Assignment_1 - Deep Learning Multi-Class Classification Assignment Using Keras

Due – Sep 27, 2024

Submission instructions in Lecture_1

Context: You are tasked with assisting a manager at an automobile company in predicting customer segments for new potential customers. The company has classified customers into four segments (A, B, C, D) in the existing market, and this classification has significantly helped with targeted marketing efforts. You aim to create a multi-class classification model using **Keras** and deep learning to predict the correct customer segment for each new potential customer.

In this assignment, you will build two types of deep learning models:

1. **A model based on best practices**.
2. **A model using Keras Tuner for hyperparameter optimization**.

Assignment Instructions

Part 1: Data Loading and Preprocessing (20%)

1. **Load the Train and Test Datasets**:
    - You are provided with two datasets: **train.csv** and **test.csv**.
        - train.csv: Contains features and target labels for training.
        - test.csv: Contains only features for testing.
    - Check for missing values and handle them appropriately.
    - Perform **encoding** on categorical variables and **scaling** on numerical features.
2. **Prepare the Data**:
    - **For train.csv**:
        - Split the data into **X_train** (features) and **y_train** (target labels).
    - **For test.csv**:
        - Prepare the features (X_test), ensuring that all preprocessing steps (scaling, encoding) applied to train.csv are replicated on test.csv.

Part 2: Model 1 - Best Practices Model (30%)

1. **Model Architecture**:
    - Build a neural network model with **at least two hidden layers** using the **ReLU** activation function.
    - Use **softmax** as the activation function for the output layer to handle multi-class classification.
    - Use **categorical cross-entropy** as the loss function and **Adam** optimizer.
2. **Model Training**:
    - Train the model using the train.csv data (X_train, y_train) for **50 epochs** with a batch size of 32.
    - Use **early stopping** to prevent overfitting (monitor the validation loss and set a patience of 5 epochs).
    - Use **20% of the training data** as a validation set during training.
3. **Model Summary**:
    - Print the model architecture summary to review the structure.

Part 3: Model 2 - Hyperparameter Optimization Using Keras Tuner (30%)

1. **Implement Keras Tuner**:
    - Use Keras Tuner to optimize the following hyperparameters:
        - Number of hidden layers.
        - Number of neurons in each layer.
        - Learning rate.
        - Activation functions.
    - Set up a search space that explores different combinations of these hyperparameters. Experiment =5
2. **Tune and Train the Model**:
    - Use the best hyperparameters found by the tuner and train the model on the training data (X_train, y_train).

Part 4: Model Evaluation and Performance (20%)

1. **Evaluate on Training Data**:
    - Evaluate both models on the training data using **accuracy** as the metric.
    - Generate and print the **classification report** showing **precision**, **recall**, and **F1-score** for each customer segment.

```
from sklearn.metrics import classification_report

# Assuming 'y_pred' is your model's predictions and 'y_true' is the true labels
y_pred_classes = model.predict(X_test)
y_pred_labels = np.argmax(y_pred_classes, axis=1)
y_true_labels = np.argmax(y_test, axis=1)

print(classification_report(y_true_labels, y_pred_labels))
```

2.Training vs. Validation Loss Plot:

- After training, plot the **training vs. validation loss** to assess overfitting or underfitting.

```
import matplotlib.pyplot as plt

# Assuming 'history' contains the model's training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training vs. Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

3.Make Predictions on test.csv:

- For both models, use the **test.csv** data (X_test) to generate predictions for the customer segments.
- Convert the softmax output to predicted class labels (A, B, C, or D).

4.Confusion Matrix (Optional - if ground truth is provided for test data):

- If the test.csv file contains labels, generate and visualize the confusion matrix to evaluate the model's performance.

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_true_labels, y_pred_labels)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

5.Save Predictions:

- Save the predictions for the test set in a CSV file in the following format:
    - Customer_ID, Predicted_Segment

Part 5: Presentation and Recommendations (10%)

1. **Model Comparison**:
    - Summarize the performance of both models and compare their accuracy, precision, recall, and F1-scores.
    - Highlight which model performed better and why.
2. **Business Recommendations**:
    - Based on the model's predictions, provide recommendations on how the company can effectively target customers in each segment for the new market.
    - Suggest any additional features or data that could further improve model performance in the future.

Deliverables:

1. **Python Notebook**:
    - All code from data loading, preprocessing, model building, tuning, and evaluation should be included.
    - Include relevant code outputs, plots, and metrics.
    - Save the predictions from the test set in the format: Customer_ID, Predicted_Segment.
2. **Presentation Deck**:
    - Slide 1: Problem statement and overview of the dataset.
    - Slide 2: Best Practices Model architecture and performance.
    - Slide 3: Keras Tuner Model architecture and performance.
    - Slide 4: Training vs. Validation Loss plot.
    - Slide 5: Classification report for both models.
    - Slide 6: Confusion Matrix (if applicable).
    - Slide 7: Recommendations for customer segmentation.

Evaluation Criteria:

- **Data Preprocessing (20%)**: How well you handled missing values, scaling, and encoding.
- **Model Architecture (30%)**: Design of the two neural networks, including Keras Tuner for hyperparameter optimization.
- **Model Performance (20%)**: Model accuracy on the test data, prediction results, and analysis.
- **Visualization (10%)**: Clarity of training/validation loss plots, confusion matrix, and classification report.
- **Presentation and Business Recommendations (10%)**: Clarity of insights and recommendations for customer segmentation

Guidelines:

- Use a clean and structured coding approach, separating preprocessing, model building, and evaluation steps.
- Ensure that you document your code well, especially the parts involving Keras Tuner and model evaluation.
- Provide clear business insights from your results and predictions.