Seizure detection by EEG data classification using traditional kNN classifier and neural network

Sayemul Islam
Dept. of Electrical and Computer Engineering, Temple University
sayemul@temple.edu

Introduction: EEG or Electroencephalography is a method of monitoring electrical activity of brain or neural signal. EEG reflects correlated synaptic activity caused by post-synaptic potentials of cortical neurons. Typically, non-invasive electrodes are places along the sculp to measure the electrical activity. Although brain's activity is spontaneous and there are always some electrical activities on the neurons, it is observed that, unusual events are usually reflected on the EEG signals such as epileptic seizures. In this paper, we describe two methods to classify EEG signal into 'Seizure' and 'Non-Seizure' events. We will be using classical kNN classifier and find the classification error rate for training data, dev data and evaluation data and compare the results found using a Neural Network system developed using MATLAB's Neural Network Tool. The given dataset for the exam is collected from TUH EEG Seizure Corpus. We will train and evaluate the data on the training set and dev test set. The data contains roughly 19,000 training vectors, 2,000 development test vectors and 1,174 evaluation set vectors. Given EEG data is shown in Figure 1, Figure 2 and Figure 3. The vectors are single feature vectors taken from the middle of seizure event. The given data is in ascii format where the first column is the class label. The remaining columns are the corresponding feature vectors. Here we will be describing the methods we used to classify the data and built a classifier that will be able to characterize an EEG evaluation set.

Method 1 (kNN Classifier): kNN or k-nearest neighbor algorithm is a non-parametric method used for classification and regression. The input consists of k closest training examples in the feature space. The output depends on whether the kNN is used for classification or regression. kNN is a type of instant based learning or lazy learning where the function is only approximated locally, and all computation is differed until classification. It is also one of the simplest method of machine learning algorithm. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In classification phase, k is a user defined constant and an unlabeled vector is classified by assigning the label which is most frequent among the training samples nearest to that query point. Figure 5 shows the error rate as function of number of neighbors, k respectively for the training data, dev data and the testing data for the kNN classifier. We can see from Figure 5 that as the number of 'k' is increasing, the error rate of training data is increasing. Therefore, usually, high number of training samples are used to train classifier. Due to high number of training samples and proximity of the given training data, as number of k increases, there are more chances that the nearest neighbors can come from the other categories, increasing chances of the error. However, we can also observe from the Figure 5, compared to the training data, test data are independent and small in number. The test data usually comes from the actual class distribution. As number of nearest neighbor 'k' increases, density of the data from the same class is higher near any point, x, of that particular class. The chances that the closest neighbors belong to the same class also increases, increasing the chances of correct class assignment (using majority voting scheme) and reduction of the error rate. From the plot, the error rate settled around 38% using KNN classifier at k=75, for given training data.

Method 2 (Artificial Neural Network): A neural network for machine learning algorithm can be constructed in several ways. The main objective will however remain the same which is: sequential processing of the data, feature extraction, event spotting and post processing. A generic architecture of machine learning algorithm can include the input layer, few hidden layers and the output layer. The hidden layers can consist of sequential modeler, epoch posteriors, stacked denoising encoders, finite state machine etc. The neural network that we have built for this purpose used Long Short-Term Memory (LSTM)

networks. We can use LSTM networks to train a deep learning network to classify sequence data. An LSTM network is a type of recurrent neural network (RNN) that can learn long-term dependencies between time steps of sequence data. In a traditional neural network system, the gradient signal can end up being multiplied many times by the weight matrix during gradient back propagation phase which is associated with the connections between the neurons of the recurrent hidden layer. Which means, magnitude of weights in the transition matrix can possibly have acute impact on the neural network's learning process. If the weight in the matrix is too small, it can lead to vanishing gradients situation where the gradient signals get so small that the learning becomes incredibly slow or stops working. It also makes it more difficult for the network to learn long-term dependencies in the data. If the weight in the matrix are large it can lead to a situation where the gradient signal is so large it may diverge the learning process. This is often referred to as exploding gradients.

