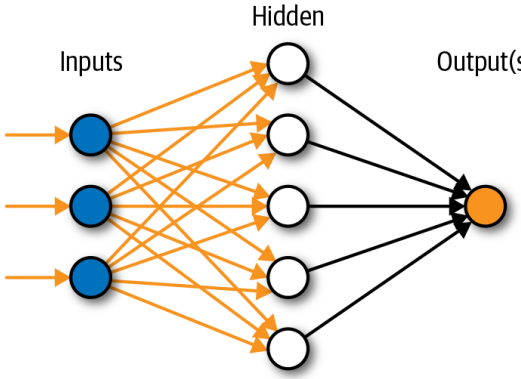
Random Forest Classifier



Linear Regression



Artificial Neural Network



01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Class discussion:
Supervised Learning Pros & Cons

01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Class discussion: Supervised Learning Pros & Cons

Pros

- You will have an exact idea about the classes in the training data
- It a simple process for you to understand.
- You can find out exactly how many classes are there before giving the data to training
- After the training is completed, you don't need to keep the training data in your memory.
- Great choice for recommendation and classification problems
- Predicting numerical target values for given data and labels

Cons

- It is limited in a variety of sense
- Can not give you unknown information from the training data
- In case of classification, if the input does not belong to any classes, you would get an incorrect result
- In broader term, if input belongs to a data domain beyond the training, you would get an incorrect result
- Sometimes we can not always give lots of information with supervision. It is not suitable for exploratory research

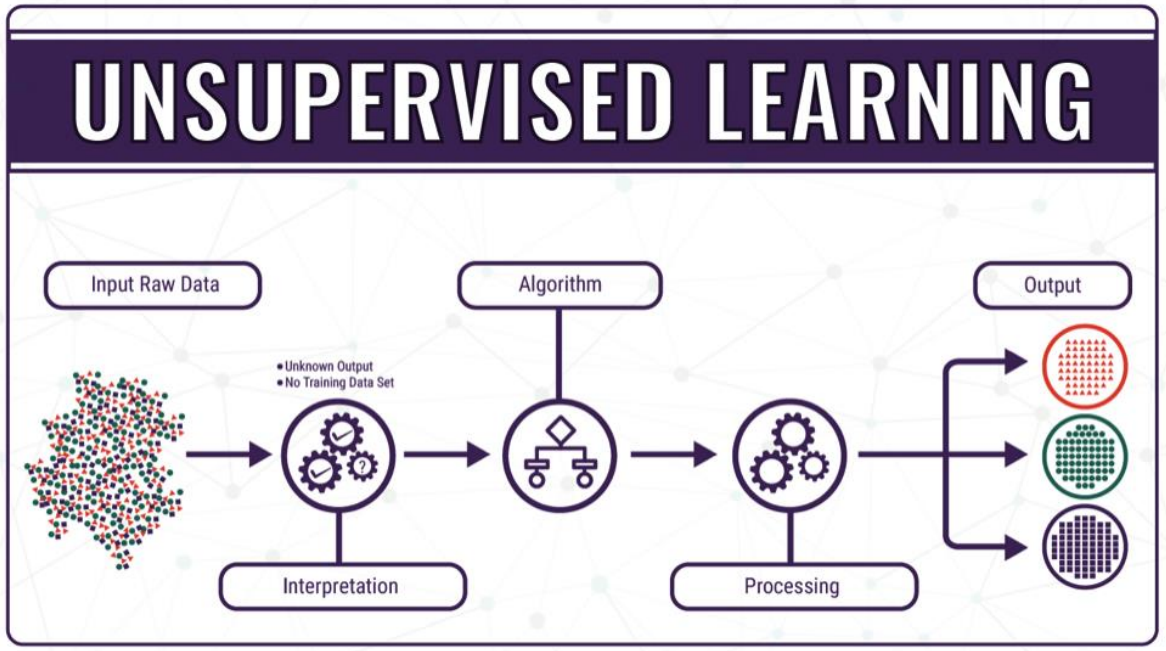
01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Unsupervised Learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention



