APPLICATION OF SUPPORT VECTOR MACHINES TO LONGITUDINAL FUNCTIONAL NEUROIMAGING DATA

by

Alexander Adam Rudiuk

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
November 2016

For my Mother and Father

List of Tables

List of Figures

Abstract

The principal objective of this thesis was to test a novel adaptation of the support vector machine (SVM), called a longitudinal support vector machine(LSVM), on longitudinal functional neuroimaging data. LSVM performance was compared to a traditional SVM and logistic regression (LR) using classification accuracy and interpretability of feature weights. Classification accuracy was measured as the percentage of subjects placed into their correct categories, and feature weights by how closely they matched the known signal. The first study involved purely simulated data, which found the LSVM had higher classification accuracy for data without heteroscedasticity, but performed worse when heteroscedasticity was introduced. The second study used real magnetoencephalography (MEG) resting state readings added to a simulated trend. The LSVM had similar classification accuracy, and only had more interpretable feature weights at the highest SNR dataset. Currently the LSVM is not recommended over the SVM/LR algorithms.

List of Abbreviations and Symbols Used

AD – Alzheimer's Disease

CVA – Canonical Variate Analysis

EEG – Electroencephalography

fMRI – Functional Magnetic Resonance Imaging

HC – Healthy Control

ICA – Independent Component Analysis

LOOCV – Leave One Out Cross Validation

LR – Logistic Regression

LSVM – Longitudinal Support Vector Machine

MCI – Mild Cognitive Impairment

MEG – Magnetoencephalography

MRI – Magnetic Resonance Imaging

PCA – Principle Component Analysis

SNR – Signal-to-Noise Ratio

SQUID – Superconducting Quantum Interference Device

SVM – Support Vector Machine

x – Input

y – Target

w – Weights

G – Gram Matrix

α – Support Vectors

b – Bias

K – Kernel

C – Regularization Term

t – Session Number

N – Subject Count

β – Beta Term of LSVM

m – Mean

N – Normal Distribution

σ^2 – Standard Deviation at Session 1

ϕ^2 – Standard Deviation at Session 2

τ – Scaling Term

ψ – Heteroscedasticity Term

Acknowledgements

I would like to thank my thesis supervisors, Dr. Tim Bardouille and Dr. Steven Beyea, of the School of Biomedical Engineering at Dalhousie University. Dr. Bardouille was always available to provide guidance when I ran into trouble with my research and writing. Dr. Beyea also provided valuable insight into my writing. They allowed for this paper to be my own work, but helped steered me.

I would also like to thank my committee, Dr. Thomas Trappenberg and Dr. Robert Adamson, for their expert guidance. Dr. Trappenberg primarily for his help with the machine learning aspects of the project, and Dr. Adamson for his help with developing comparison models.

I would also like to thank BIOTIC and my lab members: Ronald Bishop, John Lincoln, Santosh Vema, and Sarah McLeod. I thank Dr. Steve Patterson for his discussions about machine learning, statistics, and life which helped me throughout my thesis. I would like to thank Jessica Luedi for help with editing this paper. I would also like to thank my friends Max Wawer, Carl Kooka, and Scott Cameron who have provided me with help throughout my degree.

Finally, I would like to acknowledge my parents for always supporting me through the good times and bad. Without their constant support, I would not be the person I am today. Thank you.

Chapter 1 Introduction

Interpreting functional neuroimaging data is a complicated task. Collected data is typically high dimensional and low sample size. The analysis is further complicated when the data is collected over multiple sessions (i.e., longitudinally). Time between sessions typically varies between subjects, meaning that changes in measured variables might not be consistent between different subjects. Also, responses might be correlated (not independent) as a variables' measurement at one session can be related to its measurement at other sessions. The variance of a variable might not be consistent over time (heteroscedasticity). For longitudinal data, heteroscedasticity means that repeated measures of a variable can have different distributions. Lastly, it is more likely that measurements closer in time are correlated than measurements that are farther apart in time [1-3].

To analyse functional neuroimaging data, highly regularized models need to be used. Two common regularized models are the support vector machine (SVM) and logistic regression (LR). In this thesis, I test a novel addition to the SVM, developed by Chen and Dubois[4], which explicitly takes into account the longitudinal nature of sessioned functional neuroimaging data using a model called the longitudinal support vector machine (LSVM). This work will develop an understanding of whether the LSVM has a role in predicting long term changes for patients by extending the simulations of Chen and Dubois to evaluate the LSVM performance with variation of more input parameters, and by evaluating the algorithm on simulated neuroimaging data. Classifiers that perform well with longitudinal data may have value in predicting recovery of a patient after a brain injury (prognosis). This has the possibility of giving clinicians a new tool to use when determining next steps to take during treatment.

1.1 Functional Neuroimaging

1.1.1 Measuring Brain Activity with Functional Neuroimaging

Functional neuroimaging machines measure an indicator of brain activity, such as electrical potentials on the scalp, magnetic fields propagating from the head, or metabolic changes in neurons that affect blood flow to brain regions. A typical

paradigm would have the subject performing a timed task, for which related brain activity is recorded. Functional imaging provides insight into the roles of different brain regions by being able to detect where activity is occurring during tasks. This differentiates functional neuroimaging from structural imaging, because structural imaging is not able to show where the activity is occurring. Structural imaging will only reveal the physical structure of the brain.

Functional neuroimaging is useful for studying the roles of different brain regions during tasks, and can help measure functional changes to the brain when no macroscopically apparent anatomical change is present. As an example, adults suffering from mTBI show subtle changes in their functional neuroimaging data not detected with purely neuropsychological tests, and where less than 5% of adults show signs of a distinct lesion [5, 6]. This provides evidence for functional neuroimaging being an important tool for finding indicators of change.

1.1.2 Modalities of Functional Neuroimaging

Three popular functional neuroimaging modalities are electroencephalography (EEG), functional magnetic resonance imaging (fMRI), and magnetoencephalography (MEG). EEG and MEG measure activity corresponding to electrophysiological changes in neuronal populations that are time-locked to a stimulus or task [7]. These time locked changes are called event-related potentials or event-related fields, as they refer to changes in brain activity due to some event. This event can be a visual stimulus, auditory stimulus, or physical stimulus, as well as a physical response to the stimulus. Since brain activity of interest is time locked to the event stimulus, background brain activity can be reduced by averaging. Averaging across multiple time segments around an event of interest therefore improves the signal-to-noise ratio (SNR). The signal being recorded by EEG/MEG is primarily generated by post-synaptic potentials.[8]. The post-synaptic potential at a single neuron is too small to produce a detectable signal; the measurements obtained represent the summed activity of many neurons. These measurements are either of the electric potential produced at the scalp by the post-synaptic potentials (EEG), or of the associated magnetic field (MEG). EEG and MEG

provide complimentary information as the electrical and magnetic fields are perpendicular to each other. Both measurements have a temporal resolution on the order of milliseconds due to the nature of bioelectromagnetic fields [9]. In contrast, fMRI uses blood oxygenation level dependent (BOLD) contrast [10], which is dependent on the blood flow to travel to regions of activity (i.e., neurovascular response), giving a lower temporal resolution on the order of seconds. EEG has poorer spatial resolution than MEG since the electrical potentials are spatially "smeared" by the high conductivity in the skull and scalp [8, 11].

1.1.3 Magnetoencephalography

MEG measures the magnetic fields of the brain using superconducting quantum interference device (SQUID) sensors [12]. SQUID sensors are able to measure the extremely small magnetic fields generated by brain activity; they use superconducting material that can be affected by the small magnetic fields of the brain [13]. SQUID sensors provide the required sensitivity to detect the brain's magnetic fields, but larger magnetic fields in the environment need to be removed to improve the signal-to-noise ratio (SNR) of measured brain activity [14].

To achieve improved SNR, typical MEG data is collected over a period of time and then split into discrete sections synchronized to events of interest (epoched) and averaged (evoked). The data is split into discrete sections based on when event triggers occur. This would be an auditory tone or some other stimuli. The MEG data collected after this trigger shows a signal that relates to the recent event. In the case of an auditory tone, the signal would be active in both auditory cortices.

As mentioned above, the magnetic fields being measured with MEG come from currents in the brain. If the primary source and surrounding conductivity distributions are known, the resulting magnetic field signal can be calculated [15]. The solution is modelled by using equivalent current dipoles as sources of signal in the brain, which approximate the flow of electrical current in a small area. It also assumes there is no propagation delay in the signal and no temporal derivative (quasi-static). Under these

assumptions, the magnetic field perpendicular to an MEG sensor can be calculated using the equation

$$B_z = \frac{\mu_0}{4\pi} \frac{Q \times (r - r_Q) \cdot e_z}{|r - r_Q|^3} \quad (1)$$

B_z is the magnetic field perpendicular to the sensor, r is the point where the field is computed (i.e. location of the sensor), e is given by $\frac{r}{|r|}$, r_Q is the current dipole location, and μ_0 is a permeability constant. Using this equation for the generation of magnetic fields, an equation to compute the forward model is given by

$$M = GX \quad (2)$$

M is the matrix of sensor readings, G is the forward solution described in equation (1), and X is the matrix of source amplitudes over time. By knowing the location and orientation of the source current dipoles (X) and the forward solution (G) the sensor readings can be calculated. The inverse of this task is called source estimation, where the source is estimated based on sensor readings.

1.1.4 Current Univariate and Multivariate Analysis Approaches

There are two common approaches to statistical analysis of functional neuroimaging data: univariate, and multivariate methods [16]. Data used for analysis in functional neuroimaging is typically low sample size and with a high number of dimensions. This holds true for sensor level data (i.e. B_z), and for source estimate data (i.e., current dipoles Q). Inferential methods use a statistical model on a per-voxel basis and tests variations from the null hypothesis. A voxel, in this case, refers to the smallest unit of a 3D image. The null hypothesis states that there is no statistically significant variation between one set of data and another. If there is a statistically significant variation, the null hypothesis is rejected. Non-inferential methods try to characterize the nature of the signal in the data without depending on a particular model, essentially finding patterns in the data [17].

Inferential methods use a null hypothesis that is assessed with a test statistic (a function sensitive to departure from the null hypothesis and looking at the region of interest), which gives a score of how much a voxel deviates from a voxel that is not activated. This score is a measure of the support behind a null hypothesis and is used to either accept or reject it [18]. An example would be a time series where the magnitude of activation in a voxel under one observation is compared to itself at another observation. These observations might be during different tasks the subject is asked to perform. If the value at the voxel is determined to be significantly different between the two observations, via a test against the null hypothesis, then there is said to be a change in activity at that voxel.

Examples of multivariate exploratory methods used in functional neuroimaging are principle component analysis (PCA), and independent component analysis (ICA). PCA identifies patterns in the data that are orthogonal to each other, while ICA identifies patterns that are statistically independent and spatially sparse with respect to each other. PCA requires that the principle components be orthogonal, while ICA puts no constraints on orthogonality [19]. When applied to functional neuroimaging data, these methods involve determining the components, and inspecting the projection of the components onto the data as an indicator of experimental effects. The projected data is analyzed first, and then labeled with an interpretation in the context of the experimental design [16].

In contrast with multivariate methods, univariate approaches are more focused on testing hypothesis related to a single variable [20]. This is because univariate approaches are testing a specific question about whether certain voxels show changes in activity during certain tasks, and whether these changes in activity are statistically significant. Multivariate approaches, such as PCA and ICA, generate spatial patterns of activity, and then leave the interpretation of these generated patterns to the researcher. Univariate methods, therefore, allow for a stronger link to the experimental variables when performing studies using functional neuroimaging techniques. A research question that could be asked with multivariate methods is “What patterns of

activity explain variations across the set of brain maps?", while with inferential statistics it would be "Does the activation level of a voxel vary significantly as a function of experimental condition?". Since univariate analysis is done on a per voxel basis, the hypothesis is phrased in terms of voxel-wise activation and does not look at possible shared activations that could be occurring, which is available with a multivariate approach.

1.1.5 Flaws of Univariate and Multivariate Approaches

There are several limitations associated with inferential and multivariate methods that may cause problems when answering certain types of questions in the field of functional neuroimaging. Voxel-based inferential statistics treats every voxel as being an independent data point. This is flawed as brain activity occurs in "blobs" of voxels that are correlated to each other.

Another problem with inferential techniques is that they run into type 1 error. Type 1 error is the incorrect rejection of a null hypothesis – meaning that a change is considered to be statistically significant when it is actually not. In terms of functional imaging, it means that certain voxels are considered to be activated when they are not. Since there are many non-activated voxels in a functional image, the model would lean towards predicting a voxel is activated to match the probability of the model, creating falsely identified activated voxels. Unfortunately, corrections for this tend to push the result towards type 2 error. Type 2 error is not rejecting the null hypothesis when it is false – claiming that an effect is not significant when it is.

Many multivariate exploratory methods commonly applied to functional neuroimaging data are flawed because they do not directly connect data to the experimental conditions [16]. For example, consider performing principle component analysis (PCA) on a set of MEG data. PCA will return components to the researcher representing different patterns, each component explaining some amount of the variability in the data [21]. Researchers can then project data for each condition onto these components, and relate the resulting data to the experimental conditions of the experiment. Even if a component explains a large amount of variance in the data, the

researcher still needs to relate what it means in the context of the experiment. Interpreter bias and reproducibility become serious issues given that there is no systematic way of correlating variance in an image to experimental conditions. As well, some variations might be as large as artifacts (e.g., head movement), making it difficult to select proper components [16].

Human bias and error present obstacles when analysing results from techniques like PCA. Since the components being produced need to be interpreted by the researcher, the conclusions reached are subject to bias. Subjectivity in scientific results is not desirable as it makes it difficult to reproduce the same results. Error in interpretation is also a risk as the high number of dimensions in the components can be difficult to interpret. Researcher bias can cause several different issues as findings may be motivated by what the researchers believes is the correct interpretation. If a researcher spent significant time on a project, for example, they might be more inclined to see positive results.

1.2 Machine Learning

1.2.1 What is Machine Learning?

A popular formal definition of machine learning attributed to Dr. Mitchell is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"[22]. Put more simply, machine learning is the study of algorithms that improve at some task when given experience in performing said task. There are three main classes of machine learning algorithms: supervised, unsupervised, and reinforcement learning. My thesis will focus on the application of supervised machine learning to functional neuroimaging data. In supervised learning, the computer is given the inputs (features) and the correct outputs (targets). The supervised learner then tries to develop a pattern classification model to predict the outputs from future inputs using the discovered patterns in the data [21].

1.2.2 What is a Classification Model?

Classification approaches to functional neuroimaging data associate exemplars of functional data with a discrete set of classes. Supervised machine learning is common for functional neuroimaging studies, because it allows the researcher to classify the functional neuroimaging data into chosen classes [4, 23-26]. The classification accuracy is determined by how many test sets are predicted to be in the correct class, once the algorithm has been trained to classify. By looking at what patterns of brain activity help drive classification accuracy, the relevancy of those patterns to certain tasks can be assessed. How the input data is presented to the algorithm will determine what information the algorithm uses to classify, and leads to different representations of the patterns driving accuracy. Then, when new data is introduced, the supervised learning algorithm should be able to predict which class the functional neuroimaging data belongs to.

1.2.3 Classification Models/Algorithms

Pattern based classification is a promising new way to analyze functional neuroimaging data. It is a subset of the field of machine learning. To give an idea of the benefits of using machine learning over more classic approaches, consider the following example [27]. Using fMRI, it is possible to distinguish if a participant is looking at an image of a face or a house without the use of computerized pattern based classification. One way this is done is by looking at the magnitude of activation in the fusiform face area, which responds strongly to faces, and the parahippocampal place area, which responds strongly to images containing views of houses and visual scenes. By simply being given the activity levels in these two regions, human observers were able to correctly identify the class of object the participant was looking at in 85% of trials. More difficult classification tasks like predicting the orientation of an object, which causes a very fine change in brain activity patterns, can not be solved with univariate approaches and need a machine learning algorithm to operate on patterns. An example is the accurate classification of visual stimulus class, even amongst multiple categories with overlapping brain activation patterns [28, 29]. In these cases, it is possible to increase sensitivity by distinguishing what object a subject is looking at,

even when regions overlap, using classification algorithms like support vector machines. A good example of how pattern based classification techniques are much more powerful than using univariate models is that univariate models are unable to detect when an object is rotated, even if data was gathered for hundreds of scans. Pattern recognition techniques were able to achieve 80% accuracy on just a single fMRI scan that was collected in under 2 seconds, where a single scan refers to a single measurement for every voxel of the brain [30].

1.2.4 Support Vector Machines

One of the most popular machine learning algorithms is the support vector machine (SVM). This supervised learning algorithm has been shown to classify relatively well when inputs have a large number of dimensions (or "features") and low number of class examples, which happens to be the case with most functional neuroimaging data [28]. SVMs are extensively used in modern functional neuroimaging studies [31-33]. Initial excitement in their use might stem from the results of a competition in 2003, where an SVM was able to perfectly classify the data [24, 34]. The data was composed of EEG recordings during a character selection task. Healthy controls (HCs) were shown matrices with 36 characters (6x6) and had random characters highlighted in the matrices. While the order was random, the characters were highlighted an equal amount per session. Each session had a character highlighted 12 times. Accuracy of the classifier was measured by having it predict what character the subject was looking at. The SVM was able to achieve perfect classification using only 5 sequences, the least of all tested classifiers. It also only needed ~16% of the original electrodes to achieve this accuracy.

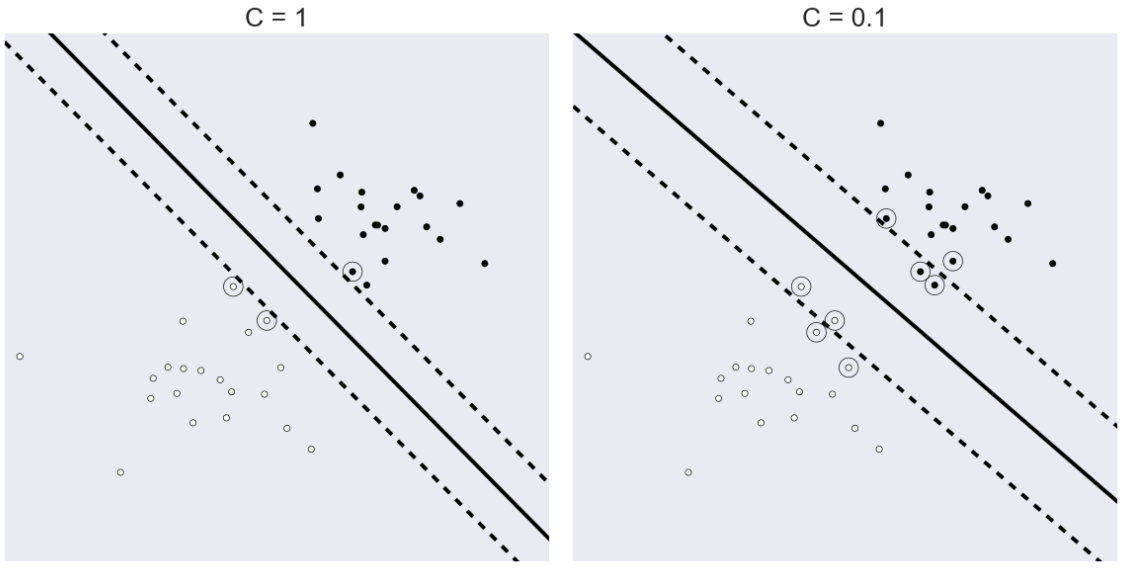

Figure 1 - Svm Classification Example

This example shows the SVM classifying two different data sets at different C values. The X and Y axis are not shown as their values are arbitrary. White circles represent class one, black circles represent class 2, and double circles are the support vectors. A C value of 1 is used for the left plot, and a C value of 0.1 is used for the right plot. Solid black lines represent the hyperplane and dashed black lines represent the margin. As the value of C decreases, the error tolerance increases, in turn increasing the width of the margin. As the width of the margin increases, more points are treated as support vectors and end up being within the margin.

An SVM operates by taking in the input data designated for training and trying to separate input vectors associated with different classes by using a hyperplane, as shown in Figure 1. Inputs to the SVM are composed of different variables, known as features.

$$x_s = [x_1, x_2, \dots, x_n], s \in (1, \dots, N) \quad (3)$$

In equation (3) the input x_s is composed of n features, with there being N subjects in total. In the case of MEG data, each feature would be a reading from a sensor at a specific time point. However, data can be preprocessed to reduce the number of features that are input to the SVM (e.g., *a priori* time/sensor selection, PCA, ICA, etc.). Each subject also has a corresponding target, y_s that defines the class

$$y_s \in [-1, 1] \quad (4)$$

The goal of the SVM algorithm is to correctly classify subjects into each class. Classification accuracy is defined as the percentage of correctly classified inputs. An SVM classifies inputs into two categories by separating the data with a hyperplane

$$h(x_s) = wx_s + b = \begin{cases} 1, if \geq 0 \\ -1, if < 0, \end{cases}, ||w|| = 1 \quad (5)$$

w is a vector of weights which multiplies all the features. Since it is of unit length, the dot product between w_s and x_s gives the projection of the input onto the weights. This result is a scalar value, which decides whether the input belongs to class 1 or class 2 (1 and -1). b is a bias term. The closest inputs to the hyperplane of each class are called the support vectors. While there may be many hyperplanes that can separate the classes, the optimal one is the one which maximises the margin, the distance between the hyperplane and the support vectors, as shown in Figure 1, where the dashed lines are the boundaries of the margin. Vapnik showed that an optimal hyperplane can be defined as the one which maximizes the distance to the closest points from either class [21, 35]. This formulation of the problem allows for a unique solution to the problem, and by assuming a simple separation rule, performance on unseen data is improved. This has been shown to be equivalent to minimizing

$$\min_{w,b} \frac{1}{2}||w||^2 \quad s.t.\ y_s(w \cdot x_s + b) \geq 1 \qquad (6)$$

It is convenient to formulate the minimization problem in the form of a Lagrangian

$$L = \frac{1}{2}||w||^2 - \sum_s^N \alpha_s[y_s(w \cdot x_s + b) - 1] \qquad (7)$$

α is the lagrangian variable, where only the support vectors have non-zero values. To find the dual of the Lagrangian, the partial derivatives with respect w and b are taken giving

$$w = \sum_s^N \alpha_s\, y_s x_s \qquad (8)$$

and

$$0 = \sum_s^N \alpha_s\, y_s \qquad (9)$$

Using equation (8) and (9) the dual of the Lagrangian is obtained by substitution

$$L = \sum_s^N \alpha_s - \frac{1}{2}\sum_s^N \sum_{s'}^N \alpha_s \alpha_{s'} y_s y_{s'} (x_s \cdot x_{s'}) \qquad (10)$$

Equations (8) and (9) can also be plugged into the hyperplane equation (5) to obtain the hyperplane equation in terms of the Lagrangian variables

$$h(x_s) = \sum_{s'}^N \alpha_i y_i (x_{s'} \cdot x_s) + b \qquad (11)$$

An important thing to notice in equations (10) and (11) is that only the dot product of inputs is necessary for computation, not the inputs themselves. This becomes useful when wanting to use non-linear separators. Not all data can be separated with a linear hyperplane. In that case, the input features are mapped to an alternate dimension (usually higher) using a kernel function. The kernel is a mapping of the input vector from its original set of features to a new feature space, where the mapping can be linear or not linear [26]. By transforming the data to a new feature space, one might be able to find a linear separator in this new feature space and use the same algorithm to find a linear hyperplane. Importantly, instead of mapping the

inputs to a new coordinate frame, the dot product can be mapped instead. This is called the *kernel trick*. This changes equation (11) to

$$h(x_s) = \sum_{s'}^{N} \alpha_i y_i K(x_{s'}, x_s) + b \qquad (12)$$

where *K* is the kernel function mapping the dot product to the new feature space.

