# ECGR_4105_HW6_Problem_1

November 28, 2024

[24]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import time

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

[17]:
```
file_url = 'https://raw.githubusercontent.com/lnguye782/ECGR-4105-Intro-to-ML/
  ↪refs/heads/main/HW6/Housing.csv'
data = pd.read_csv(file_url)

data.head()
```

[17]:
```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
```

[19]:
```
# Encode categorical variables
categorical_columns = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',
                       'airconditioning', 'prefarea', 'furnishingstatus']

data_encoded = data.copy()
for col in categorical_columns:
    le = LabelEncoder()
    data_encoded[col] = le.fit_transform(data[col])

# Separate features and target variable
X = data_encoded.drop(columns=['price'])
y = data_encoded['price']

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[21]:

```
# Split data into training and validation sets (80% train, 20% validation)
X_train, X_val, y_train, y_val = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪random_state=42)

y_train = np.array(y_train)
y_val = np.array(y_val)
```

[26]:

```
# Build the neural network
model = Sequential([
    Dense(8, input_dim=12, activation='relu'),  # Hidden layer with 8 nodes
    Dense(1, activation='linear')              # Output layer for regression
])

# Compile the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Train the model
start_time = time.time()
history = model.fit(X_train, y_train, validation_data=(X_val, y_val),␣
 ↪epochs=100, batch_size=32, verbose=0)
training_time = time.time() - start_time

# Evaluate the model
train_loss, train_mae = model.evaluate(X_train, y_train, verbose=0)
val_loss, val_mae = model.evaluate(X_val, y_val, verbose=0)

# Report results
training_time, train_loss, train_mae, val_loss, val_mae
```

[26]:

```
(9.537226676940918, 25234312790016.0, 4706488.5, 30129436753920.0, 5007497.5)
```

[27]:

```
# Build the extended neural network with two additional hidden layers
extended_model = Sequential([
    Dense(8, input_dim=12, activation='relu'),  # First hidden layer
    Dense(8, activation='relu'),               # Second hidden layer
    Dense(8, activation='relu'),               # Third hidden layer
    Dense(1, activation='linear')              # Output layer for regression
])

# Compile the extended model
extended_model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Train the extended model
start_time = time.time()
extended_history = extended_model.fit(X_train, y_train, validation_data=(X_val,␣
 ↪y_val), epochs=100, batch_size=32, verbose=0)
extended_training_time = time.time() - start_time

# Evaluate the extended model
extended_train_loss, extended_train_mae = extended_model.evaluate(X_train,␣
 ↪y_train, verbose=0)
extended_val_loss, extended_val_mae = extended_model.evaluate(X_val, y_val,␣
 ↪verbose=0)
```

[29]:

```
# Report results
results = {
    "Extended (3 Hidden Layers)": {
        "Training Time (s)": extended_training_time,
        "Training Loss": extended_train_loss,
        "Training MAE": extended_train_mae,
        "Validation Loss": extended_val_loss,
        "Validation MAE": extended_val_mae,
    }
}

results
```

[29]:

```
{'Extended (3 Hidden Layers)': {'Training Time (s)': 11.375105381011963,
  'Training Loss': 24964273012736.0,
  'Training MAE': 4682190.5,
  'Validation Loss': 29835818696704.0,
  'Validation MAE': 4984039.5}}
```