01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Unsupervised Learning - Example

https://www.youtube.com/watch?v=Lu56xVlZ40M

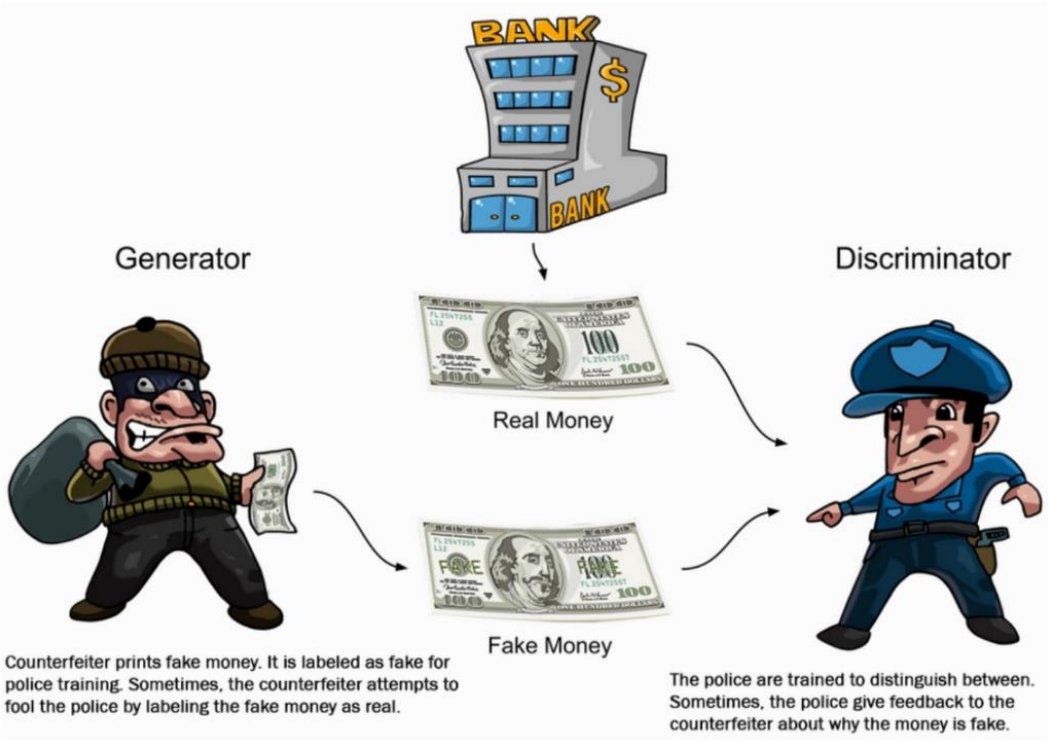
01

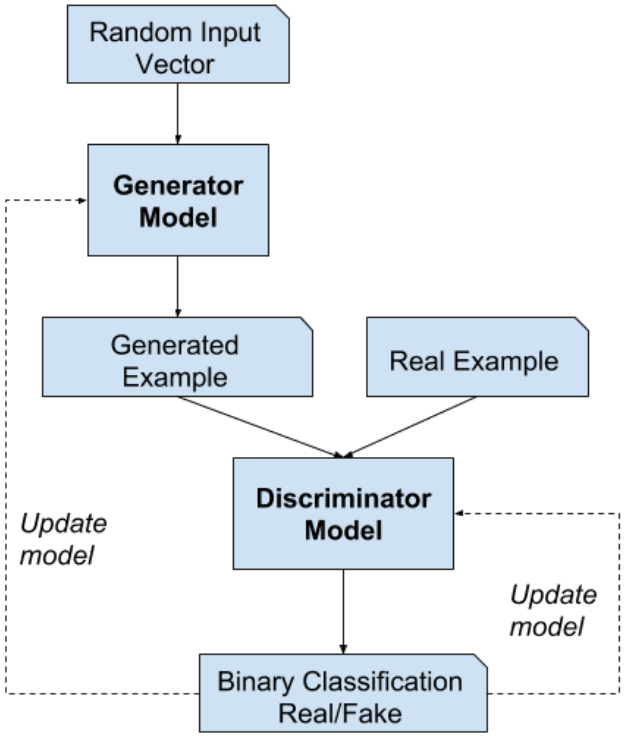
Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Unsupervised Learning – Other Examples

Generative Adversarial Networks	Automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset





01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Unsupervised Learning – Other Examples

Style Transfer	an optimization technique used to take three images, a content image, a style reference image (such as an artwork by a famous painter), and the input image you want to style — and blend them together such that the input image is transformed to look like the content image, but “painted” in the style of the style image.

