SOC '24
Report

on

SPEECH EMOTION RECOGNITION

Mentee: Arya Suwalka (23B0902)

Mentor: Aakriti Chandra, Aryaman Angurana

Table of Contents

1 Week 1: Learning Basic Python and Essential Libraries

I learned Basic Python, Numpy Library, Matplotlib Library, Pandas Library.

- **Basic Python Syntax and Programming Concepts**
  Learned basic Python syntax involving understanding how to declare variables and use various data types such as integers, floats, strings, and booleans. Control structures in Python, such as conditional statements (if, elif, else) and loops (for, while), allow for the implementation of complex logic in programs. Functions are fundamental building blocks in Python, enabling code reuse and modular programming. Additionally, understanding the scope of variables and how to import and use modules from the standard library is crucial for efficient programming.

- **NumPy Library**
  NumPy, short for Numerical Python, is a powerful library essential for numerical computations in Python. It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Creating and manipulating arrays with NumPy involves operations such as array indexing, slicing, and reshaping. NumPy also supports advanced mathematical operations like broadcasting, which allows for element-wise operations on arrays of different shapes, and includes a wide range of mathematical and statistical functions that are optimized for performance.

- **Matplotlib Library**
  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is particularly well-suited for generating plots and charts, which are essential for data analysis and presentation. Basic plotting with Matplotlib involves creating line graphs, scatter plots, and bar charts. Customizing these plots is straightforward, allowing for the addition of titles, labels, and legends to improve readability and presentation. Advanced plotting techniques in Matplotlib include creating subplots to display multiple plots in a single figure and customizing the appearance of plots with different styles, colors, and markers. These capabilities make Matplotlib a versatile tool for visualizing data in a clear and informative manner.

- **Pandas Library**
  Pandas is a powerful data manipulation and analysis library built on top of NumPy. It provides data structures like Series and DataFrames, which are designed to handle structured data easily. Series are one-dimensional labeled arrays, while DataFrames are two-dimensional, size-mutable, and potentially heterogeneous tabular data structures. Pandas makes it simple to import data from various file formats such as CSV, Excel, and SQL databases. Manipulating data with Pandas includes operations like indexing, selecting data, merging, joining, and concatenating DataFrames. Additionally, Pandas offers tools for handling missing data and performing data cleaning tasks, making it an essential library for data analysis and preprocessing.

2 Week 2: Learning Regression and Classification Models

In this week I implemented Logistic Regression and SVM classification model from scratch.

- **Difference Between Regression and Classification**
  Regression and classification are two types of predictive modeling techniques used in machine learning. Regression models predict continuous quantities, such as predicting a house price based on its features. In contrast, classification models predict discrete class labels, like determining if an email is spam or not.
  Although Logistic Regression is used for classification, it is still called regression because it predicts probabilities (continuous values) for different classes.

- **Regression Models**

  1. **Linear Regression**
     Linear Regression assumes a linear relationship between the input variables and the output. The goal is to fit a line that best represents the data by minimizing the difference between the predicted and actual values.
  2. **Logistic Regression**
     Logistic Regression is used for binary classification problems. It

uses the logistic function to model the probability that a given input point belongs to a certain class, outputting a probability value between 0 and 1.

3. **Ridge Regression**
   Ridge Regression is a type of linear regression that includes a regularization term to prevent overfitting. This term penalizes large coefficients, encouraging the model to keep them small and thus more generalizable.

- **Classification Models**

1. **Naive Bayes Classifier**
   The Naive Bayes Classifier is a probabilistic model based on Bayes' theorem. It assumes that the features are independent given the class. Despite this strong assumption, it works well in practice, especially for text classification tasks.
2. **Support Vector Machines (SVM)**
   SVMs are supervised learning models used for classification and regression. They work by finding the hyperplane that best separates the data into different classes, maximizing the margin between the data points of different classes.

3 Week 3: Learning Neural Networks and Recurrent Neural Networks (RNNs)

In this week I implemented a basic Neural Network model from scratch and an RNN model using a library. I have also answered the short answer type Questions given in the RNNques.pdf

1. **Neural Networks**

   Overview
   Neural networks are computational models inspired by the human brain. They consist of layers of interconnected nodes (neurons) that process input data and learn to perform tasks by adjusting the connections (weights) based on the data.

   Components

- **Neurons and Layers:** The basic unit of a neural network is a neuron, organized into layers. There are typically three types of layers: input layer, hidden layers, and output layer.
- **Activation Functions:** Functions like ReLU, sigmoid, and tanh introduce non-linearity into the model, enabling it to learn complex patterns.
- **Training Process:** Training involves forward propagation to compute predictions, loss calculation to measure prediction error, and backpropagation to update weights using optimization algorithms like gradient descent.

Applications
Neural networks are used in various fields such as image and speech recognition, natural language processing, and game playing.

2. **Recurrent Neural Networks (RNNs)**

Overview
RNNs are a type of neural network designed for sequential data. They have connections that form directed cycles, allowing them to maintain information about previous inputs in their hidden state, making them suitable for tasks where context and order matter.

Components

- **Hidden States:** RNNs maintain a hidden state that gets updated at each time step, capturing information from previous inputs.
- **Backpropagation Through Time (BPTT):** A variant of backpropagation used to train RNNs, accounting for the temporal dependencies in the data.

