Identifying Brain Activity from EEG Recordings

CS229: Machine Learning

Ray Iyer - rri@stanford.edu, Trenton Chang - tchang97@stanford.edu, Caroline Ho - cho19@stanford.edu
Department of Computer Science, Stanford University

## Prediction Task

- **Motivation:** classify EEG recordings as different types of brain activity
- **Application:** automate detection of disorders such as epilepsy in healthcare settings

## Data

**Epileptic Seizure Recognition Dataset** [1]
- **Size:** 11500 rows (500 people x 23 recording segments per person), 178 columns (1/178 s)
- **Input:** 1-second-long EEG recording
- **Label:** one of 5 classes of brain activity
  1. Eyes closed (non-epileptic subject)
  2. Eyes open (non-epileptic subject)
  3. Epileptic seizure (epileptic subject)
  4. From healthy area (epileptic subject)
  5. From tumor (epileptic subject)

## Feature Extraction

**Preprocessing**
- Normalize data

**Feature Sets**
- **Summary statistics:** min and std over recording
- **Spectral entropy:** treats normalized power distribution in the frequency domain as a probability distribution, and calculates the Shannon entropy
- **Raw data:** raw numeric EEG data
- **Fourier transform:** transforms raw EEG data from time to frequency domain

## Softmax Regression

- **Hypothesis:**
  \[
  \phi_i = \frac{\exp(F(x)_i)}{\sum_j \exp(F(x)_j)}
  \]
- **Loss:** multiclass cross entropy
  \[
  L(k, y) = -\sum_{j} (1 - y_j) \log \phi_j
  \]

## Hidden Markov Model (HMM)

- Use Baum-Welch algorithm (similar to EM) to generate one HMM per class
- **Assumption:** EEG data is a noisy stream modelable by transitions between n Gaussians
- **Markov assumption:**
  \[
  P(Y_{k+1} | X_{k+1}, Y_k) = P(Y_{k+1} | X_{k+1})
  \]
- **Emission probabilities:**
  \[
  P(Y_k | X_{k}, Y_{k-1}) = P(Y_k | X_k)
  \]
- **Evaluation:**
  - Test time: which HMM yields the highest probability path through states?
  - 70/30 train-test split, Trials with n=[1, 2, … 8] latent variables over 178 timesteps

## K-Nearest Neighbors

- **Predict class based on k closest EEGs**
- **Euclidean distance:**
  \[
  d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}
  \]

## Evaluation

- **Nested CV:** 10 inner + outer folds

## Models

**Convolutional Neural Network (CNN)**
- **Architecture:**
  1. 2D convolutional layers w/64 filters (ReLU)
  2. 1D max pooling layer
  3. 2D convolutional layers w/128 filters (ReLU)
  4. 1D global average pooling layer
  5. Dropout layer
  6. Dense layer (softmax)

## Experimental Results

<table>
<thead>
<tr>
<th>Features</th>
<th>Model</th>
<th>Train Acc</th>
<th>Test Acc</th>
<th>Test Macro F1</th>
<th>Test Seizure F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary stats</td>
<td>Softmax reg</td>
<td>0.465</td>
<td>0.434</td>
<td>0.399</td>
<td>0.873</td>
</tr>
<tr>
<td>kNN</td>
<td>0.591</td>
<td>0.472</td>
<td>0.466</td>
<td>0.902</td>
<td></td>
</tr>
<tr>
<td>Spectral entropy</td>
<td>kNN</td>
<td>0.768</td>
<td>0.681</td>
<td>0.653</td>
<td>0.919</td>
</tr>
<tr>
<td>Raw data</td>
<td>Softmax reg</td>
<td>0.303</td>
<td>0.195</td>
<td>0.186</td>
<td>0.323</td>
</tr>
<tr>
<td>kNN</td>
<td>0.655</td>
<td>0.423</td>
<td>0.421</td>
<td>0.799</td>
<td></td>
</tr>
<tr>
<td>HMM</td>
<td>0.432</td>
<td>0.434</td>
<td>0.420</td>
<td>0.873</td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>0.927</td>
<td>0.780</td>
<td>0.773</td>
<td>0.974</td>
<td></td>
</tr>
<tr>
<td>Fourier transform</td>
<td>Softmax reg</td>
<td>0.284</td>
<td>0.187</td>
<td>0.177</td>
<td>0.295</td>
</tr>
<tr>
<td>kNN</td>
<td>0.656</td>
<td>0.443</td>
<td>0.448</td>
<td>0.805</td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>0.959</td>
<td>0.740</td>
<td>0.737</td>
<td>0.941</td>
<td></td>
</tr>
</tbody>
</table>

## Discussion

- CNN performs well on raw sequence data, though model overfits
  - Likely picks up on recurring patterns in sequence
  - Most errors within super-class of epileptic or non-epileptic subject
- Spectral entropy kNN performs fairly well with far fewer features
  - Achieves good separability of data

## Future Directions

- Try different CNN hyperparameters and architectures to reduce overfitting, such as regularization techniques like weight decay and dropout
- Experiment with different time-distributed architectures (RNNs)
- Refine signal processing techniques with domain knowledge to improve separability in feature space

---