Thesis

Generative Topographic Mapping of Electroencephalography (EEG) Data

Submitted by

Navini Dantanarayana

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2014

Master's Committee:

Advisor: Charles Anderson

Asa Ben-Hur
Patricia Davies

Abstract

Generative Topographic Mapping of Electroencephalography (EEG) Data

Generative Topographic Mapping (GTM) assumes that the features of high dimensional data can be described by a few variables (usually 1 or 2). Based on this assumption, the GTM trains unsupervised on the high dimensional data to find these variables from which the features can be generated. The variables can be used to represent and visualize the original data on a low dimensional space. Here, we have applied the GTM algorithm on Electroencephalography (EEG) signals in order to find a two dimensional representation for them. The 2-D representation can also be used to classify the EEG signals with P300 waves, an Event Related Potential (ERP) that occurs when the subject identifies a rare but expected stimulus. Furthermore, unsupervised feature learning capability of the GTM algorithm is investigated by providing EEG signals of different subjects and protocols. The results indicate that the algorithm successfully captures the feature variations in the data when generating the 2-D representation, therefore can be efficiently used as a powerful data visualization and analysis tool.

Acknowledgements

I would like to extend my deepest gratitude to my advisor, Prof. Chuck Anderson, for his continuous thoughtful encouragement and careful guidance from the beginning to the very end of this exciting journey. I am also thankful to Prof. Asa Ben-Hur and Prof. Patricia Davies for their support in improving the final outcome of this endeavor. I'm blessed to have them as my committee.

My thanks also go to all the faculty, non-faculty members and the BCI group in the Department of Computer Science, for their direct and indirect assistance during this venture.

Last, but not least, I would like to thank my parents, family and loved ones for their encouraging words and unceasing support, without which this journey would not have been possible.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Brain computer interfacing (BCI) is an emerging field that focuses on bridging brain signals with external devices so that the devices can be controlled by brain activity. This idea was initiated in the mid 1960's by the United States Department of Defense in a program to help pilots with their air crafts so that pilots' mental workload can be reduced. However, the program was canceled due to unavailability of sophisticated technology to support this type of complex task at the time [5]. Since then, this idea has grown into a separate research area and has expanded to many disciplines.

One important application of BCI, and perhaps the biggest motivation for research, is providing assistance for people who are paralyzed or have lost control of their muscles in performing basic day to day tasks only with their thoughts. The main target groups are patients with spinal cord injury or patients with *Amyotrophic Lateral Sclerosis (ALS)*, which is an incurable, neurodegenerative disease that affects the ability to control muscle movements [6]. Many patients may still be capable of controlling facial muscles, which can be used as a trigger to control external devices. However, later stages of ALS can gradually lead to complete paralysis, which is known as the 'locked-in' syndrome, where the brain cells and intellectual functions are active but the nerve cells that carry signals from the brain to muscles have ceased functioning. In these cases, it would be helpful for the patients if they can use their brain signals to control external devices such as wheel chairs, mouse cursors or prosthetic limbs in order to perform simple tasks or communicate with others. This is where BCI comes into play. BCI systems can extract brain signals from the patients' brain and process them into a form that can be understood by a computer or other device.

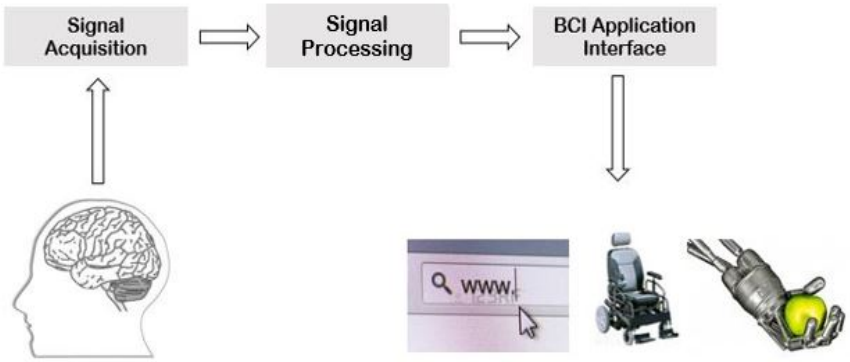


FIGURE 1.1. Basic design of a BCI System (reproduced using [1])

Constructing a BCI system basically consists of two parts: finding a brain signal that the subject can easily and reliably reproduce, and detecting, processing and analyzing these signals from an external device such as a computer software. Figure 1.1 illustrates the basic components of a BCI System.

There are two types of brain activity that are considered to produce reliable brain signals that can be used in the BCI systems. One is where the subject voluntarily performs some mental tasks that produce different brain patterns. These mental tasks can be simple arithmetic tasks or imagined motor tasks. The other type is where external stimuli are provided to the subject and the evoked responses of the brain for the stimuli are recorded. The subject is asked to focus his attention on the external stimulus, which can be auditory, somatosensory or visual, and the brain signal patterns that are evoked for the stimulus are used for the BCI system. The response of the brain for an external stimulus is known as an Event Related Potential (ERP). ERPs can be a direct result of either sensory, cognitive or motor events. Some ERPs used frequently in research are P300, a positive deflection in the EEG amplitude with a latency of about 300ms after a stimulus that is rare but expected, ELAN, a negative peak in the amplitude with a latency of about 200ms after a stimulus such as word structure errors, N400, a negative deflection in the amplitude with a latency

of about 400ms after a meaningful stimulus such as faces, words or pictures, and P600, a peak in amplitude after about 600ms for a stimulus such as reading or hearing grammatical errors or other syntactic anomalies [7]. This project uses P300 that is stimulated by a visual external stimulus, which in this case is displaying a letter on a computer screen.

To record the signals emitted from the brain, different acquisition systems are being used. The acquisition methods can be either invasive or non-invasive. One invasive method is Electrocorticography (ECoG), in which electrodes are placed above the cortex, beneath the dura mater of the brain. Non-invasive methods are electroencephalography (EEG), magnetoencephalography (MEG) and functional magnetic resonance imaging (fMRI) [8]. Even though non-invasive methods produce poor signal resolution because the skull dampens the signals, they are preferred over invasive methods because of lower risk to the subject and lower set-up cost. Out of the three non-invasive methods, EEG has been the most studied brain signal in order to interpret brain activity, mainly because of its fine temporal resolution, ease of use, portability and low set-up costs. EEG is the method used to acquire the data used in this project. More details about the P300 brain phenomena and the EEG acquisition method can be found in Chapter 2.

1.1. Challenges in BCI

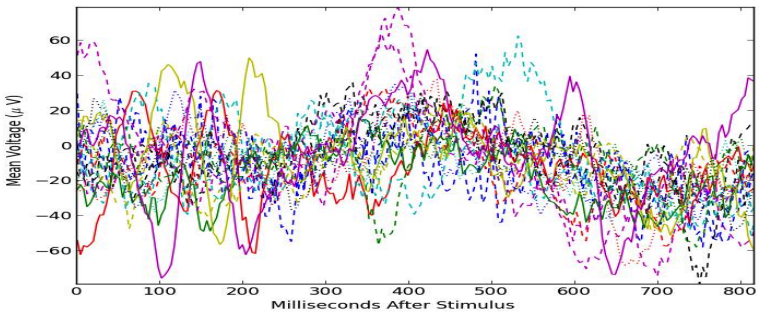
The main problem with BCI systems today is speed and accuracy. Speed of a BCI system is how fast the subject's thoughts can be translated to the intended output. Input-output delay is a problem in every BCI system from a cursor moving application to a wheelchair controlling application. This delay consists of the time taken to acquire and transfer brain signals and to process these signals to produce a meaningful output. Today, the most successful BCI systems work at a transfer rate of less than 30 bits per minute [9].

Accuracy of BCI solely depends on correctly classifying the brain signals to interpret the intended output. The biggest challenge with classifying EEG signals is that they are highly susceptible to noise. Brain signals inherently have very small signal-to-noise ratio and get coupled by a wide variety of noise sources. Noise can be electrical potentials that are generated from brain/motor activity that is not task related or electrical noise in the surrounding of the subject. Therefore, obtaining a pure brain signal for a task is nearly impossible.

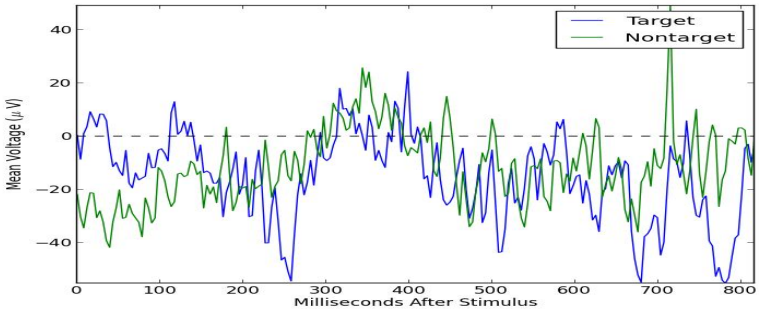
Furthermore, identifying a specific ERP from a distorted brain signal can be even more challenging. In addition to amplitude and latency variation of the P300 due to factors mentioned in Section 2.3, there are many other reasons why detecting the P300 wave from EEG can be a difficult task. Electrical potentials that are generated by facial muscle movements, eye movements and eye blinks create amplitude spikes, hence are major problems when recording P300 waves. Additionally, when the subject sees a target stimulus, other regions of the brain also start processing the perceived information, generating electrical fields on the scalp which can affect the significance of the P300 wave. Figure 1.2 illustrates the difficulty in identifying the P300s in some EEG signals. Figure 1.2a is a collection of 20 EEG signals that are evoked for a stimulus. But the P300 is hardly noticeable. Figure 1.2b–Figure 1.2d illustrate average of the same stimulus responses in Figure 1.2a and compare with average of non stimulus signals varying the number of signals that are averaged. As the number of averaged signals increases, the P300 in the stimulus signals becomes more apparent.

1.2. Research Questions for the Thesis

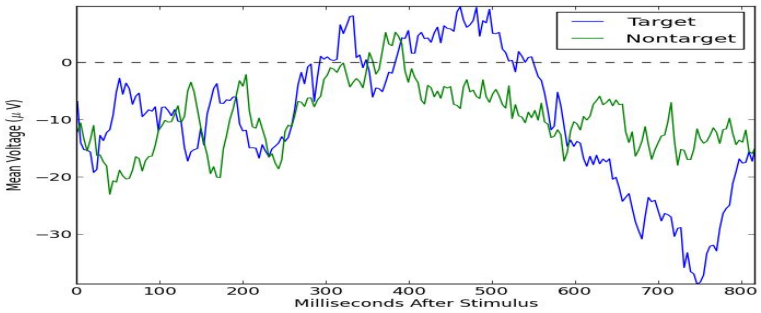
For applications that make use of the P300 wave, it is important to accurately identify when P300s are generated and when they are not. Therefore, precise discrimination of EEG



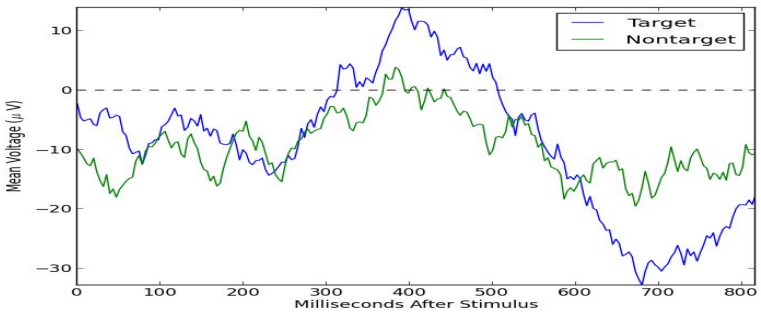
(A) EEG Signals of 20 ERPs for a stimulus. P300 is not very evident in many of the signals.



(B) One stimulus EEG and one non stimulus EEG



(C) Average of 10 stimulus and 10 non stimulus EEG



(D) Average of 20 stimulus and 20 non stimulus EEG

FIGURE 1.2. P300 waves in stimulus evoked EEG signals

signals with P300 waves from signals without P300 waves plays a major role in the success of BCI applications that uses the P300 wave. The idea explored in this thesis is whether the Generative Topographic Mapping algorithm can improve the classification accuracy of P300 signals. Additionally, it is investigated whether the GTM algorithm can produce a good 2 dimensional representation of the EEG signals, which can be used to visually analyze the high dimensional EEG data.

1.3. Related Work

Currently, there are many classification algorithms that have been applied to P300 signals. A comparison done between four linear methods: Pearson's Correlation Method (PCM), Fisher's Linear Discriminant (FLD), Stepwise Linear Discriminant Analysis (SWLDA) and Linear Support Vector Machine (LSVM) and one nonlinear method, Gaussian Kernel Support Vector Machine (GSVM) has concluded that SWLDA and FLD provide the best overall performance and implementation characteristics for practical classification of data [10] collected using the P300 Speller paradigm, originally described by Farwell and Donchin [11]. These data were collected from 8 healthy subjects. The average classification accuracy of all the algorithms were in the range of 80-90%, PCA falling behind at 80% and SWLDA and FLD reaching 90%. They also go on to say that in general, all of the algorithms were capable of adequately classifying the data. Another such study compares FLD, Stepwise Linear Discriminant Analysis (SWLDA), Bayesian Linear Discriminant Analysis (BLDA), LSVM, GSVM, a method based on Feature Extraction and Artificial Neural Network (NN) [12]. The data they use came from 12 subjects who are (partially) disabled suffering from amyotrophic lateral sclerosis (ALS), middle cerebral artery (MCA) stroke and subarachnoid hemorrhage (SAH). Their results have shown that nonlinear classifiers perform worse or equal to linear

classifiers, maybe because nonlinear classifiers tend to overfit the training data. The classification accuracy of all the algorithms of this research fell between 60-80%, BLDA and LDA taking the lead at around 80% accuracy. The poorest performer was the NN, giving about 60% maximum classification accuracy.

The GTM algorithm was first introduced as an alternative to the widely used Self Organizing Map (SOM) of Kohonen and to overcome most of the significant limitations of the SOM [3]. In the original paper, the algorithm was demonstrated on a problem of determining the fraction of oil in a multi-phase pipeline carrying a mixture of oil, water and gas. Since then, the algorithm has been used for numerous different applications. In [13] GTM is used to map the BioPharmaceutics Drug Disposition data data based on the compound solubility and degree of metabolism. GTM was also utilized for mapping of sparse data sequences in [14]. A GTM map has been created to visualize the distribution of text based documents. Another application where GTM was used is classification of bearing fault data [15]. GTM has been used in the area of BCI to represent the embedded dynamics of time-bounded EEG in the two dimensional latent space of the GTM model [16]. However, they haven't gone on to classify the EEG signals using the learned model.

1.4. Overview of this Thesis

In Chapter 2, the theoretical background of the concepts used for the project is presented. It starts off with explanations of Electroencephalography and P300 signals. Next, the theory of the GTM algorithm will be presented in detail. It explains the theory behind Radial Basis Function (RBF) networks and Gaussian Mixture Models (GMM), which are basically the building blocks of the Generative Topographic Mapping algorithm, and also brings it all

together to explain the GTM algorithm as a whole. Finally, the chapter goes on to elaborate the theory of Neural Networks, which is the classifier used in this project.

Chapter 3 is about the data being used and the preprocessing done on the data before they were applied to the GTM algorithm. Raw EEG data can be distorted with various noise added from potentials other than EEG, hence can produce incorrect results. Thus, it is important to minimize the effect from these artifacts before we try to process the signals. Preprocessing is done on the raw EEG data for this purpose. The chapter also goes onto give the implementation details of the GTM.

The results obtained by the experiments are presented in Chapter 4. The GTM algorithm is performed on EEG data to analyze whether it can help to improve the classification accuracy of P300 waves. Additionally, experiments were carried out to determine how well the GTM algorithm can detect inherent features of EEG signals of multiple subjects and multiple trials. Furthermore, EEG signals is analyzed to get an insight on how the aspects of the signal can affect the performance of the GTM algorithm.

Concluding discussions and directions for future work are given in Chapter 5. These include changes that can be done to the parameters of the GTM algorithm and other methods of using the GTM algorithm in BCI systems.

CHAPTER 2

Background

In this chapter, the background for the theoretical concepts used in the project will be explained. To start off, some background details of EEG, which is the brain signal acquisition method used and of P300 signals, which is the phenomenon we use in this project to classify brain signals, are provided. Then, the algorithm behind the Generative Topographic Mapping is explained. Later in the chapter, the basics of the Neural Network learning algorithm is presented, because Neural Networks are used in the experiments for classification purposes.

2.1. EEG Recording and P300 Signals

2.1.1. EEG Recording. Electroencephalography (EEG) is the recording of the electrical signals collected from the scalp. The first discovery of electrical potentials generated from a human brains dates back to 1929, made by the German psychiatrist Hans Berger. He announced that 'it is possible to record the electric currents generated in the brain, without opening the skull, and to depict them graphically on paper'. Berger also discovered that EEG depends on the mental state of the subject [17]. Hence, EEG signals can represent the brain activity of a patient. This discovery forms the foundation to non-invasive BCI systems. Figure 2.1 is a representation of EEG signals recorded from 8 channels placed on the scalp for about 3.9 seconds.

EEG records the fluctuations of the voltages along the scalp that results from ionic current flows within the neurons of the brain [18]. It has been found later that a significant part of the EEG signals are emanated from neurons in the cerebral cortex, which is a thin layer of neurons packed densely surrounding the two hemispheres of the brain. The electrical

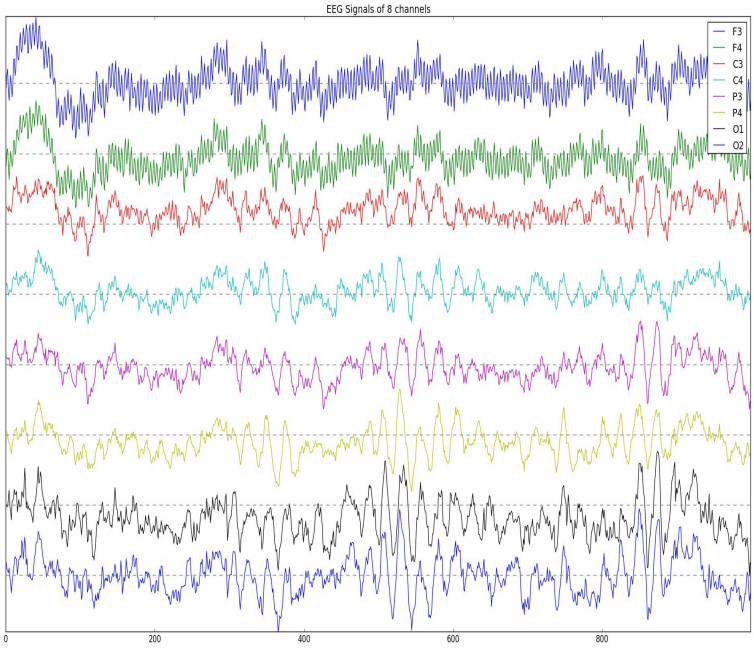


FIGURE 2.1. EEG signals recorded from 8 electrodes

potential generated from one neuron itself is not significant enough to record an EEG signal. Hence, EEG signals record an accumulation of electrical potentials of thousands or millions of synchronized neurons that have similar spatial orientation. Waves in the EEG signals are generated by cells that do not have similar spatial orientation. The EEG signal potentials are in the range of micro-volts, and the magnitude of the signals change with the state of the brain. When the subject is a asleep, the brain is resting and input from other sensory organs is minimal, the EEG signals have larger amplitudes because the neurons are more synchronized as there is minimal external disturbances. Also, the signals oscillate at a variety of frequencies, that have been shown to correlate with different states of behavior in the subject. When the brain is active higher frequencies are recorded, and when the brain is resting, the frequencies are lower.

EEG signals are recorded using electrodes placed on the subject's scalp. The most commonly used electrode type is surface electrodes, which are basically small metal discs that are placed directly on the scalp. The other types are needle electrodes, which are inserted under the scalp skin, and clip electrodes, which are a special form of surface electrodes that can be clipped to the earlobe. The types of metals that are generally used for electrodes are gold or silver, coated with a layer of silver chloride, platinum or other metal that does not chemically interact with the skin. They are made to optimize the electrical and mechanical contact and conductivity. The electrodes are connected to amplifiers, which filters and magnifies the recorded potentials. The output from the amplifier is sampled and stored for digital processing.

The placement of electrodes on the scalp is important because different regions of the cortex have different functionality, therefore, the position of the electrodes greatly affects the electrical activity recorded by the electrode. Hence, to avoid inaccurate recordings and to compare different recordings, a standard EEG electrode placement system was brought forward in 1947. The system is commonly known as the 10-20 system [19]. Figure 2.2 is a detailed illustration on how the electrodes are placed in the 10-20 system.

EEG can be used to record brain signals of voluntary mental tasks or evoked responses to stimuli. P300 is the most popular ERP used for BCI systems.

