Part III Project

NAME REMOVED
Department of Physics

14th May 2018

Machine Learning Tools for Predicting Cognitive Health

Supervisors:	Contact:
Professor Zoe Kourtzi	zk240@cam.ac.uk
Joseph Giorgio	jjg50@cam.ac.uk
Department of Psychology	Adaptive Brain Lab

Abstract

Predicting cognitive decline using simple to obtain data has a large application in ageing populations, and the earlier a disease is diagnosed, the more effective treatments are. In this project I apply Generalized Matrix Learning Vector Quantization (GMLVQ) classification on a dataset from ADNI, which includes cognitive data (MMSE), structural MRI scans and working memory slope scores collected over 8 years. I aim to classify between healthy and MCI patients, and then within the MCI class between converters to AD and non-converters. A Privileged Information Theoretic Metric Learning approach is taken to attempt to improve the classification accuracy of the cognitive data using the structural data as a privileged feature space. This approach has a potential clinical application as, if the classification accuracy of the widely administered MMSE could be improved upon, it could help to flag up early cognitive decline using a cheap and non-invasive method. Voxel Based Morphometry (VBM) and Partial Least Squares (PLS) regression is used to extract features from the structural MRI scans in the gray and white matter which are predictive of (future) diagnosis. My results show that the Privileged Information approach was unsuccessful at boosting the MMSE score for predicting conversion to AD within a clinically diagnosed MCI population, which is evidence showing the MMSE to be a poor predictive metric. However the gray and white matter did a good job at differentiating between converters and non-converters, achieving 67.2% mean classification accuracy compared to just 52.3% by the cognitive data. I pull out these structural features and the relationships between them in the classifier. Finally I assert based on these findings that more longitudinal studies allowing for prediction within healthy populations, namely converters vs non-converters would be an advancement in this field.

Contents

1 Introduction

In 2017, Alzheimer's Disease (AD) in the United States of America was estimated to be the most expensive disease of any kind, costing $259 billion per year [1]. One in three people born in 2015 will likely develop the disease [2]. The symptoms are short-term memory loss, loss of language, orientation and self-care abilities, with a life expectancy from diagnosis of 3-9 years [3]. AD is a multifactoral disease which results from environmental and genetic factors. Mild Cognitive Impairment (MCI) can bee seen as a stage prior to AD, with a measurable but not significant decline in memory or cognition [4].

While clinical diagnosis and treatment are discrete objectives, they are linked in that our understanding of the multiple causes of AD is still an active and unresolved field, with treatments available which delay the onset of disease rather than cure it. There is a strong emphasis in this field for prediction of cognitive decline since the earlier a diagnosis, the more effective treatments are, and can prolong healthy function for many years [5]. Hence this project is motivated by seeking to predict the onset of cognitive decline by building models which can also be interpreted from a neuroscientific perspective. Predictive modeling using machine learning and big data has become a large and promising field in recent years. However, many models have focused on prediction of clinical diagnosis with the highest possible classification accuracy, often in small samples where over-fitting has meant the models have failed to generalise successfully to other samples.

This project aims to solve some of these issues by

- using a large data set from ADNI [6], collected by multiple scanners
- incorporating longitudinal data by comparing people who convert to Alzheimer's Disease and those whose diagnosis does not change over time
- attempting to move away from simply predicting clinical diagnoses which are unreliable and have been redefined twice since the ADNI data collection began, in 2011 and again in 2018 [7]
- using partial least squares regression to build interpretable, and not over-fit feature spaces
- seeking a potential clinical application by attempting boost predictive accuracy and sensitivity of less invasive feature spaces such as cognitive tests and structural data, by using a highly predictive feature space available from the ADNI database.

I build upon the work of Alahamadi [8] by attempting to boost the predictive accuracy of a simple cognitive test - the MMSE [9] - by incorporating a privileged information theoretic metric learning approach. If successful, this makes mass-scale individual cognitive health risk prediction more viable from a cost and resource perspective. The initial goal of this project was to explore the viability of classifying between healthy and MCI groups using the baseline structural MRI scans, mirroring the work of Alahamadi who did the same with functional MRI data.

Based on the initial results and the available data, the task lent itself naturally to considering AD converters vs non-converters with the same baseline MCI diagnosis, where the same methods were applied. This is more useful as one can classify risk profiles of future cognitive decline from within a pre-Alzheimer's population, based on structural differences which are present pre-diagnosis. I provide evidence that classifying patients with a converter/non-converter framework is far more useful than relying on a clinical diagnosis, backing up work done by Ritter *et al.* [10]. In addition, I attempt to extract structural features within the brain which are predictive of an individual's risk of developing Alzheimer's Disease.

2 Literature Review

Much of the motivation for this project comes from a recent review in Nature Neuroscience [11], which summarises the work done in translational neuroscience, and the potential future direction of the field. The review highlights the need for models with "neuroscientific validity" and simplicity, which in machine learning terms relates to an interpretable feature space, and suitable mathematical models which make sense in terms of existing neurobiological and clinical knowledge.

Many studies have shown that it is possible to achieve excellent classification between healthy controls and patients with AD, however this is no longer an informative or useful task. Prediction within healthy/MCI patients is required to achieve an early diagnosis, and hence administer effective treatment. This is lacking in the literature and requires more interpretation than a computer science classification task to have clinical application. Alahamadi [8] used fMRI to classify between healthy and MCI patients, however the study had a small sample with significant class imbalance (only 9 MCI, 27 healthy controls). The study incorporated privileged information to boost the classification accuracy between control and MCI of a cognitive test from 61.5% to 68.3%, however this was not a commonly applied clinical cognitive test such as the MMSE, which if successfully improved would have a widespread clinical application.

One issue is that any method achieving a very high classification accuracy of clinical diagnosis (say over 90%), for example using a Deep Learning Convolutional Neural Network approach [12], is almost certainly over-fit since the clinical diagnoses they are based off are unreliable. AD can be misdiagnosed as Parkinson's or cerebrovascular disease [13]. Additionally, MCI is diagnosed using a heavy weighting on a patient's memory complaints and MMSE score, which can vary day-to-day or be affected by other conditions such as depression or anxiety, hence is a poor metric for differentiating between MCI and healthy controls [14]. MCI has been shown to be susceptible to false-positive, and potentially false negative diagnoses [15]. Hence, care must be taken when building predictive models which aim to classify class labels from a clinical diagnosis.

Another trend in the field is to employ deep neural networks, especially on the structural image data that is used in this project, such as the work done by Lu *et al.* [16]. While promising, these studies admit that the features found by the deep neural network are difficult or impossible to interpret. Comparable results using a SVM approach with a much more interpretable feature space - 'eigenbrains' generated by Principal Component Analysis (PCA) [17] have been achieved. Partial Least Squares Regression (PLS) has been shown to be more even more effective at extracting discriminating information from structural MRI data [18], generating similar eigenbrains. This approach allows for the validation of the feature extraction method by comparing the areas of interest that are extracted with known regions of the brain associated with matter atrophy in cognitive decline, such as the medial temporal lobe [19] and the hippocampus [20]. In addition it may find other significant areas which may signal the onset of decline.

The analysis of longitudinal data (maintenance vs decline) can give insight into the issues of clinical misdiagnoses, and also help move away from it. The dataset in this project lends itself well to a multivariate diagnostic classification task, such as attempted with SPARE-AD model developed by Davatzikos *et al.* [21]. However this is a fairly speculative model which cannot be tested in a GLMVQ classifier framework. Recent work successfully groups MCI to AD converters, for example Cabral *et al.* 2015 [22]. It is clear that a large data set, with extensive longitudinal patient tracking will be needed, as very few models have been successfully validated out-of-sample [11].

3 Theory

3.1 Feature Generation

3.1.1 Partial Least Squares Regression

Partial Least Squares (PLS) regression [23] [24] is a linear regression technique used for the prediction of one variable from another by finding a multivariate relationship between their two matrices **X** (brain) and **Y** (behaviour). The model seeks to find the orthogonal latent variables - components - in the X-space which which show the maximum covariance in the Y-space. PLS has been successfully applied in neuroimaging [25] as it is well suited to data with a larger number of predictors than observations, which is the case with the **X** and **Y** matrices used in this project. They are defined as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{v}_1^\intercal \\ \mathbf{v}_2^\intercal \\ \vdots \\ \mathbf{v}_n^\intercal \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

Where $\mathbf{v}_i$ are the condensed gray or white matter voxel intensity column vectors for each subject i, and c_i are binary values representing the class of each subject, usually their clinical diagnosis. There are n subjects in the sample.

PLS regression will find the X-score (or component) matrix $\mathbf{T}$ which are projections of $\mathbf{X}$, and can also be used to predict $\mathbf{Y}$.

