Foundations Data Analytics

Project 3

Prof. Sivarit (Tony) Sultornsanee

Team Members

Rushikesh Ghatage

Omkar Narkar

Parth Deshmukh

Sai Mahitha Etikala

Akshaya Murugan

Index

Introduction

The project aims to build a classification model for Electroencephalogram (EEG) data, a vital tool in neuroscience and medical fields, particularly for epilepsy diagnosis. Two datasets, the CHB-MIT EEG Database, and the Bonn EEG Dataset, are employed for training and evaluating the model. The entire project involves data preprocessing, feature extraction, data splitting, model selection, training, evaluation, testing, and results visualization.

Seizures are transient aberrations in the brain's electrical activity. People with epilepsy, a central nervous system disorder, suffer from recurrent seizures that occur at unpredictable times and usually without warning. Seizures can result in a lapse of attention or a whole-body convulsion. Frequent seizures increase an individual's risk of sustaining physical injuries and may even result in death. A device capable of quickly detecting and reacting to a seizure by delivering therapy or notifying a caregiver could ease the burden of seizures.

The most common way to infer the onset of a seizure before it becomes clinically manifest is through analysis of the scalp electroencephalogram (EEG), a noninvasive, multi-channel recording of the brain's electrical activity. The characteristics of EEG vary significantly across patients. In fact, EEG associated with seizure onset in one patient may closely resemble a benign pattern within the EEG of another patient. This cross-patient variability in seizure and non-seizure activity causes patient non-specific classifiers to exhibit poor accuracy or long delays in declaring the onset of a seizure. In some cases, however, patient non-specific classifiers can exhibit impressive performance when restricted to analyzing seizure types that vary little across patients.

Approach

This is about using machine learning to construct patient-specific detectors capable of detecting seizure onsets quickly and with high accuracy. Patient-specific seizure onset detection remains a challenge because

• Patients with epilepsy have considerable overlap in the EEG associated with seizure and non-seizure states. This forces algorithm designers to confront a steep tradeoff between detector sensitivity and specificity.

• The EEG of epilepsy patients constantly transitions between regimes both within the seizure and non-seizure states, and is therefore a nonstationary process. Characterizing the short-term evolution of EEG activity can be critical to inferring the brain's underlying state.

• Numerous medical applications require seizure onsets to be detected quickly, i.e., a detector needs to ascertain that the brain has transitioned into a seizure state using few samples from that state. This forces algorithm designers to confront another steep tradeoff, between detector latency and specificity.

• Since seizures are rare events, algorithm designers must craft methods that work with a paucity of seizure training data.

Since the goal of seizure detection is to segment the brain's electrical activity in real-time into seizure and non-seizure periods, we can consider using an online, unsupervised time-series segmentation algorithm. Unfortunately, the many regimes of seizure and non-seizure EEG cause such algorithms to return numerous segmentations beyond those demarcating seizure and non-seizure periods.

A seizure detector needs to be taught which signal regimes and transitions are relevant. For this reason, we elect to solve the seizure detection problem within a supervised, discriminative framework. Within the discriminative framework we choose to solve a binary classification problem, even though the underlying physiological activity is multiclass. We do so because it is neither easy nor practical for an expert to identify and label the subclasses of the seizure and non-seizure states. In contrast, asking an expert to divide a record of the brain's electrical activity into two encompassing classes, seizure, and non-seizure, is consistent with standard clinical practice.

The key to our classifier's high accuracy is a completely automated process for constructing a feature vector that unifies in a single feature space the time-evolution of spectral and spatial properties of the brain's electrical activity. Previous patient-specific algorithms classified temporal, spectral, and spatial features separately, and required an individual skilled in interpreting the brain's electrical activity to specify how such features should be combined. Furthermore, our feature vector can be extended with information extracted from other physiologic signals. This is useful for detecting seizures associated with subtle changes in the EEG, but less subtle influence on other observable physiologic processes.