2.1.2. P300 Signal. The P300 wave is an event related potential (ERP) that is generated involuntarily when a subject identifies a task-relevant, or target, stimulus. The name 'P300' denotes that it is a positive deflection in amplitude in the EEG wave that occurs about 300-500ms after the stimulus. This wave is most dominant in the parietocentral area of the scalp [20]. The amplitude of the P300 wave is about $10\mu V$. Figure 2.3 plots an average of 20 target signals and 80 nontarget signals. Here, we can clearly see that averaged target

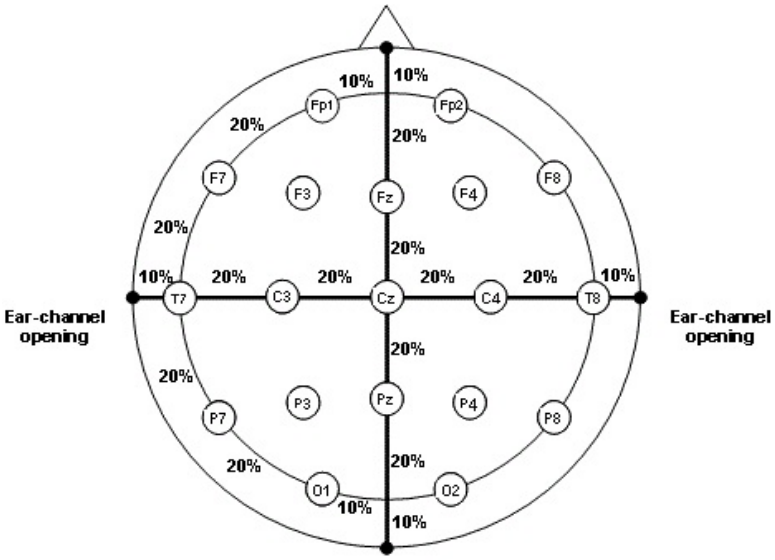


FIGURE 2.2. 10-20 system for electrode placement [2]

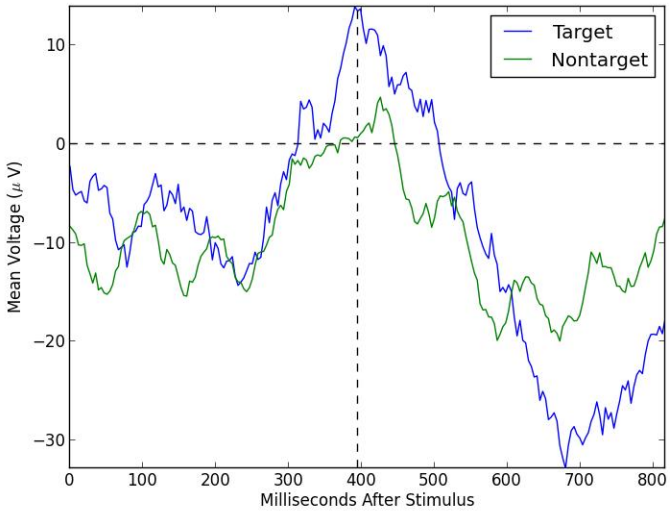


FIGURE 2.3. Average of 20 target signals and 80 non-target signals

signals has a significant positive deflection after around 400ms that is not present in the averaged non-target signals.

The amplitude of the P300 depends on many factors. The amount of conscious attention paid to the stimulus by the subject, task relevance of the stimulus, perceptual demands of other tasks (if any) running simultaneously with the primary task, probability of the target

stimulus over non-targets, difficulty in discriminating between target stimulus and non-target stimulus, subject's personality and lifestyle can be listed as few of the factors. Also, studies have shown that factors such as subject's age, lifestyle, uncertainty of the target stimulus can affect the latency of the positive peak of the P300. However, what phenomena in the brain generates scalp recorded P300 wave or what is the function of the P300 wave are still unanswered questions.

2.2. Generative Topographic Mapping Algorithm

The basic idea behind the GTM algorithm is to find a low dimensional representation of data that lies in a high dimensional space. The low dimensional space is initially unknown, hence latent. Usually the latent space is selected to be either one or two dimensional. The mapping is achieved by considering a nonlinear function $\mathbf{y(x;W)}$ which maps points in the latent space to corresponding points in the data space. Here $\mathbf{W}$s are the parameters that govern the mapping. The nonlinear mapping of data from the latent space to the higher dimensional data space is usually done by a Radial Basis Function (RBF) network. The latent space points are mapped to the observed high dimensional space via the smooth function. This results in a low dimensional manifold embedded in the high dimensional data space. However, it is highly improbable that the actual data points will be restricted only on the low dimensional manifold. Therefore, a noise model is added. For real valued data, it is convenient to use a spherical Gaussian, making the manifold a mixture model of Gaussians. The low dimensional probability distribution, smooth map and noise functions are optimized by the Expectation Maximization (EM) algorithm.

In the following subsections, the theoretical aspects of RBF networks, Gaussian Mixture Models and GTM algorithm are explained.

2.2.1. Radial Basis Functions (RBF) Network. The Radial Basis Function (RBF) network is a form of an artificial neural network (Section 2.3). It uses Radial Basis Functions as its activation functions. That is, instead of computing nonlinear functions of the scalar product of the input vector and a weight vector, the activation of the hidden units in an RBF network is given by a nonlinear function of the distance between the input vector and a weight vector.

RBF networks have a two-stage training procedure which is considerably faster than the methods used to train multi-layer perceptrons. This property makes RBF networks more attractive than multi-layer perceptrons. In the first-stage, the parameters of the basis functions are set so that they model unconditional data density. In the second stage, the weights in the output layer are determined, and this is a quadratic optimization problem, which can be solved efficiently using methods from linear algebra. The Generative Topographic Mapping (GTM) needs a nonlinear mapping algorithm to map the latent space points to data space, and should be retrained at each iteration of the EM algorithm used to train the GTM. Since RBF networks are faster than other nonlinear modeling algorithms because of above mentioned reasons, they make an ideal candidate for GTM to keep the execution time low [21].

The RBF network has three layers: an input layer (x), a hidden layer with a nonlinear RBF activation function (ϕ) and a linear output layer (y). It can be written as follows:

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x) + w_{k0} \quad \text{where} \quad \phi_j = \phi(||x - c_j||)$$

Here, w_{kj} are the output layer weights and M is the number of hidden units. c_j is the center vector for hidden unit j. Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name radial basis function. The

norm is typically taken to be the Euclidean distance and the radial basis function is commonly taken to be Gaussian:

$$\phi(||x - c_j||) = e^{[-\beta||x - c_j||^2]}$$

Since the basis functions are positive definite, we can ensure that the network as a whole can approximate probabilities. The common approach is that the sum of all the basis functions $\sum_{j=1}^{M} \phi_j$ should form a representation of the unconditional probability density of the input data. If we assume we are solving a classification problem with c classes and use M density functions,

$$p(x|C_k) = \sum_{j=1}^{M} p(x|j)P(j|C_k), \quad k = 1, \ldots, c$$

To find the unconditional data density, we need to sum over all the classes,

$$\begin{aligned} p(x) &= \sum_{k=1}^{c} p(x|C_k)P(C_k) \\ &= \sum_{k=1}^{c} \sum_{j=1}^{M} p(x|j)P(j|C_k)P(C_k) \\ &= \sum_{j=1}^{M} p(x|j)P(j) \quad \text{where} \quad P(j) = \sum_{k=1}^{c} P(j|C_k)P(C_k) \end{aligned}$$

Then we can compute the posterior probabilities of class membership by Bayes' theorem,

$$P(C_k|x) = \sum_{j=1}^{M} w_{kj}\phi_j(x)$$

where the basis functions ϕ_j are given by

$$\phi_j = \frac{p(x|j)P(j)}{\sum_{j'=1}^{M} p(x|j')P(j')}$$

$$= P(j|x)$$

and the second layer weights are given by

$$w_{kj} = \frac{P(j|C_k)P(C_k)}{P(j)}$$

$$= P(C_k|j)$$

2.2.2. Gaussian Mixture Models and the EM Algorithm. Gaussian Mixture Models (GMM) is a method of probabilistic clustering of data. Here, it is assumed that the data points are generated by K normal distributions, each with their own mean and covariance matrix. The proportion of points coming from each Gaussian is governed by a prior $\pi = (\pi_1, \ldots, \pi_k)$. If we consider a data set $D = x_n, \quad n = 1, \ldots, N$ and assume that the observed data points are generated from a single Gaussian independently,

$$p(D|\mu, \Sigma) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \Sigma)$$

This is a function of the parameters explaining the conditional probability of the data being generated by the Gaussian given its parameters. This function is known as the likelihood function. The objective is to maximize this probability. It is equivalent to maximizing the log likelihood function.

Since the Gaussians are linearly super-positioned, the probability of each data point depends on all the contributing Gaussians.

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1 \quad 0 \leq \pi_k \leq 1$$

The mixing coefficients, π_k can be interpreted as prior probabilities, $p(k)$, giving

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k)$$

So to generate a data point, first we need to pick one of the Gaussian components with probability π_k, and then draw a sample x_n from that component. These two steps need to be repeated for each new data point.

But, our problem at hand is the inversion of this process, that is, given the data set, we need to find the corresponding parameters: mixing coefficients, means and covariances of the Gaussian components. If we knew which component generated each data point, the maximum likelihood solution would involve fitting each component to the corresponding cluster of data points. However, the problem is when the data set is unlabeled. Then we need to find the parameters unsupervised. Since the labels are hidden, we can refer to them as latent variables.

For a given data point x, we can evaluate the corresponding posterior probabilities, which is known as the responsibilities. This is the probability of each Gaussian component, given the data point x.

$$\begin{aligned} p(k|x) &= \frac{p(k)p(x|k)}{p(x)} \\ &= \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)} \end{aligned}$$

Our goal is to maximize the log likelihood function, to maximize the likelihood that the data is generated from the given Gaussian. The log likelihood function takes the following form,

$$ln\ p(D|\pi,\mu,\Sigma) = \sum_{n=1}^{N} ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)\}$$

Direct maximization of the log likelihood function is quite difficult numerically, because of the sum of terms inside the logarithm. The likelihood function gets larger as we add more Gaussian components to the model. There is, however, a simpler approach to maximize this function. It is known as the Expectation-Maximization (EM) algorithm [22].

The EM algorithm maximizes the function in two steps which are invoked alternatively until convergence. First, initial guesses are made for the parameters. Next, the E-step is carried out, where the responsibilities are calculated using the initial parameters. Then the M-step is carried out, where the parameters are updated using the maximum likelihood result. The E-step and the M-step are invoked alternatively until the likelihood converges to a maximum. Each EM cycle guaranteed not to decrease the likelihood.

2.2.3. Generative Topographic Mapping Algorithm. The GTM model finds a representation for the distribution $p(t)$ of data in a D-dimensional space $t = (t_1, \ldots, t_n)$ in terms of a number L of latent variables $x = (x_1, \ldots, x_L)$ [3]. This is done by considering a function $y(x; W)$ which maps points x in the latent space into corresponding points $y(x; W)$ in the data space. The parameters that govern the mapping, W could be weights and biases. Since dimensionality reduction is highly preferred, we consider the dimensionality L of the latent space to be much less than the dimensionality D of the data space. For visualization purposes L is restricted to be either 1 or 2. In this implementation, the dimensionality of the latent space L is 2. The transformation $y(x; W)$ then maps the latent-variable space into an

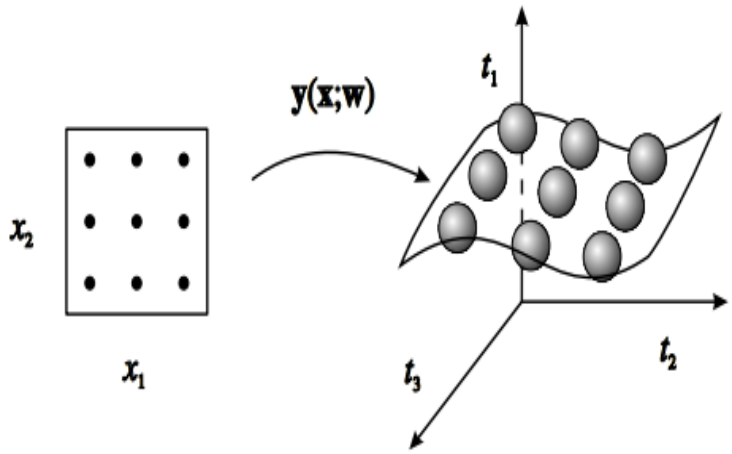


FIGURE 2.4. Mapping from 2 dimensional latent space to a 3 dimensional latent space [3]

L-dimensional non-Euclidean manifold S embedded within the data space. Figure 2.4 shows an illustration where a 2 dimensional latent space is mapped to a 3 dimensional data space.

In the GTM algorithm, the function $y(x; W)$ is chosen to be a RBF network. Therefore,

$$y(x; W) = W\phi(x)$$

The theoretical aspects of an RBF network are explained in Section 2.1.

The probability distribution on the latent-variable space is $p(x)$, which is the prior distribution of x. This introduces a corresponding distribution $p(y|W)$ in the data space. Since $L < D$, the distribution in t-space would be confined to the L-dimensional manifold and hence would be singular. However, in reality, the data will only approximately live on a lower-dimensional manifold, so it is appropriate to include a noise model for the t vector. We can choose the distribution of t, for a given x and W, to be radially symmetric Gaussian centered on $y(x; W)$ having variance β^{-1} so that

$$p(t|x, W, \beta) = (\frac{\beta}{2\pi})^{\frac{D}{2}} e^{\{-\frac{\beta}{2}||y(x;W)-t||^2\}}$$

The distribution in t-space, for a given value of W, is then obtained by integration over the x-distribution

$$p(t|W,\beta) = \int p(t|x,W,\beta)p(x)dx$$

For a given data set $D = (t_1, \ldots, t_N)$ of N data points, we can determine the parameter matrix W, and the inverse variance β, using maximum likelihood. In practice it is convenient to maximize the log likelihood, given by

$$\mathcal{L}(W,\beta) = ln \prod_{n=1}^{N} p(t_n|W,\beta)$$

Here, the prior distribution $p(x)$ is chosen to be a sum of delta functions centered on the nodes of the regular grid in the 2 dimensional data space.

$$p(x) = \frac{1}{K} \sum_{i=1}^{K} \delta(x - x_i)$$

Each point x_i is then mapped to the data space by $y(x_i; W)$, which forms the center of a Gaussian density functions. So the distribution function in data space takes the form

$$p(t|W,\beta) = \frac{1}{K} \sum_{i=1}^{K} p(t|x_i,W,\beta)$$

and the log likelihood function becomes

$$\mathcal{L}(W,\beta) = \sum_{n=1}^{N} ln\{\frac{1}{K} \sum_{i=1}^{K} p(t_n|x_i,W,\beta)\}$$

We can see that, for a particular noise model $p(t|x,W,\beta)$, the distribution $p(t|W,\beta)$ corresponds to a constrained Gaussian Mixture Model and the centers of the Gaussians are given by $y(x_i; W)$. Hence, we can use the EM algorithm as explained in Section 2.2.2,

to maximize the log likelihood. Also, since the mapping function $y(x;W)$ is smooth and continuous, the projected points $y(x_i;W)$ will have a topographic ordering such that any two points x_A and x_B which are close in the latent space will map to points $y(x_A;W)$ and $y(x_B;W)$ which are close in the data space.

As explained in Section 2.2.2, the next step is to determine the GMM parameters using the EM algorithm. The first step is to calculate the posterior probabilities using assumed values of the parameters. Let's refer to these initial parameters as W_{old} and β_{old}. So the responsibility can be calculated using

$$\begin{aligned} R_{in}(W_{old},\beta_{old}) &= p(x_i|t_n,W_{old},\beta_{old}) \\ &= \frac{p(t_n|x_i,W_{old},\beta_{old})}{\sum_{i'=1}^{K} p(t_n|x_{i'},W_{old},\beta_{old})} \end{aligned}$$

The next step is to calculate the log likelihood using the responsibilities and carry out the M-step, where W and β which maximizes the likelihood are calculated. Expectation of the data log likelihood is

$$\langle \mathcal{L}_{comp}(W,\beta) \rangle = \sum_{n=1}^{N}\sum_{i=1}^{K} R_{in}(W_{old},\beta_{old}) ln\{p(t_n|x_i,W,\beta)\}$$

Maximizing the log likelihood with respect to W gives

$$\sum_{n=1}^{N}\sum_{i=1}^{K} R_{in}(W_{old},\beta_{old})\{W_{new}\phi(x_i) - t_n\}\phi^T(x_i) = 0$$

This can be written in matrix notation in the form

$$\boldsymbol{\phi^T G_{\text{old}} \phi W^T_{\text{new}} = \phi^T R_{\text{old}} T}$$

where $\boldsymbol{\phi}$ is a K x M matrix with elements $\phi_{ij} = \phi_j(x_i)$, $\boldsymbol{T}$ is a N x D matrix with elements t_{nk}, $\boldsymbol{R}$ is a K x N matrix with elements R_{in}, and $\boldsymbol{G}$ is a K x K diagonal matrix with elements

$$G_{ii} = \sum_{n=1}^{N} R_{in}(W, \beta)$$

Similarly, maximizing the log likelihood with respect to β gives

$$\frac{1}{\beta_{new}} = \frac{1}{ND} \sum_{n=1}^{N} \sum_{i=1}^{K} R_{in}(W_{old}, \beta_{old}) ||W_{new}\phi(x_i) - t_n||^2$$

By solving these, we can obtain W_{new} and β_{new}. Then, these values are substituted to find the responsibilities and again the M-step is carried out using the new responsibilities. These two steps are alternatively carried out until the log likelihood converges to a maximum.

GTM uses a probabilistic model to represent the data, and hence results in benefits such as handling missing data values by simple modification of the EM algorithm and visualization of data from the low dimensional modeled distribution.

After learning the model, the probability density functions (PDF) calculated for a given data set both in the input and the latent spaces can feed a Bayesian classification model [13]. Using Bayes' theorem, we can find the conditional probability of a class given the latent space node because, in GTM, all latent nodes share responsibility of each data points. These conditional probabilities can then be used for a test set instance to estimate the class probabilities. The class with the largest probability is assigned to the test instance.

2.2.4. Visualization. One advantage of GTM is that we can visualize the high dimensional data in a low dimensional latent space. To use GTM for visualization, it is necessary to map data points t_n to corresponding points in latent space. We can do this using the posterior density $p(x|t_n)$. Since our prior distribution was chosen as a summation of delta

functions, this posterior density is given by a sum of delta functions centered at the lattice points x_j with weights given by the responsibilities R_{jn}. However, in order to visualize the whole data set in a single plot, we need a statistic to summarize the distribution. One convenient method is to use the mean:

$$\langle x|t_n, W, \beta\rangle = \sum_{j=1}^{K} R_{jn}x_j$$

However, the mapping is nonlinear, and the posterior probability distribution can be multi-modal, hence the mean might be a misleading representation. An alternative choice is the posterior mode, given by

$$j_{max} = \arg\max_j R_{jn}$$

2.3. Neural Networks

Neural networks are computational algorithms that were inspired by the brain's ability of recognizing pattern and learning from them. The network consists of a set of interconnected 'neurons', mimicking the brain, and these neurons hold computational capability that enables them to take in an input and produce an output. A neural network consists of three layers, the input layer, one or many hidden layers and the output layer [23]. Each layer consists of nodes or neurons. Each node in the input layer is connected to each node in the hidden layer, and each node in the hidden layer are connected to each node in the output layer or each node in the consecutive hidden layer. Figure 2.5 is an illustration of a one hidden layer neural network.

Each connection between the hidden layer and the output layer carry an adaptive weight, a numerical coefficient that is tuned during the learning process. The data traveling on the connection gets weighted based on this value. The nodes in the hidden layer are capable of

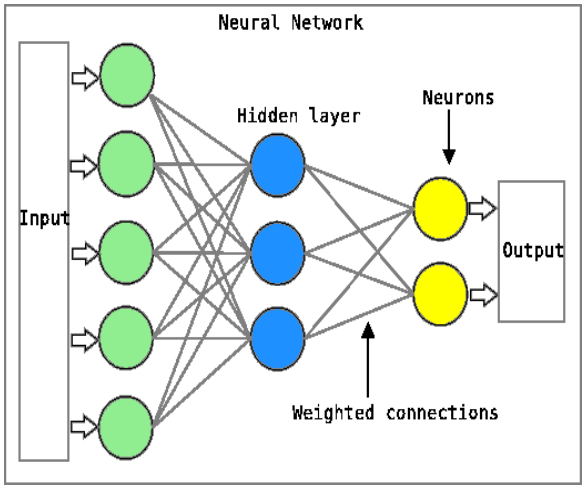


FIGURE 2.5. Neural network with one hidden layer [4]

approximating nonlinear functions (known as activation functions) of their inputs to produce their output.

For the classification purposes of this project, a single hidden layered neural network was used. The following subsection discusses the theory of the network.