One important property to note about equations (11) and (12) is that these solutions do not allow for incorrect classification. To get around this, a slack variable ξ is added to equation (13).

$$\begin{gathered} \min_{w,b} \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{s=1}^{N}\xi_s \\ s.t.\ y_s(w \cdot x_s + b) \geq 1 - \xi_i \quad \xi_s \geq 0 \end{gathered} \qquad (13)$$

ξ measures the distance from the proper side of the hyperplane an input is allowed to be during training. *C* is the parameter controlling error tolerance. Making the value of C larger in equation (13) will punish the SVM more for having examples on the wrong side, resulting in smaller margins. Smaller values of C will punish the classifier less for misclassification and allow for larger margins. Taking the slack variable into account and finding the Lagrangian minimization problem in the form of equation (10) gives

$$\begin{gathered} \min_{\alpha} \frac{1}{2}\sum_{s,s'} a_s a_{s'} y_s y_{s'} K(x_s, x_{s'}) - \sum_{s=1}^{N}\alpha_s \\ s.t.\ C \geq a_s \geq 0\ and\ \sum_s \alpha_s y_s = 0 \end{gathered} \qquad (14)$$

Finally, the notation can be greatly simplified if the y values and the kernel K are put into a matrix called the Gram matrix

$$\begin{gathered} \min_{\alpha} \frac{1}{2}\alpha^T G\alpha - \alpha \\ s.t.\ C \geq a_s \geq 0\ and\ \sum_s \alpha_s y_s = 0 \end{gathered} \qquad (15)$$

Where

$$G = \begin{bmatrix} y_1 y_{1'} K(x_1, x_1) & \cdots & y_1 y_{N'} K(x_1, x_N) \\ \vdots & \ddots & \vdots \\ y_N y_{1'} K(x_N, x_1) & \cdots & y_N y_{N'} K(x_N, x_N) \end{bmatrix} \quad (16)$$

Each cell in the gram matrix is the product of the targets and kernel of the dot product between two variables. In the top left corner, both inputs start at the first index. Moving right and down, both increase the indices for one of the inputs.

A key example of the use of SVMs with functional neuroimaging data is the work of LaConte et al. [26]. In this paper, LaConte *et al.* collected data from sixteen right-handed volunteers performing two repeated runs of a static force task, alternating between six rest and five force periods (45 s/period; (200, 400, 600, 800, 1000) g randomized target force with thumb and forefinger). fMRI data was collected using a Siemens 1.5T scanner. An SVM was compared to a multivariate analysis technique called canonical variate analysis (CVA), mainly in terms of classification accuracy and model interpretation. Hyperparameters of the models were tuned using a cross-validation set, which also acted as the estimate of classification accuracy. This classification accuracy estimation method results in optimistic predictions, as noted in the paper. A more optimal method is splitting the data three ways where classification accuracy is computed on held out data. 2880 SVM models were generated, and 1600 CVA models were generated. This is not an ideal method for comparing classifiers as the chances of type 1 error increases when subject count stays the same, but model count increases. Model complexity was controlled by using the C value, and kernel for the SVM, while for the CVA (as with ICA), the complexity was controlled by the number of components.

It was found that the linear kernel was the best performing kernel for the SVM. Another finding was that the C value would vary when using different subjects in the validation set, while the CVA model stayed mostly consistent. Another finding was that the prediction accuracy would only go down at low C values, though this is likely an effect of using the validation set as the test set. The SVM was also found to be less sensitive to different pre-processing techniques than the CVA.

One of the better model interpretation techniques, described in the work by LaConte et al., looked at support vectors to see which features were ambiguous between classes, and which were not. Nonlinear kernels do not suit model interpretation, as weights in a different dimension do not preserve their meaning, making interpretation ambiguous. The paper suggests further work is needed to properly evaluate the importance of features, but it does provide a helpful quote by Mjolsness and DeCoste[36]:

“Discriminative models make no attempt to explicitly capture the true underlying physics of the phenomena. Nevertheless, as many recent successful applications of methods such as SVMs have shown, such classifiers can provide strong insights into the nature of the phenomena, including such aspects as which input dimensions are most useful, which examples are most likely to be outliers, and what new observations might be most worthwhile to gather.”

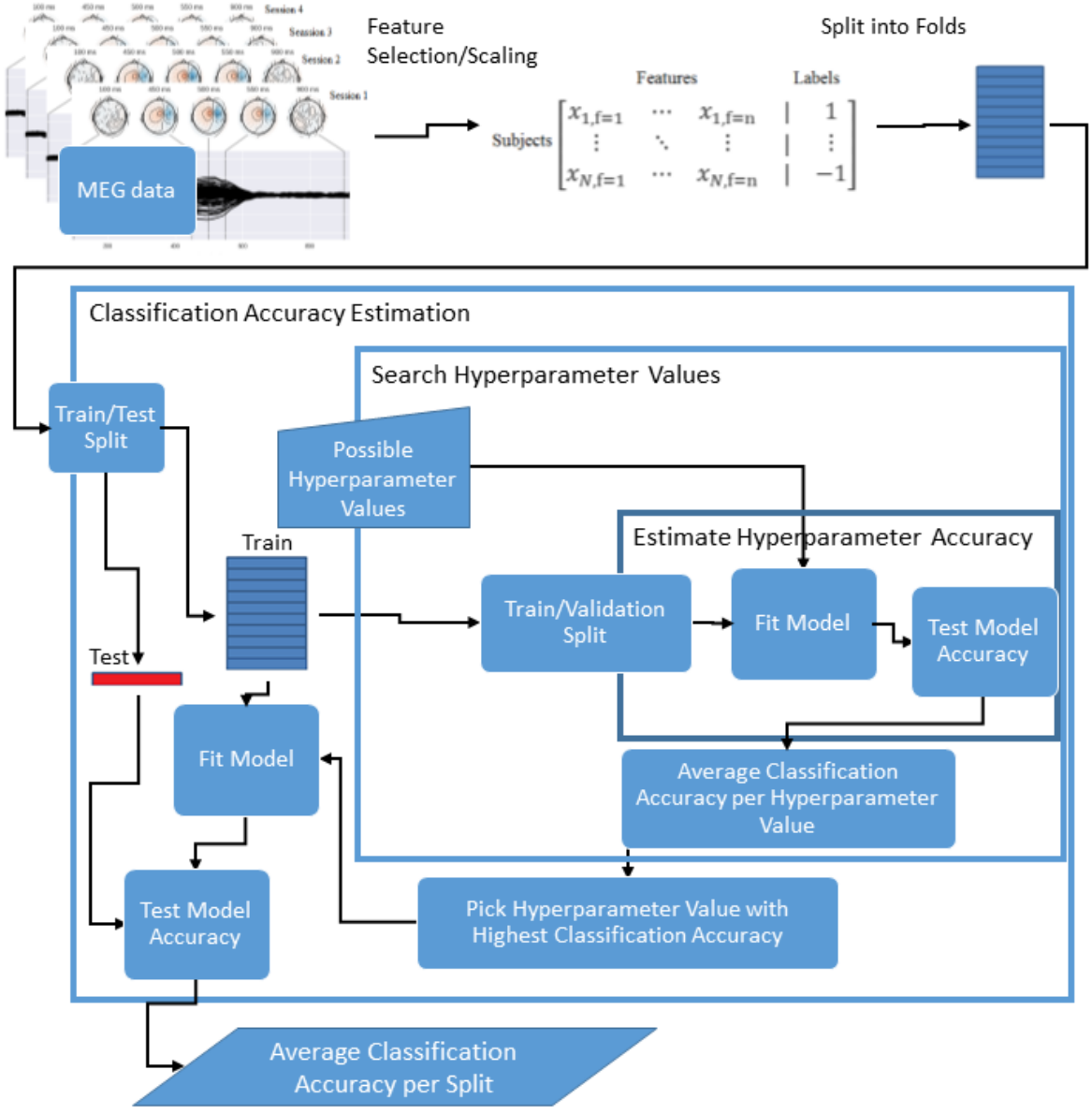

Figure 2 - Outline of Workflow for Machine Learning Classification

Figure 2 shows the workflow for classifying MEG data. Starting at the top left of the figure is the collected MEG data, which initially undergoes feature selection and scaling. Feature selection and scaling transforms the data into a set of vectors composed of features and labels. These vectors are then randomly split into folds. The folds are fed into a loop where the folds are randomly put into a train and test set. The train data is then fed into a loop that searches through various hyperparameter values.

Further, the current hyperparameter value being tested is fed into a loop along with a random split of the train set into a train/validation set. This innermost loop is further explained in Figure 3. From this, the tuned hyperparameters are obtained by picking the hyperparameters giving the largest classification accuracy. Tuned hyperparameters and train/test folds are used to fit the machine learning model. Classification accuracy on the test data is computed, and the average of classification accuracies is output by the outermost loop.

1.2.5 Logistic Regression

Logistic regression (LR) is similar to support vector machines, and serves as a good comparison for SVMs to more standard classification approaches. Instead of separating data with a hyperplane like an SVM, LR tries to fit the data to a continuous function [21]

$$F(x) = \frac{1}{1+e^{wx+b}} \qquad (17)$$

This function quickly switches between 0 and 1, representing the probability of an example belonging to class 1. The "hyperplane" in LR would represent the point where probability is 0.5. A minimization problem can be formulated by taking the log likelihood giving

$$\min_{w,b} \sum_{s=1}^{N} log\left(e^{-y_s\left(x_s^T w+b\right)} + 1\right) \qquad (18)$$

1.2.5.1 Logistic Regression vs SVM

Since LR fits the data to a continuous function, all data points are taken into account, unlike an SVM which only looks at the support vectors. An SVM does not naturally give the probabilities as a result. For an SVM, information about the distributions is missing, as only the support vector are used during test time, meaning information about each classes distribution is missing. Solutions to this problem exist like Platt scaling. In Figure 1, where the SVM only takes into account the circled support

vectors, LR would take every point into account. Taking all the points into account makes LR more computationally expensive than using an SVM. An SVM and LR can both use the kernel trick described above so that non-linear models can be developed by both methods. The motivations for the SVM and LR also differ. The SVM is more geometrically motivated, as it tries to find a hyperplane to separate the data, while logistic regression tries to fit the data to a function. In practice, LR and linear SVMs tend to have similar performance [37].

1.2.6 Feature Selection and Scaling: Importance of Pre-Processing

Processing the raw data prior to training involves feature selection, normalization, and any modifications to the raw data. Feature selection is the choice of what data to feed into the machine learning algorithm, and the transformation of the data into a form that can be fed into the machine learning algorithm. It is important to perform feature selection because redundant features or useless features would just make the training process take a longer time to find a good solution, require more data, or even be biased. Removing interdependence and redundancy is helpful because fewer features means faster execution of the algorithm and training of the pattern based classification model. Feature selection is not always performed; various machine algorithms can still give good classification accuracy for functional neuroimaging data simply by using every voxel [26]. However, feature selection has been shown to reduce prediction error and improve interpretability [38, 39].

A popular method for feature selection is recursive feature elimination, where different sets of features are used to train a machine learning algorithm and the subset with the lowest error is used as the official set of features for the training. It is difficult to know beforehand how many features need to be selected before moving on, but there are some principles one can follow as a general guide to the process. If previous knowledge exists about the system undergoing pattern recognition, then certain features can be selected as being useful *a priori*. If the features have interdependence, then one can try to modify them so that they are independent and normalize the features [40].

Normalization/scaling can be applied to each feature independently so that no variable is considered more important simply because its magnitude is larger. Normalizing each feature independently is commonly done when different features have different scales. Normalization, along with realignment and filtering, are important pre-processing steps [17]. Realignment centers the data around 0 so that no variable is considered more important simply because of its magnitude. Low-pass filters remove frequencies above a certain value. This is useful when collecting real world data, as it can remove unwanted signals like power line noise (50/60 Hz). High pass filtering removes frequencies below a certain value. This can remove drift in the signal.

The input data can also be transformed into an alternate representation. For example, multivariate analysis can be used to separate artifacts like head movement from brain activity by finding their independent components and removing them from the data [41]. Instead of using multivariate analysis, another machine learning algorithm called auto-associative networks can be also be used to reduce the dimensionality of the data [42], but no such usage was found in literature for functional neuroimaging.

1.2.7 Cross Validation, Parameter Tuning, and Generalizability

Any model can be fit to a set of data with perfect accuracy, given the model is sufficiently complex to describe the patterns in the data. Perfect fits are not typically the desired result as it can mean fitting to noise present in the data. Fit models typically want to be applied to examples outside of the data that the model was trained on. The ability to classify data outside of the experimental data used to train the algorithm is the desired outcome as it allows for the algorithm to be applied to new data. This is known as generalizability. One of the keys to generalizing properly is using a cross-validation data set [43, 44]. Cross-validation involves splitting the data into separate sets composed of training, cross-validation, and test sets as shown in Figure 3 (sometimes the last two are treated as one if there is a small amount of data available).

Cross-validation is used in order to estimate classification accuracy and select hyperparameters [21, 45]. In cross-validation, the entire dataset is split into smaller

sets. Subsets are created by randomly drawing from the entire dataset without replacement, and making sure there is an equal number of each class in each subset (stratified). A popular cross-validation scheme is K-Folds, where the data is split into different folds. For example, consider N sets of inputs. For K folds there would be K different sets of input vectors, with each iteration containing K-1 training sets, and one validation set. If there were three input vectors and leave-one-out cross validation (LOOCV) were used, then the three different trains would be: Train= [1,2] CV = [3], Train= [2,3] CV = [1], and Train= [3,1] CV = [2].

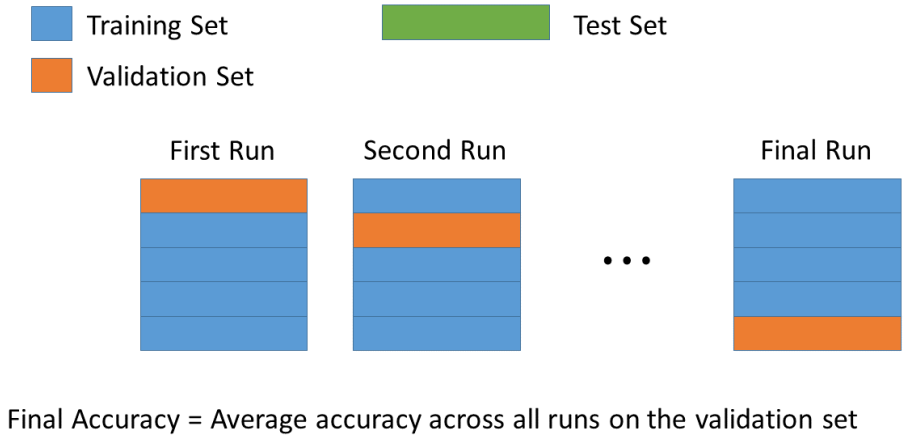

Figure 3 – Cross Validation Outline

Figure showing how cross-validation works with 11 sets. Initially, the data is randomly ordered and split into 11 sets of data. One of these sets is set aside at the beginning and only used again at the end to test classification accuracy. The remaining 10 sets are then used for picking the best hyperparameters by training on all the sets except for one, and using the excluded set to estimate performance. This repeats multiple times to find the best hyperparameter value. In the end, all of the 10 sets are used for training with the optimal hyperparameters, and the test set is used to estimate accuracy.

Classifiers have different hyperparameters to tune during cross-validation before classification accuracy is estimated. All the classifiers in this study tuned their hyperparameters using the cross-validation scheme outlined in Figure 3. There are two main phases in this scheme, tuning and then error estimation. One set is put aside at the very beginning to use for error estimation later on. The rest of the data is then used for tuning of the hyperparameters. This is done by assigning some value for the hyperparameter in question, and then training on all the sets except one. Performance of the model using that specific value for the hyperparameter is evaluated by testing on the held out set. This is repeated until every set is held out to obtain an average accuracy for the model with that hyperparameter. Then, another value for the hyperparameter is picked and the process repeats. Once the search is finished, the optimal hyperparameter is picked and tuning is complete. The tuned hyperparameters are then used to estimate the accuracy of the classification model on unseen data.

1.2.7.1 Trade-Offs

Tuning hyperparameters involves a trade off between bias and variance. Bias is the error which measures how close the model's predictions are to the true value across multiple splits (in the context of cross-validation). The variance is a measure of error the model has for a single split. When cross-validation is performed, the algorithm determines the optimal trade off between the two. It does this by using the held out data set, called the validation set (as seen in Figure 3 and Figure 4), to estimate bias error. A model fit that gives high bias but low variance means the model is only good at predicting for that single data set. Cross-validation increases the variance in order to decrease the bias. Achieving high variance, ignoring bias, is actually trivial given the model is sufficiently complex. With an SVM, a large C value generally decreases variance at the cost of bias. A small C value increases variance, but decreases bias.

1.2.8 Diagnosis and Prognosis

SVMs have started to see application in the diagnosis and prognosis of neurological and psychological disorders [46]. Merriam-Webster defines prognosis as "A doctor's opinion about how someone will recover from an illness or injury", and

diagnosis as “The act of identifying a disease, illness, or problem by examining someone or something” [47]. In the context of functional neuroimaging data, diagnosis would be the development of a classifier able to tell whether there is a certain brain disease or injury, and prognosis would be looking at a functional image to predict the disease trajectory. Using multivariate pattern analysis, one can significantly improve sensitivity in early detection or diagnosis, in particular when the changes are very subtle [45].

According to Orrù et al., pattern based classification studies of neurological and psychiatric disorders can be split into three categories [46], “(i) studies which examine the diagnostic value of neuroimaging data by comparing patients and healthy controls (HCs); (ii) studies which examine the potential of neuroimaging data for predicting the onset of a disease by comparing the brain scans (acquired at baseline) of individuals with prodromal symptoms who subsequently did and did not become ill, and (iii) studies which examine the prognostic value of imaging data by comparing the brain scans obtained from patients prior to treatment onset who subsequently did and did not respond.”.

First are studies which examine the diagnostic value of neuroimaging data by comparing patients and controls. SVMs have been able to distinguish not only between HCs and patients with dementia, but also between different types of dementia: frontotemporal lobar degeneration and Alzheimer’s disease (AD) [48]. In this study by Dukart et al., different biomarkers were investigated to try and improve detection and differentiation of different types of dementia. 21 subjects with AD, 14 with frontotemporal lobar degeneration (FTLD), and 13 HCs were scanned using positron emission tomography (FGD-PET) scanning and magnetic resonance imaging (MRI), together with clinical and behavioral assessments as benchmarks. Support vector machines with a linear kernel were used as the classification tool. The researchers obtained a differentiation accuracy of 92% between the groups by using the structural MRI and FGD-PET information together. 94% accuracy was obtained when differentiating AD and FTLD patients.

Second are studies which examine the potential of neuroimaging data for predicting the onset of a disease by comparing brain scans of individuals with initial symptoms who subsequently do, or do not become ill. SVMs have been applied to the detection of mild cognitive impairment (MCI), which is thought to be a transition between normal ageing and AD associated dementia. Early detection of MCI can help provide treatments that will help alleviate deficits in cognition coming from the further progression of AD [49]. A study by Nho K et al. predicted conversion from MCI to AD associated dementia given data from follow-up periods of 1, 2, and 3 years using structural MRI with 72.3% prediction accuracy [23]. They used an SVM to classify between subjects who had AD and HCs. They also attempted to predict conversion between MCI and AD. They were able to achieve 90.5% classification accuracy in differentiating the groups and 72.3% in predicting the conversion. Since model interpretation was not a concern in this study a radial basis function kernel was used.

Third are studies which examine the prognostic value of imaging data by comparing the brain scans obtained from patients prior to treatment onset, who subsequently did or did not respond to the treatment. For predicting the recovery of patients following treatment, the majority of studies have been focused on major depression. A study by Gong et al. found that SVMs were able to distinguish between patients that did and did not respond to medication using structural MRI in a longitudinal study with an accuracy of 69.57% (based on gray matter) and 65.22% (based on white matter) [50]. This study evaluated 61 drug-naïve adults suffering depression and 42 HCs. Subjects were given medication in the drug-naïve group and evaluated for depression at a future date. The SVM's task was to predict which subjects would and would not respond to the medication. In this study a non-linear kernel was used to avoid the risk of over-fitting the data, as suggested by the authors.

Proper function of the brain is an important component to living normally. Over time brain activity can change based on disease, injury, or recovery. Measurements that are indicative of different prognoses can be found by collecting data from subjects longitudinally. Current prognosis of brain injury, such as concussion, primarily uses

measurements of symptoms to make clinical decisions. Longitudinal functional neuroimaging data may help inform these decisions, if models that are predictive of diagnosis or prognosis can be generated.

1.3 Longitudinal Functional Neuroimaging Data

1.3.1 What is Longitudinal Functional Neuroimaging Data

Typical functional neuroimaging studies, such as those described above, are constrained to a single session with each participant, meaning the researcher only has cross-sectional information about brain activity over a group (or groups) of individuals. This is useful for determining which sections of the brain are active for specific tasks, but does not allow for trend level analysis. Trend level analysis gives information about neural development: be that positive, neutral, or negative trends. Longitudinal functional imaging takes multiple scans of the brain over time to give more information about a subject's trajectory. As an example, attention related brain activity could be probed for processing efficiency using reaction time measures with multiple sessions over some period of time [51]. A trend-level analysis can be completed when multiple cross-sections of the attention related activity over time are recorded. This is possible because a longitudinal approach increases sensitivity to the changes in the brain by reducing between-subject variability and accounting for the temporal order of scans [1, 2]. Increased sensitivity means the trajectory of outcome measures can be more accurately assessed based on within-subjects' changes. Assessing recovery trajectories would allow for better treatments for those suffering from neurological problems by being able to predict if someone is recovering or not. Longitudinal functional neuroimaging data also helps us understand the brain by being able to see how longterm changes in the brain can affect behaviour.

1.3.2 Example: Prognosis for mTBI

Current prognosis of brain injury, such as concussion, primarily uses measurements of symptoms to make clinical decisions. Over time, brain activity can change based on disease, injury, or recovery. Measurements that are indicative of different prognoses can be found by collecting data from subjects longitudinally.

Those who suffer mild traumatic brain injuries (mTBIs), commonly known as concussion, can have changes in brain activity that lead to a diminished quality of life. This is made worse for those who are members of vulnerable groups like children. Deficits in attention are one example of negative changes arising from mTBIs for students who recently suffered mTBIs [52-54]. Neurologists assess post concussion symptom severity and recovery rate (or "trajectory") using standardized questionnaires that may elicit symptoms not reported freely, such as the Concussion Symptom Inventory (CSI) [55]. The CSI is a self-report questionnaire that rates twelve symptoms on a scale of 0 (absent) to 6 (severe). Symptoms assessed relate to sensation (i.e., headache, sensitivity), vision (i.e., blurring), and cognition (i.e., memory, attention). Other examples of common questionnaires are the Sports Concussion Assessment Tool 3 (SCAT3) and the Immediate Postconcussion Assessment and Cognitive Testing (ImPACT) test battery [56]. The questionnaire-style test includes a symptom inventory, cognitive assessment of attention and memory, and physical evaluation that includes the Glasgow Coma Scale and a balance assessment [57, 58]. The ImPACT test battery includes demographic data, neuropsychological tests, and the Post-Concussion Symptom Scale [59]. The battery measures deficits in verbal memory, visual memory, processing speed, and reaction time, and has been shown to have high sensitivity and specificity for detecting effects relating to sports-related mTBI [60]. Towards prognosis in a cohort of male football players, a combination of scores from the ImPACT battery collected in the early stages following mTBI achieved 80% sensitivity in classifying short or long recovery times with a cut-off of 14 days [61]. However, such classification algorithms require validation on an additional follow-up cohort, which has not yet been reported.

Components of the SCAT3 and ImPACT administered by a second party are problematic, in that they are confounded by inter-rater variability. The self-report sections are confounded by the subjective nature of the patient's rating, which can deemphasize cognitive and emotional reactivity symptoms [62]. In addition, patients with a vested interest (i.e., athletes hoping to return to sport) may underrate symptoms

in order to create the perception of recovery. Further, compensatory changes within the brain can mask the clinical manifestation of damage to the brain, which generates a significant confound to behavioural measures [63]. An objective measure of the recovery of brain function following mTBI provided by functional neuroimaging, which looks directly at brain activity, would be an attractive alternative, and may be developed using pattern classification algorithms.

1.3.3 Challenges for Analysis

Taking the temporal aspect of longitudinal data into account is important because it is information not available with typical cross-sectional data. However, longitudinal data analysis has several difficulties that are not present when looking at cross-sectional data. Since data is being collected over a period of time, subjects can miss some scans for unforeseeable reasons, resulting in unbalanced data, which some SVMs can handle [64]. The amount of time between scans might differ between subjects as well, giving non-uniform data. Certain statistical aspects need to be taken into account as well: responses are correlated (not independent), variance of repeated measures often change and increase steadily with time (heteroscedasticity), and measures closer in time are likely to be more highly correlated than measurement pairs that are further separated in time [1-3]. The variance between subjects is not consistent over time. The variance between two subjects can increase over time if the trajectories diverge, or it may decrease if the trajectories start to converge.

