Brain kernel: a new spatial covariance function for fMRI data

Anqi Wu[1†], Samuel A. Nastase[2,3], Christopher A. Baldassano[4], Nicholas B. Turk-Browne[5], Kenneth A. Norman[2,3], Barbara E. Engelhardt[6], Jonathan W. Pillow[2,3]

1 Center for Theoretical Neuroscience, Columbia University, New York City, NY, USA
2 Princeton Neuroscience Institute, Princeton University, Princeton, NJ, USA
3 Department of Psychology, Princeton University, Princeton, NJ, USA
4 Department of Psychology, Columbia University, New York City, NY, USA
5 Department of Psychology, Yale University, New Haven, CT, USA
6 Department of Computer Science, Princeton University, Princeton, NJ, USA

† Correspondence email: anqiwu.angela@gmail.com

Abstract

A key problem in functional magnetic resonance imaging (fMRI) is to estimate spatial activity patterns from noisy high-dimensional signals. Spatial smoothing provides one approach to regularizing such estimates. However, standard smoothing methods ignore the fact that correlations in neural activity may fall off at different rates in different brain areas, or exhibit discontinuities across anatomical or functional boundaries. Moreover, such methods do not exploit the fact that widely separated brain regions may exhibit strong correlations due to bilateral symmetry or the network organization of brain regions. To capture this non-stationary spatial correlation structure, we introduce the *brain kernel*, a continuous covariance function for whole-brain activity patterns. We define the brain kernel in terms of a continuous nonlinear mapping from 3D brain coordinates to a latent embedding space, parametrized with a Gaussian process (GP). The brain kernel specifies the prior covariance between voxels as a function of the distance between their locations in embedding space. The GP mapping warps the brain nonlinearly so that highly correlated voxels are close together in latent space, and uncorrelated voxels are far apart. We estimate the brain kernel using resting-state fMRI data, and we develop an exact, scalable inference method based on block coordinate descent to overcome the challenges of high dimensionality (10-100K voxels). Finally, we illustrate the brain kernel's usefulness with applications to brain decoding and factor analysis with multiple task-based fMRI datasets.

Keywords: Brain kernel, Gaussian process, latent variable model, brain decoding, factor modeling, resting-state fMRI, task fMRI.

1 Introduction

An important problem in neuroscience is to characterize the covariance of high-dimensional neural activity. Understanding this covariance structure could provide insight into the brain's functional organization and help regularize estimates of encoding or decoding models. Although advances have been made in both theory and methodology for estimating large covariance [1, 2] and precision matrices [3, 4], few methods have been designed with the particular challenges of functional magnetic resonance imaging (fMRI) data in mind (see [5] for an exception).

One of the challenges of modeling the covariance of fMRI data is that the spatial discretization of the brain may differ across experiments. FMRI measures blood oxygenation level dependent (BOLD) signals in discrete spatial regions called "voxels". Each voxel represents a tiny cube of brain tissue. Although brains are typically registered to an anatomical template in a standard space, such as the volumetric

Montreal Neurological Institute (MNI) template or the surface-based template used by HCP [6], these spaces differ in resolution and geometry [7]. In many cases, brains are aligned onto the "same" 3D space but with different voxel coordinates. A covariance matrix for one set of voxels cannot be applied to data registered to a different set of voxels. Thus, modeling the covariance of fMRI data presently requires a new covariance matrix to be constructed whenever a different set of voxels is used.

A second challenge for fMRI covariance estimation is spatial nonstationarity. Standard spatial smoothing models assume that correlation falls off as a function of the Euclidean distance between voxels. In real brains, however, correlation patterns depend on relationships to anatomical and functional boundaries, and may exhibit strong dependencies over long distances due to bilateral symmetry and the network organization of brain regions.

To address these challenges, we propose the *brain kernel*, a continuous covariance function for whole-brain fMRI data. This function arises from a generative model of fMRI data, and seeks to describe covariance of neural signals across the entire brain [8]. Specifically, the brain kernel defines, for any finite set of n voxel locations in the brain, a positive definite $n \times n$ covariance matrix over n-dimensional vectors of neural activity at those locations. The brain kernel improves upon prior work by i) capturing fMRI voxel covariance matrices for any registration reference, and ii) capturing spatial nonstationarity through a nonlinear latent manifold.

Our approach uses a Gaussian process to parametrize a continuous nonlinear mapping from 3D brain coordinates to a latent embedding space, such that correlations in neural activity fall off as a fixed function of distance in the latent space. Thus, the nonlinear function seeks to warp the 3D brain in order to place locations with correlated neural activity at nearby locations in the latent space. Locations with uncorrelated activity, conversely, are mapped to more distant points in the latent space, even if they are physically close together in the brain.

The paper is organized as follows. In Sec. 2 we provide a brief overview of Gaussian process models. In Sec. 3, we formally introduce the brain kernel model for fMRI data. In Sec. 4, we describe an efficient inference method for fitting the brain kernel, and illustrate the challenges and benefits of an exact inference method using simulated and real fMRI datasets. In Sec. 5, we describe the brain kernel fit to whole-brain resting-state fMRI data. Finally, in Sec. 6, we demonstrate the usefulness of the inferred brain kernel with applications to decoding and factor modeling.

2 Mathematical background

Before introducing the brain kernel model, we briefly review the mathematical building blocks for Gaussian process models.

2.1 Gaussian processes (GPs)

Gaussian processes provide a flexible and tractable prior distribution over nonlinear functions [9]. A GP is parametrized by a mean function $m(\mathbf{x})$, which specifies the mean value of the function $f(\mathbf{x})$ at input point $\mathbf{x}$, and a covariance function $k(\mathbf{x}, \mathbf{x}')$, which specifies $\mathrm{cov}\big(f(\mathbf{x}), f(\mathbf{x}')\big)$, the covariance between the function values $f(\mathbf{x})$ and $f(\mathbf{x}')$, for any pair of inputs $\mathbf{x}$ and $\mathbf{x}'$.

Technically a GP is a random process for which the values taken at any finite set of input points has a well-defined multivariate Gaussian distribution. The mean and covariance of that Gaussian are given by evaluating the mean and covariance functions at the corresponding set of input points. Let $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ denote a collection of n points in the input domain. If a function f has a Gaussian process distribution, $f \sim \mathcal{GP}(m, k)$, then the vector of function values $\mathbf{f} = \big(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\big)^\top$ has a multivariate Gaussian

distribution:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}), \tag{1}$$

where $\mathbf{m} = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n))^\top$ is the mean vector, and $\mathbf{K}$ is the $(n \times n)$ covariance matrix whose i, j'th element is $k(\mathbf{x}_i, \mathbf{x}_j)$.

2.2 GP regression

A common application of GPs is to predict function values at test points given a set of training data consisting of observed inputs and function values. In GP regression, these predictions come from the conditional distribution over unknown function values given the observed values.

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ denote a set of n input points and let $\mathbf{f} = \big(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\big)^\top$ denote the vector of function values observed at these points. Here we assume the observed data is noiseless, and we will consider noisy observations in Sec. 3. We consider a set of n^* novel input points $\mathbf{X}_* = (\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+n^*})$, for which we would like to predict the corresponding (unobserved) function values, denoted $\mathbf{f}_* = \big(f(\mathbf{x}_{n+1}), \ldots, f(\mathbf{x}_{n+n^*})\big)^\top$.

The GP gives us the following joint prior distribution over the observed and unobserved function values:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m} \\ \mathbf{m}_* \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{nn} & \mathbf{K}_{n*} \\ \mathbf{K}_{*n} & \mathbf{K}_{**} \end{bmatrix} \right), \tag{2}$$

where $\mathbf{m}_* = \big(m(\mathbf{x}_{n+1}), \ldots, m(\mathbf{x}_{n+n^*})\big)^\top$ is the mean for novel test points in $\mathbf{X}_*$, and matrices $\mathbf{K}_{n*}$, $\mathbf{K}_{*n}$, and $\mathbf{K}_{**}$ are of size $(n \times n^*)$, $(n^* \times n)$, and $(n^* \times n^*)$, respectively, formed by evaluating the covariance function k at the relevant points in $\mathbf{X}$ and $\mathbf{X}_*$.

By applying the standard formula for Gaussian conditional distributions, we obtain the following conditional distribution of $\mathbf{f}^*$ given $\mathbf{f}$:

$$\mathbf{f}_* | \mathbf{f} \sim \mathcal{N}\big(\mu(\mathbf{X}_*), \sigma^2(\mathbf{X}_*)\big), \tag{3}$$

where mean and covariance are given by

$$\mu(\mathbf{X}_*) = \mathbf{K}_{*n}\mathbf{K}_{nn}^{-1}(\mathbf{f} - \mathbf{m}) + \mathbf{m}_*, \tag{4}$$

$$\sigma^2(\mathbf{X}_*) = \mathbf{K}_{**} - \mathbf{K}_{*n}\mathbf{K}_{nn}^{-1}\mathbf{K}_{n*}. \tag{5}$$

GP regression uses eq. 4, the posterior mean of the function given the training data, to predict function values at test points $\mathbf{X}_*$ given the training data $\{\mathbf{X}, \mathbf{f}\}$.

Although we have assumed so far that the function f is scalar-valued, we can extend the GP regression framework to vector-valued functions by using a separate GP for each output dimension of f.

2.3 Gaussian process latent variable model (GPLVM)

The Gaussian process latent variable model (GPLVM) [10] extends the GP regression model by treating the inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ as latent variables, distributed independently with standard normal distributions: $\mathbf{x}_i \sim \mathcal{N}(0, I)$, for $i = 1, \ldots, n$. Unlike the regression setting, the goal of the GPLVM is to account for structure of an observed dataset in terms of a set of locations in a latent space, which are transformed by a smooth function to generate the observed function values. The primary quantity of interest in the GPLVM is therefore $P(\mathbf{X}|\mathbf{f})$, the conditional distribution over the latent variables given the observations.

The smoothness induced by the GP prior distribution ensures a smooth mapping from latent variables to observed data. This means that when observed values $\mathbf{f}_i$ and $\mathbf{f}_j$ are close together, they are likely to arise from from points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are close together in the latent space. Conversely, if observed values $\mathbf{f}_i$ and $\mathbf{f}_j$ are far apart, smoothness of f implies that the corresponding input values lie far apart in the latent space.

In the following, we will derive the brain kernel model, which extends the GPLVM to allow latent locations arise from a smooth nonlinear transformation of the 3D voxel locations in the brain. The brain kernel model can also be seen as a "deep Gaussian process" model [11] with two layers (see Appendix A for complete details).

3 The brain kernel model

Here we introduce the brain kernel (BK) model, which is a probabilistic model of fMRI measurements at an arbitrary set of 3D spatial voxel locations. The brain kernel itself is a covariance function for neural activity that arises under this BK model, which we will infer from registered fMRI data.

3.1 Nonlinear embedding function

The first component of the brain kernel model is a nonlinear function, $f : \mathbb{R}^3 \longrightarrow \mathbb{R}^d$, which provides a continuous nonlinear mapping from 3D brain coordinates to a d-dimensional latent embedding space. The goal of this mapping is to embed brain regions with similar activity at nearby locations in the embedded space. Typically, we consider $d > 3$, so that the embedding is higher-dimensional than the three physical dimensions of the brain. This gives the embedding flexibility to capture complex non-smooth dependencies between brain regions. One example of such a dependency is the functional symmetry of the two hemispheres [12–14], which suggests that one might wish to map symmetric points on the two hemispheres to nearby points in the embedded latent space. This would not be possible with a continuous mapping in three dimensions. But, in four dimensions, one can fold the three-dimensional brain along the fourth dimension, analogous to the way that folding a 2D brain slice along the mid-line would allow for close alignment of symmetric points from the two hemispheres.

Let $\mathbf{x} \in \mathbb{R}^3$ denote an input vector, specifying the three-dimensional location of a voxel in the brain, and let $\mathbf{z} \in \mathbb{R}^d$ denote the output of f, so that $\mathbf{z} = f(\mathbf{x})$ is the d-dimensional embedding location of a voxel at $\mathbf{x}$. Thus, for a set of voxel locations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, the embedded locations in the latent space are $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n) = \big(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\big)$.

To impose smoothness on the embedding function, we place a GP prior on f. We use a linear mean function, $m(\mathbf{x}) = \mathbf{B}\mathbf{x}$, where $\mathbf{B}$ is a $d \times 3$ matrix. This choice ensures that the embedding defaults to a linearly stretched version of the brain in the absence of likelihood terms. Because f is a vector function with outputs of dimension d, the mean function output is also d-dimensional. For the covariance function, we use a Gaussian or "radial basis function" (RBF) covariance for each output dimension of f:

$$k_f(\mathbf{x}_i, \mathbf{x}_j) = r \ \exp\left(-\tfrac{1}{2\delta^2}||\mathbf{x}_i - \mathbf{x}_j||_2^2\right). \tag{6}$$

The hyperparameters governing this covariance function consist of a marginal variance r and a length scale δ, which control the range and smoothness of f, respectively. The GP prior over each output dimension of f can therefore be written as

$$f_j(\cdot) \sim GP(\mathbf{b}_j, k_f), \tag{7}$$

where $f_j(\cdot)$ denotes the jth output dimension of the function $f(\cdot)$, and $\mathbf{b}_j$ is the jth row of the matrix $\mathbf{B}$. The prior is therefore governed by a set of hyperparameters denoted $\theta_f = \{\mathbf{B}, r, \delta\}$. Thus, all output

dimensions of the function f are assumed *a priori* independent with the same covariance function and differing mean functions.

This GP prior over the function f implies a multivariate normal prior over any set of embedded voxel locations $\mathbf{Z}$. Let $\mathbf{z}_j$ denote the jth latent embedding of the entire set of brain voxels in the training data. Then the prior over $\mathbf{z}_j$ given the true voxel locations $\mathbf{X}$ is

$$p(\mathbf{z}_j|\mathbf{X}) = \mathcal{N}(\mathbf{b}_j\mathbf{X}, \mathbf{K}), \tag{8}$$

where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the covariance matrix with the i, jth element given by $k(\mathbf{x}_i, \mathbf{x}_j)$ (eq. 6).

3.2 From embedding space to neural activity

The second component of the brain kernel model is a probability distribution over neural activity as a function of locations in embedding space. Our modeling assumption is that neural activity changes smoothly as a function of locations in embedding space, or equivalently, that correlations in neural activity decrease smoothly with distance in latent space. We formalize this assumption using the brain kernel, which provides a mapping from latent embedding locations to a covariance matrix for neural activity.

Let $\mathbf{v} \in \mathbb{R}^n$ denote a vector of neural activity from n voxels with positions $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and latent embedding locations $\mathbf{Z} = \big(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\big)$. The BK model assumes this neural activity vector has a multivariate Gaussian distribution with zero mean and covariance determined by the brain kernel:

$$\mathbf{v} \sim \mathcal{N}(0, \mathbf{C}_{nn}), \tag{9}$$

where

$$\mathbf{C}_{nn} = \begin{bmatrix} \kappa_{BK}(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa_{BK}(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ \kappa_{BK}(\mathbf{x}_n, \mathbf{x}_1) & \cdots & \kappa_{BK}(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \tag{10}$$

is the covariance matrix, which results from applying the brain kernel $\kappa_{BK}(\cdot, \cdot)$ to every pair of voxel locations $(\mathbf{x}_i, \mathbf{x}_j)$ in the set $\mathbf{X}$.

The brain kernel itself is the bivariate function $\kappa_{BK} : \mathbb{R}^3 \times \mathbb{R}^3 \longrightarrow \mathbb{R}$ from pairs of 3D voxel locations to a covariance of neural activity at those pairs of locations:

$$\kappa_{BK}(\mathbf{x}_i, \mathbf{x}_j) = \rho \exp\left(-\tfrac{1}{2}||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2^2\right) = \rho \exp\left(-\tfrac{1}{2}||\mathbf{z}_i - \mathbf{z}_j||_2^2\right), \tag{11}$$

where ρ is the marginal variance. The brain kernel therefore specifies a positive semidefinite covariance matrix for neural activity at any set of 3D voxel locations, which is a function of the embedded latent locations of those voxels via the nonlinear function f. The brain kernel model transforms the data representation from voxel space to the latent embedding space, and this transformation explicitly estimates the covariance over neural activity (Figure 1).

3.3 From neural activity to BOLD signal

Next, we assume that the experimenter does not directly measure the neural activity vector $\mathbf{v}$, but instead receives measurements corrupted by independent Gaussian noise. If $\mathbf{y}$ denotes the vector of fMRI measurements, we assume $y_i = v_i + \xi_i$, where i is the index for voxels and $\xi_i \sim \mathcal{N}(0, \sigma^2)$ represents measurement noise. This induces the following marginal distribution over fMRI measurements given the embedding:

$$\mathbf{y}|f \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_{nn} + \sigma^2 I_n). \tag{12}$$

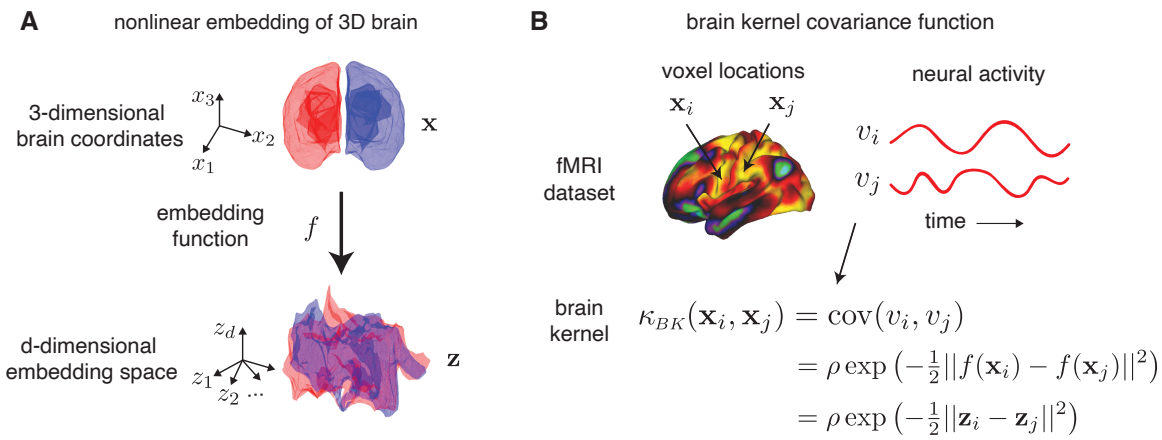

Fig 1. The diagram of the brain kernel model. **A**. The nonlinear latent embedding of the 3D coordinates into a d-dimensional latent space using a function f. This f function is sampled from a GP prior. **B**. Given the voxel locations and the BOLD activity of these two locations in an fMRI dataset (top right), our goal is to construct the brain kernel with the locations as inputs that matches their covariance of the BOLD activity (bottom right). The covariance is equal to the Euclidean-based kernel of the voxels' embedding in the latent space.

Note however that the brain kernel generates $\mathbf{C}_{nn}$, the covariance of the underlying neural activity $\mathbf{v}$, as opposed to the covariance of the noisy fMRI measurements $\mathbf{y}$; the covariance of these measurements is $(\mathbf{C}_{nn} + \sigma^2\mathbf{I}_n)$. For simplicity, we will omit the subscript in $\mathbf{C}_{nn}$ in the following text and write simply $\mathbf{C}$.

The full set of hyperparameters governing the brain kernel model are therefore $\theta = \{\mathbf{B}, r, \delta, \rho, \sigma^2\}$, where $\{\mathbf{B}, r, \delta\}$ describe the nonlinear embedding function, ρ is the marginal variance of neural activity, and σ^2 is the variance of additive Gaussian noise.

