UCL DEPARTMENT OF
COMPUTER SCIENCE

# Research Note

RN/17/09

# Feature Adjustment in Kernel Space when using Cross-Validation

November 28, 2017

Anil Rao Janaina Mourao-Miranda

**Abstract**

Kernel methods are a powerful set of techniques for learning from data. One of the attractive properties of these techniques is that they rely only on a kernel function which provides the user-defined notion of similarity between two observations, to train the models. This report describes a strategy for evaluating kernel-based predictive models within a cross-validation framework when we also have a set of confounds which are used to 'adjust' the data based on their relationship with the data within the training sample. We show that for a linear kernel function, this adjustment can be performed in kernel space which removes the need to reload or retain the data in memory during the cross-validation procedure. Furthermore, we show how a similar strategy can be used with other kernel functions such as the Gaussian Radial Basis function.

## I. Introduction

Kernel methods are a powerful set of techniques for performing both unsupervised and supervised learning from data. The key component of these methods is the so-called kernel function $k(\mathbf{x}, \mathbf{x}')$, which defines the notion of similarity between two observations $\mathbf{x}$ and $\mathbf{x}'$ [1]. Many well-known machine learning algorithms, such as principal components analysis, support vector machines and ridge regression can be formulated as kernel-based methods, and they have been successfully used in a large number of applications.

This report considers the evaluation of kernel-based supervised learning algorithms in the context of predictive models in neuroimaging. Here, the observations contained in the data $\mathbf{X}$ are typically high-dimensional images, and we train a model to learn the relationship between the images and corresponding diagnostic labels or clinical scores contained in $\mathbf{y}$. Evaluation of the modelling procedure is performed using cross-validation, as is common when data is scarce [2]. Our focus is on the situation where we additionally have auxiliary data $\mathbf{C}$, consisting of variables that will affect the imaging data but which we do not want to influence the predictive model. Typical examples of such variables are age and gender, and we will refer to such variables as 'confounds' in this report. One way of building a predictive model in the presence of confounds is to 'regress out' or 'adjust' the imaging data by the confounds in order to remove their effects before model training [3]. This is typically performed by learning the relationship between the confounds and the imaging data, and then subtracting off the fitted values from the imaging data. However, if we want to do this within a cross-validation framework in which only the training sample is used to determine the confound-imaging relationship, it appears that we would need to retain or reload $\mathbf{X}$ into memory throughout the procedure.

The main contribution of this report is to show how this adjustment can be performed in kernel space, rather than in feature space, when using a linear kernel function. This enables the model to be evaluated using cross-validation without the need to retain or reload the high-dimensional dataset $\mathbf{X}$ into memory. We also show how we can adapt our approach for model evaluation using other kernels, such as the Gaussian Radial Basis function.

## II. Methods

### A. Kernel Methods

In this section, we describe how we can evaluate a kernel-based method if we wish to use the original, rather than adjusted, features during the modelling procedure. Subsequent sections will focus on the scenario in which feature adjustment is performed.

We assume that we have a dataset of $n$ observations $\mathbf{X}$ with corresponding targets $\mathbf{y}$. In a neuroimaging context the observations are often images, of dimensionality $p \gg n$, and the entries of $\mathbf{y}$ may be a categorical variable eg. a disease indicator or a continuous target such as a clinical score. We wish to evaluate the predictive accuracy of a kernel-based method when learning the relationship between the images and the target. For domains in which data is scarce (such as neuroimaging), this tends to be performed using resampling methods such as cross-validation, in which the dataset is repeatedly partitioned into a training sample which is used to build a model, and a test sample which is used to evaluate the model[1]. The final measure of model performance is given by the average of model predictive accuracy over all partitions.

Let us now consider a particular partition of $\mathbf{X}, \mathbf{y}$ into training and test samples $\mathbf{X}_1, \mathbf{y}_1$ and $\mathbf{X}_2, \mathbf{y}_2$, consisting of $n_1$ and $n_2$ observations respectively. Let us also assume that $\mathbf{y}$ is a continuous target and we wish to evaluate the predictive performance of Gaussian Process Regression (GPR), using kernel function $k$, on these data. The mean and covariance of the gaussian predictive distribution for the test sample is given by [4]:

$$\begin{aligned} \hat{\mathbf{y}}_2^{\mu} &= K(\mathbf{X}_2, \mathbf{X}_1) \left[ K(\mathbf{X}_1, \mathbf{X}_1) + \sigma^2 \mathbf{I} \right]^{-1} \mathbf{y}_1 \\ \hat{\mathbf{y}}_2^{\Sigma} &= K(\mathbf{X}_2, \mathbf{X}_2) - K(\mathbf{X}_2, \mathbf{X}_1) \left[ K(\mathbf{X}_1, \mathbf{X}_1) + \sigma^2 \mathbf{I} \right]^{-1} K(\mathbf{X}_2, \mathbf{X}_1)^T + \sigma^2 \mathbf{I} \end{aligned} \tag{1}$$

where $\sigma > 0$ is a hyper parameter. Here $K(\mathbf{X}_2, \mathbf{X}_1)$ denotes the $n_2 \times n_1$ matrix of evaluations of $k$ for all pairs of test and training observations, and we use this shorthand for all expressions of the form $K(\cdot, \cdot)$ throughout this report. In practice, the prediction for the test samples is taken to be the predictive mean $\hat{\mathbf{y}}_2^{\mu}$, while the covariance $\hat{\mathbf{y}}_2^{\Sigma}$ is used to give predictive confidence. As in all kernel-based methods, the predictions do not require the original data $\mathbf{X}$, but depend only on kernel matrices $K(\mathbf{X}_1, \mathbf{X}_1)$ and $K(\mathbf{X}_2, \mathbf{X}_1)$, while the Bayesian nature of GPR also provides predictive confidence which additionally requires $K(\mathbf{X}_2, \mathbf{X}_2)$.

In neuroimaging applications, the data tends to be extremely high-dimensional with many more features than observations, which increases the risk for overfitting. For this reason a simple linear kernel of the form

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}\mathbf{x}'^T \tag{2}$$

tends to be used when building predictive models, and we assume this particular kernel function from now until section II-D. The kernel matrix $K(\mathbf{X}, \mathbf{X}')$ of two sets of observations $\mathbf{X}$ and $\mathbf{X}'$ can then be simply determined as

$$K(\mathbf{X}, \mathbf{X}') = \mathbf{X}\mathbf{X}'^T \tag{3}$$

[1]The partitions in cross-validation have the added constraint that, across the partitions, the test samples are non-overlapping and their union is the complete dataset. These constraints are not necessary for the development in this report, indeed what we describe can be used for other evaluation schemes such as random splitting.

Since this kernel has no hyperparameters, we can use the strategy outlined in Algorithm 1 if we want to evaluate GPR, and indeed any kernel-based method, using cross-validation. (For clarity, Algorithm 1 does not mention the averaging of the predictive accuracies for the test samples.) The essential idea is to determine the kernel matrix $K(\mathbf{X}, \mathbf{X})$ using the entire dataset prior to

**Algorithm 1** Evaluation of Kernel-Based Methods (Linear Kernel)

Determine kernel matrix $K(\mathbf{X}, \mathbf{X}) = \mathbf{X}\mathbf{X}^T$ using entire dataset.
Release $\mathbf{X}$ from memory.
**for all** Cross-validation partitions **do**
  Extract kernel matrices $K(\mathbf{X}_1, \mathbf{X}_1)$, $K(\mathbf{X}_2, \mathbf{X}_1)$, $K(\mathbf{X}_2, \mathbf{X}_2)$ from $K(\mathbf{X}, \mathbf{X})$
  Plug kernel matrices into equation (1) to train model and give predictions for test sample
**end for**

cross-validation, and then release the dataset from memory. During cross-validation, the kernel matrices required for training and prediction can then be obtained by extracting the relevant entries from $K(\mathbf{X}, \mathbf{X})$. In this way, we minimise the requirements for retaining or reloading the high-dimensional dataset into memory during cross-validation, and it is indeed the strategy adopted by the Pattern Recognition in NeuroImaging Toolbox (PRoNTo) [5].[2]

### *B. Feature-Space adjustment*

Now consider the case in which our dataset additionally has corresponding confounds $\mathbf{C}$ of dimensionality $q$, and we wish to evaluate our model after adjusting the data by the confounds. A particular partition of the data during cross-validation now consists of training and test samples and corresponding confounds $\mathbf{C}_1$, $\mathbf{C}_2$ obtained from taking the analogous partition of $\mathbf{C}$.