2.3.1. NETWORK FUNCTION. Let the input to the neural network be $\boldsymbol{X}$, the weights of the connections from the input layer to the hidden layer be $\boldsymbol{V}$, the activation function of each hidden layer node be h, the output from the hidden layer be $\boldsymbol{Z}$, the weights of the connections from the hidden layer to the output layer be $\boldsymbol{W}$ and lastly, the output from the output layer of the network be $\boldsymbol{Y}$. Therefore, the relationships can be presented as below.

$$\boldsymbol{Z} = h(\boldsymbol{X}^T\boldsymbol{V})$$

$$\boldsymbol{Y} = \boldsymbol{Z}^T\boldsymbol{W}$$

$$\text{hence,} \qquad \boldsymbol{Y} = (h(\boldsymbol{X}^T\boldsymbol{V}))^T\boldsymbol{W}$$

The activation function, h, is chosen to be the hyperbolic tangent function, which ranges from -1 to 1.

$$h(a) = tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \qquad \text{where} \qquad a = \boldsymbol{X}^T \boldsymbol{V}$$

In this project, we need to use the neural network as a classifier. If the values produced by the output layer can be thought of as probabilities, we can assign a class to each output node, so that the class of the input sample can be that of the output node which produces the maximum probability for that sample. However, neural networks can't be used directly for this because the output of the neural network is not guaranteed to be between 0 and 1 and sum to 1. Therefore, we can do a modification as follows.

$$g_k(\mathbf{x}_n) = \begin{cases} \dfrac{e^{y_k(\mathbf{x}_n)}}{1 + \sum_{m=1}^{K-1} e^{y_m(\mathbf{x}_n)}}, & \text{for } k = 1, \ldots, K-1 \\ \dfrac{1}{1 + \sum_{m=1}^{K-1} e^{y_m(\mathbf{x}_n)}}, & \text{for } k = K \end{cases}$$

$g_k(\mathbf{x}_n)$ is the value at the k^{th} output node for the n^{th} input sample. The class of the input sample is determined by the node that produces the maximum output value.

Now that we have proper outputs, we need to train the network until it classifies the data well. We are using the supervised neural network, therefore, the expected outputs ($\boldsymbol{T}$) are provided during training. After the network produces an output by passing the input through the weighted connections and hidden layer nodes, the calculated output is compared to the expected output. The mean squared error between the calculated and the expected output can be found as following,

$$E = \frac{1}{N}\frac{1}{K}\sum_{n=1}^{N}\sum_{k=1}^{K}(g_{n,k} - t_{n,k})^2$$

where N is the number of samples in the input $\boldsymbol{X}$, and K is the number of dimensions (or number of classes) in the output $\boldsymbol{Y}$.

The network should be trained with backpropagation until the log likelihood function is maximized [24]. During training, the weights of the network are adjusted to minimize E. Since the error function is not a linear function of the weights, we cannot use the simple derivative equals zero method to find the parameters that minimizes E. Instead, we use the Scaled Conjugate Gradient Descent algorithm to find values for $\boldsymbol{V}$ and $\boldsymbol{W}$ that minimizes the error.

2.3.2. Scaled Conjugate Gradient Descent. Conjugate Gradient Descent methods are ways of finding a local minimum of a function by changing the parameters to take a step in the direction of the negative gradient. Scaled conjugate gradient descent algorithm uses this principle, and additionally it decides how far of a step the parameters should take in one iteration in order to reach the minimum of the function fast.

Conjugate gradient descent methods make use of the second derivative of the goal function rather than the first derivative, which is used by simpler techniques of function minimization/maximization. Usage of second order derivative allows to find a better way to a (local)minimum than the first derivative, but at higher computational costs. Furthermore, conjugate gradient descent methods are different from standard gradient descent methods because conjugate gradient descent techniques proceed in a direction which is conjugate (orthogonal) to the directions of the previous step, therefore these produce non-interfering directions of search. This implies that in the k^{th} iteration, the error has been minimized over the whole subspace spanned by all previous search directions [25]. And in Scaled Conjugate Gradient Descent, the parameter update values are found by assuming the error function is

locally quadratic, approximating the second derivative of the error function, and solving for the new value of weights that would minimize the quadratic function [26].

CHAPTER 3

Experimental Design

This chapter describes the data that is used in the experiments of this projects and the preprocessing done on the data before any algorithm was applied to them. Then, the implementation details of the GTM algorithm are also provided in the last Section of the chapter.

3.1. EEG Data

All the EEG signals used in this project are recorded by the Brain-Computer Interface (BCI) research group at Colorado State University. This data is publicly available from their website [27]. The database contains data recorded using three different EEG systems, Biosemi ActiveTwo which has 32 channels, g.Tec g.GAMMAsys with 8 channels and Neuropulse Mindset with 19 channels. Data has been collected from both impaired subjects recorded in the home and nonimpaired subjects recorded in the lab. The database contains data that was recorded from 8 different protocols, where each protocol required the subjects to either perform a mental task or concentrate on an external visual stimulus.

For the experiments in this project, we have decided to use data collected using the g.Tec g.GAMMAsys. This system has been enhanced to avoid or reduce artifacts from movements and electromagnetic interference [28]. The electrodes can be fixed on to a cap that is made from a flexible material so that it is comfortable to wear and fits for most people. An image of the system is shown in Figure 3.1. To improve the conductivity of the electrodes, a drip of gel should be injected to the electrodes. The system has only 8 channels. Therefore data available has been recorded using channels positioned at F3, F4, C3, C4, P3, P4, O1 and O2. Out of these 8 channels, our experiments will be only using P3 and P4 electrodes because

FIGURE 3.1. g.Tec g.GAMMMAsys EEG recording cap with electrodes

the P300 signal is most prominent in the parietal-central part of the brain. The sampling rate of the device is 256Hz, therefore a sample is taken approximately every 0.0039 seconds.

Since we are focusing on the P300 signal of the EEG data we need a protocol which stimulates an ERP that includes P300 in the brain. Hence we are using data recorded by the letter-p, letter-d and letter-b protocols. These are trials which provided the subjects with an external visual stimulus and recorded their brain signals. Each subject was seated in front of a computer screen and asked to count the number of occurrences of a given letter within a sequence of flashing characters containing 20 target and 60 non-target characters. And, the EEG signals were recorded during the whole 80 character sequence, therefore, each dataset contains 80 trials, 20 targets and 60 non-targets. This process was repeated three times with the target letters b, d and p, which were selected to represent a difficult scenario. The letter being presented in the center of the computer screen is indicated in the stimulus channel of the data. For the g.tec g.GAMMAsys data files, this is the ninth channel. The entire session was repeated on three separate days using a different EEG system during each session. Upon completion of the final session, each user completed a questionnaire regarding their experience.

The EEG database contains g.GAMMAsys recorded data of 9 nonimpaired, able-bodied subjects recorded in the lab and 4 impaired subjects recorded at their homes. Out of these, we have selected 4 non-impaired (S20, S21, S23 and S24) and 3 impaired (S11, S15 and S16) subjects' data to run our experiments on.

3.2. Preprocessing

Each g.GAMMAsys EEG data set consists of 9 rows for the 9 channels (8 channels+stimulus channel) and about 17,695 columns indicating the signal potential at each sample recorded by the device. From these data, we need to extract the data samples recorded when the target letters were presented and when the non-target letters were presented. The stimulus channel indicates which character was shown on the computer screen in which sample during the trial. The time window between each stimulus averages to about 215 samples, approximately 840ms (the sampling rate of the g. GAMMAsys is 256Hz), and the minimum length is 212 samples. So we have limited the number of time samples in the window for each stimulus to 210. The window is enough to see the P300 signal because it occurs around 300-400ms after the stimulus, which converts to approximately 75-103 samples after the stimulus. So the extracted dataset for a particular subject, one protocol, one channel consists of 80 stimulus points, 20 targets and 60 nontargets, with 210 time samples each.

3.2.1. Smoothing by Regularization. The EEG signals are noisy by nature. Figure 2.1 is a sample of EEG data and we can see how fluctuating the signals can be. Amplitude and frequency of the signals vary rapidly causing irregularity of the signals. These fluctuations are due to the noise being added to the EEG signal. Some sources of noise are frequency interference from environment, skin potentials and noisy signals from brain itself. These noisy signals can cause errors when trying to process and find a pattern in the signal.

Since our experiments focus on how well the GTM algorithm can identify inherent features of the EEG signals, it is very important that the signals are less noisy while still preserving the important characteristics, so that the algorithm won't be mislead. The main feature we are looking for in the EEG signal is a positive peak around 300ms after the stimulus though there may be other less well-known features of the signal that relate to the target stimulation. Smoothing the signals will emphasize this peak by smoothing out other variances of the signal. Hence, a smooth fit to the signal is preferred over the original noisy signal.

To find a smooth fit to our EEG signals, we have used a regularization method.

$$\sum_{i=1}^{m}(y_i - z_i)^2 + \lambda \sum_{i=1}^{m}(Dz)^2 \qquad \text{where} \qquad Dz = \Delta^2 z$$

The basis of this method is to find a new, smoothed function z to replace the noisy function y, by minimizing both the curvature of y and the squared error between y and z at the same time [29–32]. By minimizing the error between the original and the smoothed version of the signal ($\sum_{i=1}^{m}(y_i - z_i)^2$), we ensure that the patterns of the signal is preserved because the smoothed signal should always be in close proximity to the original signal. By minimizing the curvature of the smoothed signal ($\lambda \sum_{i=1}^{m}$), we ensure that the smoothed signal does not fluctuate rapidly as the original signal we need to smooth. λ is a parameter chosen depending on how smooth we want our new signal to be. Larger the λ, the signal will be far apart from the original signal, changing its shape.

3.2.2. Amplitude Modification. Different EEG signals may be shifted along in voltage due to various potentials. This might lead to incorrect training of the model which learns patterns of the EEG signals. So to rectify this, all signals are brought individually to zero mean, so that we can ensure the amplitudes of the peaks are comparable in all the

EEG signals. To normalize one signal of 210 samples, the mean (μ) of the 210 samples is calculated. Next each sample (x) in the signal is changed as follows to obtain the modified sample (z).

$$z = x - \mu$$

3.3. Implementation

The project implements the GTM algorithm in python. All the experiments were run on machines with Linux 3.16 Intel (R) Core (TM)2 Duo CPUs. The algorithm is utilized to analyze EEG data and to see if it can separate P300 wave signals and non-P300 signals. The ultimate goal is that the GTM model will learn the inherent feature differences between signals with P300 waves and signals without, unsupervised, and they will be separately clustered as two separate Gaussians in the latent space. Since the subject is focusing on recognizing the target letter in the experiments, their EEGs should record a P300 wave when the target letter is displayed on the screen and no P300 for nontarget letters. This is the basis for training the model.

The summary of the learning algorithm can be expressed as follows.

(1) Generate the grid of latent points $\{x_k\}; k = 1, ..., K$

(2) Generate the grid of basis function centers for the RBF network $\{\mu_m\}; m = 1, ..., M$

(3) Select the basis function width σ

(4) Compute the matrix of basis function activations Φ

(5) Initialize W using Principal Components

(6) Initialize β (variance of the noise added)

(7) Train by alternating between E-step and M-step

   (a) Evaluate log likelihood at the end of each cycle for convergence

(8) Plot the prior probabilities on a two dimensional latent space using a visualization method

3.3.1. Parameter Initialization. Before applying the data to the algorithm, we need to set the parameters that govern the initial execution. The main parameters we need to initialize are number of latent space points (K) and the latent space grid points, number of hidden units in the RBF network (M), Gaussian basis function centers of the RBF network (μ), Gaussian basis function width of the RBF network (σ), weights of the mapping from latent space to data space (W) and variance of the noise added (β). As for K and M, we can experimentally choose values that give good output. The latent space grid points are selected such that they are equally spaced in a latent space that span [(-1,-1), (-1,1), (1,-1), (1,1)].

There are two ways of selecting the centers of an unsupervised RBF network. The simplest approach is to randomly select a number of training examples as RBF centers. Alternatively, RBF centers may be obtained with a clustering procedure such as k-means [33]. In this project we have stuck with the simpler method. Since our initial training data are the K latent space grid points, we have selected the M number of centers to be a sample from that. The widths (σ) are set to be the largest squared distance between any two centers.

The original algorithm publications [34, 3] suggest two ways of initializing the weights, W and the variances, β. One such possibility is to draw the weights from a Gaussian distribution with zero mean and with expected variance over y equaling to the variance of the training data. The other possibility, which is often the better alternative, is to initialize the weights so that the L latent variables map to the L-dimensional hyperplane spanned by the L first principal components of the data set. If the weights are initialized randomly, then the variances are set to reciprocal of the average squared distance between the Gaussian

mixture centers and the actual data points. If the weights are set using PCA, the variances are set so that its inverse equals to the larger of either $(L+1)^{\text{th}}$ principal component of the data set, which is the largest variance orthogonal to the L-dimensional hyper-plane on the Gaussian mixture components lie, or half the average minimal distance between the mixture components.

3.3.2. Classification and Testing. After visualizing the prior probabilities on the latent space, the two dimensional latent space can be used for classification. For the classification, neural networks are used [23]. A neural network with two input nodes and two output nodes is sufficient for classifying the latent space points. For comparison, the pre-processed raw EEG data were also classified using a neural network of 210 input nodes and 2 output nodes. The number of hidden units that will produce the optimal results will be determined experimentally.

For training and testing of the neural network, the leave-one-out cross validation technique was used. The main reason for this is that the datasets used do not have many data samples. During L-O-O CV, the neural network is trained using all data samples in the dataset except one, and the left out data sample is used to test the network. This process is repeated until all data samples in the data set have been used for testing. The accuracy is calculated on how the test data sample was classified.

The accuracy measurement used in this project is the Balanced Success Rate (BSR), which takes classification accuracy of each class and calculates the average of them to give the final accuracy. It is important to do so in our case, because the number of samples in the two classes are unequal, 20 targets and 60 nontargets, hence we need to give equal weight to

both classes when calculating the accuracy. The formula for BSR is shown below.

$$BSR = \frac{\frac{TruePositive}{Positive} + \frac{TrueNegative}{Negative}}{2}$$

True positive (negative) value is the number of positive (negative) samples, that were correctly identified as positive (negative) during testing. The positive (negative) values are the total number of positive (negative) samples in the dataset. The classification accuracies presented in this thesis are averages of 10 runs of the L-O-O CV algorithm. In each of the runs, the weights of the neural network were reinitialized to values drawn from a uniform distribution between -0.1 and 0.1.

CHAPTER 4

Experimental Results

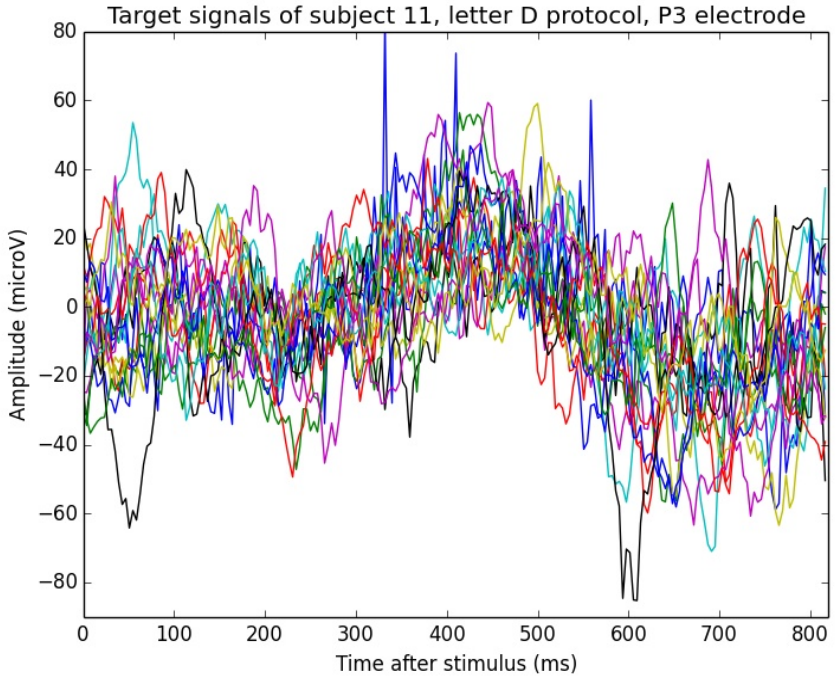
In this chapter, the results obtained by the experiments are presented and discussed. The experiments were run on 42 datasets, EEG signals of 7 subjects, collected from 2 electrodes (P3 and P4), for 3 different protocols (Letter p, b and d). In some experiments, results of only a subset of datasets are presented for convenience purposes. However, the subset was chosen such that either the results of all the datasets are fairly represented or as proof for an argument.

4.1. GTM Algorithm Demonstration

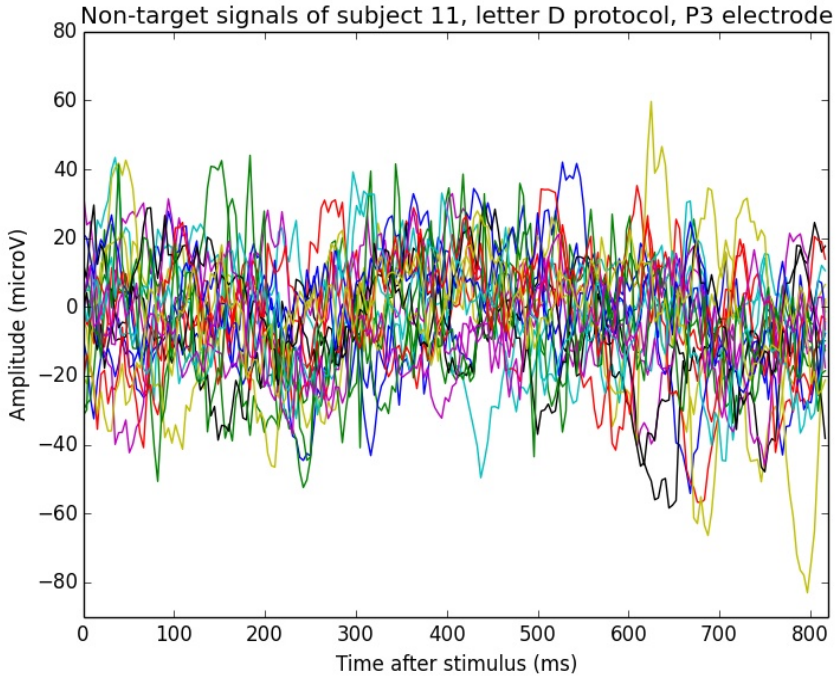
The basic idea of Generative Topographic Mapping algorithm is mapping high dimensional data on to a low dimensional space. In this Section, this concept is demonstrated. The EEG data used for all the experiments in this project is 210 dimensional. Figures 4.1a and 4.1b show the collection of 210 dimensional target EEG signals and nontarget EEG signals of a subject extracted from the P3 electrode for the letter d protocol.

From the figure, we can see that the target signals together form a slight positive peak around the 100th sample (around 300-400ms after the stimulus) where the P300 wave is supposed to be, but the peak might not be that obvious in each individual target signal. The non-target signals show no specific pattern to extract any information.

The GTM algorithm can be used to represent this EEG data on a lower, in this case two, dimensional space. The unlabeled EEG signals were applied to the GTM algorithm and it was trained to identify the differences between the signals. Figure 4.2 illustrates the two dimensional representation of the EEG data that was depicted in Figures 4.1a and 4.1b. The visualization is done using the means of the posterior probability of each point on the



(A) Target signals from Subject 11, Letter d protocol, P3 electrode



(B) Non-target signals from Subject 11, Letter d protocol, P3 electrode

FIGURE 4.1. 210 dimensional EEG signals of Subject 11

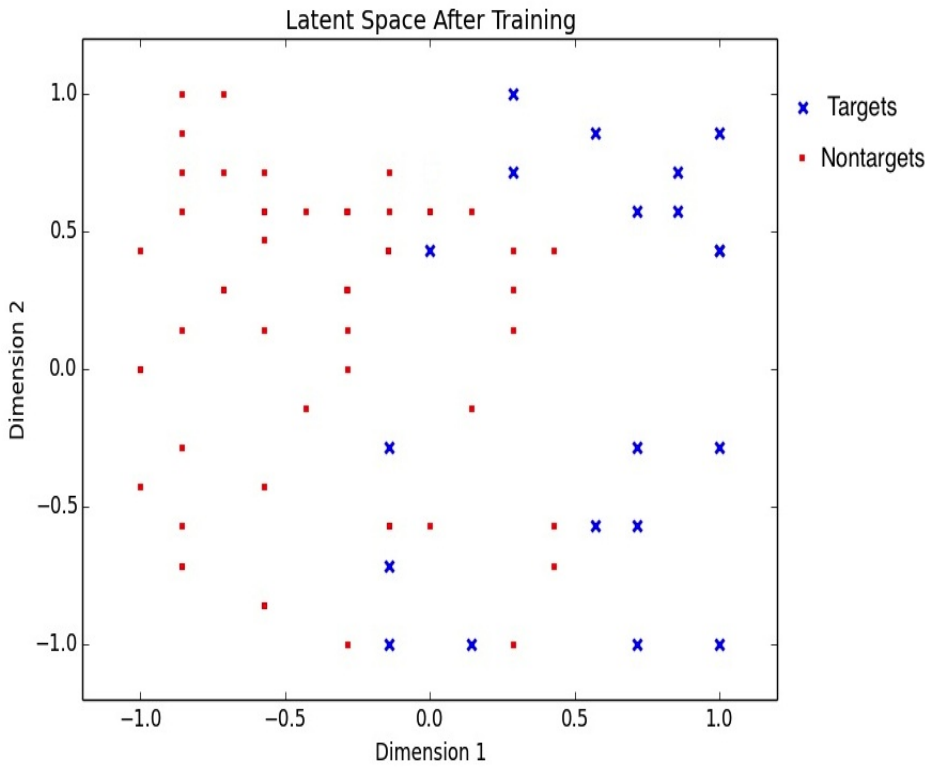


FIGURE 4.2. Latent space mapping of P3 electrode EEG signal of subject 11 for letter d protocol

latent space, that is, the mean of each latent point's responsibility on the data point. The labels of the classes were used to color code the visualization.