4 Inference methods

To fit the brain kernel model, we estimate the latent embeddings of all voxels $\mathbf{Z}$ as well as the model hyperparameters $\theta = \{\mathbf{B}, r, \delta, \rho, \sigma^2\}$ given a time series of whole-brain fMRI measurements $\mathbf{Y}$ as well as voxels' 3D locations $\mathbf{X}$. More specifically, $\mathbf{Y} = (\mathbf{y}_1, \ldots \mathbf{y}_T) \in \mathbb{R}^{n \times T}$, where $\mathbf{y}_t \in \mathbb{R}^n$ refers to the vector of fMRI measurements at time index $t \in \{1, \ldots, T\}$. Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{3 \times n}$, where $\mathbf{x}_i \in \mathbb{R}^3$ and $i \in \{1, \ldots, n\}$, denote the set of n 3D voxel locations for this dataset. Let $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n) = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) \in \mathbb{R}^{d \times n}$, where $\mathbf{z}_i \in \mathbb{R}^d$ denotes the set of n voxels' d-dimensional latent representations.

We propose two empirical estimators: penalized least squares (PLS) and maximum a posteriori (MAP). Briefly, PLS formulates an objective function by minimizing the squared error between the sample covariance of the data $\mathrm{cov}(\mathbf{Y})$, and the model-defined covariance $\mathbf{C}$ (eq. 10). MAP solves the problem using conventional Bayesian probabilistic inference. Since we have already defined the distribution to generate fMRI measurements from the latent embeddings (eq. 12) and the prior for the latent embeddings (eq. 8), we can estimate the latent embeddings $\mathbf{Z}$ using MAP methods. The goal of both inference methods is to find the latent embedding locations $\mathbf{Z}$ and the model hyperparameters such that the neural activity covariance $\mathbf{C}$ resembles the sample covariance of $\mathbf{Y}$ as closely as possible.

4.1 Penalized least squares (PLS) estimation

Estimating large covariance matrices is a fundamental problem in modern multivariate analysis. Researchers attempt to find an estimator that resembles the sample covariance while also satisfying structural assumptions about the data [15, 16]. In prior work, Fan et al. [16] showed that a generalized thresholding covariance estimator can be cast as a penalized least squares (PLS) problem:

$$\hat{\mathbf{C}} = \underset{\mathbf{C}}{\operatorname{argmin}} \left\{ ||\mathbf{S} - \mathbf{C}|| + R(\mathbf{C}) \right\}, \tag{13}$$

where $R(\cdot)$ is a penalty function that imposes structure on the covariance matrix $\mathbf{C}$, and $\mathbf{S}$ is the sample covariance of measured neural activity, defined as

$$\mathbf{S} = \frac{1}{T-1} \sum_{t=1}^{T} (\mathbf{y}_t - \bar{\mathbf{y}})(\mathbf{y}_t - \bar{\mathbf{y}})^\top, \quad \bar{\mathbf{y}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{y}_t. \tag{14}$$

Sparsity in $\mathbf{C}$ is often encoded using a shrinkage penalty. However, we abandon sparsity in the brain kernel because the covariance of brain activity may have dense structures. Instead, we regularize the latent subspace of the covariance matrix using the normal prior on the latent embedding $\mathbf{Z}$ (eq. 8), which is a Bayesian regularization of the log likelihood with the form tr $\left[(\mathbf{Z} - \mathbf{BX})\mathbf{K}^{-1}(\mathbf{Z} - \mathbf{BX})^\top\right]$. This is equivalent to an l_2-norm penalty on $\mathbf{K}^{-\frac{1}{2}}(\mathbf{Z} - \mathbf{BX})^\top$. Including the noise variance term σ^2, the loss function for the empirical estimator is

$$\mathcal{L}_{\text{PLS}}(\mathbf{Z}, \theta) = ||\mathbf{S} - \mathbf{C} - \sigma^2 \mathbf{I}_n||_2^2 + \text{tr}\left[(\mathbf{Z} - \mathbf{BX})\mathbf{K}^{-1}(\mathbf{Z} - \mathbf{BX})^\top\right]. \tag{15}$$

Minimizing $\mathcal{L}_{\text{PLS}}$ w.r.t. $\mathbf{Z}$ and θ, we derive the empirical PLS estimators $\hat{\mathbf{Z}}_{\text{PLS}}$ and θ_{PLS}. Here, the least square term estimates the embedding $\mathbf{Z}$ using the sample covariance, while the second term regularizes $\mathbf{Z}$ with the kernel matrix $\mathbf{K}$ and mean values $\mathbf{BX}$ constructed from $\mathbf{X}$.

4.2 Maximum a posteriori (MAP) estimation

Another common approach to estimating covariance matrices uses maximum likelihood methods. We have already defined the data distribution $\mathbf{Y}$ given the latent embedding $\mathbf{Z}$ in eq. 12 and described the prior over $\mathbf{Z}$ in eq. 8.

The joint distribution is then

$$\begin{aligned} p(\mathbf{Y}, \mathbf{Z}|\mathbf{X}, \theta) &= \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{Z}, \rho, \sigma^2) \prod_{j=1}^{d} p(\mathbf{z}_j|\mathbf{X}, \mathbf{B}, r, \delta) && (16) \\ &= \prod_{t=1}^{T} \mathcal{N}(\mathbf{y}_t|0, \mathbf{C} + \sigma^2 \mathbf{I}_n) \prod_{j=1}^{d} \mathcal{N}(\mathbf{z}_j|\mathbf{b}_j \mathbf{X}, \mathbf{K}), && (17) \end{aligned}$$

where $\mathbf{C}$ is a function of $\mathbf{Z}$ and ρ. Thus, the loss function for the maximum a posteriori (MAP) estimator is

$$\begin{aligned} \mathcal{L}_{\text{MAP}}(\mathbf{Z}, \theta) &= -\log p(\mathbf{Y}, \mathbf{Z}|\mathbf{X}, \theta) \\ &= \text{tr}\left[(\mathbf{C} + \sigma^2 \mathbf{I}_n)^{-1}\mathbf{S}\right] + T\log|\mathbf{C} + \sigma^2 \mathbf{I}_n| + \text{tr}\left[(\mathbf{Z} - \mathbf{BX})\mathbf{K}^{-1}(\mathbf{Z} - \mathbf{BX})^\top\right]. && (18) \end{aligned}$$

Minimizing $\mathcal{L}_{\text{MAP}}$ w.r.t. $\mathbf{Z}$ and θ, we derive the MAP estimators $\hat{\mathbf{Z}}_{\text{MAP}}$ and θ_{MAP}.

4.3 Exact inference by block coordinate descent

Calculating the log posterior (eq. 18) for MAP inference has a computational complexity of $O(n^3)$, due to the need to compute the inverse and the determinant of the covariance matrix. This cost is often impractical in fMRI settings, where the number of voxels n may be on the order of thousands to hundreds of thousands.

To optimize $\mathbf{Z}$ and the model hyperparameters, we could use gradient descent or Newton's method as the optimizer; however, this approach is computationally impractical. Thus, we need to consider a scalable inference method. Existing scalable inference methods for large datasets [17–19] exploit low-rank approximations to the full Gaussian process. However, these approximations suffer from a loss of accuracy in covariance estimation. Thus, we develop a block coordinate descent (BCD) algorithm as an exact inference method for the brain kernel model (see Methods, Algorithm 1). Coordinate descent has been successfully applied to solve penalized regression models [20], to estimate covariance graphical lasso models [21], and to compute large-scale sparse inverse covariance matrices [4]. Our PLS and MAP estimators are non-convex smooth functions. We apply an iterative block coordinate descent method solved by the proximal Newton approach [22]. Given such a scalable optimizer, we alternate between optimizing $\mathbf{Z}$ and the hyperparameters using either eq. 15 (PLS) or eq. 18 (MAP) as the objective loss function. More details about the optimization can be found in the Methods section.

In practice, we used the PLS estimate to initialize the MAP estimate, as PLS optimization is faster and requires less memory, but the MAP estimate is more principled and achieves a higher accuracy. We used the BCD algorithm to optimize both the PLS and MAP objectives.

4.4 Predicting activity for new voxels

Although we fit the brain kernel model to data collected with a particular grid of voxels, our framework allows us to apply the model to fMRI measurements collected using different voxel grids. To do so, we use the fact that the brain kernel is defined using a Gaussian process; the mean of this GP provides a smooth mapping from 3D voxel space to the d-dimensional latent embedding space, which can be evaluated at any 3D brain locations. We obtain the embedding location of a novel 3D voxel location $\mathbf{x}^*$ using the posterior mean of this GP:

$$\mathbf{z}^* = \mathbf{B}\mathbf{x}^* + (\mathbf{Z} - \mathbf{B}\mathbf{X})\mathbf{K}^{-1}\mathbf{k}^*, \tag{19}$$

where $\mathbf{k}^* = [k_f(\mathbf{x}^*, \mathbf{x}_1), \cdots, k_f(\mathbf{x}^*, \mathbf{x}_n)]^\top$ represents the vector formed by evaluating the RBF covariance function for the voxel at location $\mathbf{x}^*$ and all the observed voxels in $\mathbf{X}$. The brain kernel for any arbitrary pair of voxel locations in the test set $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ is therefore given by:

$$\kappa_{BK}(\mathbf{x}_i^*, \mathbf{x}_j^*) = \rho \exp(-\frac{1}{2}||\mathbf{z}_i^* - \mathbf{z}_j^*||_2^2), \tag{20}$$

with $\mathbf{z}_i^*$ and $\mathbf{z}_j^*$ the corresponding latent embeddings (eq. 19).

4.5 Synthetic experiments

To illustrate the performance of our proposed inference method to optimize the brain kernel objective function, we began with an application to simulated data, where the ground-truth embedding is known. We first compared our block coordinate descent (BCD) method (Algorithm 1), which maximizes the exact log evidence (eq. 18), with two variational inference methods that optimize a lower bound on log evidence: an inducing-point method [17], and a dynamical variational inference method (dynamic VIP[1]) [18].

[1] https://github.com/SheffieldML/GPmat

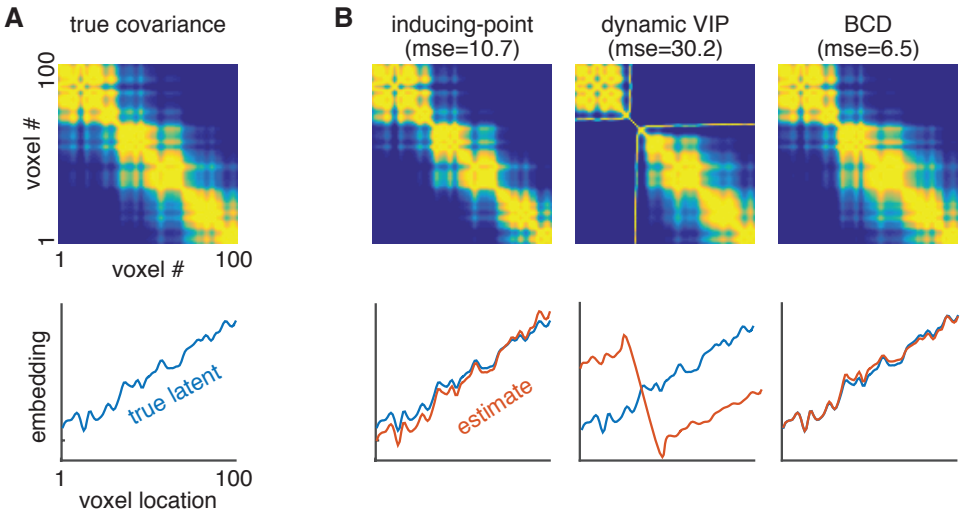

Fig 2. Recovery of 1D brain kernel from synthetic data. **A**. The true covariance matrix for neural activity at 100 evenly-spaced voxels in a 1D brain (top), generated by a 1D latent embedding function sampled from the brain kernel model (bottom, blue curve). **B**. Model estimates. The first column is the inducing-point method; the second column is dynamic VIP; and the last column is our BCD method. In each column, we show the estimated covariance matrix (top) and the estimated 1D latent embeddings (bottom, red curve). We also show the mean squared error (MSE) between the estimated covariance matrix and the true covariance matrix.

We created a simulated dataset using a 1-dimensional brain and 1-dimensional latent embedding function, sampled from the brain kernel model (Fig. 2). Here, the latent embedding is simply a nonlinearly warped version of the 1D brain. We considered a set of 100 voxel locations on an evenly spaced 1D grid: $\mathbf{X} = [1, 2, \ldots, 100]^\top$. We then sampled the voxels' latent locations $\mathbf{Z}$ from a GP with mean $m(\mathbf{x}) = 0.6\mathbf{x}$ and RBF covariance with length scale $\delta = 10$ and marginal variance $r = 9$: $\mathbf{Z} \sim \mathcal{N}(0.6\mathbf{X}, \mathbf{K})$, where $\mathbf{K}_{ij} = 9\exp(-(i-j)^2/200)$. Given this embedding function, the brain kernel defines a covariance matrix for neural activity at these 100 voxels, denoted $\mathbf{C}$, with the i, jth entry given by $\mathbf{C}_{ij} = \exp(-(\mathbf{z}_i - \mathbf{z}_j)^2/2)$ (Fig. 2A top). To obtain simulated fMRI measurements, we sampled 750 observations from a Gaussian distribution with zero mean and covariance $\mathbf{C} + 5I$, where $\sigma^2 = 5$ represents the variance of additive measurement noise.

We compared the different estimators on the task of recovering the true covariance and latent embedding function from this dataset (Fig. 2B). The inducing-point method with six inducing points (column 1) performed well at recovering the latent embedding function, although it was outperformed by our BCD estimator in terms of mean squared error (BCD; column 3). The dynamic VIP estimate with six inducing points (column 2) converged to a local optimum far from the true latent embedding, yielding a substantially higher error. In contrast, exact inference using BCD outperformed both inducing point-based approximate methods in terms of accuracy at recovering the true brain kernel from simulated data.

Next, we conducted a set of synthetic experiments to examine how well different models captured the covariance of simulated fMRI data, using voxels on a 1-dimensional, 2-dimensional, or 3-dimensional grid. We fit these simulated datasets using the brain kernel model optimized with (1) BCD, (2) variational inducing-point, and (3) the dynamic VIP method, and two additional models: (4) a linear brain kernel (LBK) model, and (5) a GP with RBF covariance function (RBF). Note that the first three methods are based on the original brain kernel model but have different inference methods, while the last two methods

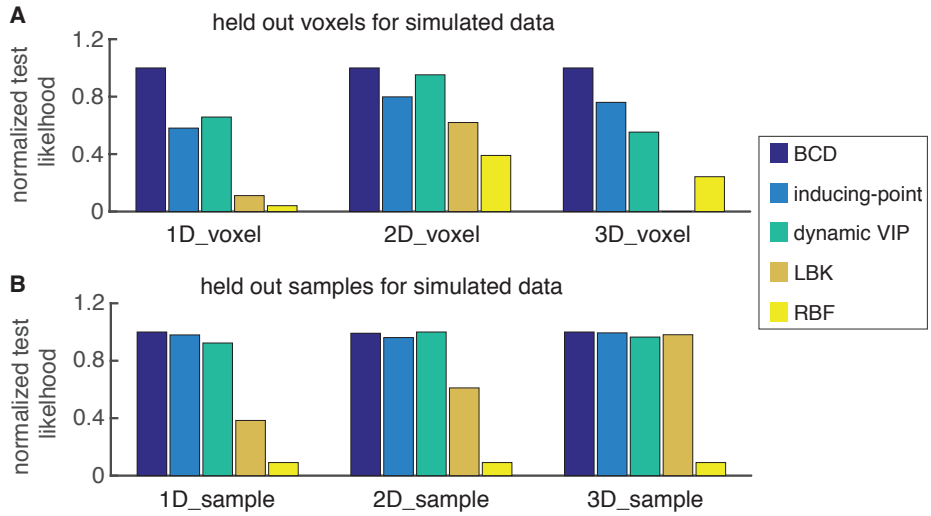

Fig 3. Quantitative comparisons for BCD, inducing-point, dynamic VIP, LBK, and RBF on three simulated datasets with 1D, 2D, and 3D input voxel locations. **A** is the held-out voxel experiment, and **B** is the held-out sample experiment. Within each experiment, we show the normalized test likelihood values for each method with three different simulated datasets.

represent simplifications of the proposed brain kernel model. The linear brain kernel model assumes a purely linear embedding function; it thus allows rotating and linearly dilating the voxel grid, but does not allow for nonlinear warping of voxel locations. The covariance function for neural activity $\mathbf{v}$ under the LBK model is given by:

$$\kappa_{LBK}(\mathbf{x}_i, \mathbf{x}_j) = \rho \exp\left(-\tfrac{1}{2}||\mathbf{B}\mathbf{x}_i - \mathbf{B}\mathbf{x}_j||_2^2\right), \tag{21}$$

where $\mathbf{B}$ is a $d \times v$ linear embedding matrix and v is the number of dimensions of x, e.g., $v = 2$ for a 2-dimensional grid of voxels, and d is the dimension of the latent space. To assess the importance of the structured latent embedding in covariance estimation, we fit the data with a standard GP with RBF covariance function:

$$\kappa_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \rho \exp\left(-\tfrac{1}{2}||\mathbf{x}_i - \mathbf{x}_j||_2^2 / l^2\right), \tag{22}$$

where l is the length scale for the RBF kernel. This model imposes smoothness using Euclidean distance, without estimating any (linear or nonlinear) transformation of the true voxel locations. To fit the LBK and RBF models, we maximized the following negative log likelihood for hyperparameters θ:

$$\begin{aligned} \mathcal{L}(\theta) &= -\log p(\mathbf{Y}|\mathbf{X}, \theta) \\ &= \text{tr}\left[(\mathbf{C}(\mathbf{X}, \theta) + \sigma^2 \mathbf{I}_n)^{-1}\mathbf{S}\right] + T\log|\mathbf{C}(\mathbf{X}, \theta) + \sigma^2 \mathbf{I}_n|, \end{aligned} \tag{23}$$

where $\mathbf{C}(\mathbf{X}, \theta)$ is the model-generated covariance matrix (given by eq. 21 for the linear brain kernel with $\theta = \{\mathbf{B}, \rho\}$, and by eq. 22 for the RBF model with $\theta = \{l, \rho\}$), $\mathbf{S}$ is the sample covariance (eq. 14), σ^2 is the noise variance, and T is the number of samples in the simulated dataset.

For each grid dimension, we simulated ten independent datasets, each with different nonlinear embedding functions sampled from the brain kernel model. For the 1D experiments, we generated each dataset with 500 voxels and 750 samples, and embedded the 1D voxel space to a 1D latent embedding space. For the 2D experiments, we used a 25×25 voxel grid, embedded nonlinearly in a 3D latent embedding space, and generated datasets of 1000 samples, where each sample is a vector of 625 noisy measurements of brain activity at voxel locations. For the 3D datasets, we used a $10 \times 10 \times 10$ grid of voxels, embedded nonlinearly into a six-dimensional latent space, and we generated 1500 samples per dataset.

To compare models, we performed two different cross-validation tests: (i) prediction on held-out voxels (Fig. 3A), and (ii) predictions on held-out samples (Fig. 3B).

For the first of these, we removed ten randomly-selected voxels $\mathbf{X}^*$ from the simulated 1D datasets and set them aside as test data, and we optimized the model parameters on a training set of measurements from the remaining 490 voxels. (For experiments with 2D and 3D grids, we set aside 62 and 100 voxels as held-out data, and we trained models using the remaining 563 and 900 voxels, respectively.) For the BK and LBK models, we computed the embedding locations $\mathbf{Z}^*$ for the test voxels using the GP posterior mean given the inferred embedding locations $\mathbf{Z}$ (eq. 19), and we used these locations to evaluate the covariance of the test voxel activity (eq. 12). (For the RBF model, there is no embedding, so the covariance of the test data depends only on the test voxel locations $\mathbf{X}^*$.) We used the resulting predictive covariances to compute the log likelihood of the test data. We found that the BCD-optimized brain kernel estimate outperformed other methods, while the LBK and RBF models performed worse, presumably due to their inability to capture the nonlinear embedding of the simulated data (Fig. 3A).

