Cross-validation in high-dimensional spaces: a lifeline for least-squares models and multi-class LDA

Matthias S. Treder

Cardiff University Brain Research Imaging Centre (CUBRIC), Cardiff University, United Kingdom

Abstract

Least-squares models such as linear regression and Linear Discriminant Analysis (LDA) are amongst the most popular statistical learning techniques. However, since their computation time increases cubically with the number of features, they are inefficient in high-dimensional neuroimaging datasets. Fortunately, for k-fold cross-validation, an analytical approach has been developed that yields the exact cross-validated predictions in least-squares models without explicitly training the model. Its computation time grows with the number of test samples. Here, this approach is systematically investigated in the context of cross-validation and permutation testing. LDA is used exemplarily but results hold for all other least-squares methods. Furthermore, a non-trivial extension to multi-class LDA is formally derived. The analytical approach is evaluated using complexity calculations, simulations, and permutation testing of an EEG/MEG dataset. Depending on the ratio between features and samples, the analytical approach is up to 10,000x faster than the standard approach (retraining the model on each training set). This allows for a fast cross-validation of least-squares models and multi-class LDA in high-dimensional data, with obvious applications in multi-dimensional datasets, Representational Similarity Analysis, and permutation testing.

Keywords: MVPA, classification, cross-validation, permutation testing, LDA, high-dimensional spaces, RSA

1. Introduction

Multivariate pattern analysis (MVPA) is a statistical technique in which a target variable such as a brain state or reaction time is predicted based on multivariate patterns of brain activity [1]. The spadework in MVPA is performed by regression models if the dependent variables is continuous (e.g. reaction time), or by classifiers if the dependent variable is categorical (e.g. stimulus type) [2]. Due to their simplicity, relatively low computational demands, and high interpretability, least-squares models have been popular for both regression problems (linear regression, ridge regression) and for classification problems (Linear Discriminant Analysis [3]).

The increase in storage capabilities, working memory, and computational power, and the increasing availability of high-performance compute clusters paved the way for large-scale analyses of neuroimaging data. Analyses can deal with larger throughput than ever before, such as higher field strengths in fMRI and larger number of electrodes in EEG, or simply a larger amount of derived features such as time-frequency and connectivity metrics. It is worth stressing that most neuroimaging datasets have a $P \gg N$ shape, that is the number of features P is much larger than the number of samples N. An extreme example of this is gene expression data comprising tens of thousands of genes (features) but not more than a few hundred patients (samples) [4, 5]. In cognitive neuroscience, the number of samples for an analysis is naturally capped by limits of experiment time and group size. For level 1 analyses, the number of trials is limited by the amount of time the subject can spend in the scanner. Even in fast-paced EEG/MEG experiments, it is very rare that more than 10,000 trials are collected. For level 2 analyses, sample size is equal to the number of subjects, which is typically less than a few hundred. Summarising, the principal challenge in large neuroimaging datasets is to efficiently cope with high-dimensional data.

Unfortunately, this is exactly the Achilles heel of least-squares methods (LSM) such as linear regression, ridge regression, and Linear Discriminant Analysis (LDA). The computationally most expensive part in LSM is the inversion of the features $\times$ features scatter matrix. Computation time increases cubically with the number of features. It can therefore be intractable for even a few thousand features if a large number of training-testing iterations is needed, such as in permutation testing or in Representational Similarity Analysis [6] with many experimental conditions. This is one of the reasons that some researchers explore kernel methods such as Support

Vector Machines [7] whose complexity grows with the number of samples rather than number of features.

Does this mean that, for all practical purposes, high-dimensional datasets are beyond reach for LSM? Fortunately, for cross-validation [8], an alternative has been developed that addresses this issue. The analytical approach for LSM has the following [8]useful property: instead of requiring the inversion of a feature× features scatter matrix on each training set, it instead relies on the inversion of a matrix that grows with the number of test samples. It is therefore only mildly affected by the number of features. For leave-one-out cross-validation, the analytical approach is well-known in the linear regression literature [9, 10, 11], and it has been generalised to k-fold cross-validation [12, 13].

The aim of this study is to show that LSM can successfully meet the challenges of high-dimensional data when using the analytical approach. Because of the formal equivalence between linear regression and LDA (resp. ridge regression and regularised LDA), it suffices to focus on LDA alone. All results automatically generalise to linear regression and ridge regression. Equipped with regularisation techniques such as ridge regularisation [14] or shrinkage regularisation [15], LDA is robust to overfitting in high-dimensional data. It often performs similarly to more sophisticated linear classifiers such as linear support vector machines (SVM) while being significantly faster to train [16].

The novel contributions in this manuscript are a detailed empirical evaluation of the analytical approach for cross-validation using simulations and complexity calculations. Furthermore, to the best of my knowledge, this is the first application in permutation testing which is a popular approach in statistical testing of classifier performance [17, 18, 19, 20, 21]. It is also the first time that the approach is formally extended to multi-class LDA using an optimal scoring approach [22].

The manuscript is structured as follows. Firstly, cross-validation, binary LDA and a regression formulation of LDA are introduced. Then the analytical approach is developed for cross-validation and permutation testing, and both ridge regularisation and shrinkage regularisation are considered. Finally, it is formally extended to multi-class LDA [23, 22]. The computation time of the approach is then compared to the computation time of the standard approach (retraining the model for every fold) using complexity calculations, simulations, and a permutation analysis of a publicly available EEG/MEG dataset [24].

2. Method

Matrices will denoted by bold upper case letters, for instance $\mathbf{X}$. Vectors are denoted as bold lower case letters, $\mathbf{x}$, and are assumed to be column vectors. For scalars, normal font type is used. Upper case scalars are used for specifying the dimensionality of the matrices or vectors. N is the number of samples, P the number of features or predictors, K is the number of cross-validation folds, C is the number of classes, and T is the total number of training-testing iterations in during permutation testing.

2.1. Cross-validation

Cross-validation allows to estimate predictive performance while at the same time controlling for overfitting and making efficient use of the available samples [8, 21, 2]. In k-fold cross-validation, the dataset is randomly partitioned into K equally sized folds. The classifier is trained on all but one of the folds, and then tested on the held out fold. This procedure is repeated until every fold served as test set once. Classification performance is then averaged across the test folds. To reduce the variance stemming from the random partitioning of data into folds, the cross-validation can be repeated several times, finally averaging across the repeats.

2.2. Linear Discriminant Analysis (LDA)

For two classes, LDA is equivalent to Fisher Discriminant Analysis (FDA) [3, 15, 25, 26]. The multi-class case is considered further below. Geometrically speaking, LDA seeks a projection $\mathbf{w}$ from feature space to a 1-dimensional subspace such that the projected class means are maximally separated while at the same time the projected variance within classes is minimised [3, 15, 25, 26]. Using the LDA derivation in Duda & Hart [26] this can be formalised as:

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_b \, \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_w \, \mathbf{w}}$$

where $\mathbf{S}_b \in \mathbb{R}^{P \times P}$ is the between-classes scatter matrix measuring the distance between the classes and $\mathbf{S}_w \in \mathbb{R}^{P \times P}$ is the within-class scatter matrix measuring the spread within each class. These quantities are defined as

$$\mathbf{S}_b = \sum_{l \in \{1,2\}} N_l \, (\mathbf{m}_l - \overline{\mathbf{m}})(\mathbf{m}_l - \overline{\mathbf{m}})^\top \quad \text{(between-classes scatter)}$$
$$\mathbf{S}_w = \sum_{l \in \{1,2\}} \sum_{i \in \mathcal{C}_l} (\mathbf{x}_i - \mathbf{m}_l)(\mathbf{x}_i - \mathbf{m}_l)^\top \quad \text{(within-class scatter)}$$

where $\mathcal{C}_l$ is an index set representing the samples in class l, N_l is the number of samples in class l, and the means are given by

$$\begin{aligned} \mathbf{m}_l &= \frac{1}{N_l} \sum_{i \in \mathcal{C}_l} \mathbf{x}_i \qquad \text{(class mean)} \\ \overline{\mathbf{m}} &= \frac{1}{N} \sum_{i \in \{1,2,\ldots,N\}} \mathbf{x}_i \quad \text{(sample mean)} \end{aligned} \tag{1}$$

Obviously, the sample mean and the two class means are related by $N\,\overline{\mathbf{m}} = N_1 \mathbf{m}_1 + N_2 \mathbf{m}_2$. In the two-classes case, $\mathbf{S}_b$ further simplifies to

$$\mathbf{S}_b = \frac{N_1 N_2}{N} \, (\mathbf{m}_1 - \mathbf{m}_2) \, (\mathbf{m}_1 - \mathbf{m}_2)^\top \quad \text{(between-classes scatter)} \tag{2}$$

Setting $\lambda = J(\mathbf{w})$ one arrives at the generalised eigenvalue problem $\mathbf{S}_b \, \mathbf{w} = \lambda \, \mathbf{S}_w \, \mathbf{w}$. For a binary classification problem and a positive definite within-class scatter matrix, the eigenvector corresponding to the largest eigenvalue is proportional to

$$\mathbf{w} = \mathbf{S}_w^{-1} \, (\mathbf{m}_1 - \mathbf{m}_2) \quad \text{(weight vector)} \tag{3}$$

This is proved in Lemma 1 in Appendix C. It is expedient to define the bias as the center between the projected class means because this prevents the classifier from being biased towards one of the classes if the number of training samples per class is not equal:

$$b_{\text{LDA}} = -\mathbf{w}^\top (\mathbf{m}_1 - \mathbf{m}_2)/2 \quad \text{(bias term)} \tag{4}$$

The classifier output $\hat{y}$ for a new sample $\mathbf{x}$ is calculated as $\hat{y} := \mathbf{w}^\top \mathbf{x} + b$. This quantity is the signed distance to the hyperplane, more generally known as *decision value*. It is these decision values that are subject to the analytical approach developed below. The class labels can be derived from the sign of the decision value, with class "+1" for $\hat{y} \geq 0$ and class "−1" for $\hat{y} < 0$ [25].