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\intercal} \tag{1}$$

$$\widehat{\mathbf{Y}} = \mathbf{T}\mathbf{B}\mathbf{C}^{\intercal} \tag{2}$$

Plus associated error terms. $\mathbf{P}$ and $\mathbf{C}$ are the weights (or loadings) matrices, and $\widehat{\mathbf{Y}}$ is the predicted $\mathbf{Y}$ matrix. Refer to Krishnan *et al.* [25] for more details, and to the appendix for the implementation in this project. The PLS can find multiple orthogonal components in X which explain the variance in Y - where the number of components q chosen to be used in the model is a hyperparameter to be optimised.

It is the X weights that are of interest, as they form a vector mask which can be multiplied by test data $\mathbf{X}_{\text{test}}$ to construct a scalar score s^{g}_{qi} for each component q for each subject i, for grey matter g and white matter w. These scores will be used as the feature space in the GMLVQ classifier.

3.1.2 Cross-Validation and Bootstrapping

The statistical method of k-fold cross-validation is used in machine learning to test predictive models, and how they might generalise out of sample [26]. The data sample is split randomly into k sub-samples of equal size, with one of the k sub-samples taken as the test data, and the other $k-1$ sub-samples as the training. The sub-samples contain the same class sizes in the same proportion as the whole data sample.

The hyperparameters - the parameters fixed prior to any learning, for example the learning rate - will be fixed by looping over many candidate permutations using cross-validation within the $k = 1$ training sub-fold. See Figure 1 for the specific implementation. This process can become very computationally expensive, as it scales with $O(n^p)$, where n is the time taken to run a classifier with sufficient iterations over the desired range for a single hyperparameter, and p is the number of hyperparameters to be optimised in this way, hence it is reasonable to optimise within the $k = 1$ fold and fix for all other folds. This approach also prevents overfitting.

Bootsrapping is a statistical method used to gain inference about a larger population using the the data sample available [27]. This is achieved by multiple resamplings with replacement, each yielding their own mean value, from which the variance of the resampled mean can be calculated. This can be applied to other statistics or metrics other than the mean. Here it will be used on the voxel values created from the PLS regression weights in the mask-building process to variance-normalise each voxel, reducing the noise generated by the PLS regression.

The issue of class imbalance is addressed by down-sampling the majority class, to ensure that balanced classes are always used in the analysis. This is appropriate for a large data set with close to balanced classes, which is the case for the ADNI dataset.

3.2 Classification

3.2.1 Generalized Matrix Learning Vector Quantization

A Learning Vector Quantization (LVQ) [28] algorithm is a supervised learning classifier which assigns L prototype vectors $\mathbf{w}_q \in \mathbb{R}^m, q = 1, 2, \ldots, L$ for each of the K different classes $c_q = c(\mathbf{w}_q) \in \{1, \ldots, K\}$ in m dimensional feature space. The model trains using a Hebbian online (sequential), winner-takes-all learning scheme. The positions of the prototypes are updated during the learning phase by moving the prototype closer to a training point of the same class and further away from the closest training point of a different class. The number of prototypes L is a free parameter to be optimised in the model.

The Generalized Matrix Learning Vector Quantization (GMLVQ) [29] algorithm is LVQ with a $m \times m$ positive definite matrix global metric tensor Λ (trace normalised) to generalise the distance measure between training point $\mathbf{x}_i$ and prototype $\mathbf{w}$: $d_\Lambda(\mathbf{x}_i, w) = (\mathbf{x}_i - \mathbf{w})^\intercal \Lambda (\mathbf{x}_i - \mathbf{w})$. $\Lambda = \Omega^\intercal \Omega$, where $\Omega \in \mathbb{R}^{m \times m}$ is a full-rank matrix [30]. Λ, which can be the standard Euclidean metric, defines the parameters which are adapted during training. $\Delta\mathbf{w}$ and $\Delta\Omega$ are adapted, such that the distance from $\mathbf{x}_i$ to $\mathbf{w}$ relies on Λ. The particular details and implementation of this adaptation is detailed in Schneider [31].

In the training phase the algorithm seeks to minimise a cost function f_{GMLVQ}, where the monotonic function ϕ is taken as the identity $\phi(l) = l$:

$$f_{GMLVQ} = \sum_{i=1}^{n} \phi(\mu_\Lambda(\mathbf{x}_i)) \tag{3}$$

There are n subjects in the data set, and the training data is $(\mathbf{x}_i, y_i) \in \mathbb{R}^m, i = 1, 2, \ldots, n$, for m dimensional feature space with y_i representing one of the K different classes, $c_i = c(\mathbf{x}_i) \in \{1, \ldots, K\}$. The function μ_Λ is used to minimise the distance between points of the same class, and maximise the distance between points of a different class:

$$\mu_\Lambda(\mathbf{x}_i) = \frac{d_\Lambda(\mathbf{x}_i, \mathbf{w}^+) - d_\Lambda(\mathbf{x}_i, \mathbf{w}^-)}{d_\Lambda(\mathbf{x}_i, \mathbf{w}^+) + d_\Lambda(\mathbf{x}_i, \mathbf{w}^-)} \tag{4}$$

$d_\Lambda(\mathbf{x}_i, \mathbf{w}^+)$ is the distance from $\mathbf{x}_i$ to the nearest prototype of the same class i.e. where $c_i = c(\mathbf{w}^+) = y_i$, and likewise for $\mathbf{w}^-$, but for the closest prototype of different class label $c(\mathbf{w}^-)$ to y_i.

Using GMLVQ allows for the scaling of pairwise correlations between features, as well as enabling the incorporation of the privileged information theoretic metric learning model.

3.2.2 Privileged Information Theoretic Metric Learning

The idea of privileged information learning [30] is that the GMLVQ classifier is applied in a new privileged feature space $\mathbb{X}^*$ yielding another metric tensor Λ^*, in addition to Λ in the original space $\mathbb{X}$. This tensor in the privileged space $\mathbb{X}^*$ is used to learn a new metric Λ_{new} in $\mathbb{X}$. The training points are now $(\mathbf{x}_i, \mathbf{x}_i^*, y_i) : \mathbf{x}_i \in \mathbb{X}, \mathbf{x}_i^* \in \mathbb{X}^*, i = 1, 2, \ldots, N$. The squared distances in the original and privileged spaces respectively are defined below:

$$d_\Lambda(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\intercal \Lambda (\mathbf{x}_i - \mathbf{x}_j) \ \ , \ \ d_{\Lambda^*}(\mathbf{x}_i^*, \mathbf{x}_j^*) = (\mathbf{x}_i^* - \mathbf{x}_j^*)^\intercal \Lambda^* (\mathbf{x}_i^* - \mathbf{x}_j^*) \tag{5}$$

In addition, the distance over the new metric Λ_{new} is defined in the original space as: $d_{\Lambda_{\text{new}}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\intercal \Lambda_{\text{new}} (\mathbf{x}_i - \mathbf{x}_j)$, where Λ_{new} is to be learned. By considering a threshold percentage (as a hyperparameter to be optimised) of all pairwise squared distances $d_{\Lambda^*}(\mathbf{x}_i^*, \mathbf{x}_j^*)$, u^* is defined as the upper bound for the squared distance between a similar pair in the privileged space, and l^* as the lower bound for the squared distance between a dissimilar pair in the privileged space, the following construction is used to label points as similar S_+ or dissimilar S_-:

- If $d_{\Lambda^*}(\mathbf{x}_i^*, \mathbf{x}_j^*) \leq u^*$ and the two points have the same class label $c(x_i) = c(x_j) = y_i$ then $(x_i, x_j) \in S_+$.
- If $d_{\Lambda^*}(\mathbf{x}_i^*, \mathbf{x}_j^*) \geq l^*$ and the two points have a different class label $c(x_i) \neq c(x_j) \neq y_i$ then $(x_i, x_j) \in S_-$.

Λ_{new} is optimised such that $d_{\Lambda_{\text{new}}}(\mathbf{x}_i, \mathbf{x}_j)$ is shrunk if $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ are similar, or enlarged if not. This process is fully detailed in Fouad *et al.* 2013 [30]. The implementation is detailed in the script privscript.m which was adapted from the library provided by Joseph Giorgio.

4 Methods

The implementation of the methods can be found in the MATLAB code in the appendix.

4.1 Data

The data used came from the Alzheimer's Disease Neuroimaging Initiative (ADNI) [6]. While the data was collected from different scanning sites, the protocol used to calibrate, collect and process data was strict to ensure consistency in the sample [32]. The raw MRI scans are 50MB NIfTI (.nii) files, which are 3-D images composed of a few million voxels (3-D pixels) each with a gray-scale intensity value. They come from 3T scanners with MP RAGE sampling.