Next, we evaluated cross-validation performance on held-out samples, which were measured at the same set of voxels as the training set. For these simulations, we randomly selected 75 samples as test data for the 1D datasets, and used the remaining 675 samples to fit the five models. (For 2D and 3D grids, we used a train-test split of 900:100 and 1350:150 samples, respectively.) We estimated a covariance function using measurements in the training set, and computed a test likelihood using this covariance function on the test set. We generated ten random splits for each dataset and computed the normalized test log likelihood. In this predictive task, the BCD estimate again outperformed other methods (Fig. 3B).

Overall, these simulations demonstrated that the inducing point-based GP methods, which are often used due to their scalability, had higher error than our exact BCD inference method. In addition, by comparing to the linear brain kernel and the standard GP model with an RBF kernel, we showed that the ability to capture a nonlinear transformation of the voxel locations is critical for accurately modeling the covariance of simulated brain activity.

5 Inferring the brain kernel from resting-state fMRI data

Now that we have described the brain kernel model and validated our inference method using simulated data, we turn to the problem of inferring the brain kernel from real data. We fit the brain kernel model to large-scale publicly-available resting-state fMRI data from the Human Connectome Project (HCP) [6]. An advantage of this approach is the large size of the HCP sample, which mitigates overfitting and allows us to learn an embedding function that captures correlation patterns common to a vast collection of different brains. However, applying the resulting brain kernel to task-based fMRI datasets assumes that the correlations presented in resting-state fMRI data are applicable to activation patterns in other brain states. Previous studies have shown interesting relations between brain activity during a task and at rest. One study showed that coherent spontaneous activity accounted for variability in event-related BOLD responses [23]. A second study concluded that functional networks used by the brain in action were continuously and dynamically “active” even when at “rest” [24]. A third study showed that resting-state activation patterns had strong statistical similarities to cognitive task activation patterns [25]. These provide reasons for optimism, though the possibility of changes in correlation across different tasks could potentially affect the application of the brain kernel which we will show empirically later.

We examined resting-state fMRI data from 812 subjects collected via the Human Connectome Project (HCP) [6]. These data were acquired in four fMRI runs of approximately 15 minutes each, two runs in one session and two in another session, with eyes open and relaxed fixation on a projected bright cross-hair on a dark background. A sophisticated preprocessing pipeline was used to align voxels across subjects. Detailed information about imaging protocols, image acquisition, and preprocessing can be found in [26].

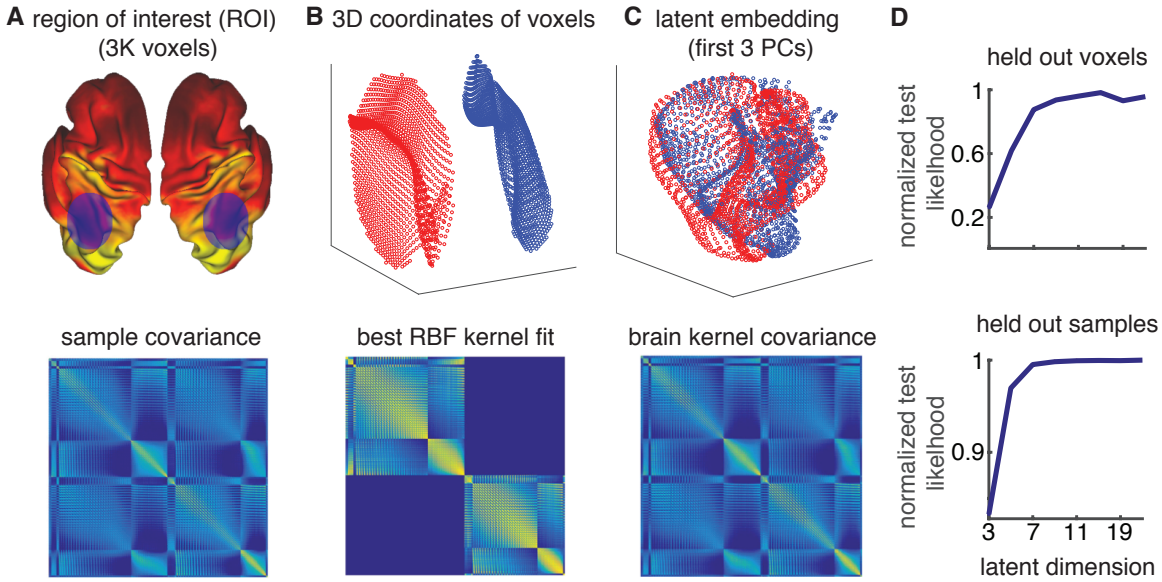

Fig 4. 3D embeddings and the corresponding covariance matrices of resting-state fMRI data. **A**. The two example regions of interest (ROIs) we selected in the left and right parietal lobes (top) and the full covariance for the 3K voxels in these two ROIs (bottom). **B**. The original 3D voxel coordinates (top) and the best-fitting RBF kernel (bottom). **C**. A 3-dimensional projection of the estimated 20-dimensional brain kernel embedding (top) and the corresponding brain kernel covariance matrix (bottom). **D** shows the influence of the latent dimension on the predictive performance for held-out voxels and held-out samples with the resting-state fMRI data.

The resulting dataset consisted of 59,412 voxels in a cortical surface coordinate system. Although the full covariance matrix of these data is of size $\approx 59K \times 59K$, we fit the model using the first 4500 eigenvectors of the full matrix provided by HCP.

We fit the brain kernel with different numbers of latent dimensions and computed the test log likelihood with held-out voxels and samples (Fig. 4D). We found that test performance plateaued with increasing dimensionality, and selected $d = 20$ dimensions for subsequent analyses. We then estimated the brain kernel by fitting the 20-dimensional latent embedding for each voxel using BCD optimization of the log marginal likelihood. The resulting function is a matrix of embedding locations of size $\approx 59K \times 20$, where each row contains the embedded location of a single voxel in the resting-state fMRI dataset. This embedding, in addition to the hyperparameters (the linear projection matrix $\mathbf{B}$ and the hyperparameters $\{\gamma, \delta\}$ for $\mathbf{K}$), provide a full parametrization of the brain kernel.

Although it is impossible to visualize the 20-dimensional nonlinear embedding that defines the brain kernel, we can gain insight into its shape by plotting low-dimensional projections on subsets of voxels. To visualize the brain kernel, we selected two symmetric ROIs in the left and right parietal lobes (Fig. 4A, top). Taken together, these two ROIs contain 3K voxels. We visualize the sample covariance of these voxels, computed directly from the eigenvectors (Figure 4A, bottom). This covariance contains four identifiable blocks: two diagonal blocks that correspond to the covariance of voxels within each ROI, and two off-diagonal blocks that correspond to cross-ROI covariances. The off-diagonal blocks reveal that the two ROIs have reasonably strong correlations despite being spatially distant in the brain.

The original 3D voxel coordinates (Fig. 4B, top) and the covariance given by the best-fitting RBF kernel (eq. 22; Fig. 4B, bottom) show that the RBF kernel model fails to capture the the off-diagonal blocks of the

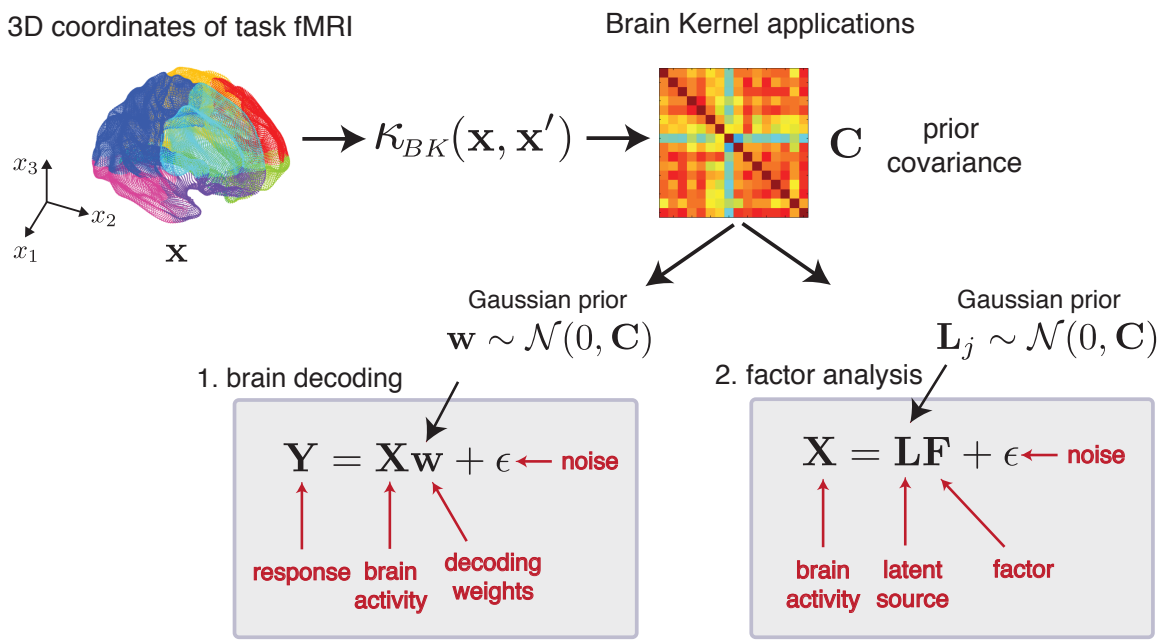

Fig 5. Schematic figure illustrating two applications of the brain kernel to task fMRI data. After fitting the brain kernel to resting-state data, we applied it to task fMRI data with n voxels by evaluating the brain kernel at the 3D voxel locations. This results in a $n \times n$ prior covariance matrix for the task data, denoted $\mathbf{C}$. We then used the covariance $\mathbf{C}$ as the prior covariance for two modeling tasks: (1) brain decoding and (2) factor analysis. Two unknown parameters $\mathbf{w}$ and $\mathbf{L}$ are both random variables with a Gaussian prior whose covariance is $\mathbf{C}$. Thus, we effectively imposed assumptions on the structure of $\mathbf{w}$ and $\mathbf{L}$ via $\mathbf{C}$.

sample covariance, corresponding to covariances between voxels in opposite hemispheres, due to the fact that the RBF kernel depends only on the Euclidean distance between voxels. In contrast, a 3-dimensional projection of the estimated 20-dimensional brain kernel embedding (Fig. 4C, top) and the corresponding brain kernel covariance (Fig. 4C, bottom) appears to capture this cross-ROI structure in the covariance matrix. The 3D projection corresponds to the first three principal components of the 20-dimensional latent embeddings, and shows that paired voxels from opposite hemispheres are embedded close to each other under the brain kernel. Conversely, some voxels that are physically close together in the brain are embedded far apart in the embedding space. This nonlinear embedding allows the brain kernel to more accurately capture the covariance of the real data.

6 Applications

We applied the brain kernel to two different fMRI data analyses (Figure 5). Note that we estimated the brain kernel using the resting-state fMRI from the HCP database as described above. We kept the same brain kernel across all analyses. We didn't re-estimate the brain kernel using any task fMRI. For both applications, we took the 3D brain coordinates for n voxels from a task fMRI dataset of interest. We input these coordinates and corresponding observations into the brain kernel, which produced a $n \times n$ covariance matrix $\mathbf{C}$ (using eq. 19 and eq. 20). We then used this covariance as the prior covariance for Bayesian analyses for different modeling tasks, brain decoding and factor analysis. We describe these applications in detail below.

6.1 Brain decoding

In this section, we illustrate how the brain kernel can be applied to fMRI classification (or "decoding") tasks.

6.1.1 HCP tasks

We first examined the task fMRI datasets in the HCP database. We explored the working memory task, the gambling task [27], the language processing task [28], the motor task [29], the emotion processing task [30], the relational processing task [31], and the social cognition task (more details found in [26, 32]). We will elaborate on the working memory and gambling tasks and finally summarize the result with all datasets.

Working memory task

We obtained the working memory task fMRI measurements from the HCP [32]. The stimuli consisted of four types of pictures: places, tools, faces, and body parts. Stimuli were presented for 2 s on each trial followed by a 500-ms inter-trial interval (ITI). Each task block consisted of ten trials of 2.5 s each. Each run contained 8 task blocks, half for a 2-back working memory task and half for a 0-back working memory task, along with 4 fixation blocks. Two runs were collected for each subject, with 405 volumes per run or approximately 5 minutes. Because the task fMRI was from the same HCP project as the resting-state fMRI, they shared the same coordinate system and preprocessing pipeline. The task fMRI was aligned with the MNI template with the same 59,412 voxels as the resting-state fMRI data. Instead of analyzing the whole-brain data for brain decoding, we worked with functional ROIs. For each presented type of pictures, we took the group-average task contrast for the 2-back vs 0-back working memory task and kept all voxel coordinates whose z-statistics were -1.96 or +1.96 standard deviations away from the mean, indicating that the voxels were statistically significant in the contrast map. We repeated the same thresholding procedure for all objects and took a union set of all coordinates to formulate the functional ROIs for the working memory task. We didn't select the ROIs using the contrast among objects, therefore the resulting ROIs contained no discriminative information with respect to the decoding task.

The experiment required observers to perform a working memory task using four different types of objects. We tested the ability to decode these objects from fMRI data by fitting a binary linear classifier for each pair of objects. We used Bayesian linear regression classifiers to solve these six binary classification problems. A Bayesian linear regression classifier has the form,

$$y = \text{sign}(\mathbf{x}\mathbf{w} + \epsilon), \tag{24}$$

where $\mathbf{x}$ is a vector of all voxels for one fMRI measurement or sample, y is a ± 1 label indicating the binary object category for that sample, $\mathbf{w}$ is a vector of regression coefficients, and ϵ is independent zero-mean Gaussian noise with variance σ^2. To regularize the estimate of $\mathbf{w}$, we assumed a zero-mean Gaussian prior, $\mathbf{w} \sim \mathcal{N}(0, \mathbf{C})$, with prior covariance $\mathbf{C}$. We considered three different choices of prior covariance: (1) a ridge prior, which corresponds to a diagonal covariance with a positive constant along the diagonal; (2) a radial basis function covariance (eq. 22), which imposes smoothing based on the voxels' 3D locations in the brain, and (3) a covariance derived from the brain kernel. For the brain kernel prior, we used the covariance function

$$\kappa_{BK}(\mathbf{x}_i, \mathbf{x}_j) = \rho \exp\left(-\tfrac{1}{2}||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2^2 / l^2\right), \tag{25}$$

where f is the nonlinear embedding function from the brain kernel, and $\{\rho, l\}$ are tuneable hyperparameters.

We trained the classifier with these three priors on one run and calculated accuracy performance on the second run, then repeated the same procedure with test and training sets reversed. When we trained

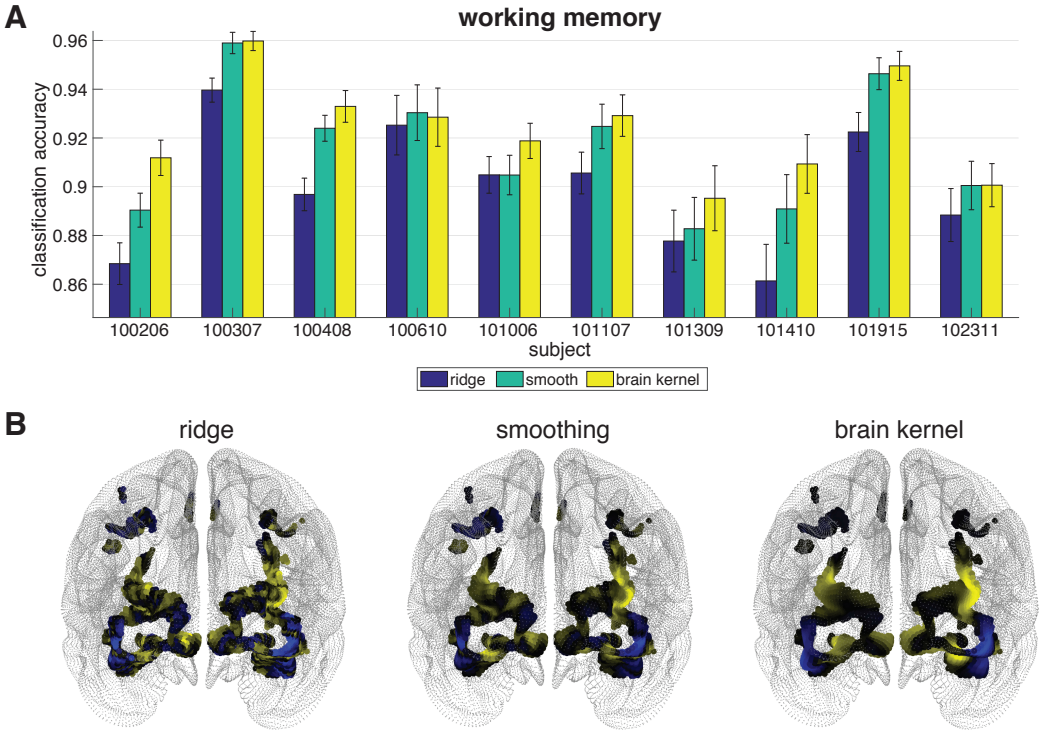

Fig 6. A. Accuracy performance on the working memory task. The x-axis indicates subject identifiers. The y-axis is accuracy performance. We compared our brain kernel with a ridge prior and a smooth RBF kernel, color coded. The error bars indicate standard errors. **B**. Visualization of an example set of decoding weights. Blue indicates negative values and yellow indicates positive values.

the model, we randomly split the training run into five folds and selected the optimal hyperparameters in the covariance functions via 5-fold cross validation. After cross validation, we applied the optimal hyperparameters to the test run for each prior model. We repeated this 5-fold cross validation experiment ten times to reduce variability. We used linear regression to train the classifier, so the ± 1 labels were treated as continuous target values, and test accuracy was evaluated by taking the sign of the prediction. We plotted averaged accuracy across both runs and all repeats for the three priors for ten randomly-selected subjects (Fig. 6A). The RBF prior outperformed the ridge prior, but the brain kernel prior outperformed both of the other priors, indicating that smoothing in a nonlinear embedding space defined by correlations of fMRI signals provided additional benefits in regularizing weights for a classification task. Moreover, the framework of the brain kernel for regularization allows us to visualize the inferred decoding weights overlaid on a 3D brain (Figure 6B).

Gambling task

We next examined fMRI data in a gambling task from the HCP [32], adapted from a prior study [27]. Participants were asked to play a card-guessing game. They were shown a mystery card with a number that could range from 1 to 9. They needed to guess whether it was more or less than 5 by pressing on of two buttons. If they made the correct guess, the card showed a green up arrow with “$1” for rewards; if they guessed wrongly, the card showed a red down arrow with “-$0.5” for losses; if the true value was 5, they got a neutral response without win or loss. Participants had 1500 ms to guess, and the feedback was presented for 1000 ms, followed by a 1000-ms inter-trial interval (ITI). Each task block consisted of eight trials that were either mostly reward or mostly loss. Two runs were collected for each subject.