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Class discussion:
Unsupervised Learning Pros & Cons

01

Supervised Vs. Unsupervised

Machine Learning, Neural Networks use cases in supervised/unsupervised Learning

Class discussion: Unsupervised Learning Pros & Cons

Pros

- Complicated architecture at times results in seeing features human minds cannot visualize
- Ability to detect hidden patterns which hold utmost importance in the industry and has widespread real-time application
- Lesser complexity compared to the supervised learning. Complications is at architecture level
- Much easier to obtain unlabeled data

Cons

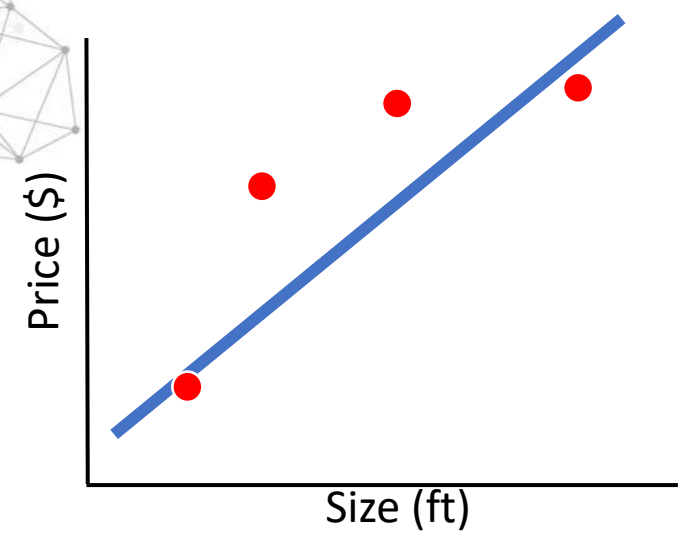
- It is costlier as it may require human intervention with domain knowledge
- Usefulness of the obtained results are up to debate since there is no label or output measure to confirm
- Results often have lesser accuracy
- Ethics of application (e.g. Deep Fake)

Debugging a learning algorithm

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

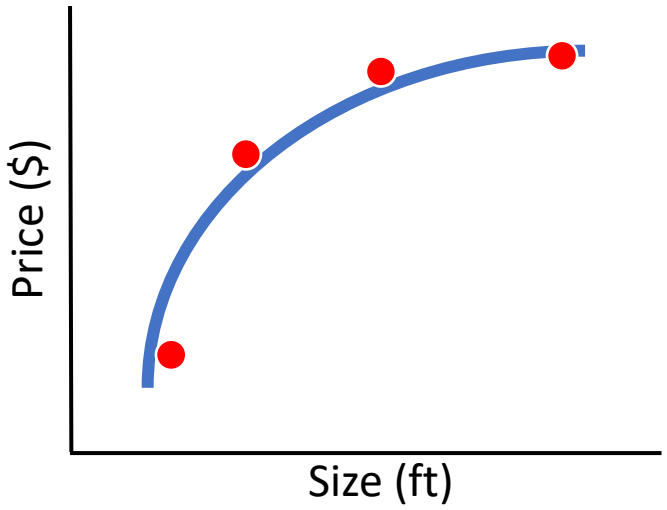
- You have built you awesome linear regression model predicting price
- Work perfectly on you testing data
- Then it fails miserably when you test it on the new data you collected
- What to do now?