The factors affecting the trajectory can also not be of interest. A subject's performance of a task can change over time from getting used to the paradigm, or even decline in some cases. A good model should account for increasing variance within subjects over time, and dissociate performance-related changes from changes relating to the covariates of interest. Predictive modelling can be utilized to assess the probable trajectory of additional participants, based on the acquired data [4]. However, predictive modelling requires the use of additional tools to learn the patterns that exist in longitudinal data that are most predictive of trajectory.

1.4 Longitudinal Support Vector Machine

The longitudinal support vector machine (LSVM) is an adaption of the SVM that specifically takes the temporal component of longitudinal data into account [4, 25].

1.4.1 Derivation

The LSVM starts with the SVM model of classification, but adds some modifications. Firstly, the way of interpreting an input is changed. Instead of thinking of the feature space as being made from independent features like in equation (3) it is composed of measurements that are repeated across sessions

$$x_s = \begin{pmatrix} x_{1,t=1} & \cdots & x_{n,t=1} \\ \vdots & \ddots & \vdots \\ x_{1,t=T} & \cdots & x_{n,t=T} \end{pmatrix} \quad (19)$$

Here t represents the sessions number at which the feature was measured. A new variable β is added that combines the features across sessions defining linear trends of change. Thus, the input for the LSVM is re-formulated as below:

$$x_s = x_{s,1} + \beta_2 x_{s,2} + \beta_3 x_{s,3} + \cdots + \beta_T x_{s,T} \quad (20)$$

The hyperplane is still defined properly by (5), but the length of w and x_s are different from the standard SVM. Note that the first β is a 1. In order to learn the β values at the same time as α a custom gram matrix is defined for equation (15)

$$G_m = \begin{bmatrix} \tilde{X}_{t=1}^T \tilde{X}_{t=1} & \cdots & \tilde{X}_{t=1}^T \tilde{X}_{t=T} \\ \vdots & \ddots & \vdots \\ \tilde{X}_{t=T}^T \tilde{X}_{t=1} & \cdots & \tilde{X}_{t=T}^T \tilde{X}_{t=T} \end{bmatrix} \quad (21)$$

where

$$\tilde{X}_{t=k} = [y_1 x_{1,t=k}, y_2 x_{2,t=k}, \ldots, y_N x_{n,t=k}] \quad (22)$$

giving

$$\min_{\alpha} \frac{1}{2} \alpha^T G \alpha - \alpha$$
$$s.t. C \geq \alpha_s \geq 0, \sum_t^T \sum_s^N \alpha_{(s+tN)} y_s = 0, for\ s \in 1 \ldots N\ and\ t \in 1 \ldots T \quad (23)$$

There are several new things going on in equation (23). One is the gram matrix which is composed of the dot product between features at the same time point, instead of all the features at once like in equation (15). The other is that the C constraint only applies to the first α of every session as that is the only alpha value not combined with a β term as shown in the equation below when expanding the Lagrangian parameterized form of the weights

$$w = \sum_{s=1}^{n} y_s \alpha_s \left(x_{s,1} + \beta_2 x_{s,2} \ldots + \beta_T x_{s,T}\right) \quad (24)$$

A more convenient formulation of the problem is recommended by Chen et al using an iterative procedure that splits the gram matrix into four parts when using two data points

$$\begin{bmatrix} \alpha \\ \beta\alpha \end{bmatrix}^T \begin{bmatrix} G_m^{0,0} & G_m^{0,T} \\ G_m^{T,0} & G_m^{T,T} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta\alpha \end{bmatrix} - 1'\alpha \quad (25)$$

$G_M^{0,0}$and $G_M^{T,T}$represent only the first and last sessions respectively, while the submatrices fill in the rest of the values. In the iterative procedure, α is found first by assuming β is known, followed by the reverse. For the following steps the bias $-1'\alpha$ will be ignored. Since the βs are assumed to be known in the first step the objective simplifies to

$$\begin{aligned} &\alpha^T (G_m^{0,0} + \beta G_m^{T,0} + \beta G_m^{0,T} + \beta^2 G_m^{T,T})\alpha \\ &= \alpha^T (G')\alpha \end{aligned} \quad (26)$$

Which can be solved using the same techniques as equation (15). When the alphas are assumed to be known the problem simplifies to

$$G_{m_\alpha}^{0,0} + \beta G_{m_\alpha}^{T,0} + \beta G_{m_\alpha}^{0,T} + \beta^2 G_{m_\alpha}^{T,T} \quad (27)$$

Taking the derivative and setting it equal to zero gives

$$\beta = -\frac{G_{m_\alpha}^{T,0} + G_{m_\alpha}^{0,T}}{G_{m_\alpha}^{T,T}} \quad (28)$$

Which gives the β values.

The LSVM handles several of the issues raised by longitudinal functional neuroimaging data. First is that the responses measured across sessions are related, instead of being independent. Secondly is that the model, like the SVM, should be robust against noise – like that introduced by heteroscedasticity. This model does not account for missing data, or that measurements closer in time are more related than measurements that are further away in time.

1.4.2 Differences from the SVM and LR

There are two main differences between the LSVM and SVM/LR. First is that the temporal component is explicitly taken into account by having a β term define the change in features between sessions in the longitudinal data as shown in equation (20). A measurement that was repeated over several sessions is reduced to one feature by combining the longitudinal values using the learned β terms. The other difference is that, unlike an SVM, the LSVM takes all of an examples measurements or none of them. This is shown in equation (24) where a single α value is shared across all sessions for a subject.

The LSVM is expected to provide higher classification accuracy on data that fits its assumptions, because of how it reduces the number of features by a factor equal to the number of sessions, as compared to SVM and LR. The assumptions that need to hold true for this are that the important features for classification share a similar (ideally the same) magnitude of change across sessions. Higher classification accuracy should also lead to more stable weights that are being discovered, meaning their interpretation should be more consistent.

1.5 Objectives and Hypotheses

Unique to longitudinal data compared to cross-sectional functional neuroimaging data is the temporal ordering of the data. Explicitly utilizing this temporal component should improve classification accuracy by removing the amount of information that needs to be learned during training. The objective of this thesis is to implement the LSVM, and to test its performance with longitudinal data relative to an SVM and LR. To do this two studies were done. Both studies involved the simulation of

data that had two sessions, and two classes. In the first study, purely simulated data was used similar to what was done by Chen et al. in the paper describing the LSVM [4]. I hypothesize that the longitudinal support vector machine, by explicitly including the temporal component of longitudinal data, will have a higher classification accuracy and produce more interpretable feature weights, as compared to classification models that do not explicitly account for the temporal component. This hypothesis is based on the preliminary results of Chen et al., and the assumption that higher classification accuracies will lead to feature weights that more strongly reflect the simulated input data.

In the second study, a simulated signal was inserted in resting state MEG data to generate simulated longitudinal functional neuroimaging data. For this study, I also hypothesize that the longitudinal support vector machine, by explicitly including the temporal component of longitudinal data, will have a higher classification accuracy and produce more interpretable feature weights, as compared to classification models that do not explicitly account for the temporal component. This is also based on the work of Chen et al. as they also showed improved performance with neuroimaging data in terms of classification accuracy. Importantly, this second study will allow us to determine the magnitudes of temporal trends in MEG data for which the LSVM outperforms the SVM and LR.

Chapter 2 Methods

This thesis contains two main studies. Each study involved trying to correctly classify a subject into either a “stable” or “manipulated” class. In the case of functional neuroimaging data, the class may relate to disease state (“healthy” vs. “patient”) or prognosis (“recovering” vs. “declining”). Each class example is composed of features, where each feature in MEG would be the sensor reading at a given latency. In the first study features for each subject were purely simulated. Features for both classes were drawn from normal distributions of varying means and standard deviations. The first study replicated, and expanded upon, the work from the original LSVM paper by Chen and Dubois [4]. Modifications on the work done by Chen and Dubois resulted in a follow-up simulation that looked at the question, “could the LSVM find interpretable weights for more complex datasets?”. The second study used real resting state MEG data to add real noise levels to a simulated MEG signal.

2.1 Study One: Pure Simulation

2.1.1 Simulation of Longitudinal Data

Data was generated using the procedure described by Chen and DuBois [4]. Two hundred subjects of data were generated, simulating a situation wherein half of the subjects were part of a stable class, and the other half were part of the manipulated class (i.e., different distribution between session one and two). Two sessions were simulated for each subject, with 100 features per session. Each subject’s features were drawn from a normal distribution with varying mean and standard deviation. The point of the first simulation was to emulate the work of Chen and DuBois to verify if the LSVM algorithm outperforms the SVM in terms of classification accuracy. Going beyond the work described in [4], I further modified the first simulation to see how the different algorithms (including LR) would behave in cases that were more challenging to classify, and with more complicated patterns in the features. Finally, I performed a heteroscedasticity study which tested how the algorithms performed when there was increased variation at the second session, which is very different from the initial simulation by Chen and DuBois.

2.1.1.1 Simulation One

The first simulation in Study One implemented the simulation used by Chen and Dubois in their paper. The stable class had no change in the mean used to generate the features between sessions one and two. The manipulated class had an increase in the mean between session one and two. Following the work by Chen and Dubois the data at the first session was generated using

$$x_{t=1}(\overline{m}_c, \sigma^2) = N(\overline{m}_c, \sigma^2) \qquad (29)$$

In this equation $\overline{m}_c$ and σ^2 are the mean and variation of the normal distribution respectively. Half the subjects of each class at each session had their $\overline{m}_c$ value shifted up by a magnitude of 1. The value of $\overline{m}_c$ is controlled by the index *c* which defines the class from the class example is drawn from. The standard deviation was kept at a constant value of 1.

Session 2 values were drawn from

$$x_{t=2}(\overline{m}_c, \sigma^2, x_{t=1}) = x_{t=1} + N(\overline{m}_c, \sigma^2) \qquad (30)$$

Here the result of equation(29) was added to a random value drawn from a normal distribution. If the subject was part of the stable class (*c* = 0) then the mean was 0, if the subject was part of the manipulated class (*c* = 1) the mean was 1. This meant that only the manipulated class would have added signal that was not due to noise at the second session. The standard deviation was 0.1 for the $x_{t=1}$ term in equation (30).

To test the robustness of the classifiers a modification was performed to equation (30) so that the magnitude of change in the mean at the second session for the manipulated class could be controlled

$$x_{t=2}(\overline{m}_c, \phi^2, \tau, x_{t=1}) = x_{t=1} + N(\tau\overline{m}_c, \phi^2) \qquad (31)$$

A new variable, τ, was introduced to define the magnitude of change for the second session, and was varied between 0.0001 and 100. By controlling the magnitude of the change the SNR is also controlled as it becomes harder to differentiate the two

classes at small values of τ. Once the data was generated, the feature space (i.e., input matrix for machine learning) was an N x M matrix, where N is the number of examples (200, 100 of each class) and M is the number of features (100 per session times 2 sessions – 200 features overall). Additionally, a vector of class labels (-1, 1) was included to indicate the class for each example.

2.1.1.2 Simulation Two

Next a pair of diverging trends across the feature space was simulated. This was done to see how the classifiers would perform in terms of weight interpretability and classification accuracy when a more complex feature pattern was used. The dataset was more complex by having some features contain more information than others. This simulates the case in functional neuroimaging where some sensors or voxels may be more informative than others.Feature values at the first session were simulated the same way as that described by equation (30). Feature values for the second session were drawn from a modified version of equation (31)

$$x_{t=2}(\overline{m}_c, \phi^2, \tau, x_{t=1}, f) = x_{t=1} + N\left(\frac{f}{100}\tau\overline{m}_c, \phi^2\right) \qquad (32)$$

Equation (32) adds a new variable f which is the index of the feature (starting at 1). The added scaler $\frac{f}{100}$ scales the features so that the normal distribution from which the initial feature is drawn has a small mean and the last feature's normal distribution mean is equivalent to $\tau\overline{m}_c$. For subjects in the stable class the mean was -1 and for the manipulated class it was 1. These mean values cause for the two trends to diverge across features at session 2. The τ values were selected to give a range of classification accuracies between 50% and 100%. A small τ value would mean that the two classes would not diverge as much from each other as when a larger τ value was used.

2.1.2 Feature Selection and Scaling

Feature selection was not performed for any of the data sets in study one. All data was scaled by normalizing each feature to have unit variance and centering around 0. The mean was calculated using

$$\overline{f_m} = \frac{1}{n}\sum_{i=1}^{n} x_{i,f} \quad (33)$$

And the variation by

$$\overline{f_{\sigma^2}} = \frac{1}{n}\sum_{i=1}^{n}\left(x_{i,f} - \overline{f_m}\right)^2 \quad (34)$$

Each feature was then scaled by

$$f = \frac{f - \overline{f_m}}{\sqrt{\overline{f_{\sigma^2}}}} \quad (35)$$

To prevent overestimating classification accuracy, the mean and variation were calculated using only the training set during cross validation. The resulting mean and variation were then used to transform the test set. If the mean and variation were calculated using the entire dataset, then information would be "leaked" from the test set that should not have been seen by the algorithm to the training process. This information leakage makes the algorithm perform better than it should be as information about the distribution of the data that should not be available is used during training.

2.1.3 Implementation of Classifiers

2.1.3.1 Logistic Regression

LR was implemented using the scikit-learn Python library [65, 66]. The implementation used equation (17) with added terms to control overfitting

$$\min_{w,b} C\sum_{s=1}^{N} log\left(e^{-y_s\left(x_s^t w + b\right)} + 1\right) + \frac{1}{2}w^T w \quad (36)$$

This equation is L2 regularized meaning that large weight values, w, are punished by adding the dot product of w with itself. During optimization the only hyperparameter is the C term, which controls the error tolerance. Smaller values of C allow for greater error during training, and larger values the reverse.

2.1.3.2 Support Vector Machine

SVMs were implemented using the scikit-learn (version 0.18.dev0) Python library. Python release version 2.7.11 was used. The minimization problem described in equation (15) was implemented. A hyperparameter C was used to regulate error tolerance during training, although the measurement of error was different compared to LR. Only a linear kernel was used with the SVM.

2.1.3.3 Longitudinal Support Vector Machine

The LSVM pipeline was implemented using the iterative procedure outlined in equations (25)->(28). All code was written in Python using tools from the SciPy stack (important packages: NumPy - 1.11.0, SciPy – 0.17.0, pandas – 0.17.1, iPython – 4.2.0, Matplotlib – 1.4.3, Seaborn – 0.7.0, [67]. The primary expression being worked with is

$$\alpha^T(G_m^{0,0} + \beta G_m^{T,0} + \beta G_m^{0,T} + \beta^2 G_m^{T,T})\alpha \qquad (37)$$

For the implementation α was the first variable to be estimated, followed by β. To estimate α, a β had to be provided. β was initially estimated by taking a uniformly distributed random value between -10 and 10. With an estimated β, α was calculated using the minimization problem outlined in equation (26) by creating a single gram matrix. This optimisation problem is actually the standard SVM formulation, so the SVM solver provided by Sklearn was used. β was then solved for by using the calculated alpha and the derivative to obtain equation (27).

2.1.3.3.1 $\boldsymbol{\beta}$ Optimisation

β is unique to the LSVM classifier. This parameter, along with α, is found during optimization of the classifier. It was found during testing that β would not always converge. In order to help convergence of the β term, random starting points of β were used. If convergence within an error of 0.001 did not occur within 100 iterations then the algorithm was restarted. This helped reduce cases where the algorithm would become stuck between two values.

2.1.4 Cross-Validation

Following the cross validation scheme outline in Figure 3, the data set was randomly partitioned into a train and test set. This was done 10 times to obtain an

average accuracy. For Study 1, 80% of the data was used for training, and 20% was used for testing. With 200 subjects this results in a split of 160/40 for train/test. During the validation step the C parameter for all classifiers was tuned. Classification accuracy for each of the 10 runs was calculated by dividing the sum of correctly classified subjects by the total number of subjects in the test set. Values obtained from classification (weights, C, β, etc) were averaged across the 10 trials by using the classification models obtained after validation.

2.1.5 Model Interpretation

Models were interpreted based on the feature weights, w, that were found. For interpretation to make sense a linear kernel was used, so no kernel mapping was done for any of the classifiers. Kernel mapping refers to mapping the inputs to a higher dimension, which can help with the separation of data. The LSVM has half the features of an SVM or LR. For this reason, the SVM and LR weights are interpreted by the difference between feature weights at the second and first measurement. Feature weights were plotted against features on the x-axis. To compare different classifiers, the weight plots were compared based on how well they match the simulated data, and how large the error bars were. Compared to the LSVM, the other methods should be less robust in capturing the change from session one to session two as they are not explicitly programmed to do so. This would be seen primarily by the LSVM having smaller error bars on the weight plots, and more closely matching the input.

2.1.6 Evaluating Relative Performance of Classifiers

2.1.6.1 SNR Impact on Classification Accuracy

Classifier performance was measured by looking at classification accuracy on the test set for various SNR values. For simulations one and two, τ was used to quantify how easily separable the different trends, or classes, were as a function of SNR. If τ was small, then there was a small change between sessions in the second class (and no change between sessions in the first class). I expected that the LSVM would demonstrate higher classification accuracy than the SVM and LR across different values of SNR.

2.1.7 Heteroscedasticity

To investigate the impact of heteroscedasticity on classification accuracy for the classifiers, equation (31) was modified to

$$x_{t=2}(\overline{m}_c, \psi, x_{t=1}) = x_{t=1} + N(\overline{m}_c, \psi) \qquad (38)$$

Where ψ was a value in the set $[0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0]$. This equation makes it so that the variance in the second session is controlled by ψ. All other aspects of the simulation were the same as study one: simulation one. Since the standard deviation for session two was 0.01, a ψ value of 0.01 results in no change in the heteroscedasticity. ψ values 0.1, 1.0, 10.0, 100.0, 1000.0, and 10000.0 result in increased heteroscedasticity. Classification accuracy as a function of heteroscedasticity was investigated by running training and testing as described above for each model, with $\overline{m}_c$ set to 1.0 for the manipulated class, and 0.0 for the stable class.

2.2 Study Two: Resting State MEG Data with Simulated Trends

2.2.1 Simulation of Longitudinal MEG Data

The goal of the second study was to generate data more similar to actual MEG data. Specifically, this study simulates longitudinal changes in an evoked field due to brain activity that would be generated during a left handed motor task. The evoked field data is superimposed onto MEG sensor data recorded from humans at rest. Thus, we can test the classifiers under conditions that are closer to tasks that generate well known MEG signals. This should make the simulation more applicable to real MEG studies.

2.2.1.1 Simulating a Trend

A current dipole was created to simulate brain activity. The strength of the current dipole was simulated over 500 time points, simulating a 1000ms amplitude time course being sampled once every 2ms (100ms of baseline was also added to the beginning). The current dipole strength over time followed a Gaussian distribution that peaked at 500ms with a standard deviation of 62.5ms. The peak magnitude of the current dipole was easily changed by modifying the height of the Gaussian curve. For each class, six dipole time courses were generated, such that different pairs of time

courses could be used as session one and session two data to make a variety of temporal trends. For the stable class, the peak magnitude was 10 nAm for all six sessions, and for the manipulated class, the peak magnitude ranged from 10 to 35 nAm in increments of 5. Different pairs of sessions (e.g., one and four) were selected to investigate a range of temporal trend magnitudes.

The current dipole was forward facing, and located slightly off-centre of the brain – roughly on the motor cortex. The MEG coordinate system used was a standard Euler x, y, and z based system. The x axis went from the left to right ear, y axis from back to front of head, and z from bottom to top of head. Head shape was approximated as a sphere. The dipole was inserted using polar coordinates, 2cm from the scalp and rotated 30 degrees upwards in the x/z plane. This location was picked as the motor cortex is a commonly studied area.

2.2.1.2 Adding a Trend to Resting State Data

Processing of MEG data was performed using mne-python [68-70] in Python, and FieldTrip [71] in MATLAB. The current dipole signal was added to resting state data to simulate realistic levels of noise. Resting state data was collected during a passive task where the subject is instructed to relax and focus their attention on a cross in front of them. This minimizes head and eye movement. Twenty adult subjects were recorded using a 306-channel MEG system (Elekta Oy, Helsink, Finland; 204 planar gradiometers; 102 axial magnetometers). Nine subjects were male and eleven were female. The average age of subjects was 25.8 with a standard deviation of 4.2 years. Before scanning, seven electrodes were placed on the subject to track certain signals: eye movements were recorded with electro-oculogram (EOG) using four sensors (one superior and one inferior to the left eye, and one lateral to each eye), heart rhythms with electro-cardogram (ECG) with two sensors (one on the inside of each arm), and a ground (collarbone). Four head position indicator (HPI) coils were placed on the subject's head to track head movement: two on the forehead and one on each mastoid process. Video footage was used to see if the subject fell asleep. ECG and EOG recordings were concurrent with MEG throughout the scan. HPI coils were continuously

activated to generate alternating magnetic fields at frequencies between 293 and 321 Hz. Data was collected at either 1000Hz or 1500Hz.

Resting state MEG data was preprocessed using MaxFilter (MaxFilter, 2.2 [2016], Elekta AB, Stockholm, Sweden). Temporal signal space separation was applied to remove MEG signal from outside the brain, including components from the environment. A low pass filter of 125Hz was used and the data was down sampled to a 500Hz sampling rate. The data was further low pass filtered using a cut-off frequency of 40Hz because this is the usual frequency range over which evoked responses are observed. The subject group was then split so that every odd subject would receive the constant trend (stable class), and every even subject would receive the increasing trend (manipulated class).

The simulated trend was added to the resting state by computing the forward solution with FieldTrip [71], based on equation (1). These MEG sensor readings for the simulated current dipole were then summed with the resting state data at 100 randomly selected segments without replacement or overlap. Each subject had two scans of resting data. Both resting state scans for the subject were used to simulate MEG data; half of the simulated data was generated by superimposing the simulated signal on the first resting scan, and the other half of the simulated data was generated using the second resting scan. These segments simulated “trials” of neuronal activity occurring within ongoing brain activity. “Event markers” were added to the MEG data to indicate the data sample at which each trial began. Each trial contained 100ms of data prior to the onset of the current dipole activity (to act as a baseline measure), and 1000ms following onset to capture the entire simulated activity. Once the resting state sensor data was added to the current dipole sensor data, the combined data was averaged across trials to attenuate uncorrelated signals (a common practice to accentuate MEG signals). This process resulted in the MEG evoked field data for each subject.

Once the data was generated, the feature space (i.e., input matrix for machine learning) was an N x M matrix, where N is the number of examples (20, 10 of each class) and M is the number of features (102 sensors x 550 timepoints x 2 sessions = 112200 features). Additionally, a class vector was included with a label (-1, 1) for each example to indicate the class. There are 102 sensors because only the magnetometers were used. 550 times points were available as 500 samples (1000ms) were used for the signal, and 50 samples (100ms) for baseline. This gave data with very large feature size, and small sample size.

2.2.2 Feature Selection and Scaling

2.2.2.1 Feature Selection

Two approaches were attempted for feature selection. The first approach used all the sensor data. In the second approach, PCA decomposition was used to reduce the dimensionality of the feature space. For both approaches features were scaled the same way as study one.PCA

PCA was implemented using the scikit-learn library. PCA decomposes a multivariate data set into orthogonal components that can be ranked by how much variance in the data set they explain. Specifically, for MEG evoked field data it would find spatial patterns over time that explain the maximum amount of variance. If the MEG evoked field data had a single large signal in one part of the brain, a component that mimicked this signal would explain most of the variance in the data. To make sure that no classification information was leaked the PCA was only used on the training data, and the resulting PCA weights were used to transform the train and test sets, similar to how normalization was done in study one. PCA was done on the final session of each example in the training set, and then applied to the initial session. This was done as applying PCA across all features, disregarding that they are from different sessions, would cause the PCA to combine features across measurements. This would then invalidate assumptions of the LSVM, as it combines related features across measurement sessions.

PCA changes input size from 20 x 112200 to 20 x 2F, where 2F is the dimensionalilty after the PCA transformation. The number of features is 2F as PCA is only applied to the second session, which gives F features, and is then applied to the first session giving a second F amount of features. This results in a lower number of features, where the remaining features explain x amount of variance in the data. x being a percentage of variance defined by the user, either 0.80, 0.90, or 0.95.

2.2.2.2 Scaling Strategies

Different scaling strategies had to be used for different classifiers. For SVM and LR, optimal performance was reached by using the same scaling strategy as in Study One. For the LSVM, the β term acted as a scaling factor, so no scaling provided better performance than scaling.