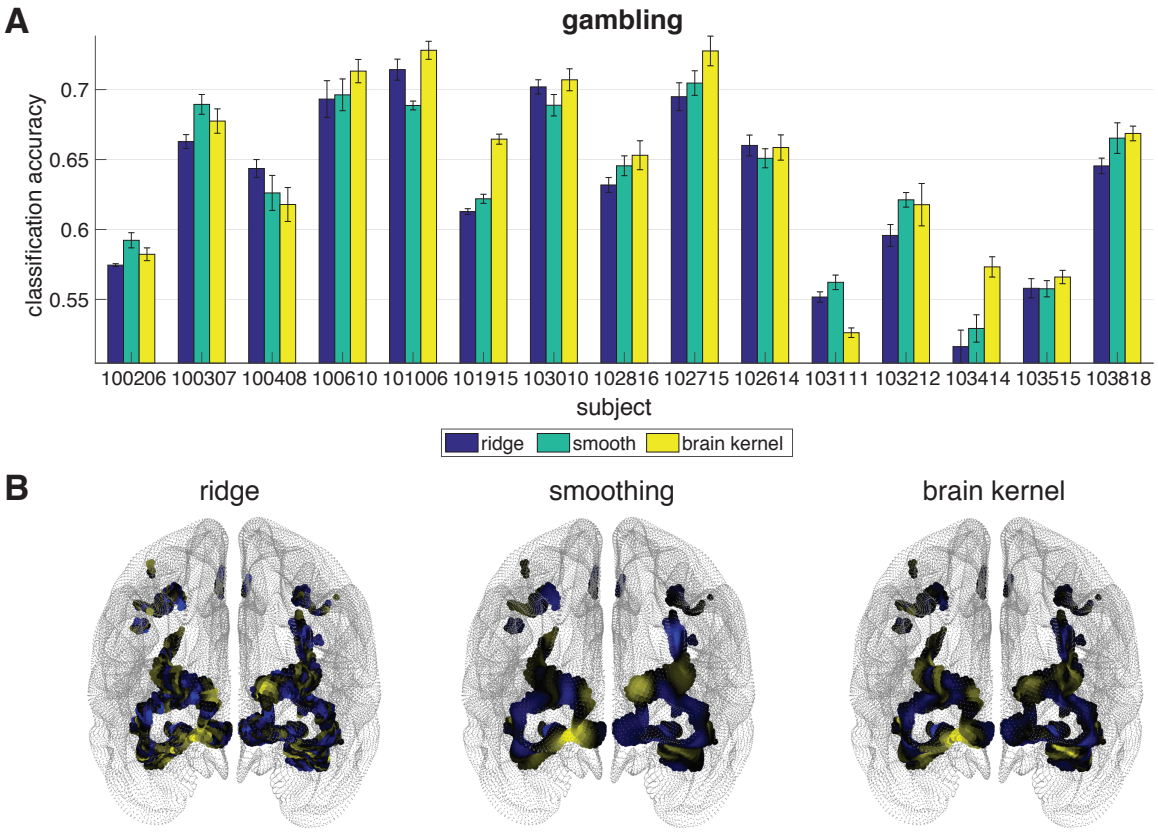

Fig 7. A. Accuracy performance on the gambling task. The x-axis indicates subject identifiers. The y-axis is accuracy performance. We compared our brain kernel with a ridge prior and a smooth RBF kernel, color coded. **B**. Visualization of an example set of decoding weights. Blue indicates negative values and yellow indicates positive values.

Each run contained two mostly reward and two mostly loss blocks, interleaved with four fixation blocks. There were 253 volumes per run, which lasted approximately 3 minutes. The task fMRI was aligned to the MNI template and had the same 59,412 voxels as the resting-state fMRI data. Instead of analyzing the whole-brain data for brain decoding, we worked with functional ROIs which were selected using the same approach as described in the working memory task.

We formulated the task as a binary classification problem separating reward trials and punishment trials. We used the same Bayesian linear regression classifiers as described in the working memory section. We trained the classifier with three priors on one run and calculated the accuracy performance on the second run, then switched the training and test runs. This procedure was repeated ten times. The ± 1 labels were treated as continuous target values during training, and the test accuracy was evaluated by taking the sign of the prediction. We computed the averaged accuracies across two runs and 10 repetitions for the three priors for 15 subjects (Fig. 7A). For 11 out of 15 subjects, the brain kernel improved accuracy over both the ridge prior and the smooth RBF prior. For two of the remaining four subjects, the brain kernel outperformed the ridge prior. The discrimination problem with the gambling data was more difficult than the working memory task. Many activations occur in the visual cortex and the prefrontal cortex for working memory, whereas critical activations for the reward task may be localized to the striatum, which is not included in the cortical data used here. However, with the cortical brain kernel, we were still able to improve the predictive ability of the Bayesian model.

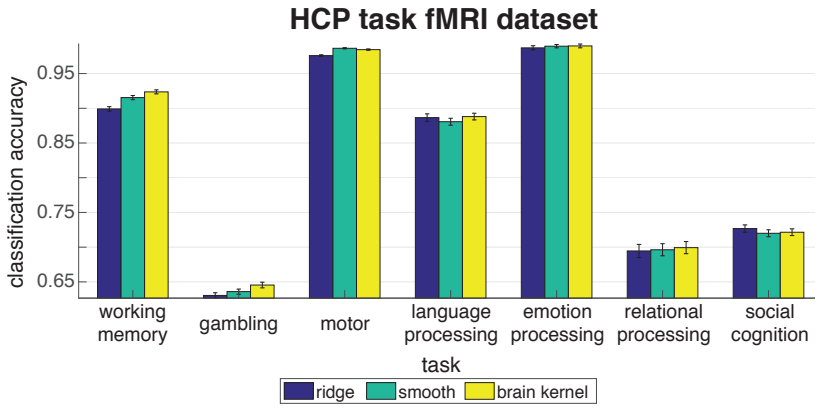

Fig 8. Averaged accuracy performance for all task fMRI datasets in the HCP database. The x-axis indicates the task. The y-axis is accuracy performance.

All HCP tasks

We've elaborated on the working memory and gambling tasks above. We also achieved the classification accuracy performance for all other tasks (presented in Appendix B). Here we summarize the averaged accuracy over all subjects for each task in Fig. 8. Consistent with the above results, we succeeded in achieving the best performance with the working memory and gambling tasks using the brain kernel. For other tasks, the brain kernel performed mildly better than the ridge prior and the smoothing prior estimates. Regularizing the decoding weights with the brain kernel did not improve performance in a dominating way, suggesting that the covariance of resting-state fMRI did not provide significantly useful information for classifying fMRI measurements in these tasks, or at least that the brain kernel model was not capable of fully capturing it. It remains possible, however, that a brain kernel trained on task fMRI datasets might offer benefits for decoding fMRI data from these tasks.

6.1.2 Visual recognition task

Next, we examined the problem of decoding faces and objects from fMRI measurements during a visual recognition task. Just to remind, the brain kernel was estimated using the resting-state fMRI from HCP, and we applied it to a popular fMRI dataset from a study of human ventral temporal cortex [33] for the decoding task. We extended the application beyond HCP, i.e., constructing the brain kernel on HCP and then trying on a completely different dataset. Therefore, we were looking at across-dataset generalization, not just across-task generalization within HCP. In this visual recognition experiment, six subjects were asked to recognize eight different types of objects (bottles, houses, cats, scissors, chairs, faces, shoes, and scrambled control images, examples in Fig. 9). Each subject participated 12 scanning runs. In each run, the subjects viewed images of eight object categories, with 11 whole-brain measurements per category. Each subject's fMRI data was preprocessed using the fMRIprep package[2] [34] and aligned to the MNI template. Both voxels in this dataset and voxels in the HCP database were all aligned in the same MNI space, allowing us to use eq. 19 to get the brain kernel covariance for the present dataset. Instead of analyzing the whole-brain data, we extracted ROIs with 1645 voxels in the ventral temporal cortex, which is thought to be involved in object recognition. The ROI mask was obtained from Nilearn [35].

We assessed performance by training Bayesian linear regression classifiers to discriminate between pairs

[2] https://github.com/poldracklab/fmriprep

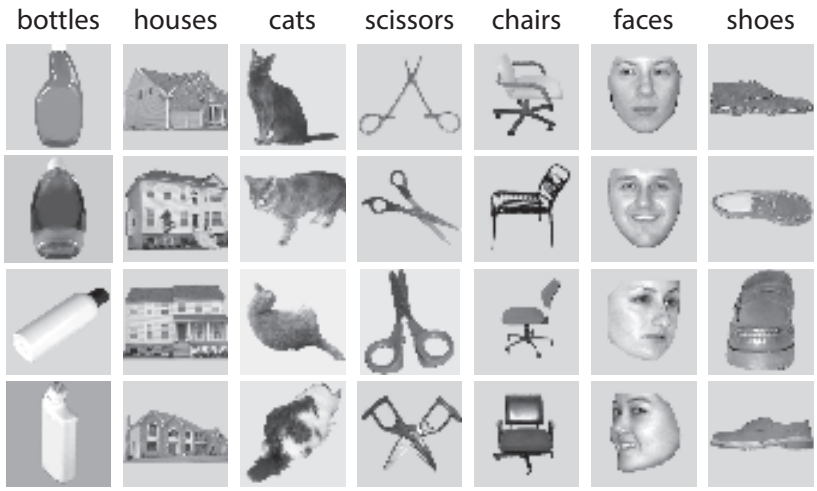

Fig 9. Examples of the stimuli for 7 categories (except for scrambled control images) [33].

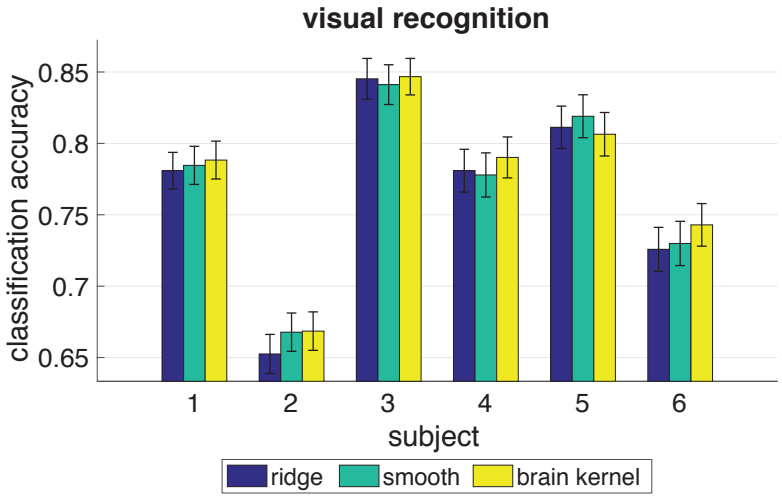

Fig 10. Accuracy performance on the visual recognition task. The x-axis indicates subject IDs. The y-axis is accuracy performance. We compared our brain kernel with a ridge prior and a smooth RBF kernel, color coded.

of objects, e.g., face vs. bottle, for each of the 28 possible binary classifications among the eight objects (Fig. 9). We trained the weights $\mathbf{w}$ for each model using linear regression from fMRI measurements $\mathbf{x}$ to binary labels $y \in \{-1, +1\}$, and assessed accuracy on the test set using predicted labels $\hat{y} = \text{sign}(\mathbf{x}\mathbf{w})$. Here, we split the training and test sets by subjects. In this visual recognition dataset, we had six subjects but only $11 \times 2 \times 12 = 264$ measurements in a 1645-voxel space for each subject in a binary decoding task. The number of training measurements was not sufficiently large to train a good classifier and lead to a reasonable generalization performance. Therefore, different from the HCP tasks, we chose to do inter-subject analyses by using 5 subjects for training and one subject for test. We repeated this leave-one-subject-out manner for six times with each subject being used as the test set once and obtained the result in Fig. 10.

The averaged accuracy performance across four repeated runs for six subjects shows that the brain kernel performed comparably to the ridge and smooth priors with better accuracy performance for 5 out of 6 subjects (Fig. 10). This indicates that the brain kernel can provide functional and structural support for most subjects and tasks. The improvement was statistically modest overall based on the standard errors, which could be a result of several factors: misalignment of the coordinate space to the HCP coordinate space used to estimate the brain kernel; mismatch between the resting-state covariance used to construct the brain kernel and covariance present during the visual recognition task; or the object recognition tasks may rely on fine-grained spatial response topographies that are poorly aligned across individuals.

6.2 Factor modeling

In this section, we illustrate a second type of application of the brain kernel. Instead of using it to regularize decoding weights, as in the previous section, we use it as a spatial prior for Bayesian factor analysis.

6.2.1 Sherlock movie watching task

We examined the Sherlock fMRI dataset, in which participants were scanned while they watched the British television program “Sherlock” for 50 min [36]. The fMRI data comprised 1,973 TRs (Repetition Time), where each TR was 1.5 s of the movie. Before performing any analysis, the fMRI data were preprocessed and aligned to MNI space using the techniques described in prior work [36]. We examined the brain data averaged across all subjects to smooth out individual variability. We identified 11 ROIs previously implicated in processing naturalistic stimuli, comprising the default mode network (DMN-A, DMN-B), the ventral and dorsal language areas, and the primary auditory and visual cortices [37].

For each ROI, we performed a standard factor analysis (FA) to factorize the voxel-by-time fMRI data into a latent source matrix and a factor matrix. Thus fMRI images can be considered as being generated by a covariate-dependent superposition of latent sources. Some following analyses, such as decoding and encoding tasks, can be performed with the factor matrix. The number of latent sources is much fewer than the number of time points, thus providing a parsimonious description of neural activity patterns that avoids many of the pitfalls of traditional voxel-based approaches. Similar FA based models have been proposed in [38, 39]. Here, we performed a Bayesian factor analysis of the form

$$\mathbf{X} = \mathbf{L}\mathbf{F} + \epsilon, \tag{26}$$

where $\mathbf{X} \in \mathbb{R}^{N \times T}$ is the fMRI image with N voxels and T time points, $\mathbf{L} \in \mathbb{R}^{N \times K}$ is the latent source matrix encoding the canonical spatial pattern (over voxels) associated with each latent source. $\mathbf{F} \in \mathbb{R}^{K \times T}$ is the factor matrix containing the timeseries for K latent sources. ϵ is a Gaussian noise with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We assumed that the ith factor (column) is from a standard normal distribution, i.e., $p(\mathbf{F}_i) = \mathcal{N}(0, \mathbf{I})$. We also assumed that the jth latent source (column) is from a Gaussian prior $p(\mathbf{L}_j) = \mathcal{N}(0, \mathbf{C})$ where $\mathbf{C}$ is the covariance matrix. Like the decoding tasks, we used a ridge covariance, a smooth RBF kernel, and the brain kernel in the place of the covariance matrix $\mathbf{C}$. With different prior covariances, the latent sources $\mathbf{L}_{:,j}$ showed different characteristics imposed by these distinct regularizations.

Our goal was to infer the latent variables $\mathbf{L}$ and $\mathbf{F}$, and the hyperparameters for $\mathbf{C}$ and σ^2, denoted as θ. To quantify performance, we compared data explanation under the three prior covariances with the same number of latent sources which was set to 10. In practice, we first standardized the fMRI image $\mathbf{X}$, then solved the optimization of the hyperparameters using the marginal distribution, and finally inferred the factor matrix and the latent sources via maximum likelihood parameter estimation (details in Methods). To evaluate the performance of the three priors, we used a method called “co-smoothing" [40], depicted in Fig. 11. We first split the time points into two equal sets by taking the first half as one set and the latter half as the other set.

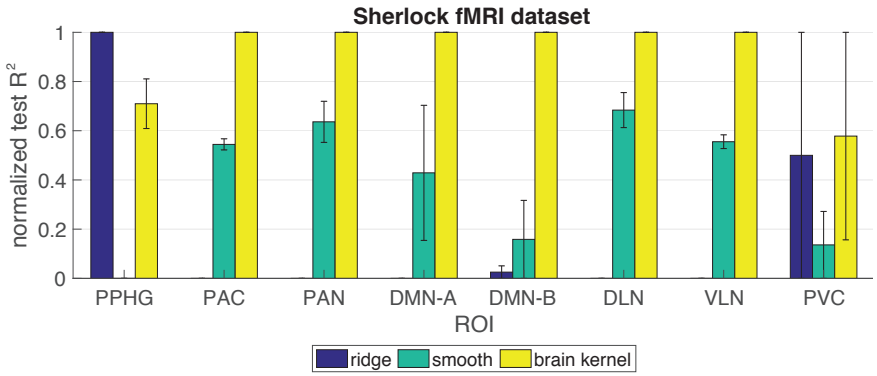

Fig 12. Normalized test R^2 performance on the Sherlock fMRI dataset. The x-axis indicates ROIs (PPHG – Posterior Parahippocampal Gyrus; PAC – Primary Auditory Cortex; PAN – Primary Auditory Network; DMN-A – Default Mode Network-A; DMN-B – Default Mode Network-B; DLN – Dorsal Language Network; VLN – Ventral Language Network; PVC – Primary Visual Cortex). The y-axis is the normalized R^2 performance on the test set (higher values indicate better performance). The error bars indicate standard errors. We compared our brain kernel with a ridge prior and a smooth RBF kernel, color coded.

We took the first set for training (the blue region) and the second set for inference (the pink region) and test (the yellow region). We trained the model with the neural activity in the training set to obtain the estimated latent sources $\mathbf{L}^*$. We then kept the latent sources fixed for the inference and test purpose. Next we split the second set into five folds along the voxel axis, four for inference and one for test. We used the neural activity in the inference set to infer the factor matrix (mapping the latent sources to the time series) during the inference period given the latent sources. Finally we predicted the neural activity for the left-out voxels in the test set given the latent sources and factor matrix. We repeated the inference and test step five times in a cross-validation fashion and obtained an averaged R^2 value representing the performance of the prior. After the first run, we achieved three R^2 values for the three priors (ridge, smooth and brain kernel). To better visualize the difference, we normalized the three R^2 values so that the maximum was 1 and the minimum was 0. We then launched a second run by using the second set as the training set and the first set as the inference and test set. The final normalized test R^2 value was an average of the two runs.

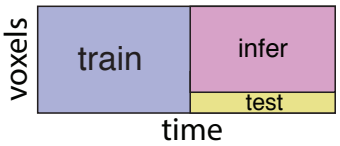

Fig 11. Co-smoothing evaluation. The blue region is the data used for training; the pink region contains voxels that are used to infer the factor matrix during the test period; and the yellow region is used for test.

We compared the normalized test R^2 value for the three priors for each ROI (Fig. 12). The error bars indicate standard errors. In most regions, the brain kernel outperformed the ridge covariance and the smooth RBF kernel. This implies that when performing Bayesian FA, the brain kernel provides a superior prior covariance for the latent source matrix and may enhance performance in terms of data explanation.

6.2.2 HCP tasks

We examined the task fMRI datasets in the HCP database with the same Bayesian FA model. We collected all the task fMRIs for the same 10 subjects and performed Bayesian FA for each subject in each task. We implemented the same “co-smoothing" evaluation and compared the normalized test R^2 for the three priors averaged over all subjects for each task (Fig. 13). In most tasks, the brain kernel outperformed the

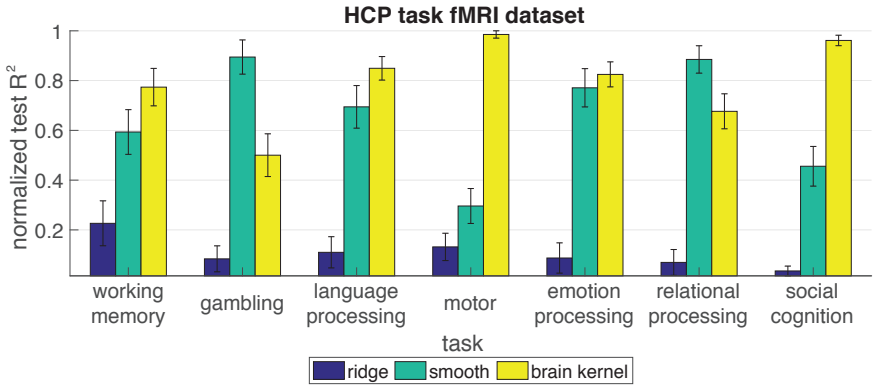

Fig 13. Normalized test R^2 performance on the HCP task fMRI datasets. The x-axis indicates the task. The y-axis is the normalized test R^2 performance averaged over 10 subjects for each task (higher values indicate better performance). The error bars indicate standard errors. We compared our brain kernel with a ridge prior and a smooth RBF kernel, color coded.

ridge covariance and the smooth RBF kernel. This implies that the brain kernel provides a superior prior covariance for the latent source matrix and may enhance performance in terms of data explanation for the HCP database.

7 Discussion

We introduced the brain kernel, a new model for the spatial covariance of fMRI data. This model takes the form of a covariance function, meaning that it can take any two continuous voxel locations in the brain as inputs and return the prior covariance of their activities. Unlike standard smoothness-inducing covariance functions used in the Gaussian process literature, which assume that correlation falls off monotonically with distance, the brain kernel allows widely-separated voxel locations (e.g., on opposite sides of the brain) to have high correlation, while pairs of nearby voxels can be nearly uncorrelated. This property arises from the fact that the nonlinear embedding function, parametrized with Gaussian processes, warps and stretches the brain in a higher-dimensional space so that widely-separated pairs of voxels can be mapped to nearby embedding locations, while pairs of nearby voxels can be moved far apart in embedding space.