For this project only the baseline scans were used as structural data, with longitudinal information available for each of the 600 subjects, for example the progression of their clinical diagnoses and memory scores for the years after their initial baseline diagnosis. This simplifies the task, but also proves useful as none of the subjects were diagnosed AD at the time of scanning, hence focusing the framework for predicting decline from the stage of cognitively healthy or MCI. Classifying AD from healthy patients using structural data has been extensively and successfully attempted in the literature, and is not part of this project. At each time point each patient has a diagnosis that is one of [Healthy Control (HC), Early MCI (EMCI), Late MCI (LMCI), Alzheimer's Disease (AD)]. For baseline scans, the average age is 72.7 ± 7.34 years. No baseline patients had AD, but many converted over the 5-8 year time window the data was taken over.

Data to be incorporated into the first feature space is the simple to collect, cognitive data, which is less highly predictive [33]: age, gender, MMSE, years of education. The structural data is used to construct scores for gray s^g_{qi} and white matter s^w_{qi} from the first q orthogonal PLS components, which forms an intermediate space - more predictive than the cognitive data, but harder to obtain. Additionally, the change in working memory scores are available as well as genetic and PET beta amyloid data, which are all highly predictive metrics.

4.2 Voxel-Based Morphometry

The Method of Voxel-Based Morphometry (VBM) was used to process the structural MRI scans. VBM is an analysis technique which allows for the comparison in brain anatomy between different brains, by normalising each brain to MNI space (a 'standard brain' from the Montreal Neurological Institute [34]). The VBM processing was performed in the SPM12 software package [35], which runs in MATLAB. The VBM methodolgy template outlined by Ashburner [36] was followed exactly to ensure consistency and to make sure that the same processing can be carried out on other data sets. The script was written and tested using scans from the IXI dataset [37] before being run over the ADNI scans. The process is outlined in the following steps:

1. Pre-process images to ensure correct formatting and spatial orientation. Remove or replace corrupted files.
2. Segment images into gray and white matter files. This is done by fitting suitable intensity ranges to match the voxel intensity. For specific details refer to Ashburner [36].
3. Use the Dartel package in SPM12 to find the nonlinear deformations to warp and match the newly formed gray and white matter images.

4. Normalise the images to MNI space. Generate smoothed warped gray and white matter images. The smoothing process involves replacing the intensity of each voxel by the average of the surrounding voxels. The smoothed images represent regional tissue volumes for each brain. 3mm x 3mm x 3mm Gaussian FWHM smoothing was chosen to reduce the variance across subjects and hence increase the sensitivity to detect changes, without losing significant structural details.

The major advantage of VBM is its use in analysing a large data set. For small numbers of scans, it has proven to be effective when compared to a radiologist's visual assessment [38], however for the large dataset VBM is the most appropriate method of data analysis which allows for feature extraction from gray and white matter density.

4.3 Feature Generation

4.3.1 Creating Masked Vectors

Once the VBM processed gray and white matter images for each patient have been obtained, the 3-D NIfTI files were converted into 1-D vectors. Since these vectors have over 2×10^6 voxel intensity values, the areas not in the gray or white matter were set to zero using a standard mask created in SPM12. All zero intensity values are then removed resulting in a much smaller, condensed vector $\mathbf{v}_i$ of around 1×10^5 voxels. This process is reversible so the 3-D images can be reconstructed by placing the condensed vector values in the non-zero mask positions, and reshaping back to 3-D.

The vector is condensed to reduce computational time and to prevent the PLS trying to fit noise in areas that are not relevant.

4.3.2 Incorporating the PLS into the Cross-Validation

The PLS must be run within the training data of the k^{th} crossfold only since it sees the class label, so must be kept separate from the test data when building the weighted mask. For each test subject, and for each component q_g of gray matter and q_w of white matter, a scalar score is generated by multiplying the voxel intensity vectors by the mask (weights $\mathbf{P}$), and summing the values (dot product). The threshold cutoff of the mask was optimised as a hyperparameter, and the fixed value resulted in each score being generated by approximately 1000 voxel values, which corresponds to small, isolated regions which could be viewed in 3-D using imaging software (see results for examples of these maps).

The PLS regression was applied using the inbuilt MATLAB function 'plsregress' [39].

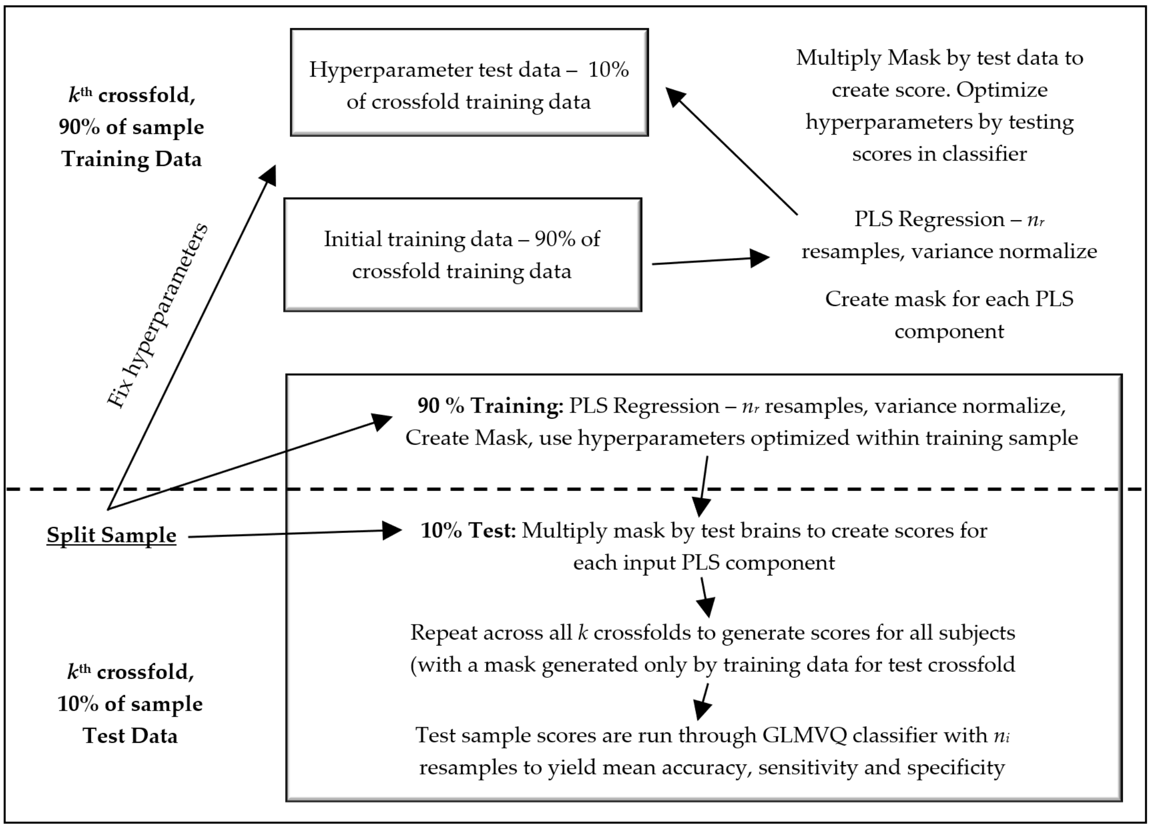

kth crossfold, 90% of sample Training Data
Hyperparameter test data – 10% of crossfold training data
Multiply Mask by test data to create score. Optimize hyperparameters by testing scores in classifier
Fix hyperparameters
Initial training data – 90% of crossfold training data
PLS Regression – nr resamples, variance normalize
Create mask for each PLS component
90 % Training: PLS Regression – nr resamples, variance normalize, Create Mask, use hyperparameters optimized within training sample
Split Sample
10% Test: Multiply mask by test brains to create scores for each input PLS component
Repeat across all k crossfolds to generate scores for all subjects (with a mask generated only by training data for test crossfold
kth crossfold, 10% of sample Test Data
Test sample scores are run through GLMVQ classifier with ni resamples to yield mean accuracy, sensitivity and specificity

Figure 1: Pipeline of k-crossfold validation method used for feature generation, hyperparameter optimization and to test the GLMVQ classifier. Note, the hyperparameters were fixed in the $k = 1$ training sample.

4.4 Classification

4.4.1 Testing the GMLVQ Classifier and Fixing Hyperparameters

The PLS regression was run within each crossfold over 1000 resamples (bootsrapping). It was checked that the components did not flip by comparing correlations, so the mean values of the weights generated for each component q could be taken and then variance normalised to reduce noise.