2.2.3 Cross-Validation

The same cross-validation scheme used for Study One was used for Study Two. Since there were 20 subjects instead of 200, the 80/20 split resulted in 16 subjects used for training and 4 subjects used for testing.

2.2.4 Evaluating Relative Performance of Classifiers

Classifiers were compared by their test set classification accuracy at different SNRs. In this case, the SNR was calculated by dividing the signal strength by the standard deviation of the baseline for the MEG data in the second session. First, the MEG sensor that showed the strongest activation at 500 ms (i.e., peak latency) was isolated as the peak sensor to calculate SNR. The numerator was calculated by taking the average of 5 points before and after the peak sensor values at 500ms. The denominator was calculated by taking the standard deviation of the first 50 points (prior to current dipole activity) to compute the standard deviation.

$$SNR = \frac{signal_{avg}}{baseline_{std}} \quad (39)$$

2.2.5 Model Interpretation

2.2.5.1 Interpretation in the Context of MEG

As with study one, the weights were used for model interpretation. Additional formatting steps were performed to interpret the data in the context of MEG data.

Weights after classification are in a $sensors * time$ matrix. To obtain the weights in an understandable format the weights were mapped to a butterfly plot that shows the magnitude of each sensor over 1000ms, and topographical maps that show spatial activity at specific times. Important visual factors for interpretation were the location of the magnetic fields, and how they changed over time.

A more objective measure of the model interpretability was a correlation computed between the feature weights and the actual data. So that the SVM and LR could be compared to the LSVM, the SVM/LR feature weights at the first session were subtracted from the second session. The feature weights were then normalized the same way as that outlined in equation (35). The simulation session with the highest SNR was used to correlate with the feature weights. This correlation measure was not used on the Study One data, as the true signal in study one had a variance of 0.

Chapter 3 Results

3.1 Study One

3.1.1 Simulation One

The first simulation served as a way of replicating the work by Chen and DuBois, while expanding on it by adding a check for robustness of classifiers by comparing classifiers across various magnitudes of temporal trend.

3.1.1.1 Simulation

Figure 4 and Figure 5 show how the simulated data looks for a τ of 1. τ is the measure of separability of the data. Specifically, it defines the scale of change between the first and second session for the manipulated group, and the size of the difference between half the subjects in each class. With larger values of τ the size of the jump between the first and second session increases. Figure 4 shows the averaged feature values at a τ of 1, for the stable (subjects 1-100) and manipulated (subjects 101-200) classes. The change in feature values can be seen between the first and second session of the green class. The variability within the class can be seen between every consecutive set of 50 subjects. Figure 5 shows the same data as Figure 4, but this time features are on the x-axis and they are averaged across subjects. The change in feature values in session two for the manipulated class can be seen in the bottom part of the figure. The mean feature values for the stable class are around 0.5 due to the dual distribution within class (i.e., half the subjects in each class are offset by 1) and the mean feature values for the manipulated class are around 1.5 for the same reason.

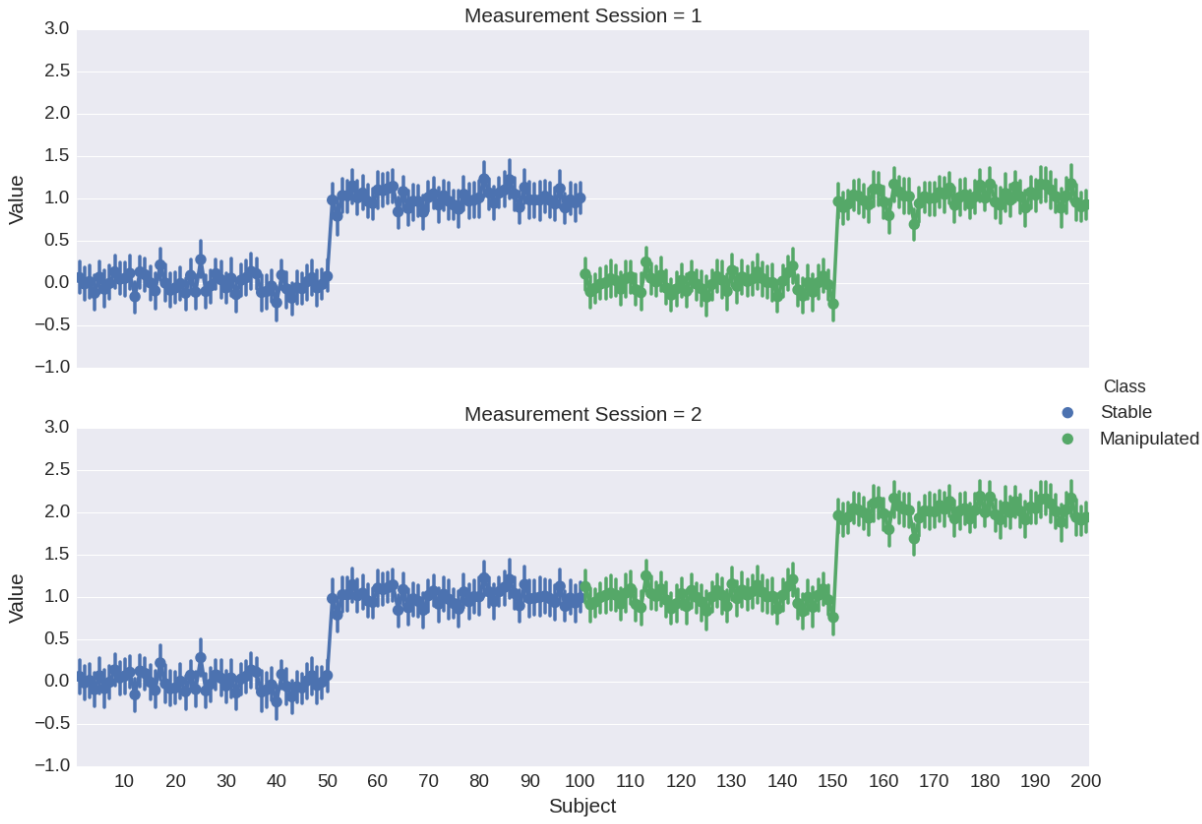

Figure 4 – Chen and DuBois Simulation with Averaged Features

Plot of each subjects averaged features. The y-axis represents the arbitrary value of the features, with the x-axis representing subjects. The top plot is from the first measurement session, and the bottom plot is from the second session. Blue represents the stable class where this is no change from the first session to the second. Green represents the manipulated class where there is an increase in the feature values. Error bars represent 95% confidence intervals.

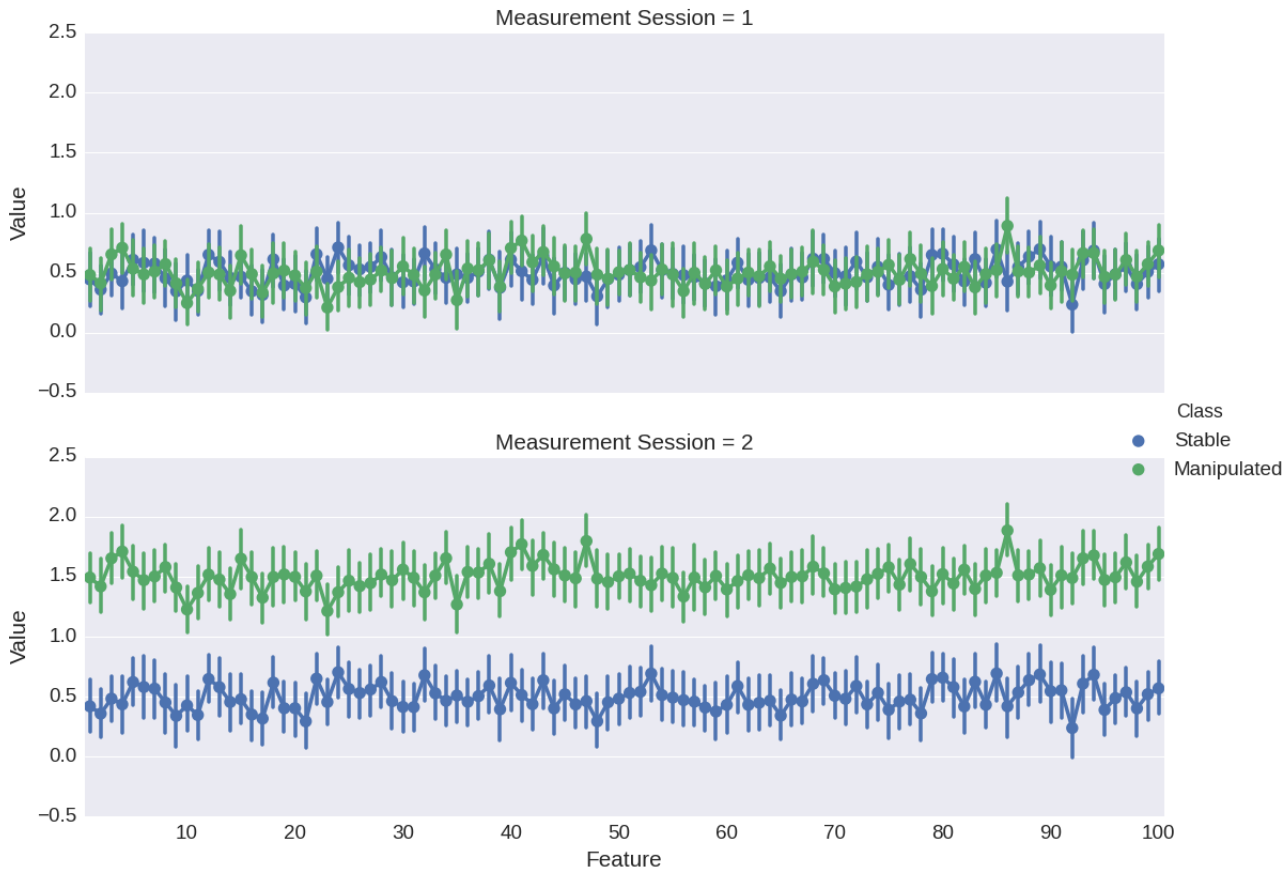

Figure 5– Chen and DuBois Simulation with Averaged Subjects

Plot of averaged features across subjects. The y-axis represents the arbitrary value of the features, with the x-axis representing features. The top plot is from the first measurement session, and the bottom plot is from the second measurement. Blue represents the stable class where this is no change from the first session to the second. Green represents the manipulated class where there is an increase in the feature values. Error bars represent 95% confidence intervals.

3.1.1.2 $\boldsymbol{\beta}$ Values

Figure 6 shows the average β value determined by the LSVM at each value of τ. β is used in the LSVM algorithm as the vector of scalars which define the magnitude relationship between features across sessions. In the case of two sessions, β is a single scaler which defines the scalar multiplier applied to the second session as it is added to the first session. In Figure 6, most of the β values are -1 indicating that a stable solution is simply subtracting the second session from the first session. The massive variation at the first τ value is likely as sign that the LSVM can not find an optimal solution. Evidence for this is in Figure 8 where the LSVM has 50% classification accuracy.

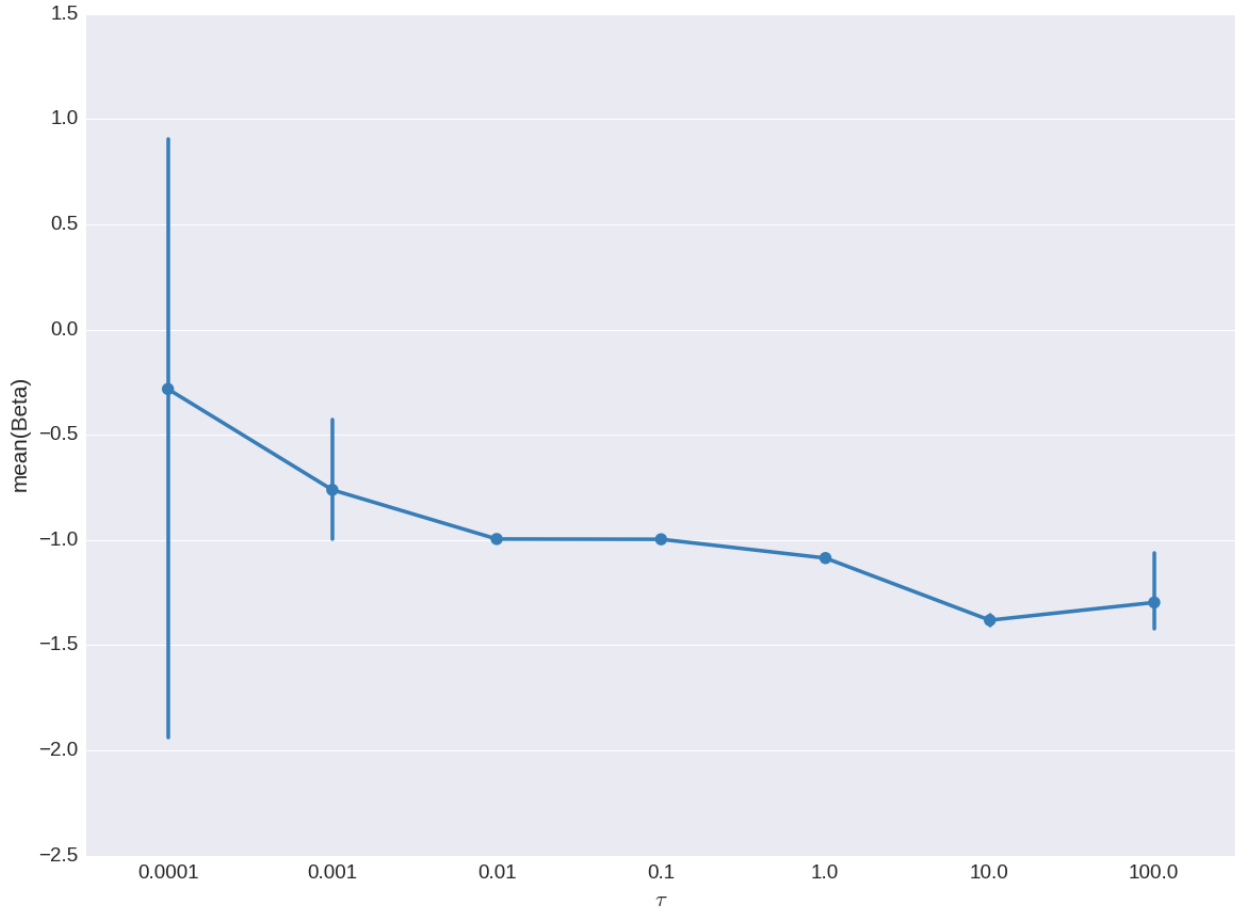

Figure 6 – Simulation One Beta Values

Plot of average β at various τ values. The x axis is the τ value, and the y axis is the average β value. The βs tended to be around -1. Error bars represent 95% confidence intervals.

3.1.1.3 C Values

Optimal C values during training were recorded in Figure 7. The C value refers to the amount of training error allowed by the algorithms. The LSVM and SVM share the same meaning behind C, that is that it refers to the magnitude of the penalty put on points that are on the wrong side of the margin. For LR, C is the penalty magnitude for error as measured by the log likelihood. As τ becomes larger the data becomes easier to separate, and results in smaller optimal C values.

Figure 7 – Simulation One C Values

Plot of Cs *at various* τ *values for various classifiers. Blue is used for LSVM, green for SVM, and red for LR. Dark solid colors represent the test data set, and light dashed colors represent the training sets. Lines represent the average result, while dots are the individual values. Error bars represent 95% confidence intervals. The trend seems to be that easier to separate data results in lower optimal C values.*

3.1.1.4 Classification Accuracy

Based on the results in Figure 8, the LSVM performs better than the SVM and LR, in terms of classification accuracy, but only over a range of τ values. This was calculated using all 200 subjects of data (100 subjects in each class). When the separation parameter τ is 0.001 the LSVM test performance starts to go up before the other classifiers, and peaks with perfect accuracy at a τ of 0.1. The SVM and LR classifiers do not reach perfect classification accuracy until a τ of 1.0. An interesting piece of information to note is that the SVM and LR classification accuracies seem to stay at chance until a τ of 0.1 is reached.

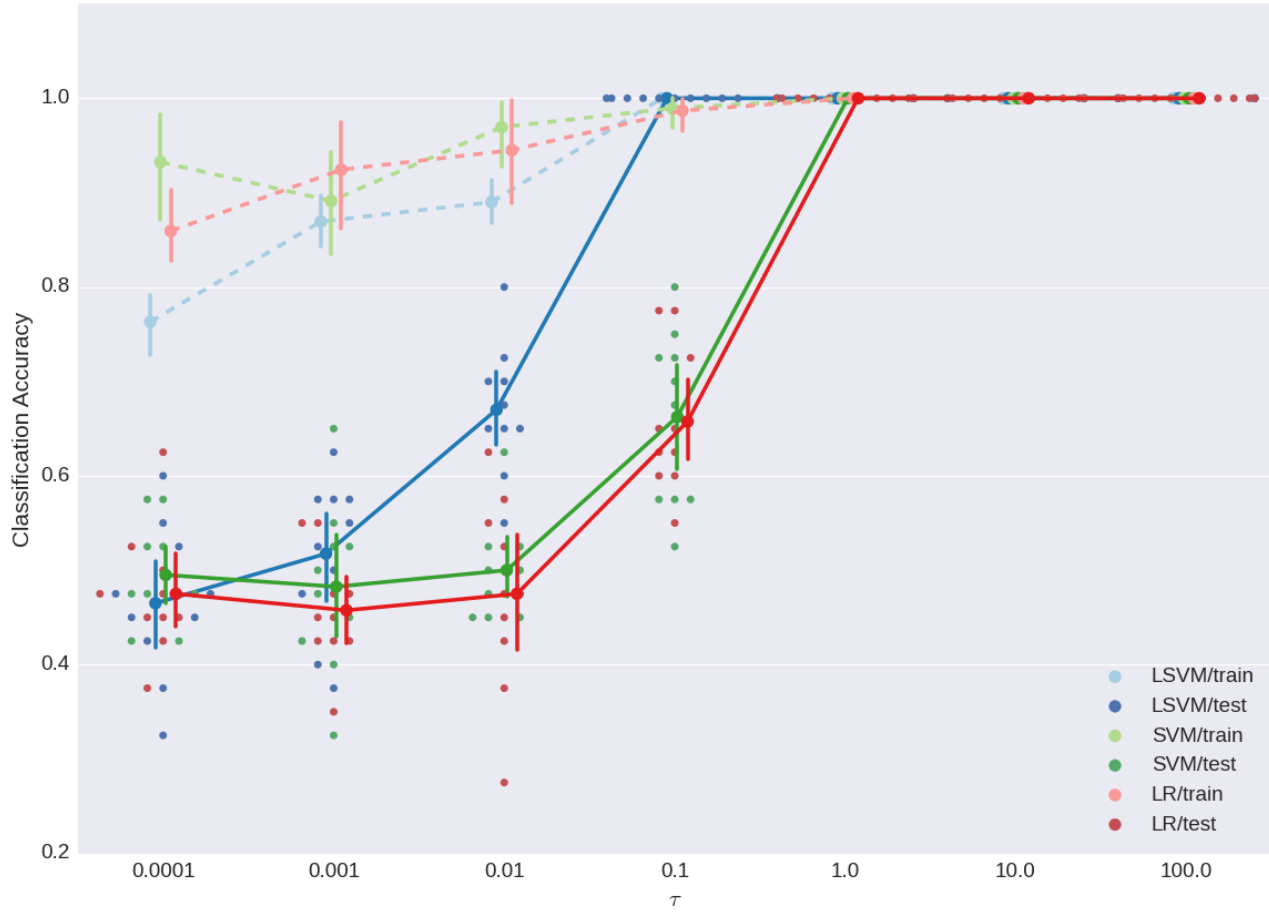

Figure 8 – Simulation One Classification Accuracies

Plot of classification accuracy at various τ values. The y-axis represents the classification accuracy, with the x-axis representing the τ value. Blue is used for the LSVM, green for SVM, and red for LR. Dark solid colors represent the test data set, and light dashed colors represent the training sets. Lines represent the average accuracy and dots represent the individual results. Error bars represent 95% confidence intervals. LSVM appears to be more robust as it has a higher classification accuracy over a larger range.

3.1.1.5 Model Interpretation

3.1.1.5.1 Weights

For the first simulation classifier weights were fairly flat, as the longitudinal trend in the features is just a constant increase selected from the same distribution for all features. The LSVM encompassed this property of the feature input space by having all the output feature weights around a constant negative value as shown in Figure 9. The SVM and LR classifiers show this property of the feature input space by having the second session weights being positive, and the first session weights being negative. This essentially shows the same thing as the LSVM; the most useful representation of the data is the difference between sessions irrespective of which feature is being considered.

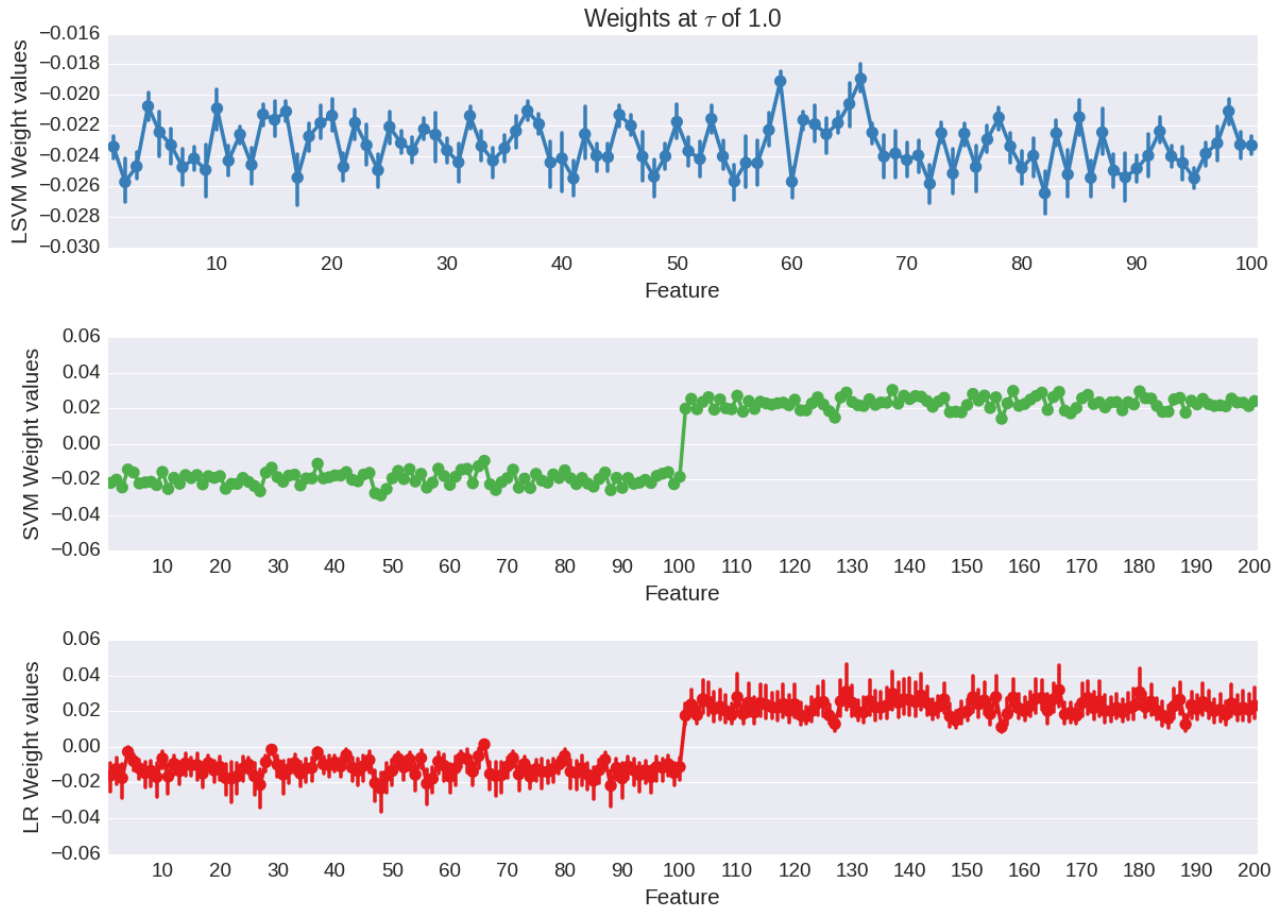

Figure 9 – Simulation One Weights at τ of 1

Plot of weight values for the various classifiers when τ was 1.0. The y-axis represents the weight, with the x-axis representing the feature. Blue is used for LSVM, green for SVM, and red for LR. Error bars represent 95% confidence intervals. All classifiers show through their weights that the most distinguishing feature of whether something is class one or two is the difference between session one and two features.