The brain kernel model is a hierarchical extension of the Gaussian process latent variable model. We fit the brain kernel model using a nonlinear embedding of the 3D brain in a 20D space. We introduced an exact inference method for fitting the brain kernel using block coordinate descent (BCD), and trained it using a large publicly-available dataset of resting-state fMRI measurements from many subjects. We found that the resulting brain kernel accurately captured the covariance of resting-state fMRI measurements.

We further demonstrated the applicability of the brain kernel using two different modeling tasks: brain decoding and factor analysis. In both cases, we used the brain kernel to define a prior distribution in place of a more conventional prior based on ℓ_2 shrinkage or local smoothness. In both cases we showed that the brain kernel can achieve gains in performance, illustrating that the correlations in resting-state fMRI may be usefully mined to improve the accuracy of task fMRI.

Limitations

Despite the successes presented above, it is important to acknowledge that we applied the brain kernel to several other datasets for both the decoding and the factor modeling tasks beyond the examples shown in the main section, for which it did not yield improvements over standard priors.

For decoding, we applied the brain kernel to the Sherlock movie dataset [36] whose decoded vectors contained semantic descriptions for the movie scenes. We observed that the smoothing prior and the brain kernel prior both converged to the ridge prior after hyperparameter estimation, i.e., the estimated length scale was very small and thus smoothness did not help the decoding task. We also applied it to a public fMRI dataset in which subjects viewed 5000 visual images (BOLD5000) [41]. The binary labels were living objects versus nonliving objects. We observed that the brain kernel did not reliably outperform the ridge prior and the smoothing prior estimates across subjects. More details can be found in Appendix B. For these cases, regularizing the decoding weights with the brain kernel did not improve performance, suggesting that the covariance of resting-state fMRI did not provide useful information for classifying fMRI measurements in these tasks, or at least that the brain kernel model was not capable of capturing it. It remains possible, however, that a brain kernel trained on task fMRI datasets might offer benefits for decoding fMRI data from these tasks.

For factor modeling, we also applied the brain kernel to the visual recognition task and the BOLD5000 dataset except for the HCP dataset and the Sherlock movie dataset presented in the paper. For most subjects in these two datasets, the brain kernel didn't show a dominating performance over both the ridge covariance and the smooth RBF kernel (Appendix C).

The lack of benefit observed on non-HCP (outside the HCP) datasets may arise from a mismatch between the covariance of resting-state fMRI observations used to fit the brain kernel and the covariance of task fMRI datasets. However, it might also arise from differences in acquisition parameters, preprocessing steps, or alignment between HCP and non-HCP datasets. Although we aligned all voxels with the MNI template, misalignment might still result from differences between processing pipelines.

Outlook and future directions

Although the brain kernel did not achieve improved performance in all datasets and applications we considered, we feel it nevertheless holds great potential for fMRI data analysis. First, by using the Gaussian process latent variable modeling (GPLVM) framework, we cast the problem of estimating covariance of fMRI datasets as that of estimating a nonlinear mapping from 3D brain coordinates to a latent embedding space. This results in a compact representation of the full brain covariance matrix, requiring storage of only a $N_{voxels} \times 20$ matrix of embedding locations (when the embedding dimensionality is 20), as opposed to a full $N_{voxels} \times N_{voxels}$ covariance matrix. Moreover, because the brain kernel is a continuous covariance function, we can use it to model the covariance at arbitrary voxel locations, even those not contained in the original training dataset.

In addition to providing a compact parametrization of fMRI covariance, the brain kernel's nonlinear embedding function may be useful for gaining insights into the geometry of correlations within and across brain regions. Examining the embedding of different brain regions (e.g., as shown in Fig. 4) allows researchers to directly visualize correlations in terms of distances between embedded voxels.

Although the brain kernel fit to the HCP resting-state fMRI data did not yield improvements on all task fMRI datasets we explored, it is possible that other methods for training or applying the brain kernel might produce bigger gains. For example, one might train the brain kernel on task fMRI datasets, or train a hierarchical version that gains statistical strength from combining datasets, while preserving flexibility to capture differences between the two kinds of data. A more ambitious possibility is to formulate

a hierarchical version of the brain kernel that allows for per-subject variability. This would produce a hierarchical brain kernel in which each brain has own specific brain kernel, allowing for detailed differences in correlation maps across brains. All such applications will benefit from more robust methods for alignment and preprocessing, and it may be that these alone will be enough to improve the performance of the brain kernel.

Finally, an exciting possibility for future work is to combine the brain kernel with other advanced statistical modeling techniques. Methods for modeling structured variability, such as topographic ICA [39], and methods for structured sparsity, such as GraphNet [42], sparse overlapping sets lasso [43], and dependent relevance determination [44,45], rely on capturing dependencies between nearby voxels. All such methods might thus be improved by using nearness in functional embedding space provided by the brain kernel, instead of nearness in 3D Euclidean space inside the brain. Likewise, methods for linear alignment of functional data from multiple subjects such as hyperalignment [46,47] might be extended using nonlinear warping of brain coordinates under the brain kernel. Therefore, we feel that the brain kernel holds promise for inspiring new data-driven prior distributions and new modeling approaches for capturing structure in fMRI data.

Methods

Block coordinate descent for the brain kernel

The penalized least squares (eq. 15, PLS) has a computational complexity of $O(n^2)$ and memory storage of $O(n^2)$; however, the maximum a posteriori (eq. 18, MAP) has a computational complexity of $O(n^3)$ – to invert the covariance matrix – and memory storage of $O(n^2)$. n is the number of voxels, which often exceeds 10k. Gradient descent or Newton's method is computationally impractical as the optimizer. Thus, we need to use a scalable inference method. Existing inference methods for large datasets [17–19] exploit low-rank approximations to the full Gaussian process, which, however, suffer from a loss of accuracy in covariance estimation. Thus, in this section, we develop a block coordinate descent algorithm as an exact inference method for the brain kernel. Coordinate descent has been successfully applied to solve penalized regression models [20], to estimate covariance graphical lasso models [21], and to compute large-scale sparse inverse covariance matrices [4].

Our PLS and MAP estimators are non-convex smooth functions. We apply an iterative block coordinate descent method solved by the proximal Newton approach [22]. We first divide the voxel set $\{1, \ldots, n\}$ into blocks. Next, we iterate over all blocks, minimizing the functions with respect to the voxels within each block. Without loss of generality, we split the voxel set into two blocks, $\{1, \ldots, m\}$ (block 1) and $\{m+1, \ldots, n\}$ (block 2), where $m \ll n$, and focus on the first m columns of $\mathbf{Z}$ for the update. We partition $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{S}$, and $\mathbf{K}^{-1}$ as follows:

$$
\begin{aligned}
&\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \mathbf{Z}_2 \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{K}_{11}^{-1} & \mathbf{K}_{12}^{-1} \\ \mathbf{K}_{12}^{-\top} & \mathbf{K}_{22}^{-1} \end{bmatrix}, \quad \boldsymbol{\gamma} = \begin{bmatrix} c(\mathbf{z}_1, \mathbf{z}_1) & \cdots & c(\mathbf{z}_1, \mathbf{z}_m) \\ \vdots & \ldots & \vdots \\ c(\mathbf{z}_m, \mathbf{z}_1) & \cdots & c(\mathbf{z}_m, \mathbf{z}_m) \end{bmatrix}, \\
&\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{12}^{\top} & \mathbf{S}_{22} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \boldsymbol{\gamma} & \boldsymbol{\beta} \\ \boldsymbol{\beta}^{\top} & \mathbf{C}_{22} \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} c(\mathbf{z}_1, \mathbf{z}_{m+1}) & \cdots & c(\mathbf{z}_1, \mathbf{z}_n) \\ \vdots & \ldots & \vdots \\ c(\mathbf{z}_m, \mathbf{z}_{m+1}) & \cdots & c(\mathbf{z}_m, \mathbf{z}_n) \end{bmatrix}.
\end{aligned}
\tag{27}
$$

Here, subscripts represent the block indices, so $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are the first m columns and the last $n-m$ columns of $\mathbf{Z}$ and subscript 12 indicates the block matrix across the first m variables and the last $n-m$ variables. Only $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ contain the active variables in $\mathbf{Z}_1$ to optimize. $\mathbf{Z}_2$ is fixed.

Applying the block representation to eq. 15, we get the block PLS objective function for solving $\mathbf{Z}_1$:

$$\begin{aligned} l_{\text{PLS}}(\mathbf{Z}_1) = &\text{tr}\left[(\mathbf{S}_{11} - \boldsymbol{\gamma} - \sigma^2\mathbf{I}_n)(\mathbf{S}_{11} - \boldsymbol{\gamma} - \sigma^2\mathbf{I}_n)^\top + 2(\mathbf{S}_{12} - \boldsymbol{\beta})(\mathbf{S}_{12} - \boldsymbol{\beta})^\top\right] \\ &+ \text{tr}\left[(\mathbf{Z}_1 - \mathbf{B}\mathbf{X}_1)\mathbf{K}_{11}^{-1}(\mathbf{Z}_1 - \mathbf{B}\mathbf{X}_1)^\top + 2(\mathbf{Z}_1 - \mathbf{B}\mathbf{X}_1)\mathbf{K}_{12}^{-1}(\mathbf{Z}_2 - \mathbf{B}\mathbf{X}_2)^\top\right], \end{aligned} \tag{28}$$

where $\mathbf{X}_1$ and $\mathbf{X}_2$ are the first m columns and the last $n-m$ columns of $\mathbf{X}$.

To formulate the block MAP estimator for $\mathbf{Z}_1$, we first apply the block matrix inversion to $\mathbf{C}$,

$$\mathbf{C}^{-1} = \begin{bmatrix} (\boldsymbol{\gamma} - \boldsymbol{\beta}\mathbf{C}_{22}^{-1}\boldsymbol{\beta}^\top)^{-1}, & -\boldsymbol{\gamma}^{-1}\boldsymbol{\beta}(\mathbf{C}_{22} - \boldsymbol{\beta}^\top\boldsymbol{\gamma}^{-1}\boldsymbol{\beta})^{-1} \\ -\mathbf{C}_{22}^{-1}\boldsymbol{\beta}^\top(\boldsymbol{\gamma} - \boldsymbol{\beta}\mathbf{C}_{22}^{-1}\boldsymbol{\beta}^\top)^{-1}, & (\mathbf{C}_{22} - \boldsymbol{\beta}^\top\boldsymbol{\gamma}^{-1}\boldsymbol{\beta})^{-1} \end{bmatrix}. \tag{29}$$

Incorporating this matrix inversion into the MAP estimator, we get the objective function

$$\begin{aligned} l_{\text{MAP}}(\mathbf{Z}_1) \quad = \quad & \log|\boldsymbol{\gamma} + \sigma^2\mathbf{I}_n| + \text{tr}\left[\boldsymbol{\beta}\mathbf{C}_{22}^{-1}\mathbf{S}_{22}\mathbf{C}_{22}^{-1}\boldsymbol{\beta}^\top(\boldsymbol{\gamma} + \sigma^2\mathbf{I}_n)^{-1}\right] \\ & -2\text{tr}\left[\mathbf{S}_{12}\mathbf{C}_{22}^{-1}\boldsymbol{\beta}^\top(\boldsymbol{\gamma} + \sigma^2\mathbf{I}_n)^{-1}\right] + \text{tr}\left[\mathbf{S}_{11}(\boldsymbol{\gamma} + \sigma^2\mathbf{I}_n)^{-1}\right] \\ & +\text{tr}\left[(\mathbf{Z}_1 - \mathbf{B}\mathbf{X}_1)\mathbf{K}_{11}^{-1}(\mathbf{Z}_1 - \mathbf{B}\mathbf{X}_1)^\top + 2(\mathbf{Z}_1 - \mathbf{B}\mathbf{X}_1)\mathbf{K}_{12}^{-1}(\mathbf{Z}_2 - \mathbf{B}\mathbf{X}_2)^\top\right]. \end{aligned} \tag{30}$$

The time complexity of the block PLS estimator is $O(nm^2)$ per iteration, where m is the size of the block, and the time complexity of the MAP estimator is $O(n^2m)$ per iteration. For comparison, greedy gradient descent has complexity $O(n^3)$ per iteration. In the experiments, the block MAP estimator for $\mathbf{Z}$ with the block PLS estimator as a warm start is a practical optimization approach.

We now describe the block coordinate descent (BCD) update. We assume that the voxel indices $\{1, \ldots, n\}$ are divided into k blocks $\{I_j\}_{j=1}^k$, where I_j is the set of indices corresponding to the columns of $\mathbf{Z}$ in the j'th block. Denote I_j columns of $\mathbf{Z}$ to be $\mathbf{Z}_{I_j}$. We cluster indices into blocks based on the spatial locations of the voxels and assume smooth measurements for nearby voxels. Thus, the size of a block should be at least one length scale of the region defined by the kernel in $\mathbf{x}$ space to encourage dependencies among neighboring voxels. This smoothness assumption leads to a block-wise but not an element-wise update, which separates our BCD method from Informative Vector Machine (IVM) [48]. At each iteration t, we choose a nonempty index subset $I \in \{I_j\}_{j=1}^k$. Then the objective functions $l(\mathbf{Z}_I)$ in eq. 28 and 30 are optimized w.r.t. $\mathbf{Z}_I$ via L-BFGS [49]. After the t'th iteration, $\mathbf{Z}_I$ is updated, holding all other blocks fixed.

For high-dimensional non-convex problems, a good initialization is essential to finding a good optimum. Here we describe an effective and tractable initialization for a single block of $\mathbf{Z}$, inspired by Laplacian eigenmaps.

The Laplacian eigenmap (LE) algorithm is a popular dimensionality reduction method that solves a generalized eigendecomposition [50]. LE defines a neighborhood graph on the data $\{\mathbf{y}_i \in \mathbb{R}^T\}_{i=1}^n$, such as k nearest neighbors or an ϵ-ball graph, and weighs each graph edge $\mathbf{y}_i \sim \mathbf{y}_j$ by a symmetric affinity function $V(\mathbf{y}_i, \mathbf{y}_j) = v_{ij}$, typically Gaussian: $v_{ij} = \exp\left(-\frac{||\mathbf{y}_i - \mathbf{y}_j||^2}{2s^2}\right)$ with s the length scale. Given this weighted graph, LE seeks latent points $\{\mathbf{z}_i \in \mathbb{R}^d\}_{i=1}^n$ that are solutions to the optimization problem

$$\min_{\mathbf{Z}} \text{tr}(\mathbf{Z}^\top\mathbf{L}\mathbf{Z}) \quad s.t. \quad \mathbf{Z}^\top\mathbf{D}\mathbf{Z} = \mathbf{I},\ \mathbf{Z}^\top\mathbf{D}\mathbf{1} = \mathbf{I}, \tag{31}$$

with the symmetric affinity matrix $\mathbf{V} \in \mathbb{R}^{n\times n}$, the degree matrix $\mathbf{D} = \text{diag}(\sum_{i=1}^n v_{ij}) \in \mathbb{R}^{n\times n}$, the graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{V} \in \mathbb{R}^{n\times n}$, and $\mathbf{1} = [1, \ldots, 1]^\top$. Constraints eliminate the two trivial solutions $\mathbf{Z} = \mathbf{0}$ by setting an arbitrary scale and $\mathbf{z}_1 = ... = \mathbf{z}_n$ by removing $\mathbf{1}$, which is an eigenvector of $\mathbf{L}$ associated with a zero eigenvalue.

Following the previous two-block example, let $\mathbf{Z}$ be partitioned into $\mathbf{Z}_1$ and $\mathbf{Z}_2$. To update $\mathbf{Z}_1$ given $\mathbf{Z}_2$, the objective function is:

$$\min_{\mathbf{Z}_1} \text{tr}\left(\begin{bmatrix}\mathbf{Z}_1 & \mathbf{Z}_2\end{bmatrix}\begin{bmatrix}\mathbf{L}_{11} & \mathbf{L}_{12} \\ \mathbf{L}_{12}^\top & \mathbf{L}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{Z}_1^\top \\ \mathbf{Z}_2^\top\end{bmatrix}\right). \tag{32}$$

We don't need to use the constraints from eq. 31 because the trivial solutions are removed given $\mathbf{Z}_2$. The solution is

$$\widetilde{\mathbf{Z}}_1 = -\mathbf{Z}_2\mathbf{L}_{12}^\top\mathbf{L}_{11}^{-1}. \tag{33}$$

Given the current latent embeddings for all other coordinates, the algorithm seeks the best latent embedding of the unknown dimensions. Because of the computational efficiency of this approach, we use $\widetilde{\mathbf{Z}}_1$ to initialize the BCD algorithm for each iteration and $\mathbf{Z}_2$ collapses all the fixed blocks.

In addition, if $\mathbf{V}$ has a Gaussian affinity function, the latent embedding $\mathbf{Z}$ is mapped from the observation $\mathbf{Y}$ with a radial basis function nonlinearity. For covariance estimation, we enforce the resemblance between another radial basis function (RBF) kernel on $\mathbf{Z}$ and the sample covariance of $\mathbf{Y}$. We are essentially trying to map the observation space to itself with double layers of exponential transformations, which would result in a bad latent embedding for initialization. Therefore, instead of using a Gaussian function for the weights $\mathbf{V}$, we use a function that inverts the RBF nonlinearity on a covariance matrix, defined as $v_{ij} = -f(\mathbf{y}_i\mathbf{y}_j^\top)$. Here, $f(x) = \text{sign}(x)\log(|x|+1)$ is the log-modulus transformation [51], which distributes the magnitude of the data while preserving the sign of the data in order to control against negative covariance values when taking the logarithm.

Hyperparameter estimation for the brain kernel

Our model includes five hyperparameters: $\{\delta, r, \mathbf{B}, \rho, \sigma^2\}$. We set these parameters as follows.

Estimate δ: δ is the length scale of the GP kernel mapping from coordinate space to latent space. We can optimize this parameter by taking the derivative of the GP log-likelihood w.r.t. δ; this involves inverting the kernel matrix with an $O(n^3)$ cost, which is computationally infeasible here. An inducing-point method [18] introduced extra inducing variables to optimize, which further increased the computational burden. Instead, we use a scalable spectral formulation for learning the length scale δ of the GP kernel.

