Tensorflow

Basics of ML and TF

1. ML algorithm figures out specific patterns on the each set of data that determines the distinctiveness of each.
2. Keras - API from Tensorflow. Dense --> to define a layer of neurons in neural network. Sequential --> layers, since they are sequenced inputs.
3. Functions in TF:
    a. Loss Functions: NN has no idea between relationship of X and Y. To measure the accuracy of the guess of the relationship, Loss function is used. Ex: Mean squared error
    b. Optimizer: Helps in faster or efficient convergence. Reducing the loss function is done by an optimizer. Ex: Stochastic GD.
4. Numpy: Data representation in the form of lists.
5. Fit Function: Training happens here. Input to fit function is training set (both predictors and labels) i.e. X and Y and the no. of iterations, aka Epochs.
6. Predict Function: This predicts the output given the new input test data.
7. Sequential Class in Keras: Groups a linear stack of layers into tf.keras.Model class. It also provides training and inference features on this model. A sequential model is

a plain stack of layers where each layer has exactly one input tensor and one output tensor. It is defined by providing a list of layers to the sequential constructor The syntax:

```
tf.keras.Sequential(layers = none, name = none)
```

The sequential class has 2 methods add() and pop() to add and remove layers from the top of the stacked layers. add(layer) takes a layer instance. (ex: tf.keras.layer.Dense(units, input_size))

8. Basic components of NN:
    a. Unit: The amount of neurons in a layer.
    b. Shapes: Tuples representing how many elements does the array/tensor has in every dimension. For example, shape = (10,2,3) means, the tensor has 10 elements in the first dimension, 2 in the second and 3 in the third, thereby having 10*2*3 = 60 elements all together.
    c. Input Shape: (Needs to be explicitly defined, based on your training data) What flows in between the layers are tensors. In keras, the input layer is not a layer but a tensor, whose shape is equal to your input training set. So if you have 30 images of 50x50 pixels in 3 channels (RGB), then your input tensor size would be (30,50,50,3). Every kind of layer needs different way of taking dimensions. For example, **Dense** layer takes input in the form, Dense(batch_size, input_size) where, batch size is the number of neurons in the layer and input size is the incoming size of the data coming to that layer. In input shape, the batch size is the no. of training rows or elements in your data.
    d. Output Shape: The "units" of each layer will define the output shape (the shape of the tensor that is produced by the layer and that will be the input of the next layer). A Dense layer has an output shape of Dense(batch_size, units)
    e. dim: If the input data has only 1 dimension, you can give the input_dim as scalar value. Or if you have to provide an input_shape = (3,) which means 1 dimension with 3 elements. BUt when we talk about Tensor dimensions, if a tensor has shape (12, 5), the tensor has 2 dimensions.

9. Dense Class: regular densely-connected NN layer. Dense function implements the following:

> 💡 output = activation(dot(input, kernel) + bias)
>
> activation —> element wise activation function passed
>
> kernel —> matrix of the weights created by the layer
>
> bias —> bias vector created by the layer (only if use_bias = True)

10. Find the simple NN model trained to predict a function here.