Machine Learning Specialization: Arjunan K

Machine Learning Specialization by Andrew Ng in collaboration between DeepLearning.AI and Stanford Online in Coursera, Made by Arjunan K.

Supervised Machine Learning: Regression and Classification - Course 1

Intro to Machine Learning

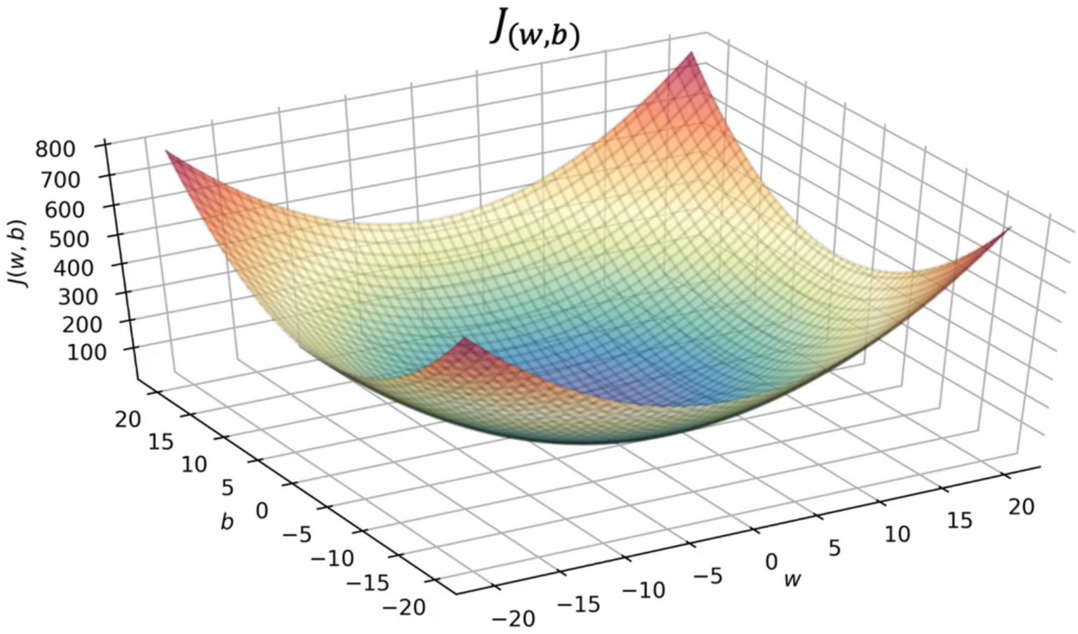
Machine Learning is the Ability of computers to learn without being explicitly programmed. There are different types of Machine Learning Algorithms.

1. Supervised Learning
2. Unsupervised Learning
3. Recommender Systems
4. Reinforcement Learning

Supervised Learning

Machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. It find a mapping function to map the input variable(x) with the output variable(y). Some use cases are given below,

- Spam Filtering
- Speech Recognition
- Text Translation
- Online Advertising
- Self-Driving Car
- Visual Inspection (Identifying defect in products)



Gradient Descent

Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks. Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates. Until the function is close to or equal to zero, the model will continue to adjust its parameters to yield the smallest possible error. We start with

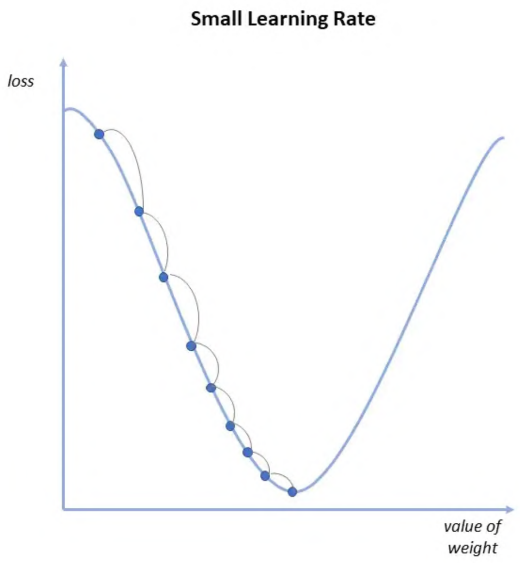
Normal Equation (Alternative for Gradient Descent)

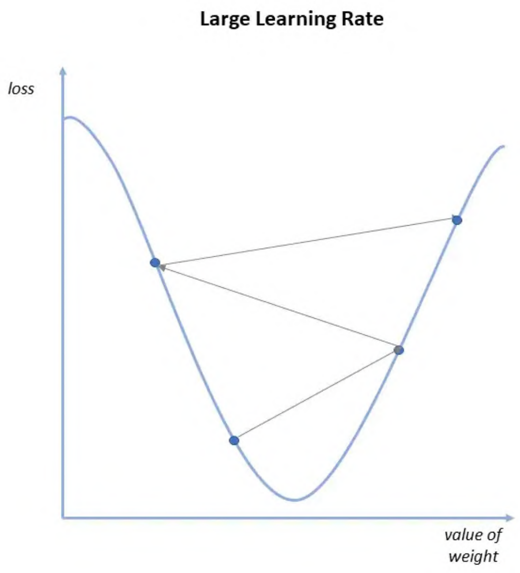
- Only for Linear Regression
- Solve w and b without iteration.
- But not a generalized one for other learning algorithm.
- when number of features is large > 1000, it is slow.
- most libraries use this under the hood.
- but gradient is better and general.