For a stationary Gaussian process $f \sim \mathcal{GP}(m, k)$, when f has a high degree of smoothness, the prior covariance $\mathbf{K}$ becomes approximately low rank, meaning that it has a small number of non-negligible eigenvalues. Because the kernel function for $\mathbf{x}$ space is shift invariant, the eigenspectrum of $\mathbf{K}$ has a diagonal representation in the Fourier domain, a consequence of Bochner's theorem [52–54],

$$g \sim \mathcal{N}(\alpha(\boldsymbol{\omega}), \Sigma(\boldsymbol{\omega})), \tag{34}$$

where g is the Fourier transform of f, $\boldsymbol{\omega}$ is the frequency, $\alpha(\boldsymbol{\omega})$ is the Fourier transform of the mean function $m(\mathbf{x})$, and $\Sigma = \text{diag}(s(\boldsymbol{\omega}))$ is diagonal. This means that Fourier components are a priori independent, with prior variance

$$s(\boldsymbol{\omega}) = (2\pi\delta^2)^{h/2} r \, \exp\left(-2\pi^2\delta^2\boldsymbol{\omega}^2\right), \tag{35}$$

where h is the dimension of the input $\mathbf{x}$. Without loss of generality, we assume the size of the spectral domain for each input dimension is w, and thus $\boldsymbol{\alpha} = \alpha(\boldsymbol{\omega}) \in \mathbb{R}^{w^h}$ and $\boldsymbol{\Sigma} = \Sigma(\boldsymbol{\omega}) \in \mathbb{R}^{w^h \times w^h}$ given a spectral representation $\boldsymbol{\omega}$. The mapping between $f(\mathbf{x})$ and its Fourier transform $g(\boldsymbol{\omega})$ is then

$$f(\mathbf{x}) = \sum_j e^{2\pi i \mathbf{x}^\top \boldsymbol{\omega}_j} g(\boldsymbol{\omega}_j) = \mathbf{e}^\top g(\boldsymbol{\omega}), \tag{36}$$

where $\mathbf{e}$ is a column vector with entries $e^{2\pi i \mathbf{x}^\top \boldsymbol{\omega}_j}$ on the j'th position for the j'th frequency. Let $\mathbf{e}_i \in \mathbb{R}^{w^h}$ denote the Fourier vector for $\mathbf{x}_i$, then we can further define $\mathcal{B} = (\mathbf{e}_1, \ldots, \mathbf{e}_n) \in \mathbb{R}^{w^h \times n}$ to be the Fourier matrix for an input coordinate matrix $\mathbf{X}$. Let $\mathbf{z}_j \in \mathbb{R}^n$ denote the j'th latent embedding of $\mathbf{X}$ and $j \in \{1, \ldots, d\}$. Note that $\mathbf{z}_j$ is the j'th row of the matrix $\mathbf{Z}$. We can write $\mathbf{z}_j = \mathbf{g}_j^\top \mathcal{B}$, where

$\mathbf{g}_j \in \mathbb{R}^{w^h} \sim \mathcal{N}(\boldsymbol{\alpha}_j, \boldsymbol{\Sigma})$. This implies that each latent embedding deterministically depends on a unique spectral function. These spectral functions are all sampled from multivariate Gaussian priors with different mean functions but the same covariance function in the spectral domain. Bringing the Fourier matrix $\mathcal{B}$ into the Gaussian prior (eq. 34), we derive the Gaussian prior over the latent $\mathbf{z}_j$ expressed with the spectral formulation as

$$\mathbf{z}_j \sim \mathcal{N}\left(\boldsymbol{\alpha}_j^\top \mathcal{B}, \mathcal{B}^\top \boldsymbol{\Sigma} \mathcal{B}\right). \tag{37}$$

We can use this representation to optimize δ. If δ is in a large-value regime, we can ignore Fourier components above a certain high-frequency cutoff which leads to a lower-dimensional $\boldsymbol{\omega}$ and a lower-dimensional optimization problem. Because $h = 3$ in fMRI analyses, we can control w to be small enough so that w^h is tractable relative to the large n, and $\mathcal{B}$ is a manageable high-dimensional matrix. Because we generally do not assume uniform gridding of coordinates $\mathbf{X}$, $\mathcal{B}$ is a non-uniform DFT (not orthogonal). Although the Kronecker tricks cannot be used for scaling up with non-uniform DFT, we can employ block matrix inverse lemma to transform the spectral formulation into a lower-dimensional problem.

Then the standard negative GP log-likelihood to optimize for δ is

$$l(\delta) = \text{tr}\left[(\mathbf{Z} - \mathbf{B}\mathbf{X})(\mathbf{K} + \epsilon\mathbf{I})^{-1}(\mathbf{Z} - \mathbf{B}\mathbf{X})^\top\right] + \log|\mathbf{K} + \epsilon\mathbf{I}|, \tag{38}$$

where ϵ is a small noise variance in the latent space to avoid ill-conditions for $\mathbf{K}$. Let $\mathbf{A} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d)^\top \in \mathbb{R}^{d \times w^h}$. With the spectral representation, eq. 38 can be re-written as

$$l(\delta) = \text{tr}\left[(\mathbf{Z} - \mathbf{A}\mathcal{B})(\mathcal{B}^\top \boldsymbol{\Sigma}\mathcal{B} + \epsilon\mathbf{I})^{-1}(\mathbf{Z} - \mathbf{A}\mathcal{B})^\top\right] + \log|\mathcal{B}^\top \boldsymbol{\Sigma}\mathcal{B} + \epsilon\mathbf{I}|. \tag{39}$$

Calculating $(\mathcal{B}^\top \boldsymbol{\Sigma}\mathcal{B} + \epsilon\mathbf{I})^{-1}$ is still computationally expensive, but with the spectral factorization, we are able to use the block matrix inverse lemma as in eq. 29,

$$\begin{aligned}
&(\mathcal{B}^\top \boldsymbol{\Sigma}\mathcal{B} + \epsilon\mathbf{I})^{-1} = \\
&\begin{bmatrix} \mathbf{I} - \mathcal{B}_1^\top(\mathbf{P}_2^{-1} + \mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\mathcal{B}_1, & -\mathcal{B}_1^\top(\boldsymbol{\Sigma}\mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\boldsymbol{\Sigma}(\mathcal{B}_{22}\boldsymbol{\Sigma} - \mathcal{B}_{22}\boldsymbol{\Sigma}(\mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_2 \\ -\mathcal{B}_2^\top(\boldsymbol{\Sigma}\mathcal{B}_{22} + \epsilon\mathbf{I})^{-1}\boldsymbol{\Sigma}(\mathcal{B}_{11}\boldsymbol{\Sigma} - \mathcal{B}_{11}\boldsymbol{\Sigma}(\mathcal{B}_{22}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_{22}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_1, & \mathbf{I} - \mathcal{B}_2^\top(\mathbf{P}_1^{-1} + \mathcal{B}_{22} + \epsilon\mathbf{I})^{-1}\mathcal{B}_2 \end{bmatrix}
\end{aligned} \tag{40}$$

where $\mathcal{B}_1$ and $\mathcal{B}_2$ correspond to the Fourier bases for $\mathbf{X}_1$ and $\mathbf{X}_2$ respectively. Note that neither $\mathcal{B}_1$ nor $\mathcal{B}_2$ is invertible. Moreover, $\mathcal{B}_{11} = \mathcal{B}_1\mathcal{B}_1^\top$, $\mathcal{B}_{22} = \mathcal{B}_2\mathcal{B}_2^\top$, $\mathbf{P}_1 = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}(\mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_{11}\boldsymbol{\Sigma}$ and $\mathbf{P}_2 = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}(\mathcal{B}_{22}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_{22}\boldsymbol{\Sigma}$. All matrices have size $w^h \times w^h$, which is tractable to invert. We know that the spectral expression of $\mathbf{K}^{-1}$ is $(\mathcal{B}^\top \boldsymbol{\Sigma}\mathcal{B} + \epsilon\mathbf{I})^{-1}$. We can also express $\mathbf{K}_{11}^{-1}$ and $\mathbf{K}_{12}^{-1}$ with the spectral formulation as

$$\begin{aligned}
\mathbf{K}_{11}^{-1} &= \mathbf{I} - \mathcal{B}_1^\top(\mathbf{P}_2^{-1} + \mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\mathcal{B}_1, \\
\mathbf{K}_{12}^{-1} &= -\mathcal{B}_1^\top(\boldsymbol{\Sigma}\mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\boldsymbol{\Sigma}(\mathcal{B}_{22}\boldsymbol{\Sigma} - \mathcal{B}_{22}\boldsymbol{\Sigma}(\mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_2.
\end{aligned} \tag{41}$$

Consequently, the negative GP log-likelihood w.r.t. δ in the block form is represented as

$$\begin{aligned}
l(\delta) \quad = \quad & (\mathbf{Z}_1 - \mathbf{A}\mathcal{B}_1)(\mathbf{I} - \mathcal{B}_1^\top(\mathbf{P}_2^{-1} + \mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\mathcal{B}_1)(\mathbf{Z}_1 - \mathbf{A}\mathcal{B}_1)^\top \\
& + (\mathbf{Z}_2 - \mathbf{A}\mathcal{B}_2)(\mathbf{I} - \mathcal{B}_2^\top(\mathbf{P}_1^{-1} + \mathcal{B}_{22} + \epsilon\mathbf{I})^{-1}\mathcal{B}_2)(\mathbf{Z}_2 - \mathbf{A}\mathcal{B}_2)^\top \\
& + \log|\boldsymbol{\Sigma}\mathcal{B}_{22} - \boldsymbol{\Sigma}\mathcal{B}_{11}(\boldsymbol{\Sigma}\mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\boldsymbol{\Sigma}\mathcal{B}_{22} + \epsilon\mathbf{I}| + \log|\boldsymbol{\Sigma}\mathcal{B}_{11} + \epsilon\mathbf{I}| \\
& - 2(\mathbf{Z}_1 - \mathbf{A}\mathcal{B}_1)\mathcal{B}_1^\top(\boldsymbol{\Sigma}\mathcal{B}_{11} + \epsilon\mathbf{I})^{-1}\boldsymbol{\Sigma}(\mathcal{B}_{22}\boldsymbol{\Sigma} - \mathcal{B}_{22}\boldsymbol{\Sigma}(\mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1} \\
& \cdot \mathcal{B}_{11}\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}\mathcal{B}_2(\mathbf{Z}_2 - \mathbf{A}\mathcal{B}_2)^\top.
\end{aligned} \tag{42}$$

With eq. 42, the computational cost is reduced to $\max\big(O(nw^h), O(w^{3h})\big)$, where w is the dimension of the spectral form per input dimension and h is the number of input dimensions. This complexity is much smaller than $O(n^3)$ when $n \gg w$ and $h \leq 3$. Another benefit of this formulation is that δ only exists in the diagonal of $\boldsymbol{\Sigma}$ (eq. 35), which makes the estimation straightforward.

Algorithm 1 Block Coordinate Descent Algorithm for Brain Kernel

Input: sample covariance $\mathbf{S}$, voxel coordinate $\mathbf{X}$
Output: latent variable $\mathbf{Z}$, linear projection $\mathbf{B}$, length scale δ, marginal variance r
Generate the index set $\{I_j\}_{j=1}^k$ given $\mathbf{X}$
Randomly initialize $\mathbf{Z}$
for $t = 1, 2, ...$ **do**
  Pick $I \in \{I_j\}_{j=1}^k$, and solve eq. 33 to get an initialization for $\mathbf{Z}_I^{(t)}$
  Find the optimal $\mathbf{Z}_I^{(t)}$ and $\mathbf{B}^{(t)}$ by solving eq. 28 or 30
  Estimate the optimal length scale δ^* and the optimal marginal variance r^* by solving eq. 42
  Update $\mathbf{K}_{11}^{-1(t)}$ and $\mathbf{K}_{12}^{-1(t)}$ given δ^* and r^* according to eq. 41
end for

Estimate r: r determines the scale of the latent embedding $\mathbf{Z}$. Since r also exists on the diagonal of $\mathbf{\Sigma}$ (eq. 35), the optimization for r can be combined with learning δ using the same spectral representation in eq. 42.
Estimate ρ: ρ is the marginal variance of the covariance function. We assume that the data has been normalized to have zero mean and variance one. Thus we set $\rho = 1$.
Estimate $\mathbf{B}$: $\mathbf{B}$ is the linear projection of the mean function. We can estimate $\mathbf{B}$ jointly with $\mathbf{Z}$ in each BCD iteration by optimizing the same objective function (eq. 28 and 30).
Estimate σ^2: σ^2 is the observation noise variance. We estimate σ^2 using the eigenvalues of the sample covariance $\mathbf{S}$ [55].

Algorithm 1 describes the complete BCD algorithm for the brain kernel. The convergence of the BCD algorithm (without parameter estimation) to a stationary point is addressed in the theoretical results in previous work [22]. There, a general block-coordinate-descent approach is analyzed to solve minimization problems of the form $F(x) = f(x) + \lambda h(x)$, which is composed of the sum of a smooth function $f(\cdot)$ and a separable convex function $h(\cdot)$, in our case $h(x) = 0$. According to Part (e) of Theorem 4.1 in [22], if I_t at the t'th iteration is chosen by the generalized Gauss-Seidel rule,

$$\bigcup_{i=0,1,\ldots,T-1} J_{i+t} \supseteq \mathcal{V} \quad \forall t = 1, 2, \ldots, T, \tag{43}$$

which is necessary to ensure that all variables are updated every T steps and $\mathcal{V}$ is the set of all variables, then each coordinate-wise minimum point of $\{\mathbf{Z}_{I_t}\}$ is a stationary point of $f(\mathbf{Z})$.

Hyperparameter optimization and latent variable inference for Bayesian factor analysis

Our goal was to infer the latent variables $\mathbf{L}$ and $\mathbf{F}$ and the hyperparameters for $\mathbf{C}$ and σ^2, denoted as θ. We assumed that

$$\mathbf{X} = \mathbf{LF} + \epsilon \sim \mathcal{N}(\mathbf{LF}, \sigma^2\mathbf{I}), \tag{44}$$

$$\mathbf{L}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{C}), \quad \text{for the } j\text{th column}, \tag{45}$$

and

$$\mathbf{F}_i = \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \text{for the } i\text{th column}. \tag{46}$$

We first aimed at estimating the hyperparameters θ by marginalizing over $\mathbf{L}$ and $\mathbf{F}$. The marginal distribution for $\mathbf{X}$ is

$$p(\mathbf{X}) = \int \mathcal{N}(\mathbf{LF}, \sigma^2\mathbf{I}) \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) \prod_{i=1}^{T} \mathcal{N}(\mathbf{F}_i|\mathbf{0}, \mathbf{I}) d\mathbf{L}_1 \dots d\mathbf{L}_K d\mathbf{F}_1 \dots d\mathbf{F}_T, \tag{47}$$

which is intractable. However, we could match the second order moment of $\mathbf{X}$ to the sample covariance $\mathbf{S} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top$ where $\tilde{\mathbf{X}}$ is the standardized data sample. The second order moment is derived as follows

$$\begin{aligned}
\mathbb{E}[\mathbf{XX}^\top] &= \mathbb{E}[\mathbf{LFF}^\top\mathbf{L}^\top] + \sigma^2 T\mathbf{I} \\
&= \int\int \mathbf{LFF}^\top\mathbf{L}^\top \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) \prod_{i=1}^{T} \mathcal{N}(\mathbf{F}_i|\mathbf{0}, \mathbf{I}) d\mathbf{L}_1 \dots d\mathbf{L}_K d\mathbf{F}_1 \dots d\mathbf{F}_T + \sigma^2 T\mathbf{I} \\
&= \int \mathbf{L} \left[\int \mathbf{FF}^\top \prod_{i=1}^{T} \mathcal{N}(\mathbf{F}_i|\mathbf{0}, \mathbf{I}) d\mathbf{F}_1 \dots d\mathbf{F}_T \right] \mathbf{L}^\top \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) d\mathbf{L}_1 \dots d\mathbf{L}_K + \sigma^2 T\mathbf{I} \\
&= \int \mathbf{L} \left[\int \sum_{i=1}^{T} \mathbf{F}_i\mathbf{F}_i^\top \prod_{i=1}^{T} \mathcal{N}(\mathbf{F}_i|\mathbf{0}, \mathbf{I}) d\mathbf{F}_1 \dots d\mathbf{F}_T \right] \mathbf{L}^\top \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) d\mathbf{L}_1 \dots d\mathbf{L}_K + \sigma^2 T\mathbf{I} \\
&= \int \mathbf{L} \left[\sum_{i=1}^{T} \int \mathbf{F}_i\mathbf{F}_i^\top \mathcal{N}(\mathbf{F}_i|\mathbf{0}, \mathbf{I}) d\mathbf{F}_i \right] \mathbf{L}^\top \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) d\mathbf{L}_1 \dots d\mathbf{L}_K + \sigma^2 T\mathbf{I} \\
&= T \int \mathbf{LL}^\top \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) d\mathbf{L}_1 \dots d\mathbf{L}_K + \sigma^2 T\mathbf{I} \\
&= T \sum_{j=1}^{K} \int \mathbf{L}_j\mathbf{L}_j^\top \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}) d\mathbf{L}_j + \sigma^2 T\mathbf{I} \\
&= TK\mathbf{C} + \sigma^2 T\mathbf{I}.
\end{aligned} \tag{48}$$

By matching eq. 48 to the sample covariance $\mathbf{S}$, we were able to estimate the hyperparameters in $\mathbf{C}$ and σ^2.

Next, we fixed the estimated hyperparameters and inferred $\mathbf{L}$. Marginalizing over $\mathbf{F}$ we got

$$p(\mathbf{X}|\mathbf{L}) = \mathcal{N}(\mathbf{0}, \mathbf{LL}^\top + \sigma^2\mathbf{I}). \tag{49}$$

Then, we obtained the joint distribution between $\mathbf{X}$ and $\mathbf{L}$ as

$$p(\mathbf{X}, \mathbf{L}) = \mathcal{N}(\mathbf{0}, \mathbf{LL}^\top + \sigma^2\mathbf{I}) \prod_{j=1}^{K} \mathcal{N}(\mathbf{L}_j|\mathbf{0}, \mathbf{C}), \tag{50}$$

whose log likelihood is written as

$$\mathcal{L}(\mathbf{L}) = -\frac{1}{2}\log|\mathbf{LL}^\top + \sigma^2\mathbf{I}| - \frac{1}{2T}\mathrm{Tr}\left(\mathbf{X}^\top(\mathbf{LL}^\top + \sigma^2\mathbf{I})^{-1}\mathbf{X}\right) - \frac{1}{2K}\mathrm{Tr}\left(\mathbf{L}^\top\mathbf{C}^{-1}\mathbf{L}\right). \tag{51}$$

We maximized eq. 51 w.r.t. $\mathbf{L}$. To speed up the optimization process, we initialized $\mathbf{L}$ via finding a closed-form solution approximately maximizing eq. 51. Denoting $\mathbf{LL}^\top + \sigma^2\mathbf{I}$ as $\mathbf{Q}$, we rewrote the log likelihood as

$$\mathcal{L}(\mathbf{Q}) = -\frac{1}{2}\log|\mathbf{Q}| - \frac{1}{2T}\mathrm{Tr}\left(\mathbf{Q}^{-1}\mathbf{XX}^\top\right) - \frac{1}{2K}\mathrm{Tr}\left(\mathbf{C}^{-1}\mathbf{Q}\right), \tag{52}$$

whose derivative is

$$\frac{\partial \mathcal{L}(\mathbf{Q})}{\partial \mathbf{Q}} = -\frac{1}{2}\mathbf{Q}^{-1} + \frac{1}{2T}\mathbf{Q}^{-1}\mathbf{X}\mathbf{X}^\top\mathbf{Q}^{-1} - \frac{1}{2K}\mathbf{C}^{-1}. \tag{53}$$

Setting the derivative to be 0, we arrived at

$$\begin{gathered}\frac{1}{K}\mathbf{Q}\mathbf{C}^{-1}\mathbf{Q} + \mathbf{Q} = \frac{1}{T}\mathbf{X}\mathbf{X}^\top \\ \Downarrow \\ (\mathbf{Q} + \frac{1}{2}K\mathbf{C})\frac{1}{K}\mathbf{C}^{-1}(\mathbf{Q} + \frac{1}{2}K\mathbf{C}) = \frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C}.\end{gathered} \tag{54}$$

We needed to decompose the right-hand side into the multiplication of a symmetric matrix, $\mathbf{C}^{-1}$ and the same symmetric matrix. Here is our solution:

- Let $\mathbf{P}$ denote the Cholesky decomposition of $\frac{1}{K}\mathbf{C}^{-1}$, i.e., $\frac{1}{K}\mathbf{C}^{-1} = \mathbf{P}\mathbf{P}^\top$.
- Denote $\mathbf{Q} + \frac{1}{2}K\mathbf{C} = \mathbf{B}\mathbf{B}^\top$ and let $\mathbf{B} = \mathbf{P}^{-\top}\mathbf{A}$ where $\mathbf{A}$ is an unknown square matrix.
- Then we can rewrite eq. 54 as

$$\begin{aligned}\frac{1}{K}\mathbf{B}\mathbf{B}^\top\mathbf{C}^{-1}\mathbf{B}\mathbf{B}^\top &= \frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C} \\ \Rightarrow \quad \mathbf{P}^{-\top}\mathbf{A}\mathbf{A}^\top\mathbf{P}^{-1}\mathbf{P}\mathbf{P}^\top\mathbf{P}^{-\top}\mathbf{A}\mathbf{A}^\top\mathbf{P}^{-1} &= \frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C} \\ \Rightarrow \quad \mathbf{P}^{-\top}\mathbf{A}\mathbf{A}^\top\mathbf{A}\mathbf{A}^\top\mathbf{P}^{-1} &= \frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C} \\ \Rightarrow \quad \mathbf{A}\mathbf{A}^\top\mathbf{A}\mathbf{A}^\top &= \mathbf{P}^\top(\frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C})\mathbf{P}.\end{aligned}$$

- Next, we represent $\mathbf{A}$ with its singular value decomposition (SVD), i.e., $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$.
- Finally we can rewrite eq. 54 as

$$\begin{aligned}\mathbf{A}\mathbf{A}^\top\mathbf{A}\mathbf{A}^\top &= \mathbf{P}^\top(\frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C})\mathbf{P} \\ \Rightarrow \quad \mathbf{U}\mathbf{S}\mathbf{V}^\top\mathbf{V}\mathbf{S}\mathbf{U}^\top\mathbf{U}\mathbf{S}\mathbf{V}^\top\mathbf{V}\mathbf{S}\mathbf{U}^\top &= \mathbf{P}^\top(\frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C})\mathbf{P} \\ \Rightarrow \quad \mathbf{U}\mathbf{S}^4\mathbf{U}^\top &= \mathbf{P}^\top(\frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C})\mathbf{P}.\end{aligned} \tag{55}$$

- Therefore, to solve eq. 54, we first factorize $\mathbf{P}^\top(\frac{1}{T}\mathbf{X}\mathbf{X}^\top + \frac{1}{4}K\mathbf{C})\mathbf{P}$ using SVD to obtain $\mathbf{U}$ and $\mathbf{S}$ according to eq. 55.
- Then $\mathbf{Q} = \mathbf{B}\mathbf{B}^\top - \frac{1}{2}K\mathbf{C} = \mathbf{P}^{-\top}\mathbf{A}\mathbf{A}^\top\mathbf{P}^{-1} - \frac{1}{2}K\mathbf{C} = \mathbf{P}^{-\top}\mathbf{U}\mathbf{S}^2\mathbf{U}^\top\mathbf{P}^{-1} - \frac{1}{2}K\mathbf{C}$.
- Ultimately, $\mathbf{L}$ can be obtained via $\mathbf{L} = \mathbf{W}_{:,:K}\mathbf{D}^{\frac{1}{2}}_{:K,:K}$ where $\mathbf{W}\mathbf{D}\mathbf{W}^\top$ is the eigen-decomposition of $\mathbf{Q} - \sigma^2\mathbf{I}$.