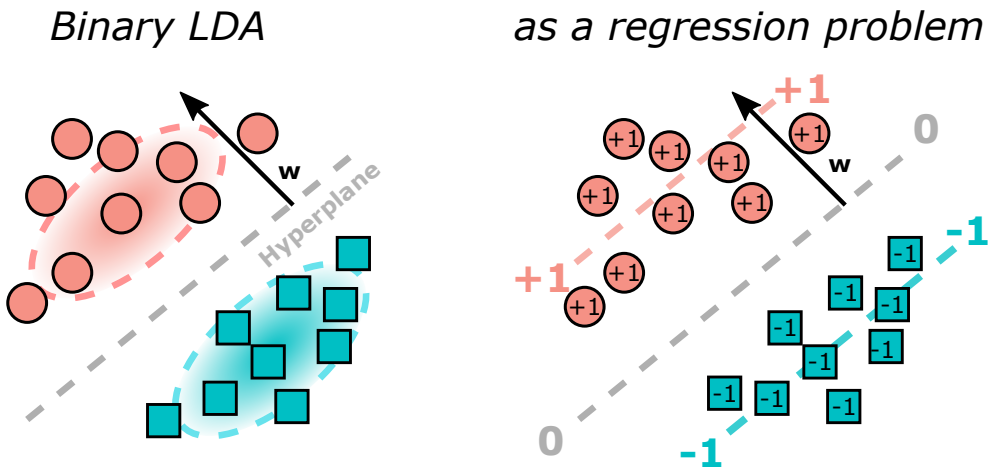

Figure 1: Two equivalent perspectives on binary LDA. *Left*: Classical view of LDA as a classification problem. Class distributions are modelled as multivariate Gaussian densities (indicated by the ellipses and shaded areas) and $\mathbf{w}$ is the normal to the optimal separating hyperplane. *Right*: LDA can be framed as a regression problem by coding each class by a number (e.g. +1 and -1) and then performing linear regression using the features as predictors and class labels as response variable. Both approaches yield the same $\mathbf{w}$ (up to scaling).

2.3. Binary LDA as a least-squares problem

There are several equivalent formulations of LDA. For two classes, LDA is formally equivalent to LCMV beamforming [27, 28, 29]. Furthermore, as illustrated in Figure 1, binary LDA can be cast as a least-squares regression problem [25, 26, 30, 31]. Let $\mathbf{X} \in \mathbb{R}^{N \times P}$ be the data matrix containing samples as rows and features as columns. Then $\widetilde{\mathbf{X}} \in \mathbb{R}^{N \times (P+1)}$ is the augmented data matrix obtained by adding a column of 1's (for the bias term) and $\boldsymbol{\beta}$ is the regression weights vector absorbing both $\mathbf{w}$ and b

$$\widetilde{\mathbf{X}} = [\mathbf{X},\ \mathbb{1}_N] \in \mathbb{R}^{N \times (P+1)}, \quad \boldsymbol{\beta} = \begin{pmatrix} \mathbf{w} \\ b_{\mathrm{LR}} \end{pmatrix} \in \mathbb{R}^{P+1}$$

where $\mathbb{1}_N$ is a vector of N ones. The bias term is denoted as b_{LR} since it is generally different from the LDA bias term b_{LDA}. The class labels are collected in a response vector $\mathbf{y} \in \mathbb{R}^N$ that uses numerical codes (e.g. $+1$ and -1) for the class labels. The standard regression problem using the full dataset can then be formulated as

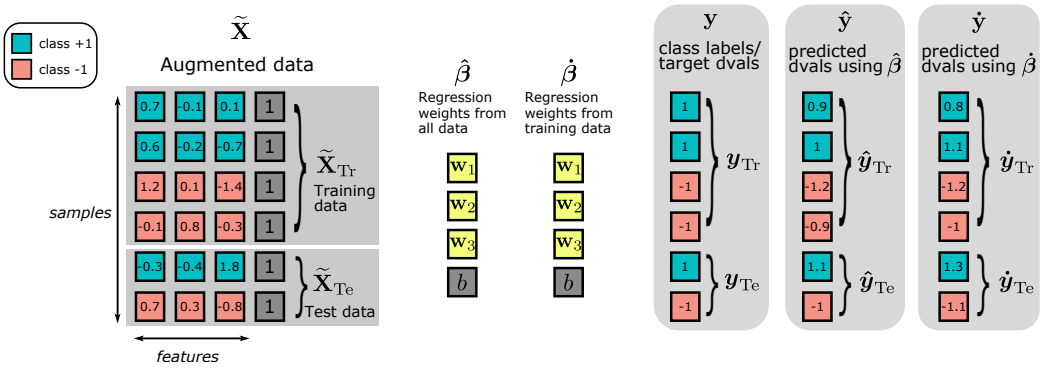

Figure 2: Visual depiction of important variables used in the derivation. Decision values are abbreviated as 'dvals'.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\widetilde{\mathbf{X}}\,\boldsymbol{\beta} - \mathbf{y}||_2^2 \tag{5}$$

with the solution given by $\hat{\boldsymbol{\beta}} = (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}})^{-1}\ \widetilde{\mathbf{X}}^\top\,\mathbf{y}$. Following the derivation in Appendix A, and assuming that the class labels are coded as $+1$ and -1, it can be shown that $\hat{\boldsymbol{\beta}}$ consists of the two components

$$\begin{aligned} \mathbf{w} &\propto \mathbf{S}_w^{-1}\ (\mathbf{m}_1 - \mathbf{m}_2) \\ b_{\mathrm{LR}} &= \frac{N_1 - N_2}{N} - \overline{\mathbf{m}}^\top \mathbf{w} \end{aligned} \tag{6}$$

In other words, the solution for $\mathbf{w}$ using linear regression is proportional to the LDA solution given in Eq. (3). Since scaling does not affect classification performance, one can say that the solutions are identical. Unless the classes have equal proportions of samples ($N_1 = N_2$), the bias term b_{LR} differs from the common choice presented in Eq. (4). However, class proportions and the exact numerical coding of the classes in $\mathbf{y}$ do not affect the direction of $\mathbf{w}$. This is shown in Appendix A.

2.4. An analytical approach to cross-validation for least-squares methods

In this section, the analytical approach to cross-validation is introduced for binary LDA. For linear regression and ridge regression, the approach is identical, with $\mathbf{y}$ simply being continuous a response variable instead of a

vector of class labels. The approach has been introduced before for leave-one-out cross-validation [9, 10, 11] and k-fold cross-validation [12, 13] but without the detailed derivation provided here.

Let Tr $\subset \{1, 2, \ldots, N\}$ be the indices of the training samples, and Te $\subset \{1, 2, \ldots, N\}$ be the indices of the test samples. Let $\mathbf{y} \in \{-1, +1\}^N$ be the vector of class labels. In the regression framework, these class labels serve as target decision values. The decision values obtained from the classifier trained on the full dataset are denoted as $\hat{\mathbf{y}}$. The cross-validated decision values obtained from a classifier trained on only the training set and then tested on the independent test set are denoted as $\dot{\mathbf{y}}$. $\mathbf{X}_{\text{Tr}}$ resp. $\mathbf{X}_{\text{Te}}$, and $\mathbf{y}_{\text{Tr}}$ resp. $\mathbf{y}_{\text{Te}}$ refer to the submatrix or subvector corresponding the training resp. test samples. To ease reading of the formulas, some important quantities used in the derivations are depicted in Figure 2.

2.4.1. Basic idea

In the regression framework, training the classifier is equivalent to calculating the vector of regression weights $\dot{\boldsymbol{\beta}}$ from the training data

$$\dot{\boldsymbol{\beta}} = (\widetilde{\mathbf{X}}_{\text{Tr}}^{\top} \widetilde{\mathbf{X}}_{\text{Tr}})^{-1} \widetilde{\mathbf{X}}_{\text{Tr}}^{\top} \mathbf{y}_{\text{Tr}} \quad \text{(model based on training data)} \tag{7}$$

In k-fold cross-validation, this process is repeated for each of the K training folds. However, as will be shown next, it suffices to train only *one* model using all available data

$$\hat{\boldsymbol{\beta}} = (\widetilde{\mathbf{X}}^{\top} \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^{\top} \mathbf{y} \quad \text{(model based on all data)}$$

and then obtain the cross-validated decision values directly via an analytical approach.

2.4.2. Hat matrix

The hat matrix $\mathbf{H} \in \mathbb{R}^{N \times N}$ is defined as

$$\mathbf{H} = \widetilde{\mathbf{X}} (\widetilde{\mathbf{X}}^{\top} \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^{\top} \quad \text{(hat matrix)} \tag{8}$$

It is a quantity well-known in linear regression [32]. Its name stems from the fact that it "puts the hat" onto the response vector $\mathbf{y}$ by mapping the

true responses onto the predicted responses $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$. The submatrix $\mathbf{H}_{\mathrm{Te}}$ is obtained from $\mathbf{H}$ by selecting only the rows and columns that correspond to test samples. $\mathbf{H}_{\mathrm{Tr,Te}}$ is obtained from $\mathbf{H}$ by selecting the rows corresponding to training samples and the columns corresponding to test samples. As will be seen below, the hat matrix arises naturally during updating.

2.4.3. Updating $\widetilde{\mathbf{X}}_{Tr}^{\top}\mathbf{y}_{Tr}$

A formula that will prove useful later on is the product $\widetilde{\mathbf{X}}_{\mathrm{Tr}}^{\top}\mathbf{y}_{\mathrm{Tr}}$ which can be obtained as follows

$$\widetilde{\mathbf{X}}_{\mathrm{Tr}}^{\top}\mathbf{y}_{\mathrm{Tr}} = \widetilde{\mathbf{X}}^{\top}\mathbf{y} - \widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\mathbf{y}_{\mathrm{Te}}. \tag{9}$$

2.4.4. Updating the inverse scatter matrix

Similarly, the scatter matrix on the training data can be obtained from the full scatter matrix by removing the scatter corresponding to the test samples,

$$\widetilde{\mathbf{X}}_{\mathrm{Tr}}^{\top}\widetilde{\mathbf{X}}_{\mathrm{Tr}} = \widetilde{\mathbf{X}}^{\top}\widetilde{\mathbf{X}} - \widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\widetilde{\mathbf{X}}_{\mathrm{Te}}. \tag{10}$$

Suppose that $\widetilde{\mathbf{X}}^{\top}\widetilde{\mathbf{X}}$ and its inverse, denoted as $\mathbf{S} := (\widetilde{\mathbf{X}}^{\top}\widetilde{\mathbf{X}})^{-1}$ have already been calculated and let $\mathbf{I}$ be the identity matrix. The matrix inversion lemma (a.k.a. Sherman-Morrison-Woodbury formula) can be used to update the inverse scatter matrix on the training data as

$$\begin{aligned}(\widetilde{\mathbf{X}}_{\mathrm{Tr}}^{\top}\widetilde{\mathbf{X}}_{\mathrm{Tr}})^{-1} &= (\widetilde{\mathbf{X}}^{\top}\widetilde{\mathbf{X}} - \widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\widetilde{\mathbf{X}}_{\mathrm{Te}})^{-1} \\ &= \mathbf{S} + \mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}(\mathbf{I} - \widetilde{\mathbf{X}}_{\mathrm{Te}}\,\mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top})^{-1}\widetilde{\mathbf{X}}_{\mathrm{Te}}\,\mathbf{S} \\ &= \mathbf{S} + \mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\widetilde{\mathbf{X}}_{\mathrm{Te}}\,\mathbf{S}\end{aligned} \tag{11}$$

This solution circumvents the explicit inversion of the scatter matrix, but it still involves a number of matrix multiplications. It therefore only serves as an intermediate result.

