# ECGR_4105_HW6_Problem_2

November 28, 2024

[6]:
```
import pandas as pd
import tensorflow as tf
import time

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

[2]:
```
file_path = 'https://raw.githubusercontent.com/lnguye782/ECGR-4105-Intro-to-ML/
  ↪refs/heads/main/HW6/cancer.csv'
cancer_data = pd.read_csv(file_path)

cancer_data.head()
```

[2]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | \ |
|---|---|---|---|---|---|---|
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | |
| 4 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | |

| | mean compactness | mean concavity | mean concave points | mean symmetry | \ |
|---|---|---|---|---|---|
| 0 | 0.27760 | 0.3001 | 0.14710 | 0.2419 | |
| 1 | 0.07864 | 0.0869 | 0.07017 | 0.1812 | |
| 2 | 0.15990 | 0.1974 | 0.12790 | 0.2069 | |
| 3 | 0.28390 | 0.2414 | 0.10520 | 0.2597 | |
| 4 | 0.13280 | 0.1980 | 0.10430 | 0.1809 | |

| | mean fractal dimension | … | worst texture | worst perimeter | worst area | \ |
|---|---|---|---|---|---|---|
| 0 | 0.07871 | … | 17.33 | 184.60 | 2019.0 | |
| 1 | 0.05667 | … | 23.41 | 158.80 | 1956.0 | |
| 2 | 0.05999 | … | 25.53 | 152.50 | 1709.0 | |
| 3 | 0.09744 | … | 26.50 | 98.87 | 567.7 | |
| 4 | 0.05883 | … | 16.67 | 152.20 | 1575.0 | |

| | worst smoothness | worst compactness | worst concavity | worst concave points | \ |
|---|---|---|---|---|---|

```
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

[4]:
```
# Separate features and target
X = cancer_data.drop(columns=["target"])
y = cancer_data["target"]

# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

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

[7]:
```
# Define the neural network with an explicit Input layer
model = Sequential([
    Input(shape=(X_train.shape[1],)),  # Define the input shape
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification output layer
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

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

# Evaluate the model
evaluation = model.evaluate(X_val, y_val, verbose=0)
```

```
training_loss = history.history['loss'][-1]
validation_accuracy = evaluation[1]

training_time, training_loss, validation_accuracy
```

[7]: (6.114588499069214, 0.013174124993383884, 0.9561403393745422)

[8]:
```
# Extend the neural network to include two additional hidden layers
model_extended = Sequential([
    Input(shape=(X_train.shape[1],)),  # Explicit Input layer
    Dense(32, activation='relu'),
    Dense(32, activation='relu'),  # First additional hidden layer
    Dense(32, activation='relu'),  # Second additional hidden layer
    Dense(1, activation='sigmoid')  # Binary classification output layer
])

# Compile the extended model
model_extended.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

# Train the extended model
start_time = time.time()
history_extended = model_extended.fit(X_train, y_train, validation_data=(X_val,␣
 ↪y_val), epochs=100, verbose=0)
training_time_extended = time.time() - start_time

# Evaluate the extended model
evaluation_extended = model_extended.evaluate(X_val, y_val, verbose=0)

training_loss_extended = history_extended.history['loss'][-1]
validation_accuracy_extended = evaluation_extended[1]

training_time_extended, training_loss_extended, validation_accuracy_extended
```

[8]: (7.382623195648193, 0.00022640445968136191, 0.9649122953414917)