Given the above procedure, we were able to find an ideal initialization for $\mathbf{L}$ which made the optimization of eq. 51 much easier. Conditioned on the optimal $\mathbf{L}^*$, we turned to inferring the optimal $\mathbf{F}^*$ via maximizing

$$\mathcal{L}(\mathbf{F}) = \log \mathcal{N}(\mathbf{X}|\mathbf{L}^*\mathbf{F}, \sigma^2\mathbf{I}) \prod_{i=1}^{T} \mathcal{N}(\mathbf{F}_i|\mathbf{0}, \mathbf{I}), \tag{56}$$

which has a closed-form solution, i.e., $\mathbf{F}^* = (\mathbf{L}^{*\top}\mathbf{L}^* + \sigma^2\mathbf{I})^{-1}\mathbf{L}^{*\top}\mathbf{X}$.

Note that in order to obtain $\mathbf{L} = \mathbf{W}_{:,:K}\mathbf{D}^{\frac{1}{2}}_{:K,:K}$, we need to guarantee that we are able to find a positive semi-definite (p.s.d.) matrix $\mathbf{Q} - \sigma^2\mathbf{I}$. Here are the lemma and the theorem:

Lemma. *If $\mathbf{A}$ is p.s.d., and $\mathbf{B}$ is symmetric p.s.d., then $\mathbf{AB}$ is also p.s.d..*

Proof. If $\mathbf{A}$ and $\mathbf{B}$ are both p.s.d. and $\mathbf{B}$ is also symmetric, then suppose λ is an eigenvalue of $\mathbf{AB}$ with corresponding eigenvector $\mathbf{x} \neq 0$, i.e., $\mathbf{ABx} = \lambda\mathbf{x}$. Then $\mathbf{BABx} = \lambda\mathbf{Bx}$ and so $\mathbf{x}^\top\mathbf{BABx} = \lambda\mathbf{x}^\top\mathbf{Bx}$. It is not hard to check that $\mathbf{BAB}$ will also be p.s.d.. For $\mathbf{x}$ s.t. $\mathbf{x}^\top\mathbf{Bx} \neq 0$, we have $\lambda = \frac{\mathbf{x}^\top\mathbf{BABx}}{\mathbf{x}^\top\mathbf{Bx}}$. Both the numerator and the denominator are non-negative values, therefore $\lambda \geq 0$. For $\mathbf{x}$ s.t. $\mathbf{x}^\top\mathbf{Bx} = 0$, we assume $\mathbf{x} = \mathbf{Ve}$ where $\mathbf{B} = \mathbf{VDV}^\top$ is the eigen-decomposition and $\mathbf{e}$ is the linear weight vector. Then $\mathbf{x}^\top\mathbf{Bx} = \mathbf{e}^\top\mathbf{V}^\top\mathbf{BVe} = \mathbf{e}^\top\mathbf{V}^\top\mathbf{VDe} = \mathbf{e}^\top\mathbf{De} = 0$. Since $\mathbf{D}$ is a diagonal matrix with non-negative values, $\mathbf{e}$ should have zero elements corresponding to the non-zero eigenvalues. Therefore $\mathbf{x}$ is a linear combination of eigenvectors of $\mathbf{B}$ whose eigenvalues are zero, i.e., $\mathbf{x} = \mathbf{V}_0\mathbf{e}$ where $\mathbf{V}_0$ contains all zero eigenvectors with $\mathbf{D}_0 = \mathbf{0}$ and $\mathbf{e}$ has no zero elements. Following that, we have $\mathbf{Bx} = \mathbf{BV}_0\mathbf{e} = \mathbf{V}_0\mathbf{D}_0\mathbf{e} = 0 \Rightarrow \mathbf{ABx} = \lambda_\mathbf{B}\mathbf{Ax} = 0 \Rightarrow \lambda = 0$. Based on the derivation, we arrive at the conclusion that $\mathbf{AB}$ has non-negative eigenvalues thus $\mathbf{AB}$ is p.s.d.. □

Theorem. *If $K\mathbf{C} - \sigma^2\mathbf{I}$ is p.s.d., then $\mathbf{Q} - \sigma^2\mathbf{I}$ is p.s.d. based on the **Lemma**.*

Proof.

$$
\begin{aligned}
& K\mathbf{C} \succeq \sigma^2\mathbf{I} \\
& \Rightarrow K\mathbf{C}\frac{1}{T}\mathbf{XX}^\top \succeq \sigma^2\frac{1}{T}\mathbf{XX}^\top \quad (\textbf{Lemma}) \\
& \Rightarrow K\mathbf{C}\frac{1}{T}\mathbf{XX}^\top \succeq \sigma^2(K\mathbf{C} + \sigma^2\mathbf{I}) \quad (\text{This is true because of eq. 48.}) \\
& \Rightarrow K\mathbf{C}\frac{1}{T}\mathbf{XX}^\top \succeq \sigma^2 K\mathbf{C} + \sigma^4\mathbf{I} \\
& \Rightarrow \frac{1}{K}\mathbf{C}^{-1}\frac{1}{T}\mathbf{XX}^\top\frac{1}{K}\mathbf{C}^{-1} \succeq \sigma^2\frac{1}{K}\mathbf{C}^{-1}\frac{1}{K}\mathbf{C}^{-1} + \sigma^4\frac{1}{K}\mathbf{C}^{-1}\frac{1}{K}\mathbf{C}^{-1}\frac{1}{K}\mathbf{C}^{-1} \quad (\textbf{Lemma}) \\
& \Rightarrow \mathbf{P}^\top\frac{1}{T}\mathbf{XX}^\top\mathbf{P} \succeq \sigma^2\mathbf{P}^\top\mathbf{P} + \sigma^4\mathbf{P}^\top\mathbf{PP}^\top\mathbf{P} \\
& \Rightarrow \mathbf{P}^\top(\frac{1}{T}\mathbf{XX}^\top + \frac{1}{4}K\mathbf{C})\mathbf{P} \succeq \frac{1}{4}\mathbf{I} + \sigma^2\mathbf{P}^\top\mathbf{P} + \sigma^4\mathbf{P}^\top\mathbf{PP}^\top\mathbf{P} \\
& \Rightarrow \mathbf{US}^4\mathbf{U}^\top \succeq \frac{1}{4}\mathbf{I} + \sigma^2\mathbf{P}^\top\mathbf{P} + \sigma^4\mathbf{P}^\top\mathbf{PP}^\top\mathbf{P} \\
& \Rightarrow \mathbf{S}^4 \succeq (\frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU})(\frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU}) \\
& \Rightarrow (\mathbf{S}^2 - (\frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU}))\mathbf{S}^2 + (\frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU})(\mathbf{S}^2 - (\frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU})) \succeq \mathbf{0}.
\end{aligned}
$$

Denoting $\mathbf{A} = \frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU}$ and $\mathbf{B} = \mathbf{S}^2 - (\frac{1}{2}\mathbf{I} + \sigma^2\mathbf{U}^\top\mathbf{P}^\top\mathbf{PU})$, we can simplify the above inequality as

$$\mathbf{BS}^2 + \mathbf{AB} \succeq \mathbf{0}. \tag{57}$$

We now prove that if inequality 57 is true, then $\mathbf{B} \succeq \mathbf{0}$, using proof by contradiction:
If $\mathbf{B} \not\succeq \mathbf{0}$, then there exists a eigenvector $\mathbf{x}$ s.t. $\mathbf{Bx} = \lambda\mathbf{x}, \lambda < 0$. Then $\mathbf{x}^\top\mathbf{BS}^2\mathbf{x} = \lambda\mathbf{x}^\top\mathbf{S}^2\mathbf{x} \leq 0$ due to the p.s.d. of $\mathbf{S}^2$. Similarly, for the same eigenvector $\mathbf{x}$, we could arrive at the same conclusion for $\mathbf{AB}$,

i.e., $\mathbf{x}^\top \mathbf{A}\mathbf{B}\mathbf{x} = \lambda \mathbf{x}^\top \mathbf{A}\mathbf{x} \leq 0$. This implies that $\mathbf{x}^\top(\mathbf{B}\mathbf{S}^2 + \mathbf{A}\mathbf{B})\mathbf{x} \leq 0$ which is contradict to inequality 57. Therefore $\mathbf{B} \succeq \mathbf{0}$. Now we could continue the deduction as follows,

$$
\begin{aligned}
&\mathbf{B} \succeq \mathbf{0} \\
&\Rightarrow \mathbf{S}^2 \succeq \frac{1}{2}\mathbf{I} + \sigma^2 \mathbf{U}^\top \mathbf{P}^\top \mathbf{P}\mathbf{U} \\
&\Rightarrow \mathbf{U}\mathbf{S}^2\mathbf{U}^\top \succeq \frac{1}{2}\mathbf{I} + \sigma^2 \mathbf{P}^\top \mathbf{P} \\
&\Rightarrow \mathbf{P}^{-\top}\mathbf{U}\mathbf{S}^2\mathbf{U}^\top\mathbf{P}^{-1} \succeq \frac{1}{2}K\mathbf{C} + \sigma^2\mathbf{I} \\
&\Rightarrow \mathbf{Q} - \sigma^2\mathbf{I} \succeq \mathbf{0}.
\end{aligned}
$$

□

We can easily ensure that $K\mathbf{C} - \sigma^2\mathbf{I}$ is p.s.d., which guarantees that $\mathbf{Q} - \sigma^2\mathbf{I}$ is p.s.d. according to the **Theorem**. Therefore $\mathbf{L} = \mathbf{W}_{:,:K}\mathbf{D}^{\frac{1}{2}}_{:K,:K}$ is valid.

Code Availability

The code is available at https://github.com/waq1129/brainkernel.

Data Availability

The HCP datasets are publicly available at https://www.humanconnectome.org/. The visual cortex dataset is publicly available at https://nilearn.github.io/modules/generated/nilearn.datasets.fetch_haxby.html. The BOLD5000 dataset is publicly available at https://bold5000.github.io/. The Sherlock movie dataset is publicly available at https://dataspace.princeton.edu/handle/88435/dsp01nz8062179.

Acknowledgments

This work was supported by the McKnight Foundation (JP), NSF CAREER Award IIS-1150186 (JP), the Simons Collaboration on the Global Brain (SCGB AWD1004351) (JP) and a J. Insley Blair Pyne Fund Award (to JP, BE, KN).

References

1. Schäfer J, Strimmer K, et al. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical applications in genetics and molecular biology. 2005;4(1):32.

2. Bickel PJ, Levina E. Regularized estimation of large covariance matrices. The Annals of Statistics. 2008; p. 199–227.

3. Hsieh CJ, Sustik MA, Dhillon IS, Ravikumar PK, Poldrack R. BIG & QUIC: Sparse inverse covariance estimation for a million variables. In: NIPS; 2013. p. 3165–3173.

4. Treister E, Turek JS. A block-coordinate descent approach for large-scale sparse inverse covariance estimation. In: NIPS; 2014. p. 927–935.

5. Varoquaux G, Gramfort A, Poline JB, Thirion B. Brain covariance selection: better individual functional connectivity models using population prior. In: NIPS; 2010. p. 2334–2342.

6. Van Essen DC, Smith SM, Barch DM, Behrens TE, Yacoub E, Ugurbil K, et al. The WU-Minn human connectome project: an overview. Neuroimage. 2013;80:62–79.

7. Brett M, Johnsrude IS, Owen AM. The problem of functional localization in the human brain. Nature reviews neuroscience. 2002;3(3):243.

8. Stein ML. Interpolation of spatial data: some theory for kriging. Springer Science & Business Media; 2012.

9. Rasmussen CE. Gaussian processes in machine learning. In: Advanced lectures on machine learning. Springer; 2004. p. 63–71.

10. Lawrence ND. Gaussian process latent variable models for visualisation of high dimensional data. In: Advances in neural information processing systems; 2004. p. 329–336.

11. Damianou A, Lawrence N. Deep gaussian processes. In: Artificial Intelligence and Statistics; 2013. p. 207–215.

12. Kitterle FL, Kaye RS. Hemispheric symmetry in contrast and orientation sensitivity. Attention, Perception, & Psychophysics. 1985;37(5):391–396.

13. Di Lollo V. Hemispheric symmetry in duration of visible persistence. Perception & Psychophysics. 1981;29(1):21–25.

14. Westcott M. Hemispheric symmetry of the EEG during the Transcendental Meditation technique. Department of Psychology, University of Durham, Durham, England. 1973;.

15. Fan J, Fan Y, Lv J. High dimensional covariance matrix estimation using a factor model. Journal of Econometrics. 2008;147(1):186–197.

16. Fan J, Liao Y, Liu H. An overview of the estimation of large covariance and precision matrices. The Econometrics Journal. 2016;19(1):C1–C32.

17. Lawrence ND. Learning for Larger Datasets with the Gaussian Process Latent Variable Model. In: AISTATS. vol. 11; 2007. p. 243–250.

18. Damianou AC, Titsias MK, Lawrence ND. Variational inference for uncertainty on the inputs of gaussian process models. arXiv preprint arXiv:14092287. 2014;.

19. Hensman J, Fusi N, Lawrence ND. Gaussian processes for big data. arXiv preprint arXiv:13096835. 2013;.

20. Wu TT, Lange K. Coordinate descent algorithms for lasso penalized regression. The Annals of Applied Statistics. 2008; p. 224–244.

21. Wang H. Coordinate descent algorithm for covariance graphical lasso. Statistics and Computing. 2014;24(4):521–529.

22. Tseng P, Yun S. A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming. 2009;117(1):387–423.

23. Fox MD, Snyder AZ, Zacks JM, Raichle ME. Coherent spontaneous activity accounts for trial-to-trial variability in human evoked brain responses. Nature neuroscience. 2006;9(1):23–25.

24. Smith SM, Fox PT, Miller KL, Glahn DC, Fox PM, Mackay CE, et al. Correspondence of the brain's functional architecture during activation and rest. Proceedings of the National Academy of Sciences. 2009;106(31):13040–13045.

25. Cole MW, Ito T, Bassett DS, Schultz DH. Activity flow over resting-state networks shapes cognitive task activations. Nature Neuroscience. 2016;.

26. WU-Minn H. 1200 subjects data release reference manual; 2017.

27. Delgado MR, Nystrom LE, Fissell C, Noll D, Fiez JA. Tracking the hemodynamic responses to reward and punishment in the striatum. Journal of neurophysiology. 2000;84(6):3072–3077.

28. Binder JR, Gross WL, Allendorfer JB, Bonilha L, Chapin J, Edwards JC, et al. Mapping anterior temporal lobe language areas with fMRI: a multicenter normative study. Neuroimage. 2011;54(2):1465–1475.

29. Buckner RL, Krienen FM, Castellanos A, Diaz JC, Yeo BT. The organization of the human cerebellum estimated by intrinsic functional connectivity. Journal of neurophysiology. 2011;.

30. Hariri AR, Brown SM, Williamson DE, Flory JD, De Wit H, Manuck SB. Preference for immediate over delayed rewards is associated with magnitude of ventral striatal activity. Journal of Neuroscience. 2006;26(51):13213–13217.

31. Smith R, Keramatian K, Christoff K. Localizing the rostrolateral prefrontal cortex at the individual level. Neuroimage. 2007;36(4):1387–1396.

32. Barch DM, Burgess GC, Harms MP, Petersen SE, Schlaggar BL, Corbetta M, et al. Function in the human connectome: task-fMRI and individual differences in behavior. Neuroimage. 2013;80:169–189.

33. Haxby JV, Gobbini MI, Furey ML, Ishai A, Schouten JL, Pietrini P. Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science. 2001;293(5539):2425–2430.

34. Esteban O, Blair R, Markiewicz CJ, Berleant SL, Moodie C, Ma F, et al.. poldracklab/fmriprep: 1.0.0-rc5; 2017. Available from: https://doi.org/10.5281/zenodo.996169.

35. Abraham A, Pedregosa F, Eickenberg M, Gervais P, Mueller A, Kossaifi J, et al. Machine learning for neuroimaging with scikit-learn. Frontiers in neuroinformatics. 2014;8:14.

36. Chen J, Leong YC, Honey CJ, Yong CH, Norman KA, Hasson U. Shared memories reveal shared structure in neural activity across individuals. Nature neuroscience. 2017;20(1):115.

37. Simony E, Honey CJ, Chen J, Lositsky O, Yeshurun Y, Wiesel A, et al. Dynamic reconfiguration of the default mode network during narrative comprehension. Nature communications. 2016;7:12141.

38. Gershman SJ, Blei DM, Pereira F, Norman KA. A topographic latent source model for fMRI data. NeuroImage. 2011;57(1):89–100.

39. Manning JR, Ranganath R, Norman KA, Blei DM. Topographic factor analysis: a bayesian model for inferring brain networks from neural data. PloS one. 2014;9(5):e94914.

40. Wu A, Pashkovski S, Datta SR, Pillow JW. Learning a latent manifold of odor representations from neural responses in piriform cortex. In: Advances in Neural Information Processing Systems; 2018. p. 5378–5388.

41. Chang N, Pyles JA, Marcus A, Gupta A, Tarr MJ, Aminoff EM. BOLD5000, a public fMRI dataset while viewing 5000 visual images. Scientific data. 2019;6(1):49.

42. Grosenick L, Klingenberg B, Katovich K, Knutson B, Taylor JE. Interpretable whole-brain prediction analysis with GraphNet. NeuroImage. 2013;72:304–321.

43. Rao N, Cox C, Nowak R, Rogers TT. Sparse overlapping sets lasso for multitask learning and its application to fmri analysis. Advances in neural information processing systems. 2013;26:2202–2210.

