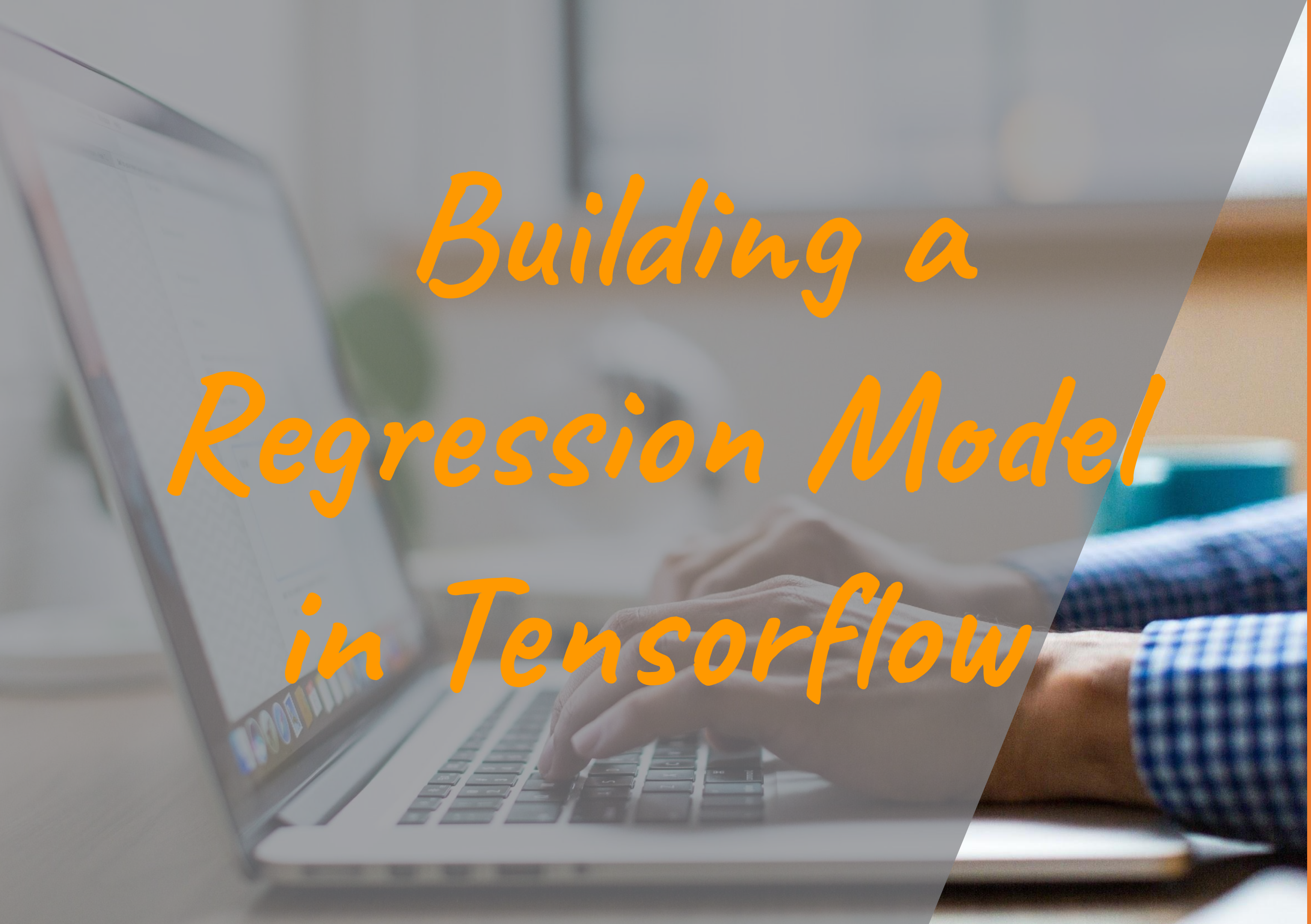
Building a Regression Model in Tensorflow