In this section we show how the adjustment could be performed if we retain the original data $\mathbf{X}$ during cross-validation. In that case we could directly adjust the features in $\mathbf{X}_1$ and $\mathbf{X}_2$ using the relationship between the confounds and the data. We learn this relationship using only the training sample by performing linear least squares fits of the confounds $\mathbf{C}_1$ to $\mathbf{X}_1$ (assuming $q < n_1$). In this case the adjusted samples $\mathbf{X}_{1a}, \mathbf{X}_{2a}$ are given by:

$$\begin{aligned} \mathbf{X}_{1a} &= \mathbf{X}_1 - \mathbf{C}_1 \left[\mathbf{C}_1^T \mathbf{C}_1\right]^{-1} \mathbf{C}_1^T \mathbf{X}_1 \\ \mathbf{X}_{2a} &= \mathbf{X}_2 - \mathbf{C}_2 \left[\mathbf{C}_1^T \mathbf{C}_1\right]^{-1} \mathbf{C}_1^T \mathbf{X}_1 \end{aligned} \tag{4}$$

where, in a slight abuse of notation, we have added a column of ones to both $\mathbf{C}_1$ and $\mathbf{C}_2$. The above can be expressed more compactly as

$$\begin{aligned} \mathbf{X}_{1a} &= \mathbf{X}_1 - \mathbf{C}_1 P \mathbf{X}_1 \\ \mathbf{X}_{2a} &= \mathbf{X}_2 - \mathbf{C}_2 P \mathbf{X}_1 \end{aligned} \tag{5}$$

where $P = [\mathbf{C}_1^T \mathbf{C}_1]^{-1} \mathbf{C}_1^T$ is the pseudo-inverse of $\mathbf{C}_1$. Once we have the adjusted samples, we can determine kernel matrices $K(\mathbf{X}_{1a}, \mathbf{X}_{1a})$, $K(\mathbf{X}_{2a}, \mathbf{X}_{1a})$, $K(\mathbf{X}_{2a}, \mathbf{X}_{2a})$ using equation (3), and then plug them into equation (1) to give model predictions and predictive confidences for the test sample. Note that, due to the dependency of the adjusted samples on the training sample $\mathbf{X}_1$, these kernel matrices need to be recomputed for every partition during cross-validation. This contrasts with the situation when evaluating a model using the original features, where it was sufficient to determine the kernel matrix for the complete dataset prior to cross-validation.

### *C. Kernel-Space adjustment*

As described in the previous section, if we want to adjust the features directly during cross-validation, we will have to retain the data in memory and recompute the kernel matrices for every partition. Now we show how we can avoid retaining or reloading the original data into memory by rewriting the kernel matrices of the adjusted data in terms of kernel matrices of the original data and the confounds.

Firstly we derive the kernel matrix $K(\mathbf{X}_{1a}, \mathbf{X}_{1a})$:

$$\begin{aligned} K(\mathbf{X}_{1a}, \mathbf{X}_{1a}) &= \mathbf{X}_{1a} \mathbf{X}_{1a}^T \\ &= (\mathbf{X}_1 - \mathbf{C}_1 P \mathbf{X}_1)(\mathbf{X}_1 - \mathbf{C}_1 P \mathbf{X}_1)^T \\ &= (\mathbf{I} - \mathbf{C}_1 P)\, \mathbf{X}_1 \mathbf{X}_1^T\, (\mathbf{I} - \mathbf{C}_1 P)^T \\ &= (\mathbf{I} - \mathbf{C}_1 P)\, K(\mathbf{X}_1, \mathbf{X}_1)\, (\mathbf{I} - \mathbf{C}_1 P)^T \end{aligned} \tag{6}$$

[2]For some kernel-based methods (including GPR), PRoNTo actually uses a scaled linear kernel rather than the linear kernel in equation (2). The scaled linear kernel is discussed in section II-D1

We can see that $K(\mathbf{X}_{1a}, \mathbf{X}_{1a})$ is now expressed in terms of the confounds $\mathbf{C}_1$ and the kernel matrix of the original data $K(\mathbf{X}_1, \mathbf{X}_1)$. We can similarly write $K(\mathbf{X}_{2a}, \mathbf{X}_{1a})$, $K(\mathbf{X}_{2a}, \mathbf{X}_{2a})$ in terms of confounds and kernel matrices of the original data:

$$\begin{aligned} K(\mathbf{X}_{2a}, \mathbf{X}_{1a}) &= \mathbf{X}_{2a}\mathbf{X}_{1a}^T \\ &= (\mathbf{X}_2 - \mathbf{C}_2 P \mathbf{X}_1)(\mathbf{X}_1 - \mathbf{C}_1 P \mathbf{X}_1)^T \\ &= \mathbf{X}_2\mathbf{X}_1^T - \mathbf{X}_2\mathbf{X}_1^T \left[\mathbf{C}_1 P\right]^T - \mathbf{C}_2 P \mathbf{X}_1 \mathbf{X}_1^T + \mathbf{C}_2 P \mathbf{X}_1 \mathbf{X}_1^T \left[\mathbf{C}_1 P\right]^T \\ &= K(\mathbf{X}_2, \mathbf{X}_1) - K(\mathbf{X}_2, \mathbf{X}_1)\left[\mathbf{C}_1 P\right]^T - \mathbf{C}_2 P K(\mathbf{X}_1, \mathbf{X}_1) + \mathbf{C}_2 P K(\mathbf{X}_1, \mathbf{X}_1)\left[\mathbf{C}_1 P\right]^T \\ &= \left(K(\mathbf{X}_2, \mathbf{X}_1) - \mathbf{C}_2 P K(\mathbf{X}_1, \mathbf{X}_1)\right)\left(\mathbf{I} - \left[\mathbf{C}_1 P\right]^T\right) \end{aligned} \tag{7}$$

$$\begin{aligned} K(\mathbf{X}_{2a}, \mathbf{X}_{2a}) &= \mathbf{X}_{2a}\mathbf{X}_{2a}^T \\ &= (\mathbf{X}_2 - \mathbf{C}_2 P \mathbf{X}_1)(\mathbf{X}_2 - \mathbf{C}_2 P \mathbf{X}_1)^T \\ &= \mathbf{X}_2\mathbf{X}_2^T - \mathbf{X}_2\mathbf{X}_1^T \left[\mathbf{C}_2 P\right]^T - \mathbf{C}_2 P \mathbf{X}_1 \mathbf{X}_2^T + \mathbf{C}_2 P \mathbf{X}_1 \mathbf{X}_1^T \left[\mathbf{C}_2 P\right]^T \\ &= K(\mathbf{X}_2, \mathbf{X}_2) - K(\mathbf{X}_2, \mathbf{X}_1)\left[\mathbf{C}_2 P\right]^T - \mathbf{C}_2 P K(\mathbf{X}_2, \mathbf{X}_1)^T + \mathbf{C}_2 P K(\mathbf{X}_1, \mathbf{X}_1)\left[\mathbf{C}_2 P\right]^T \end{aligned} \tag{8}$$

The above shows that the kernel matrices of the adjusted features required by equation (1) can be determined by applying simple matrix operations to the corresponding kernel matrices of the original features and the confounds. This suggests the strategy in Algorithm 2 for evaluating a kernel-based method on adjusted features using cross-validation. As in Algorithm 1, we firstly compute the kernel matrix $K(\mathbf{X}, \mathbf{X})$ of the original data prior to cross-validation, and release the data $\mathbf{X}$ from memory. For a particular partition of data in cross-validation, we extract the kernel matrices as in Algorithm 1, but we then apply equations (6)-(8) to give the kernel matrices of adjusted data, which are then used for training and prediction. Hence, we can perform cross-validation using adjusted features without the need for retaining or reloading the high-dimensional dataset into memory.

**Algorithm 2** Evaluation of Kernel-Based Methods using adjusted features (Linear Kernel)

Determine kernel matrix $K(\mathbf{X}, \mathbf{X}) = \mathbf{X}\mathbf{X}^T$ using entire dataset.
Release $\mathbf{X}$ from memory.
**for all** Cross-validation partitions **do**
  Extract kernel matrices $K(\mathbf{X}_1, \mathbf{X}_1)$, $K(\mathbf{X}_2, \mathbf{X}_1)$, $K(\mathbf{X}_2, \mathbf{X}_2)$ from $K(\mathbf{X}, \mathbf{X})$
  Compute adjusted kernel matrices $K(\mathbf{X}_{1a}, \mathbf{X}_{1a})$, $K(\mathbf{X}_{2a}, \mathbf{X}_{1a})$, $K(\mathbf{X}_{2a}, \mathbf{X}_{2a})$ using equations (6)-(8).
  Plug adjusted kernel matrices into equation (1) to train model and give predictions for test sample
**end for**

### *D. Extensions to other Kernels*

We have described a strategy for performing adjustment in kernel space for the specific case in which model training and prediction uses the kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}\mathbf{x}'^T$. However, it may be that we wish to adjust the data in the same way as in section II-B, i.e. by determining the relationship between the confounds and the training sample $\mathbf{X_1}$, but then use a different kernel function $k^*$ to perform model training and prediction. Once again, we would like to avoid retaining or reloading the high-dimensional data $\mathbf{X}$ into memory during cross-validation.