Additionally, the percentage of variance in the Y-space explained by the PLS - PCTVAR - was reconstructed in the test sample by multiplying the X scores by the test data, correlating to the Y scores (class) and comparing to the PCTVAR in the training data. While a correlation is not the most reliable means to test if a score will classify well, by comparison with test/training correlations insight could be gained into how well the model built using the training data would generalise to the test data. After this initial test phase, it was concluded that no more than $q = 3$ components would be needed in the PLS for each of gray and white matter - higher components explained considerably less variance and more noise. The number to be used in the classifier was set in the optimisation phase in the $k = 1$ crossfold - and was found to be $q_g = 1, 2, 3$ for gray matter and $q_w = 1$ for white matter.

The other hyperparameters fixed were the learning rate, the number of training epochs, the threshold value of the mask (a trade-off between retaining information and removing noise) and the number of prototypes L. See Figure 1 for the method pipeline.

Initially the classifier was tested with balanced classes by using cognitive data (MMSE, age, education, sex) M-CD to differentiate between healthy controls (HC) and MCI (EMCI and LMCI grouped). Then using the same classes, it was tested using the structural data M-SD, with the hope that it would perform better.

M-SD for HC vs MCI performed worse than expected, so the subjects were reclassified as MCI converters (to AD) MCI-C vs non-converters (stay MCI) MCI-S by looking at their clinical diagnosis over the next 5-8 years (depending on the data available) from baseline. This reduced the class sizes significantly, from 300 per class to 83, however it was a way to incorporate the longitudinal data, and to tackle the single clinical mis-diagnosis problem. The downside is that this method may have missed some converters, who would convert to AD some time after final clinical diagnosis was collated.

All classification accuracy results stated were obtained by using a Macroaveraged Mean Absolute Error, MMAE [8] method. It is a weighted sum of the classification errors across classes. Classification accuracy = 1 minus the error. These numbers were obtained over 100 resamples of the data in the classifier, tested for all k crossfolds.

4.4.2 Incorporating The Privileged Information Theoretic Metric Learning

Once the classifier has been tested for different spaces and baseline classification accuracies obtained, the features can be incorporated into the privileged ITML model. The hyperparameters were fixed in exactly the same way as before (outlined in Figure 1), but there are a few additional hyperparameters - notably the slack parameter - to be optimised.

5 Results

The initial preliminary tests of the classifier were run with the classes HC vs MCI. However I noticed that in some of the crossfolds the classifier was failing almost completely to differentiate between the HC and MCI subjects, particularly those subjects who were diagnosed EMCI and did not convert to LMCI or AD. Hence, it is more informative to compare the HC subjects against MCI-C and MCI-S, as this data was available:

Sensitivity (true positive rate) is defined as (true positives)/(false negatives + true positives), for example a sensitivity of 1.000 would mean all people who converted to AD were correctly identified. Specificity (true negative rate) is defined as (true negatives)/(false positives + true negatives), so a higher specificity would mean that more non-converters were correctly classified.

5.1 HC vs MCI

Model	Mean Acc	Std-Dev	Sensitivity	Specificity
M-CD	0.661	0.125	0.578	0.744
M-SD	0.753	0.108	0.742	0.764

Table 1: Classification performance table comparing mean classification accuracy (Mean Acc) of the two classes Healthy Control (HC) and MCI-Converters (MCI-C) (convert to AD), using different data. Balanced classes of size 100 HC and 100 MCI-C.

Model	Mean Acc	Std-Dev	Sensitivity	Specificity
M-CD	0.611	0.131	0.577	0.646
M-SD	0.584	0.144	0.532	0.635

Table 2: Classification performance table comparing mean classification accuracy (Mean Acc) of the two classes Healthy Control (HC) and MCI-Stay (MCI-S), using different data. Balanced classes of size 100 HC and 100 MCI-S.

It is notable from Table 1 and Table 2 how the structural data is far more informative when it comes to differentiating between converters. The classifier struggles to differentiate between those diagnosed as healthy and those who do not convert to AD within 5-8 years, suggesting that there are likely a number of misdiagnosed people in these two classes, which are probably much closer together than the HC/MCI-C/AD classes.

5.2 EMCI vs LMCI

Next, I tested using the newly defined converter classes, how the classifier performed looking between EMCI and LMCI, converters vs non-converters:

Model	Mean Acc	Std-Dev	Sensitivity	Specificity
M-CD	0.605	0.134	0.600	0.610
M-SD	0.660	0.142	0.645	0.675

Table 3: Classification performance table comparing mean classification accuracy (Mean Acc) of the two classes EMCI-Stay (EMCI-S) (non-converters) and EMCI-Converters (MCI-C) (convert to AD), using different data. Balanced classes of size 50 EMCI-S and 50 EMCI-C.

Model	Mean Acc	Std-Dev	Sensitivity	Specificity
M-CD	0.591	0.109	0.600	0.616
M-SD	0.684	0.126	0.734	0.634

Table 4: Classification performance table comparing mean classification accuracy (Mean Acc) of the two classes LMCI-Stay (LMCI-S) (non-converters) and LMCI-Converters (MCI-C) (convert to AD), using different data. Balanced classes of size 50 LMCI-S and 50 LMCI-C.

Model	Mean Acc	Std-Dev	Sensitivity	Specificity
M-CD	0.479	0.168	0.554	0.488
M-SD	0.507	0.147	0.500	0.486

Table 5: Classification performance table comparing mean classification accuracy (Mean Acc) of the two classes EMCI-Stay (EMCI-S) and LMCI-Stay (LMCI-S), using different data. Balanced classes of size 50 EMCI-S and 50 LMCI-S.

The results displayed in Tables 3-5 highlight how the classifier struggled separating EMCI and LMCI, and it is much more informative to consider converters vs. stayers.

5.3 MCI-S vs MCI-C

The most important, useful, and best candidate for boosting using the privileged information approach, was to consider MCI-S vs MCI-C. This section also includes images of the structural features, and the Λ matrices.

Model	Mean Acc	Std-Dev	Median	Sensitivity	Specificity
M-CD	0.523	0.113	0.500	0.498	0.548
M-SD	0.672	0.131	0.688	0.727	0.617
M-MEM	0.763	0.103	0.781	0.744	0.781

Table 6: Classification performance table comparing mean classification accuracy (Mean Acc) of the two classes MCI-Stay (MCI-S) (non-converters) and MCI-Converters (MCI-C) (convert to AD), using different data. Balanced classes of size 83 MCI-S and 83 MCI-C. Figures 2-6 outline these results in more detail, and show the structural features used in the M-SD classification.

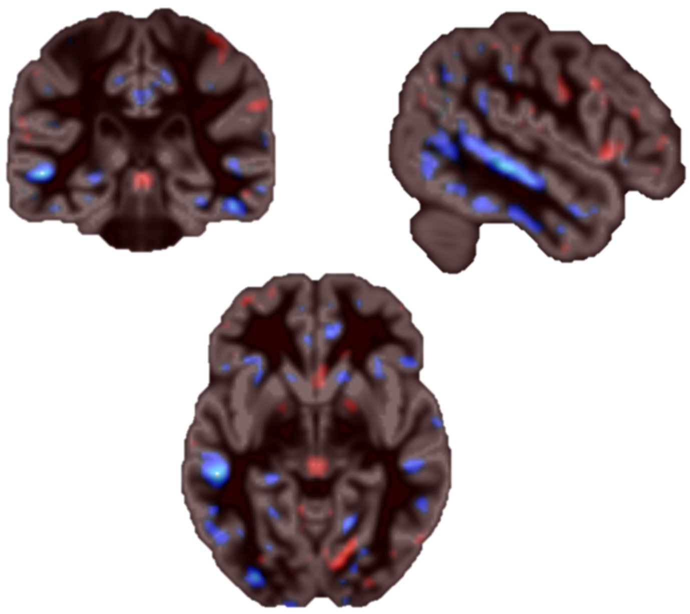


Figure 2: 3 plane slices of statistical map at the 95th percentile through the MNI brain for gray matter component 1. Blue = negative loading with conversion to AD (atrophy), Red = positive loading. Note the strong negative loading in the temporal lobes, a well known atrophic region associated with the onset of AD.