2.4.5. Updating the weights

To calculate the weights on the training data, Eq. (9) and Eq. (11) can be plugged into Eq. (7). This yields

$$
\begin{aligned}
\dot{\boldsymbol{\beta}} = & \left(\mathbf{S} + \mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\,(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\widetilde{\mathbf{X}}_{\mathrm{Te}}\,\mathbf{S}\right)\;(\widetilde{\mathbf{X}}^{\top}\mathbf{y} - \widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\,\mathbf{y}_{\mathrm{Te}}) \\
&= \hat{\boldsymbol{\beta}} - \left(\mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\,(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\right)\;\left([\mathbf{I} - \mathbf{H}_{\mathrm{Te}}]\,\mathbf{y}_{\mathrm{Te}} - \widetilde{\mathbf{X}}_{\mathrm{Te}}\,\hat{\boldsymbol{\beta}} + \mathbf{H}_{\mathrm{Te}}\,\mathbf{y}_{\mathrm{Te}}\right) \\
&= \hat{\boldsymbol{\beta}} - \left(\mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\,(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\right)\;\left(\mathbf{y}_{\mathrm{Te}} - \widetilde{\mathbf{X}}_{\mathrm{Te}}\,\hat{\boldsymbol{\beta}}\right) \\
&= \hat{\boldsymbol{\beta}} - \left(\mathbf{S}\,\widetilde{\mathbf{X}}_{\mathrm{Te}}^{\top}\,(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\right)\;(\,\mathbf{y}_{\mathrm{Te}} - \hat{\mathbf{y}}_{\mathrm{Te}})
\end{aligned}
\tag{12}
$$

where $\hat{\mathbf{e}}_{\mathrm{Te}} := \mathbf{y}_{\mathrm{Te}} - \hat{\mathbf{y}}_{\mathrm{Te}}$ is the estimation error on the test samples between the correct and the predicted decision values using a model trained on the full dataset. As will be seen next, $\dot{\boldsymbol{\beta}}$ does not need to be calculated explicitly.

2.4.6. Updating the decision values

The goal is to derive the cross-validated decision values for the test samples denoted as $\dot{\mathbf{y}}_{\mathrm{Te}}$. As an intermediate step, the corresponding estimation error is calculated first.

$$
\dot{\mathbf{e}}_{\mathrm{Te}} = \mathbf{y}_{\mathrm{Te}} - \dot{\mathbf{y}}_{\mathrm{Te}} \quad \text{(cross-validated estimation error)} \tag{13}
$$

Inserting Eq. (12) then leads to the desired analytical approach

$$
\begin{aligned}
\dot{\mathbf{e}}_{\mathrm{Te}} &= \mathbf{y}_{\mathrm{Te}} - \widetilde{\mathbf{X}}_{\mathrm{Te}}\,\dot{\boldsymbol{\beta}} \\
&= \underbrace{\mathbf{y}_{\mathrm{Te}} - \hat{\mathbf{y}}_{\mathrm{Te}}}_{=\hat{\mathbf{e}}_{\mathrm{Te}}} + \mathbf{H}_{\mathrm{Te}}(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\,\hat{\mathbf{e}}_{\mathrm{Te}} \\
&= (\mathbf{I} - \mathbf{H}_{\mathrm{Te}} + \mathbf{H}_{\mathrm{Te}})\,(\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\,\hat{\mathbf{e}}_{\mathrm{Te}} \\
&= (\mathbf{I} - \mathbf{H}_{\mathrm{Te}})^{-1}\,\hat{\mathbf{e}}_{\mathrm{Te}} \qquad \text{(analytical approach)}
\end{aligned}
\tag{14}
$$

It is now easy to obtain the cross-validated decision values on the test set by simply solving Eq. (13) for $\dot{\mathbf{y}}_{\mathrm{Te}}$. Finally, these decision values can be used to calculate classification accuracy, AUC, or any other desired metric of classification performance.

2.5. Adjusting the bias term

The bias term resulting from the regression approach does not generally coincide with the bias term used in LDA. However, the bias terms to coincide

if $N_1 = N_2$. Hence, for unbalanced data, undersampling of the majority class or oversampling of the minority class is a remedy. Alternatively, if area under the ROC curve (AUC) is used as classifier performance metric, the bias term is irrelevant.

If it is not possible to use one of these approaches, the bias term needs to be adjusted. To this end, the class means and the sample mean on the training need to be calculated and projected onto $\mathbf{w}$ (see definition of b_{LDA} and b_{LR}). Fortunately, it is not required to explicitly calculate $\mathbf{w}$. Instead, one can determine the decision values of the cross-validated model on the training set, $\dot{\mathbf{y}}_{\mathrm{Tr}}$, and then calculate b_{LR} and b_{LDA} directly. This is achieved by applying Eq. (14) to the training data:

$$\begin{aligned} \dot{\mathbf{e}}_{\mathrm{Tr}} &= \hat{\mathbf{e}}_{\mathrm{Tr}} + \mathbf{H}_{\mathrm{Tr,Te}} \left(\mathbf{I} - \mathbf{H}_{\mathrm{Te}}\right)^{-1} \hat{\mathbf{e}}_{\mathrm{Te}} \\ \dot{\mathbf{y}}_{\mathrm{Tr}} &= \mathbf{y}_{\mathrm{Tr}} - \dot{\mathbf{e}}_{\mathrm{Tr}} \end{aligned} \tag{15}$$

Finally, the operation $\dot{\mathbf{y}}_{\mathrm{Te}} \leftarrow \dot{\mathbf{y}}_{\mathrm{Te}} - b_{\mathrm{LR}} + b_{\mathrm{LDA}}$ adjusts the bias.

2.5.1. Summary: analytical approach

It has been shown that in k-fold cross-validation, it suffices to train just one regression model on the whole dataset. By evaluating Eq. (14) the decision values for each of the folds are obtained directly without explicitly calculating any of the K models.

2.6. Regularisation

Since neuroimaging data is often low-dimensional, or the number of samples is smaller than the number of features, the within-class scatter matrix tends to be ill-conditioned. Two similar regularisation approaches have been explored in the literature. They are equivalent in that they define the same family of classifiers (up to scaling of $\mathbf{w}$).

2.6.1. Ridge regularisation

A multiple of the identity matrix is added to the within-class scatter matrix [14, 33, 34]. The regularised within-class scatter matrix is $\mathbf{S}_w + \lambda\mathbf{I}$, where $\lambda \in [0, \infty]$ is the regularisation term and $\mathbf{I}$ is the identity matrix. $\lambda = 0$ yields the ordinary, unregularised solution. The ridge solution for $\mathbf{w}$ is then given by

$$\mathbf{w} = \left(\mathbf{S}_w + \lambda\mathbf{I}\right)^{-1} \left(\mathbf{m}_1 - \mathbf{m}_2\right) \tag{16}$$

In Appendix B it is proven that, in the regression framework, the corresponding solution is given by

$$\hat{\boldsymbol{\beta}} = (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0)^{-1}\ \widetilde{\mathbf{X}}^\top \mathbf{y} \tag{17}$$

where $\mathbf{I}_0 \in \mathbb{R}^{(P+1)\times(P+1)}$ is a diagonal matrix that is identical to the identity matrix except for the last element which is 0 instead of 1. This construction assures that the bias term corresponding to the last entry is not subjected to regularisation.

Analogous to Eq. (10), the regularised scatter matrix for the training data can be obtained as an update on the full model:

$$\widetilde{\mathbf{X}}_{\mathrm{Tr}}^\top \widetilde{\mathbf{X}}_{\mathrm{Tr}} + \lambda \mathbf{I}_0 = \underbrace{\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0}_{\text{full scatter}} - \underbrace{\widetilde{\mathbf{X}}_{\mathrm{Te}}^\top \widetilde{\mathbf{X}}_{\mathrm{Te}}}_{\text{update}}$$

After redefining $\mathbf{S} := (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0)^{-1}$ and correspondingly including the regularisation term in the hat matrix $\mathbf{H} = \widetilde{\mathbf{X}}\ (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0)^{-1}\ \widetilde{\mathbf{X}}^\top$, the analytical approach is identical to Eq. (14).

2.6.2. Shrinkage regularisation

The within-class scatter matrix is replaced by a convex combination of the empirical covariance matrix and a scaled identity matrix, $(1-\lambda)\ \mathbf{S}_w + \lambda\nu\mathbf{I}$, where $\nu = \mathrm{trace}(\mathbf{S}_w)/P$ is a scaling parameter that equalises the traces of $\mathbf{S}_w$ and $\nu\mathbf{I}$, and $\lambda \in [0, 1]$ [15]. Unfortunately, shrinkage regularisation does not allow for simple low-rank updates as before. This can be seen when one inspects an update of the regularised scatter matrix

$$(1-\lambda)\widetilde{\mathbf{X}}_{\mathrm{Tr}}^\top \widetilde{\mathbf{X}}_{\mathrm{Tr}} + \lambda\nu_{\mathrm{Tr}}\mathbf{I}_0 = \underbrace{(1-\lambda)\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda\nu\mathbf{I}_0}_{\text{full scatter}} - \underbrace{\Big((1-\lambda)\widetilde{\mathbf{X}}_{\mathrm{Te}}^\top \widetilde{\mathbf{X}}_{\mathrm{Te}} + \lambda(\nu - \nu_{\mathrm{Tr}})\mathbf{I}_0\Big)}_{\text{update}}$$

where ν is the scaling calculated on the full dataset and ν_{Tr} is the scaling calculated on the training data. The problem is that it is necessary to update the regularisation term as well. This is caused by the scaling factor ν_{Tr}, which changes for each training set, thereby changing the amount of regularisation. This turns a low-rank update into a full rank update, precluding significant performance gains by updating.