AGENDA OF THE VIDEO

- An Overview of neural networks.
- Types of layers in a neural networks.
- Types of activation function in a neural networks.
- Introduction to model building library called "Keras".
- Alternative to Keras.
- Keras Model Building Procedure.
    - Make the architecture of Neural Network.
    - Compile the model with optimizers, loss function and metrics
    - Fit (Train) the Model
    - Predict on a new dataset
- The dataset that we will be working with...
- What is data scaling? Why is it required before building Neural Networks?
- Pseudocode for Regression in Tensorflow with keras.
- Task for you....

An Overview of neural networks

Neural Networks is a technique of modelling the interconnection of neurons in a human brain.

Building Neural Networks from scratch is a huge task. There are many concepts that needs to be taken care and still things can go wrong while coding the neural networks from scratch. Luckily for us, there are many people who are doing wonderful job by making neural network libraries and making it open source.

One such library is tensorflow which is one of most popular and powerful library for doing deep learning.

So, let us get started.

Layers in a neural networks

First type of classification...

- Input Layer = For regression, it would be a list of features.
- Output Layer = For regression, it would be a single number as prediction.
- Hidden Layers, can be any number of hidden layers, the more the number, the more complex patterns the network will be able to learn.

Another Classification of layers:

- Dense, fully connected layers = Common to all types of tasks
- Convolution layers = Common to Image and Video Related Tasks
- Long short term memory (LSTM) layers = Common to Sequence data types (such as Time series, Stock data, Language data etc)

Activations Functions

Activations functions are what allow us to take the data of one layers and transfer it to the next layers.

Further, these functions are what allow the network to behave somewhat (not exactly) like human brain.

Human brain has billions of neurons. All these neurons work in a cohesiveness to by transferring and receiving data from other neurons.

Which neurons will take part in one decision making? How is this decided?

If a certain neurons fires, then it will take part in the decision while if doesn’t, then it won’t. IT WAS A VERY HIGH LEVEL OVERVIEW OF NEURONS.

Intro to Model Building Library Called Keras

Keras is one of the API which is provided when you install tensorflow and it a very high level wrapper and makes really easy to build neural networks of any kind:

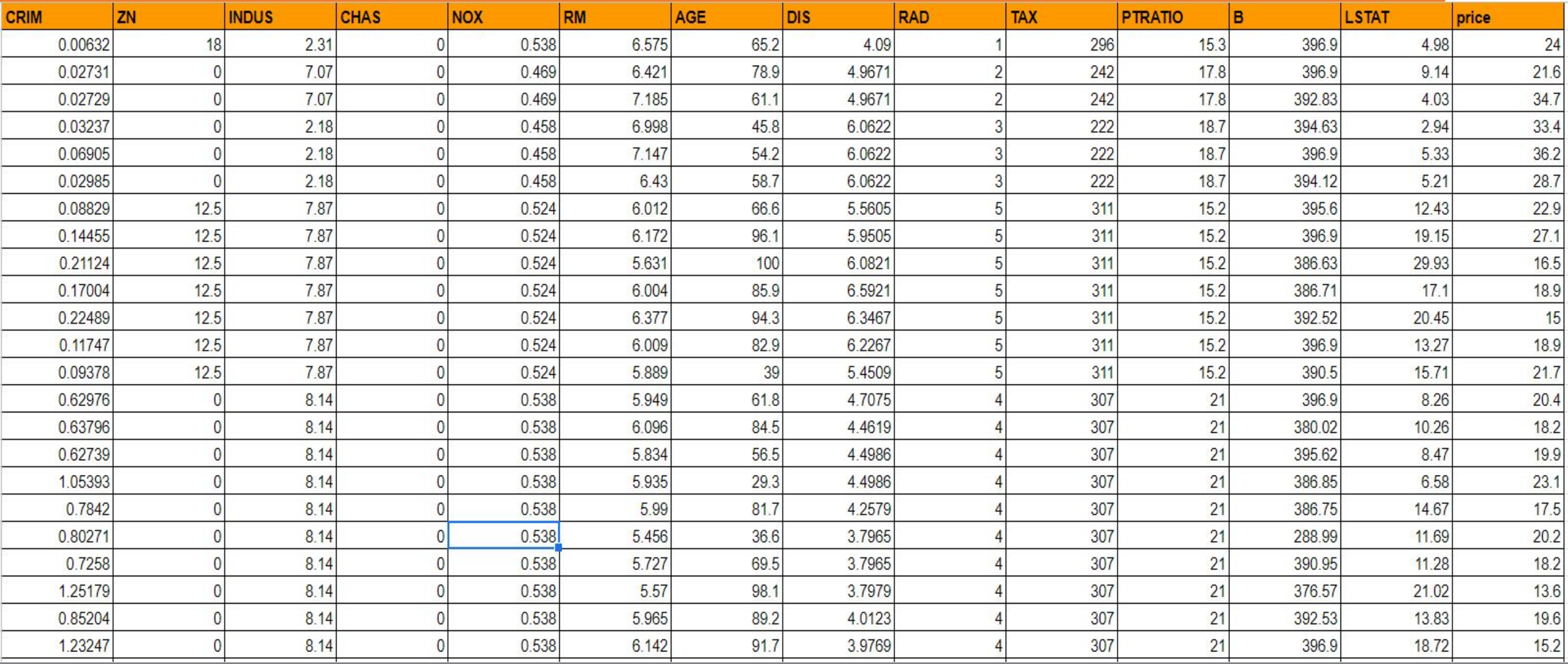
- Perceptrons
- Simple Neural Networks with as many hidden layers
- CNNs
- RNNs
- LSTM + RNNs
- Etc

Adding layers in a model is as easy as adding two numbers in Keras.

Model Building in Keras

1. Make the model architecture using the various combination of layers
   a. Sequential API
   b. Functional API
2. Compile the Model once the architecture is ready. This step is needed for telling the model about the type of optimizer, loss and metric.
3. Train the model using the architecture and wait until it runs.
4. Once the training process is completed, use your model to make predictions.

Dataset



CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1
0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67	17.5
0.80271	0	8.14	0	0.538	5.456	36.6	3.7965	4	307	21	288.99	11.69	20.2
0.7258	0	8.14	0	0.538	5.727	69.5	3.7965	4	307	21	390.95	11.28	18.2
1.25179	0	8.14	0	0.538	5.57	98.1	3.7979	4	307	21	376.57	21.02	13.6
0.85204	0	8.14	0	0.538	5.965	89.2	4.0123	4	307	21	392.53	13.83	19.6
1.23247	0	8.14	0	0.538	6.142	91.7	3.9769	4	307	21	396.9	18.72	15.2

Data Scaling

Data Scaling is one of the many steps that we follow before actually modelling our data.