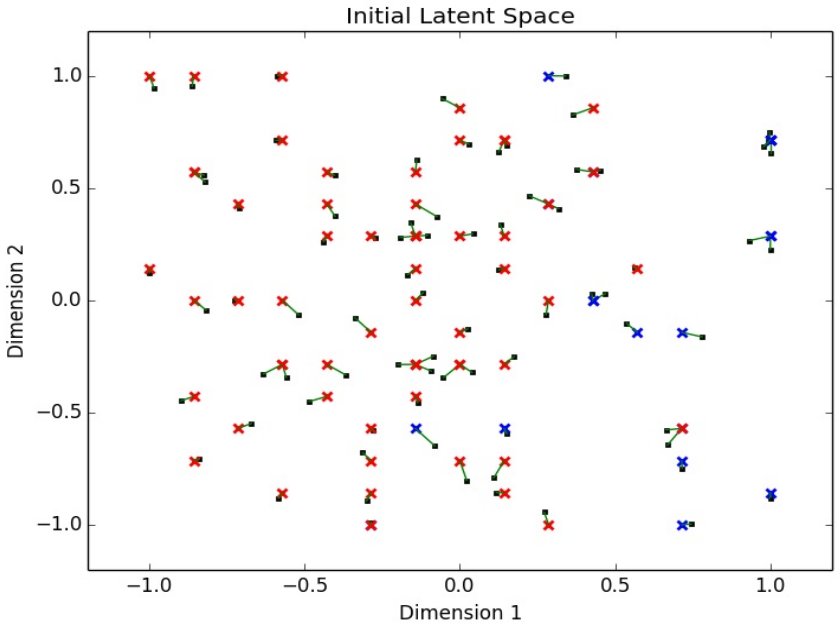
From Figure 4.2, we can see that the mapping of the above mentioned EEG signals have somewhat clustered in to groups of target points and non-target points. The GTM algorithm is an unsupervised learning algorithm, so it doesn't use information about the classes of the data during the mapping process. Therefore, we can assume that the GTM algorithm looks at the inherent features of the EEG signals during training and the presence and absence of the P300 wave might have caused this separation of points in the latent space. There are some target points that are quite separated from its own kind and mapped amongst nontargets. These points can be of data where the P300 wave is not so significant in the targets. This figure provides more insight to the data in the sense that the difference between

the targets and nontargets can be easily spotted because of the clustering. More analysis on the EEG signals that lead to this mapping is presented in Section 4.5.

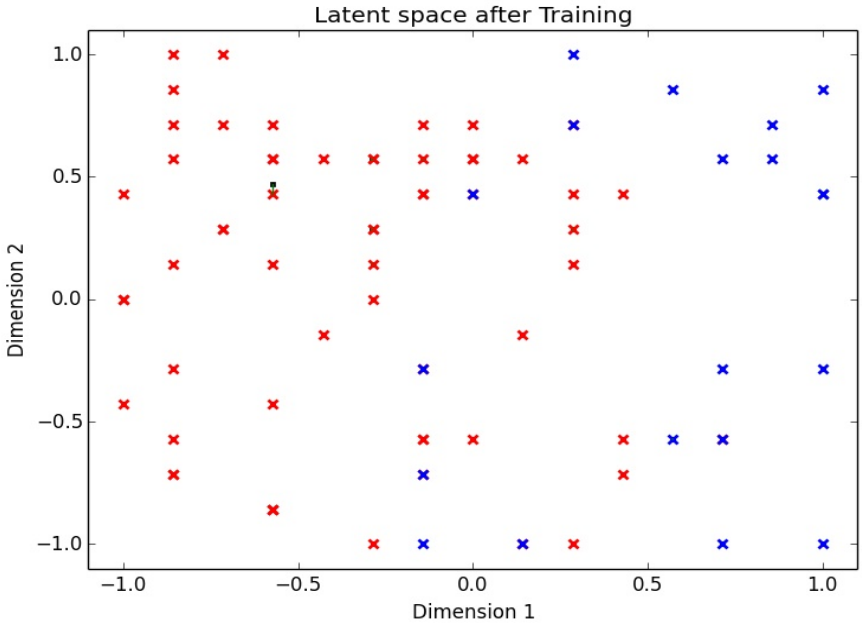
However, the nonlinear mapping can result in multi-modal posterior probability. That means more than one latent point can share high responsibility for a particular data point. Hence, means might misrepresent the data in the latent space. To explore this, the modes of the posterior probabilities were also plotted on the latent space together with the means. Figure 4.3a illustrate the initial map resulted by the GTM algorithm before training to maximize the log-likelihood (before training). Figure 4.3b is the map resulted by the GTM algorithm after the log-likelihood has been maximized (after training). The corresponding mean and mode of each data point are connected by a green line.

If the posterior distribution is unimodal and symmetric (or reasonably close to being so), then the mean and mode will be close together. If the posterior distribution is more complicated, perhaps because the manifold is twisted or highly curved in the region of a given data point, then the two may be further apart. In Figure 4.3a, we can see that the means and modes are located pretty close to each other for almost all the data points. Therefore, we can deduce that the posterior distribution of the initial distribution itself of the dataset is close to unimodal. Another observation is that the same latent space point can be most responsible for many data points, even though the responsibility distribution of each of those data points seems to be very different.

However, after the training process, we can see in Figure 4.3b that there is a latent point for each data point that is significantly more responsible than the other latent points. That means that during training, the mixture component centers have been adjusted to almost exactly mimic our data distribution, which was the objective. For the visualization purposes



(A) Means and Modes of the posterior probability on the latent space before training



(B) Means and Modes of the posterior probability on the latent space after training

FIGURE 4.3. Means and Modes of the posterior probability on the latent space of subject 11, letter d protocol, P3 electrode

of this thesis, we will be using the posterior probability means distribution for the rest of the experiments.

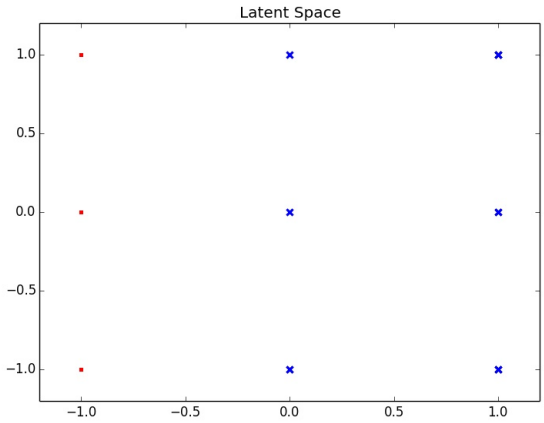
Next, we can explore how the parameters K, the number of latent points and M the number of RBF hidden nodes affect the mapping of the EEG signal.

4.1.1. Effect of GTM Parameters. In Section 3.3.1, we suggested that the number of latent space points and number of hidden units of the RBF network can be determined empirically. In this Section, we examine the latent space map of the same EEG data used above and see how it varies according to the changing parameters.

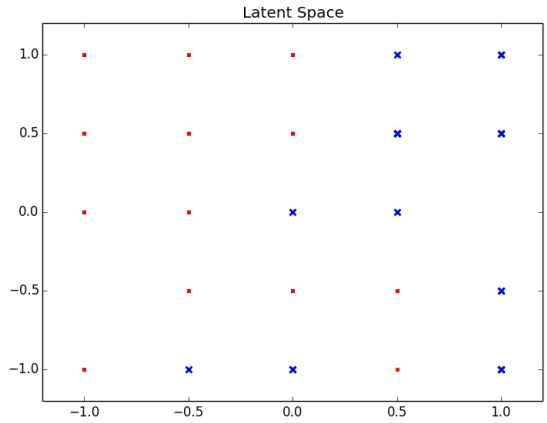
The number of latent space points, K, we define in our algorithm determines the number of Gaussian Mixture centers we have in our data space hence the number of points we have in the latent space to represent the data. Figure 4.4 illustrates the resulting map obtained by changing the number of latent space points defined. The number of RBF nodes were set at 16 for all the these experiments.

From figure 4.4, we can see that lesser the number of latent space points defined, lesser points there are in the latent space to represent our data. Therefore, multiple data points will be represented by the same latent space point, thus, creating ambiguity, specially when K is less than the number of data samples we have ($N = 80$). We can observe that clearly in Figures 4.4a, 4.4b and 4.4c. Figure 4.4e show improvement over Figure 4.4d. However, there is only slight visible difference between maps of Figure 4.4e and 4.4f, even though number latent space points have increased from 225 to 400. Therefore, I have selected 225 to be the optimum number of latent space points that produce unambiguous latent space maps for the EEG data.

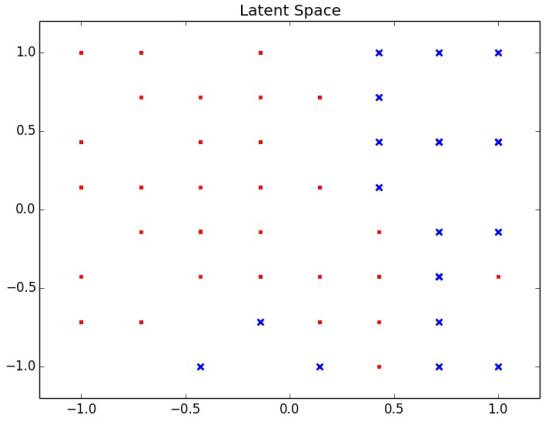
Next, we try to find the optimum number of RBF network nodes (M) that produce a good mapping from the latent space to the data space, which in turn result in good latent



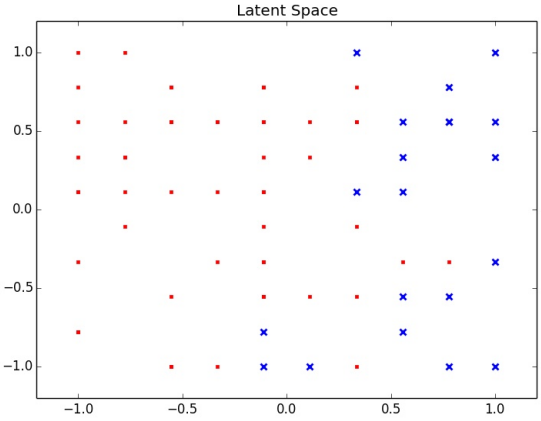
(A) K = 9



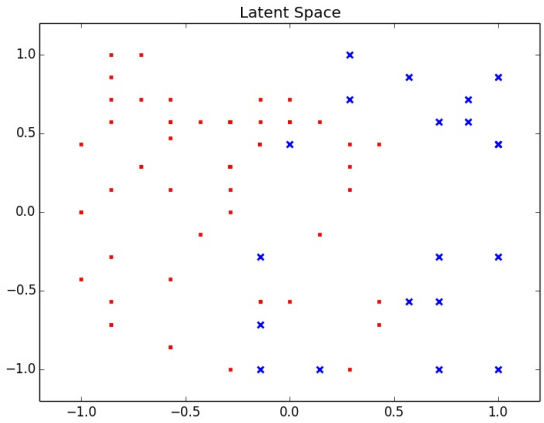
(B) K = 25



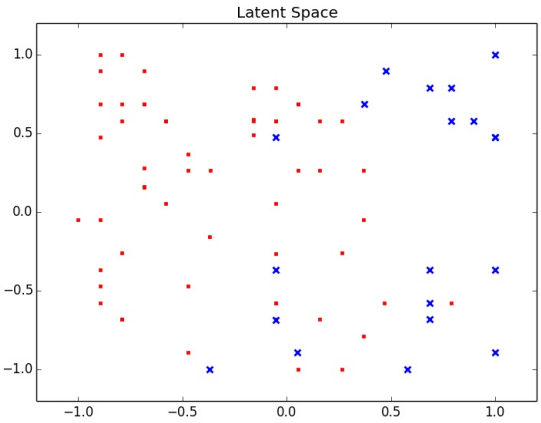
(C) K = 64



(D) K = 100



(E) K = 225



(F) K = 400

FIGURE 4.4. Latent Space Maps resulted by varying the number of latent space points

space representation of the data. Figure 4.5 presents the resulting latent space maps by varying M from 4 to 64. The number of latent space points K was kept at a constant 225 through every trial.

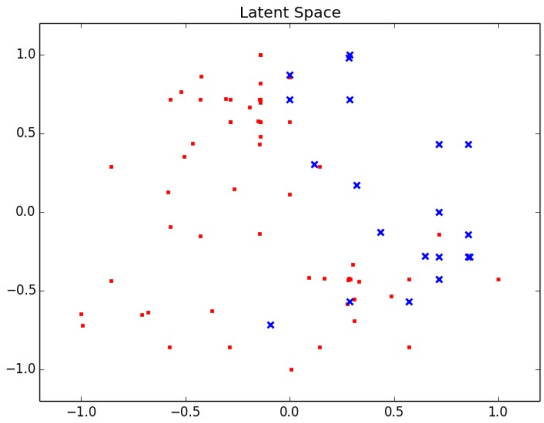
As the number of hidden nodes in the RBF network was changed from 4 to 64, we can see the separation between targets and nontargets increase from Figure 4.5. What we can see is that when M is high (Figure 4.5e and 4.5f), the latent space points get more clustered together, because the network overtrains for the training data hence, the prior probabilities become more precise. However, we have considered M to be 16 in our experiments as a compensation of both precision and computational complexity.

Now that we have mapped the 210 dimensional EEG signals onto a 2 dimensional space and selected reasonable parameters to do so, we can use this 2-D mapped data to classify the EEG signals into targets and nontargets. The next Section discusses the results of this classification.

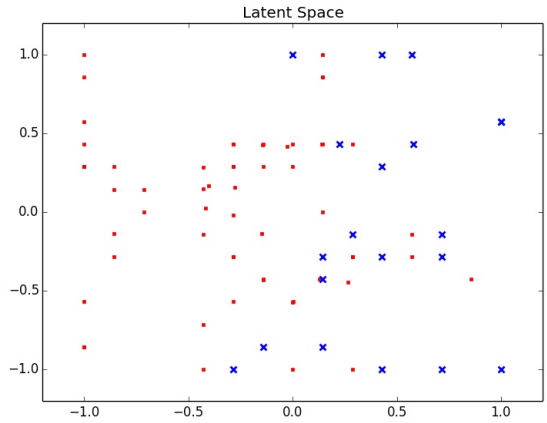
4.2. Effect of GTM on classification of EEG signals

The main research question for this project is to figure out whether application of the GTM algorithm on EEG data can improve its classification. Since the GTM algorithm is essentially a mapping of data from a high dimensional space to a lower dimensional space, the lower dimensional data representation can be a good candidate for classification rather than using the very high dimensional raw EEG data itself. Not only might it reduce the complexity of the classification process, but also, GTM might enhance the significant features and reduce the effect unnecessary features of the EEG data during the mapping process.

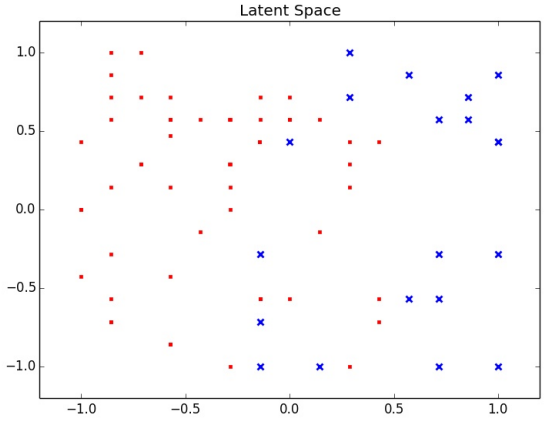
To examine this, we have classified the two dimensional representation of the EEG data using a neural network. Neural network was chosen as the classifier for the datasets because



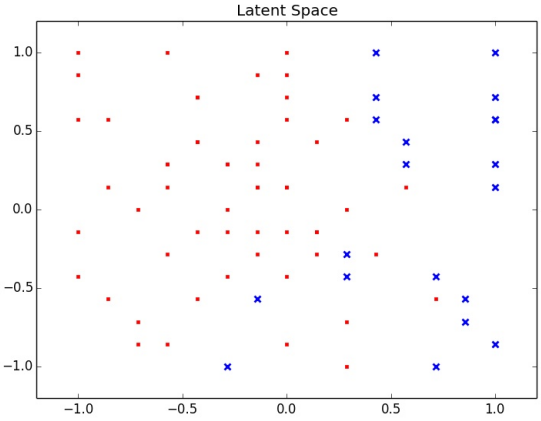
(A) M = 4



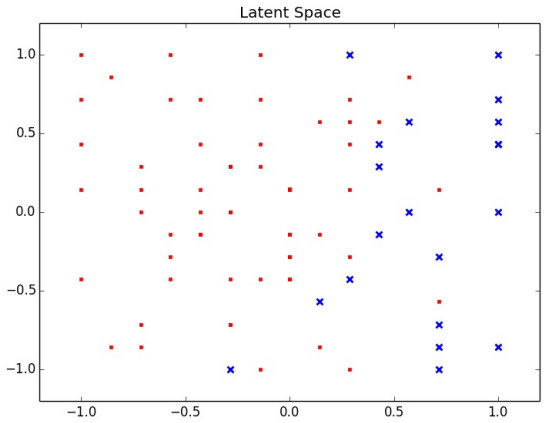
(B) M = 9



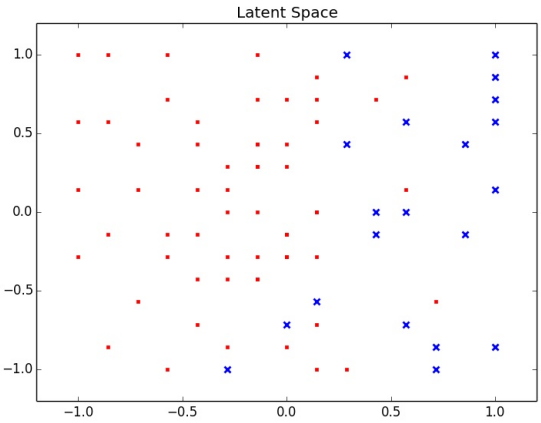
(C) M = 16



(D) M = 25



(E) M = 36



(F) M = 64

FIGURE 4.5. Latent Space Maps resulted by varying the number of RBF nodes

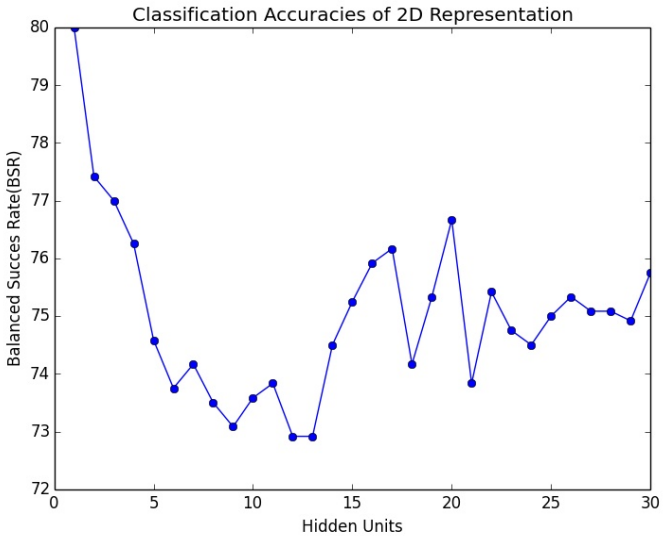


FIGURE 4.6. Classification accuracy of GTM applied EEG data against the number of hidden units of the neural network

from the example latent map obtained from GTM (Figure 4.2), it seems that a nonlinear classifier is better suited to classify the points rather than a linear classifier. The classification accuracy results were compared to accuracies obtained by classifying the 210 dimensional raw EEG data with a neural network. The accuracies were calculated using BSR.

For 2D representation of EEG data, we need a neural network with 2 input units, and 2 output units to represent the two classes, targets and nontargets. However, we cannot guess the number of hidden units that will give the best classification accuracy. For that, we need to check the classification accuracy by varying the number of hidden units. Figure 4.6 illustrates the plot of classification balanced success rate versus the number of hidden units in the neural network.

As we can see, the balanced success rate of the latent space points is fairly high at 80%, with a neural network of just one hidden unit. When the number of hidden units increase, the accuracy has dropped. A reason for this can be that higher number of hidden units trains the network to overfit the 2 dimensional training data, hence misclassifying test data.