For this reason, it is recommended to resort to ridge regularisation. If a researcher is used to work with shrinkage, the following simple relation can be used to transform a given shrinkage parameter λ_{shrink} into a corresponding ridge parameter λ_{ridge}. Given a fixed value for λ_{shrink}, the goal is to find a λ_{ridge} such that the regularised scatter matrices are proportional:

$$(1 - \lambda_{\text{shrink}})\, \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda_{\text{shrink}}\, \nu\, \mathbf{I}_0 \quad \overset{!}{\propto} \quad \widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda_{\text{ridge}}\, \mathbf{I}_0$$

Obviously, this relation holds when the ridge parameter is defined as

$$\lambda_{\text{ridge}} = \frac{\lambda_{\text{shrink}}}{1 - \lambda_{\text{shrink}}}\, \nu \tag{18}$$

2.7. Using the analytical approach for permutation testing

The hat matrix $\mathbf{H}$ is invariant under class label permutations because it depends on the features alone. Consequently, it does not need to be recalculated when the class labels are permuted. Let the permuted class labels be denoted as $\mathbf{y}^\sigma$. If $\mathbf{y}$ and $\hat{\mathbf{y}}$ are adjusted accordingly

$$\begin{aligned} \mathbf{y} &\leftarrow \mathbf{y}^\sigma \\ \hat{\mathbf{y}} &\leftarrow \mathbf{H}\, \mathbf{y}^\sigma \end{aligned}$$

the formulas in the previous section directly apply. They are compiled in Algorithm 1.

2.8. Multi-class LDA

Multi-class LDA is the generalisation of binary LDA to more than two classes. Like binary LDA, it involves a projection step and a thresholding step. In the projection step, the data is mapped onto a $(C-1)$-dimensional subspace, where C is the number of classes. In the second step, a new sample is assigned to the class with the closest class centroid. LDA thus acts as a prototype classifier. The scatter matrices are calculated as before, but now information is pooled across all classes.

$$\begin{aligned} \mathbf{S}_b &= \sum_{j \in \{1,2,\ldots,C\}} n_j\, (\mathbf{m}_j - \overline{\mathbf{m}})(\mathbf{m}_j - \overline{\mathbf{m}})^\top \qquad \text{(between-classes scatter)} \\ \mathbf{S}_w &= \sum_{j \in \{1,2,\ldots,C\}} \sum_{i \in \mathcal{C}_j} (\mathbf{x}_i - \mathbf{m}_j)(\mathbf{x}_i - \mathbf{m}_j)^\top \qquad \text{(within-class scatter)} \end{aligned}$$

Algorithm 1 Fast cross-validation and permutations for binary LDA

$\mathbf{H} \leftarrow \widetilde{\mathbf{X}} \, (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0)^{-1} \, \widetilde{\mathbf{X}}^\top$
for all permutations σ **do**
  $\mathbf{y} \leftarrow \mathbf{y}^\sigma$
  $\hat{\mathbf{y}} \leftarrow \mathbf{H}\,\mathbf{y}^\sigma$
  for all test sets Te **do**
    $\dot{\mathbf{e}}_{\text{Te}} \leftarrow (\mathbf{I} - \mathbf{H}_{\text{Te}})^{-1} \, \hat{\mathbf{e}}_{\text{Te}}$
    $\dot{\mathbf{y}}_{\text{Te}} \leftarrow \mathbf{y}_{\text{Te}} - \dot{\mathbf{e}}_{\text{Te}}$
    Calculate classification performance on current test set
  end for
  Average classification performances across test sets
end for
Output: classification performance for each permutation

Assuming that there are more features than classes, the between-classes scatter matrix $\mathbf{S}_b$ has rank $C-1$. Consequently, there are multiple non-trivial solutions that again can be obtained via the generalised eigenvalue problem

$$\mathbf{S}_b \, \mathbf{W} = \mathbf{S}_w \, \mathbf{W} \mathbf{\Lambda} \tag{19}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues. A set of discriminant coordinates is obtained corresponding to non-zero eigenvalues of the eigenvalue problem. They are collected in a matrix $\mathbf{W} \in \mathbb{R}^{P \times (C-1)}$ and scaled such that $\mathbf{W}^\top \mathbf{S}_w \mathbf{W} = \mathbf{I}$ [25]. As for binary LDA, ridge regularisation can be applied to the within-class scatter matrix by replacing $\mathbf{S}_w$ by $\mathbf{S}_w + \lambda \mathbf{I}$ [14].

2.9. Multi-class LDA in a regression framework

Unfortunately, multi-class LDA is not equivalent to multivariate linear regression using a class indicator matrix as response matrix [2]. Nevertheless, there is a close relationship between both approaches [22, 2, 35, 36]. A useful characterisation is given in Hastie et al. [22]. They show that multi-class LDA is equivalent to optimal scoring (OS) wherein regression is performed using a response vector with optimal numerical scores for each class. Finding the optimal scores is an optimisation problem that is solved jointly with the regression problem. Let $\mathbf{Y} \in \mathbb{R}^{N \times C}$ be the class indicator matrix whose (i,j)-th element is defined as

$$\mathbf{Y}_{ij} = \begin{cases} 1 & \text{if sample } i \text{ belongs to class } j \\ 0 & \text{otherwise} \end{cases}$$

Let $\boldsymbol{\theta} \in \mathbb{R}^C$ be the vector containing the optimal scores. Then the response vector of optimal scores can be written as $\mathbf{Y}\boldsymbol{\theta}$, and the optimal scoring problem is given by

$$\arg\min_{\boldsymbol{\beta},\theta} ||\widetilde{\mathbf{X}}\boldsymbol{\beta} - \mathbf{Y}\boldsymbol{\theta}||_2^2 \quad \text{(optimal scoring)}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are jointly optimised. The additional constraint $N^{-1}||\mathbf{Y}\boldsymbol{\theta}||^2 = 1$ avoids trivial solutions. Hastie et al. [22] show that this optimisation problem can be broken up into two successive steps.

Step 1: A multivariate regression is performed on the class indicator matrix $\widetilde{\mathbf{B}} = \arg\min ||\widetilde{\mathbf{X}}\widetilde{\mathbf{B}} - \mathbf{Y}||_F^2$, where $||\cdot||_F$ is the Frobenius norm. The result is a matrix of regression weights $\widetilde{\mathbf{B}} \in \mathbb{R}^{(P+1)\times C}$, where each column of regression weights corresponds to the respective column of $\mathbf{Y}$. This yields the matrix of regression fits $\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}$.

Step 2: The optimal score vector is found via an eigendecomposition of $\hat{\mathbf{Y}}^\top\mathbf{Y}$. Let $\boldsymbol{\Theta} \in \mathbb{R}^{C\times(C-1)}$ be the eigenvectors of this decomposition, also called optimal scores, where the column corresponding to the trivial eigenvalue 0 (if $\widetilde{\mathbf{X}}$ is centered) or 1 (if $\widetilde{\mathbf{X}}$ is not centered) has been removed. Let $\alpha_1^2, \alpha_2^2, ..., \alpha_{C-1}^2$ be the corresponding eigenvalues. Let $\mathbf{B}$ be the submatrix of $\widetilde{\mathbf{B}}$ with the last row (bias term) omitted. Then the columns of $\mathbf{B}\boldsymbol{\Theta}$ point in the same directions as the discriminant coordinates obtained in multi-class LDA but their scaling differs. To scale the discriminant coordinates, they are right-multiplied with the diagonal matrix

$$\mathbf{D} = \sqrt{N}^{-1} \begin{pmatrix} \sqrt{\alpha_1^2(1-\alpha_1^2)} & 0 & \dots & 0 \\ 0 & \sqrt{\alpha_2^2(1-\alpha_2^2)} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \sqrt{\alpha_{C-1}^2(1-\alpha_{C-1}^2)} \end{pmatrix}^{-1}$$

Note that the normalisation $\sqrt{N}^{-1}$ does not appear in the original definition of the scaling matrix ([22], p. 83) but is necessary here because the multi-class LDA has been calculated using the within-class scatter matrix. In contrast, Hastie et al. use the covariance matrix which differs by a scaling factor of N. As main result of their derivation, the relationship between the discriminant coordinates $\mathbf{W}$ in Eq. (19) and the optimal scoring results is given by

$$\mathbf{W} = \mathbf{B}\boldsymbol{\Theta}\mathbf{D} \tag{20}$$

2.10. The analytical approach for multi-class LDA

How can these findings be used to develop an analytical approach for multi-class LDA? Starting from Eq. (20), a dot is used to indicate that the matrices have been estimated using the training data. Left-multiplication with the test data then yields

$$\begin{aligned} \mathbf{X}_{\mathrm{Te}}\,\dot{\mathbf{W}} &= \mathbf{X}_{\mathrm{Te}}\,\dot{\mathbf{B}}\dot{\boldsymbol{\Theta}}\dot{\mathbf{D}} \\ \Leftrightarrow\ \breve{\mathbf{Y}}_{\mathrm{Te}} &= \dot{\mathbf{Y}}_{\mathrm{Te}}\,\dot{\boldsymbol{\Theta}}\dot{\mathbf{D}} \end{aligned}$$

where $\breve{\mathbf{Y}}_{\mathrm{Te}}$ is used to denote the desired discriminant scores for the test data (obtained in step 2). This notation is necessary to differentiate them from the cross-validated regression fits $\dot{\mathbf{Y}}$ (obtained in step 1). After calculating $\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}$, $\dot{\mathbf{Y}}_{\mathrm{Tr}}$ and $\dot{\mathbf{Y}}_{\mathrm{Te}}$ can be obtained by applying Eq. (14) and Eq. (15) using the matrix of estimation errors $\hat{\mathbf{E}} = \mathbf{Y} - \hat{\mathbf{Y}}$. $\dot{\boldsymbol{\Theta}}$ and $\dot{\mathbf{D}}$ are then obtained via the eigenanalysis $\mathtt{eig}(\dot{\mathbf{Y}}_{\mathrm{Tr}}^{\top}\,\mathbf{Y}_{\mathrm{Tr}}/N_{\mathrm{Tr}})$ on the training data. Note that in practice, the augmented data matrix $\widetilde{\mathbf{X}}$ and the regression weights $\widetilde{\mathbf{B}}$ can be used. The classification results are equivalent since the distance of a sample to the class centroids is unaffected by the constant shift incurred by the bias term.