3.1.1.5.2 Hyperplane Distances

Figure 10 shows the distance from the hyperplane for each subject for the LSVM and SVM classifiers. As the value of τ decreases the separability of the data also decreases. In Figure 10, the hyperplane distance is shown for τ = 0.1, where the LSVM achieved perfect accuracy, but not the SVM as shown in Figure 8. For the SVM the distances are more variable, and in some cases ends up on the wrong side of the hyperplane. The LSVM maintains significant separation from the hyperplane where the SVM does not, which matches the findings regarding classification accuracy.

Figure 10 – Simulation One Hyperplane Distance at τ of 0.1

Plot of distances each subject is from the hyperplane at a separation of 0.1. The x axis is the subject, and the y axis is the distance from the hyperplane to the subject. Blue is used for the LSVM and green is used for the SVM. With a smaller separation the variance in distance from the hyperplane for subjects has increased, with some examples being incorrectly classified. Error bars represent 95% confidence intervals.

3.1.2 Simulation Two

After the first simulation in Study One, the next study sought to check if classifiers were able to correctly classify and obtain interpretable weights from more complex data. In simulation two, the features were linearly scaled between two values as seen in Figure 11. The stable class and the manipulated were scaled between two different values at session two, but the features were drawn the same way at session 1 as simulation one.

3.1.2.1 Simulation

Figure 11 displays the different patterns of variability in input features at a τ value of 1. The bottom of Figure 11, session two, shows how the two different classes scale their feature between 1 and -1, as described in equation (32). The stable class goes from 0 to 1, while the manipulated class goes from 0 to -1. τ controls the value to which the features trend towards. This makes the behaviour of this simulation similar to simulation one where smaller values of τ lead to a harder classification problem as the features between classes would overlap more.

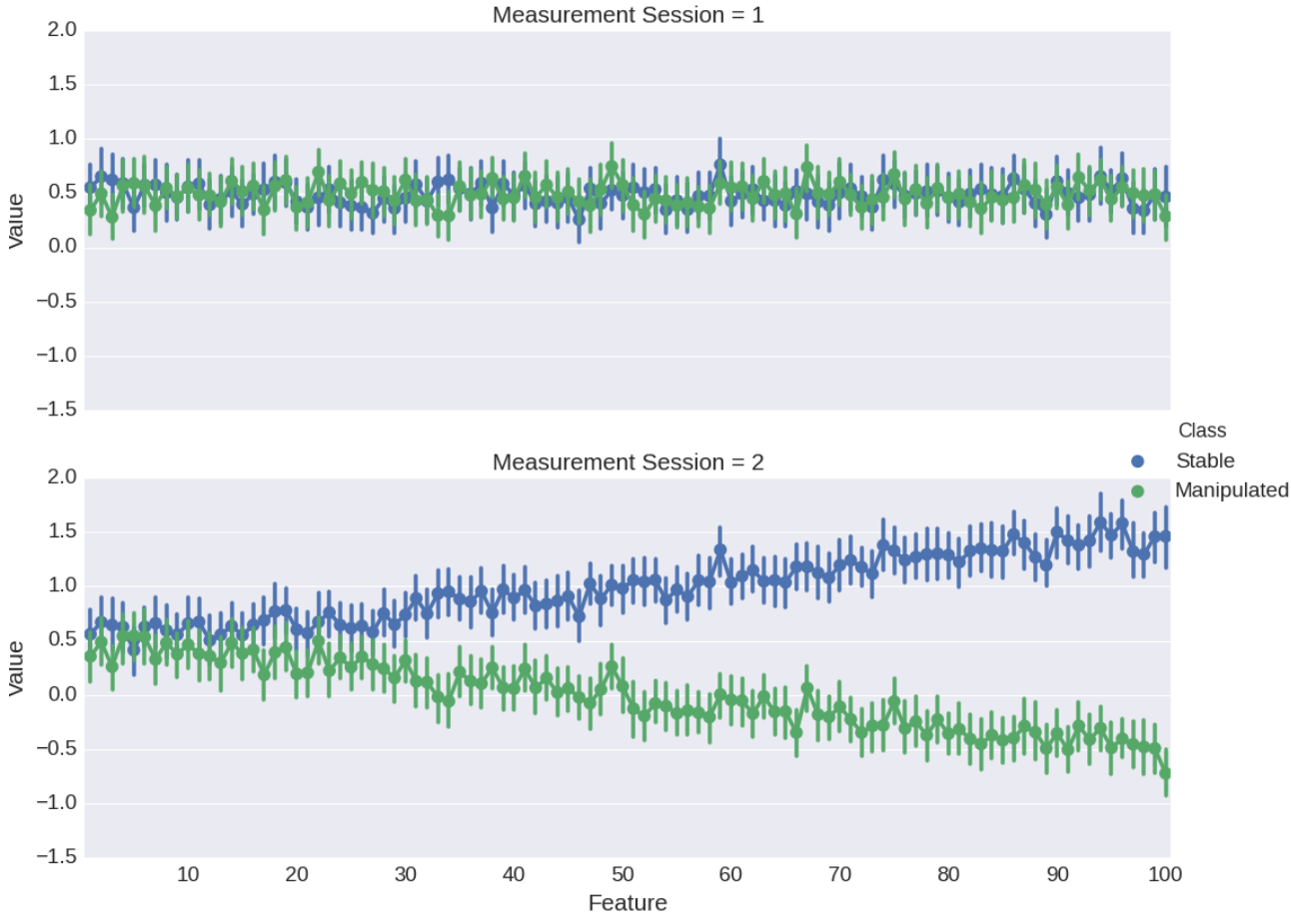

Figure 11 – Simulation Two With Averaged Features

Plot of averaged features across subjects. The y-axis represents the arbitrary value of the features, with the x-axis representing features. The top plot is from the first measurement session, and the bottom plot is from the second measurement. Blue represents the stable class where this is no change from the first session to the second. Green represents the manipulated class where there is an increase in the feature values. Error bars represent 95% confidence intervals. Diverging trend is clearly visible in the bottom portion of the image.

3.1.2.2 Classification Accuracy

Figure 12 shows the classification accuracy of various classifiers. This was calculated using all 200 subjects of data. The LSVM performs better again than the SVM and LR, in terms of classification accuracy, but only over a range of τ values. The LSVM is more robust at a τ of 0.1. From a value of τ value of 0.0001 to 0.01 all classifiers perform around chance. From a τ value of 1.0 and onward all classifiers achieve perfect classification accuracy.

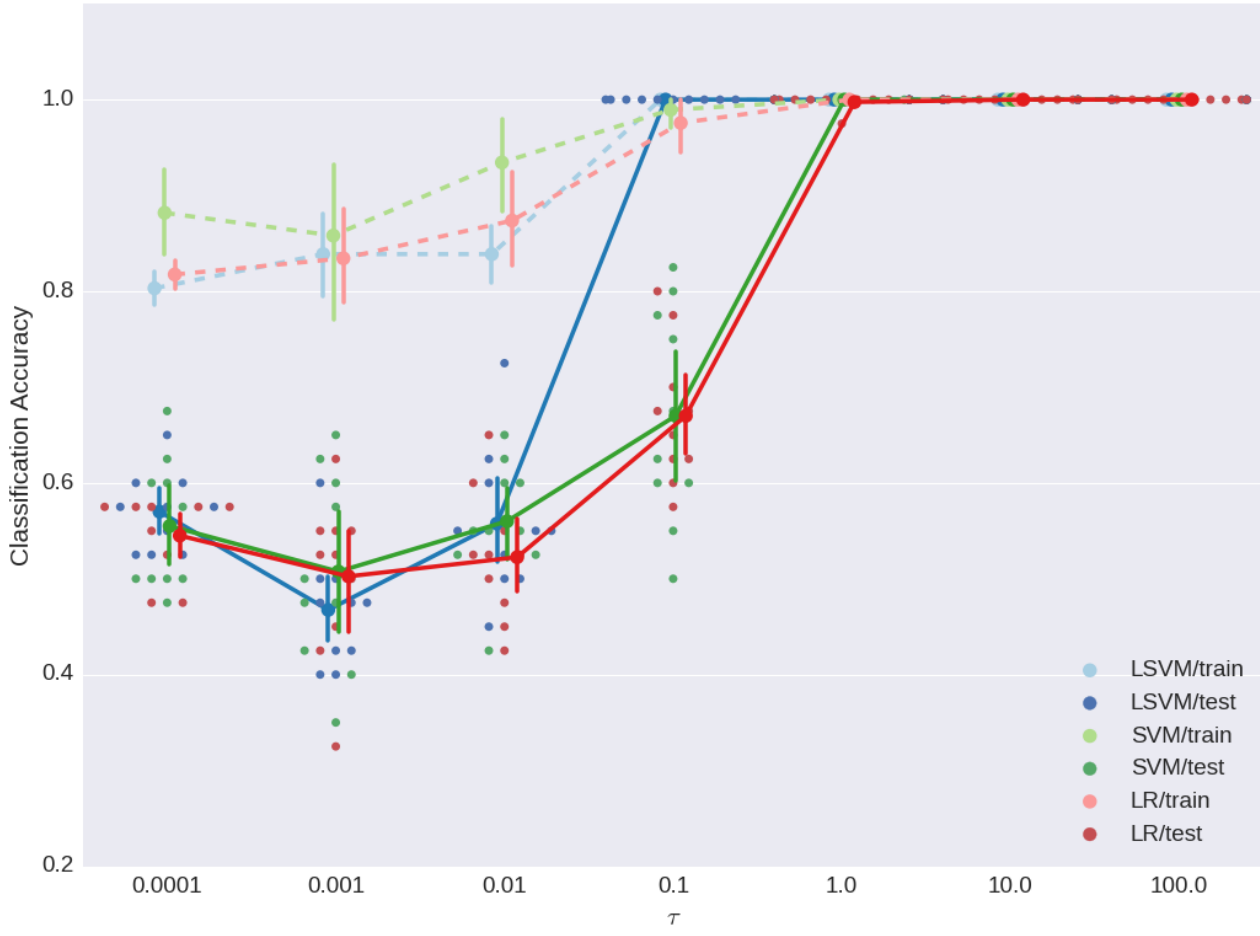

Figure 12 – Classification Accuracy of Simulation Two

Plot of classification accuracy at various τ values. The y-axis represents the classification accuracy, with the x-axis representing τ values. Blue is used for LSVM, green for SVM, and red for LR. Dark solid colors represent the test data set, and light dashed colors represent the training sets. Line represent the average accuracy and dots represent the individual results. Error bars represent 95% confidence intervals. LSVM is more robust as it increases in classification accuracy first.

3.1.2.3 Model Interpretation

3.1.2.3.1 Weights

Figure 13 shows the weights for various classifiers at a τ of 1.0, where all models have achieved perfect classification accuracy. Important to note is that all the classifiers clearly picked up the diverging trend by having weights that got larger at later features. For the LSVM this is shown by the constantly increasing weight values. For the SVM and LR this is shown by having the features at the second session start going down in magnitude. Interestingly, though some of the classifiers appear to have more noise than their counterparts, they still result in perfect classification accuracy. This is evident by the error bars for feature weights being larger for LR than the LSVM or SVM.

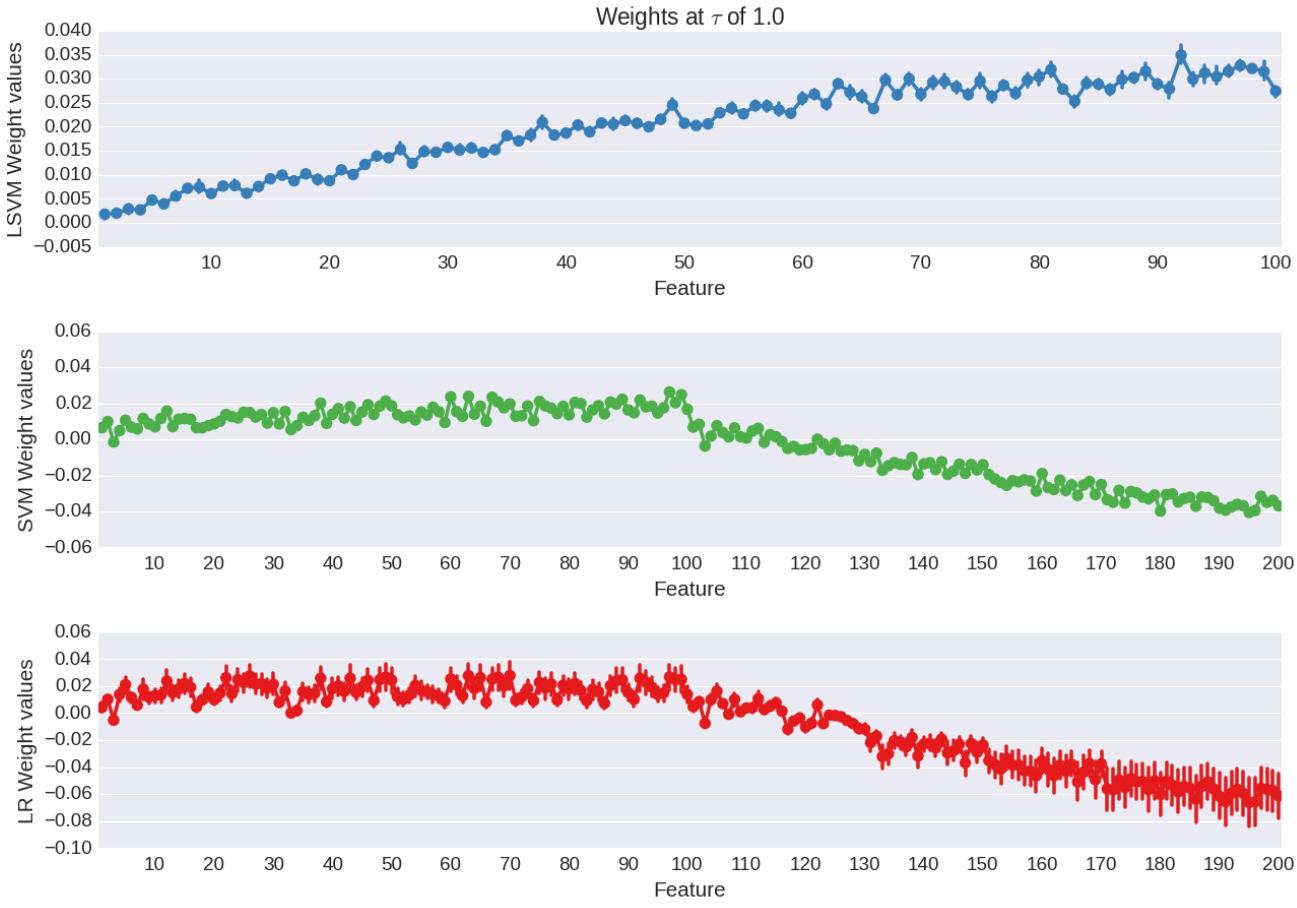

Figure 13 – Simulation Two Weights

Plot of weight values for the various classifiers when τ was 1.0. The y-axis represents the weight, with the x-axis representing the feature. Blue is used for LSVM, green for SVM, and red for LR. Error bars represent 95% confidence intervals. All classifiers show increasing importance placed on later features.

3.1.3 Heteroscedasticity

To simulate heteroscedasticity, the procedure outline in Section 2.2 was used. This was calculated using all 200 subjects of data. Variance of the distribution from which input feature values were drawn was varied for the second session data for both classes. This resulted in the classification accuracy vs. heteroscedasticity plot shown in Figure 14. In this case, ψ represents the amount of noise being added to the trends. The simulated signal is the same as Figure 4, except with ψ controlling the standard deviation of the distribution from which features at the second session are drawn from. Figure 14 shows that as the value of ψ increases, the classification accuracy decreases, likely due to increasing overlap between the two sessions input feature values due to increasing variance. The LSVM classification accuracy starts to decrease at larger values of ψ before the SVM and LR classification accuracy. This means that the LSVM is less robust against heteroscedasticity.

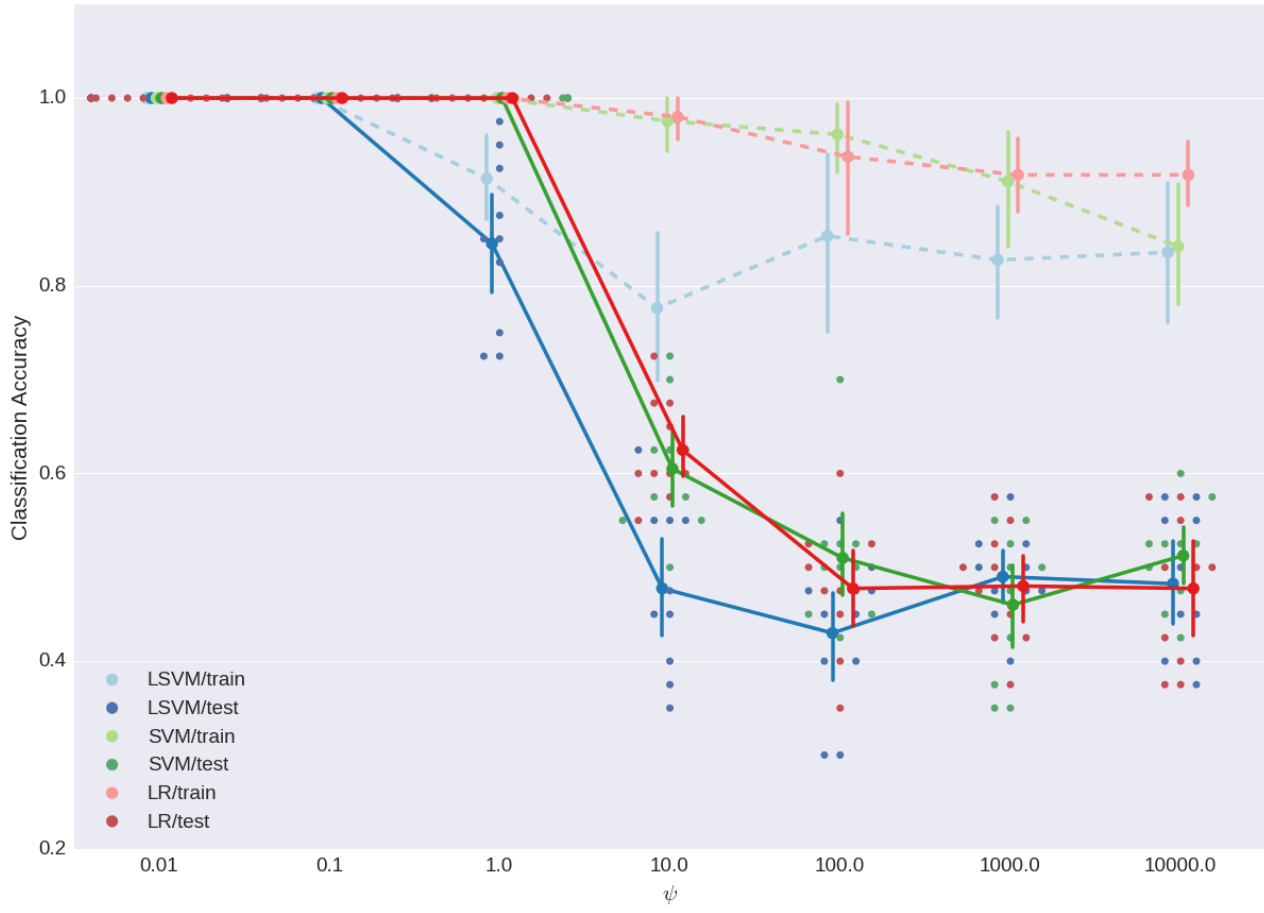

Figure 14 – Heteroscedasticity Classification Accuracy

Plot of classification accuracy at various ψ values. The y-axis represents the classification accuracy, with the x-axis representing the ψ. Blue is used for LSVM, green for SVM, and red for LR. Dark solid colors represent the test data set, and light dashed colors represent the training sets. Lines represent the average accuracy and dots represent the individual results. Error bars represent 95% confidence intervals. LSVM is less robust as it decreases in classification accuracy first.

3.2 Study Two: Resting State with Simulated Trend

3.2.1 Simulation

The top of Figure 15 shows the different peak magnitudes used for the Gaussian distribution for the two classes. To simulate the time course of MEG data the amplitude of a current dipole was manipulated to match the Gaussian distribution, shown in the bottom of Figure 15. The simulated data included 100ms of zero-value data before 0ms so that a baseline can be established during computation. The actual magnitude of the baseline varies since the current dipole is combined with real resting state MEG data. This is evident in the error bars in the top of Figure 15, which are based on (39. This variation is evidence that the real resting state MEG data contributed a non trivial amount of noise to the simulated signal. Figure 16->Figure 21 show the projection of the simulated current dipole onto the MEG sensor signals for the different sessions. Insertion of the current dipole is verified by the Gaussian like structure of the simulated time courses. The final data that is fed into the classifiers is composed of the base session combined with with a numbered session.

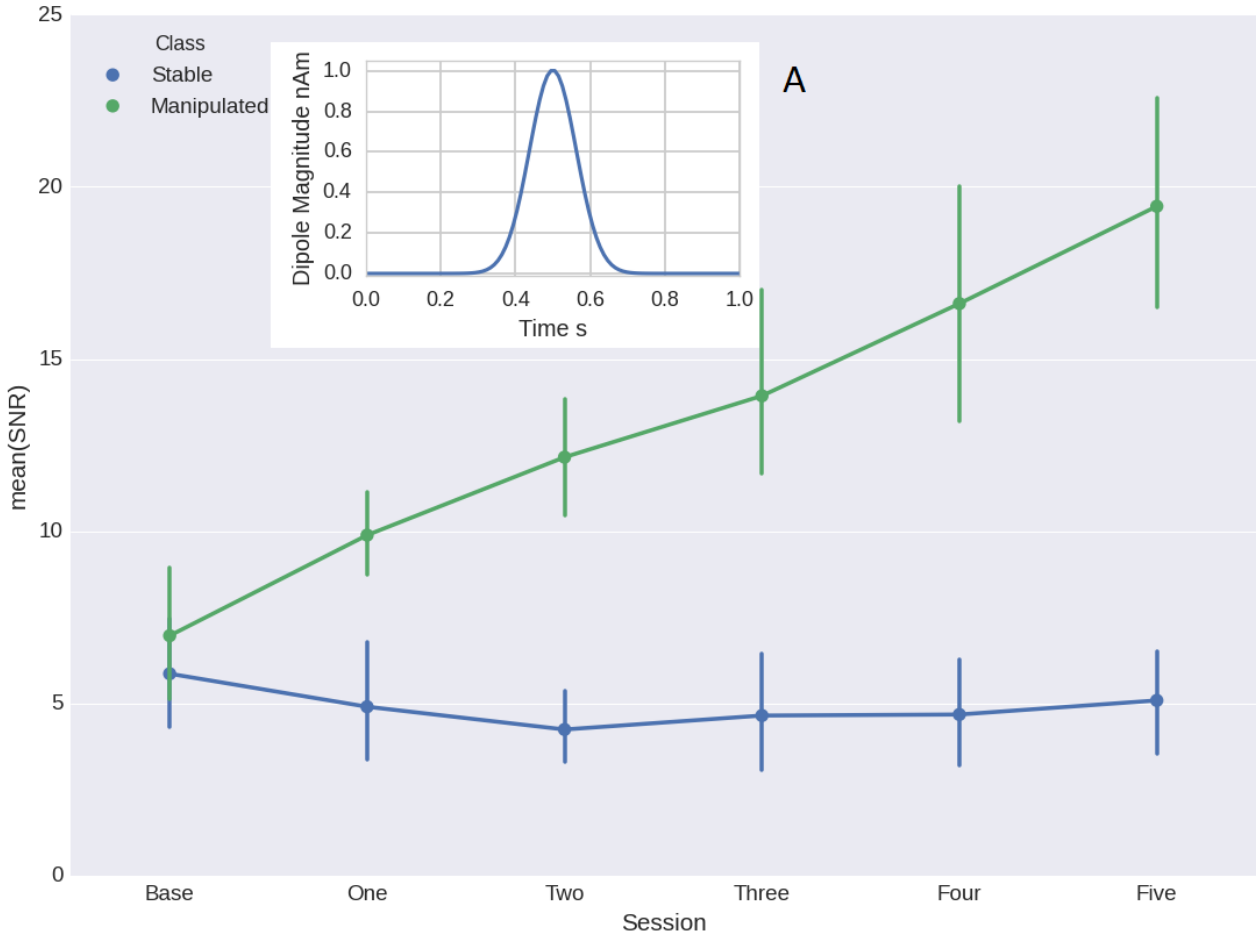

Figure 15 – Current Dipole Simulation

Inner A: Current dipole magnitude distribution over one second. X-Axis is time in seconds, and y axis is the dipole magnitude in nAm. Outer: Current dipole magnitudes for different sessions. X-Axis is the session, and y-axis is the mean SNR of the dipole once combined with real resting state MEG data. Error bars represent 95% confidence intervals across the 10 subjects of each class.

