PMDS603P Deep Learning Lab Experiment 6

August 2025

1 Work to do today

Note: Make a single PDF file of the work you are doing in a Jupyter notebook. Upload with the proper format. Please mention your name and roll no properly with the Experiment number on the first page of your submission.

Load the Boston Housing dataset available in Keras. The dataset contains 13 numerical features about houses (crime rate, average rooms, property tax, etc.) and the target is the median house price in $1000s.

```
from tensorflow.keras.datasets import boston_housing

(x_train, y_train), (x_test, y_test) = boston_housing.load_data()
```

Answer the following
Question1. Building and Training a Neural Network

Load the Boston Housing dataset using Keras.
Build a simple feedforward neural network with:
Two hidden layers (choose appropriate number of neurons in each).
Use ReLU activation for hidden layers.
Use activation = 'linear' for the output layer (since it's regression).
Use SGD with momentum as the optimizer.
Use Mean Squared Error (MSE) as the loss function.
Include early stopping as well in your model.
Train the model on the training data and evaluate performance on the test data.

Question 2: Hyperparameter Tuning

Perform hyperparameter tuning for the following hyperparameters:
Number of neurons in each hidden layer.
activation function in each hidden layer.
Learning rate of the optimizer.
Momentum parameter in SGD.

Report the best set of hyperparameters that improves performance.

This way inside the build_model function you can incorporate

```
learning_rate = hp.Choice('learning_rate', values=[0.001, 0.01, 0.05])
momentum = hp.Choice('momentum', values=[0.0, 0.5, 0.9])

optimizer = SGD(learning_rate=learning_rate, momentum=momentum)
model.compile(optimizer=optimizer, loss='mse', metrics=['mse'])
```

to add choices for momentum parameter and learning_rate. Same way you can add for activation functions also in the hidden layers.

Question 3: Optimizer Comparison

Try at least three optimizers (e.g., SGD with momentum, RMSprop, Adam).
Train the same model architecture with each optimizer.
Compare their performances (using test MSE).
Report which optimizer gives the best results.
This can be incorporated in the build_model function in tuning part to accomplish the same.

```
optimizer_choice = hp.Choice('optimizer', ['SGD', 'RMSprop', 'Adam'])

if optimizer_choice == 'SGD':
    optimizer = SGD(
        learning_rate=hp.Choice('lr', [1e-2, 1e-3, 1e-5]),
        momentum=hp.Choice('momentum', [0.8, 0.9, 0.95, 0.99])
    )
elif optimizer_choice == 'RMSprop':
    optimizer = RMSprop(
        learning_rate=hp.Choice('lr_rms', [1e-2, 1e-3, 1e-5])
    )
else:
    optimizer = Adam(
        learning_rate=hp.Choice('lr_adam', [1e-2, 1e-3, 1e-5])
    )

model.compile(optimizer=optimizer, loss='mse', metrics=['mse'])
return model
```