Will Spaeth
2-7-20

Homework 1

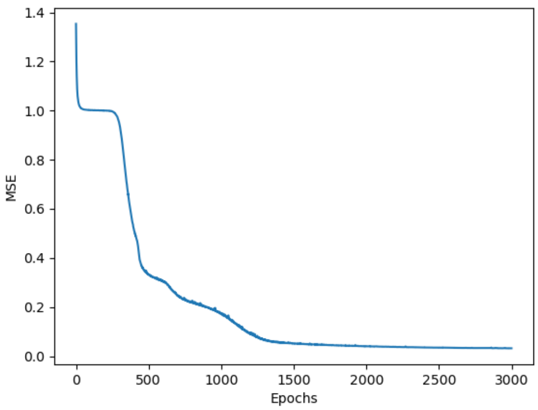
./hw1_script.py

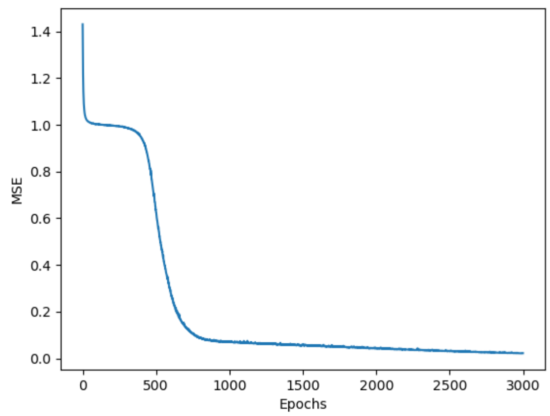
```
#!/usr/bin/env python3

import pickle
import os

import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
import numpy as np

def main():

   # Load data
   # Ins: (256, 8), float64
   # Outs: (256, 1), float64
   with open("hw1_dataset.pkl", "rb") as fp:
      hw1_dataset = pickle.load(fp)

   # Create & train 10 different models
   prediction_errors = []
   for i in range(10):

      # Build & compile model
      model = dnn(hidden_sizes=[8, 4, 4], hidden_act="elu", output_act="linear")
      model.compile(optimizer="adam", loss="mse")
      model.summary()

      # Train model
      model.fit(
            x=hw1_dataset["ins"],
            y=hw1_dataset["outs"],
            epochs=3000,
            batch_size=32
            )

      # Predict and calculate absolute error
      predictions = model.predict(hw1_dataset["ins"])
      prediction_errors.append( np.squeeze(np.abs(hw1_dataset["outs"] - predictions)) )


      # Plot learning curve for this model
      plot_learning_curve(i, model.history)

   # Compute the absolute prediction errors for all runs, combine the data and generate a histogram of the
absolute errors
   prediction_errors = np.concatenate(prediction_errors, axis=0)
```

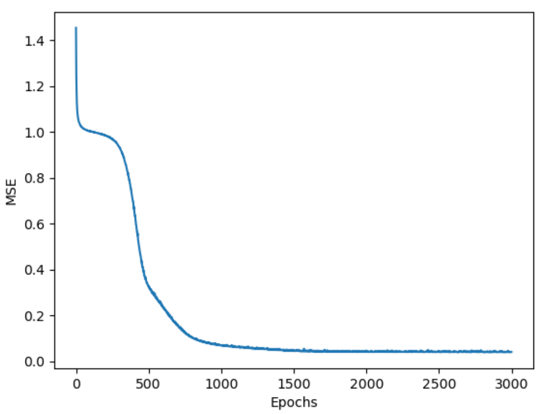
```
    create_histogram(prediction_errors)

def dnn(hidden_sizes, hidden_act="sigmoid", output_act="tanh"):
    """Construct a simple deep neural network"""

    inputs = Input(shape=(8,))

    hidden_stack_out = hidden_stack(hidden_sizes, hidden_act)(inputs)

    outputs = Dense(1, activation=output_act)(hidden_stack_out)

    return Model(inputs=inputs, outputs=outputs)

def hidden_stack(hidden_sizes, hidden_act="sigmoid"):
    """Represents a stack of neural layers"""

    layers = []
    for size in hidden_sizes:
        layers.append(Dense(size, activation=hidden_act))

    def hidden_stack_layer(inputs):
        """Layer hook for stack"""

        for i in range(len(layers)):
            if i == 0:
                carry_out = layers[i](inputs)
            else:
                carry_out = layers[i](carry_out)

        return carry_out

    return hidden_stack_layer

def create_histogram(prediction_errors):

    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    ax.hist(prediction_errors, 50)
    plt.ylabel("Count")
    plt.xlabel("Error")

    fig.savefig("error_histogram.png", dpi=fig.dpi)

def plot_learning_curve(experiment_num, history):

    save_path = "learning_curves/"
    if not os.path.exists(save_path):
        os.mkdir(save_path)

    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    ax.plot(history.history["loss"])
    plt.ylabel("MSE")
```

