ANN REGRESSION INTERNSHIP WEEK-2

Building an artificial neural network (ANN) regression model involves several steps:

1. Import TensorFlow: Start by importing the TensorFlow library, which will be used to buildmodel

Import tensorflow as tf

1. Define the model architecture: The next step is to define the architecture of the model. In TensorFlow, you can use the **Sequential** class from the **tf.keras** module to define a sequential model. Then, you can add layers to the model using the **add** method.

For an ANN regression model, you can start with a single dense layer that has a single unit and takes inputs of the same shape as your input data. You can use the **Dense** layer from the **tf.keras.layers** module to create this layer

model = tf.keras.Sequential()

model.add(tf.keras.layers.Dense(units=1, input\_shape=[1]))

1. Compile the model: After defining the architecture of the model, you need to compile it by specifying a loss function and an optimizer. For regression problems, you can use the mean squared error (MSE) loss function, which measures the difference between the true values and the predictions made by the model.

For the optimizer, you can use any of the available optimizers in TensorFlow, such as stochastic gradient descent (SGD), Adam, or Adagrad. In this example, we'll use the SGD optimizer with a learning rate of 0.001.

model.compile(optimizer=tf.keras.optimizers.SGD(learning\_rate=0.001),

loss='mean\_squared\_error')

1. Train the model: The next step is to train the model on your data. You can use the **fit** method of the model to train it for a specified number of epochs on your input and output data.

xs = [0, 1, 2, 3, 4, 5]

ys = [1.0, 1.5, 2.0, 2.5, 3.0, 3.5]

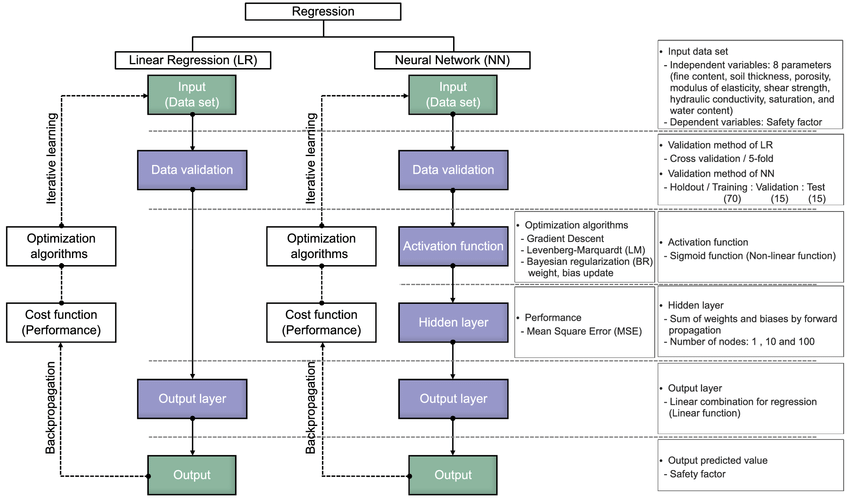
model.fit(xs, ys, epochs=500)

1. Make predictions: Once the model is trained, you can use it to make predictions on new data. You can use the **predict** method of the model to make predictions for a given input.

Print(model.predict([10.0]))

Here's a flow chart that summarizes the steps for building an artificial neural network (ANN) regression model using TensorFlow:

1. Import TensorFlow library
2. Define the model architecture a. Initialize a sequential model b. Add a dense layer with 1 unit and input shape
3. Compile the model a. Specify a loss function (e.g., mean squared error) b. Specify an optimizer (e.g., stochastic gradient descent)
4. Train the model on input and output data
5. Use the model to make predictions on new data

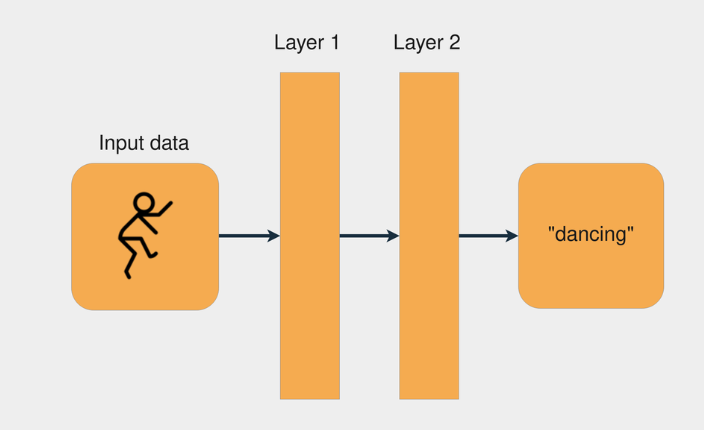


**Neural Networks: Main Concepts**

A neural network is a system that learns how to make predictions by following these steps:

1. Taking the input data
2. Making a prediction
3. Comparing the prediction to the desired output
4. Adjusting its internal state to predict correctly the next time

Each layer transforms the data that comes from the previous layer. One cool thing about neural network layers is that the same computations can extract information from any kind of data. This means that it doesn’t matter if you’re using image data or text data. The process to extract meaningful information and train the deep learning model is the same for both scenarios.

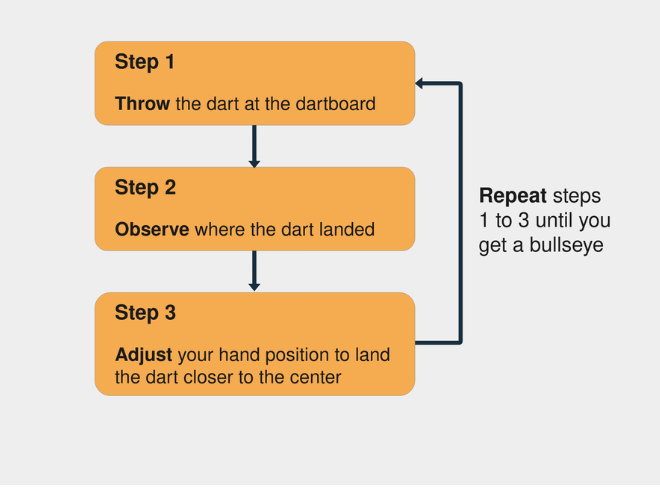


Each layer transforms the data that came from the previous layer by applying some mathematical operations.

### **The Process to Train a Neural Network**

Training a neural network is similar to the process of trial and error. Imagine you’re playing darts for the first time. In your first throw, you try to hit the central point of the dartboard. Usually, the first shot is just to get a sense of how the height and speed of your hand affect the result. If you see the dart is higher than the central point, then you adjust your hand to throw it a little lower, and so on.

These are the steps for trying to hit the center of a dartboard:



With neural networks, the process is very similar: you start with some random **weights** and **bias** vectors, make a prediction, compare it to the desired output, and adjust the vectors to predict more accurately the next time. The process continues until the difference between the prediction and the correct targets is minimal.

Knowing when to stop the training and what accuracy target to set is an important aspect of training neural networks, mainly because of [overfitting and underfitting](https://realpython.com/linear-regression-in-python/#underfitting-and-overfitting) scenarios.