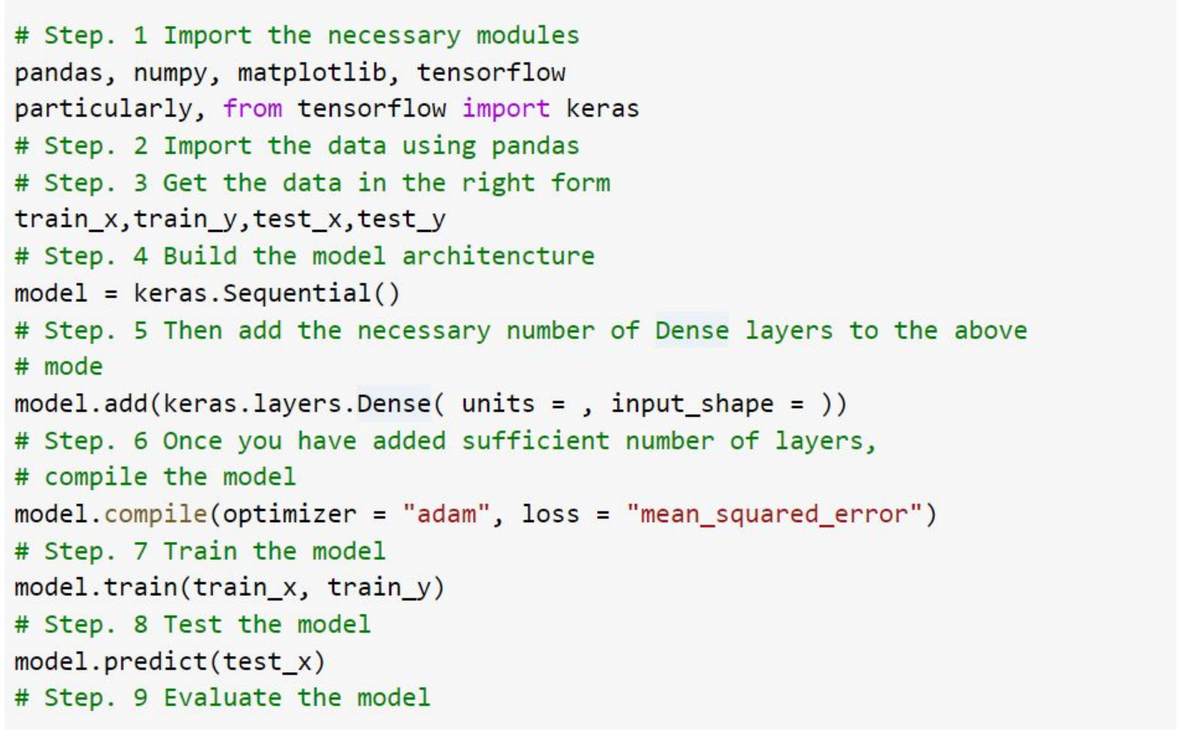
It becomes even more important when we have to train a really huge neural network.

First what is data scaling?

Data scaling is a procedure which is followed to bring our data ranges in a permissible range. Say, I have a variable income. Some people might be quite rich having income close to 1 billion dollars while some might just have in 10000 dollars. In this situation, what we do is we perform the step of data scaling to bring our data in some low data ranges.

The reason data scaling is important is because in optimization problems when the model is trying to figure the best values of weights, very large values can distort the model. It might make the model overshoot the optimum values or just make the model learn weights really slow.

Pseudocode



```
# Step. 1 Import the necessary modules
pandas, numpy, matplotlib, tensorflow
particularly, from tensorflow import keras
# Step. 2 Import the data using pandas
# Step. 3 Get the data in the right form
train_x,train_y,test_x,test_y
# Step. 4 Build the model architencture
model = keras.Sequential()
# Step. 5 Then add the necessary number of Dense layers to the above
# mode
model.add(keras.layers.Dense( units = , input_shape = ))
# Step. 6 Once you have added sufficient number of layers,
# compile the model
model.compile(optimizer = "adam", loss = "mean_squared_error")
# Step. 7 Train the model
model.train(train_x, train_y)
# Step. 8 Test the model
model.predict(test_x)
# Step. 9 Evaluate the model
```

Task for you..

We have just successfully fitted a multilayered neural network model on our boston price prediction data. Now, your task would be to execute the following:

1. Experiment with adding as many number of hidden layers as you want to add. Compare the results you get and compare the time it takes to get those results.
2. Experiment with adding more neurons to each hidden layers. See, whether it improves or makes your model poor.
3. Change the number of epochs. How does that affect the training time and accuracy?
4. Also, try to play with batch_size and see if it has any impact on time and performance.
5. Finally, make a neural network on your own data, any data. Just make it run.

Thank you