Concluding, an analytical approach for step 1 of OS has been developed. There is no straightforward way to update the eigenvalue decomposition in step 2. However, the eigenanalysed matrix is of dimensions $C \times C$, so for most practical applications the computational costs are negligible. Algorithm 2 compiles these results.

Algorithm 2 Fast cross-validation and permutations for multi-class LDA

$\mathbf{H} \leftarrow \widetilde{\mathbf{X}} \, (\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0)^{-1} \, \widetilde{\mathbf{X}}^\top$
for all permutations σ **do**
  $\mathbf{Y} \leftarrow \mathbf{Y}^\sigma$
  $\hat{\mathbf{Y}} \leftarrow \mathbf{H} \, \mathbf{Y}^\sigma$
  for all test sets Te **do**
    (step 1)
    $\dot{\mathbf{Y}}_{\mathrm{Tr}} \leftarrow \mathbf{Y}_{\mathrm{Tr}} - \dot{\mathbf{E}}_{\mathrm{Tr}}$
    $\dot{\mathbf{Y}}_{\mathrm{Te}} \leftarrow \mathbf{Y}_{\mathrm{Te}} - \dot{\mathbf{E}}_{\mathrm{Te}}$
    (step 2)
    $(\dot{\mathbf{\Theta}}, (\alpha_1^2, \alpha_2^2, ...)) \leftarrow \mathtt{eig}(\dot{\mathbf{Y}}_{\mathrm{Tr}}^\top \, \mathbf{Y}_{\mathrm{Tr}} / N_{\mathrm{Tr}})$
    $\breve{\mathbf{Y}}_{\mathrm{Te}} \leftarrow \dot{\mathbf{Y}}_{\mathrm{Te}} \, \dot{\mathbf{\Theta}} \dot{\mathbf{D}}$
    Calculate classification performance on current test set
  end for
  Average classification performances across test sets
end for
Output: classification performance for each permutation

Method	Classes	Complexity
Standard	Binary	$\mathcal{O}(KNP^2 + KP^3)$
analytical approach	Binary	$\mathcal{O}(KN^3)$
Standard	Multi-class	$\mathcal{O}(KNP^2 + KCP^2 + TP^3)$
analytical approach	Multi-class	$\mathcal{O}(KN^3C)$

Table 1: Computational complexity of training LDA classifiers in a permutation testing regime. The standard approach is compared with the analytical approach presented in this paper. K: #folds; N: #samples; P: #features; C: #classes

2.11. Computational complexity of the analytical approach

The asymptotic computational complexity for classifier validation using cross-validation is quantified in terms of floating point operations. The analytical approach developed in this paper is compared to the standard approach wherein a classifier is trained from scratch on every training set. Regularisation is not considered separately since the addition of the regularisation term inflicts negligible costs and is the same in both algorithms. For simplicity, the complexity is given in terms of the standard textbook algorithms. Speed-ups can of course be achieved using more sophisticated algorithms. The complexity calculations are summarised in Table 1.

2.11.1. Binary LDA

For training a single binary LDA classifier, two class means need to be calculated which involves adding up training samples and features and dividing two times, leading to a complexity of $\mathcal{O}(NP)$, where P is the number of features. Calculating the within-class scatter matrix requires $N(P + P^2)$ steps, where the P is for subtracting the class mean and the P^2 is for calculating the outer vector product ($\mathcal{O}(NP^2)$). Instead of then calculating the inverse of the within-class scatter matrix, one can solve the system of linear equations $\mathbf{S}_w \mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2$ ($\mathcal{O}(P^3)$). Calculation of the bias b_{LDA} requires $\mathcal{O}(P)$. Taken together, the complexity for training a single classifier amounts to $\mathcal{O}(NP^2 + P^3)$. This process is repeated K times, K being the number of folds. This yields an overall complexity of $\mathcal{O}(KNP^2 + KP^3)$ since the lower-order terms can be ignored for asymptotic complexity.

For the analytical approach based on the regression approach, the hat matrix needs to be calculated initially at a complexity of $\mathcal{O}(N^2P + NP^2 + P^3)$. Then Eq. (14) needs to be evaluated for each training iteration ($\mathcal{O}(KN^3)$). If the bias term needs to be corrected, operations at $\mathcal{O}(KN^2)$ are required. This yields an asymptotic complexity of $\mathcal{O}(KN^3)$ for the analytical approach.

2.11.2. Multi-class LDA

The asymptotic complexity for training a single multi-class LDA classifier is provided first. Calculating means for each of the C classes, involves adding up training samples and features and dividing C times ($\mathcal{O}(NP)$ + $\mathcal{O}(CP)$). Calculating the within-class scatter matrix is equal to the binary case ($\mathcal{O}(NP^2)$). Calculating the between-classes scatter matrix involves calculating C outer vector products $\mathcal{O}(CP^2)$. The generalised eigenvalue de-

composition has an overall complexity of $\mathcal{O}(P^3)$. Repeating this process K times yields an overall complexity of $\mathcal{O}(KNP^2 + KCP^2 + KP^3)$.

For the analytical approach based on the optimal scoring approach, the hat matrix needs to be calculated initially at a complexity of $\mathcal{O}(N^2P + NP^2 + P^3)$. Obtaining the cross-validated regression fits $\dot{\mathbf{Y}}_{\text{Tr}}$ and $\dot{\mathbf{Y}}_{\text{Te}}$ involves a complexity of $\mathcal{O}(KN^3C)$ each. This is followed by the calculation and eigendecomposition of $\dot{\mathbf{Y}}_{\text{Tr}}^{\top}\mathbf{Y}_{\text{Tr}}$ ($\mathcal{O}(KC^2N + KN^3C)$). Finally the discriminant scores are calculated ($\mathcal{O}(KC^2N)$). This yields an asymptotic complexity of $\mathcal{O}(KN^3C)$.

2.12. Simulations

To vet the analytical approach its efficacy is compared to the standard approach using simulated data. The data is created as follows: Each class centroid is randomly placed on the surface of a unit hypersphere in feature space. A common covariance matrix is randomly sampled from a Wishart distribution. Samples are then created by randomly sampling from a multivariate normal distribution parameterised by the corresponding class centroid and the common covariance matrix.

The number of features was varied from 10 to 1000 in 40 logarithmic steps. For binary LDA, cross-validation was performed using 5 folds, 10 folds, 20 folds, and leave-one-out. Simulations were run separately for 100 and 1000 samples. Simulations were run with 10-fold cross-validation and 100, 1000, or 10,000 permutations. The number of samples and the number of features were set to either 100 or 1000. For every combination of parameters, the simulation was repeated 20 times.

For multi-class LDA, 10-fold cross-validation was used with data being split into 5 classes or 10 classes with equal class proportions. For cross-validation, the number of samples was either 100 or 1000. For permutations, the number of features was fixed to 100 or 1000. The number of permutations was limited to 10 or 100 to keep overall computation time tractable. For every combination of parameters, the simulation was repeated 20 times for cross-validation. For permutations, it was repeated 10 times.

Both binary and multi-class LDA update rules were compared to the vanilla approach wherein the classifier is trained on each training set and then applied to the test set. Different datasets and different folds were randomly created for each choice of parameters. However, for each of the two methods (analytical approach vs classical approach) the random seed was reset to assure equal data and equal folds. All analyses were performed in MATLAB (Natick, USA). The `tic` and `toc` functions were used to measure the total

computation time for cross-validation and permutation testing iterations. As target measure, *relative effiency* was computed, defined as

$$\text{Relative efficiency} = log_{10}\ \frac{\text{time(standard approach)}}{\text{time(analytical approach)}}$$

This quantity has a simple interpretation in terms of *orders of magnitude of improvement in computation time* of the analytical approach over the standard method. For instance, a value of 0 means that both methods are at parity. A value of 1 means that the analytical approach is 10 times faster than the standard method, a value of 2 means that the analytical approach is $10^2 = 100$ times faster, and so on. Simulations were run on a Thinkpad X1 Carbon with 16 GB of RAM and an Intel Core i7-6600U CPU @ 2.60GHz $\times$ 4 processor.

2.13. EEG data

The analytical approach was applied to a publicly available multi-modal dataset of participants watching greyscale images of faces and scrambled faces [24]. The 16 EEG/MEG datasets with a total of 380 EEG/MEG channels were read into MATLAB using FieldTrip [37]. Epochs were created from -0.5 s to 1 s relative to image onset, and the pre-stimulus interval was used for baseline correction. Finally, data were downsampled to 200 Hz. The type of stimulus (face vs scrambled) was used as class label for binary LDA. For multi-class LDA, the face stimuli were further split in order to create 3 classes. The total number of trials varied across subjects, with 787 trials on average.

For both binary and multi-class LDA, two different analyses were conducted. In the first case, classification was performed separately for every time point across time interval ranging from -0.5 s to 1 s. At each time point, 100 permutations were conducted with shuffled class labels and a 10-fold cross-validation in each permutation. The amplitudes in each channel and were used as features (380 features). In the second case, the post-stimulus interval was divided into successive, non-overlapping windows. The amplitudes in each channel were averaged within these windows and then all averaged amplitudes were concatenated to a single feature vector. For binary LDA, 100 ms windows were used ($10 * 380 = 3800$ features). For multi-class LDA, 200 ms windows were used ($5 * 380 = 1900$ features).