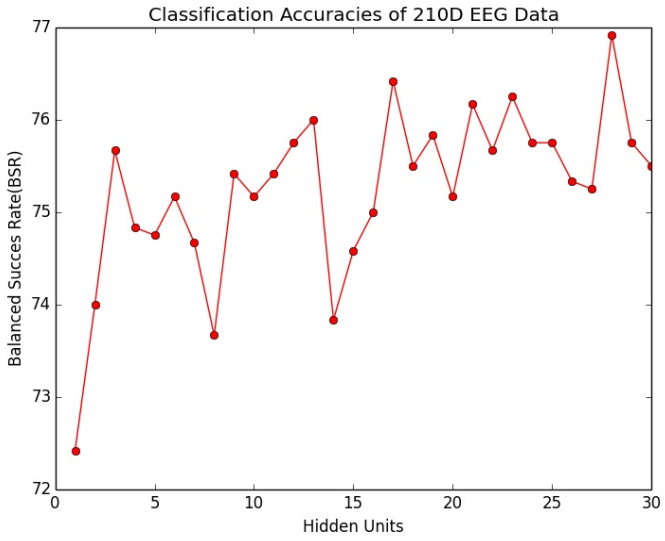


FIGURE 4.7. Classification accuracy of preprocessed raw EEG data against the number of hidden units of the neural network

However, with only this result, we cannot conclude that application of GTM is a good option. Let's compare the above classification results to classification accuracy of preprocessed raw EEG data with 210 dimensions. The neural network used for this contains 210 input units and 2 output units. The classification accuracy with varying number of hidden units is shown in Figure 4.7. The maximum accuracy obtained by classification of preprocessed raw EEG data is 76.917% with 28 hidden unit neural network.

Nonetheless, we still cannot conclude that GTM will give better classification accuracy with a simpler classifier by just testing with one data set. Therefore, we tested with 39 datasets, collected from 7 subjects, 3 protocols (letter b, d, p) and 2 electrodes (P4, P3). All the datasets were applied to the GTM algorithm and the resulting 2D representation was classified using neural network. To compare, all raw datasets (210D) were also classified using neural networks. In both cases, classification was done 10 times, each time reinitializing the neural network, and the average classification accuracies were calculated. Additionally,

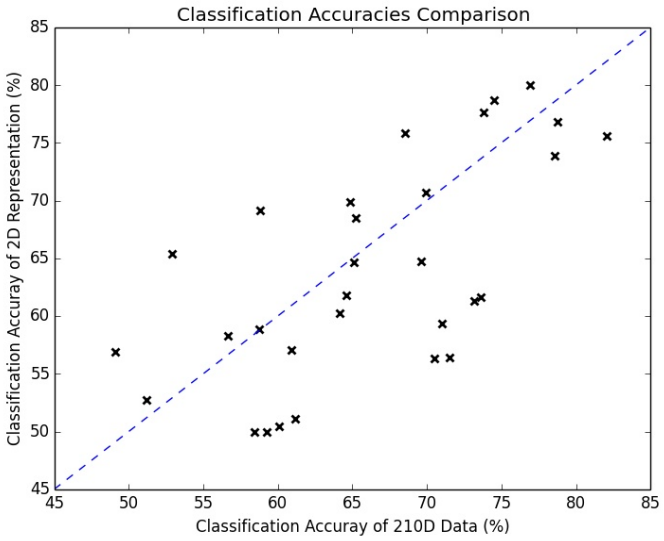


FIGURE 4.8. Classification accuracy of 210D EEG data vs 2D representation

in both cases of classifying 2D and 210D data, the number of hidden units of the network for each dataset were varied from 1-30 and the optimum classification accuracy was picked.

When analyzing the classification accuracies, in 17 datasets out of 39 (43.5%), we could see an improvement in classification accuracy when 2D representation was used instead of the 210D representation. Figure 4.8 plots the classification accuracy of 210D dataset versus the classification accuracy of 2D representation of each of the 39 dataset.

From the Figure 4.8, we see that not many datasets perform better when the 2D representation is used. In some datasets, the classification accuracy has been reduced when the 2D representation is used instead of the 210D data and for some, it has increased. But most of the datasets lie in close proximity to the 45 degree line suggesting that for most datasets, the classification accuracy is more or less the same.

The neural network is a supervised classification algorithm. However, the GTM mapping is not supervised. So, if the EEG data set in question does not have many target signals with significant P300 waves, the unsupervised GTM algorithm would not identify the difference

between the targets and nontargets, hence, there would not be a clear separation on the latent map. This would lead to poor classification results. However, since the neural network is supervised, it can learn to minimize the error of classification by using the data labels and 210 dimensions of data when raw data is applied. Therefore, such raw EEG data will be classified better with the supervised classification algorithm, than classifying the 2 dimensional latent map points of the unsupervised algorithm.

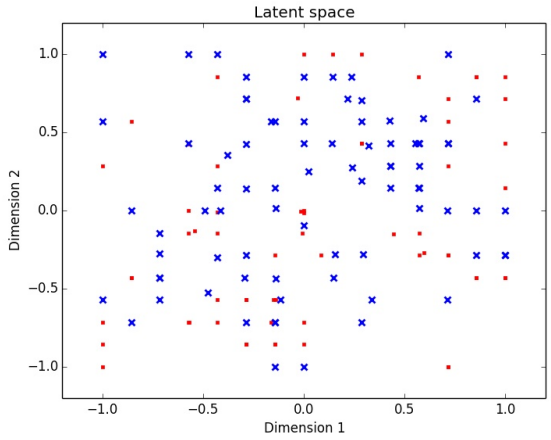
4.3. Other Applications

So far, we have only used the GTM algorithm to check whether it can identify inherent features of EEG signals with the P300 wave and without. We can extend the concept to see whether the GTM algorithm can distinguish EEG signals of different subjects and protocols. The following Sections present the results of these experiments.

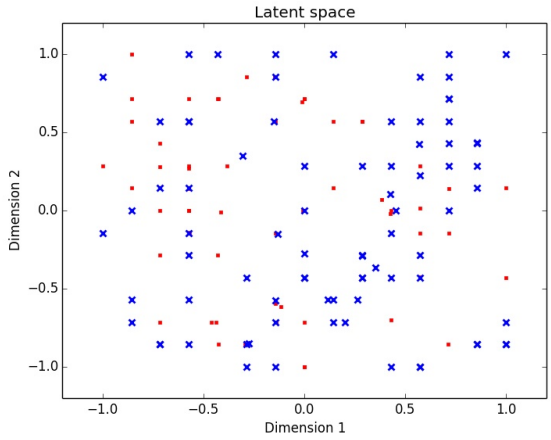
4.3.1. Inter-Subject Experiment. In this experiment, we have explored whether the GTM algorithm can distinguish between data collected from different subjects by a particular protocol and electrode. We have used data from 7 different subjects for the experiments, where 3 of them were impaired and the other 4 were not. The able-bodied subjects' data were collected inside a lab and the impaired subjects' data were collected in their homes. More details about the subjects can be found in Section 3.1. This type of classification between subjects can be useful in applications such as multi player games using BCI systems.

Figure 4.9 represents some latent space mapping done between two subjects at a time. The plots in the figure show no separation between the latent space mapped points of two subjects, whether the two subjects in focus are impaired, non impaired or one of each.

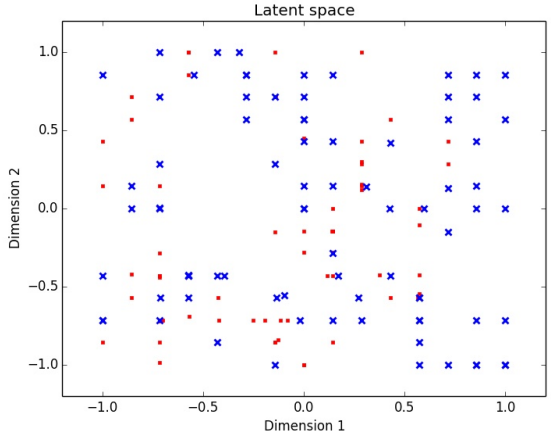
However, some datasets showed a vague separation between the points of the two subjects on the latent space. Figure 4.10 illustrates latent spaces of some of those datasets.



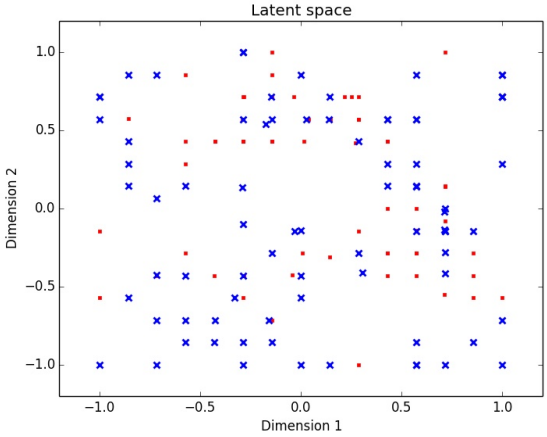
(A) Subject 15 vs Subject 16 (Letter b/P4)



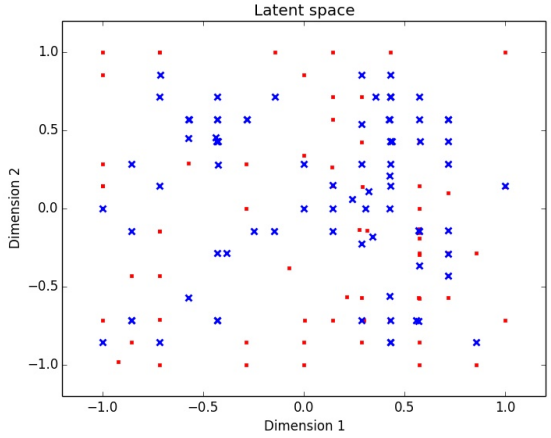
(B) Subject 15 vs Subject 20 (Letter d/P3)



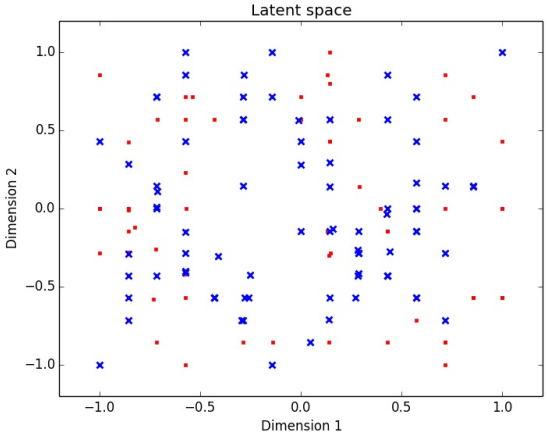
(C) Subject 16 vs Subject 20 (Letter b/P3)



(D) Subject 20 vs Subject 21 (Letter d/P4)

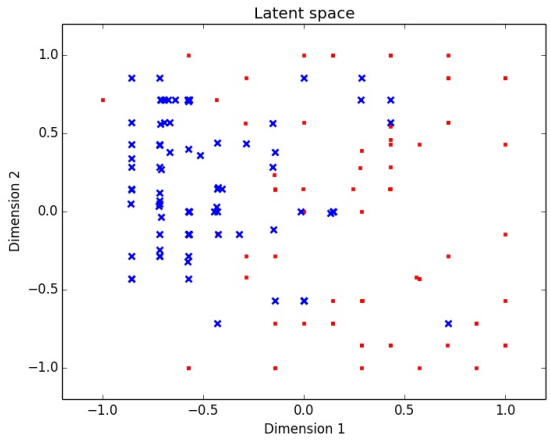


(E) Subject 21 vs Subject 24 (Letter p/P4)

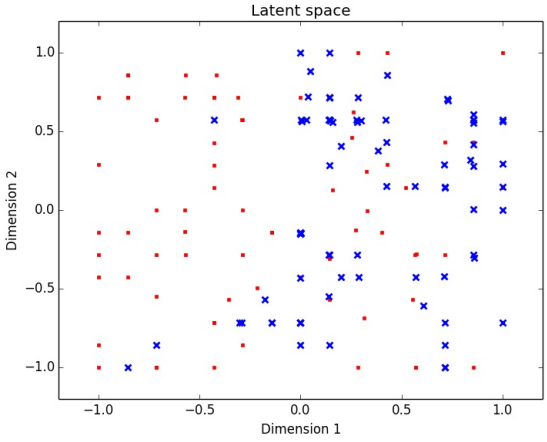


(F) Subject 23 vs Subject 24 (Letter d/P3)

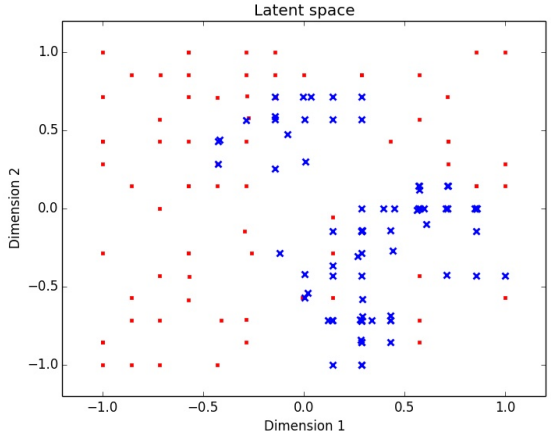
FIGURE 4.9. Latent space maps of data of two different subjects



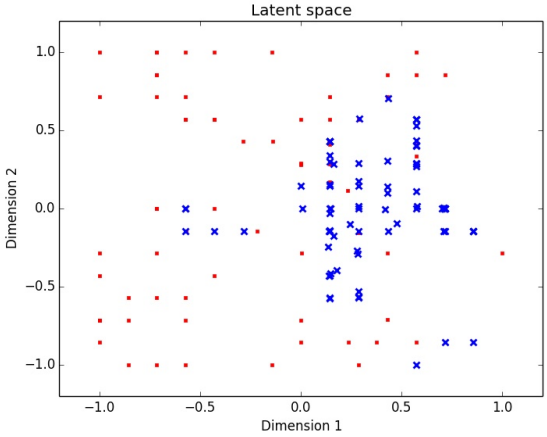
(A) Subject 11 vs Subject 16 (Letter b/P4)



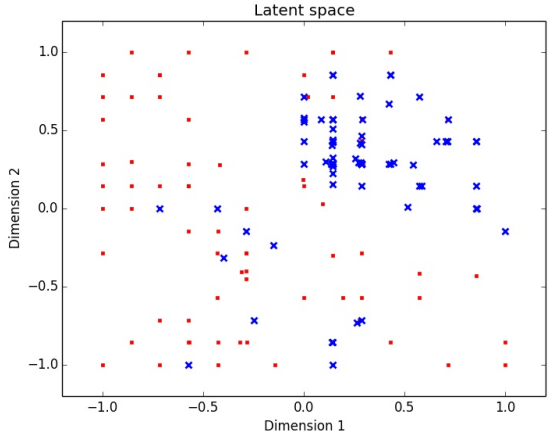
(B) Subject 11 vs Subject 23 (Letter b/P4)



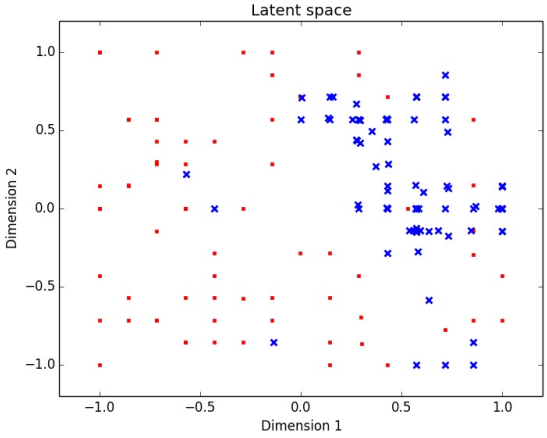
(C) Subject 11 vs Subject 15 (Letter d/P3)



(D) Subject 11 vs Subject 20 (Letter d/P4)



(E) Subject 11 vs Subject 21 (Letter p/P3)



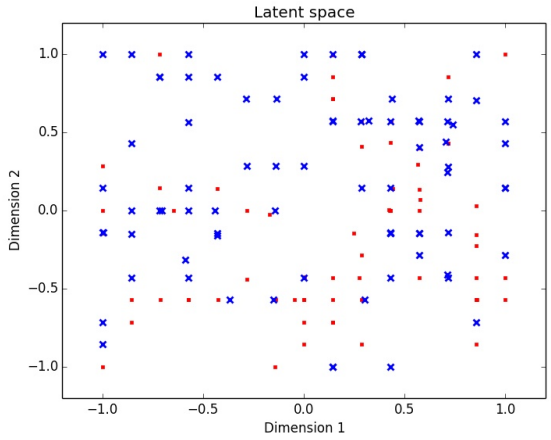
(F) Subject 11 vs Subject 16 (Letter p/P4)

FIGURE 4.10. Latent space maps of data of two different subjects that showed separation of the datasets

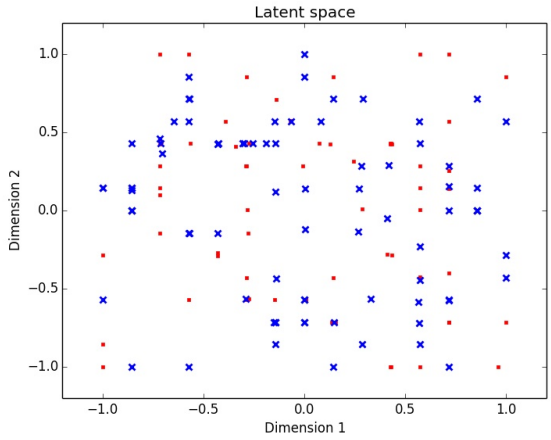
From Figure 4.10, we can see that the GTM algorithm has identified some inherent features between the two subjects and has mapped them in distinct areas of the latent space. We will looking more into detail about what caused the separation in Section 4.5.

4.3.2. Inter-Protocol Experiment. In this experiment, we applied EEG data collected from the same subject, same electrode but collected from different trials. The purpose was to see whether the GTM algorithm separates EEG signals from 2 different protocols. The protocols considered here are the letter-b, letter-p and letter-d protocols, where the subject had to count the number of times the target letter appeared on the computer screen within a flashing character sequence of 20 target letters and 60 nontarget letters.
From Figure 4.11, we can see that there is no separation between latent points of two different protocols. This result is expected because P300 wave cannot be used to identify what caused the stimulation, but only indicates at what time the simulation is presented to the subject. P300 wave occurs in the brain when the subject identifies a task relevant stimulus, but is assumed to not vary according to which kind of visual stimulus is provided. Therefore, there shouldn't be a difference between target signals collected from letter p, b or d protocols. The results obtained from the GTM algorithm supports this assumption.

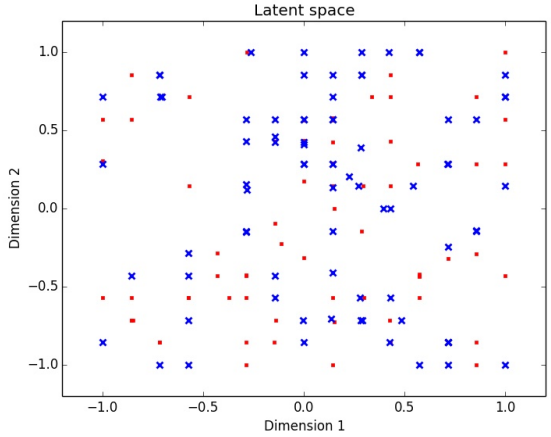
Figure 4.12 illustrates only the 20 target signals of the two protocols mapped on the latent space. We can see that the latent points of target signals of both protocols have more or less clustered into one part of the latent map. This is because both of the signals have similar P300 waves. Therefore, the GTM algorithm has identified this P300 and have clustered them together during the mapping process.