Debugging a learning algorithm – Evaluate your hypotheseis



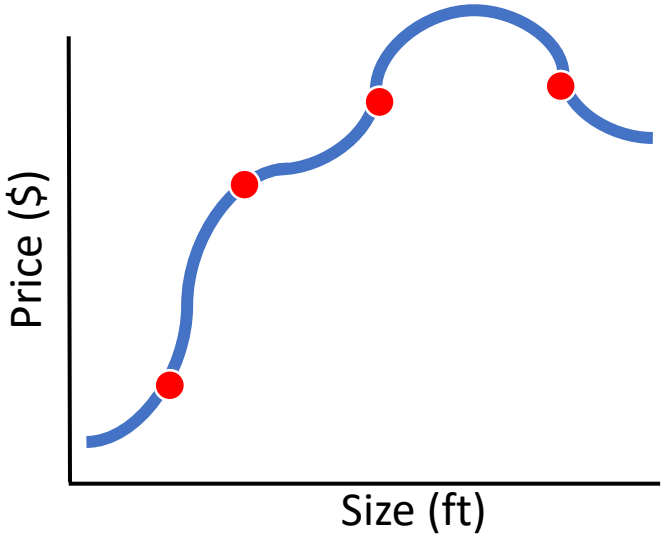
$$\theta_0 + \theta_1 x$$

Underfit



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

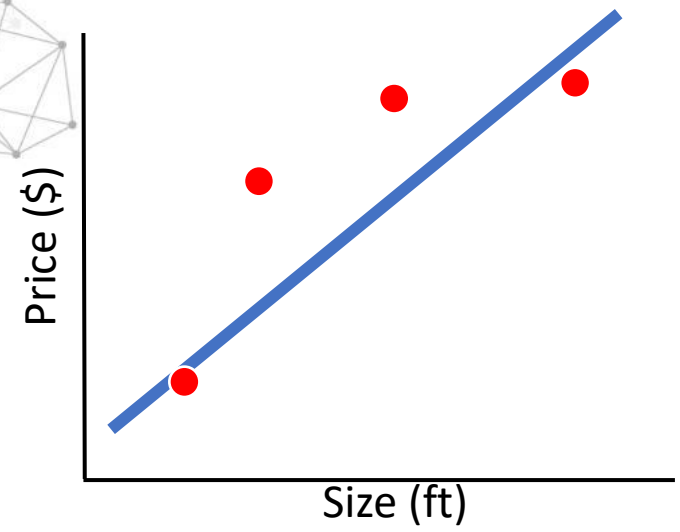
Just right



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

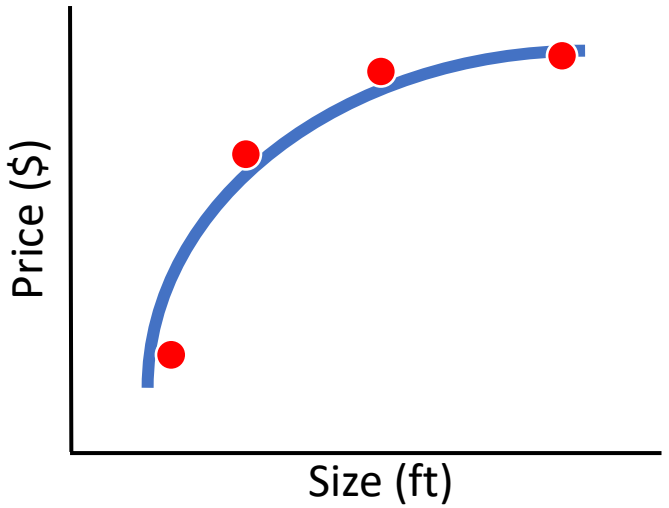
Overfit

Debugging a learning algorithm – Bias and Variance



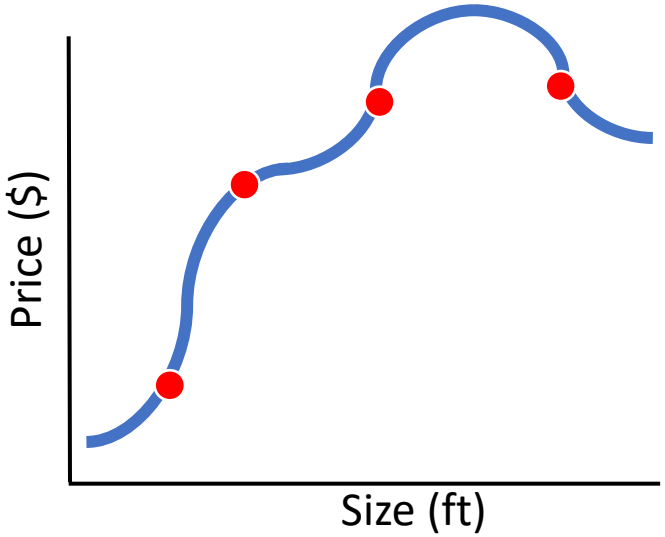
$$\theta_0 + \theta_1 x$$