Analyses were run on the high-performance cluster at the Cardiff University Brain Research Imaging Centre (CUBRIC). Each compute node consists

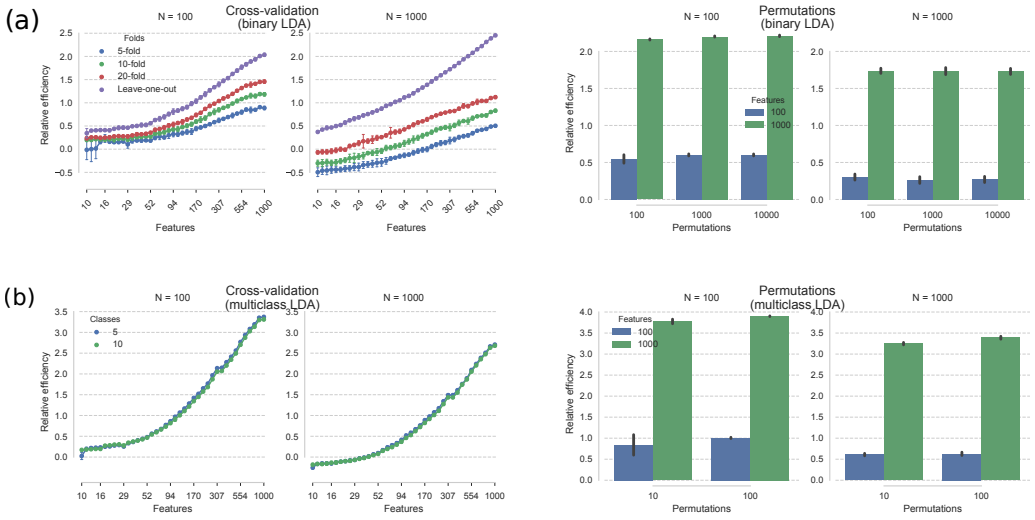

Figure 3: Results of the simulations denoted in terms of relative efficiency. A relative efficiency of 1 means that the analytical approach is 10x faster than the standard approach, 2 means 100x faster, and 3 means 1000x faster. (a) Binary LDA. (b) Multi-class LDA.

of 12 cores with 192 GB RAM and an Intel(R) Xeon(R) X5660 CPU running at 2.80GHz.

2.14. Software

MATLAB implementations of the analytical approach and the scripts reproducing the results are publicly available on GitHub (`github.com/treder/Fast-Least-Squares`) Note that parts of the code require the MVPA-Light toolbox (`github.com/treder/MVPA-Light`) to run.

3. Results

3.1. Simulations

Binary LDA. A three-way analysis of variance (ANOVA) was run on the cross-validation analysis (Figure 3, top left). A continuous variable (features) and two categorical variables, samples N (100 or 1000) and folds (5, 10, 20, leave-one-out), were used as predictors, and relative efficiency was used as dependent variable. There were significant main effects of features ($F = 32051.69; p < .001$), N ($F = 1316.19; p < .001$), and folds ($F = 3119.03; p < .001$). Furthermore, the effect of features increased with folds (features $\times$

N, $F = 806.49; p < .001$), and there was an N $\times$ folds interaction ($F = 812.7; p < .001$). Furthermore, there was a three-way interaction N $\times$ folds $\times$ features ($F = 37.16; p < .001$).

A separate three-way ANOVA was performed on the permutations data (Figure 3, top right) using N, permutations, and features as predictors. There were significant main effects for N ($F = 6899.92; p < .001$), permutations ($F = 4.35; p = .014$), and features ($F = 111506.59; p < .001$). Significant interactions were N $\times$ permutations ($F = 26.52; p < .001$) and N $\times$ features ($F = 273.66; p < .001$), illustrating that the effects of permutations and features were larger for N=1000 than for N=100. Other interactions were not significant (permutations $\times$ features, $p = .58$; N $\times$ permutations $\times$ features, $p = .08$).

Multi-class LDA. A three-way analysis of variance (ANOVA) was run on the cross-validation analysis (Figure 3, bottom left). A continuous variable (features) and two categorical variables, samples N (100 or 1000) and classes (5, 10), were used as predictors, and relative efficiency was used as dependent variable. There were significant main effects for N ($F = 1023.97; p < .001$), features ($F = 38270.22; p < .001$) but not for classes ($p = .15$). There was a significant features $\times$ N interaction ($F = 125.74; p < .001$) signifying a smaller effect of features for larger N. The other interactions were not significant (features $\times$ classes, $p = .1$; N $\times$ classes, $p = .86$, features $\times$ N $\times$ classes, $p = .462$).

A separate three-way ANOVA was performed on the permutations data (Figure 3, bottom right) using N, permutations, and features as predictors. There were significant main effects of N ($F = 366.2; p < .001$), permutations ($F = 27.4; p < .001$), and features ($F = 16970.31; p < .001$). Again, there was a significant N $\times$ features interaction ($F = 24.93; p < .001$) signifying a smaller effect of features for larger N. The other interactions were not significant (N $\times$ permutations, $p = .13$; permutations $\times$ features, $p = .35$, N $\times$ permutations $\times$ features, $p = .06$).

3.2. EEG/MEG data

Results are depicted in Figure 4. The data for binary LDA and multi-class LDA were combined into a single two-way ANOVA model. Features (small = 380, large = 3800/1900) and type of classifier (binary LDA, multi-class LDA) were used as predictors. There were significant main effects of features ($F = 826.04; p < .001$) and classifier ($F = 388.2; p < .001$). Moreover,

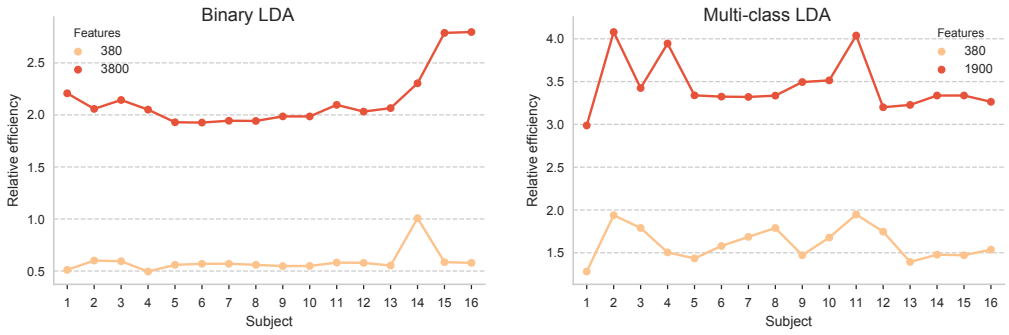

Figure 4: Results of the permutation analysis of the EEG/MEG dataset using 100 permutations. Relative efficiency is plotted for each subject (x-axis), and two different numbers of features, with different panels for binary LDA and multi-class LDA.

there was a significant features $\times$ classifier interaction ($F = 6.01; p = .017$) signifying a larger effect of features for multi-class LDA than for binary LDA.

4. Discussion

Due to its robustness and competitive speed, regularised LDA is an excellent candidate for classification problems involving many training-testing iterations. The analytical approach explored in this paper boosts the performance of least-squares models and multi-class LDA, particularly for high-dimensional data.

The simulations revealed a persistent speed gain using the analytical approach to cross-validation as opposed to the standard approach wherein the classifier is retrained on every training fold. Relative efficiency increases notably with the number of features. Furthermore, it increases with the number of cross-validation folds, and it decreases when the number of samples increases. Cross-validation was significantly faster, by up to 3 orders of magnitude (1000x faster) for binary LDA, and close to 4 orders of magnitude (10,000x faster) for multi-class LDA.

In line with this, the analytical approach consistently outperformed the standard approach in the EEG/MEG analysis. This was particularly prevalent in multi-class LDA, where for 1900 features, the analytical approach was between 1000x and 10,000x faster than the standard approach.

Why is relative efficiency higher for multi-class LDA than for binary LDA? A possible explanation is that multi-class LDA is more involved computa-

tionally since a generalised eigenvalue problem needs to be solved, whereas binary LDA requires only matrix inversion. Crucially, the analytical approach requires only a single matrix inversion in both the binary and the multi-class case. This warrants a larger computational benefit for multi-class LDA.

4.1. Is it just a trade-off between samples and features?

It is worth noting that the analytical approach does not simply trade in the number of samples (N) for the number of features (P). If both quantities are equal, e.g. N = P = 1000, binary LDA is about 10x faster than the standard approach for 10-fold cross-validation and about 100x faster for leave-one-out. For multi-class LDA and 10-fold cross-validation, relative efficiency is close to 3 (almost 1000x faster).

The complexity calculations of the analytical approach grows cubically with the number of test samples whereas the standard method grows cubically with the number of features. Consequently, the standard method and the analytical approach are at parity when the number of test samples is roughly equal to the number of features, i.e. $N/K \approx P$. One can deduce the rule of thumb that it is beneficial to use the analytical approach in cross-validation as soon as $P > N/K$. The approach becomes more efficient as K increases with the upper limit being leave-one-out (K = N).

4.2. What is the practical use of the analytical approach?

The analytical approach yields a significant increase in speed, but is this practically relevant for typical neuroimaging analyses? There are a number of scenarios in modern neuroimaging analyses wherein a large number of training-testing iterations is performed and hence the approach developed here can be useful:

- *Multi-dimensional data.* Sometimes, statistical analysis is repeatedly performed along multiple data dimensions. For instance, in time-frequency data, a classifier may be validated for every combination of time point and frequency. In time generalisation, a classifier is trained and tested at every combination of time points in a trial. In searchlight analysis [38], a classifier is validated on a local neighbourhood centered on a voxel, and this operation is repeated for all voxels.

- *Condition-rich designs.* Some experimental designs, often used in the context of Representational Similarity Analysis (RSA) [6], feature a large number of stimulus conditions. To build the Representational Dissimilarity Matrix, distances between each pair of conditions are required. Hence with C conditions, $C(C-1)/2$ cross-validations are required for every subject. Initially RSA was based on simple Pearson correlation between samples, but more recent work has increasingly focused on classifier-based approaches, including LDA classification accuracy and LDA-related measures such as Linear Discriminant Contrast (LDC) [39, 40].

- *Permutation testing.* For permutation testing, a classification regime needs to be repeated thousands of times. For instance, Stelzer et al. [18] developed a cluster test for fMRI data that involved repeated 100 classification analyses with permuted class labels for each searchlight position and every subject. The results were passed on to the second level to perform group inference. Similarly, in Allefeld et al. [17], permutations at the subject level are computed for deriving a minimum-statistic used in group inference.