Learning Rate

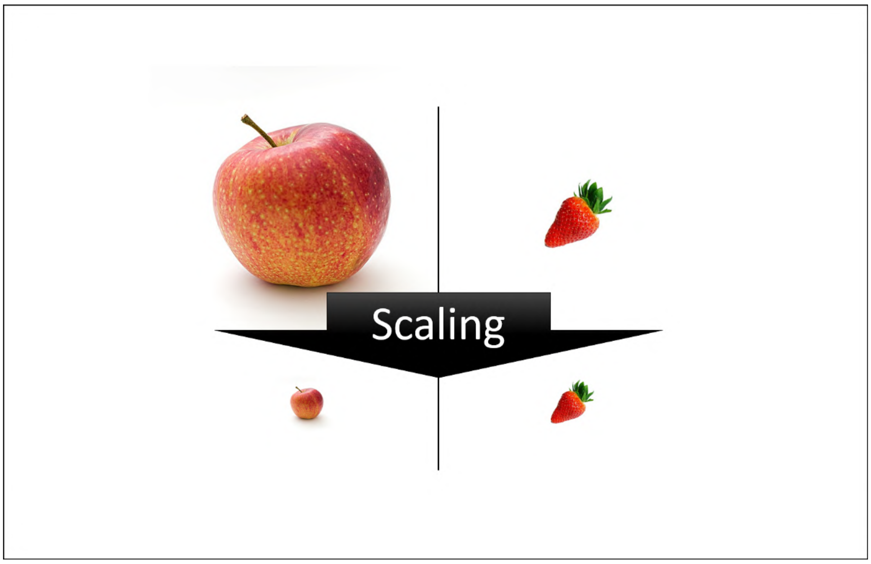
The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.

- Small learning rate may result slow gradient descent.
- Large learning rate may result in divergence of descent (Fail to converge to minimum).
- If the cost function reaches local minimum, slope become zero. So w = w, not going to descent after that.
- Fixed learning rate has no problem, since derivative part decrease as we decent.
- **Alpha** value can be 0.001 and increased 3X times based on requirement. If Cost function is increasing decrease **Alpha** and Vice Versa.





weight and 1 represents highest weight instead of representing the weights in kgs.



Types of Feature Scaling:

1. Standardization

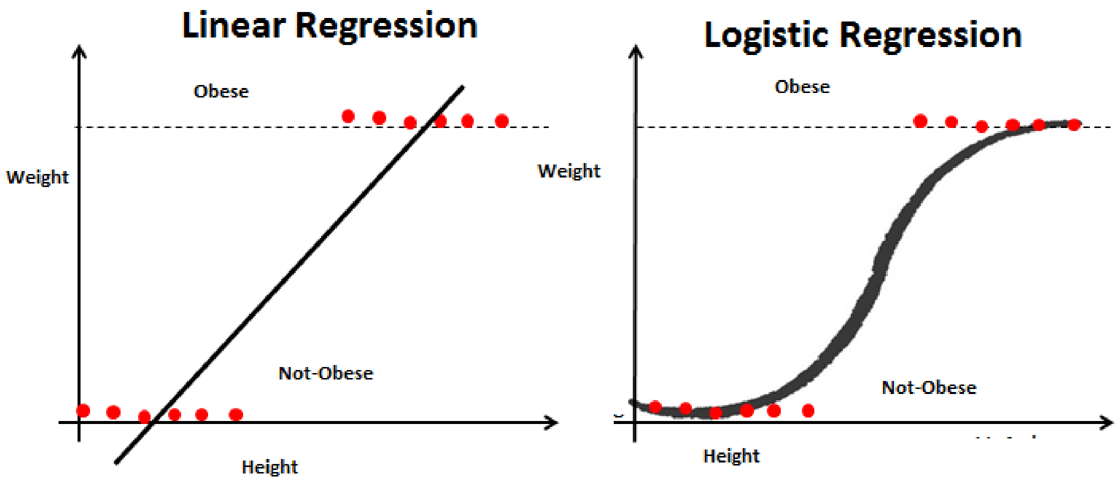
- Standard Scaler - (Z Score Normalization)

2. Normalization

- Min Max Scaling
- Max Absolute Scaling
- Mean Normalization
- Robust Scaling

Standardization (Standard Scaler)

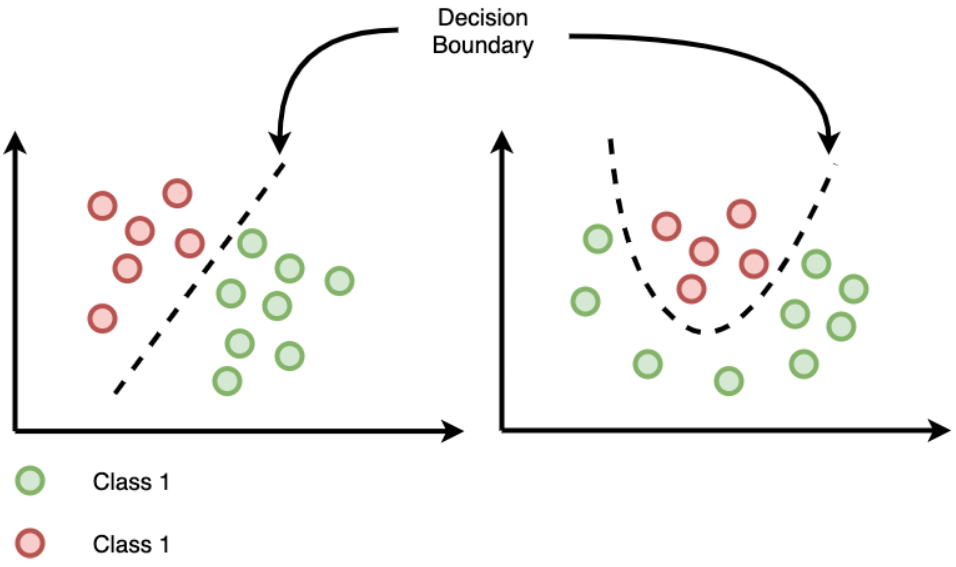
Standardization is a scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. Also known as Z Score Normalization.



