C1W1_Assignment

February 26, 2023

1 Week 1 Assignment: Housing Prices

In this exercise you'll try to build a neural network that predicts the price of a house according to a simple formula.

Imagine that house pricing is as easy as:

A house has a base cost of 50k, and every additional bedroom adds a cost of 50k. This will make a 1 bedroom house cost 100k, a 2 bedroom house cost 150k etc.

How would you create a neural network that learns this relationship so that it would predict a 7 bedroom house as costing close to 400k etc.

Hint: Your network might work better if you scale the house price down. You don't have to give the answer 400…it might be better to create something that predicts the number 4, and then your answer is in the 'hundreds of thousands' etc.

```
[ ]: # IMPORTANT: This will check your notebook's metadata for grading.
     # Please do not continue the lab unless the output of this cell tells you to␣
      ↪proceed.
     !python add_metadata.py --filename C1W1_Assignment.ipynb
```

NOTE: *To prevent errors from the autograder, you are not allowed to edit or delete non-graded cells in this notebook . Please only put your solutions in between the* ***### START CODE HERE*** *and* ***### END CODE HERE*** *code comments, and also refrain from adding any new cells.* ***Once you have passed this assignment*** *and want to experiment with any of the non-graded code, you may follow the instructions at the bottom of this notebook.*

```
[ ]: # grader-required-cell

     import tensorflow as tf
     import numpy as np
```

```
[ ]: # grader-required-cell

     # GRADED FUNCTION: house_model
     def house_model():
         ### START CODE HERE
```

```
    # Define input and output tensors with the values for houses with 1 up to 6␣
↪bedrooms
    # Hint: Remember to explictly set the dtype as float
    xs = None
    ys = None

    # Define your model (should be a model with 1 dense layer and 1 unit)
    model = None

    # Compile your model
    # Set the optimizer to Stochastic Gradient Descent
    # and use Mean Squared Error as the loss function
    model.compile(optimizer=None, loss=None)

    # Train your model for 1000 epochs by feeding the i/o tensors
    model.fit(None, None, epochs=None)

    ### END CODE HERE
    return model
```

Now that you have a function that returns a compiled and trained model when invoked, use it to get the model to predict the price of houses:

```
[ ]: # grader-required-cell

     # Get your trained model
     model = house_model()
```

Now that your model has finished training it is time to test it out! You can do so by running the next cell.

```
[ ]: # grader-required-cell

     new_x = 7.0
     prediction = model.predict([new_x])[0]
     print(prediction)
```

If everything went as expected you should see a prediction value very close to 4. **If not, try adjusting your code before submitting the assignment.** Notice that you can play around with the value of `new_x` to get different predictions. In general you should see that the network was able to learn the linear relationship between `x` and `y`, so if you use a value of 8.0 you should get a prediction close to 4.5 and so on.

Congratulations on finishing this week's assignment!

You have successfully coded a neural network that learned the linear relationship between two variables. Nice job!

Keep it up!

Please click here if you want to experiment with any of the non-graded code.

Important Note: Please only do this when you've already passed the assignment to avoid problems with the autograder.

On the notebook's menu, click "View" > "Cell Toolbar" > "Edit Metadata"

Hit the "Edit Metadata" button next to the code cell which you want to lock/unlock

Set the attribute value for "editable" to:

"true" if you want to unlock it

"false" if you want to lock it

```
    </li>
    <li> On the notebook's menu, click "View" > "Cell Toolbar" > "None" </li>
</ol>
<p> Here's a short demo of how to do the steps above:
    <br>
    <img src="https://drive.google.com/uc?export=view&id=14Xy_Mb17CZVgzVAgq7NCjMVBvSae3xO1" al
```