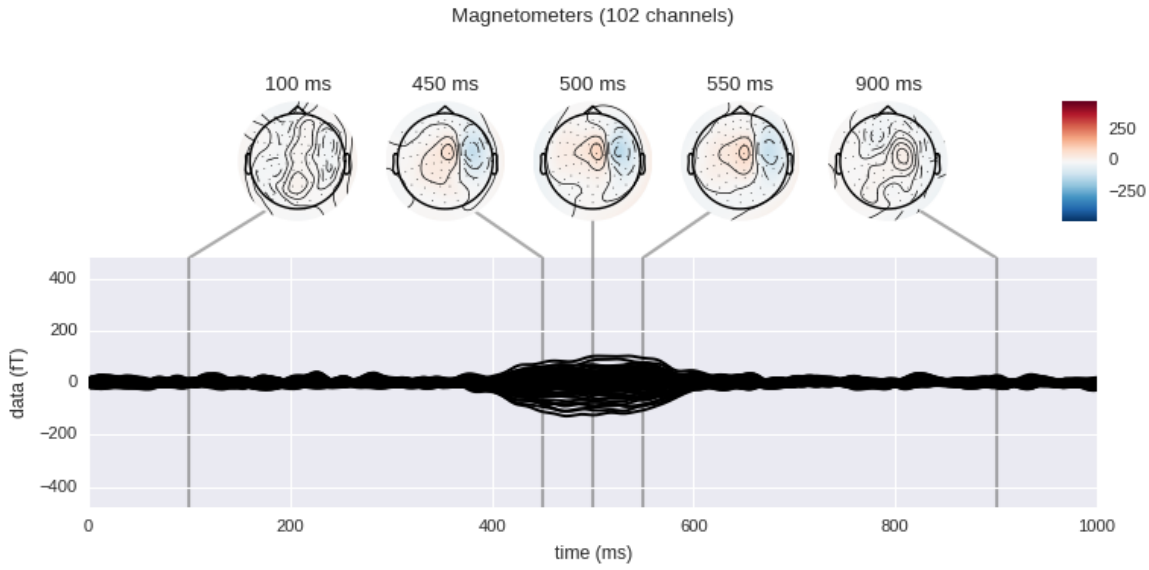

Figure 16 – First Session Butterfly Plot

Simulated subject for the first session after baseline, showing an example from the manipulated class. The x axis represents the time, and the y axis represents the amplitude of the sensor readings. Topographical representations of spatial activity are above the graph at specific latencies. Note the dipolar pattern of activity in the brain at 500ms. The field strength of this dipolar pattern increases across sessions, as shown in Figure 17->Figure 21.

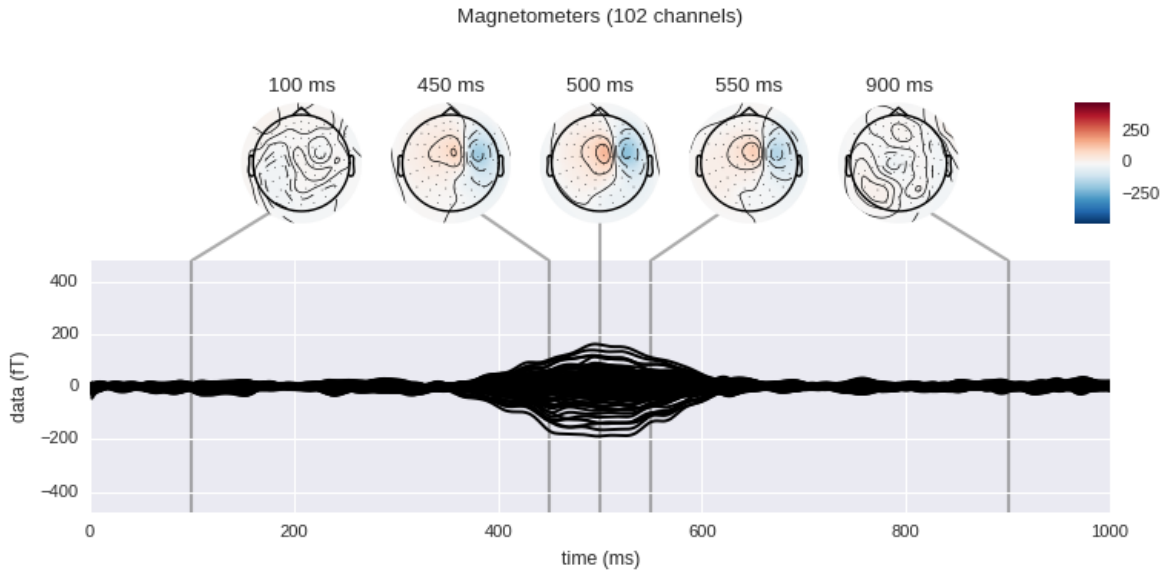

Figure 17– Second Session Butterfly Plot

Simulated subject for the second session after baseline, showing an example from the manipulated class. The x axis represents the time, and the y axis represents the amplitude of the sensor readings. Topographical representations of spatial activity are above the graph at specific latencies.

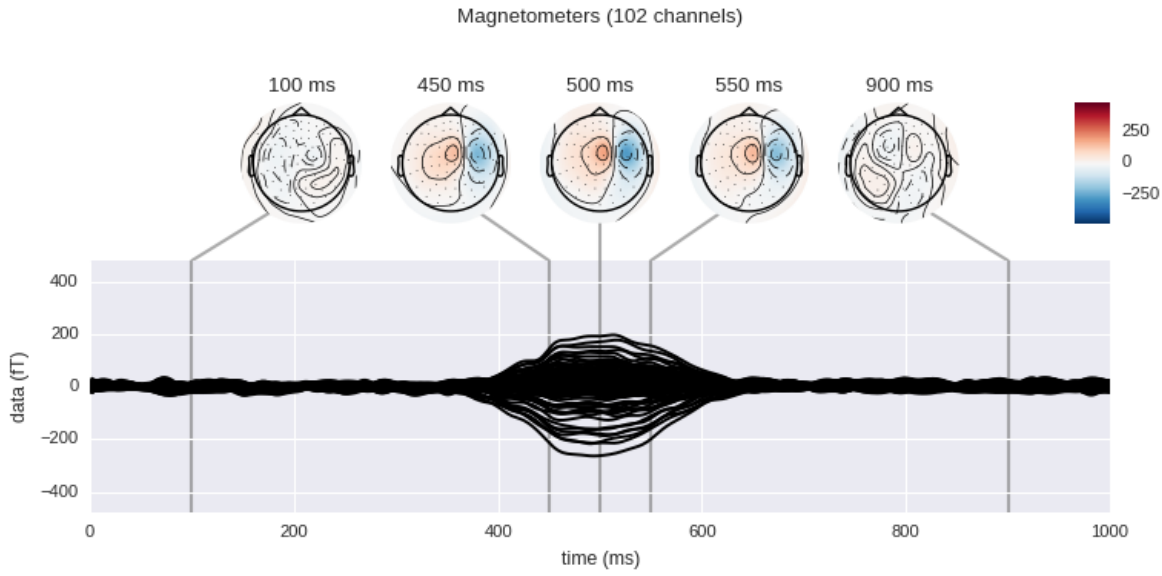

Figure 18– Third Session Butterfly Plot

Simulated subject for the third session after baseline, showing an example from the manipulated class. The x axis represents the time, and the y axis represents the amplitude of the sensor readings. Topographical representations of spatial activity are above the graph at specific latencies.

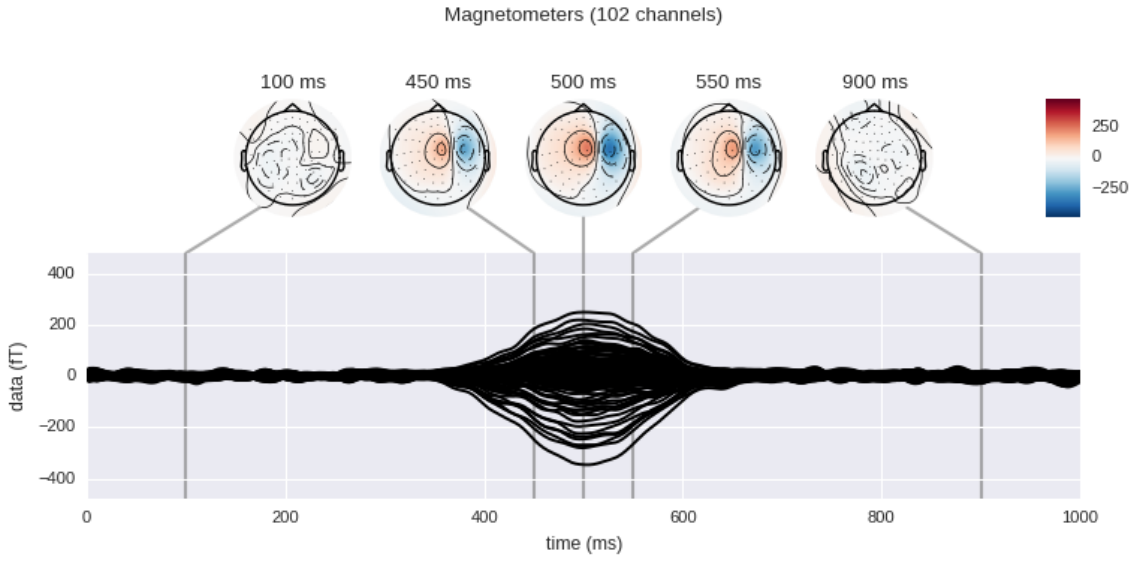

Figure 19– Fourth Session Butterfly Plot

Simulated subject for the fourth session after baseline, showing an example from the manipulated class. The x axis represents the time, and the y axis represents the amplitude of the sensor readings. Topographical representations of spatial activity are above the graph at specific latencies.

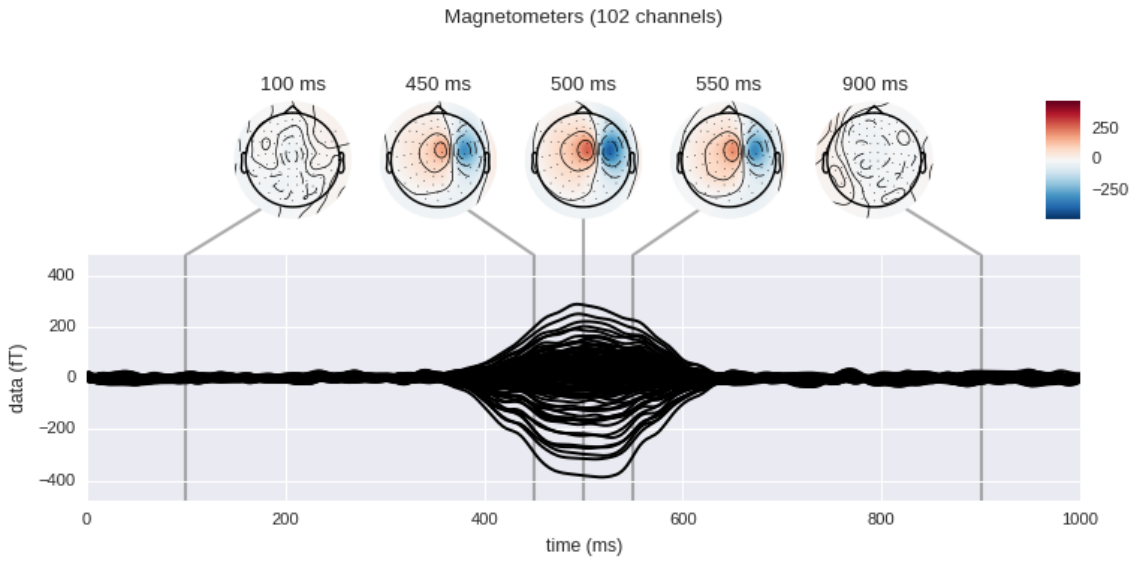

Figure 20– Fifth Session Butterfly Plot

Simulated subject for the fifth session after baseline, showing an example from the manipulated class. The x axis represents the time, and the y axis represents the amplitude of the sensor readings. Topographical representations of spatial activity are above the graph at specific latencies.

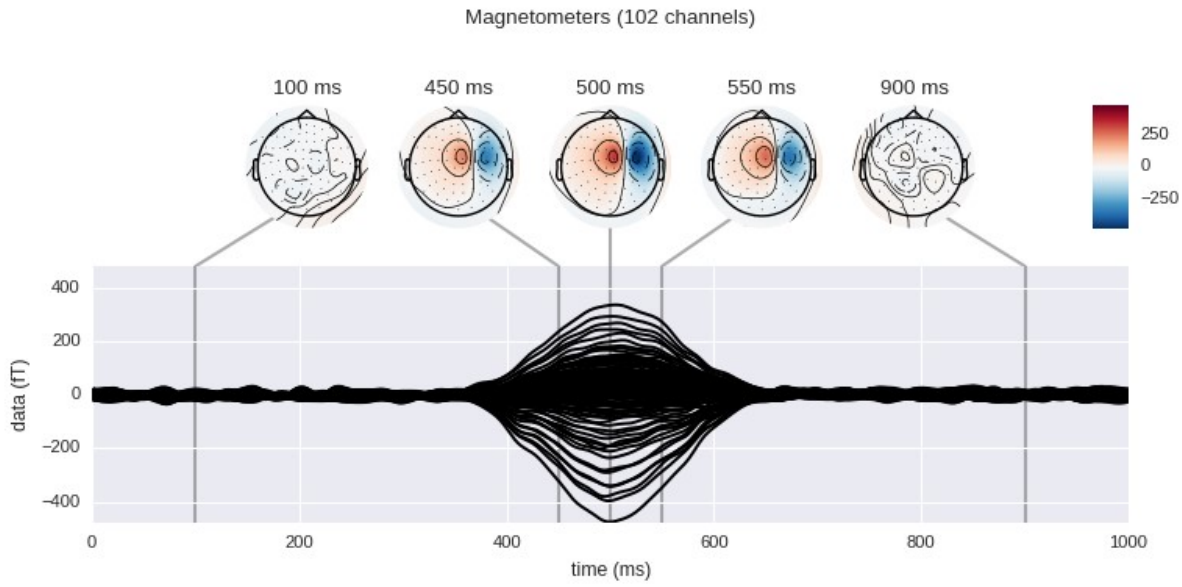

Figure 21– Sixth Session Butterfly Plot

Simulated subject for the sixth session after baseline, showing an example from the manipulated class. The x axis represents the time, and the y axis represents the amplitude of the sensor readings. Topographical representations of spatial activity are above the graph at specific latencies.

3.2.2 $\boldsymbol{\beta}$ Values

Figure 22 shows the average β value at various SNR values. β is used in the LSVM algorithm as the vector of scalars which adds future sessions to the baseline. In the case of two sessions, β is a single scalar which defines the scalar multiplier applied to the second sessions as it is added to the first session. β becomes a smaller negative number as the SNR value increases. This is likely due to the gap between the first session feature values and second session features values becoming larger. β then becomes smaller so that when the first and second session feature values are combined the output is close to 1. This behaviour is likely a result of the data not being scaled for the LSVM.

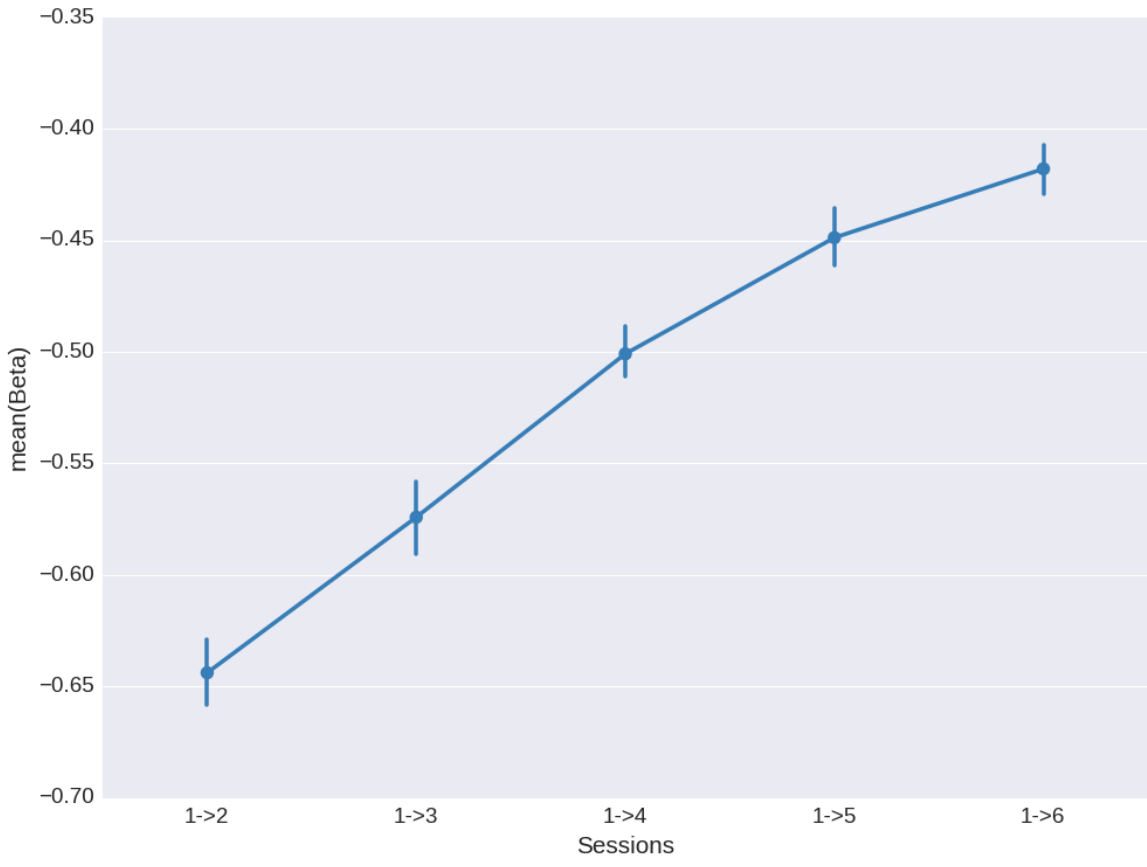

Figure 22 – Study Two β values

Plot of βs at various SNR values. The x axis is the SNR value, and the y axis is the average β value. As the separation between session one and two becomes larger the β values becomes larger. Error bars represent 95% confidence intervals.

3.2.3 Classification Accuracies

Figure 23 shows how different classifiers perform with and without scaling on the study two data. Classification accuracy was was calculated using all 20 subjects of data. The top performing classifiers were the no scaling LSVM, and scaling SVM/LR. This result is very clear as their counterparts had around 50% classification accuracy at the different SNR values (1->2, 1->3, etc.). The results from this figure were then used to choose which classifiers to show in Figure 24. Specifically, I moved forward with an LSVM with no scaling, SVM with scaling, and LR with scaling.

Figure 24 shows how the classifiers all had very similar performance at all SNR values. It is not possible to say any classifier is better than another one since the error bars overlapped. It is difficult to apply statistical tests to see if any of the classifiers are superior as the variation and distribution in classification accuracy at various points is unknown. As such, many standard statistical tests cannot be applied. Instead, the classification accuracies were compared qualitatively by comparing their overlap across all folds. The relatively similar performance of all three methods was also evident when PCA feature selection was used, in an effort to boost performance. It is also worth noting that PCA did not seem to have much effect on the classification accuracies.

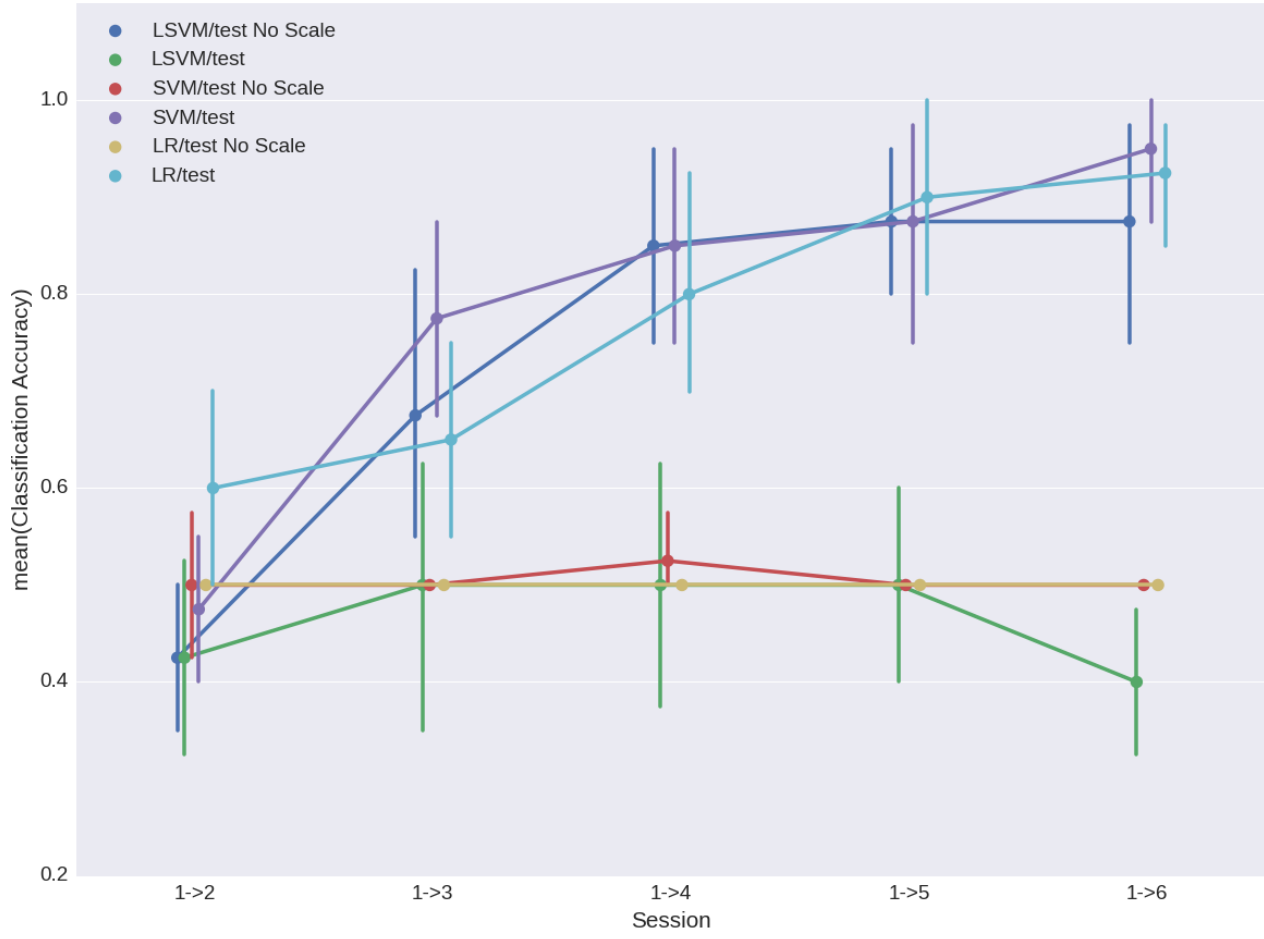

Figure 23 – Study Two Scaling Comparisons

Plot of classification accuracy at various SNR values, scaling options, and classifiers. The y-axis represents the classification accuracy, with the x-axis representing the SNR, with smaller differences between the numbers representing smaller SNR values. Error bars represent 95% confidence intervals. The LSVM with no scaling is clearly better than the LSVM with scaling, and the SVM/LR with scaling are better than the ones without scaling.