Decision Boundary – Logistic Regression

- The line or margin that separates the classes.
- Classification algorithms are all about finding the decision boundaries.
- It need not be straight line always.
- For a logistic function **f(x)**, the **g(z)**, where **z = wx+b**,

  z = 0 gives the decision boundary. ie **wx+b = 0**



What is Log Loss?

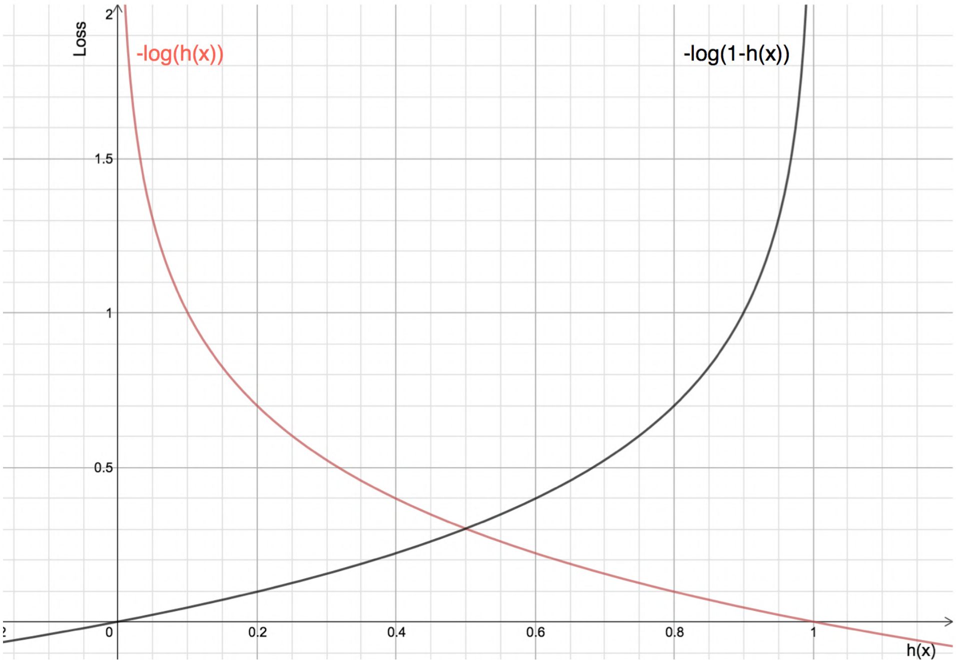
Log-loss is indicative of how close the prediction probability is to the corresponding actual/true value (0 or 1 in case of binary classification).

- Lower log loss value means better prediction
- Higher log loss means worse prediction

Equation of Log Loss Cost Function

$$LogLoss = -\frac{1}{n}\sum_{i=0}^{n}[y_i log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

- For Y = 0, Log Loss graph show low loss for y = 0 and high loss for y = 1
- For Y = 1, Log Loss graph show high loss for y = 0 and low loss for y = 1



Learning Curve, Vectorization, Feature Scaling all works same for Logistic Regression just like Linear Regression.

Types of Regularization Techniques

There are two main types of regularization techniques: L1(Lasso) and L2(Ridge) regularization

1) Lasso Regularization (L1 Regularization)

In L1 you add information to model equation to be the absolute sum of theta vector (θ) multiply by the regularization parameter (λ) which could be any large number over size of data (m), where (n) is the number of features.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{m} \sum_{j=1}^{n} |\theta_j|$$

2) Ridge Regularization (L2 Regularization)

In L2, you add the information to model equation to be the sum of vector (θ) squared multiplied by the regularization parameter (λ) which can be any big number over size of data (m), which (n) is a number of features.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Advanced Learning Algorithms - Course 2

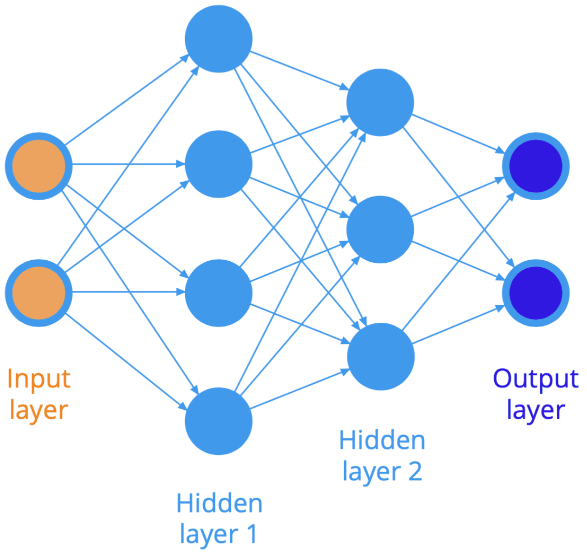
NEURAL NETWORKS

- Learning Algorithms that mimic the Human Brain.
- Speech Recognition, Computer Vision, NLP, Climate Change, Medical Image Detection, Product Recommendation.
- Just like human neuron, neural networks use many neurons to receive input and gives an output to next layer of network.

- As amount of data increased the performance of neural networks also increased.
- Layer is a grouping of neurons, which take input of similar features and output a few numbers together.
- Layer can have single or multiple neurons.
- If the layer is first, it is called Input Layer
- If the layer is last it is called Output Layer
- If the layer in middle, it is called Hidden Layer
- Each neurons have access to all other neurons in next layer.

Activations (a)

- Refers to the degree of high output value of neuron, given to the next neurons.
- Neural Networks learn it's own features. No need of manual Feature Engineering.



Demand Prediction of a Shirt?