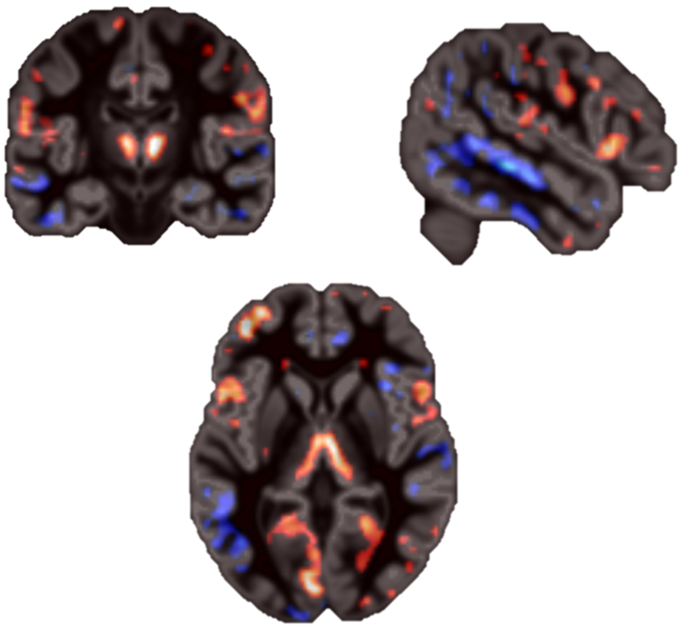


Figure 3: 3 plane slices of statistical map at the 95th percentile through the MNI brain for gray matter component 2. Blue = negative loading with conversion to AD (atrophy), Red = positive loading. Note positive loadings in the visual cortex and brain stem, as well as the left dorsolateral prefrontal cortex. Negative loadings are seen in the temporal lobes as with the first component.

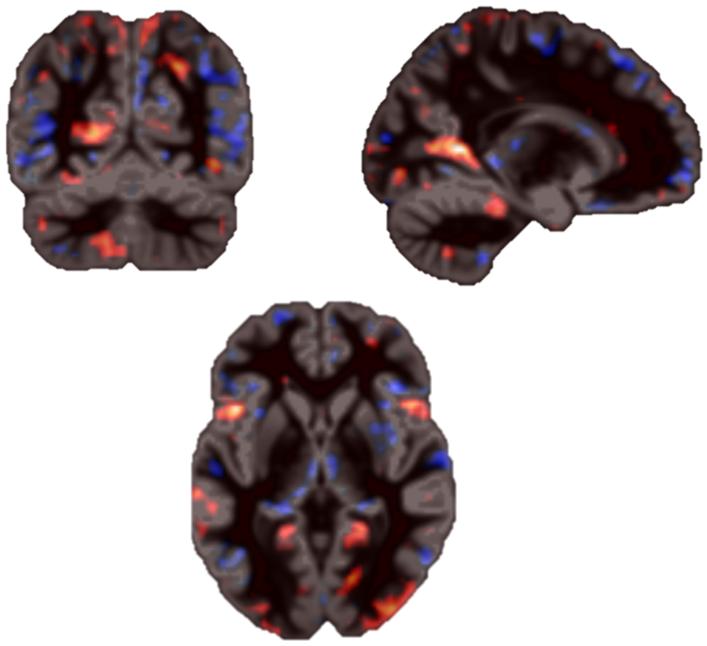
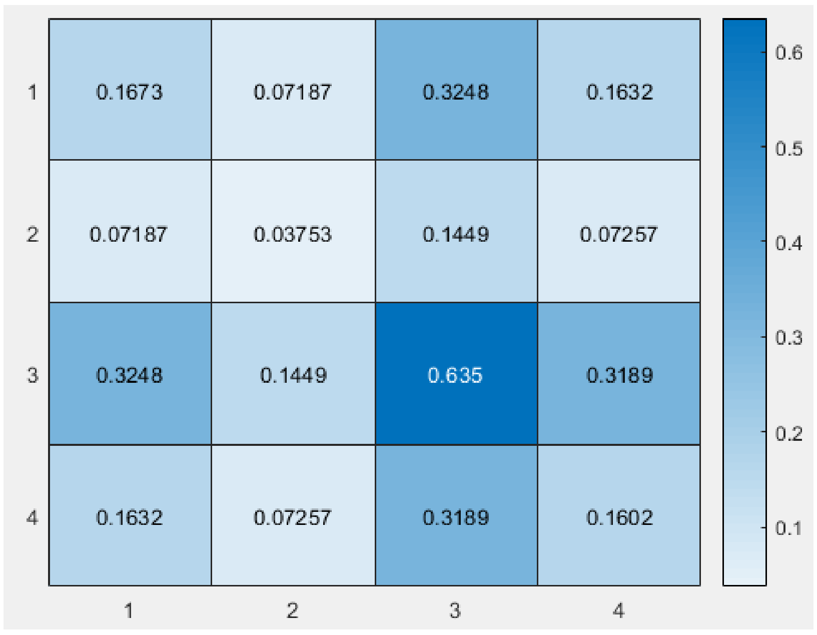


Figure 4: 3 plane slices of statistical map at the 95th percentile through the MNI brain for gray matter component 3. Blue = negative loading with conversion to AD (atrophy), Red = positive loading. Note the strong loading in the visual cortex.



	1	2	3	4
1	0.1673	0.07187	0.3248	0.1632
2	0.07187	0.03753	0.1449	0.07257
3	0.3248	0.1449	0.635	0.3189
4	0.1632	0.07257	0.3189	0.1602

Figure 5: Λ matrix showing cross-terms for features 1 = first component of white matter, 2,3,4 = first, second and third components of gray matter respectively for MCI-S vs MCI-C. The heaviest weighting is on component 2 in the gray matter

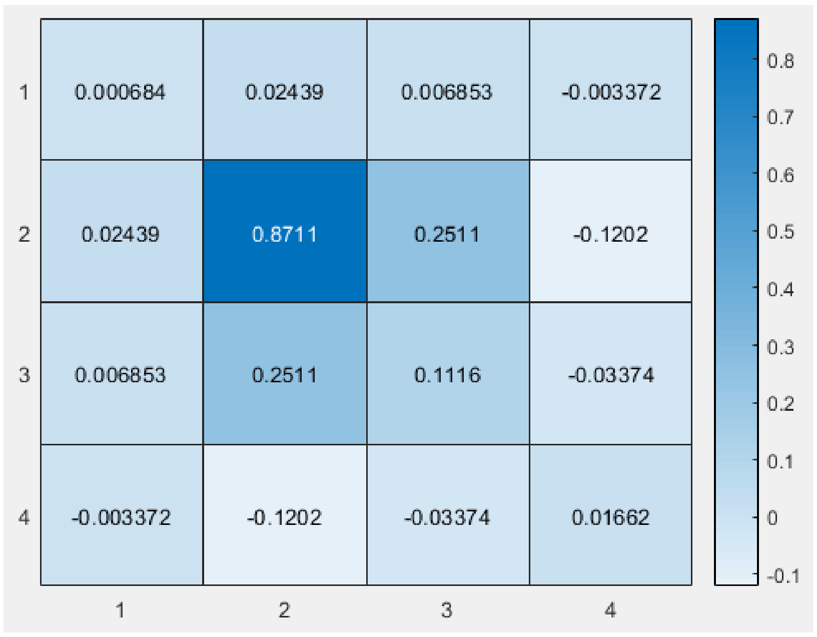


Figure 6: Λ matrix showing cross-terms for features 1 = Age, 2 = MMSE, 3 = Sex, 4 = Years of Education. Very heavily weighted on the MMSE, however the classifier still struggled to identify AD converters using this data.

5.4 MCI-S vs MCI-C - Using Privileged Information Theoretic Metric Learning

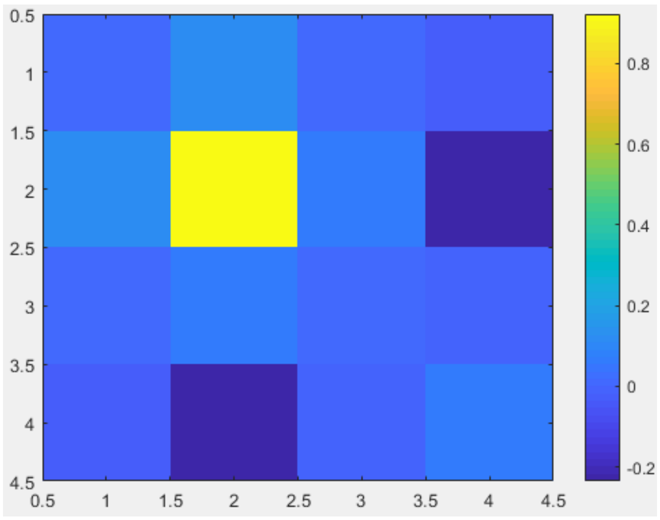


Figure 7: Λ_{new} matrix showing cross-terms for features 1 = Age, 2 = MMSE, 3 = Sex, 4 = Years of Education. There has been a slight pull on the MMSE/Education interaction, however the Λ matrix has been largely unchanged by the structural data as privileged information.