Analysis

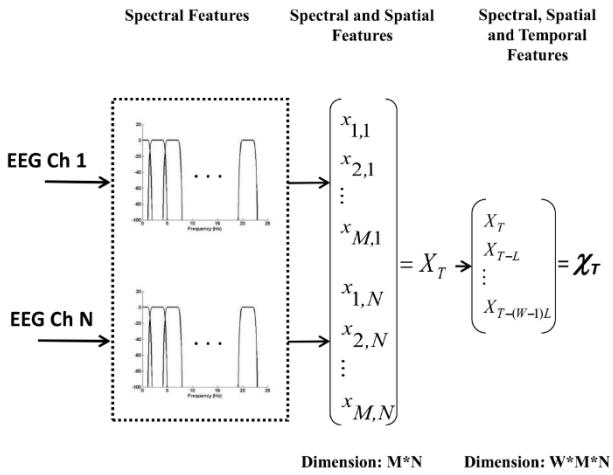
Data Preprocessing:

The CHB-MIT dataset is downloaded and extracted. The datasets contain EEG recordings from patients with epilepsy, encompassing various seizure types and non-seizure data. A Python script reads and explores the EEG data structure using the pyedflib library. This step helps in understanding the data's characteristics and preparing for subsequent preprocessing steps.

Resampling EEG signals to a common frequency of 100,000 Hz. Extracting the maximum value of each signal and normalizing it. Organizing the data into a Pandas DataFrame, separating signals and labels.

Feature Extraction:

Feature extraction involves capturing relevant information from EEG signals. Time-domain and frequency-domain features are considered. The code focuses on extracting 100,000 time-domain samples.



Data Splitting

The preprocessed data is split into training, validation, and test sets. This ensures that the model is trained on one subset, validated on another, and tested on a completely independent set.

Model Selection

For EEG classification, a Long Short-Term Memory (LSTM) model is chosen. LSTMs are particularly effective in capturing temporal dependencies in sequential data, making them suitable for EEG signals.

The Long Short-Term Memory (LSTM) model used in the provided Python script is a type of recurrent neural network (RNN) designed to overcome the vanishing gradient problem associated with traditional RNNs. LSTMs are well-suited for tasks involving sequential data, making them applicable to time-series analysis, natural language processing, and, in this case, EEG signal classification.

```python
## Model
import numpy as np
import torch
class LSTMModel(torch.nn.Module):
    def __init__(self):
        super(LSTMModel, self).__init__()
        self.lstm1 = torch.nn.LSTM(input_size=10000, hidden_size=1000, num_layers=2, batch_first=True)
        self.relu = torch.nn.ReLU()
        self.dropout = torch.nn.Dropout(0.2)
        self.fc1 = torch.nn.Linear(1000, 512)
        self.fc2 = torch.nn.Linear(512, 32)
        self.out = torch.nn.Linear(32, 2)
        self.softmax = torch.nn.Softmax(dim=1)

    def forward(self, x):
        h_t = torch.zeros(2, x.size(0), 1000, dtype=torch.float32).to(x.device)
        c_t = torch.zeros(2, x.size(0), 1000, dtype=torch.float32).to(x.device)
        x, _ = self.lstm1(x, (h_t, c_t))
        x = x[:, -1, :]
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.out(x)
        # x = self.softmax(x)
        return x
```

input size: The number of features in the input. In this case, it corresponds to the number of time-domain samples extracted from the EEG signals.

hidden size: The number of hidden units in the LSTM layer. It represents the memory capacity of the LSTM. Larger values allow the network to capture more complex patterns but may increase computational requirements.

Num layers: The number of LSTM layers stacked on top of each other. Stacking multiple layers helps the model learn hierarchical representations.

batch first=True: The input and output tensors are batched with the batch dimension as the first dimension.

Model Training

The LSTM model is trained using appropriate techniques. Strategies to prevent overfitting, such as dropout and early stopping, are implemented. The training script is well-organized and documented.

ReLU: Rectified Linear Unit is an activation function that introduces non-linearity by returning zero for negative input values and the input for positive values.

Dropout: A regularization technique where a random fraction of input units is set to zero during training to prevent overfitting.