Underfit
High bias



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Just right



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Overfit
High Variance

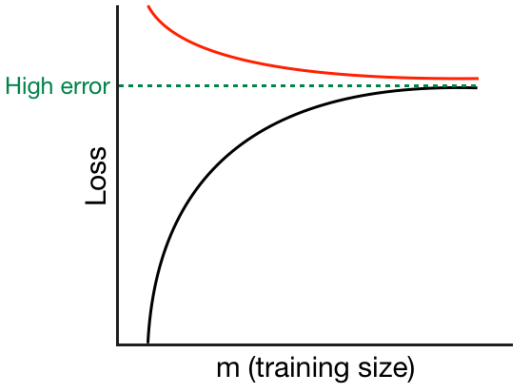
Model Accuracy

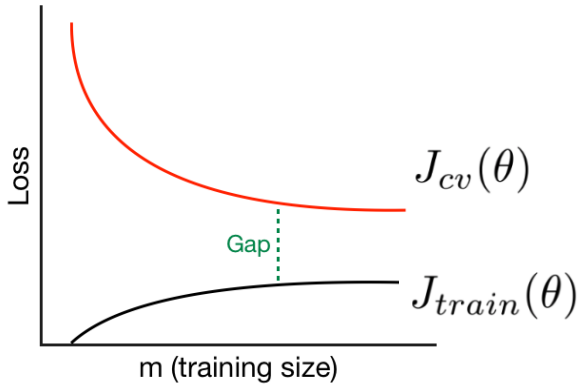
Overfitting, Underfitting, Bias & Variance

Debugging a learning algorithm – Adjusting Learning Curve

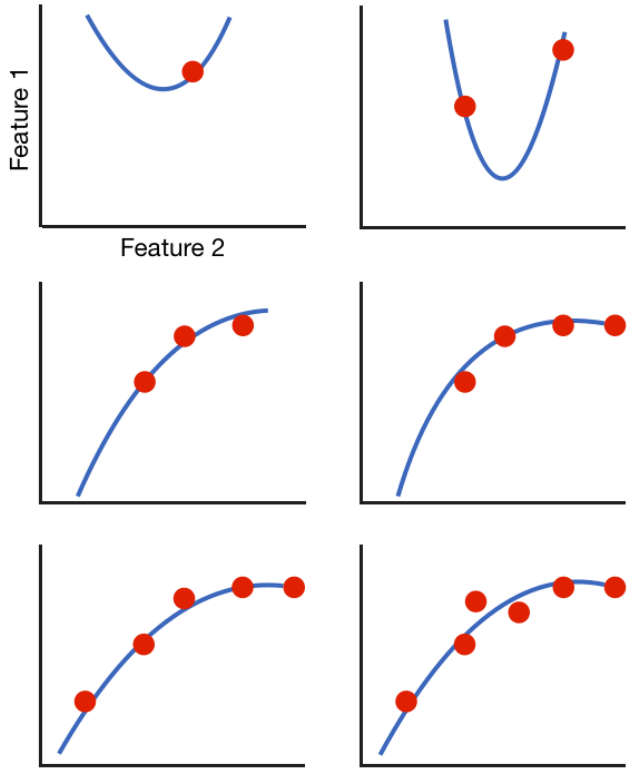
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$





