EEG data classification using traditional and neural network methods

Jun Chen
Department of Mechanical Engineering, Temple University
tuh39080@temple.edu

Introduction: Electroencephalography (EEG) is an electrophysiological monitoring method to record electrical activity of the brain. EEG measures voltage fluctuations resulting from ionic current within the neurons of the brain. In Dr. Picone's project, they are focusing on clinical electroencephalogram (EEG) recordings conducted at Temple University Hospital (TUH) from 2002 to 2013 (and beyond). The first release of the corpus is expected to contain over 20,000 EEGs. This set of data, where the first column is the class label and the remaining columns are the corresponding feature vector, is now to be classified using both k-nearest neighbors algorithm (kNN) and Backpropagation (Bp) algorithm. After training these data, a closed-loop evaluation data set and a open-loop development data set are used to test the accuracy of classifications. By doing this experiment, we practically implement machine learning algorithm into a real set of data to verify the principles. At the same time, we are able to briefly compare conventional pattern classification algorithms with neural network methods during this process and find some interesting conclusions.

Matlab is used as the environment to conduct the experiment in this report. In order to implement KNN and Bp algorithms, Statistics and Machine Learning Toolbox and Neural Network Toolbox are used. Statistics and Machine Learning Toolbox provides functions and apps to describe, analyze, and model data. It provides supervised and unsupervised machine learning algorithms including k-nearest neighbor. At the same time, Neural Network Toolbox provides algorithms, pretrained models, and apps to create, train, visualize, and simulate both shallow and deep neural networks. A feed-forward backpropagation network can be established using this toolbox, which we can use to implement neural network classification in this experiment. Also, Matlab is a good visualized tool that enable us to plot our results and watch the real-time value of each variable, matric or vector.

k-nearest neighbors algorithm (kNN): kNN algorithm is one of the simplest classification algorithms as well as one of the most common-used learning algorithms. The reason why kNN is chosen is that by using this kind on nonparametric technique, we do not have to know the underlying distribution of the EEG data and do not even necessary to make an assumption. In fact, in our case, it's hard to give an assumption of how these data are distributed. The principle of kNN is based on feature similarity. For a certain data, we find k nearest data vectors most commonly by calculating Euclidean distance between them. k is a user-defined constant. Generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct.

In Matlab, we use Statistics and Machine Learning Toolbox to implement kNN method. The first step is to make sure that the feature vectors have been normalized in order to be fit. However, the labels are not necessary to be normalized since they are already in 0 and 1 format in this case. The second step is to train a kNN classifier using function *fitcknn*. By using *fitcknn*, we can difine the number of k and standardize noncategorical predictor data. The method to calculate distance between data vectors is Euclidean distance in function *fitcknn*. After that, the third step is that we use the function predict to acquire predicted class labels for both open-loop and closed-loop test data using function *predict*. At last, accuracy is computed to evaluate the classification.

Backpropagation algorithm (Bp): Backpropagation is a supervised learning technique for training of a multilayer perceptron (MLP). A multilayer perceptron is a class of feedforward artificial neural network which consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. In a neural network we randomly initialize and update the weights to optimize output using gradient descent. Backpropagation is a way to efficiently obtain the gradients while trying to minimize the loss function for a neural network. Four steps are being processed in backpropagation algorithm: 1) Calculate the forward phase; 2) Calculate the backward phase; 3) Combine the individual gradients; 4) Update the weights.

To implement backpropagation algorithm in Matlab, we use the Neural Network Toolbox. At first, we define a three-layer network and set numbers of nods for each of these layers. Then we establish the function between input and output using tangent sigmoid transfer function for both hidden layers, and linear transfer function for output layer. Function used in this step is *newff*. After that, we train the data set using function *train* in Neural Network Toolbox. Prediction results can be simulated by comparing net structure with feature character vectors using function *sim*. At last accuracy is computed to evaluate the classification. It is worth realizing the importance of setting up a minimum global error for each data to fit the classification model. This is to avoid overfitting, since it would make no sense when the boundary is trying to be as close as it can to each data point.

Results: Table 1 concludes the results for both k-nearest neighbors algorithm and Backpropagation algorithm. To obtain the highest accuracy, the numerical value of k is being adjusted and the results are compared for each time. The k values for each of the three datasets, training, developing and evaluation, to achieve the best accuracy are not always the same. However, in backpropagation, we try to set the best numbers of nodes, as well as the minimum global errors for each of the hidden layer by experience and experiment. These values turn out to be the same when three data set are all obtaining their highest accuracy.

Here are values of error rates obtained in this experiment. For training data set, kNN gives 26.80% error rate when k is set to be 5. At the same time, Bp gives 34.59% error rate when numbers of nodes are both 25 for each layer, and the minimum global error is 10E-3. For develop test data set which is open-loop, kNN gives an error rate of 42.97% when k is 10, while Bp gives 38.76% under the same conditions as previous. For evaluation data set which is closed-loop, kNN shows an error rate of 47.53% when k is 6, while Bp shows 46.33% under the same conditions as previous. As we can see, for both open-loop and closed-loop set of data, Bp performs better than kNN. Neural network method shows a better result in this experiment.