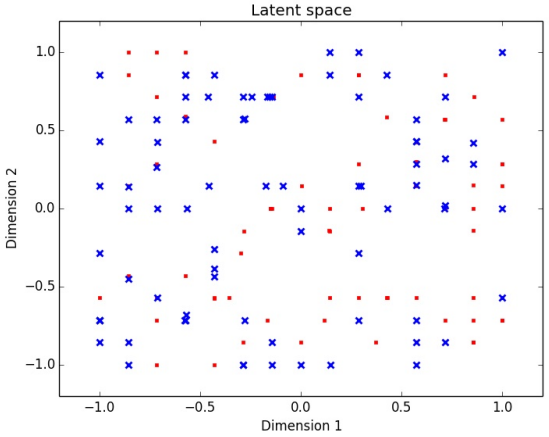
(A) Letter b vs Letter p (Subject11/P3)



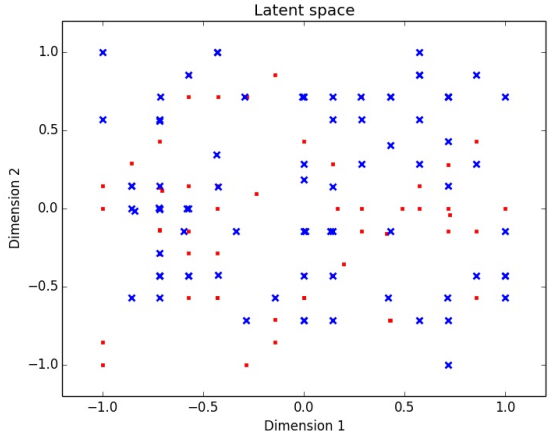
(B) Letter d vs Letter p (Subject15/P4)



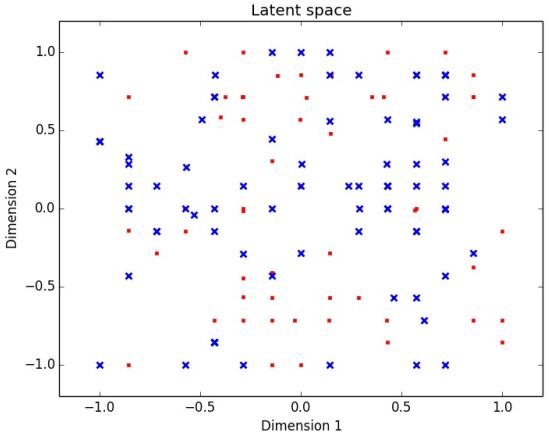
(C) Letter b vs Letter p (Subject16/P3)



(D) Letter d vs Letter p (Subject20/P3)

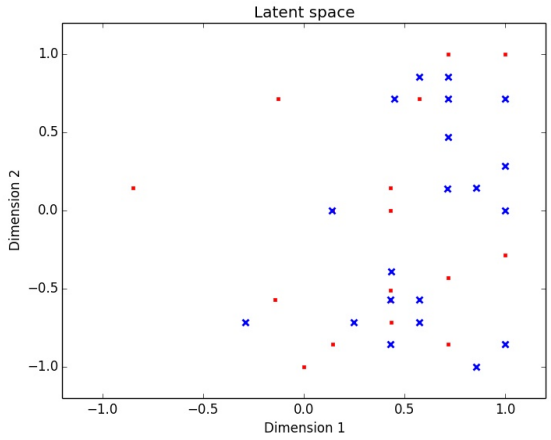


(E) Letter b vs Letter d (Subject23/P3)

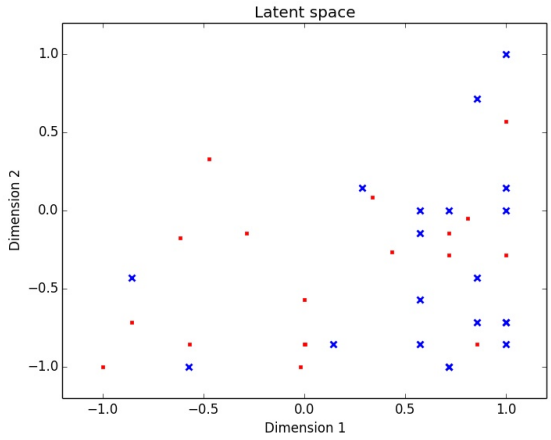


(F) Letter b vs Letter d (Subject24/P4)

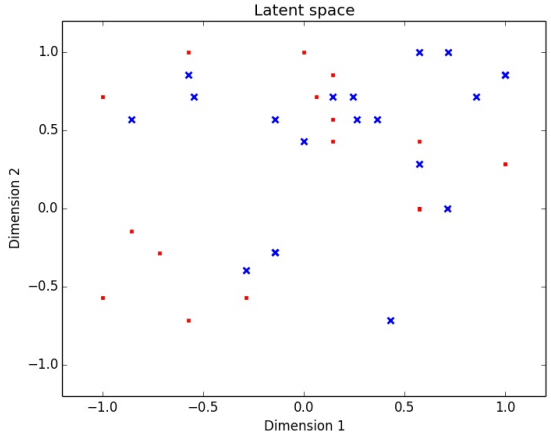
FIGURE 4.11. Latent space maps of target signals of two different protocols



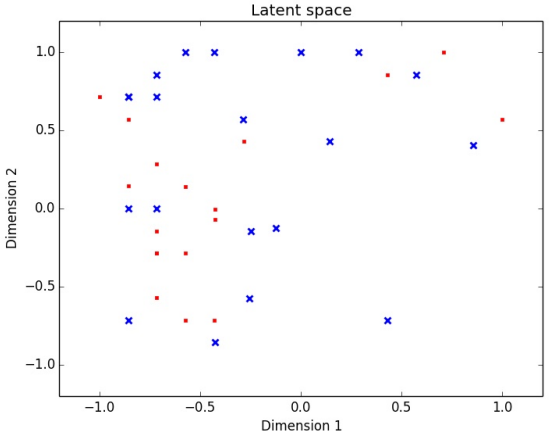
(A) Letter b vs Letter d (Subject11/P3)



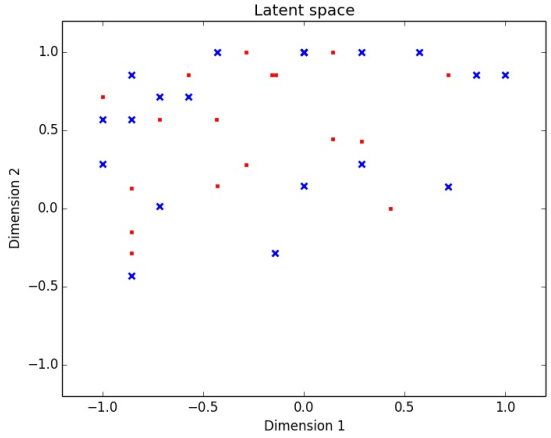
(B) Letter b vs Letter d (Subject11/P4)



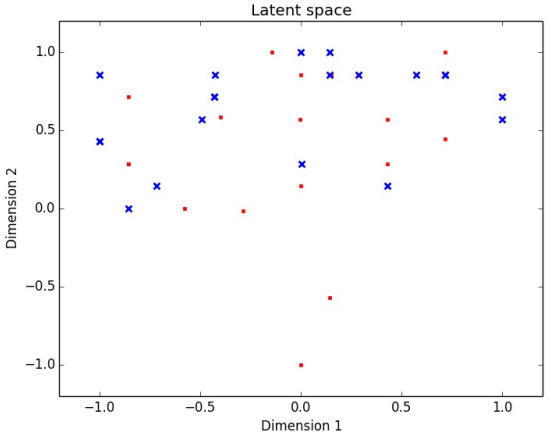
(C) Letter b vs Letter p (Subject15/P4)



(D) Letter b vs Letter d (Subject21/P4)



(E) Letter b vs Letter d (Subject24/P3)



(F) Letter b vs Letter d (Subject24/P4)

FIGURE 4.12. Latent space maps of target signals of two different protocols

4.4. Performance Improvement Considerations

In this Section, we discuss methods we considered to improve the classification results obtained by the data resulted from the GTM algorithm. The following subsections present the experiments and the results obtained.

4.4.1. Using Multiple Electrodes. So far, we tried analyzing EEG signals obtained from a single electrode, either P3 or P4. Both electrodes are good candidates for observing a P300 signal for an external stimulus. Ideally, when an external stimulus is presented, both P3 and P4 electrodes should record P300 waves. Hence, if we use both of these electrodes at once, we may be able to increase our classification accuracy. This experiment focuses on that.

If we concatenate the time samples collected from both P3 and P4 electrodes, it will result in a 420 dimensional dataset (210 time samples in each electrode's dataset). A target data sample drawn from this dataset should ideally contain 2 peaks of P300 within its 420 dimensions of features. Since now there are additional features in the data for the GTM algorithm, we expect it to identify the difference between targets and nontargets better and therefore, map them with a clear separation on the latent space. This in turn, should result in better classification accuracy.

Table 4.1 presents a comparison between the classification accuracy obtained for several datasets by using individual electrodes (P4 and P3) separately, and the classification accuracy obtained when data from both electrodes are used together.

The highlighted rows are instances where we were able to obtain a better performance over classifying data collected from individual electrodes by concatenating the data of the two electrodes. From the table, we can see that 46.15% of the time, we were able to obtain

TABLE 4.1. Classification accuracy comparison by using data of two electrodes. The rows where using both electrodes increased the accuracy are highlighted in green

Dataset	P3 electrode	P4 electrode	P3 and P4 electrodes
Subject 11 Letter p	70.66%	77.66%	64.67%
Subject 15 Letter b	56.33%	61.33%	53.5%
Subject 15 Letter d	61.83%	61.66%	51.17%
Subject 15 Letter p	50.416%	56.083%	59.833%
Subject 16 Letter b	52.75%	56.916%	52.25%
Subject 16 Letter p	50%	51.083%	54.083%
Subject 20 Letter b	50%	57.083%	58.417%
Subject 20 Letter d	68.5%	59.33%	52%
Subject 20 Letter p	65.416%	58.83%	51.583%
Subject 21 Letter b	80%	64.75%	55.67%
Subject 23 Letter b	58.25%	69.16%	65%
Subject 23 Letter d	52.83%	53.66%	54.83%
Subject 24 Letter b	55.916%	57.083%	59.917%

better performance by using data of both the electrodes together. For the other 53.85% of the datasets, the classification accuracy of two electrodes together is worse than that of the individual electrodes.

We might obtain a higher accuracy by concatenating the two datasets because the GTM algorithm has extra features to learn from, ideally two P300s for each target data sample and absence of any for each non-target data sample. However, from above examples we have seen that not all targets contain P300 waves in both P3 and P4 electrodes and some non-target signals can have a pattern similar to P300 wave. Instances like this can mislead the GTM algorithm, therefore, in some datasets, the classification accuracy with concatenated

electrodes was either an average or lower than that of the two electrodes. If we observe closer, we can see that for the datasets where the concatenated performance was lower than individual performance, the individual electrode's classification accuracy is anyways low. That means targets and nontarget of those data are not distinguishable by the GTM algorithm. Therefore, concatenating them made the training process even more misleading for the algorithm.

4.4.2. Using Multiple Protocols. Another method we can use that might improve the performance of the training process of the GTM algorithm is by feeding it with more data samples. More data samples means more examples for the algorithm to train from. For this, we can use the 210 dimensional datasets from the three trials, letter b. p and d. In Section 4.3.2, we saw that the EEG signals from the three protocols do not differ from each other. Targets in all three protocols' data should contain P300 waves and non-targets shouldn't. This experiment uses this aspect to try to improve the classification performance.

Here, we have collected all data from the three protocols of a particular subject/electrode into one dataset. The resulting dataset contains 240 data samples, 60 targets and 180 nontargets. Each data sample is 210 dimensional. This large dataset is used to train the GTM algorithm, hoping that more data samples will result in better performance.

Table 4.2 presents a comparison between classification accuracies by classifying the latent space points obtained when data of individual protocols (80 data samples per dataset) and the stacked dataset (240 data samples) are applied to the GTM algorithm.

From the table, we can see that 71.4% of the time, the classification accuracy of the stacked dataset is somewhere between the lowest and highest accuracies of the individual protocol's dataset. In 28.6%, the accuracy obtained by the stacked dataset is lower than each individual protocol's accuracy. Even though we have more target data samples now, if

TABLE 4.2. Classification accuracy comparison by using data of all protocols

Subject ID	Electrode	Letter b Protocol	Letter d Protocol	Letter p Protocol	All Protocols
11	P3	78.66%	76.83%	70.66%	74.89%
	P4	73.917%	75.583%	77.66%	77.49%
15	P3	56.33%	61.83%	50.416%	57.472%
	P4	61.33%	61.66%	56.417%	59.3 %
20	P3	50%	68.5%	65.417%	51.972%
	P4	57.083%	59.33%	58.83%	54.03%
21	P3	80%	64.66%	75.83%	70.33%
	P4	64.75%	56.417%	69.917%	63.19%
23	P3	58.25%	52.83%	52.75%	55.52%
	P4	69.16%	53.66%	57%	52.7 %
24	P3	55.917%	69.083%	65.416%	67.88%
	P4	57.083%	59.33%	58.83%	57.03 %

they don't have clear P300 signals, the GTM algorithm may not be able to distinguish them from non-target samples no matter how many data samples we provide. So if we combine a good set of target samples with clear P300 waves with a bad set of samples that don't have good P300s, that is essentially like adding noise to the good dataset. That is why in most cases the stacked dataset's performance is lower than the highest classification accuracy of the individual protocols. On the other hand, stacking a bad data set with a good one, would improve the performance than classifying just the bad dataset because now the GTM algorithm has more good samples to learn from. Therefore, providing more data samples to the algorithm would not make the performance of the classification better unless the data of all three datasets have good target signals with clear P300s.

4.5. Analysis of EEG Signals on GTM Latent Space

In this Section, we explore more into detail about the mapping by the GTM algorithm. Basically, what we are interested in is to see whether the latent space mapping of a data point is directly related to the properties of the EEG signal corresponding data point. This way, we can justify the existence or a non-existence of separation between target and nontarget points on the latent space by the patterns of the EEG signals that caused the mapping.

4.5.1. Analyzing EEG signals on Latent Space. First of all, we are going to analyze an EEG dataset that was mapped onto the latent space with a clear separation between the targets and non-targets by the GTM algorithm, which consequently resulted in better classification accuracy than raw EEG data. Figure 4.13 illustrates the latent space mapping of a dataset that gave 76.83% BSR when classifying the latent space points. To explore the EEG signals of the data, this figure has the EEG signal corresponding to each data point superimposed at the latent space point of mean posterior probability of the data point.

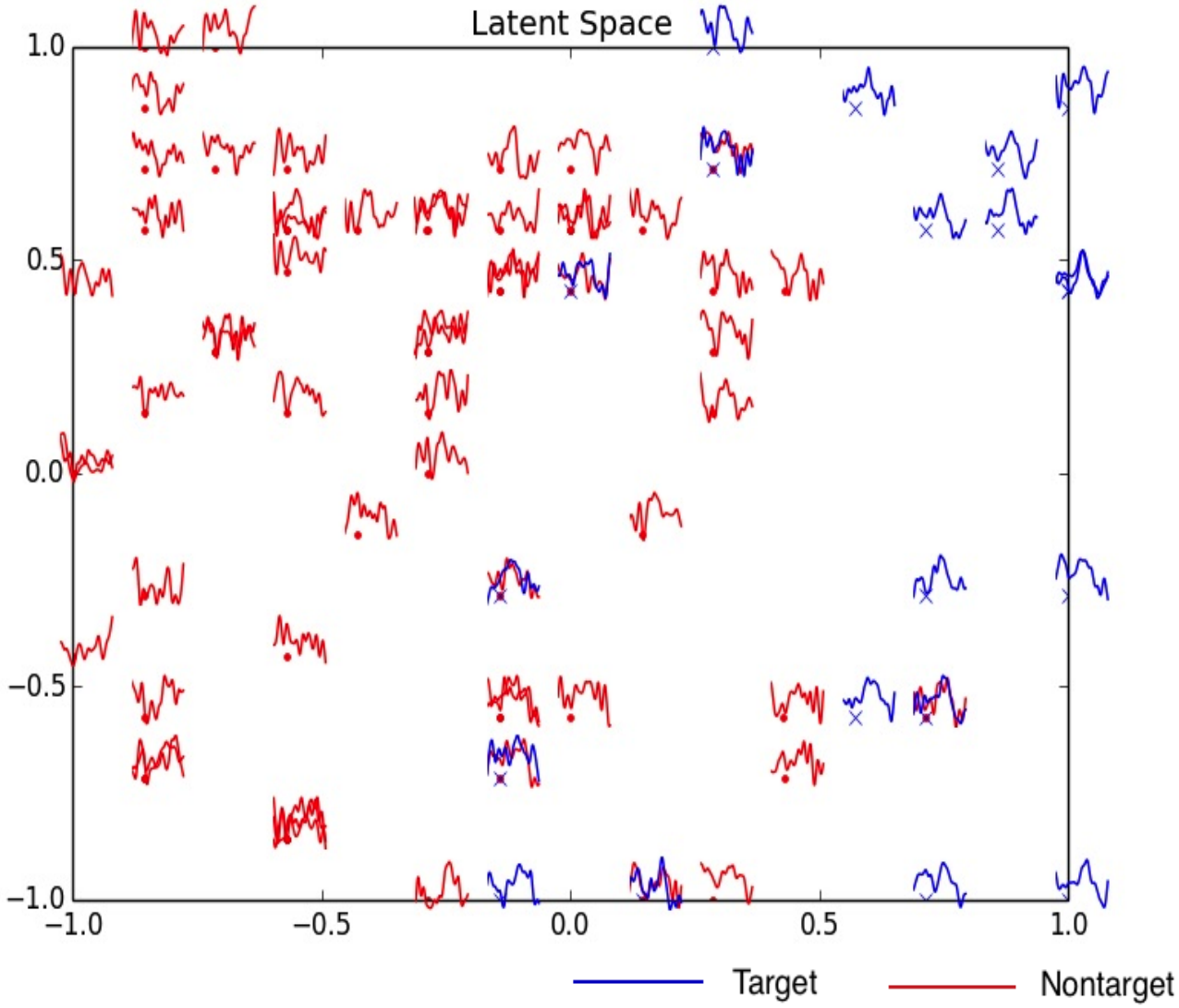


FIGURE 4.13. Latent space mapping of P3 electrode EEG signal of subject 11 for letter d protocol

Let's take a closer look at the EEG signals of points that lie close together in the latent space. We have picked 4 locations on the map to look at the signals that lie at those locations. The locations are indicated on Figure 4.14 by numbered boxes. Figure 4.15 is the collection of the EEG signals at each of the locations identified by the box number.

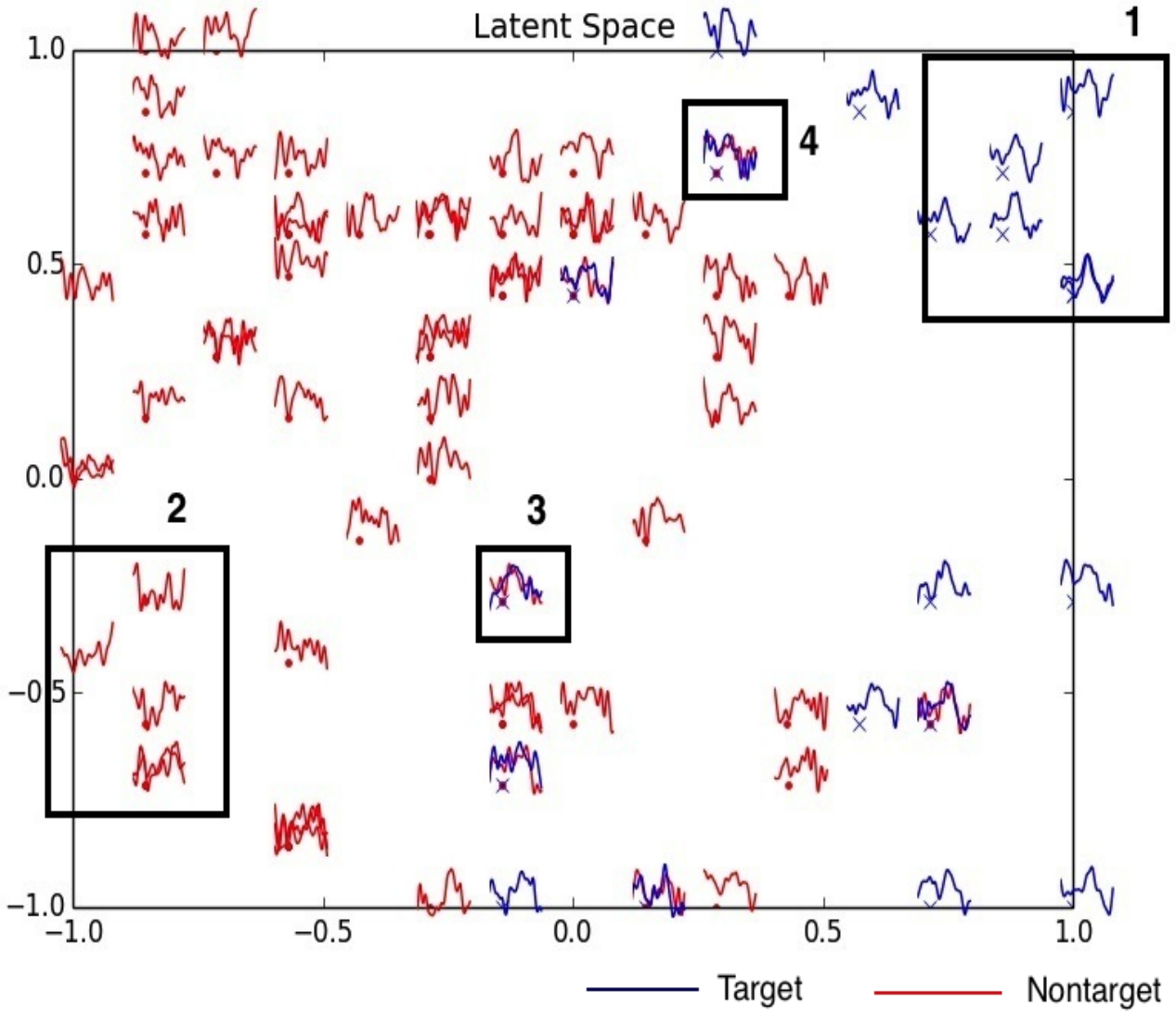
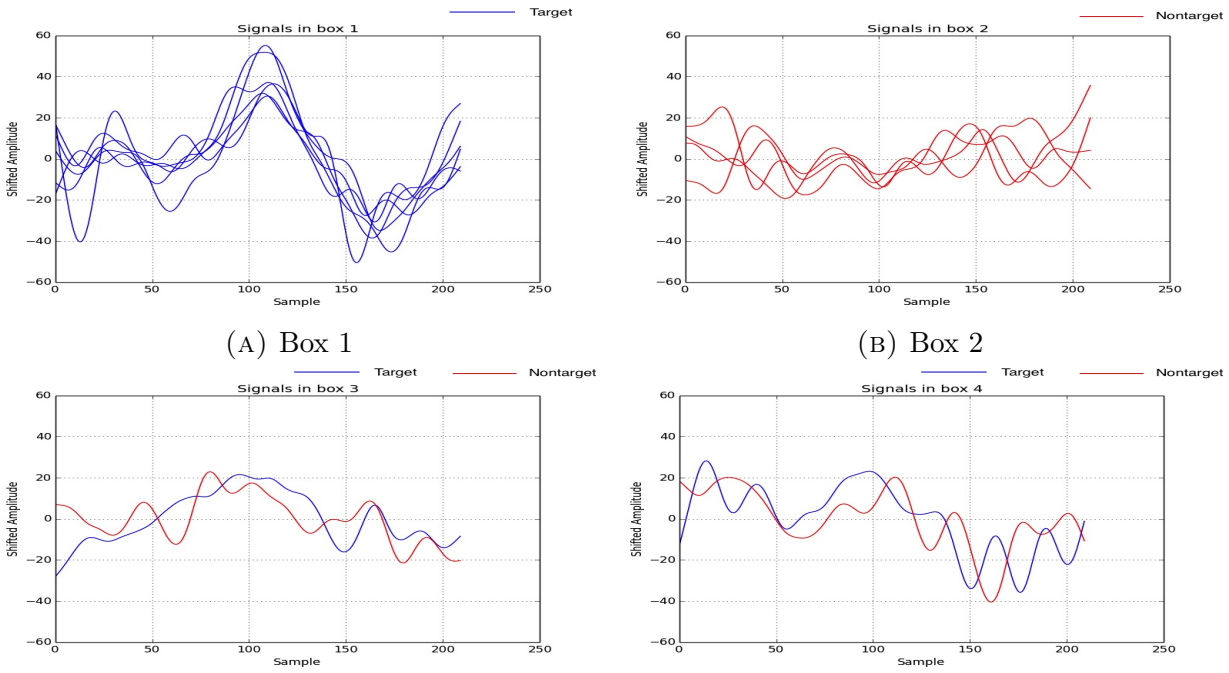


FIGURE 4.14. Latent space mapping of P3 electrode EEG signal of subject 11 for letter d protocol



(A) Box 1

(B) Box 2

(C) Box 3

(D) Box 4

FIGURE 4.15. EEG signals mapped closeby on the latent space. Box 1 all target trials. Boxes 3 and 4 contain both target and nontarget trials. Box 2 contain only nontarget trails

Figure 4.15a illustrates the EEG signals of the points in Box 1 of Figure 4.14, which lies at top right corner of the latent map, which is a cluster of targets. We can see that these signals contain very clear P300 peaks after the 100th sample (390ms) after the stimulus. On the contrast, signals in Box 2, which lies in the bottom left corner of Figure 4.14, presented in Figure 4.15b, does not have a positive deflection where the P300 peak is supposed to be. Therefore, it is a good indication that the GTM algorithm has identified the features of the P300 wave from the data. If we observe along the boxes, we can see that target and non-target data points that have been mapped close by on the latent space have very similar EEG signals. From this exploration, we can conclude that the GTM algorithm has really looked into the behavior of the EEG signals of the data points when mapping to the latent space.