4.3. LDA vs. other least-squares approaches

Although the analyses presented in this paper focus on LDA, all results readily extend to other least-squares methods such as linear regression and ridge regression. If the vector of class labels is replaced by a vector of continuous responses, then all equations and results apply equally. Furthermore, since multi-class LDA is closely related to Canonical Correlation Analysis (CCA) [22], speed-ups for CCA might be possible using a similar approach.

4.4. Analytical approach vs. kernel methods

Kernel-based methods such as Support Vector Machines [7] are based on a samples $\times$ samples kernel matrix $\mathbf{K}$ that can be thought of as representing pair-wise similarities between samples according to some non-linear similarity measure. The calculation of the kernel matrix is affected by the number of features, but once the matrix it is available, optimisation algorithms such as Dual Coordinate Descent for SVM [41] directly operate on the samples dimension rather than the feature dimension. For linear SVM, the kernel matrix comprises standard dot products, $\mathbf{K}_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$. There is a close relationship between the linear kernel and the hat matrix which consists of

the entries $\mathbf{H}_{ij} = \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{x}_j$. For $\lambda > 0$, this quantity is positive-definite and hence a valid dot product. In other words, the hat matrix is simply a linear kernel whereby the samples have been pre-whitened with respect to the regularised scatter matrix. If the covariance of the samples is normalised and spherical, we have $\mathbf{H} = \mathbf{K}$.

However, kernel methods such as SVM require iterative optimisation algorithms as well as optimisation of hyperparameters. The analytical formula for least-squares methods and multi-class LDA hence yields a computational advantage in many cases.

4.5. What about big data?

Due to increasing levels of data sharing and large-scale studies, cognitive neuroscience is on the verge of becoming a big data science [42, 43, 44, 45]. In a big data setting, both the number of samples and the number of features is extremely large. This poses a challenge to all statistical learning approaches. For least-squares models, this challenge is admittedly not resolved with the present contribution. However, the following measures can be used to cope with either too many samples or too many features.

- *Too many samples.* The principal problem is that for a very large number of samples (e.g. $> 100,000$) it might be impossible to store the hat matrix in memory. Since a kernel matrix has the same size as the hat matrix, a similar problem occurs in the optimisation of SVMs. In SVM, on-the-fly calculation of the required kernel matrix entries has been proposed as a solution [41]. If the number of features is small enough, the matrix $\mathbf{X}^\top \mathbf{X}$ can be stored in memory and submatrices of the hat matrix can be calculated on the fly. Furthermore, the submatrices $\mathbf{I} - \mathbf{H}_{\mathrm{Te}}$ that need to be inverted are roughly of size N/K. Hence, one can always find a K large enough such that these matrices are small enough to be invertible efficiently.

- *Too many features.* If the number of features is too large, it is impossible to store the scatter matrix $\mathbf{X}^\top \mathbf{X}$ in memory. Random projections can offer a solution to this problem. There is evidence that if $\mathbf{X} \in \mathbb{R}^{N \times P}$ is multiplied by a sparse matrix $\mathbf{A} \in \mathbb{R}^{P \times Q}$ with $Q \ll P$, the covariance structure of the original data is approximately preserved in the smaller, sparsified matrix $\mathbf{XA} \in \mathbb{R}^{N \times Q}$ [46]. This matrix can then be used instead of the scatter matrix.

An alternative approach that deals with both issues simultaneously is ensemble learning [2], wherein a large number of statistical models called weak learners is trained in parallel. Each model uses a subset of features and a subset of samples. If these subsets are small enough, even large datasets can be digested by the ensemble. Furthermore, since each weak learner is trained independently of the others, ensemble learning can be efficiencly parallelised on compute clusters.

4.6. Conclusion

For least-squares methods and multi-class LDA, an analytical approach to cross-validation allows for an increase of computation speed up to several orders of magnitude. The analytical approach enables least-squares methods and multi-class LDA to be used in high-dimensional feature spaces, particularly in the $P \gg N$ setting (many features, few samples) often encountered in neuroimaging data. Target applications in modern neuroimaging studies include multi-dimensional datasets, Representational Similarity Analysis, and permutation testing.

Appendix A. Relationship between linear regression and binary LDA

The regression problem Eq. (5) leads to the normal equations

$$\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}}\, \boldsymbol{\beta} = \widetilde{\mathbf{X}}\mathbf{y} \tag{A.1}$$

Recall that $\widetilde{\mathbf{X}}$ is the augmented data matrix consisting of the original data $\mathbf{X}$ and a column of 1's. Without loss of generality, one can assume that the samples in the data matrix $\mathbf{X}$ have been arranged as $\mathbf{X} = [\mathbf{X}_1; \mathbf{X}_2]$ such that samples corresponding to class 1 come first and samples corresponding to class 2 come last. The response vector $\mathbf{y} \in \mathbb{R}^N$ contains the numerical codes for the class labels. Class 1 is represented by $z_1 \in \mathbb{R}$, class 1 is represented by $z_2 \in \mathbb{R}$, class 2 is represented by $z_1 \neq z_2$. Accordingly, $\mathbf{y}$ consists of N_1 times z_1 followed by N_2 times z_2. Plugging this into Eq. (A.1) yields

$$\begin{bmatrix} \mathbf{X}_1^\top & \mathbf{X}_2^\top \\ \mathbb{1}_{N_1}^\top & \mathbb{1}_{N_2}^\top \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbb{1}_{N_1} \\ \mathbf{X}_2 & \mathbb{1}_{N_2} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1^\top & \mathbf{X}_2^\top \\ \mathbb{1}_{N_1}^\top & \mathbb{1}_{N_2}^\top \end{bmatrix} \begin{bmatrix} z_1 \mathbb{1}_{N_1} \\ z_2 \mathbb{1}_{N_2} \end{bmatrix} \tag{A.2}$$

Multiplying the matrices and using $\mathbf{X}^\top\mathbf{X} = \mathbf{S}_w + N_1\mathbf{m}_1\mathbf{m}_1^\top + N_2\mathbf{m}_2\mathbf{m}_2^\top$ one obtains

$$\begin{bmatrix} \mathbf{S}_w + N_1\mathbf{m}_1\mathbf{m}_1^\top + N_2\mathbf{m}_2\mathbf{m}_2^\top & N\,\overline{\mathbf{m}} \\ N\,\overline{\mathbf{m}}^\top & N \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} N_1 z_1 \mathbf{m}_1 + N_2 z_2 \mathbf{m}_2 \\ N_1 z_1 + N_2 z_2 \end{bmatrix} \quad \text{(A.3)}$$

with $\mathbf{m}_1, \mathbf{m}_2$, and $\overline{\mathbf{m}}$ as defined in Eq. (1). Solving the last row of the equation for b yields $b = N_1 z_1/N + N_2 z_2/N - \overline{\mathbf{m}}^\top\mathbf{w}$. Plugging this into the first equation in Eq. (A.3) yields

$$(\mathbf{S}_w + N_1\mathbf{m}_1^2 + N_2\mathbf{m}_2^2 - N\,\overline{\mathbf{m}}^2)\,\mathbf{w} = N_1\mathbf{m}_1 - N_2\mathbf{m}_2 - (N_1 z_1 + N_2 z_2)\,\overline{\mathbf{m}}$$

where $\mathbf{m}^2$ is short for $\mathbf{m}\mathbf{m}^\top$. Using $N\overline{\mathbf{m}} = N_1\mathbf{m}_1 + N_2\mathbf{m}_2$ and the relation $N_1 - N_1^2/N = (N_1 N_2)/N$ one obtains

$$(\mathbf{S}_w + \underbrace{\frac{N_1 N_2}{N}\,(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^\top}_{\mathbf{S}_b})\,\mathbf{w} = \frac{2N_1 N_2\,(z_1 - z_2)}{N}(\mathbf{m}_1 - \mathbf{m}_2) \quad \text{(A.4)}$$

The vector $\mathbf{S}_b\,\mathbf{w}$ is a multiple of $(\mathbf{m}_1 - \mathbf{m}_2)$, hence there exists $\alpha \in \mathbb{R}$ such that

$$\mathbf{S}_b\,\mathbf{w} = (\frac{2N_1 N_2\,(z_1 - z_2)}{N} - \alpha)\;(\mathbf{m}_1 - \mathbf{m}_2) \quad \text{(A.5)}$$

Inserting Eq. (A.5) in Eq. (A.4) and assuming that $\mathbf{S}_w$ is regular yields

$$\mathbf{S}_w\mathbf{w} = \alpha\;(\mathbf{m}_1 - \mathbf{m}_2) \Leftrightarrow \mathbf{w} = \alpha\;\mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

This proves that $\mathbf{w}$ in the linear regression approach is (up to scaling) identical to the LDA solution. Furthermore, the exact numerical codes z_1 and z_2 for the classes determine b and the scaling of $\mathbf{w}$, but they do not affect the direction of $\mathbf{w}$.

Appendix B. Ridge regularisation for binary LDA

In this section, the correspondence between the regularised LDA in Eq. (16) and ridge regression solution in Eq. (17) is established. To simplify the math, it is assumed that in $\mathbf{y}$, class 1 is coded as $+1$ and class 2 is coded as -1. The assertion is that regularised LDA can be cast in a least-squares framework using the normal equations

$$(\widetilde{\mathbf{X}}^\top \widetilde{\mathbf{X}} + \lambda \mathbf{I}_0)\, \boldsymbol{\beta} = \widetilde{\mathbf{X}} \mathbf{y} \tag{B.1}$$

where $\mathbf{I}_0$ is defined like in Eq. (17). Following the derivation in the previous section, one arrives at

$$\left(\begin{bmatrix} \mathbf{X}_1^\top & \mathbf{X}_2^\top \\ \mathbb{1}_{N_1}^\top & \mathbb{1}_{N_2}^\top \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbb{1}_{N_1} \\ \mathbf{X}_2 & \mathbb{1}_{N_2} \end{bmatrix} + \lambda \mathbf{I}_0 \right) \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1^\top & \mathbf{X}_2^\top \\ \mathbb{1}_{N_1}^\top & \mathbb{1}_{N_2}^\top \end{bmatrix} \begin{bmatrix} \mathbb{1}_{N_1} \\ -\mathbb{1}_{N_2} \end{bmatrix} \tag{B.2}$$

and finally

$$\begin{bmatrix} (\mathbf{S}_w + \lambda \mathbf{I}) + N_1 \mathbf{m}_1 \mathbf{m}_1^\top + N_2 \mathbf{m}_2 \mathbf{m}_2^\top & N\, \overline{\mathbf{m}} \\ N\, \overline{\mathbf{m}}^\top & N \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} N_1 \mathbf{m}_1 - N_2 \mathbf{m}_2 \\ N_1 - N_2 \end{bmatrix} \tag{B.3}$$

The rest of the proof follows the approach in the previous section, with $\mathbf{S}_w$ being replaced by $\mathbf{S}_w + \lambda \mathbf{I}$. This proves the normal equations in Eq. (B.1) correspond to regularised LDA.