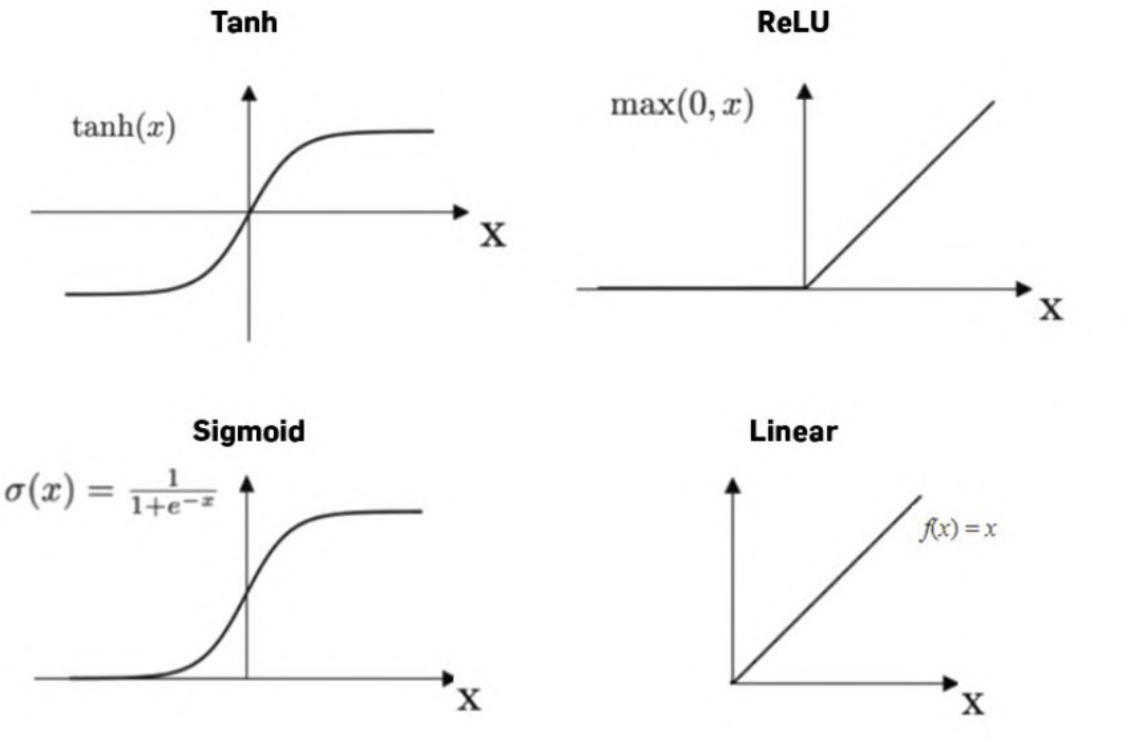
- affordability, awareness, perceived quality are activations of each neurons.
- It is basically a Logistic Regression, trying to learn much by it's own.
- activation, a = g(z) = 1 / (1 + e ^ -z)
- Number of layers and neurons are called Neural Network Architecture.

```
    [
        tf.keras.Input(shape=(2,)),
        Dense(3, activation='sigmoid', name = 'layer1'),
        Dense(1, activation='sigmoid', name = 'layer2')
     ]
)

model.compile(
    loss = tf.keras.losses.BinaryCrossentropy(),
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.01),
)

model.fit(
    Xt,Yt,
    epochs=10,
)
```

General Implementation of Forward Prop in Single Layer

- Uppercase for Matrix
- Lowercase for Vectors and Scalers

```
def my_dense(a_in, W, b, g):
    """
    Computes dense layer
    Args:
      a_in (ndarray (n, )) : Data, 1 example
      W    (ndarray (n,j)) : Weight matrix, n features per unit, j units
      b    (ndarray (j, )) : bias vector, j units
      g    activation function (e.g. sigmoid, relu..)
    Returns
      a_out (ndarray (j,))  : j units|
    """
    units = W.shape[1]
    a_out = np.zeros(units)
    for j in range(units):
        w = W[:,j]
        z = np.dot(w, a_in) + b[j]
        a_out[j] = g(z)
    return(a_out)

def my_sequential(x, W1, b1, W2, b2):
    a1 = my_dense(x,  W1, b1, sigmoid)
    a2 = my_dense(a1, W2, b2, sigmoid)
    return(a2)
```



Choose Activation Function

For Output Layer

- Binary Classification - Sigmoid
- Regression - Linear
- Regression Positive Only - ReLU

For Hidden Layer

- ReLU as standard activation

Why do we need activation functions?

- If we use linear activation function across all the neurons the neural network is no different from Linear Regression.
- If we use linear in hidden and sigmoid in output it is similar to logistic regression.
- **Don’t use linear in hidden layers. Use ReLU**
- We wont be able to fit anything complex

- Get more training data
- try small set of features
- try large set of features
- try adding polynomial features
- try decreasing and increasing learning rate and lambda (regularization parameter)
- we can use different diagnostic systems

Evaluating your Model

- Split the data training and testing.
- we can find training error and testing error
- cost function should be minimized
- MSE for Linear Regression
- Log Loss for Logistic Regression - Classification
- **For classification problem we can find what percentage of training and test set model which has predicted false, which is better than log loss**
- we can use polynomial regression and test the data for various degree of polynomial
- But to improve testing we can split data into training, cross validation set and test set
- Even for NN we can use this type of model selection
- Then we can find the cost function for each set

Bias and Variance - Model Complexity or High Degree Polynomial

- We need low bias and low variance
- Bias - Difference between actual and predicted value
- Variance - Perform well on training set and worse on testing set
- degree of polynomial should not be small or large. It should be intermediate

- Model with high bias pays very little attention to the training data and oversimplifies the model.
- It always leads to high error on training and test data.

VARIANCE

- Variance is the variability of model prediction for a given data point or a value which tells us spread of our data.
- Model with high variance pays a lot of attention to training data and does not generalize on test data.
- As a result, such models perform very well on training data but has high error rates on test data.