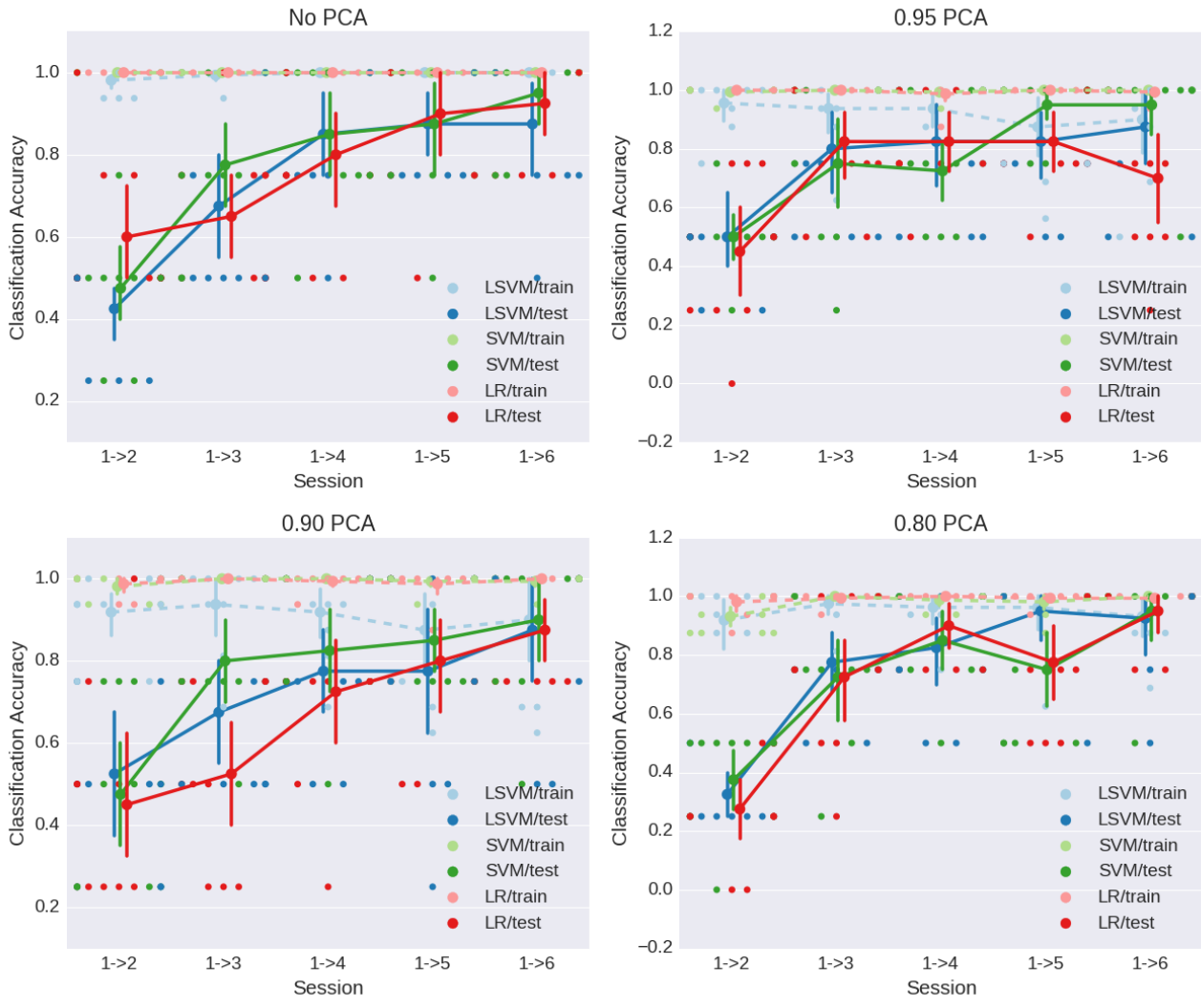

Figure 24 – Study Two Classification Accuracies

Plot of classification accuracy at various SNR values and different amounts of PCA preprocessing. The y-axis represents the classification accuracy, with the x-axis representing the separation of classes (larger values mean the data is easier to separate). Blue is used for LSVM, green for SVM, and red for LR. Dark solid colors represent the test data set, and light dashed colors represent the training sets. Lines represent the average, and dots are the individual values. The dots are quantized at 0% 25%, 50%, 75%, and 100% for the test set because there were 4 subjects in each test set (20% of the data). Error bars represent 95% confidence intervals. None of the classifiers

can be judged to be better than any of the other ones due to the large variance in classification accuracies.

3.2.4 Model Interpretation

Figure 25, Figure 26, and Figure 27 show the feature weights represented in the same format as MEG data for all three classifiers. The goal of reformatting the feature weights in this way was to visualize the weights in comparison to the input MEG evoked field data (as shown in Figure 25). Results for this dataset offer evidence that the LSVM was able to find weights that more closely matched the input signal than the SVM or LR. The used dataset was composed of the base and session six data (highest SNR) with no PCA. An objective measure of correlating the scaled raw signal and scaled feature weights suggest that this was the primary point of difference between the classifiers (where classification accuracy was greater than chance) as shown in Table 1. This means that the provided feature weights are the results of the "best case" scenario for this simulation.

The LSVM appears to be the best classifier in terms of interpretability of the weights, because the LSVM feature weights match more closely to the butterfly plots in Figure 21 than the other classifiers' feature weights. In terms of temporal dynamics, the LSVM feature weights (Figure 25) fall off quicker at latencies less than or greater than 500 ms, compared to the SVM (Figure 26) and LR (Figure 27) feature weights. Temporal patterns for LSVM feature weights seem to match more closely to the Gaussian curve used for the simulation by being smoother. Smoothing is a common technique to reduce the noise in a signal [21]. Signs of fit feature weights have larger variance means that noise is being fit, as the actual signal has no variance. In terms of spatial patterns, the topographical plots also give clear indication that the LSVM feature weights are more accurately matching the input signal. Most obviously at the peak latency of 500 ms, the SVM and LR feature weights both consider a very broad area of the topography (i.e., many sensors) to be of importance for classification. In contrast, the LSVM feature weights accurately accentuate the regions that show peak magnetic field strength in

Figure 25. This interpretation is supported by the increase in correlation between the grand average MEG data and the feature weights for the LSVM, as compared to SVM and LR, demonstrated in Table 1.

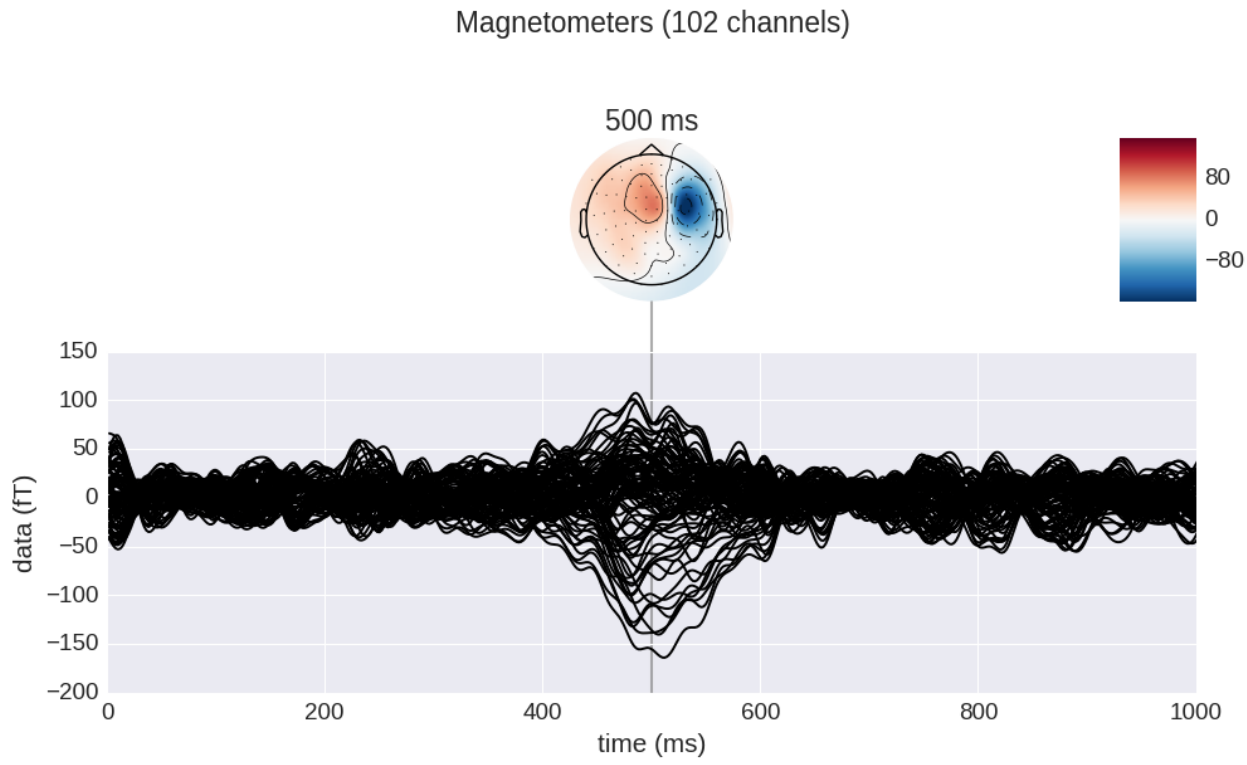

Figure 25 – Study Two Feature Weights LSVM

Plot of feature weight values for the LSVM when no preprocessing is done, and the highest SNR at session two is used. The bottom figure is a butterfly plot, showing the sensor weights over time. The y-axis represents the weight value for a sensor, with the x-axis representing the time course of the sensor. Topographical maps up top show the spatial weights at a specific time point. The LSVM topographical plots show that it was able to assign weights to specific areas in the brain where activity was simulated.

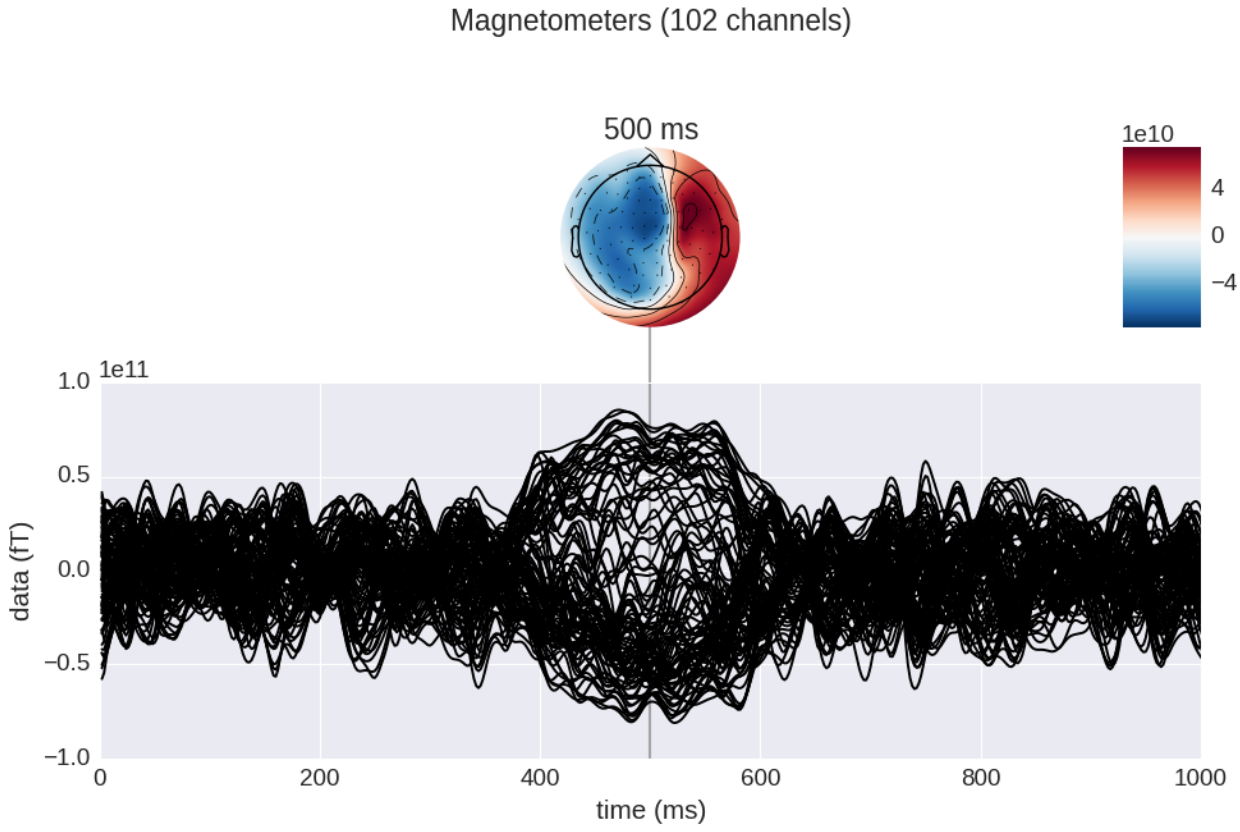

Figure 26 – Study Two Feature Weights SVM

Plot of feature weight values for the SVM when no preprocessing is done, and the highest SNR at session two is used. The bottom figure is a butterfly plot, showing the sensor weights over time. The y-axis represents the weight value for a sensor, with the x-axis representing the time course of the sensor. Topographical maps up top show the spatial weights at a specific time point. The SVM topographical plots show that the method was only precise enough to distinguish the left and right hemispheres.

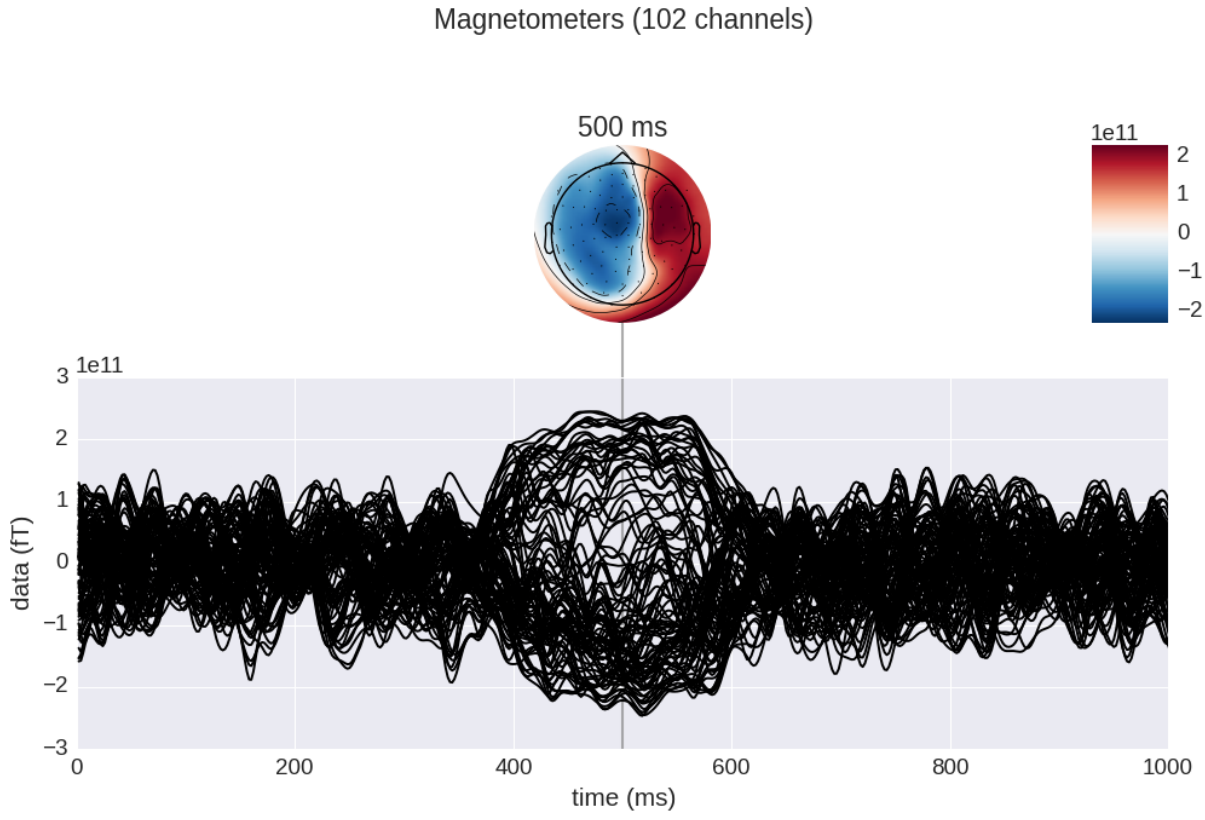

Figure 27 – Study Two Feature Weights LR

Plot of feature weight values for the LR when no preprocessing is done, and the highest SNR at session two is used. The bottom figure is a butterfly plot, showing the sensor weights over time. The y-axis represents the weight value for a sensor, with the x-axis representing the time course of the sensor. Topographical maps up top show the spatial weights at a specific time point. The SVM topographical plots show that the method was only precise enough to distinguish the left and right hemispheres.

Table 1 - Feature Weights Correlation

	LSVM Correlation	***SVM Correlation***	***LR Correlation***
1->2	*0.0938*	*0.266*	*0.284*
1->3	*0.351*	*0.378*	*0.387*
1->4	*0.413*	*0.436*	*0.446*
1->5	*0.502*	*0.51*	*0.484*
1->6	*0.679*	*0.606*	*0.597*

Chapter 4 Discussion

4.1 Summary of Main Findings

Four simulations were performed for this paper. The first simulation replicated the work of Chen and DuBois, which developed and tested the LSVM. The LSVM performed better than the SVM and LR in terms of classification accuracy, but only between τ values of 0.001 and 0.1 (Figure 8). For the first simulation classifier weights were fairly flat as the longitudinal trend in the features is a constant increase selected from the same distribution for all features at the same session. For the second simulation the LSVM performed better than the SVM and LR in terms of classification accuracy, between τ values of 0.01 and 1.0 (Figure 12). All the classifiers picked up the diverging trend by having weights that got larger at later features (Figure 13). Interestingly, LR appeared to have more noise in the feature weights than its' counterparts at perfect classification accuracy.

To simulate heteroscedasticity, the procedure outlined in Section 2.2 was used. ψ represents the magnitude of noise being added to the second session. The simulated signal was the same as the first simulation, except ψ controls the standard deviation of the distribution from which features at the second session were drawn from. As the value of ψ increased, the classification accuracy decreased, likely due to increasing overlap between the two sessions input feature values due to increasing variance (Figure 14). The LSVM classification accuracy was poorer than the SVM and LR classification accuracy for a range of values of ψ. This means that the LSVM performs poorer against heteroscedasticity.

For the second study a simulated current dipole signal was combined with resting state MEG readings (Figure 16->Figure 21). It was not possible to say any classifier was better than another in terms of classification accuracy since the error bars overlapped (Figure 24). The LSVM appeared to be the best classifier in terms of interpretability of the weights at the highest SNR dataset, because the LSVM feature weights matched more closely to the butterfly plots (Figure 21) than the other classifiers' feature weights. The topographical plots also gave clear indication that the

LSVM feature weights more accurately matched the input signal. Most obviously at the peak latency of 500 ms, the SVM and LR feature weights both consider a broad area of the topography (i.e., many sensors) to be of importance for classification. In contrast, the LSVM feature weights accurately accentuate the regions that show peak magnetic field strength (Figure 25). This interpretation is supported by the increase in correlation between the grand average MEG data and the feature weights for the LSVM, as compared to the SVM and LR, demonstrated in Table 1.

4.2 Impact of Random β Strategy on LSVM Performance

During initial replication of the LSVM algorithm developed by Chen and Dubois, I noted that the β values found during training were not always converging to a single value. Instead, the iterative algorithm would sometimes jump between two solutions, with neither being the optimal solution. To get around this, the optimal value could be found by restarting the LSVM fitting process multiple times with a random β value until convergence to a single β value. This occurred more frequently in hard to separate cases (low τ or low SNR value). One of the costs of having random starts for the β value was an increase in computation time. To get around the cost of increasing computation time, several constraints on the search were placed. The most significant was having the algorithm stop searching after a set number of iterations. This prevented the algorithm from looking too long for a β value when a bad start was provided. The computation time of the algorithm was still significant, and means that when training this algorithm in practice it will likely take a couple days or longer. Another limitation due to computation time is that only so many folds of the data can be done for cross validation, as it would take too long otherwise.

4.3 Selecting a Cross Validation Strategy

The cross validation strategy used in the study was important to prevent overestimating the accuracy of the classifier. Basing the reported classification accuracy off a single cross-validation split iteration could result in reporting accuracy for an outlier. A demonstration of this concept is that if 10 random folds of the data exist, and one of the folds happens to have easy to classify examples, the model will report a high

classification accuracy. This will be a false report, as it is giving the classification accuracy for the best performing examples in the data. When presented with unseen data that does not fit the model well, classification accuracy will be worse. This means the model will be less generalizable to data outside the training set. A more generalizable estimate of how performance would be with population data is also given by resampling the folds, as different smaller subsets of the data are tested multiple times. For this study a stratified resampling technique was used, as it allowed for multiple calculations of the classification accuracy, and therefore gave a mean and confidence interval of the distribution. Due to the computational time limit the resampling was only able to be run 10 times, with 95% confidence intervals computed from the 10 samples using bootstrapping. Bootstrapping in statistics is a technique to estimate the confidence interval of a measure, in this case the mean, by sampling a set of values thousands of times to generate a distribution. These confidence intervals were then used to estimate if the model would generalize well by looking at the mean classification accuracy, and the variability of it. For our data it was concluded that none of the classifiers performed better than any other as the error bars for each classifier's classification accuracy in Figure 24 overlapped with each other.

"Information leak" in cross-validation is an important issue that is common in machine learning since it becomes easy to accidently use some information from the test set when performing many operations on a set of data. Information leakage is bad, because it can cause researchers to overestimate generalizability of the model because a higher classification accuracy on the test set with less variability can be obtained. This occurs if information about the test set is used during training, then the model is essentially fitting to data it should not be able to see. This makes it seem like it is performing better on unseen data than it should be. An example of leakage during the study occurred through scaling the training and testing data sets together. This leakage was fixed by firstly estimating the mean and standard deviation of the data using only the training set, and then transforming the test set using the mean and standard deviation from the training set. The technique that worked best for us, to avoid leakage,

was setting the test data aside at the very beginning making sure not to touch the data until the very end. This was done by splitting the data into the train and test folds before any transformations were done, and making sure not to use the test fold until all training is done.

4.4 Comparing Classifiers

Visual inspection was used to compare the classification accuracy of classifiers as standard statistical tests which assume shared distributions, or shared variances were mostly not valid. Since the error bars on the graph corresponded to 95% variance from a bootstrap estimation it is likely that overlapping accuracies are similar. This was thought to be true even for slight overlap as the distribution is not known meaning the mean is not necessarily the area of concentration of values. An example can be seen in Figure 8 for a τ value of 0.001. At this τ value the LSVM classifier has 4 points above the the error bar, suggesting that the population concentration of values might be at the extreme, and the mean only represents that expected value because of extremes on the other side of the error bar. Using this methodology, one can say that the classifiers have similar performance at a τ of 0.0001 in the same figure, and at 0.01 the LSVM performs better.

Interpretability of feature weights was compared among classifiers by visual inspection, and for study two a correlation measure as well. Feature weights were plotted against features on the x-axis. Different classifiers were compared by how well the feature weight plots matched the simulated data, and how large the error bars are. For the first study, analysis was limited as the trends were very simple, and each classifier's feature weights essentially captured the same information. With the possible exception of LR which had noisier feature weights.

For the second study butterfly and topographical plots were used to compare the spatial and temporal aspects of the feature weights as shown in Figure 25, Figure 26, and Figure 27. As well, for the second study a correlation computed between the feature weights and the actual data provided an objective measure of model interpretability. The simulation session with the highest SNR was used when correlating

with the feature weights, as it provided the closest classification accuracy between classifiers.

As mentioned in the results, the LSVM appeared to be the best classifier in terms of interpretability of feature weights for study two. The temporal patterns for the LSVM feature weights seemed to match more closely to the simulated Gaussian curve upon inspection. The most important factor for this conclusion was the LSVM feature weights (Figure 25) falling off quicker at latencies less than or greater than 500 ms, compared to the SVM (Figure 26) and LR (Figure 27) feature weights. In terms of spatial patterns, the topographical plots gave clear indication that the LSVM feature weights are more accurately matching the input signal. Most obviously at the peak latency of 500 ms, the SVM and LR feature weights both consider a very broad area of the topography (i.e., many sensors) to be of importance for classification. In contrast, the LSVM feature weights accurately accentuate the regions that show peak magnetic field strength in Figure 25. These reasoned interpretations of the feature weights is supported by the correlation values in Table 1. A suprising result is that given the larger interpetability of the LSVM, the classification accuracy is not improved. A possible explanation for this is that the SVM/LR feature weights will still give you enough information to classify properly.