Appendix C. Proof of lemma

Lemma 1. *Let $\mathbf{S}_b\, \mathbf{w} = \lambda\, \mathbf{S}_w\, \mathbf{w}$ be the generalised eigenvalue problem associated with a binary classification problem with unequal class means ($\mathbf{m}_1 \neq \mathbf{m}_2$) and let $\mathbf{S}_w$ be positive definite. Then there is one non-zero eigenvalue $\lambda = N_1\, N_2 / N\ (\mathbf{m}_1 - \mathbf{m}_2)^\top \mathbf{S}_w^{-1} (\mathbf{m}_1 - \mathbf{m}_2) > 0$. The associated eigenvector is proportional to $\mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$.*

Proof. Define $\Delta := \mathbf{m}_1 - \mathbf{m}_2$ and $\mathbf{w} := \mathbf{S}_w^{-1} \Delta$. Since $\mathbf{S}_w$ is regular, the generalised eigenvalue problem can be written as an ordinary eigenvalue problem $\mathbf{S}_w^{-1} \mathbf{S}_b\, \mathbf{w} = \lambda\, \mathbf{w}$. Then using Eq. (2) for $\mathbf{S}_b$ one obtains

$$\mathbf{S}_w^{-1}\mathbf{S}_b\mathbf{w} = \mathbf{S}_w^{-1}(\frac{N_1\,N_2}{N}\Delta\Delta^\top)\mathbf{w} = \mathbf{S}_w^{-1}\Delta(\underbrace{\frac{N_1\,N_2}{N}\,\Delta^\top\mathbf{S}_w^{-1}\Delta}_{:=\lambda}) = \lambda\mathbf{w},$$

hence $\mathbf{w}$ is an eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$ with eigenvalue λ. Since $\mathbf{S}_w^{-1}$ is positive definite, $\lambda > 0$. Since $\mathbf{S}_w^{-1}\mathbf{S}_b$ is of rank 1, all other eigenvalues are zero. □

Acknowledgements

I would like to thank Richard Henson for helpful comments.

References

References

[1] M. Mur, P. A. Bandettini, N. Kriegeskorte, Revealing representational content with pattern-information fMRI - An introductory guide, Social Cognitive and Affective Neuroscience 4 (2009) 101–109.

[2] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, in: The Elements of Statistical Learning, Springer New York Inc., New York, NY, USA, 2009.

[3] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (1936) 179–188.

[4] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, Y. Wang, The properties of high-dimensional data spaces: implications for exploring gene and protein expression data., Nature reviews. Cancer 8 (2008) 37–49.

[5] Y. Wang, D. J. Miller, R. Clarke, Approaches to working in high-dimensional data spaces: gene expression microarrays, British Journal of Cancer 98 (2008) 1023–1028.

[6] N. Kriegeskorte, M. Mur, P. Bandettini, Representational similarity analysis - connecting the branches of systems neuroscience, Frontiers in systems neuroscience 2 (2008) 4.

[7] C. Cortes, V. Vapnik, Support-Vector Networks, Machine Learning 20 (1995) 273–297.

[8] S. Lemm, B. Blankertz, T. Dickhaus, K. R. Müller, Introduction to machine learning for brain imaging, NeuroImage 56 (2011) 387–399.

[9] G. C. Cawley, N. L. Talbot, Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers, Pattern Recognition 36 (2003) 2585–2592.

[10] R. D. Cook, S. Weisberg, Residuals and influence in regression, Chapman and Hall, 1982.

[11] G. James, D. Witten, T. Hastie, R. Tibishirani, An Introduction to Statistical Learning, 2013.

[12] T. Pahikkala, J. Boberg, T. Salakoski, Fast n-Fold Cross-Validation for Regularized Least-Squares, in: Proceedings of SCAI0, pp. 83–90.

[13] R. B. Rao, G. Fung, R. Rosales, On the Dangers of Cross-Validation. An Experimental Evaluation, in: roceedings of the 2008 SIAM International Conference on Data Mining, pp. 588–596.

[14] J. H. Friedman, Regularized Discriminant Analysis, Journal of the American Statistical Association 84 (1989) 165–175.

[15] B. Blankertz, S. Lemm, M. Treder, S. Haufe, K. R. Müller, Single-trial analysis and classification of ERP components - A tutorial, NeuroImage 56 (2011) 814–825.

[16] T. Li, S. Zhu, M. Ogihara, Using discriminant analysis for multi-class classification: an experimental investigation, Knowledge and Information Systems 10 (2006) 453–472.

[17] C. Allefeld, K. Görgen, J.-D. Haynes, Valid population inference for information-based imaging: From the second-level t -test to prevalence inference, NeuroImage 141 (2016) 378–392.

[18] J. Stelzer, Y. Chen, R. Turner, Statistical inference and multiple testing correction in classification-based multi-voxel pattern analysis (MVPA): Random permutations and cluster size control, NeuroImage 65 (2013) 69–82.

[19] M. Ojala, G. C. Garriga, Permutation Tests for Studying Classifier Performance, Journal of Machine Learning Research 11 (2010) 1833–1863.

[20] S. L. Salzberg, On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach, Data Mining and Knowledge Discovery 1 (1997) 317–327.

[21] H. Jamalabadi, S. Alizadeh, M. Schönauer, C. Leibold, S. Gais, Classification based hypothesis testing in neuroscience: Below-chance level classification rates and overlooked statistical properties of linear parametric classifiers, Human Brain Mapping 37 (2016) 1842–1855.

[22] T. Hastie, A. Buja, R. Tibshirani, Penalized Discriminant Analysis, The Annals of Statistics 23 (1995) 73–102.

[23] C. R. Rao, The Utilization of Multiple Measurements in Problems of Biological Classification, 1948.

[24] D. G. Wakeman, R. N. Henson, A multi-subject, multi-modal human neuroimaging dataset, Scientific Data 2 (2015) 150001.

[25] C. M. Bishop, Pattern Recognition and Machine Learning, Journal of Electronic Imaging 16 (2007) 049901.

[26] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, 1998.

[27] M. S. Treder, A. K. Porbadnigk, F. Shahbazi Avarvand, K.-R. Müller, B. Blankertz, The LDA beamformer: Optimal estimation of ERP source time series using linear discriminant analysis, NeuroImage 129 (2016) 279–291.

[28] M. van Vliet, N. Chumerin, S. De Deyne, J. R. Wiersema, W. Fias, G. Storms, M. M. Van Hulle, Single-Trial ERP Component Analysis Using a Spatiotemporal LCMV Beamformer, IEEE Transactions on Biomedical Engineering 63 (2016) 55–66.

[29] M. van Vliet, M. M. Van Hulle, R. Salmelin, Exploring the Organization of Semantic Memory through Unsupervised Analysis of Event-related Potentials, Journal of Cognitive Neuroscience (2017) 1–12.

[30] S. Mika, Kernel Fisher Discriminants, Ph.D. thesis, Technische Universität Berlin, 2002.

[31] Z. Zhang, G. Dai, C. Xu, M. I. Jordan, Regularized Discriminant Analysis, Ridge Regression and Beyond, Journal of Machine Learning Research 11 (2010) 2199–2228.

[32] D. C. Hoaglin, R. E. Welsch, The Hat Matrix in Regression and ANOVA, The American Statistician 32 (1978) 17–22.

[33] A. N. A. N. Tikhonov, V. I. V. I. Arsenin, Solutions of ill-posed problems, Winston, 1977.

[34] A. Ng, Feature selection, L1 vs. L2 regularization, and rotational invariance, Twenty-first international conference on Machine learning - ICML '04 (2004) 78.

[35] J. Ye, Jieping, Least squares linear discriminant analysis, in: Proceedings of the 24th international conference on Machine learning - ICML '07, ACM Press, New York, New York, USA, 2007, pp. 1087–1093.

[36] C. H. Park, H. Park, A Relationship between Linear Discriminant Analysis and the Generalized Minimum Squared Error Solution, SIAM Journal on Matrix Analysis and Applications 27 (2005) 474–492.

[37] R. Oostenveld, P. Fries, E. Maris, J.-M. Schoffelen, FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data, Computational Intelligence and Neuroscience 2011 (2011) 1–9.

[38] N. Kriegeskorte, R. Goebel, P. Bandettini, Information-based functional brain mapping, Proceedings of the National Academy of Sciences 103 (2006) 3863–3868.

[39] A. Walther, H. Nili, N. Ejaz, A. Alink, N. Kriegeskorte, J. Diedrichsen, Reliability of dissimilarity measures for multi-voxel pattern analysis, NeuroImage 137 (2016) 188–200.

[40] J. Diedrichsen, N. Kriegeskorte, Representational models: A common framework for understanding encoding, pattern-component, and representational-similarity analysis, PLOS Computational Biology 13 (2017) e1005508.

[41] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: Proceedings of the 25th international conference on Machine learning - ICML '08, ACM Press, New York, New York, USA, 2008, pp. 408–415.

[42] R. A. Poldrack, K. J. Gorgolewski, Making big data open: data sharing in neuroimaging, Nature Neuroscience 17 (2014) 1510–1517.

[43] N. B. Turk-Browne, Functional Interactions as Big Data in the Human Brain, Science 342 (2013) 580–584.

[44] A. R. Ferguson, J. L. Nielson, M. H. Cragin, A. E. Bandrowski, M. E. Martone, Big data from small data: data-sharing in the 'long tail' of neuroscience, Nature Neuroscience 17 (2014) 1442–1447.

[45] S. Choudhury, J. R. Fishman, M. L. McGowan, E. T. Juengst, Big data, open science and the brain: lessons learned from genomics., Frontiers in human neuroscience 8 (2014) 239.

[46] E. Bingham, H. Mannila, Random projection in dimensionality reduction, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01, ACM Press, New York, New York, USA, 2001, pp. 245–250.