Overfitting - High Variance and Low Bias

Underfitting - High/Low Variance and High Bias

High Variance

- Get more training data - High Variance
- try increasing lambda - Overfit High Variance
- try small set of features - Overfit High Variance

High Bias

- try large set of features - Underfit High Bias
- try adding polynomial features - Underfit High Bias
- try decreasing lambda - Underfit High Bias

Bias and Variance Neural Networks

- Simple Model - High Bias
- Complex Model - High Variance
- We need a model between them, that is we need to find a trade off between them.

Neural Networks

- Large NN are having low bias machines

    - Original audio
    - noisy background - crowd
    - noisy background - car
    - audio on a bad cellphone connection

For Image Text Recognition we can make our own data by taking screenshot of different font at different color grade

- **Synthetic Data Creation** is also now a inevitable part of majority projects.
- Focus on Data not the code

What is Transfer Learning ?

Transfer learning make use of the knowledge gained while solving one problem and applying it to a different but related problem (same type of input like, image model for image and audio model for audio.

For example, knowledge gained while learning to recognize cars can be used to some extent to recognize trucks.

- We want to recognize hand written digits from 0 to 9
- We change the last output layer we wanted from the large NN all ready pre build.

Pre Training

When we train the network on a **large dataset(for example: ImageNet)** , we train all the parameters of the neural network and therefore the model is learned. It may take hours on your GPU.

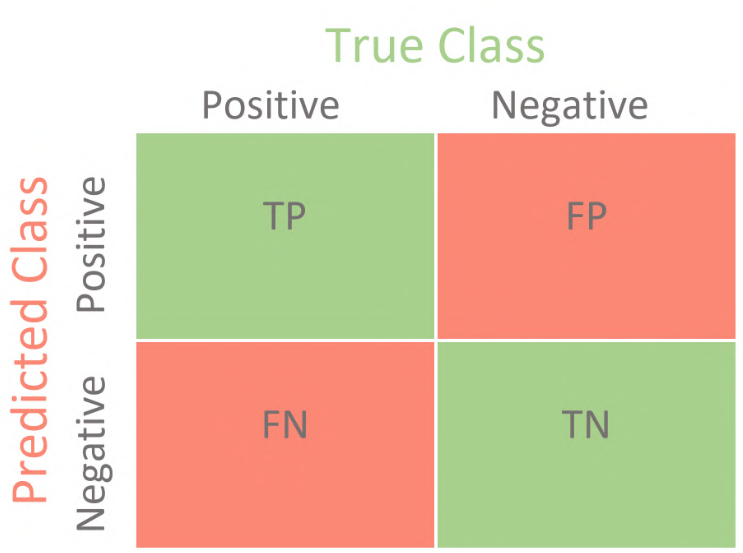
Error Metrics

- For classifying a rare disease, Accuracy is not best evaluation metrics
- Precision Recall and F1Score helps to measure the classification accuracy

Confusion Matrix in Machine Learning

Confusion Matrix helps us to display the performance of a model or how a model has made its prediction in Machine Learning.

Confusion Matrix helps us to visualize the point where our model gets confused in discriminating two classes. It can be understood well through a 2×2 matrix where the row represents the actual truth labels, and the column represents the predicted labels.



		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Accuracy

Simplest metrics of all, Accuracy. Accuracy is the ratio of the total number of correct predictions and the total number of predictions.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision

Precision is the ratio between the True Positives and all the Positives. For our problem statement, that would be the measure of patients that we correctly identify having a heart/rare

For some other models, like classifying whether a bank customer is a loan defaulter or not, it is desirable to have a high precision since the bank wouldn’t want to lose customers who were denied a loan based on the model’s prediction that they would be defaulters.

There are also a lot of situations where both precision and recall are equally important. For example, for our model, if the doctor informs us that the patients who were incorrectly classified as suffering from heart disease are equally important since they could be indicative of some other ailment, then we would aim for not only a high recall but a high precision as well.

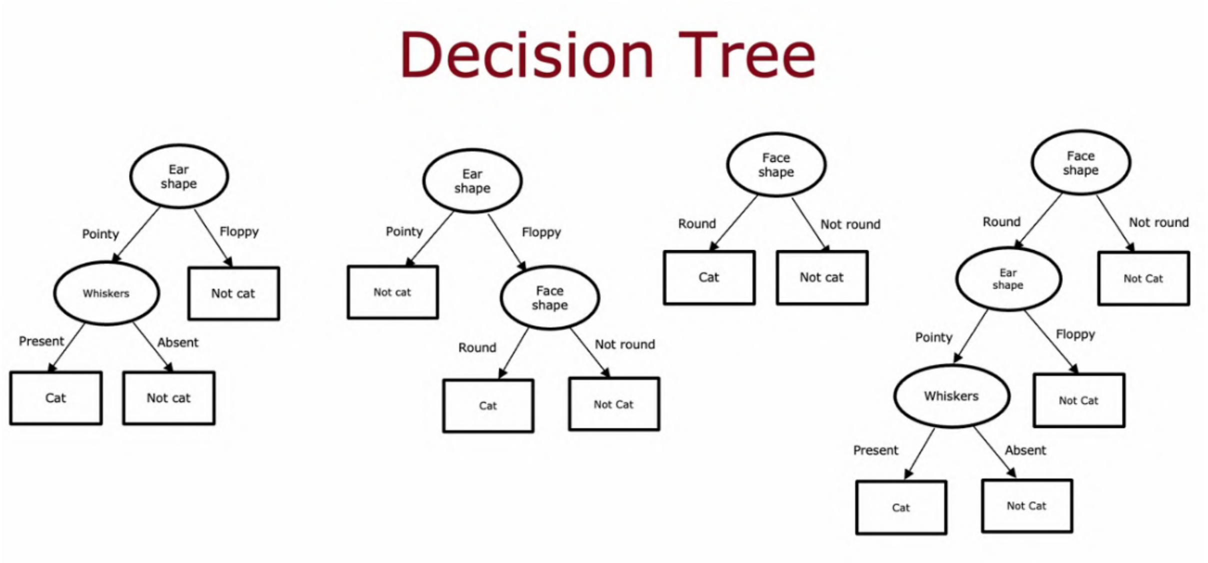
In such cases, we use something called F1-score is used . F1-score is the Harmonic mean of the Precision and Recall