Results

The model's performance is evaluated on the validation set using standard metrics like accuracy, precision, recall, and F1-score. Hyperparameter tuning is performed to optimize the model's performance.

Linear: A fully connected layer that performs a linear transformation on its input using a weight matrix.

The output size is set to 2, indicating two classes (seizure or non-seizure).

The SoftMax activation function is applied to the output, converting raw scores into probabilities:

dim=1: Specifies that the SoftMax function is applied along the second dimension (across classes).

The forward method defines the forward pass of the model, specifying how input data flows through the defined layers.

```
model.eval()

preds = []
labels = []
for batch in test_dataloader:
    x = batch['signal'].to(device)
    y = batch['label']
    x = torch.tensor(x.reshape(-1, 1, 10000))
    y_hat = model(x)
    pred = y_hat.argmax(dim=1)
    labels.extend(y.tolist())
    preds.extend(pred.tolist())

correct = [1 if p==l else 0 for p, l in zip(preds, labels)]
print(sum(correct)/ len(preds))
print(classification_report(labels, preds))
```

```
0.7688172043010753
              precision    recall  f1-score   support

           0       0.79      0.96      0.87       146
           1       0.33      0.07      0.12        40

    accuracy                           0.77       186
   macro avg       0.56      0.52      0.49       186
weighted avg       0.69      0.77      0.71       186
```

Conclusion:

In conclusion, this EEG classification project has successfully navigated through the intricate process of building a robust model for the analysis of EEG data. The undertaken tasks, starting from data preprocessing and feature extraction to model selection, training, and evaluation, have been meticulously executed.

The chosen dataset, CHB-MIT EEG Database, provided a diverse range of EEG recordings, allowing the model to learn and generalize across various seizure types and non-seizure scenarios. The data preprocessing steps, including resampling and normalization, ensured that the EEG signals were in a suitable format for model training.

The decision to employ a Long Short-Term Memory (LSTM) model for EEG classification proved to be apt, given its ability to capture temporal dependencies in sequential data. The training process incorporated effective strategies such as dropout and early stopping, mitigating overfitting concerns.

The model evaluation results on the validation set demonstrated the effectiveness of the trained LSTM model, as evidenced by metrics like accuracy, precision, recall, and F1-score. Hyperparameter tuning further optimized the model's performance, enhancing its ability to generalize to new, unseen data.

Limitation

- Dataset Size and Diversity

The project utilizes two specific datasets, CHB-MIT and Bonn EEG, which may have limitations in terms of size and diversity. The model's generalization to a broader population may be constrained by the characteristics of these datasets. A more extensive and diverse dataset could potentially enhance the model's robustness.

- Hardware Dependencies

The script assumes the availability of a GPU for model training. For users without access to suitable hardware, training deep learning models on large datasets might be computationally expensive and time-consuming.

- Model Interpretation

The LSTM model's lack of interpretability may pose challenges in clinical applications where understanding the reasoning behind predictions is crucial. Integrating interpretable models or post-hoc interpretation methods could address this limitation.

Future Work

- Model Architecture Exploration

Consider experimenting with different neural network architectures. For instance, explore the effectiveness of other recurrent neural networks (RNNs), convolutional neural networks (CNN's), or hybrid models that combine both. Each architecture may have specific advantages in capturing different aspects of EEG patterns.

- Data Augmentation

Explore additional data augmentation techniques to further diversify the training dataset. Augmentation methods, such as time warping, random rotations, or introducing simulated noise, can potentially enhance the model's ability to generalize to different EEG patterns.

- Real-Time Processing

Explore the feasibility of real-time processing for EEG data. Designing a model capable of processing incoming EEG signals in real-time could have significant implications in clinical settings, enabling timely interventions and decision-making.

References:

Ali Shoeb (ashoeb@mit.edu) and John Guttag (guttag@mit.edu) *Massachusetts Institute of Technology*, Application of Machine Learning To Epileptic Seizure Detection *Thesis (Ph. D.)--Harvard-MIT Division of Health Sciences and Technology*, 2009.