4.5 What Does C Say?

For the purely simulated data the values of C tended to decrease when the sessions were easier to separate. For the LSVM/SVM, the *C* parameter controls error tolerance. Making the value of C larger in equation (13) will punish the model more for having examples on the wrong side, resulting in smaller margins. Smaller values of C will punish the classifier less for misclassification and allow for larger margins. For the LSVM, the C value also controls error tolerance, but there is no margin for it to affect. With a small C value, if there is a point on the wrong side of the hyperplane, then the small C will counteract this to a degree. This leads to a larger margin, because according to equation (13) if C is small then the $\frac{1}{2}\left|\left|w\right|\right|^2$ term is punished more for being larger. And since the width of the margin is inversely related to the weight it increases when

the weights are forced to be smaller. Intuitively a larger margin can lead to better generalisation by taking more of the points into account.

For study one the C value shown in Figure 7 decreased as the τ value increased. This indicated that as the classes became harder to differentiate the value of C decreased. Since the value of C decreases with improving classification accuracy this suggests that tolerating some classification error provides the best solution to the problem. When classification error is higher, the value of C also has a higher variation. This was not shown in a figure as C values at low classification accuracy have massive variation with no consistent value. C likely has a larger variation at poorer classification accuracy because it is not able to find a single stable model to solve the problem, unlike at higher classification accuracy values.

4.6 Interpretability of Feature Weights

With the purely simulated data the LSVM and SVM provided better feature weights than LR based on the interpretability of the feature weights (LR having more variation in the feature weights). With MEG data the LSVM provided better feature weights when using the dataset with the highest SNR, as evidenced by visual inspection and a correlation measure. The correlation values in Table 1 suggested that the LSVM had more interpretable feature weights based on the higher correlation score between the original signal and the learned feature weights for the LSVM compared to the SVM and LR. Visual inspection suggested the LSVM feature weights were more precise about which MEG sensors and which time intervals were important for classification.

To explain why the LSVM has more interpretable weights it is important to note that the LSVM fits less data overall because it implicitly combines the first and second sessions feature measurements before finding the support vectors. A possible reason is that for all the simulations the best feature weight interpretations involved taking a combination of the features between session one and session two as there were clear simulated temporal trends. The LSVM may also have been helped by the fact that the simulated features (at least the most relevant ones) changed in the same direction. These models work well for the LSVM as the β term describes the direction of change,

and the weights can control the magnitude (though to be fair the LSVM feature weights can be opposite signs so that not all features have to change in the same direction). The LSVM makes an assumption that relevant information is obtained by combining the first and second session. It then makes sense for it to perform well when data matches these assumptions. The LSVM algorithm, however, seems to run into difficulty when the variation at the second session is larger than the first (Figure 14).

Results are only interpretable when a linear kernel is used, as this finds weights in the original space. If a polynomial kernel, for example, is used then the weights in the new space do not map back to the original space linearly, as there are more dimensions/features in the new space. Although higher classification accuracy may be achieved with the kernel trick, this means that the feature weights cannot be interpreted in the original dimension. Sometimes the extra classification accuracy that can be achieved with a kernel is worth it. For example, if the goal is not related to interpreting the weights that come out of the classifier. If the goal is diagnosis then non-linear kernels can be used, as the loss of feature interpretability is not relevant. If understanding what the classifier is doing is more important, then a linear kernel should be used so that the feature weights can be interpreted.

4.7 Different Strategies for Feature Selection

In the first study there was no feature selection as the classifiers were able to achieve a high classification accuracy without any feature selection. Feature scaling was done to centre the features around 0 and normalize the feature value spread. If the features were not scaled then the model would give the feature with larger values more importance in the objective function, even though that feature might not contain the most information. It is important to note that for the second study the LSVM did not scale the inputs as this gave higher classification accuracy. To make sure the LSVM was not cheating, by not scaling, the classifiers were all compared with no scaling as shown in Figure 23. The figure showed that the other classifiers did not benefit from no scaling, so the LSVM is unique in benefiting from that.

For the second study PCA was tried, which did not improve classification accuracy noticeably. PCA was done to see if classification accuracies could be improved by doing some feature selection, but no improvement was noted. Picking the number of components used by PCA was difficult, as that could significantly affect classifier performance. I ended up picking the number of components that would explain some percentage of the variation, as it was less arbitrary than picking an actual number of components. PCA also proved problematic with the LSVM as it could not be applied over the entire data set. If PCA was applied to the entire dataset then information could be combined between session 1 and 2, which invalidates the assumptions of the LSVM model. The LSVM model combines each feature independently across sessions, and an algorithm like PCA that could combine features across sessions might break that assumption. To get around this it was decided to apply PCA to the second session first, since I assumed that had more useful information than the first session, and then applied the same transform to the first session. It should be noted that this approach did not benefit classification in a noticeable way. Future research with LSVMs should investigate different feature selection techniques that can be applied to the entire dataset, but still preserve independence of features across sessions.

4.8 Heteroscedasticity

The problem trying to be solved with the LSVM (and other classifiers) is the correct classification of whether a subject is in one class or another. Based on the weights in Figure 9 the best performing classifiers learn to differentiate stable examples from manipulated examples by taking the difference between the features at the second session from the first session. The LSVM was found to be more robust in simulation one, and I hypothesize this is because it explicitly combines the first and second session together. Unfortunately, as the SNR was made worse the LSVM started to perform worse than the SVM and LR. As a reminder, the LSVM tries to learn from some linear combination of the sessions, where all features are combined across sessions using the same scaling term. Specifically, the simplest case of two time points is examined

$$X_s = X_{s,t=0} + \beta X_{s,t=1} \quad (40)$$

The basic idea of the simulation was that the stable class has no change between the first and second session means, while the manipulated class has the feature values shifted up by 1.0 at the second session. In the heteroscedasticity simulation the variance at the second session is varied for both classes.

I have two main hypotheses for why the LSVM reached chance performance before the SVM and LR as heteroscedasticity increased. The first is that by taking the difference before applying the learning algorithm the LSVM is throwing out information that the algorithm could possibly be using to increase the SNR. That is, the SVM and LR keep the first and second session data when developing their models. This means that they have twice as many points when developing their models, so it can be said that a greater SNR is available. My other theory is based on the triangle inequality

$$||A + B|| \leq \big||A|\big| + ||B|| \quad (41)$$

Essentially the LSVM is adding the first and second session together before computing the distance to the hyperplane, the A and B term in equation 41. The SVM and LR add the first and second session together as evidenced by the weights, but do so after computing the distances to the hyperplane. This matches the triangle equality in equation 41 so that the LSVM corresponds to the left expression, and the SVM and LR correspond to the right side. This would mean that the LSVM points are either the same distance or closer to the hyperplane than the SVM or LR.

4.9 Generalizability

In the first study, the LSVM performed better on the test sets compared to the SVM or LR. This was shown by having the LSVM achieving 100% classification accuracy on the test sets with lower values of τ. The first study was still poor evidence for the LSVM performing better in the real world as the data was purely simulated. For this reason, study two with real resting state MEG noise was done.

In the second study none of the classifiers performed better on the test set than any of the others. This is evidence that when working with real world data the LSVM

will not perform any better than the SVM or LR. I suspect that for the LSVM to perform better, better preprocessing techniques which reduce the amount of information would be necessary. It might also turn out that the LSVM is better or worse with real world data, and similar performance only occurs for this simulation. For this reason, further testing is advised before any solid conclusions can be made.

In terms of feature weight interpetability, the LSVM performed similarly or better than the other methodologies for all the studies. This suggests that when applied to real world data the LSVM could provide more interpretable feature weights for easier to classify cases. However, the results of the heteroscedasticity simulation suggest that the LSVM is likely to have trouble with real world longitudinal data (see below). It is difficult to make a solid claim about this as the trends used in this study are fairly simple and contained some simulated aspects.

A problem with the initial simulation proposed by Chen and DuBois was that the variance at the second session was not very different from the first session. When this aspect of the simulation was broken in my thesis by adding heteroscedasticity, the LSVM performed worse than SVM and LR. A stronger claim about the LSVM having more interpretable feature weights could be made once studies involving real world data are done. Since the LSVM only performed better in the simplest simulated cases (and worse with heteroscedasticity),I would not expect that the LSVM will perform better than the SVM or LR with real data.

4.10 Challenges

The LSVM proved difficult to implement. For one, the iterative solution did not always settle at a single solution. While both stages of the alpha/beta optimization are convex, it seems that together they are no longer convex in my implementation of the algorithm. Thus, the optimization proved expensive to computation time as it would leave the program to jump between solutions. This was reduced by implementing a check, and then stopping when it started happening, but computation time increased again when the β value was reset several times to try and find a stable solution. It was also complicated to implement the LSVM so that it would interact with everything

properly. The LSVM was coded so that it would interact properly with the scikit-learn library, and also implemented some of the functions from the library. This required that the LSVM code had the proper interfaces and methods. Implementing pre-processing methods also proved more difficult with the LSVM as no techniques that (like PCA) combined features across sessions could be allowed to do so. Combining features across sessions could result in an unbalance in features in session one vs session two, breaking the LSVM model. This meant that any precautions taken with the LSVM also had to be taken with the other classifiers so that they could be compared.

There are several limitations faced when analysing longitudinal neuroimaging data with the LSVM proposed by Chen and Dubois. For one the model can not handle missing session scans. If subjects had to come for multiple scans, but missed some, there is no built in way to handle the missing data. With the current LSVM model the missing data would have to be filled in with another technique or dropped. If the time at which the sessions are measured vary between subjects, then the LSVM model is not able to contain that information. That is, if one subject has a week between sessions and the other waits a month, the LSVM is not able to represent this information internally. The LSVM has also only been implemented for cases with two sessions so far, which severely limits its real world application. For future work this would be one of the essential concepts to show working.

4.11 Significance from a Clinical Perspective

Due to the highly simulated nature of this study I would not suggest applying the LSVM in a clinical setting without first applying it to a study involving real MEG data. The main reason for this being that this study suggested that with near-real MEG data classification accuracy is not different between the classifiers. Since an SVM or LR already exist and have easy to use libraries, those techniques should be used first. If a clinician wants to use the LSVM to investigate its feature weights, then I would still suggest further studies. The reason for this being the LSVM has only been shown to find better feature weights with the highest SNR data, but LSVM performance is reduced in data with heteroscedasticity, which is common in longitudinal data. To make a

statement about how it would operate with real MEG data, a study involving real MEG data would have to take place. The LSVM is still a promising technique, and has the possibility to help clinicians extract useful information about the spatial and temporal patterns in brain activity for longitudinal studies.

Chapter 5 Conclusion

In conclusion, this study showed how the LSVM technique proposed by Chen and Dubois has some potential to provide better feature weights than an SVM or LR. Better feature weights mean that the feature weights learned by the model closely match the actual brain activity occurring during the task. This would have application in determining what brain activity drives classification accuracy for prognosis or diagnosis. However, based on current results the evidence for using the LSVM over SVM/LR is not there. In terms of classification accuracy the LSVM is only superior or on par with other methods when the change between sessions is simple. The introduction of greater noise at the second session causes the LSVM to perform worse, and I would expect that to happen with real world data. In terms of feature weight interpretability the LSVM is only better (based on correlation values) when the dataset with the highest SNR is used, which is unlikely in real world application again.

To properly evaluate this potential, future studies would be needed to firstly see how the LSVM performs with real MEG data in terms of classification accuracy and feature weights. . Future studies would likely involve MEG data from a task where brain activity is well known, and can easily be modulated. This would allow for constructing a longitudinal neuroimaging dataset. Some advances to the LSVM technique might improve its performance as well. Better feature selection could be developed which allows for selecting features using all data while keeping them consistent across sessions. Another change could be finding a way to estimate α and β that converges better.

Bibliography

1. Skup, M., *Longitudinal fMRI analysis: A review of methods.* Stat Interface, 2010. **3**(2): p. 235-252.
2. Bernal-Rusiel, J.L., et al., *Statistical analysis of longitudinal neuroimage data with Linear Mixed Effects models.* Neuroimage, 2013. **66**: p. 249-60.
3. Ganesan, R., et al., *Comparative study of linear mixed-effects and artificial neural network models for longitudinal unbalanced growth data of Madras Red sheep.* Veterinary World, 2014. **7**(2): p. 52-58.
4. Chen, S. and F. DuBois Bowman, *A novel support vector classifier for longitudinal high-dimensional data and its application to neuroimaging data.* Statistical Analysis and Data Mining, 2011. **4**(6): p. 604-611.
5. Kirkwood, M.W., et al., *Management of pediatric mild traumatic brain injury: a neuropsychological review from injury through recovery.* Clin Neuropsychol, 2008. **22**(5): p. 769-800.
6. Gaetz, M. and D.M. Bernstein, *The current status of electrophysiologic procedures for the assessment of mild traumatic brain injury.* J Head Trauma Rehabil, 2001. **16**(4): p. 386-405.
7. Pfurtscheller, G. and F.H. Lopes da Silva, *Event-related EEG/MEG synchronization and desynchronization: basic principles.* Clinical Neurophysiology, 1999. **110**(11): p. 1842-1857.
8. Givre, S.J., *Essentials of Clinical Neurophysiology, Third Edition.* Journal of Neuro-Ophthalmology, 2005. **25**(1): p. 61-62.
9. Liu, Z., L. Ding, and B. He, *Integration of EEG/MEG with MRI and fMRI in Functional Neuroimaging.* IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine & Biology Society, 2006. **25**(4): p. 46-53.
10. Logothetis, N.K., et al., *Neurophysiological investigation of the basis of the fMRI signal.* Nature, 2001. **412**(6843): p. 150-157.
11. Roberts, T.P.L., et al., *MEG Detection of Delayed Auditory Evoked Responses in Autism Spectrum Disorders: Towards an Imaging Biomarker for Autism.* Autism research : official journal of the International Society for Autism Research, 2010. **3**(1): p. 8-18.
12. Ahonen, A.I., et al., *122-channel squid instrument for investigating the magnetic signals from the human brain.* Physica Scripta, 1993. **1993**(T49A): p. 198.
13. Vrba, J. and S.E. Robinson, *SQUID sensor array configurations for magnetoencephalography applications.* Superconductor Science and Technology, 2002. **15**(9): p. R51.
14. Vrba, J., *Squid Gradiometers in Real Environments*, in *SQUID Sensors: Fundamentals, Fabrication and Applications*, H. Weinstock, Editor. 1996, Springer Netherlands: Dordrecht. p. 117-178.
15. Hämäläinen, M., et al., *Magnetoencephalography: theory, instrumentation, and applications to noninvasive studies of the working human brain.* Reviews of Modern Physics, 1993. **65**(2): p. 413-497.
16. O'Toole, A.J., et al., *Theoretical, Statistical, and Practical Perspectives on Pattern-based Classification Approaches to the Analysis of Functional Neuroimaging Data.* J.Cognitive Neuroscience, 2007: p. 1735-1752.
17. Petersson, K.M., et al., *Statistical limitations in functional neuroimaging. I. Non-inferential methods and statistical models.* Philosophical transactions of the Royal Society of London.Series B, Biological sciences, 1999. **354**(1387): p. 1239-1260.

18. Petersson, K.M., et al., *Statistical limitations in functional neuroimaging. II. Signal detection and statistical inference.* Philosophical transactions of the Royal Society of London.Series B, Biological sciences, 1999. **354**(1387): p. 1261-1281.
19. Cao, L.J., et al., *A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine.* Neurocomputing, 2003. **55**(1–2): p. 321-336.
20. Friston, K.J., *Modes or models: a critique on independent component analysis for fMRI.* Trends in Cognitive Sciences, 1998. **2**(10): p. 373-375.
21. Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. 10 ed. Springer Series in Statistics. 2009: Springer-Verlag.
22. Mitchell, T., *Machine Learning*. 1997, McGraw Hill. p. 2.
23. Nho, K., et al., *Automatic Prediction of Conversion from Mild Cognitive Impairment to Probable Alzheimer's Disease using Structural Magnetic Resonance Imaging.* AMIA Annu Symp Proc, 2010. **2010**: p. 542-6.
24. Blankertz, B., et al., *The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials.* IEEE Transactions on Biomedical Engineering, 2004. **51**(6): p. 1044-1051.
25. Chen, S., et al., *Some recent statistical learning methods for longitudinal high-dimensional data.* Wiley Interdisciplinary Reviews: Computational Statistics, 2014. **6**(1): p. 10-18.
26. LaConte, S., et al., *Support vector machines for temporal classification of block design fMRI data.* NeuroImage, 2005. **26**(2): p. 317-329.
27. Haynes, J.-D. and G. Rees, *Decoding mental states from brain activity in humans.* Nat Rev Neurosci, 2006. **7**(7): p. 523-534.
28. Cox, D.D. and R.L. Savoy, *Functional magnetic resonance imaging (fMRI) "brain reading": detecting and classifying distributed patterns of fMRI activity in human visual cortex.* NeuroImage, 2003. **19**(2): p. 261-270.
29. Haxby, J.V., et al., *Distributed and Overlapping Representations of Faces and Objects in Ventral Temporal Cortex.* Science, 2001. **293**(5539): p. 2425-2430.
30. Haynes, J.D. and G. Rees, *Predicting the orientation of invisible stimuli from activity in human primary visual cortex.* Nat Neurosci, 2005. **8**(5): p. 686-91.
31. Yang, B., et al., *New KF-PP-SVM classification method for EEG in brain-computer interfaces.* Biomed Mater Eng, 2014. **24**(6): p. 3665-73.
32. Wee, C.-Y., et al., *Identification of MCI individuals using structural and functional connectivity networks.* NeuroImage, 2012. **59**(3): p. 2045-2056.
33. Fair, D.A., et al., *Distinct neural signatures detected for ADHD subtypes after controlling for micro-movements in resting state functional connectivity MRI data.* Front Syst Neurosci, 2012. **6**: p. 80.
34. Kaper, M., et al., *BCI competition 2003-data set IIb: support vector machines for the P300 speller paradigm.* Biomedical Engineering, IEEE Transactions on, 2004. **51**(6): p. 1073-1076.
35. Vapnik, V., *The Nature of Statistical Learning Theory*. 1995: Springer.
36. Mjolsness, E. and D. DeCoste, *Machine learning for science: state of the art and future prospects.* Science, 2001. **293**(5537): p. 2051-5.
37. Pereira, F., T. Mitchell, and M. Botvinick, *Machine learning classifiers and fMRI: a tutorial overview.* NeuroImage, 2009. **45**(1 Suppl): p. S199-S209.
38. Craddock, R.C., et al., *Disease state prediction from resting state functional connectivity.* Magnetic Resonance in Medicine, 2009. **62**(6): p. 1619-1628.

39. LaConte, S.M., *Decoding fMRI brain states in real-time.* NeuroImage, 2011. **56**(2): p. 440-454.
40. Guyon, I. and A. Elisseeff, *An introduction to variable and feature selection.* J. Mach. Learn. Res., 2003. **3**: p. 1157-1182.
41. Mantini, D., et al., *Improving MEG source localizations: An automated method for complete artifact removal based on independent component analysis.* NeuroImage, 2008. **40**(1): p. 160-173.
42. Tan, S. and M.L. Mayrovouniotis, *Reducing data dimensionality through optimizing neural network inputs.* AIChE Journal, 1995. **41**(6): p. 1471-1480.
43. Hornung, R., et al., *Full versus incomplete cross-validation: measuring the impact of imperfect separation between training and test sets in prediction error estimation*. 2014.
44. Braga-Neto, U.M. and E.R. Dougherty, *Is cross-validation valid for small-sample microarray classification?* Bioinformatics, 2004. **20**(3): p. 374-80.
45. Haller, S., et al., *Multivariate Pattern Recognition for Diagnosis and Prognosis in Clinical Neuroimaging: State of the Art, Current Challenges and Future Trends.* Brain Topography, 2014. **27**(3): p. 329-337.
46. Orrù, G., et al., *Using Support Vector Machine to identify imaging biomarkers of neurological and psychiatric disease: A critical review.* Neuroscience & Biobehavioral Reviews, 2012. **36**(4): p. 1140-1152.
47. Merriam-Webster. *Merriam-Webster.com*. 2016 [cited 2016 August 22].
48. Dukart, J., et al., *Combined Evaluation of FDG-PET and MRI Improves Detection and Differentiation of Dementia.* PLoS ONE, 2011. **6**(3): p. e18111.
49. Whitehead, A., et al., *Donepezil for the symptomatic treatment of patients with mild to moderate Alzheimer's disease: a meta-analysis of individual patient data from randomised controlled trials.* International Journal of Geriatric Psychiatry, 2004. **19**(7): p. 624-633.
50. Gong, Q., et al., *Prognostic prediction of therapeutic response in depression using high-field MR imaging.* NeuroImage, 2011. **55**(4): p. 1497-1503.
51. Fan, J., et al., *Testing the efficiency and independence of attentional networks.* J Cogn Neurosci, 2002. **14**(3): p. 340-7.
52. Allen, D.N., et al., *Memory and attention profiles in pediatric traumatic brain injury.* Arch Clin Neuropsychol, 2010. **25**(7): p. 618-33.
53. Babikian, T. and R. Asarnow, *Neurocognitive outcomes and recovery after pediatric TBI: meta-analytic review of the literature.* Neuropsychology, 2009. **23**(3): p. 283-96.
54. Ginstfeldt, T. and I. Emanuelson, *An overview of attention deficits after paediatric traumatic brain injury.* Brain Inj, 2010. **24**(10): p. 1123-34.
55. Randolph, C., et al., *Concussion Symptom Inventory: An Empirically Derived Scale for Monitoring Resolution of Symptoms Following Sport-Related Concussion.* Archives of Clinical Neuropsychology, 2009. **24**(3): p. 219-229.
56. McCrory, P., et al., *Consensus statement on concussion in sport: the 4th International Conference on Concussion in Sport held in Zurich, November 2012.* Br J Sports Med, 2013. **47**(5): p. 250-8.
57. McCrea, M., *Standardized Mental Status Testing on the Sideline After Sport-Related Concussion.* Journal of Athletic Training, 2001. **36**(3): p. 274-279.
58. Guskiewicz, K.M., *Assessment of postural stability following sport-related concussion.* Curr Sports Med Rep, 2003. **2**(1): p. 24-30.

59. Maroon, J.C., et al., *Cerebral concussion in athletes: evaluation and neuropsychological testing.* Neurosurgery, 2000. **47**(3): p. 659-69; discussion 669-72.
60. Schatz, P., et al., *Sensitivity and specificity of the ImPACT Test Battery for concussion in athletes.* Arch Clin Neuropsychol, 2006. **21**(1): p. 91-9.
61. Lau, B.C., M.W. Collins, and M.R. Lovell, *Cutoff scores in neurocognitive testing and symptom clusters that predict protracted recovery from concussions in high school athletes.* Neurosurgery, 2012. **70**(2): p. 371-9; discussion 379.
62. Sandel, N.K., et al., *The relationship of symptoms and neurocognitive performance to perceived recovery from sports-related concussion among adolescent athletes.* Appl Neuropsychol Child, 2013. **2**(1): p. 64-9.
63. Keightley, M.L., J.K. Chen, and A. Ptito, *Examining the neural impact of pediatric concussion: a scoping review of multimodal and integrative approaches using functional and structural MRI techniques.* Curr Opin Pediatr, 2012. **24**(6): p. 709-16.
64. Luts, J., et al., *A mixed effects least squares support vector machine model for classification of longitudinal data.* Computational Statistics & Data Analysis, 2012. **56**(3): p. 611-628.
65. Abraham, A., et al., *Machine Learning for Neuroimaging with Scikit-Learn.* Frontiers in Neuroinformatics, 2014. **8**.
66. Pedregosa, F., et al., *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 2011. **12**: p. 2825--2830.
67. Jones, E., T. Oliphant, and P. Peterson, *{SciPy}: Open source scientific tools for {Python}*. 2001.
68. Gramfort, A., et al., *MEG and EEG data analysis with MNE-Python.* Frontiers in Neuroscience, 2013. **7**.
69. Gramfort, A., et al., *MNE software for processing MEG and EEG data.* NeuroImage, 2014. **86**: p. 446-460.
70. Larson, E., et al., *MNE-Python*. 2015: https://github.com/mne-tools/mne-python.
71. Oostenveld, R., et al., *FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data.* Computational Intelligence and Neuroscience, 2011. **2011**: p. 9.