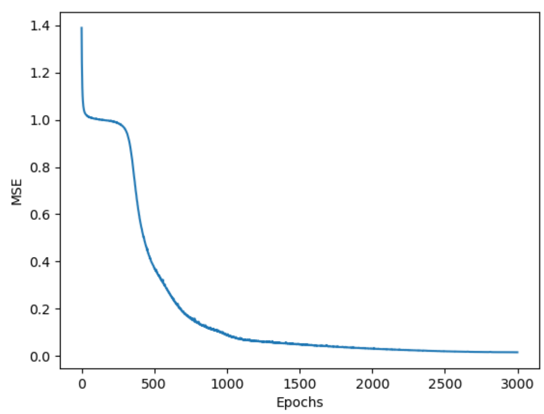
```
    plt.xlabel("Epochs")

    fig.savefig(save_path + f"experiment_{experiment_num}.png", dpi=fig.dpi)


if __name__ == "__main__":
    main()
```

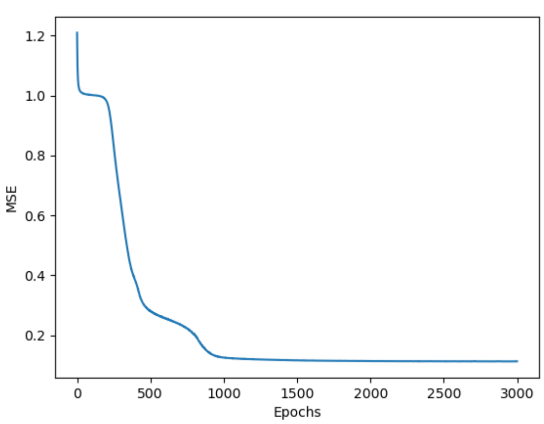
Resulting 10 Learning Curves

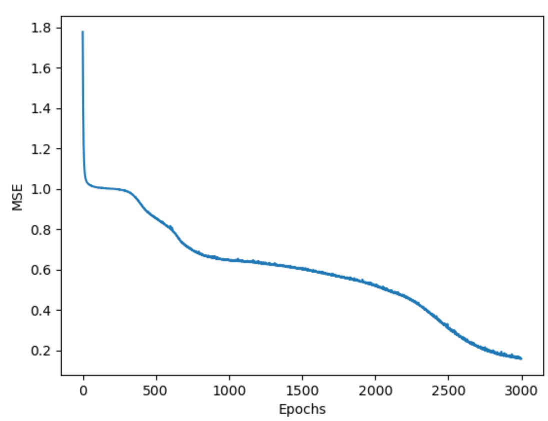


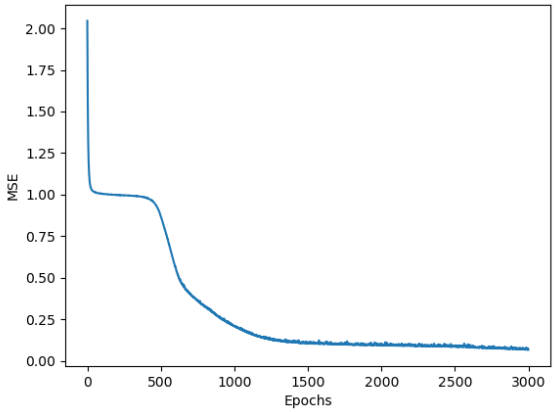


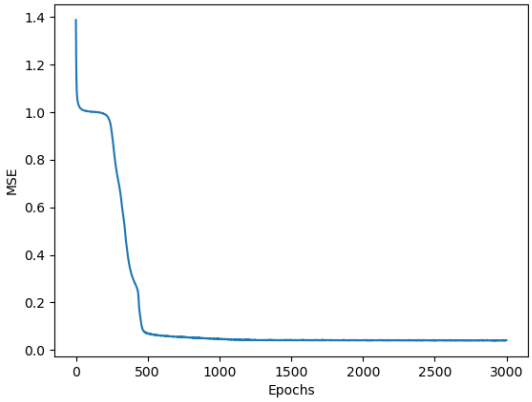


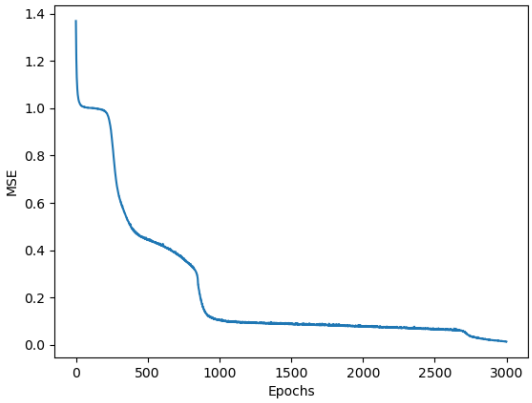


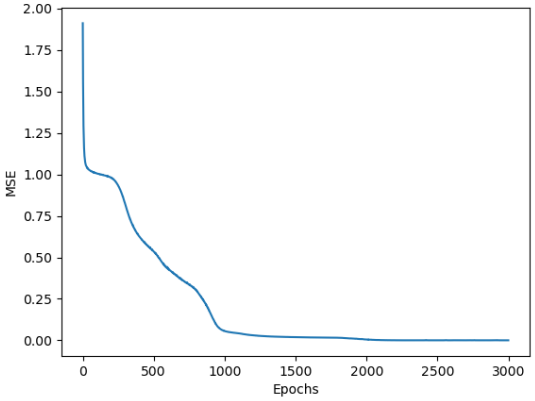





MSE
Epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
500
1000
1500
2000
2500
3000


MSE
Epochs
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000


MSE
Epochs
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
3000


MSE
Epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
500
1000
1500
2000
2500
3000

Resulting Error Histogram