In the aspect of computational speed, kNN runs much faster than Bp. Bp takes around 100 iterations to converge in a reasonable sense and to lower the error, we set the number of iterations to be 400.

Conclusions: There are both advantages and drawbacks for k-nearest neighbors algorithm and Backpropagation algorithm in this case. Bp performs better for both open-loop and closed-loop datasets. However, we can never simply draw a conclusion that deep learning algorithms are better than conventional algorithms in every espect. We learn from this experiment that performance of a certain kind of algorithm depends fairly on the properties of given dataset, and also depends on how we adjust the parameters in a certain algorithm. In this case, kNN uses concrete Euclidean distance and classify data based on feature similarity. However, even if we depend our decision so strongly on these measured distances, we can hardly tell how they can accurately describe the distribution properties of given data set. Bp makes decisions more obscurely so that the error of some single data will not be influential too much. Yet, we still need to worry about overfitting problems. Also, there is no perfect rule for one to establish a neural network with perfect values of layers, nodes, iterations and so on. The only way to obtain best parameters is to adjust them repeatedly until we get the best results.

In another hand, traditional machine learning algorithms are usually much simpler to implement and run pretty fast. However, a neural network is built on several layers and multiple nodes. It usually takes a large number of iterations to finish data training. Thus, it can be much more complicated and computational complex.

	Data Set		
Algorithm	**Train**	**Dev Test**	**Eval**
kNN(k=6)	29.78%	44.48%	47.53%
Bp	34.56%	37.48%	46.33%