The LSTM model, however introduces a new structure called memory cell which helps to bypass the above-mentioned issue. A memory cell consists of total four elements: an input date, a neuron with self-recurrent connection, a forget gate and an output gate. The input gate can allow the incoming signal to alter the state of the memory cell or block it. The self-recurrent connection has a weight of 1.0 and ensures barring any outside interference, the state of a memory cell can remain constant from one timestep to another. The gates serve to modulate the interactions between the memory cell itself and its environment. The output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. Finally, the forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed. For our neural network model in this paper, we have modeled the neural network using LSTM networks in MATLAB where the training data was classified by sequentially sending feature vectors with respective tags into the training network. The first LSTM unit takes the initial network state and the first-time step of the sequence to make a prediction, and outputs the updated network state to the next LSTM unit. Each LSTM unit takes the updated network state from the previous unit and outputs a prediction and a new updated network state. LSTM networks can remember the state of the network between predictions. The network state is useful for when we do not have the complete time series in advance, or if we want to make multiple predictions on a long-time series. Figure 4 shows a typical LSTM network architecture used in neural networks.

Results: First we have trained the kNN classifier using the training dataset to demonstrate a traditional classification method, k value kept at 75. It is observed that, the error rate for the training dataset in kNN classifier started increasing with the increase of the number k and eventually settled around 38%, the error rate for the development data also settled around 47%. The performance data obtained from the classifier indicates that, the error rate for classifying the training data in closed loop test is 38.60%, the error rate for classifying the development dataset in open loop test is 47.45%.

Then we have trained our neural network using the training dataset and after the neural network was built into the machine, we ran another closed loop test. This time the training dataset gave us a 42.92% classification error. Then the development dataset gave us 39.80% classification error rate in open loop test All result data are summarized in Table 1.

It can be observed from the results obtained from both algorithms that the neural network performed better in open loop test datasets than the traditional kNN classifier in classifying seizure events from EEG seizure corpus.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
kNN (k=75)	38.60%	47.45%	49.57%
Neural Network	42.92%	39.80%	47.70%