$$\theta_0 + \theta_1 x + \theta_2 x^2$$

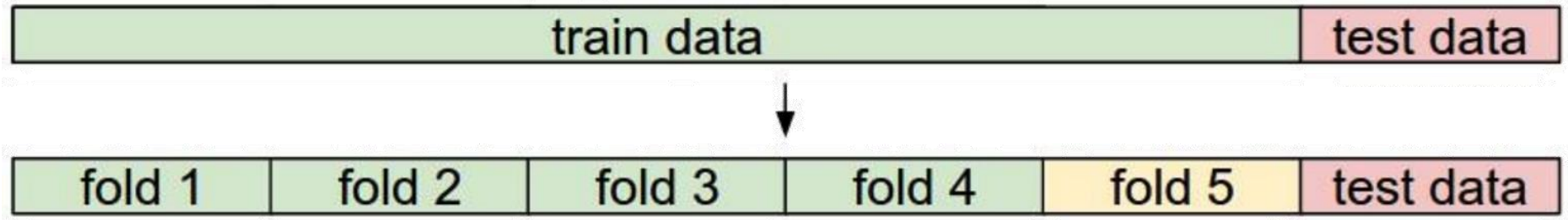


Debugging a learning algorithm – Train on unseen data

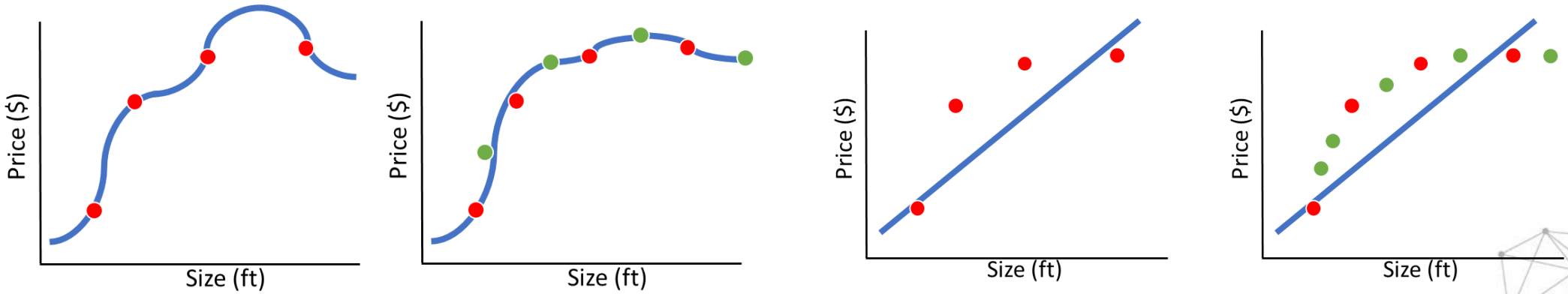
Model does not generalize to unseen data

- Fail to predict things that are not in training sample
- Pick a model that has **lower generalization error**

How to evaluate generalization error?



Debugging a learning algorithm – Adding more data to the mix



More data is likely to help when your model has high variance

More data doesn't help when your model has high bias

Model Accuracy

Overfitting, Underfitting, Bias & Variance

Safeguards:

- **Batch Normalization:** This method consists of adding normalization operation before the activation step at each node of the layer. This way, distribution of each layer's activation input will be centered at zero and will be spread within 3 standard deviation. This will reduce the risk of co-adaptation during training

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, kernel_initializer="he_normal", use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("elu"),
    keras.layers.Dense(100, kernel_initializer="he_normal", use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("elu"),
    keras.layers.Dense(10, activation="softmax")
])
```

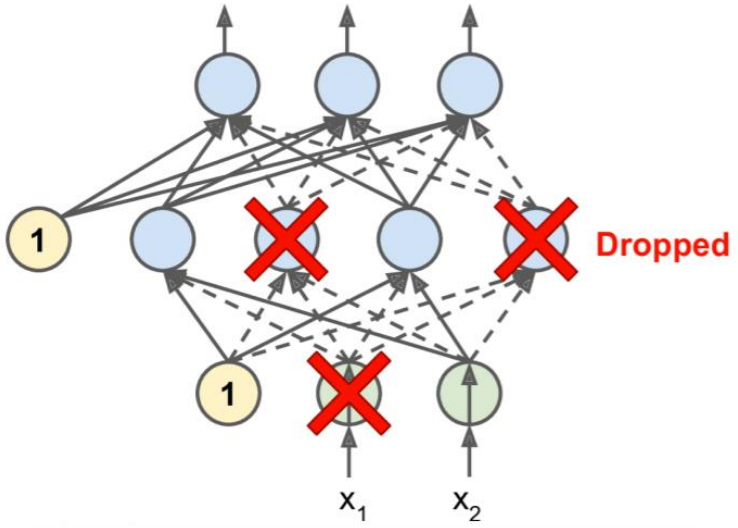
- **Gradient Clipping:** A popular technique to lessen the exploding gradients problem is to simply clip the gradients so that they never exceed some threshold.