After some time spend fixing hyperparameters, using the structural data as privileged information the PITML classifier was unable to improve the baseline classification of M-CD which was 0.523.

6 Discussion

It is interesting to note that when EMCI and LMCI were grouped into MCI, there cognitive data did considerably worse at differentiating between converters and non-converters. This is actually unsurprising because the MMSE does a better job at differentiating between EMCI and LMCI than the converters - mainly because the diagnosis of EMCI or LMCI was based heavily off this single score.

While this project was unsuccessful in using the privileged information, it is not a surprising result. The MMSE has been shown in the literature to be a poor predictor of dementia [40], and the privileged information approach cannot learn information which is not there to learn.

The cognitive data is only boostable if there is information which can be learned in the Λ matrix's cross term interaction of MMSE with sex, age or education, the latter being the most likely candidate. This information was not learned, and it may not exist. The MMSE has been shown to be a poor predictor of decline/conversion compared to other cognitive test e.g. the SAGE [45]. The best hope for this to be successful was a cross-term interaction between education and possibly age or sex, however whether these are linked to risk of Alzheimer's is speculative [41].

The regions extracted by the PLS, as shown in Figures 2-4, validate the method well. Expected features, namely the temporal lobes [19], visual cortex [42], brain stem [43] and dorsolateral prefrontal cortex [44] are found. Beyond that, it is hard to speculate what is noise and what is information. To build a robust mask, an enormous population may be helpful. One disadvantage with the PLS is that it could pull out significantly different features which each classify just as well, or explain just as much variance in the Y space. While the maps allow for a sanity check, they are likely infinite combinations of voxels which are just as predictive.

7 Conclusion

I have attempted to boost the predictive use of the MMSE test for MCI patients at risk of converting to AD using structural MRI data as privileged information. This was unsuccessful, compared to the Alahmadi [8] study which used fMRI on cognitive skill tests including working memory, cognitive inhibition and attentional skills. It is likely that the cognitive data used was far superior to the MMSE, which is a single score from 1-30. By failing to change the cognitive data metric using structural data, I provide evidence that the MMSE is a very poor test to indicate the risk of AD conversion. This backs up the literature which has shown that other tests such as the SAGE are superior predictors [45].

Using structural scans, PLS regression and k-fold cross-validation I have been able to predict with a mean classification accuracy of 67.2% whether a patient diagnosed with MCI will progress to AD within 5-8 years. This is significantly better than the MMSE score, which achieved just 52.3%. Although the structural data did not boost the cognitive accuracy using a privileged information theoretic metric learning approach, it could prove useful in it's own right if validated out of sample. Structural MRI scans are becoming increasingly easy to obtain, and the method outlined in this project can be applied to data from a 1.5T or 3T scanner.

I have picked out structural features for people with the same clinical diagnosis of MCI which may be indicative of progression to AD within 5-8 years. The next step is to test if these features can predict decline out of sample, or at least to test using the same methodology whether another sample with longitudinal data produces similar maps with the same features of significance in the gray and white matter.

I have provided evidence that EMCI and LMCI are inappropriate labels, which has very recently been addressed by Jack *et al.*, 2018 [7].

The next stage for this work would be to attempt to boost the structural data using the memory slope as the privileged information. A nice idea could be to take a step back and use the memory slope as the $\mathbf{Y}$ matrix in the PLS regression, hence moving completely away from the clinical diagnosis and relying on a highly predictive metric, and the features it extracts in the structural data. The issue is that this is harder to classify as the problem is now framed as continuous rather than discrete. This makes more sense in terms of disease progression, but will require the mathematics of ordinal classification problem, potentially becoming harder to validate.

Another class for which there was insufficient data in the ADNI to address is the HC-C (healthy control, converting to Alzheimer's disease). It would be a more difficult task to attempt to classify a healthy person's risk of developing AD, and would require longitudinal data taken over many years. It is clear that the future of this field will require even more extensive longitudinal studies than the ADNI, tracking people over say 20 years from healthy to healthy/sick, and hopefully this project has provided evidence that this would be a worthwhile investment, to enable the development of more advanced predictive models.

8 References

[1] Alzheimer's Association, 2017.
Costs of Alzheimer's to Medicare and Medicaid
`http://act.alz.org/site/DocServer/2012_Costs_Fact_Sheet_version_2.pdf?docID=7161`

[2] Alzheimer's Research UK
One in three people born in 2015 will develop dementia
`https://www.alzheimersresearchuk.org/one-in-three-2015-develop-dementia/`

[3] Querfurth, LaFerla, 2010. The New England Journal of Medicine
Alzheimer's Disease
`https://www.nejm.org/doi/full/10.1056/NEJMra0909142`

[4] Alzheimer's Association
Mild Cognitive Impairment
`https://www.alz.org/dementia/mild-cognitive-impairment-mci.asp`

[5] Medical Care Corporation, 2004.
Delaying the Onset and Progression of Alzheimer's Disease
`http://www.preventad.com/pdf/support/article/DelayingADProgression.pdf`

[6] Alzheimer's Disease Neuroimaging Initiative Data Samples
`http://adni.loni.usc.edu/data-samples/`

[7] Jack *et al.*, 2018. National Institute on Aging—Alzheimer's Association
NIA-AA Research Framework: Toward a biological definition of Alzheimer's disease
`https://www.ncbi.nlm.nih.gov/pubmed/29653606`

[8] Alahmadi *et al.*, 2016. Frontiers in Computational Neuroscience
Classifying Cognitive Profiles Using Machine Learning with Privileged Information in Mild Cognitive Impairment
`https://www.frontiersin.org/articles/10.3389/fncom.2016.00117/full`

[9] Oxford Medical Education
Mini-Mental State Examination (MMSE)
`http://www.oxfordmedicaleducation.com/geriatrics/mini-mental-state-examination-mmse/`

[10] Ritter *et al.*, 2015. Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring
Multimodal prediction of conversion to Alzheimer's disease based on incomplete biomarkers
`https://www.sciencedirect.com/science/article/pii/S2352872915000408`

[11] Woo *et al.*, 2017. Nature Neuroscience
Building better biomarkers: brain models in translational neuroimaging
`https://www.ncbi.nlm.nih.gov/pubmed/28230847`

[12] Sarraf and Tofighi, 2016.
Classification of Alzheimer's Disease Structural MRI Data by Deep Learning Convolutional Neural Networks
`https://arxiv.org/ftp/arxiv/papers/1607/1607.06583.pdf`

[13] Klatka *et al.*, 1996.
Incorrect diagnosis of Alzheimer's disease. A clinicopathologic study.
`https://www.ncbi.nlm.nih.gov/pubmed/8599556`

[14] Clark *et al.*, 1999.
Variability in Annual Mini-Mental State Examination Score in Patients With Probable Alzheimer Disease
`https://jamanetwork.com/journals/jamaneurology/fullarticle/775209`

[15] Edmonds *et al.*, 2016.
"Missed" Mild Cognitive Impairment: High False-Negative Error Rate Based on Conventional Diag-

nostic Criteria
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4879874/

[16] Lu *et al.*, 2018. Nature
Multimodal and Multiscale Deep Neural Networks for the Early Diagnosis of Alzheimer's Disease using structural MR and FDG-PET images
https://www.nature.com/articles/s41598-018-22871-z

[17] Zhang *et al.*, 2015. Frontiers in Computational Neuroscience
Detection of subjects and brain regions related to Alzheimer's disease using 3D MRI scans based on eigenbrain and machine learning
https://www.frontiersin.org/articles/10.3389/fncom.2015.00066/full

[18] Khedher *et al.*, 2014. Neurocomputing
Early diagnosis of Alzheimer's disease based on partial least squares, principal component analysis and support vector machine using segmented MRI images
https://pdfs.semanticscholar.org/0466/5bb876c9955d536b48ba0b4607d123a214a4.pdf

[19] Visser *et al.*, 2002. J Neurol Neurosurg Psychiatry
Medial temporal lobe atrophy predicts Alzheimer's disease in patients with minor cognitive impairment
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1737837/

[20] Mu, Gage, 2011. Mol Neurodegener
Adult hippocampal neurogenesis and its role in Alzheimer's disease.
https://www.ncbi.nlm.nih.gov/pubmed/22192775

[21] Davatzikos *et al.*, 2009. Brain *Longitudinal progression of Alzheimer's-like patterns of atrophy in normal older adults: the SPARE-AD index*
https://academic.oup.com/brain/article/132/8/2026/266984

[22] Cabral *et al.*, 2015. Computers in Biology and Medicine *Predicting conversion from MCI to AD with FDG-PET brain images at different prodromal stages*
https://www.sciencedirect.com/science/article/pii/S0010482515000062?via\%3Dihub