Precision, Recall and F1 Score should be close to one, if it is close to zero then model is not working well. (General case)

$$F1\,Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Decision Tree Model

- If features are categorical along with a target, for a classification problem decision tree is a good model
- Starting of DT is called **Root Node**
- Nodes at bottom is **Leaf Node**
- Nodes in between them is **Decision Node**
- The purpose of DT is to find best tree from the possible set of tree

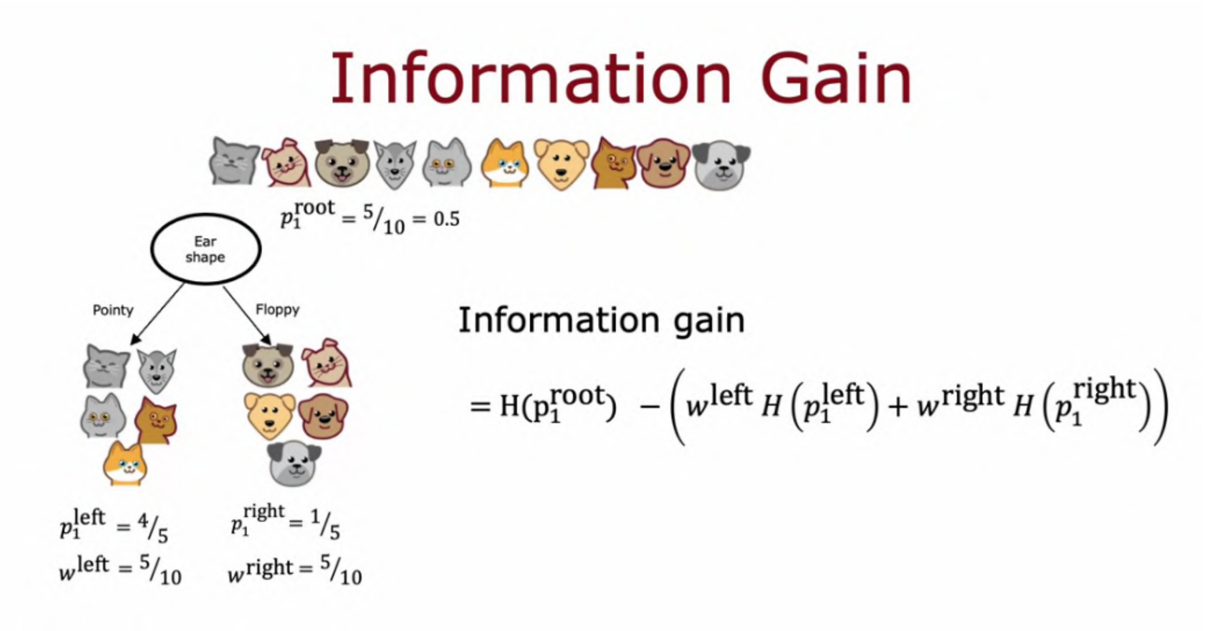


Decision Tree Learning Process

- If we reach at a node full of specific class, we stop there and set the node as leaf node.
- we want to achieve purity, that is class full of same type (Not completely possible all time).
- In order to maximize the purity, we need to decide where we need to split on at each node.

When to stop splitting?

- When node is 100% one class
- Reaching maximum depth of tree
- When depth increases - Overfitting
- When depth decreases - Underfitting
- when impurity score improvements are below a threshold
- Number of examples in a node is below a threshold



Information Gain

$p_1^{\text{root}} = 5/10 = 0.5$

$p_1^{\text{left}} = 4/5$ $\quad p_1^{\text{right}} = 1/5$

$w^{\text{left}} = 5/10$ $\quad w^{\text{right}} = 5/10$

Information gain

$$= H(p_1^{\text{root}}) - \left(w^{\text{left}} H\left(p_1^{\text{left}}\right) + w^{\text{right}} H\left(p_1^{\text{right}}\right)\right)$$

Putting it together

- Start with all examples at root node
- Calculate Information Gain for all features and pick one with large IG
- Split data into left and right branch
- keep repeating until,
  - Node is 100% one class
  - When splitting result in tree exceeding depth. Because it result in Overfitting
  - If information gain is less than a threshold
  - if number of examples in a node is less than a threshold