Challenges

- **Vanishing and Exploding Gradients:** These issues can make training RNNs difficult, as gradients can become too small or too large, leading to ineffective learning.
- **Long-Term Dependencies:** Standard RNNs struggle with learning long-term dependencies in sequences.

Applications
RNNs are widely used in language modeling, machine translation, speech recognition, and time series prediction.

4 Week 4: Audio Preprocessing with Librosa

In this week I completed an assignment that uses Librosa to preprocess data.

Why do we audio preprocessing ?
Audio preprocessing is essential for improving the quality and consistency of audio data, which enhances the performance of machine learning models. Key steps include noise reduction to remove unwanted sounds, normalization to standardize audio levels, and segmentation to break audio into manageable chunks. Feature extraction, such as obtaining MFCCs, transforms raw audio into meaningful representations. These steps help reduce overfitting, improve computational efficiency, and ensure that the models focus on relevant audio features, leading to better accuracy and reliability in tasks like speech and music recognition.

Some terms to know

- Sampling Rate: Number of audio samples per second; higher rates capture more detail.
- Amplitude: Loudness of the audio signal; higher amplitude means louder sound.
- Frequency: Number of wave cycles per second; determines pitch.
- Waveforms: Show amplitude variations over time.
- Spectrograms: Display frequency content over time, useful for detailed audio analysis.

Librosa
Librosa is a powerful Python library for audio and music analysis. It provides the necessary tools to perform audio preprocessing efficiently. Key functionalities include:

- **Loading and Visualizing Audio:** Librosa allows easy loading of audio files and provides functions to visualize waveforms and spectrograms.
- **Feature Extraction:** It supports extracting features like MFCCs, chroma features, and spectral contrast, which are essential for various audio analysis tasks.
- **Audio Manipulation:** Librosa includes functions for resampling, noise reduction, and audio segmentation, facilitating comprehensive preprocessing.

Practical Implementation
Using Librosa, we performed various audio preprocessing tasks:

- **Loading Audio Files:** I used "librosa.load" to load audio files into a format suitable for analysis.
- **Visualizing Waveforms and Spectrograms:** Functions like "librosa.display.waveplot" and "librosa.display.specshow" were used to visualize audio data.
- **Extracting Features:** Extracted MFCCs and other relevant features using "librosa.feature.mfcc" and similar functions.
- **Noise Reduction and Normalization:** Implementing noise reduction techniques and normalizing audio levels were done to prepare the data for further processing.

5 Week 5&6: Final Project

In these two weeks I made my final project, which involved building and evaluating a neural network model for emotion recognition from audio data. The task utilized various libraries including TensorFlow Keras for model building, and Librosa for audio preprocessing. This project helped consolidate my learning and apply practical skills in a comprehensive manner.
The model achieved an accuracy of 71.08% on the test data. The classification report provided further insights into the model's performance across different emotion classes.

6 Week 7&8: Tic-Tac-Toe AI Bot

In weeks 7 and 8, the focus was on creating a Tic-Tac-Toe AI bot, providing an introduction to AI and game theory. This task involved understanding fundamental concepts of AI and implementing a series of functions to enable the bot to play Tic-Tac-Toe flawlessly. The goal was to ensure that the AI bot never loses, regardless of the opponent's moves.
The activity was supported by two primary documents:

- **AI.pdf:** This document provided a comprehensive introduction to artificial intelligence and game theory concepts necessary for building the Tic-Tac-Toe bot.
- **Tictactoe_Ques.pdf:** This document outlined the specific task requirements and detailed the modifications needed in the "q1.py" file.

Implementation Overview
The implementation required modifying the "q1.py" file to execute seven critical functions. Below is a brief explanation of each function and its role in the AI bot:

1. `is_win(self)`: This function checks if the current state of the board represents a winning condition for either player. It scans the rows, columns, and diagonals for three identical symbols (either 'X' or 'O').
2. `is_draw(self)`: This function determines if the game has reached a draw. It checks if the board is full without any player winning, indicating that no more moves are possible and the game is tied.
3. `get_valid_actions(self)`: This function returns a list of valid actions (empty positions) that the player can take on the current board state. It ensures that the moves considered are legal and possible.
4. `is_terminal_history(self)`: This function evaluates whether the game history has reached a terminal state. A terminal state is when the game has ended, either by a win for one of the players or a draw.
5. `get_utility_given_terminal_history(self)`: This function calculates the utility value of a terminal state. It assigns a value indicating the outcome of the game: a win for the AI, a win for the opponent, or a draw.

6. `update_history(self, action)`: This function updates the game history with the latest action taken. It records the move and updates the board state accordingly.

7. `backward_induction(history_obj)`: This function implements the backward induction algorithm to determine the optimal strategy for the AI. It recursively evaluates all possible future moves, ensuring that the AI makes the best possible decision to avoid losing.

This activity was both challenging and enjoyable, providing a hands-on introduction to AI and game theory. By the end of weeks 7 and 8, I successfully developed a Tic-Tac-Toe AI bot that could play the game flawlessly, never losing to any opponent. This project not only reinforced my understanding of AI principles but also showcased the practical application of theoretical concepts in a fun and engaging way.