In Section 4.3.1 when we explored whether the GTM algorithm can distinguish between EEG signals of two subjects, we noticed that data of some subjects were separated on the latent space (Figure 4.10). Hence, let's analyze why this happened.

We have chosen the mapping of Subject 11 versus Subject 23 of their letter b, P4 electrode data (Figure 4.10b). We can analyze the EEG signal distribution in the latent space.

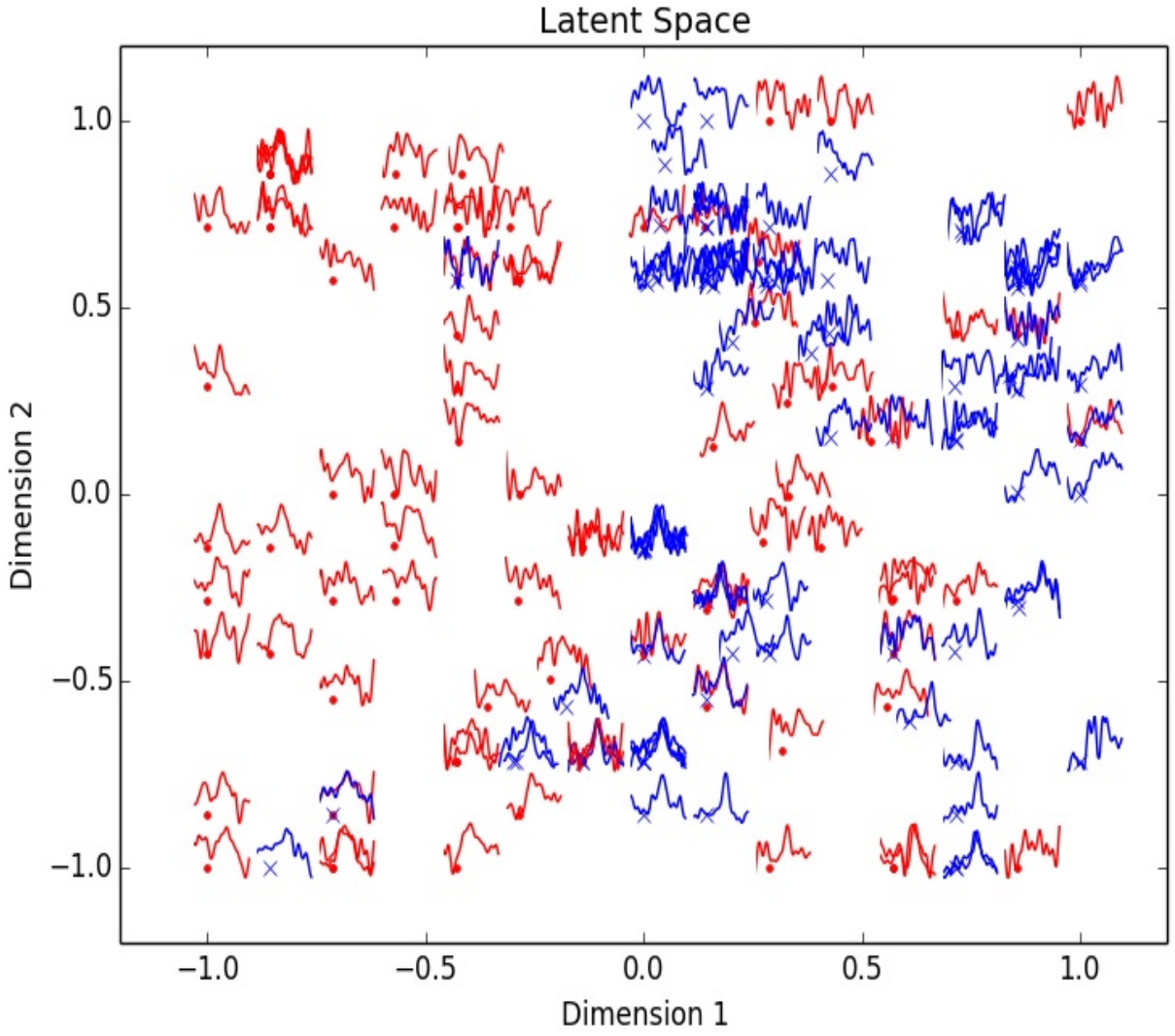


FIGURE 4.16. Latent space mapping of P4 electrode EEG signal of subject 11 vs 23 for letter b protocol

Let's look at the nontargets of both subjects. From 4.17, we can see that nontargets of both subjects have not separated much in the latent space.

Figure 4.18 represents the latent space map of only target signals of both subjects. This is where we see the separation between the two subjects. We can look at target signals of the subjects that are mapped on either side of the map.

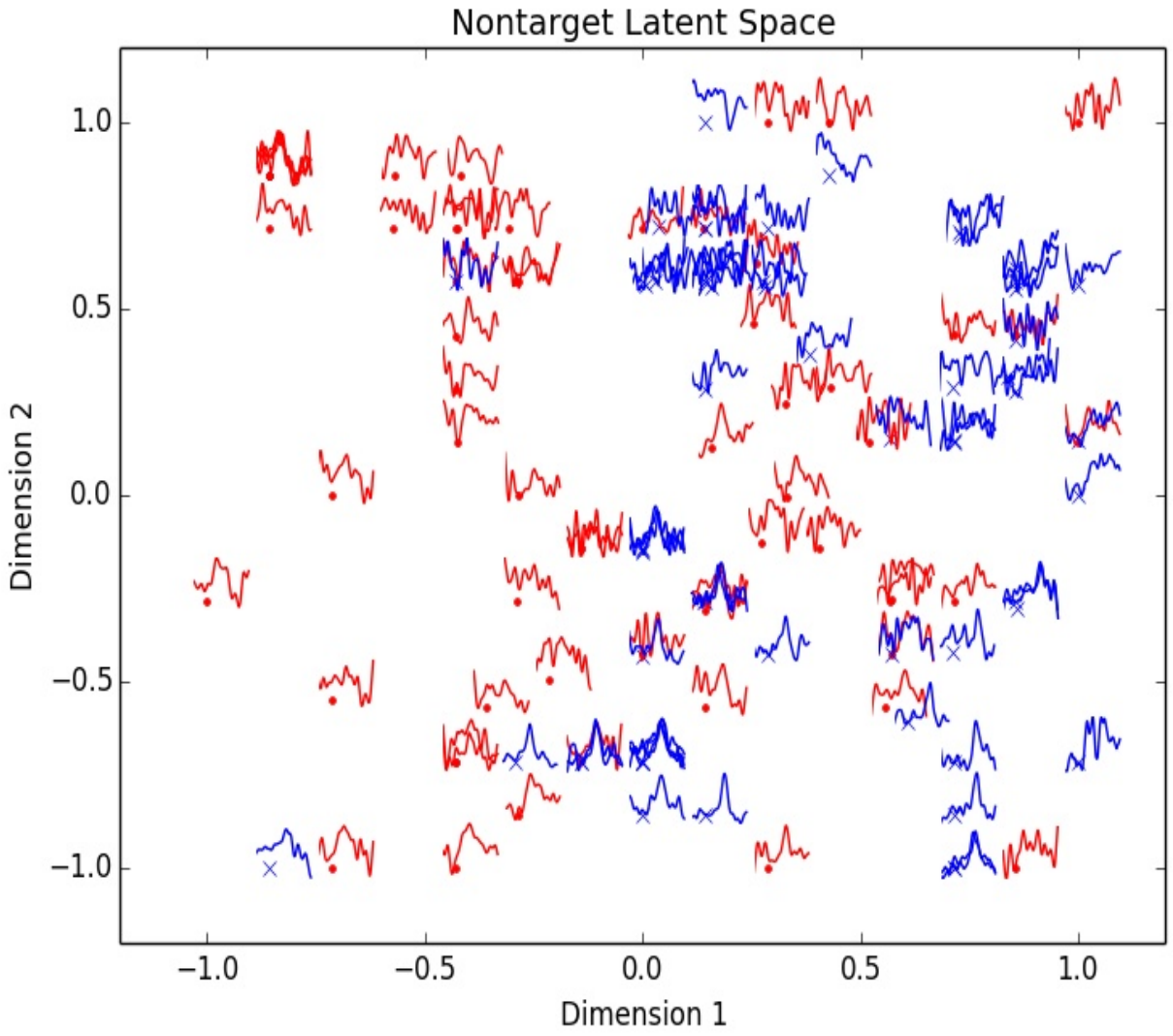


FIGURE 4.17. Latent space mapping of nontarget signals of P4 electrode EEG signal of subject 11 vs 23 for letter b protocol

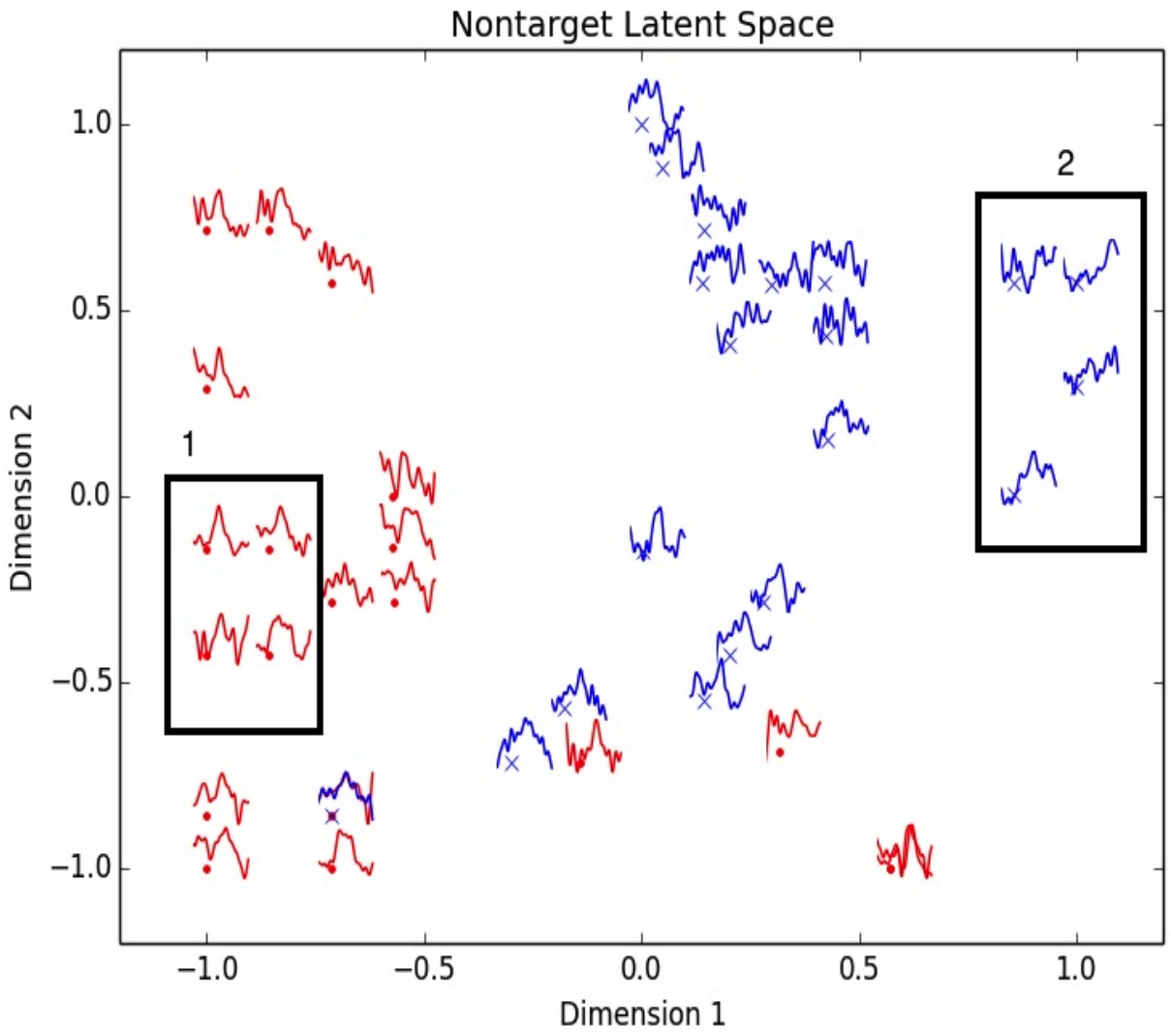
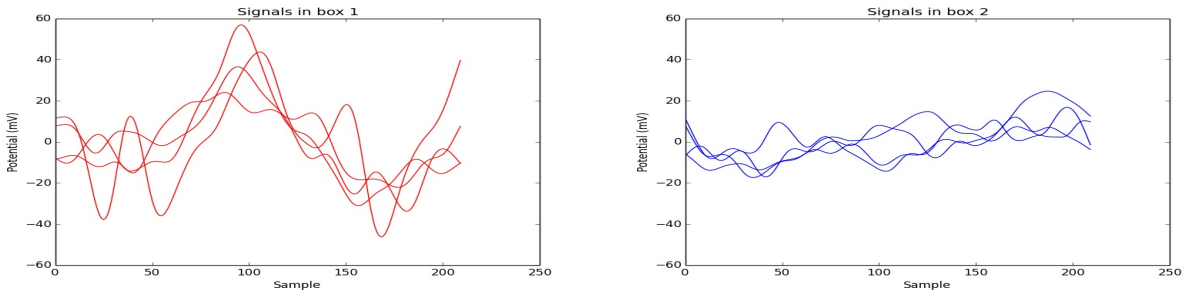


FIGURE 4.18. Latent space mapping of target signals of P4 electrode EEG signal of subject 11 vs 23 for letter b protocol



(A) Box 1 (B) Box 2

FIGURE 4.19. EEG signals mapped closeby on the latent space. Box 1 contain target signals of one subject. Box 2 contain target signals of the other subject

Figure 4.19 illustrates the target signals in the two boxes of Figure 4.18.

We can observe that the subject represented in red has clear expected P300 signals in the target signals mapped in the left side of the map. The subject represented in blue does not have obvious P300s in the target signals. Therefore, the separation between the subjects were not caused by any inherent feature of the subjects but because of the presence and absence of the P300 signal.

4.5.2. ANALYZING THE EFFECT OF THE NON-TARGET CHARACTERS. As mentioned in Section 3.1, the subjects were asked to count the number of occurrences of a particular letter during a trial. Letter b protocol trials needed them to count the letter b, letter d protocol trials needed them to count the letter d and letter p protocol trials to count the letter p. The character sequence which was used in all three protocols consisted of 20 'b' characters, 20 'd' characters, 20 'p' characters and 20 other characters in the alphabet that were neither b, p nor d. Therefore, in the letter b protocol trials, where letter b was the target, letters d and p were included in the non targets along with other letters. Likewise, letter d protocol trials had the letters b and p among the non targets and letter p protocol trials had the letters b and d among its non targets.

The subjects were first asked to perform the letter b protocol trials, then the letter d protocol trials and lastly the letter p protocol trials. Therefore, there is a speculation whether the already performed trials have any effect on the trials that were performed later. For example, when a subject is performing the letter d protocol trials, he has already performed the letter b protocol trials. So with the experience and memory of counting b's in the previous trials, when the letter b pops up in letter d trials as a nontarget, there is a possibility that the subject can get confused momentarily as to which character he is counting now and see b as the target or the subject can identify letter b as the target of the previous trial. In such situations, P300 waves will still be emitted for letter b in letter d trials. Even if the subject hasn't done any trial beforehand, he can still get confused because the letters b, d and p are relatively similar in appearance than other letters in the alphabet. Hence, if such scenarios exist, the nontarget signals will contain P300 waves, which hinders the classification performance.

In this section, we plan to analyze where the EEG signals of each separate character lie in the 2-D latent space. For example, in letter b protocol trials, we investigate where the nontarget signals of letter d and p appear on the 2-D space. Figures 4.20, 4.21 and 4.22 illustrate the latent space maps of the three protocols for subject 11, P3 electrode. The plots distinguish between the points of each letter by using a marker to represent the characters.

One apparent observation from all three figures, Figures 4.20, 4.21 and 4.22, is that the signals corresponding to the target letter is more or less confined to one side of the map. Meanwhile, signals of the nontarget characters are distributed throughout the map. That means that the area where the targets are mapped must be the area where signals with P300s are mapped on the latent space. Another observation in all three figures is that, in the areas of the maps where target signal density is relatively high, the density of the

nontargets corresponding to b, p or d is higher than the density of foil nontargets. This can mean that nontargets corresponding to letters b, p or d are more likely to produce EEG signals that are similar to target EEG signals. However, the evidence is not enough to come to a clear conclusion.

To analyze this effect, more datasets should be tested, and GTM can be used as a tool to efficiently visualize this effect. To quantify the results, one could check the false positives after classifying the 2D representation and see how many of those signals were generated by the nontarget characters that were targets in earlier experiments. Being a false positive means that the signal was mapped closed to the target cluster, hence has signals similar to targets. If most of the false positives were generated from other 2 nontarget characters in question, we can conclude that having targets of earlier experiments as nontargets in the current experiments has a major effect on classification accuracy of EEG data.