```
optimizer = keras.optimizers.SGD(clipvalue=1.0)
model.compile(loss="mse", optimizer=optimizer)
```

Model Accuracy

Overfitting, Underfitting, Bias & Variance

Safeguards:



- **Dropout:** At each iteration, the model will drop some nodes based on a probability (Drop out rate ~ 50%). This is also to avoid co-adaptation behaviour of neurons.

- **Early Stopping:** Stop training when performance on validation set starts to drop

- **Pre-trained Models:** Transfer Learning is extremely popular particularly in image segmentation architectures due to commonalities between early layers

03

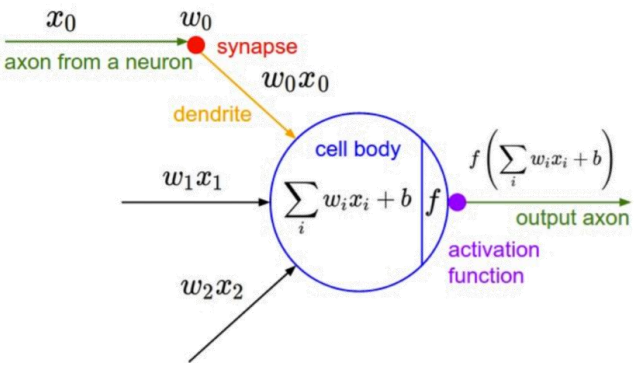
Neural Networks Mathematics

Forward/Back Propagation Calculations

Forward Propagation:

X → Linear Term $Z = w_1x_1 + w_2x_2 + b$ → Rectified Term $\hat{y} = \sigma(z)$

$\sigma(z) = \frac{1}{1+e^{-z}}$



$$X = \begin{bmatrix} 1 & 5 \\ 3 & 2 \end{bmatrix}$$ Sample 1 / Sample 2

First iteration: $Z = w_1x_1 + w_2x_2 + b$ → $\hat{y} = \sigma(z)$ ($\sigma(z) = \frac{1}{1+e^{-z}}$)

$w_1 = 3\,, w_2 = -1$

$$\hat{y} = \begin{bmatrix} 0.12 \\ 0.00 \end{bmatrix}$$

$$y_{actual} = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

Forward Propagation

the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.

03

Neural Networks Mathematics

Forward/Back Propagation Calculations

Backward Propagation:

Linear Term

$$Z = w_1x_1 + w_2x_2 + b$$

Rectified Term

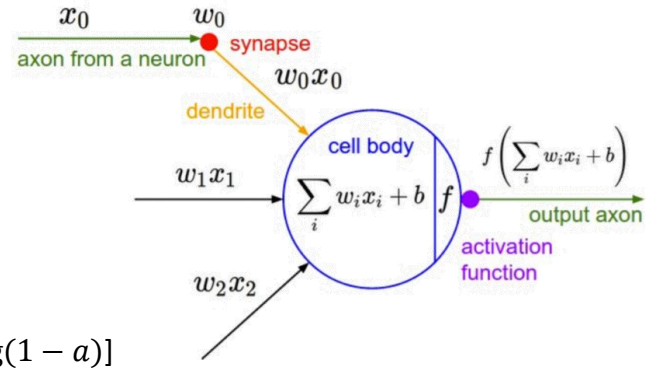
$$\hat{y} = \sigma(z)$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Loss Function

$$L(a, y)$$

$$L(a, y) = -\left[y\log(a) - (1-y)\log(1-a)\right]$$



$$X = \begin{bmatrix} 1 & 5 \\ 3 & 2 \end{bmatrix}$$

Sample 1

Sample 2

Forward Propagation

