Assignment 3 Report: Training a Neural Network Classifier

Subhan Farrakh - FA21-BCE-073

June 15, 2024

1 Introduction

This report covers the process of training a fully connected neural network to classify human spine conditions as normal or abnormal using the `spine.csv` dataset. The dataset consists of various attributes of spine characteristics, and the task is to predict the class label.

2 Libraries Used

The following Python libraries were used in this assignment:

- `matplotlib.pyplot` for plotting the confusion matrix.
- `numpy` for array operations.
- `pandas` for data manipulation.
- `sklearn.metrics` for evaluating the model performance using the confusion matrix.
- `tensorflow` and `keras` for building and training the neural network.

3 Data Loading and Preprocessing

The dataset `spine.csv` was loaded using pandas. The features and labels were then separated.

```
data = pd.read_csv('/content/drive/MyDrive/spine.csv')

x = np.array(data[['Col1','Col2','Col3','Col4','Col5',
'Col6','Col7','Col8','Col9','Col10','Col11','Col12']])
y = data[['Class_label']]
```

4 Data Splitting

The dataset was split into training and testing sets with 80% of the data used for training and 20% for testing.

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2)
```

5 Model Building

A Sequential model was built using Keras. The model consists of two layers:

1. A fully connected layer with 12 input dimensions and ReLU activation.
2. An output layer with a single neuron and sigmoid activation to output a probability score for binary classification.

```
model = Sequential()
model.add(Dense(12, input_dim=12, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

6 Model Compilation and Training

The model was compiled using the binary crossentropy loss function and Adam optimizer. The training was performed over 100 epochs with a batch size of 50.

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(xtrain, ytrain, epochs=100, batch_size=50)
```

7 Model Evaluation

The model was evaluated on the test set to determine its accuracy.

```
model.evaluate(xtest, ytest)
```

Output:

```
Test accuracy: 85.00%
```

8 Predictions and Confusion Matrix

Predictions were made on the test set, and a confusion matrix was generated to visualize the performance of the model.

```
y_pred = model.predict(xtest)
y_pred = (y_pred > 0.5)

from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(ytest, y_pred)
plt.matshow(matrix)
plt.colorbar()
plt.show()
```

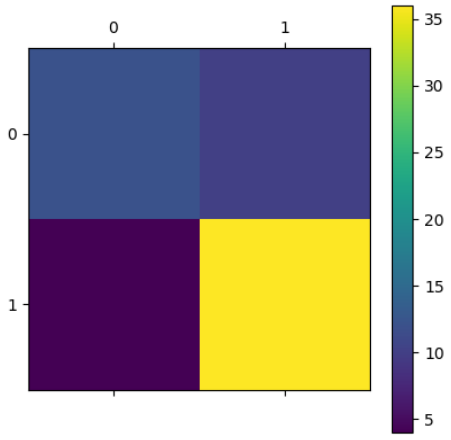

Figure 1: Confusion Matrix

9 Results

The evaluation of the model yielded the following results:

- **Accuracy**: The accuracy metric indicates the proportion of correctly classified instances. In this case, the accuracy was 85%.
- **Confusion Matrix**: The confusion matrix visualizes the number of true positive, true negative, false positive, and false negative predictions (Figure 1).