Decision Tree uses Recursive Algorithm

Using one-hot encoding of categorical features

- One hot encoding is a process of converting categorical data variables to features with values 1 or 0.
- One hot encoding is a crucial part of feature engineering for machine learning.
- In case of binary class feature we can covert in just one column with either 0 or 1
- Can be used in Logistic/Linear Regression or NN

```
# Classification
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

# Regression
from xgboost import XGBRegressor
model = XGBRegressor()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

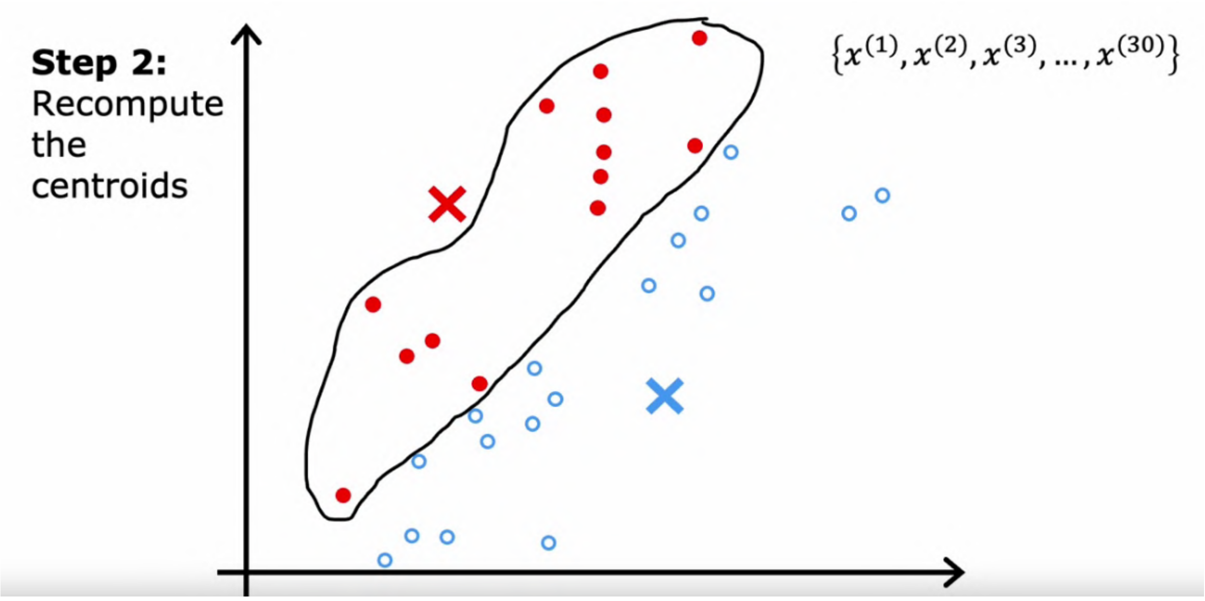
When to use Decision Trees

Decision Tree and Tree Ensembles	Neural Networks
Works well on tabular data	Works well on Tabular (Structured and Unstructured data)
Not recommended for Images, audio and text	Recommended for Image, audio, and text
fast	slower than DT
Small Decision tree may be human interpretable	works with transfer learning
We can train one decision tree at a time.	when building a system of multiple models working together, multiple NN can be stringed together easily. We can train them all together using gradient descent.

Unsupervised Learning, Recommenders, Reinforcement Learning - Course 3

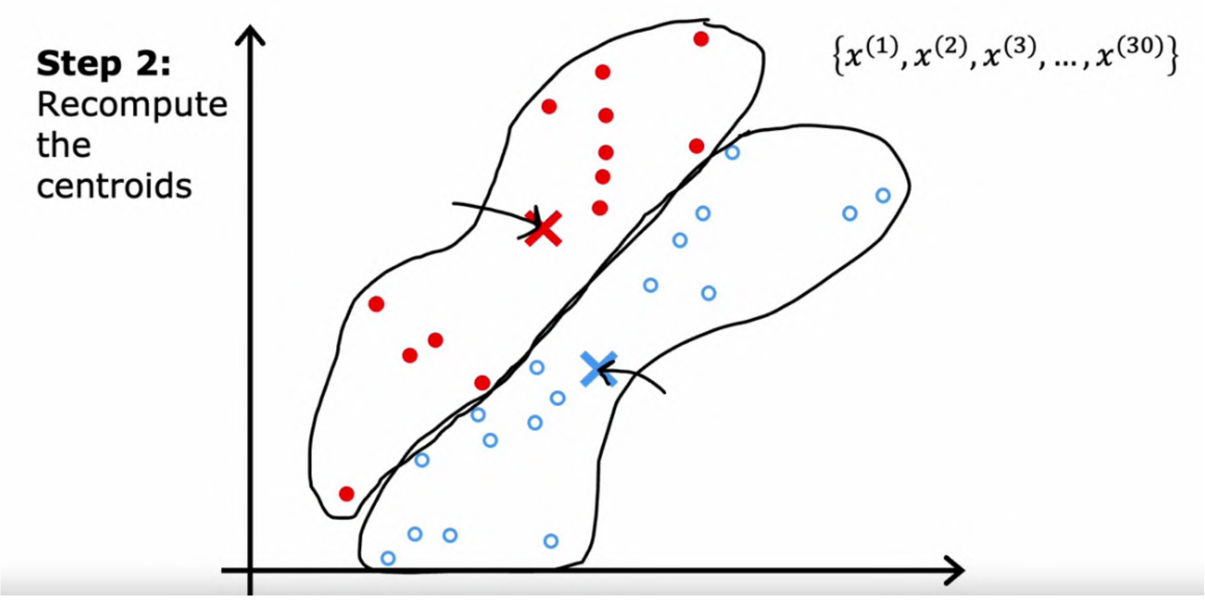
What is Clustering?

- Clustering is used to group unlabeled data
- There are various algorithms to perform the Clustering.



Step 2: Recompute the centroids

$\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(30)}\}$



Step 2: Recompute the centroids

$\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(30)}\}$

c. Top 20 movies in the country

2. Combined Retrieved items to list, removing duplicates, items already watched, purchased etc

Ranking

1. Take list retrieved and rank them based on learned model
2. Display ranked items to user

Retrieving more items results in better performance, but slower recommendations

To analyze that run it offline and check if recommendation, that is p(y) is higher for increased retrieval.

Ethical use of recommender systems

What is the goal of the recommender system?

- Movies most likely to be rated 5 stars by user
- Products most likely to be purchased

Illegal Things

- Ads most likely to be clicked on
- Products generating the largest profit
- Video leading to maximum watch time

- Now we need to combine both Vm and Vu
- So we take user_NN and item_NN as input layer of combined model
- Then take their product to make output
- Prepare the model using this NN
- Finally evaluate model by MSE cost function to train the model

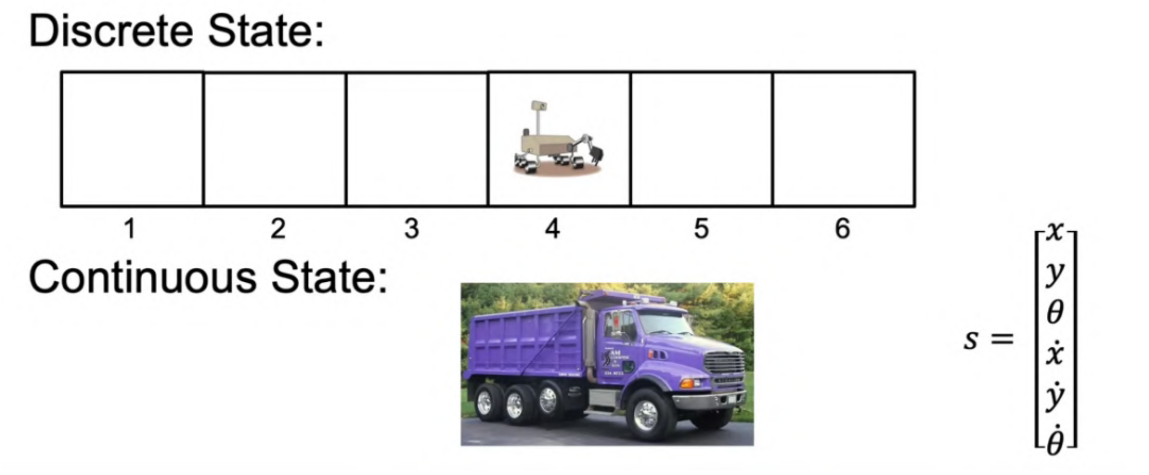
```
# create the user input and point to the base network
input_user = tf.keras.layers. Input (shape=(num_user_features))
vu = user_NN (input_user)
vu = tf.linalg.12_normalize (vu, axis=1)