In principle, this will be straightforward if the kernel function $k^*$ can be expressed in terms of evaluations of the kernel function $k$. Formally, if there is a function $f$ such that we can write the kernel $k^*(\mathbf{x}, \mathbf{x}')$ in the form

$$k^*(\mathbf{x}, \mathbf{x}') = f(k(\mathbf{x}, \mathbf{x}), k(\mathbf{x}, \mathbf{x}'), k(\mathbf{x}', \mathbf{x}')) \tag{9}$$

then we can use the strategy in Algorithm 3 to determine the adjusted kernel matrices for $k^*$. Once again, we do not need to retain or reload the original data $\mathbf{X}$ into memory during cross-validation, but compared to kernel adjustment with Algorithm 2, we have an extra step that requires us to apply $f$ to the entries of the kernel matrices produced by equations (6)-(8). In order to illustrate this approach, we now give some examples for different kernel functions $k^*$.

*1) Scaled Linear Kernel:* In this case we have $k^*(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda}\mathbf{x}\mathbf{x}'^T$, $\lambda > 0$, which can be rewritten in the functional form of equation (9) as

$$k^*(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda} k(\mathbf{x}, \mathbf{x}') \tag{10}$$

**Algorithm 3** Evaluation of Kernel-Based Methods using adjusted features (Linear & Other Kernels)

Determine kernel matrix $K(\mathbf{X},\mathbf{X}) = \mathbf{X}\mathbf{X}^T$ using entire dataset.
Release $\mathbf{X}$ from memory.
**for all** Cross-validation partitions **do**
  Extract kernel matrices $K(\mathbf{X}_1,\mathbf{X}_1)$, $K(\mathbf{X}_2,\mathbf{X}_1)$, $K(\mathbf{X}_2,\mathbf{X}_2)$ from $K(\mathbf{X},\mathbf{X})$
  Compute adjusted kernel matrices $K(\mathbf{X}_{1a},\mathbf{X}_{1a})$, $K(\mathbf{X}_{2a},\mathbf{X}_{1a})$, $K(\mathbf{X}_{2a},\mathbf{X}_{2a})$ using equations (6)-(8).
  Compute adjusted kernel matrices for $k^*$, $K^*(\mathbf{X}_{1a},\mathbf{X}_{1a})$, $K^*(\mathbf{X}_{2a},\mathbf{X}_{1a})$, $K^*(\mathbf{X}_{2a},\mathbf{X}_{2a})$ by applying equation (9).
  Plug adjusted kernel matrices for $k^*$ into equation (1) to train model and give predictions for test sample
**end for**

The adjusted kernel matrices for $k^*$ are then given by

$$\begin{aligned} K^*(\mathbf{X}_{1a},\mathbf{X}_{1a}) &= \frac{1}{\lambda}K(\mathbf{X}_{1a},\mathbf{X}_{1a}) \\ K^*(\mathbf{X}_{2a},\mathbf{X}_{1a}) &= \frac{1}{\lambda}K(\mathbf{X}_{2a},\mathbf{X}_{1a}) \\ K^*(\mathbf{X}_{2a},\mathbf{X}_{2a}) &= \frac{1}{\lambda}K(\mathbf{X}_{2a},\mathbf{X}_{2a}) \end{aligned} \tag{11}$$

*2) Polynomial Kernels:* If we have a polynomial kernel of the form $k^*(\mathbf{x},\mathbf{x}') = [1+\mathbf{x}\mathbf{x}'^T]^\alpha$, $\alpha > 0$, then it can be rewritten in the functional form of equation (9) as

$$k^*(\mathbf{x},\mathbf{x}') = [1+k(\mathbf{x},\mathbf{x}')]^\alpha \tag{12}$$

The adjusted kernel matrices for $k^*$ are then given by

$$\begin{aligned} K^*(\mathbf{X}_{1a},\mathbf{X}_{1a}) &= [\mathbf{1}_{n_1}^T\mathbf{1}_{n_1} + K(\mathbf{X}_{1a},\mathbf{X}_{1a})]^{\odot\alpha} \\ K^*(\mathbf{X}_{2a},\mathbf{X}_{1a}) &= [\mathbf{1}_{n_2}^T\mathbf{1}_{n_1} + K(\mathbf{X}_{2a},\mathbf{X}_{1a})]^{\odot\alpha} \\ K^*(\mathbf{X}_{2a},\mathbf{X}_{2a}) &= [\mathbf{1}_{n_2}^T\mathbf{1}_{n_2} + K(\mathbf{X}_{2a},\mathbf{X}_{2a})]^{\odot\alpha} \end{aligned} \tag{13}$$

where the notation $^{\odot\alpha}$ means the element-wise raising of each matrix entry to the power $\alpha$, and $\mathbf{1}_{n_1}$, $\mathbf{1}_{n_2}$ are row vectors of dimension $n_1$ and $n_2$ respectively.