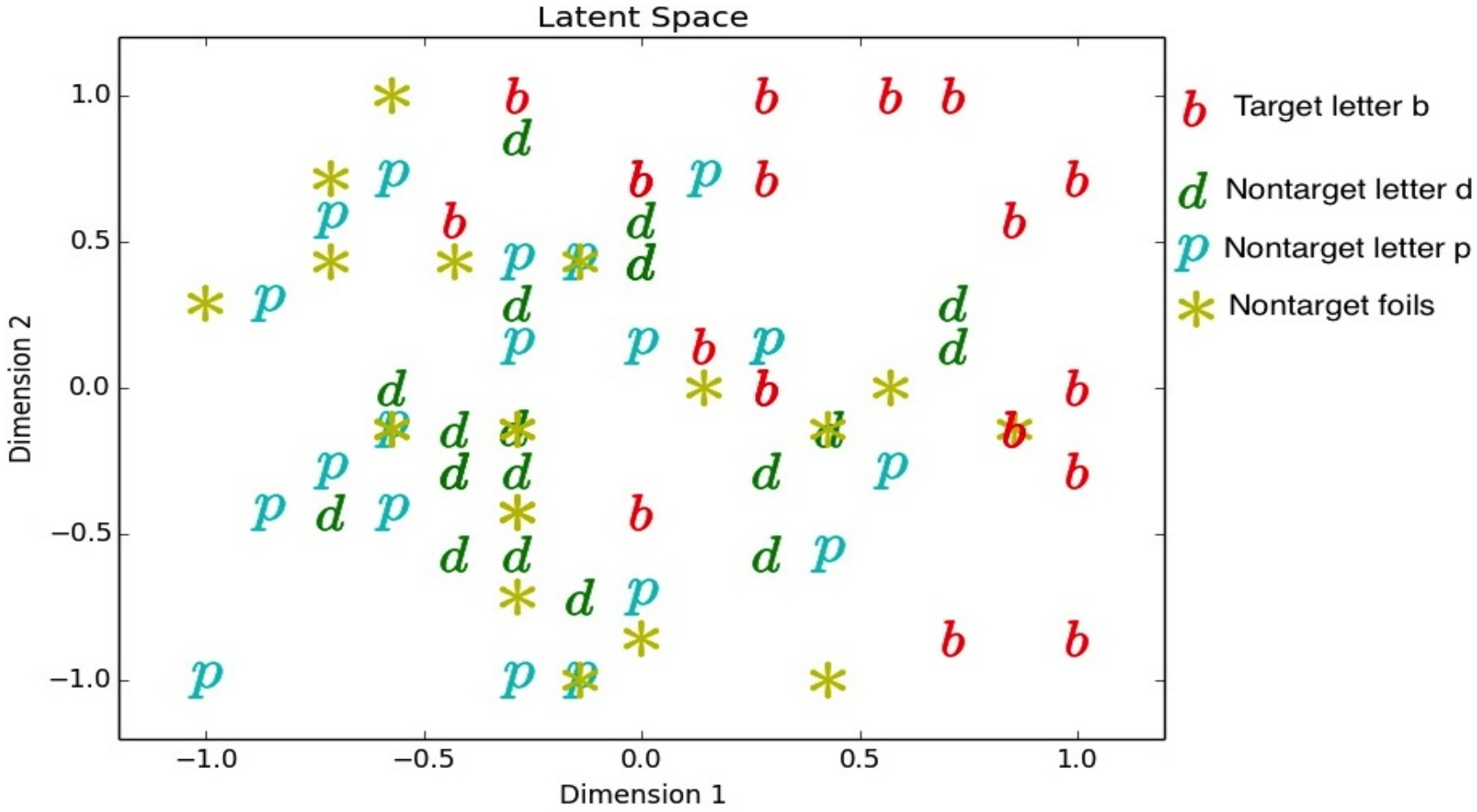


FIGURE 4.20. Latent space mapping of target letter b signals along with non target signals for letters d, p and others for subject 11, P3 electrode

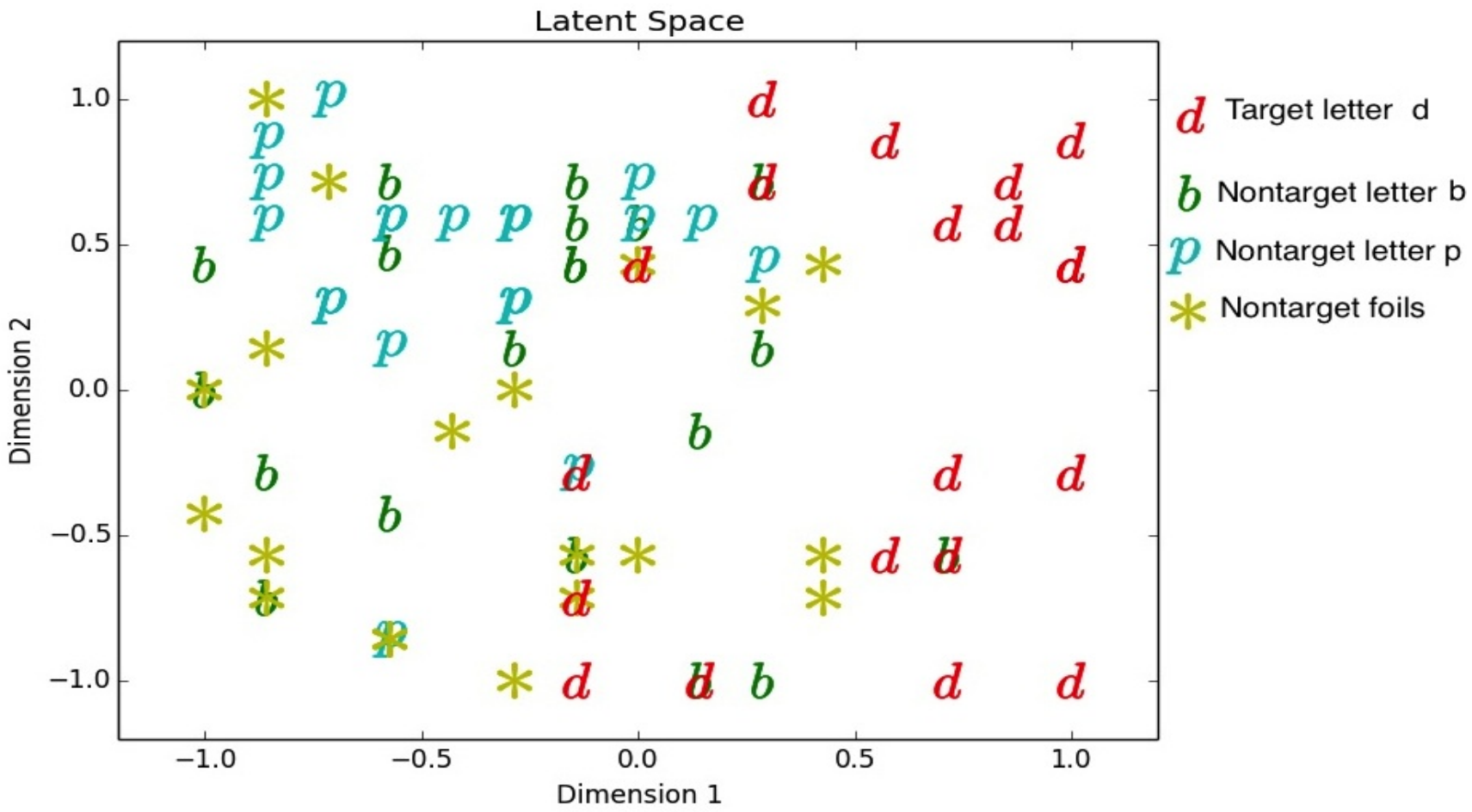


FIGURE 4.21. Latent space mapping of target letter d signals along with non target signals for letters b, p and others for subject 11, P3 electrode

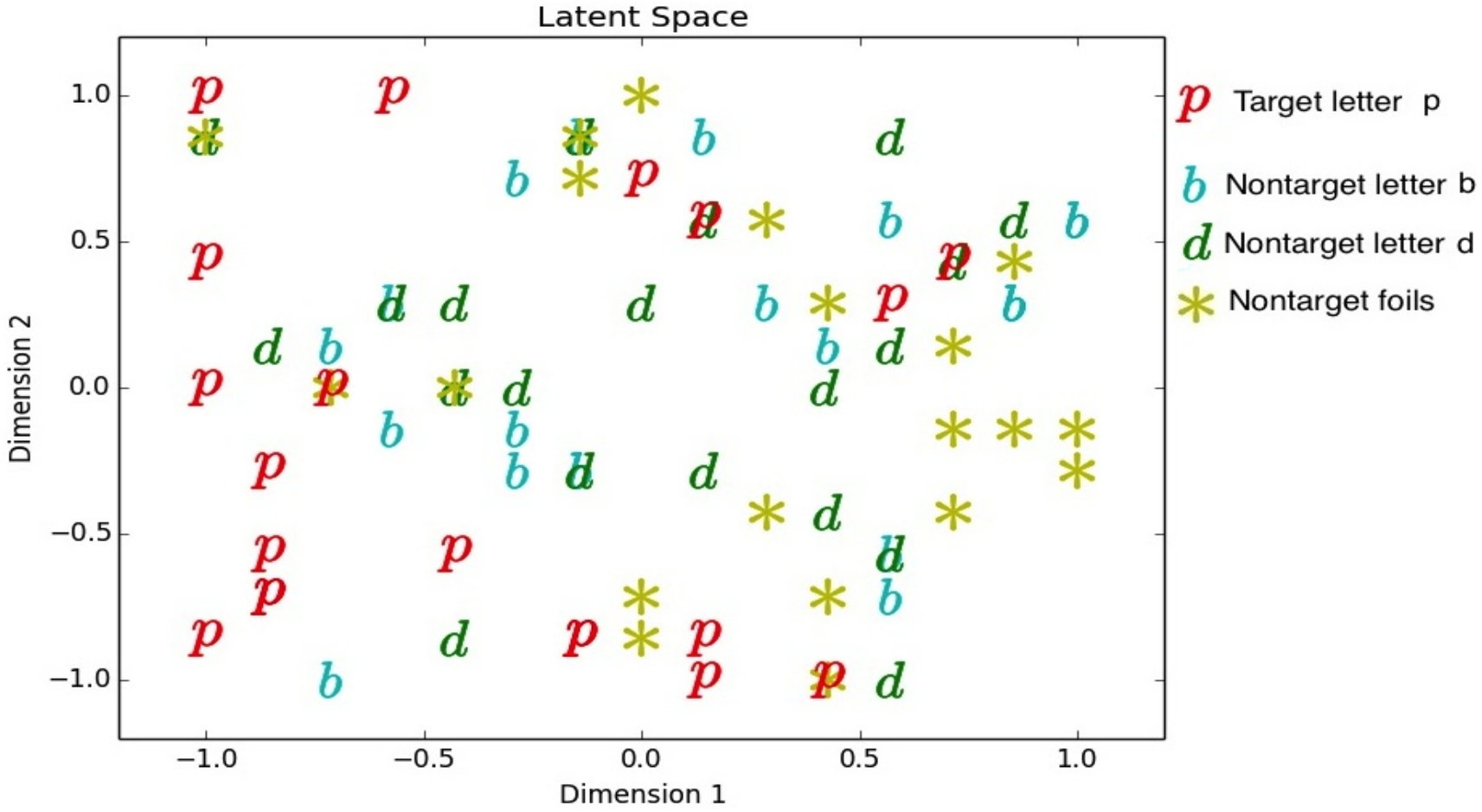


FIGURE 4.22. Latent space mapping of target letter p signals along with non target signals for letters b, d and others for subject 11, P3 electrode

CHAPTER 5

Conclusion and Future Work

This thesis was focused on presenting details of experiments carried out to investigate the performance of the GTM algorithm on EEG data. The GTM algorithm was implemented in Python and 42 different EEG datasets were used for the experiments. The GTM algorithm mapped the 210 dimensional EEG data on to a two dimensional map. This lower dimensional representation was then used for classification using a neural network.

In the experiments, I have first explored the effect of variation of algorithm parameters on the mapping process. The first parameter to consider was the number of sample points in the latent space. From results, we saw that too few sample points results in relatively independent Gaussian mixture centers in the data space and loss of smoothness properties in the mapping. However, after a certain number of latent points, the mapping was essentially the same without signs of overfitting, and causes no difficulty except the computational cost. This behavior is seen because the number of degrees in the model is controlled by the mapping function $y(x; W)$, which is an RBF network. We also tried changing the number of hidden units in the RBF network, but this resulted in no significant change of mapping for increased number of hidden nodes.

Next we analyzed the classification results of the GTM mapped EEG data. The classification was based on presence and absence of P300 waves in the target and nontarget EEG signals. The 2-D representation resulted from the GTM algorithm was classified using a neural network and was compared to the 210 dimensional EEG signal classification results, again with a neural network. Even though, for some datasets, the classification was improved, it wasn't very significant. This means that there was no clear separation between

the target and nontarget EEG signals on the latent space. Later on, when we analyzed the EEG signals mapped on the latent space, we could observe that signals mapped close together in the latent space actually do have similar EEG signals, regardless of whether they are target signals or nontarget signals. As an unsupervised algorithm, GTM only looks at the properties in the EEG signals to train itself for the mapping. Therefore, if there is no difference in the targets and non-targets, the algorithm wouldn't differentiate between them, resulting in latent maps that cannot be classified accurately. Hence, poor EEG data can be classified better with a supervised algorithm such as neural networks, rather than using an unsupervised GTM algorithm.

We also explored the ability of the GTM algorithm to identify properties of EEG signals without any supervision of labels provided. The experiments consisted of applying EEG data of different subjects and protocols to the GTM algorithm. The algorithm showed no differentiation between EEG signals from two subjects recorded using the same electrode and protocol, which is expected, because EEG signals don't depend on the subject. However, there was a little disparity between some subjects, and when investigating in more detail, we saw that the disparity was due to artifacts or complete absence of P300 waves in one subject and presence of it in the other. Also, there was no polarity in the mapping of EEG signals collected from the same subject, same electrode but for two protocols. This is expected, because P300 does not reveal any information about the stimulus being provided, but only about when the stimulus occurred.

The next step was to consider different methods of improving the classification accuracy of 2D data resulting from the GTM algorithm. One method was to concatenate data from P3 and P4 electrodes, so that the algorithm now has 420 time samples to learn the pattern from. However, the results indicated that unless each individual electrode's data have significantly

good EEG signals, the classification accuracy wouldn't be improved much. The other method was to use data from three protocols, so that the algorithm has more data samples to learn from. This again resulted in having an average of the three protocols' individual classification accuracies. These results are fitting to the argument that unless all data we are concatenating are good, it's similar to adding noise to the good data. Therefore, concatenating good EEG signals with relatively bad EEG signals reduces the algorithm's performance by misleading it during unsupervised training.

Lastly, we analyzed the EEG signals mapped on the latent space. It was very clear that signals with similar feature variations were mapped close together on the latent space. The distribution of the EEG signals on the latent space indicated the feature variations of the signals. Extending from this, we explored the effect letter b, d and p can have as a nontarget on the target classification. The 2-D GTM map gave a very easy way of analyzing where the EEG signals of each character lie on the latent space which in turn is an indication of the similarity between signals. It was noticeable that the nontarget b, d or p have more tendency of mapping close to the targets than the foils. However, evidence was not enough to conclude this speculation.

Therefore, we can conclude that the Generative Topographic Mapping of EEG data can be very helpful in EEG signal analysis. It has the ability of learning from the properties of the EEG signal, unsupervised. From the maps, we can conclude that this unsupervised learning is very accurate in finding the patterns of the EEG data. As for classification, GTM will not improve the classification unless the dataset contains good P300 signals for targets. However, the capability of visualizing the high dimensional EEG dataset in a two dimensional space is invaluable. Therefore, we can use the GTM algorithm to visually analyze the quality of an EEG dataset.

In this thesis, we have limited the number of dimensions in the latent space to just 2, for the ease of visualization. However, 2 dimensions may not be enough to encapsulate all the variations in the 210 dimensions of the EEG data. Therefore, the number of dimensions of the latent space can be increased to investigate whether more dimensions will give better clustering of EEG targets and nontargets in the latent space, which will, in return, improve the classification accuracy.

Also, the performance of the GTM algorithm in regards of dimensionality reduction should be compared to that of other commonly used dimensionality reduction techniques such as Principal Component Analysis (PCA), Neural Networks and feature selection methods. Additionally, the GTM algorithm can be extended in many ways. Estimating prior probabilities (π) using the EM algorithm rather than allowing independent mixture coefficient, generalizing the inverse noise variance (β) to be a function of latent space points (x) are a few of these extensions [3]. These suggestions can be implemented in order to improve the performance of the GTM algorithm on EEG data.

Bibliography

[1] P. Ponce, A. Molina, D. C. Balderas, and D. Grammatikou, *Cerebral Palsy - Challenges for the Future*. InTech, 2014.

[2] A. Nikolaev, "Eeg Recording." Availabel online at *http://www.aha.ru/ geivanit/EEG-manual/Recording.htm*, 1998.

[3] C. M. Bishop, M. Svensn, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, pp. 215–234, 1998.

[4] A. Farnell, "AI sound." Available online at *http://obiwannabe.co.uk/html/papers/proc-audio/node10.html*.

[5] P. Forslund, "A neural network based braincomputer interface for classification of movement related EEG." Ph.D. Thesis, Linkoping University, 2003.

[6] B. R. Brooks, R. G. Miler, M. Swash, and T. L. Munsat, "El escorial revisited: Revised criteria for the diagnosis of amyotrophic lateral sclerosis," *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*, vol. 1, pp. 293–299, 2000.

[7] Z. Breznitz, *Brain Research in Language*. Springer, 2008.

[8] R. Sitaram, N. Weiskopf, A. Caria, R. Veit, M. Erb, and N. Birbaumer, "fMRI Brain-Computer Interfaces," *Signal Processing Magazine, IEEE*, vol. 25, no. 1, pp. 95–106, 2008.

[9] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Braincomputer interfaces for communication and control," *Clinical Neurophysiology*, vol. 113, no. 6, pp. 767 – 791, 2002.

[10] D. J. Krusienski, E. W. Sellers, F. Cabestaing, S. Bayoudh, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw, "A comparison of classification techniques for the P300 speller," *Journal of Neural Engineering*, vol. 3, pp. 299–305, 2006.

[11] L. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and Clinical Neurophysiology*, vol. 70, pp. 510–523, 1988.

[12] N. V. Manyakov, N. Chumerin, A. Combaz, and M. M. Van Hulle, "Comparison of classification methods for p300 brain-computer interface on disabled subjects," *Intell. Neuroscience*, vol. 2011, pp. 2:1–2:12, Jan. 2011.

[13] H. A. Gaspar, G. Marcou, D. Horvath, A. Arault, S. Lozano, P. Vayer, and A. Varnek, "Generative topographic mapping-based classification models and their applicability domain: Application to the biopharmaceutics drug disposition classification system (BDDCS)," *Journal of Chemical Information and Modeling*, vol. 53, no. 12, pp. 3318–3325, 2013.

[14] A. Kaban, "A scalable generative topographic mapping for sparse data sequences," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 1, pp. 51–56 Vol. 1, April 2005.

[15] F. Zhong, X. Zheng, Z. Tan, and T. Shi, "Application of generative topographic mapping to the classification of bearing fault," in *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*, pp. 3095–3098, May 2007.

[16] W. El-Deredy, P. J. G. Lisboa, and A. Vellido, "Studying embedded human eeg dynamics using generative topographic mapping." Technical Report, Universitat Politecnica de Catalunya (UPC), Barcelona, Spain.

[17] R. M. Sabbatini, "Eeg brain mapping." Available online at *http://www.cerebromente.org.br/n03/tecnologia/historia.htm*, 1997.

[18] D. L. Schoer, F. L. da Silva, and E. Niedermeyer, *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams and Wilkins, 6 ed.

[19] H. H. Jasper, "Report of the committee on methods of clinical examination in electroencephalography: 1957," *Electroencephalography and Clinical Neurophysiology*, vol. 10, p. 370375.

[20] H. G. V. Jr. and W. Ritter, "The sources of auditory evoked responses recorded from the human scalp," *Electroencephalography and Clinical Neurophysiology*, vol. 28, no. 4, pp. 360 – 367, 1970.

[21] I. T. Nabney, *NETLAB: Algorithms for Pattern Recognition*. Springer, 2004.

[22] C. M. Bishop, "Mixture models and the EM algorithm," 2006. Available online at *http://www.cs.ubbcluj.ro/ csatol/gep-tan/Bishop-CUED-2006.pdf*.

[23] C. Anderson, "Classification by nonlinear logistic regression using neural networks," 2014. Available online at *http://nbviewer.ipython.org/url/www.cs.colostate.edu/ anderson/cs645notebooks/Neural%20Networks%20for%20Classification.ipynb?create=1*.

[24] T. G. Dietterich, "Learning neural networks," 2005. Available online at *http://web.engr.oregonstate.edu/ tgd/classes/534/slides/part4.pdf*.

[25] M. Moller, "Effcient training of feed-forward neural networks." Ph.D. Thesis, Aarhus University, 1997.

[26] C. Anderson, "Neural networks 2." Lecture available online at *http://nbviewer.ipython.org/url/www.cs.colostate.edu/anderson/cs645notebooks*, 2014.

[27] C. S. U. Department of Computer Science, "2011-12 bci at csu," 2011-12. Available onile at *http://www.cs.colostate.edu/eeg/main/data*.

[28] G. M. E. GMBH.

[29] P. H. Eilers, "A perfect smoother," *Analytical chemistry*, vol. 75.14, pp. 3631–3636.

[30] J. J. Stickel, "Data smoothing and numerical differentiation by a regularization method," *Computers and Chemical Engineering*, vol. 34, pp. 467–475.

[31] C. Anderson, "Smoothing p300 data and classifying trials." Available online at *http://nbviewer.ipython.org/url/www.cs.colostate.edu/ anderson/cs645notebooks/*, 2014.

[32] C. Anderson, "Smoothing P300 data and classifying trials," 2014. Available online at *http://nbviewer.ipython.org/url/www.cs.colostate.edu/ anderson-/cs645notebooks/P300%20Smoothing%20By%20Regularization.ipynb?create=1.*

[33] R. Gutierrez-Osuna, *Radial Basis Functions.* Lecture available at *http://research.cs.tamu.edu/prism/lectures/pr/pr-l19.pdf.*

[34] J. F. M. Svensen, "GTM: the generative topographic mapping." Ph.D. Thesis, Aston University, 1998.