# create the item input and point to the base network
input_item = tf.keras.layers. Input (shape=(num_item_features))
vm = item_NN (input_item)
vm = tf.linalg.12_normalize (vm, axis=1)

# measure the similarity of the two vector outputs
output = tf.keras.layers. Dot (axes=1) ([vu, vm])

# specify the inputs and output of the model
model = Model ([input_user, input_item], output)

# Specify the cost function
cost_fn= tf.keras.losses. Mean SquaredError ()
```

What is Reinforcement Learning?

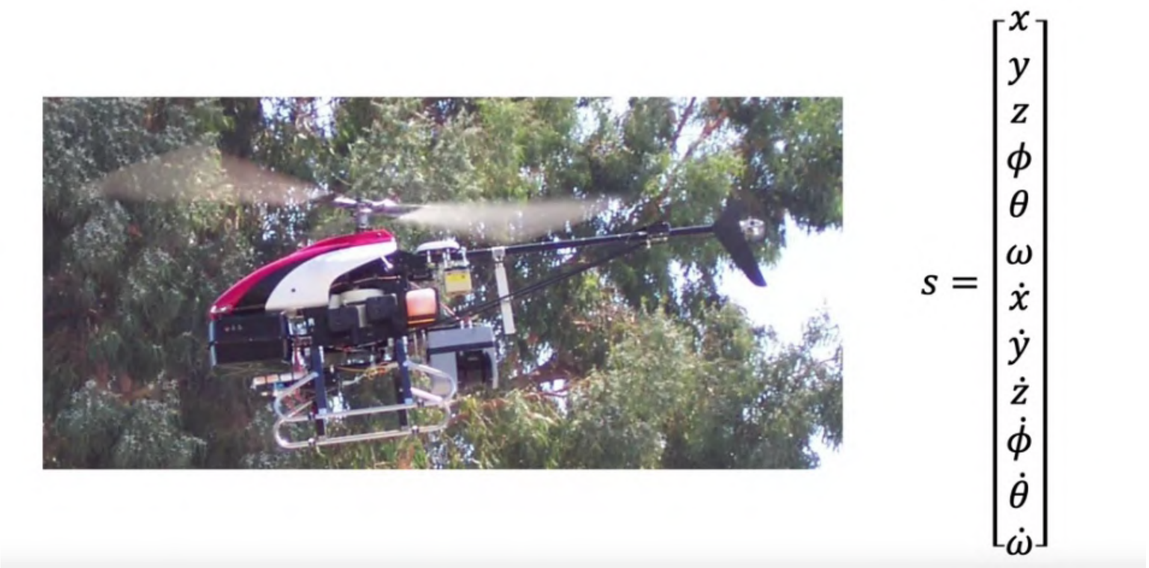
- Here we try to perform a action from it’s given state
- We can use the linear relation too, but it is not possible to get the dataset of the input x and output y
- So supervised learning won’t work here.
- Here we tell the model what to do. But not how to do
- We can make the mark the data as 1 if it performs well and -1 if it fails
- Algorithm automatically figure out what to do. We just need to say to the algorithm, if it is doing well or not using data input.

Applications

- To Fly a Helicopter
- Control Robots


Discrete State:
1
2
3
4
5
6
Continuous State:
$s = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$

- If we are controlling a helicopter, We have X, Y, Z axis and 3 angle between XY, YZ, XZ axis


$s = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \omega \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}$

Lunar Lander

It mainly has 4 actions to do

- Do nothing
- Left Thruster
- Right Thruster
- Main Thruster

We can represent them as a vector of X, Y and tilt Theta along with the binary value l and r which represent left or right leg in ground