Table 1. Error rate for different datasets and algorithms

Figures

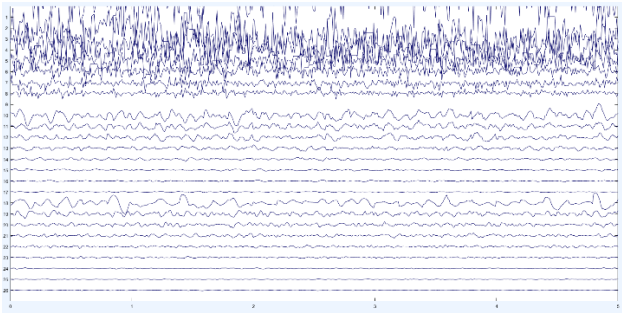

Figure 1: Training data obtained from TUH EEG Seizure Corpus

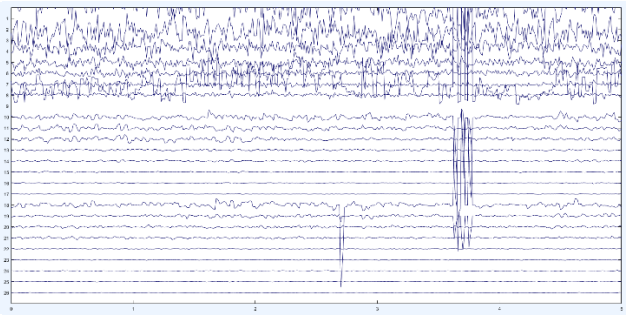

Figure 2: Dev data obtained from TUH EEG Seizure Corpus

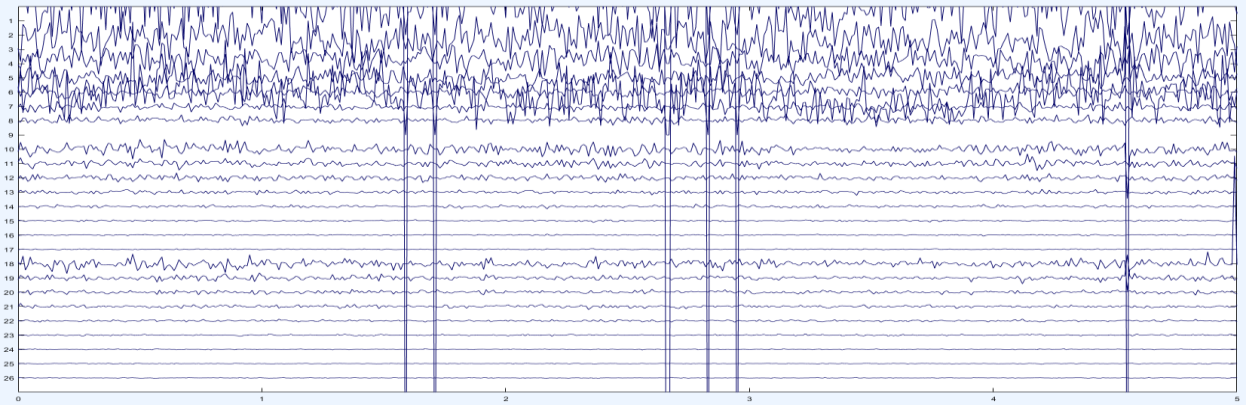

Figure 3: Evaluation data obtained from TUH EEG Seizure Corpus

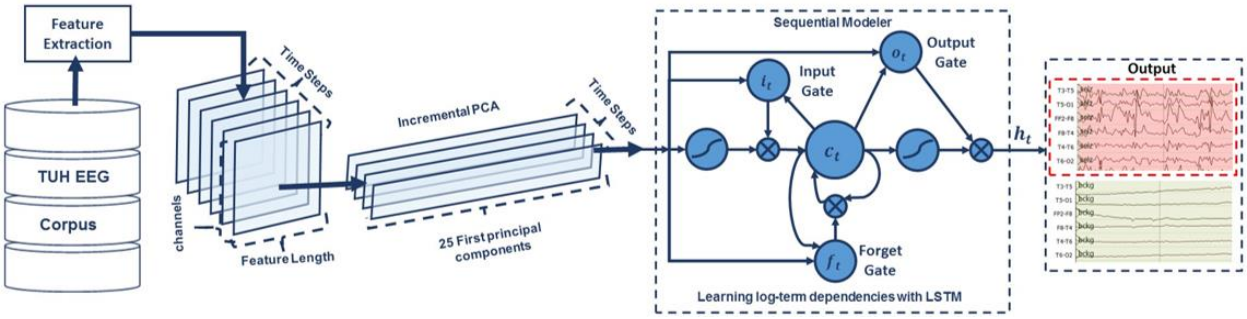

Figure 4: Typical LSTM architecture used in neural networks

Figures

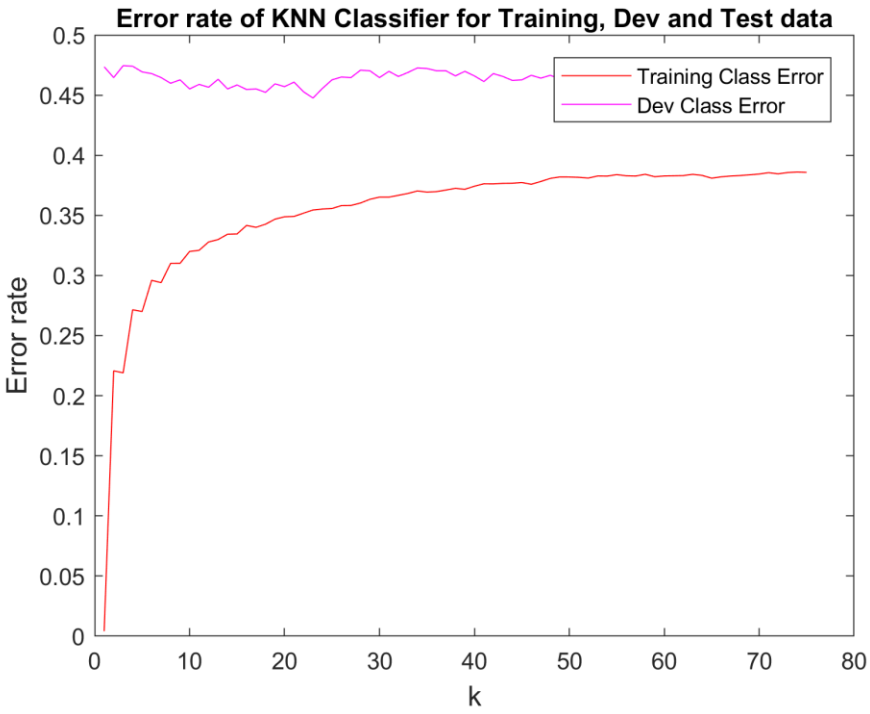

Figure 5: Error rate vs ‘k’ for Training data and Dev data