*3) Gaussian Radial Basis Kernels:* A Gaussian Radial Basis kernel $k^*(\mathbf{x},\mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$, $\sigma > 0$, can be rewritten in the functional form of equation (9) in the following way:

$$\begin{aligned} k^*(\mathbf{x},\mathbf{x}') &= e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}} \\ &= e^{-\frac{\|\mathbf{x}\|^2+\|\mathbf{x}'\|^2-2\mathbf{x}\mathbf{x}'^T}{2\sigma^2}} \\ &= e^{-\frac{k(\mathbf{x},\mathbf{x})+k(\mathbf{x}',\mathbf{x}')-2k(\mathbf{x},\mathbf{x}')}{2\sigma^2}} \end{aligned} \tag{14}$$

The adjusted kernel matrices for $k^*$ are then given by

$$\begin{aligned} K^*(\mathbf{X}_{1a},\mathbf{X}_{1a}) &= e^{\odot\left[-\frac{\mathbf{d}_{1a}\mathbf{1}_{n_1}+[\mathbf{d}_{1a}\mathbf{1}_{n_1}]^T-2K(\mathbf{X}_{1a},\mathbf{X}_{1a})}{2\sigma^2}\right]} \\ K^*(\mathbf{X}_{2a},\mathbf{X}_{1a}) &= e^{\odot\left[-\frac{\mathbf{d}_{2a}\mathbf{1}_{n_1}+[\mathbf{d}_{1a}\mathbf{1}_{n_2}]^T-2K(\mathbf{X}_{2a},\mathbf{X}_{1a})}{2\sigma^2}\right]} \\ K^*(\mathbf{X}_{2a},\mathbf{X}_{2a}) &= e^{\odot\left[-\frac{\mathbf{d}_{2a}\mathbf{1}_{n_2}+[\mathbf{d}_{2a}\mathbf{1}_{n_2}]^T-2K(\mathbf{X}_{2a},\mathbf{X}_{2a})}{2\sigma^2}\right]} \end{aligned} \tag{15}$$

where $\mathbf{d}_{1a}$, $\mathbf{d}_{2a}$ are column vectors containing the diagonal elements of $K(\mathbf{X}_{1a},\mathbf{X}_{1a})$ and $K(\mathbf{X}_{2a},\mathbf{X}_{2a})$ respectively, and $\mathbf{1}_{n_1}$, $\mathbf{1}_{n_2}$ are defined as in equation (13). The notation $e^{\odot}$ means the element-wise exponential of each matrix entry.

## III. Conclusion

In this report, we have described strategies that facilitate the evaluation of kernel methods within a cross-validation framework. The focus was on situations where the features are adjusted according to their relationship with a set of confounds, and this relationship is determined using only the training sample. We showed that for a linear kernel, this adjustment can be performed in kernel space by applying simple matrix operations to the kernel matrices of the original data and the confounds, thereby removing the need for retaining or reloading the original data into memory during cross-validation. Furthermore, we demonstrated how a similar strategy could be used with other kernel functions such as the Gaussian Radial Basis function.

We conclude this report by stating that, while we have motivated the described approaches for evaluating predictive models in neuroimaging, they can also be applied in other problem domains in which we wish to use cross-validation and feature adjustment. Code for performing the kernel-space adjustment described in this paper will be made publicly available at https://github.com/awrao/kerneladjust.

## REFERENCES

[1] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001.

[3] J. Dukart, M. L. Schroeter, and K. Mueller, "Age correction in dementia–matching to a healthy brain." *PloS one*, vol. 6, no. 7, p. e22193, Jan. 2011. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3146486\&tool=pmcentrez\&rendertype=abstract

[4] C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press, 2006.

[5] J. Schrouff, M. J. Rosa, J. M. Rondina, A. F. Marquand, C. Chu, J. Ashburner, C. Phillips, J. Richiardi, and J. Mour??o-Miranda, "PRoNTo: Pattern recognition for neuroimaging toolbox," *Neuroinformatics*, vol. 11, no. 3, pp. 319–337, 2013.