[23] Wold, 1984. SIAM J. Sci. and Stat. Comput.
The Collinearity Problem in Linear Regression. The Partial Least Squares (PLS) Approach to Generalized Inverses
https://epubs.siam.org/doi/abs/10.1137/0905052

[24] Abdi, 2003. Encyclopedia of Social Sciences Research Methods.
Partial Least Squares (PLS) Regression.
http://www.camo.com/resources/casestudies/Partial_Least_Squares_Regression..pdf

[25] Krishnan *et al.*, 2011. NeuroImage
Partial Least Squares (PLS) methods for neuroimaging: A tutorial and review
https://www.sciencedirect.com/science/article/pii/S1053811910010074

[26] Kohavi, 1995. International Joint Conference on Artificial Intelligence
A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.529

[27] Efron and Tibshirani, 1993. Chapman & Hall/CRC.
An Introduction to the Bootstrap
http://cds.cern.ch/record/526679/files/0412042312_TOC.pdf

[28] Kohonen, 1990.
Improved Versions of Vector Quantization
`https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5726582`

[29] Schneider *et al.*, 2010.
Regularization in Matrix Relevance Learning
`http://www.cs.rug.nl/~biehl/Preprints/tnn-2010.pdf`

[30] Fouad *et al.*, 2013. IEEE Transactions on Neural Networks and Learning Systems
Incorporating Privileged Information Through Metric Learning
`https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6488857`

[31] Schneider *et al.* 2009. Neural Computation
Adaptive Relevance Matrices in Learning Vector Quantization
`https://www.rug.nl/research/portal/files/2688271/2009NeurCompSchneider2.pd`

[32] ADNI Mehods & Tools
`http://adni.loni.usc.edu/methods/`

[33] ADNI Background & Rationale
`http://adni.loni.usc.edu/study-design/background-rationale/`

[34] The MNI brain and the Talairach atlas - MRC University of Cambridge
`http://imaging.mrc-cbu.cam.ac.uk/imaging/MniTalairach`

[35] SPM12, The Wellcome Trust Centre for Neuroimaging, University College London
`http://www.fil.ion.ucl.ac.uk/spm/software/spm12/`

[36] Ashburner, 2015.
VBM Tutorial
`http://www.fil.ion.ucl.ac.uk/~john/misc/VBMclass15.pdf`

[37] IXI Dataset
`http://brain-development.org/ixi-dataset/`

[38] Whitwell, 2009. Journal of Neuroscience
Voxel-Based Morphometry: An Automated Technique for Assessing Structural Changes in the Brain
`http://www.jneurosci.org/content/29/31/9661`

[39] MathWorks plsregress
`https://uk.mathworks.com/help/stats/plsregress.html`

[40] Raiha *et al.*, 2001. Scand J Prim Health Care
Poor performance in the mini-mental state examination due to causes other than dementia
`https://www.ncbi.nlm.nih.gov/pubmed/11303545`

[41] Letenneur *et al.*, 2000. Am J Epidemiol
Education and the risk for Alzheimer's disease: sex makes a difference
`https://www.ncbi.nlm.nih.gov/pubmed/10873130`

[42] Brewer, Barton, 2014. Front Psychol
Visual cortex in aging and Alzheimer's disease: changes in visual field maps and population receptive fields
`https://www.ncbi.nlm.nih.gov/pubmed/24570669`

[43] Simic *et al.* 2009. Neuropathol Appl Neurobiol
Does Alzheimer's disease begin in the brainstem?
`https://www.ncbi.nlm.nih.gov/pubmed/19682326`

[44] Kumar *et al.*, 2017. JAMA Psychiatry
Extent of Dorsolateral Prefrontal Cortex Plasticity and Its Association With Working Memory in Patients With Alzheimer Disease.
`https://www.ncbi.nlm.nih.gov/pubmed/29071355`

[45] Scharre *et al.*, 2015. Alzheimer's & dementia: the journal of the Alzheimer's Association
Longitudinal changes in self-administered gerocognitive examination (SAGE) and mini-mental state exam (MMSE) scores for subjective cognitive impairment (SCI), mild cognitive impairment (MCI), dementia converters, and Alzheimer's disease (AD) patients
`https://www.researchgate.net/publication/287375375_Longitudinal_changes_in_self-administered_gerocognitive_examination_SAGE_and_mini-mental_state_exam_MMSE_scores_for_subjective_cognitive_impairment_SCI_mild_cognitive_impairment_MCI_dementia_converter`

9 Appendices

The main script is included below: mainscript.m. This is my own work. In addition, a USB stick was provided which contained the function train_gmlvq_cv.m, which was adapted alongside the script privscript.m from the library of Joseph Giorgio. The VBM code I worked on for the pre-processing was also included.