Table 1. Best results (error rates) of each data set using kNN and Bp

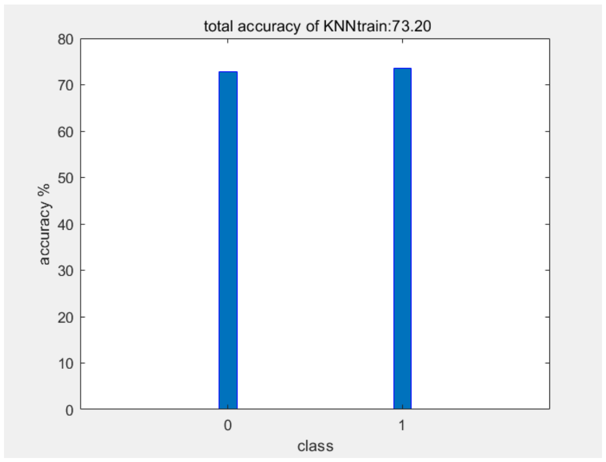


Figure 1. Total accuracy of KNN for training data

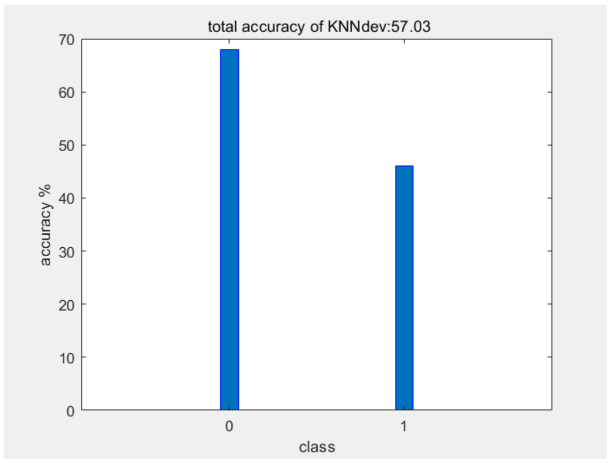


Figure 2. Total accuracy of KNN for developing data

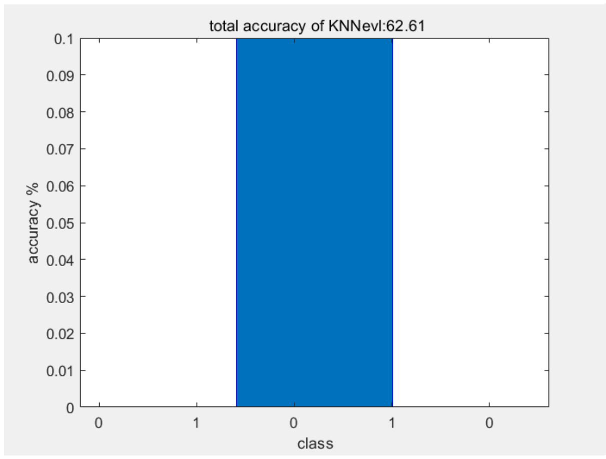


Figure 3. Total accuracy of KNN for evaluation data

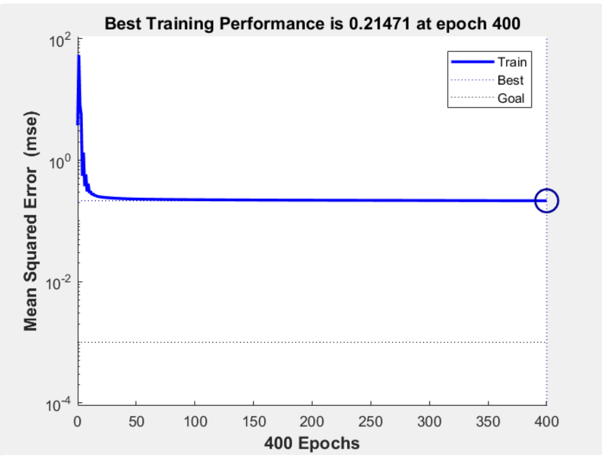


Figure 4. Epochs vs MSE

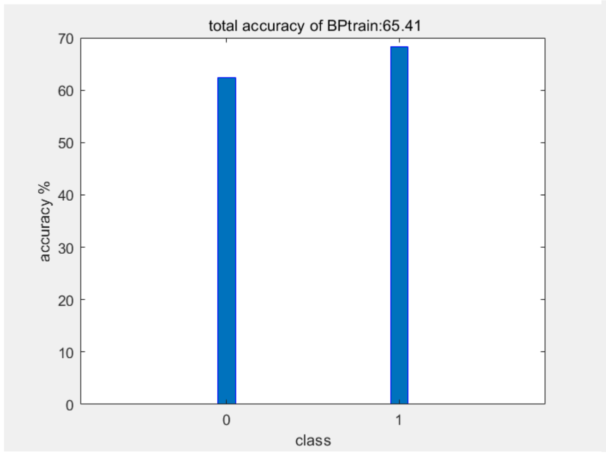


Figure 5. Total accuracy of Bp for training data

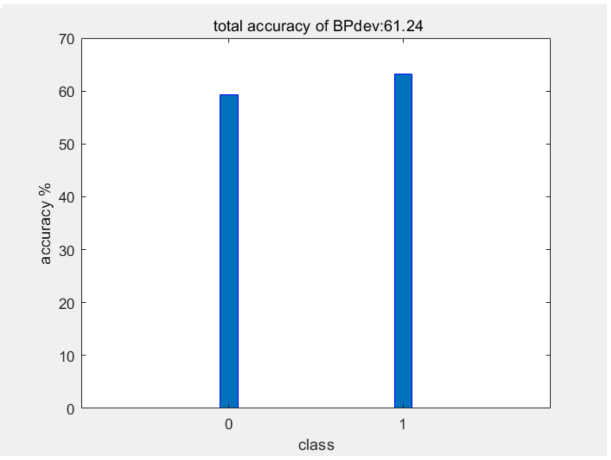


Figure 6. Total accuracy of Bp for developing data

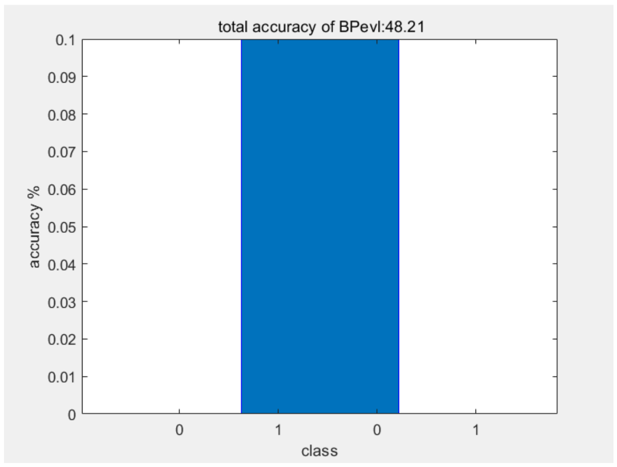
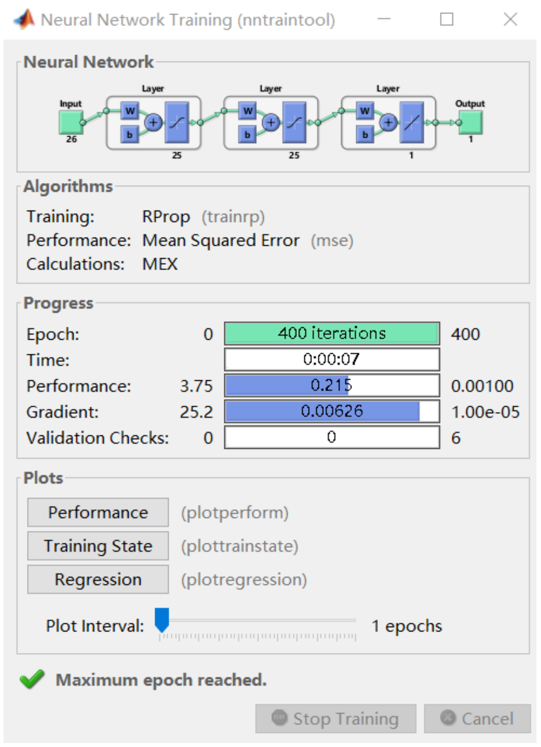


Figure 7. Total accuracy of Bp for evaluation data



Algorithms

Training: RProp (trainrp)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	400 iterations	400
Time:		0:00:07	
Performance:	3.75	0.215	0.00100
Gradient:	25.2	0.00626	1.00e-05
Validation Checks:	0	0	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Maximum epoch reached.

Figure 8. Neural Network tool interface