$$\hat{y} = \begin{bmatrix} 0.12 \\ 0.00 \end{bmatrix}$$

$$y_{actual} = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

$L(a, y)$

adjust weights

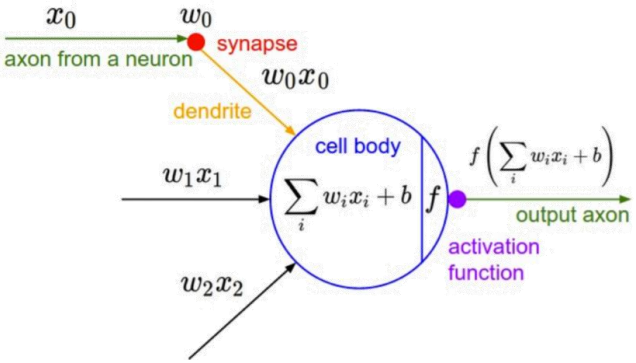
Backward Propagation

Iterate until Cost Function is minimized

Neural Networks Mathematics

Forward/Back Propagation Calculations

Backward Propagation:



$$Loss\ Function:\ L(a,y) = -\left[\mathrm{y}\log(a) - (1-y)\log(1-a)\right]$$

$$Cost\ function: J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)}, y^{(i)})$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\left[\mathrm{y}^{(i)}.\log\left(a^{(i)}\right) - \left(1-y^{(i)}\right).\log(1-a^{(i)})\right]$$

So how do we make sure our cost function is minimum?

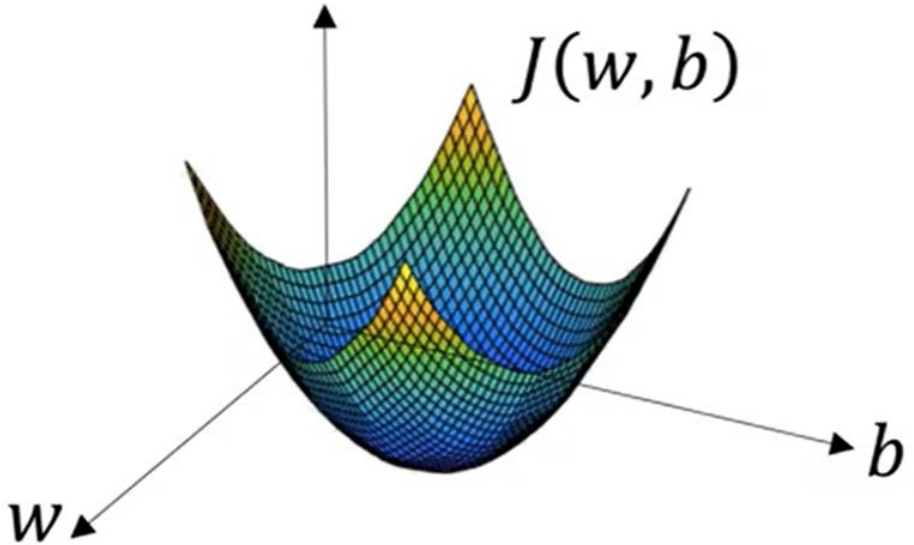
Neural Networks Mathematics

Forward/Back Propagation Calculations

Gradient Descent:

$$Loss\ Function\colon\ L(a,y) = -\left[\mathrm{y}\log(a) - (1-y)\log(1-a)\right]$$

$$Cost\ function\colon J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)}, y^{(i)}) \quad = \frac{-1}{m}\sum_{i=1}^{m}\left[\mathrm{y}^{(i)}.\log\left(a^{(i)}\right) - \left(1-y^{(i)}\right).\log(1-a^{(i)})\right]$$



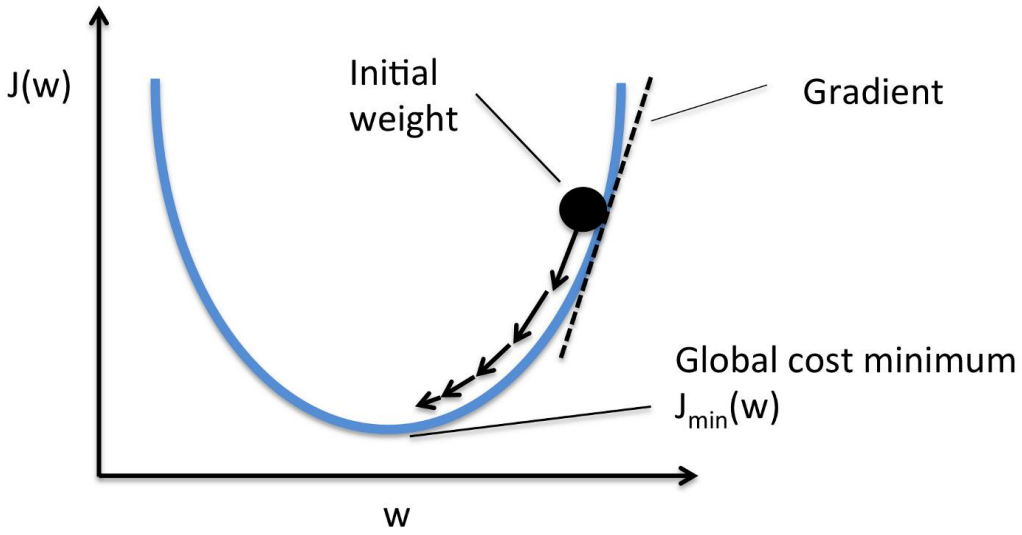
Neural Networks Mathematics

Forward/Back Propagation Calculations

Gradient Descent:

$$Loss\ Function\colon\ L(a,y) = -\left[\mathrm{y}\log(a) - (1-y)\log(1-a)\right]$$

$$Cost\ function\colon J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(a^{(i)}, y^{(i)}) \quad = \frac{-1}{m}\sum_{i=1}^{m}\left[\mathrm{y}^{(i)}.\log\left(a^{(i)}\right) - \left(1-y^{(i)}\right).\log(1-a^{(i)})\right]$$



$$w = w\ -\ \alpha\ \times \frac{dJ(w,b)}{dw}$$

$$b = b\ -\ \alpha\ \times \frac{dJ(w,b)}{db}$$

Class Exercise
Cell Tower Anomalies