The statistical brain maps were made using the software MRIcron, and the other graphs were made in MATLAB.

```
load('T:\Users\jjg50\Jan\ADNI\multiclass\classdata6.mat');

%% (1) Import nii Files
rids = classdata5(:,2);
for subject_id = 1:length(rids); %run time 10-20mins
    atemp = num2str(rids(subject_id));

    %white matter
%     chold1 = dir(['T:\Users\jjg50\Jan\ADNI\rc2300\smwrc2', atemp, '*']);
%     c1 = ['T:\Users\jjg50\Jan\ADNI\rc2300\', chold1.name,''];
%     in_aa1=load_untouch_nii(c1)
%     gm_image_exp1{subject_id}=double(in_aa1.img).*in_aa1.hdr.dime.scl_slope;
%     vector_raw1(subject_id,:) = reshape(imresize3(gm_image_exp1{1,subject_id},0.8)
    ,[1,(97*116*97)]);
%     gm_image_exp1 = [];
%
    %grey matter
    chold1 = dir(['T:\Users\jjg50\Jan\ADNI\rc1300\smwrc1', atemp, '*']);
    c1 = ['T:\Users\jjg50\Jan\ADNI\rc1300\', chold1.name,''];
    in_aa1=load_untouch_nii(c1)
    gm_image_exp1{subject_id}=double(in_aa1.img).*in_aa1.hdr.dime.scl_slope;
    vector_raw1(subject_id,:) = reshape(imresize3(gm_image_exp1{1,subject_id},0.8)
        ,[1,(97*116*97)]);
    gm_image_exp1 = [];
end

%% (2) Resize nii voxels

mask = load_untouch_nii('T:\Users\jjg50\Jan\ADNI\MASK\wmmaskgood.nii'); %< WM
mask_img = double(mask.img);
mask_img2 = imresize3(mask_img, 0.8);
maskgmvector2 = reshape(mask_img2, [1, (97*116*97)]);
mask1 = load_untouch_nii('T:\Users\jjg50\Jan\ADNI\MASK\mask.nii'); %< GM
mask_img1 = double(mask1.img);
mask_img21 = imresize3(mask_img1, 0.8);
maskgmvector21 = reshape(mask_img21, [1, (97*116*97)]);

maskgmvector = maskgmvector21;
vector_raw = vector_raw1;

%% (3) Pack
positions = find(maskgmvector2);
for i = 1:length(rids)
    condensed_vec_raw(i,:) = vector_raw(i,positions); %< raw vector*mask
    i
```

```
end


%% (4) Crossfold/Bootstrap PLS Regression

class1 = classdata5(:,1);
class = class1;
% c1alle = find(class==1);
% c2alle = find(class==2);
% c3alle = find(class==3);
% c4alle = find(class==4);
% c5alle = find(class==5);
% min_class=min([length(c1alle),length(c2alle),length(c3alle),length(c4alle),length(
    c5alle)]);
%
% c1all = c1alle(randperm(length(c1alle)));
% c2all = c2alle(randperm(length(c2alle)));
% c3all = c3alle(randperm(length(c3alle)));
% c4all = c4alle(randperm(length(c4alle)));
% c5all = c5alle(randperm(length(c4alle)));
%
%
% cstayall = [c2all(1:42);c4all(1:41)];
% c24all = [c3all(1:33);c5all(1:50)];

for bS=1:10 %< bS = crossfold % run time = long
    % SPLIT TRAINING AND TEST
      training = [cstayall(1:83);c24all(1:83)];
test=[cstayall((8*bS-7):8*bS);c24all((8*bS-7):8*bS)];%          training = setdiff(
    training, test);
    if sum(ismember(test,training))~=0
        error('Test Has Training Data')
    end

    % TRAINING ONLY REFER TO *training*
     training_classstay2=training(class(training)==2);
     training_classstay4=training(class(training)==4);
    training_class2=training(class(training)==2);
%    training_class3=training(class(training)==3);
     training_class4=training(class(training)==5);
     training_classstay = [training_classstay2;training_classstay4];
     training_class24 = [training_class2;training_class4];
    min_class=min([length(training_class24),length(training_classstay)]);

    r1ye = datasample(training_class24,min_class,'Replace',false);
    r2ye = datasample(training_classstay,min_class,'Replace',false;
    pmatyee(bS,:) = [r1ye;r2ye];
    r1ye = [];
    r2ye = [];
    gmvecye = condensed_vec_raw((pmatyee(bS,:)),:);
    for1 = class(pmatyee(bS,:));
     for1(for1==5) = 3;
     for1(for1==4) = 2;
    [XL,YL,XS,YS,BETA,PCTVAR,MSE,STATS] = plsregress(gmvecye,zscore(for1),3);
    % recreate the score for ALL training subjects as a reference
    proto=condensed_vec_raw*STATS.W;

    parfor j = 1:1000
        r1y = datasample(training_class24,min_class,'Replace',true);
        r2y = datasample(training_classstay,min_class,'Replace',true);
        pmaty(j,:) = [r1y;r2y];
        r1y = [];
        r2y = [];
        gmvecy = condensed_vec_raw((pmaty(j,:)),:);
        for1h = class(pmaty(j,:));
        for1h(for1h==5) = 3;
        for1h(for1h==4) = 2;
        [XL,YL,XS,YS,BETA,PCTVAR,MSE,STATS] =plsregress(gmvecy,zscore(for1h),3);
        statswmatrix_new{bS,j} = STATS.W;
        gmvecy = [];

    end
    %pmaty_bs{bS} = pmaty;
    %pmaty = [];
%     for j = 1:1000
%         flipped_cp(1:10)=0;
%         for bsDim=1:10
%             for cProto=1:10
%                 temp=corrcoef(bs_scores{j}(:,bsDim),proto(training,cProto));
%                 compare_val(cProto)=temp(1,2);
%             end
%             % check for absoulte correspondence
%             [~,most_similar(bsDim)]=max(abs(compare_val));
%             corr_val(bsDim)=compare_val(most_similar(bsDim));
%             % check for axis flip of closest dimension
%
%             if (corr_val(bsDim)<0)
%                 flipped_cp(bsDim)=-most_similar(bsDim);
%             end
%         end
%         %stack results for the bootstrap
%         boot_strap_order2{bS}(j,1:10)=most_similar;
%         boot_strap_fliped2{bS}(j,1:10)=flipped_cp;
%         boot_strap_correspondence2{bS}(j,1:10)=corr_val;
%     end
    bS
end



%% (5) Variance Normalise and Take Mean from PLS regressions from samples (for each
    crossfold)

for bS = 1:10
    for component = 1:3
        for sample = 1:1000
            hold = std(statswmatrix_new{bS,sample}(:,component));
            for nVox=1:length(statswmatrix_new{bS,component})
                % statswmatrix{component}(j,:) = transpose(my{j}.W(:,component));
                varsamplee{bS,sample}(component,nVox) = statswmatrix_new{bS,sample}(nVox,
                    component)./hold;
            end
            sumvarsamplee{bS}(1:3,1:length(statswmatrix_new{bS,component}))=0;
            %^run this line if it doesnt work
            sumvarsamplee{bS}(component,:) = sumvarsamplee{bS}(component,:) + varsamplee{
                bS, sample}(component,:);
            varsamplee = [];
        end
        meanvarsamplee{bS}(component,:) = sumvarsamplee{bS}(component,:)./1000;

    end
    bS %Since this runs slow, output bS to check running correctly
end

%% (6) Make Mask of Most 'Significant' Voxels. View Features as nii.
for i = 1:10
    for j = 1:10
meanacc{i,j}(1:3, 1:3) = zeros;
    end
end

% for loop_over = 1:10

for cf = 1:10
    for component = 1:3
        var_norm = meanvarsamplee{1,cf}(component,:);
        ld3_thresh=zeros(size(maskgmvector));
        ld8=zeros(size(maskgmvector));
        ld8(positions) = var_norm;
        ld9 = zeros(size(maskgmvector));
        %ld9(positions) = meansamplee{cf,component};
        ld9(positions) = var_norm;
        lpct=prctile(var_norm,2);
        hpct=prctile(var_norm,(100-2));

        % make binary mask of voxels that are significant
        ld3_thresh(ld8<=lpct)=1;
        ld3_thresh(ld8>=hpct)=1;

        % mask out non significant weights
        ld3_high_low_map=ld9.*ld3_thresh;
        ld4 = ld3_high_low_map;
        ld2 = reshape(ld4,[97,116,97]);
           reconstructed = make_nii(ld2);  %% Uncomment to view nii
       view_nii(reconstructed);  %% Uncomment to view nii
        % %
        re_create_weight=ld3_high_low_map(positions); % use only the significant weights
            to recreate the score per subject
        for i=1:length(class)
            score_sub_trialbgm{cf,1}(i, component)=re_create_weight*condensed_vec_raw(i
                ,:)'; %<check if wm/gm
        end
    end

end


% 'score_sub_trialgm/wm' is to be fed to classifer, add the required components'
% scores to data2 in columns 7:end.

%% (7) Feed to GMLVQ Classifer
```

```

cd('T:\Users\jjg50\Jan\classification_files');

% for class_loop = 1:10
%Set learning parameters
alpha = 0.0005*8;
eta=alpha/10;
epochs = 1000;
decay = 0.00;
normalize = 0;

% for param_loop1 = 1:3
% for param_loop2 = 1:3
for bS = 1:10
      training = [cstayall(1:83);c24all(1:83)];
        test=[cstayall((8*bS-7):8*bS);c24all((8*bS-7):8*bS
        training = setdiff(training, test);
    %test
    data2zi = zscore([score_sub_trialbwm{bS,1}(:,1) score_sub_trialbgm{bS,1}(:,1:3)]);
    data2z = [classdata5(:,1) classdata5(:,3:6) data2zi];
    Data_allT=data2z(test,2:5);
    labelT=data2z(test,1);
    %training
    Data_all=[data2z(training,1) data2z(training,2:5)];
    Data_all(Data_all(:,1)==5,1)=3;
    Data_all(Data_all(:,1)==4,1)=2;
    label=data2z(training,1);
    label(label==5)=3;
    label(label==4)=2;
    class1=find(label==2);
    class2=find(label==3);
    test_data = [labelT,Data_allT];
     test_data(test_data(:,1)==5)=3;
     test_data(test_data(:,1)==4)=2;

    for train_loop=1:6
        rng('default')
        rng shuffle
        resamplec1=randsample(find(Data_all(:,1)==2),round(length(class1)*0.8),0);
        resamplec2=randsample(find(Data_all(:,1)==3),round(length(class1)*0.8),0);


        resample_ind=[resamplec1;resamplec2];

        Data=Data_all(resample_ind,2:end);
        Label=label(resample_ind,:);

        result=train_gmlvq_cv ([],Data,Label,alpha,eta,1,'n_epoch',epochs,'n_proto_class'
            ,[1,1],'test_data',test_data,'decay_factor',decay,'init',1,'matrix_start',1);

        test_this_loop(bS,train_loop)=result.accuracy_test(end);
        acc(bS,train_loop) = 1-result.accuracy_test(end) %outputs classification accuracy
        train_this_loop_er(bS,train_loop)=result.accuracy_training(end);
        test_res(:,train_loop)=result.xx;
        omega_mat2{bS,train_loop}=result.omega;

        healthy_index1 = find(class(test) ==2 );
        healthy_index2 = find(class(test) ==4 );
         mci_index1 = find(class(test) ==3 );
         mci_index2 = find(class(test) ==5 );
        sensitivity(bS,train_loop) = (length(find(result.xx(mci_index1)==3))+length(find(
            result.xx(mci_index2)==3)))/(length(mci_index1)+length(mci_index2))
         specificity(bS,train_loop) = (length(find(result.xx(healthy_index1)==2))+length(
             find(result.xx(healthy_index2)==2)))/(length(healthy_index1)+length(
             healthy_index2))

    end
%     test_outer(:,bS)=round(mean(test_res,2));
%     accuracy_outter(bS)=sum((round(mean(test_res,2))-test_data(:,1))==0)/length(
    test_data);
%     train_acc=mean(mean(train_this_loop_er));
%     test_acc=mean(accuracy_outter);
%     mean_test_acc=mean(mean(test_this_loop));
end
meanacc{loop_over,class_loop}(param_loop1, param_loop2) = mean(mean(acc));
acc = [];
end
end
end
end

% Used in hyperparameter fixing only
for i = 1:10
    for j = 1:10
    meanmat(i,j) = max(max(meanacc{i,j}))
    end
end
```