44. Wu A, Park M, Koyejo OO, Pillow JW. Sparse Bayesian structure learning with dependent relevance determination priors. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, editors. Advances in Neural Information Processing Systems 27. Curran Associates, Inc.; 2014. p. 1628–1636. Available from: http://papers.nips.cc/paper/5233-sparse-bayesian-structure-learning-with-dependent-relevance-determination-priors.pdf.

45. Wu A, Koyejo O, Pillow J. Dependent relevance determination for smooth and structured sparse regression. Journal of Machine Learning Research. 2019;20(89):1–43.

46. Xu H, Lorbert A, Ramadge PJ, Guntupalli JS, Haxby JV. Regularized hyperalignment of multi-set fMRI data. In: 2012 IEEE Statistical Signal Processing Workshop (SSP). IEEE; 2012. p. 229–232.

47. Haxby JV, Guntupalli JS, Nastase SA, Feilong M. Hyperalignment: Modeling shared information encoded in idiosyncratic cortical topographies. ELife. 2020;9:e56601.

48. Lawrence ND. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In: Nips. vol. 2; 2003. p. 5.

49. Nocedal J. Updating quasi-Newton matrices with limited storage. Mathematics of computation. 1980;35(151):773–782.

50. Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation. 2003;15(6):1373–1396.

51. John J, Draper N. An alternative family of transformations. Applied Statistics. 1980; p. 190–197.

52. Stein ML. Interpolation of Spatial Data: Some Theory for Kriging. Springer; 1999.

53. Lázaro-Gredilla M, Quiñonero-Candela J, Rasmussen CE, Figueiras-Vidal AR. Sparse spectrum Gaussian process regression. The Journal of Machine Learning Research. 2010;11:1865–1881.

54. Wu A, Aoi MC, Pillow JW. Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature. arXiv preprint arXiv:170400060. 2017;.

55. Tipping ME, Bishop CM. Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 1999;61(3):611–622.

56. Vodrahalli K, Chen PH, Liang Y, Baldassano C, Chen J, Yong E, et al. Mapping between fMRI responses to movies and their natural language annotations. NeuroImage. 2018;180:223–231.

Appendices

A Relationship between brain kernel and deep GPs

The BK model can be viewed as a sequence of two nonlinear functions, each with a Gaussian process prior.

In the main text, we explained that the latent embedding $\mathbf{z}$ is obtained by applying a function f over the input 3D coordinate $\mathbf{x}$, where $f \sim \mathcal{GP}(m, k)$. This is the first layer GP. The second component of the brain kernel model is a probability distribution over neural activity as a function of location in embedding space. Our modeling assumption is that neural activity changes smoothly as a function of location in embedding space, or equivalently, that correlations in neural activity fall off smoothly with distance in latent space. This motivates the use of a second GP to model each vector of neural activity. Thus:

$$g \sim \mathcal{GP}(0, \kappa_g), \tag{58}$$

$$\mathbf{v} = g(\mathbf{z}), \tag{59}$$

where $\kappa_g(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-||\mathbf{z}_i - \mathbf{z}_j||^2/2\right)$ is a standard RBF covariance function (marginal variance and length scale = 1), and $\mathbf{v} \in \mathbb{R}^n$ denotes a vector of n voxels' neural activity at latent locations $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$. Given that latent voxel locations are themselves a nonlinear function of the true voxel locations, this allows to write, equivalently: $\mathbf{v} = g(f(\mathbf{x}_1, \ldots, \mathbf{x}_n))$. Therefore, we could eventually obtain the neural activity $\mathbf{v}$ by applying a two-layer GPs to the input 3D coordinates $\mathbf{X}$.

B More decoding results

B.1 HCP database

In this section, we present the classification accuracy performance for the motor task, the language processing task, the emotion processing task, the relational processing task, and the social cognition task.

Table 1: The basic information of the fMRI scans [26]

Task	Runs	Frames per run	Run Duration (min:sec)
Working Memory	2	405	5:01
Gambling	2	253	3:12
Motor	2	284	3:34
Language Processing	2	316	3:57
Social Cognition	2	274	3:27
Relational Processing	2	232	2:56
Emotion Processing	2	176	2:16

Similar to the working memory and gambling tasks, each task fMRI was from the same HCP project as the resting-state fMRI. They shared the same coordinate system and preprocessing pipeline. The task fMRI was aligned with the MNI template with the same 59,412 voxels as the resting-state fMRI data. Instead of analyzing the whole-brain data for brain decoding, we worked with functional ROIs whose averaged activity levels were above a certain threshold for all tasks.

We formulated each task as a binary classification problem. We used the same Bayesian linear regression classifiers as described in the working memory and gambling sections. We trained the classifier with three priors on one run and calculated the accuracy performance on the second run, then switched the training and test runs. This procedure was repeated ten times. The ± 1 labels were treated as continuous target values during training, and the test accuracy was evaluated by taking the sign of the prediction. We computed the averaged accuracies across two runs and 10 repetitions for the three priors for all subjects in each dataset. Results are presented in Fig. 14 to Fig. 18.

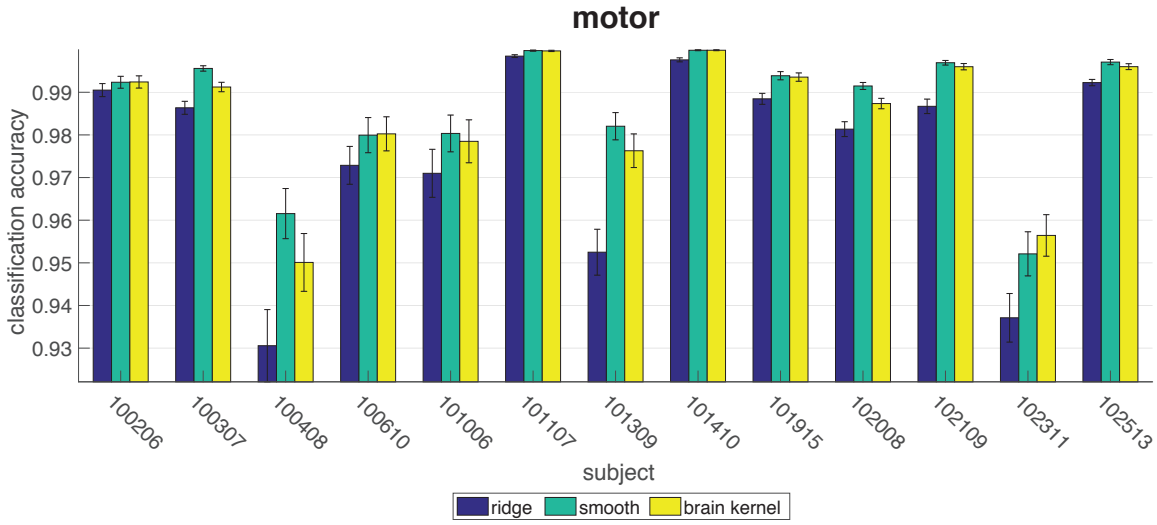

Fig 14. Accuracy performance on the motor task.

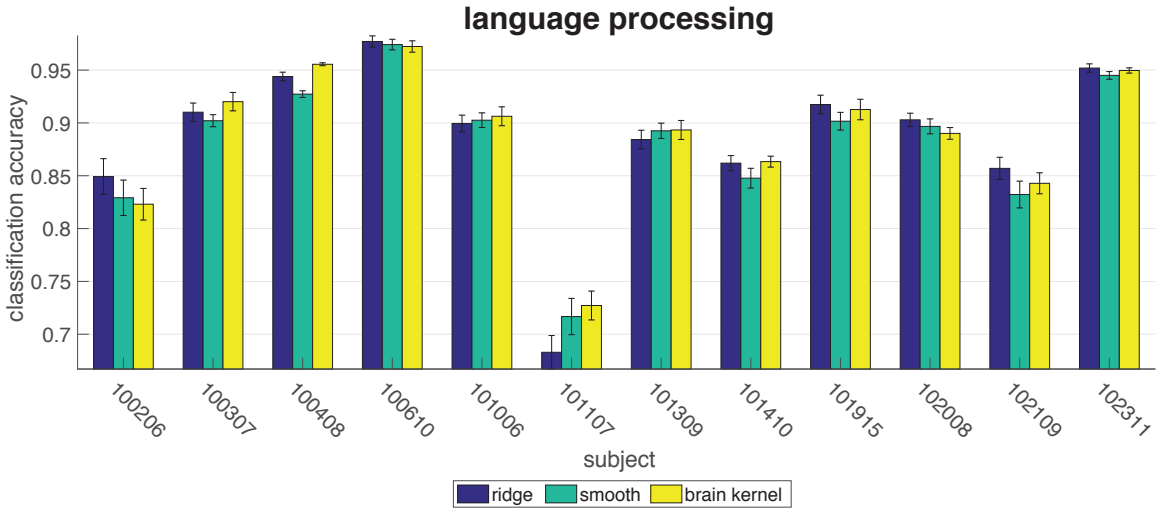

Fig 15. Accuracy performance on the language processing task.

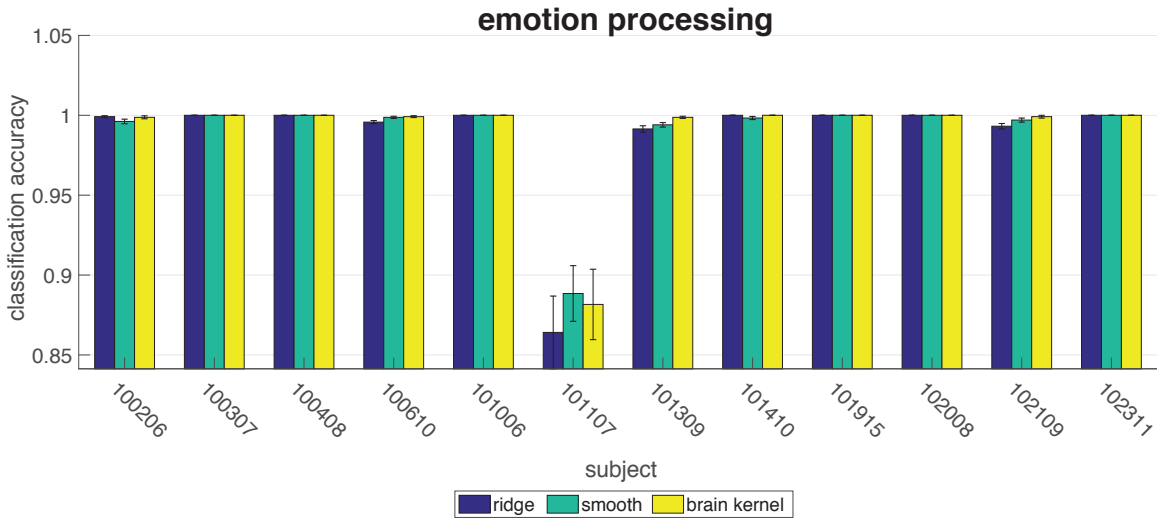

Fig 16. Accuracy performance on the emotion processing task.

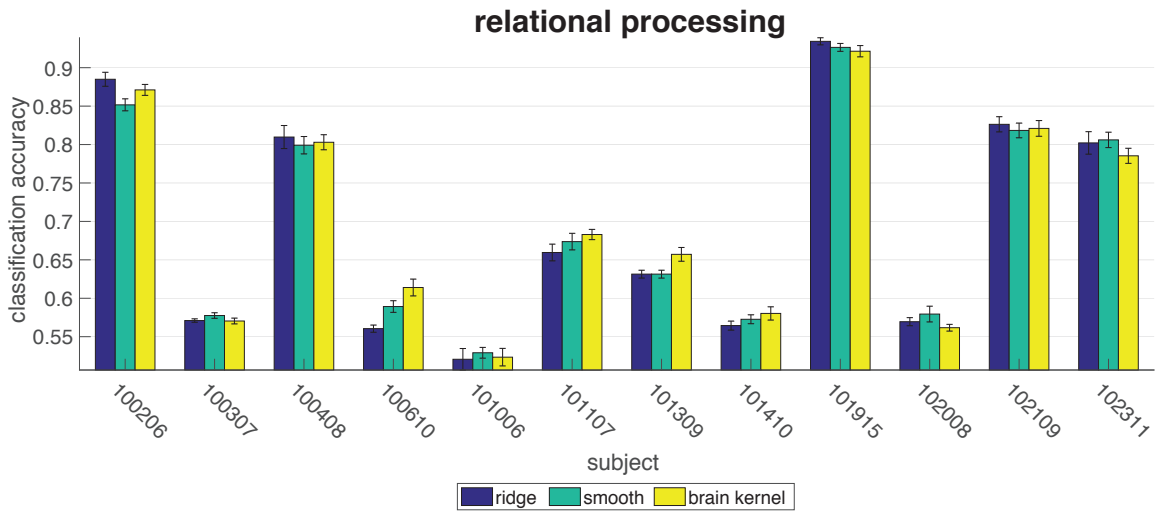

Fig 17. Accuracy performance on the relational processing task.

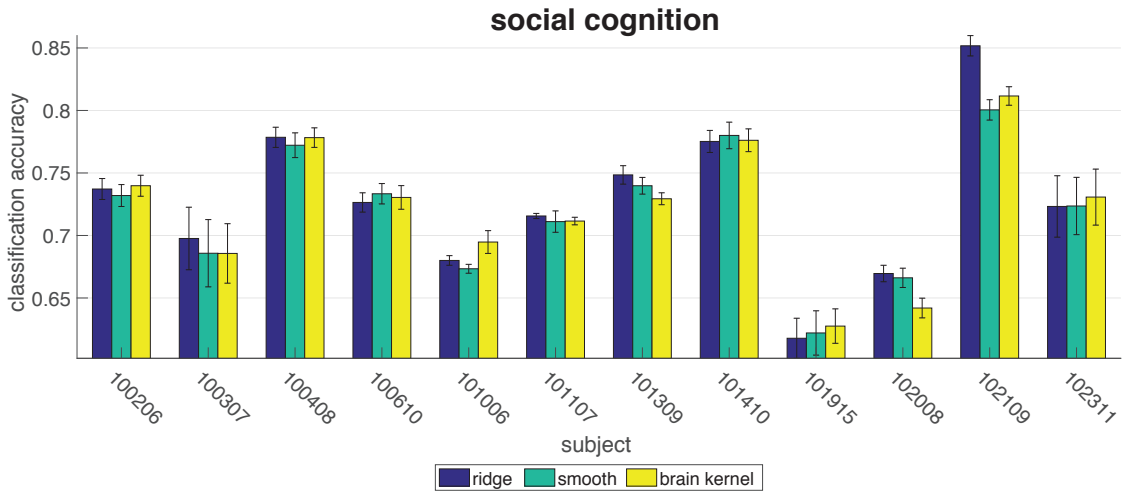

Fig 18. Accuracy performance on the social cognition task.

B.2 Sherlock movie fMRI dataset

We examined the Sherlock fMRI dataset, in which participants were scanned while they watched the British television program “Sherlock” for 50 min [36]. The fMRI data comprised 1,973 TRs (Repetition Time), where each TR was 1.5 s of the movie. Before performing any analysis, the fMRI data were preprocessed and aligned to MNI space using the techniques described in prior work [36]. We examined the brain data averaged across all subjects to smooth out individual variability. We identified 11 ROIs previously implicated in processing naturalistic stimuli, comprising the default mode network (DMN-A, DMN-B), the ventral and dorsal language areas, and the primary auditory and visual cortices [37].

For each ROI, we performed a standard decoding task to relate the fMRI signal to the representation of the semantic content of each movie frame [56]. We trained the decoding model using the first half of the movie and tested on the second half of the movie. The decoding task was called “scene classification". We divided up the second half of the movie into 25 uniformly-sized chunks, and used the correlation to match predicted semantic content with the ground truth for each chunk, and reported the percentage of the chunks that the match is perfect. Since there are 25 chunks, random chance performance at this task is 4%. Fig. 19 presents the classification accuracy performance. We observed that the smoothing prior and the brain kernel prior both converged to the ridge prior after hyperparameter estimation (3-fold cross validation during training), i.e., the estimated length scale was very small and all three priors had the same performance. Therefore, smoothness did not help the scene classification task.

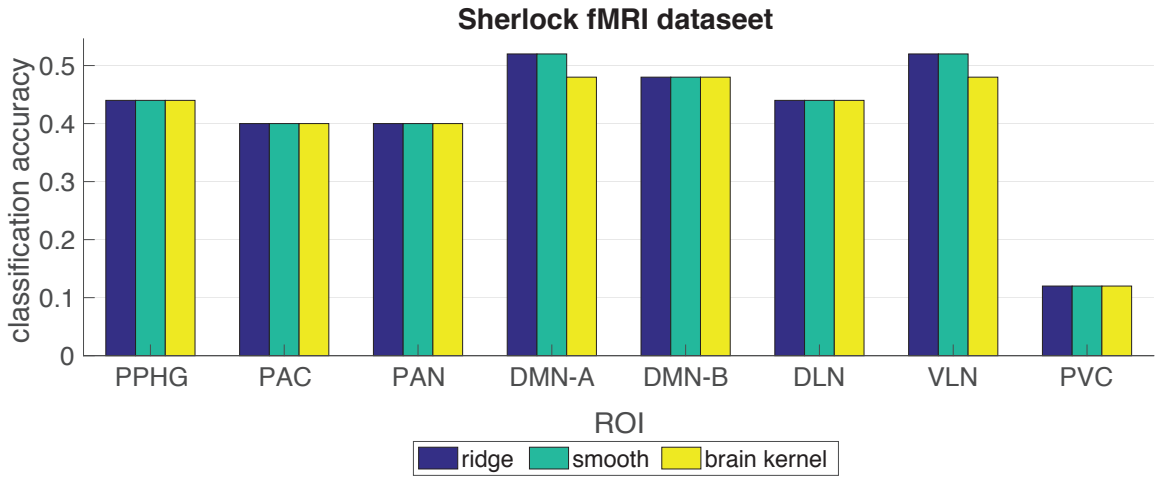

Fig 19. Accuracy performance on the Sherlock movie fMRI dataset. The ROI information is the same as Fig. 12

B.3 BOLD5000 dataset

BOLD5000 is a public fMRI dataset in which subjects viewed 5000 visual images [41]. We used the fMRI datasets collected for 4 subjects with 15 sessions per subject and 333 TRs per session. We extracted predefined functional ROIs [41] from the whole-brain data with 3804 voxels. The decoding task was binary classification. The binary labels were living objects versus nonliving objects. For each subject, we used 14 sessions for training and 1 session for test. Thus the final accuracy was an average over 15 runs by treating each session as the test set once (Fig. 20). We observed that the brain kernel did not reliably outperform the ridge prior and the smoothing prior estimates across subjects.

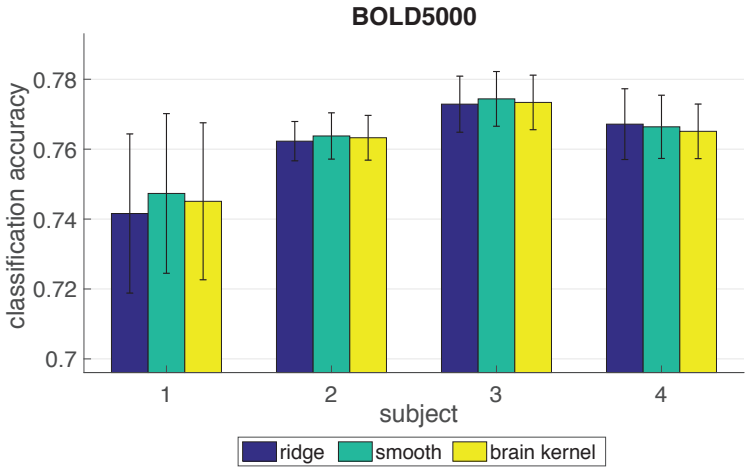

Fig 20. Accuracy performance on the BOLD5000 dataset.

C More factor modeling results

C.1 Visual recognition task

We examined the visual recognition task fMRI dataset with the same Bayesian FA model. We implemented the same “co-smoothing" evaluation as described in the main section. The only difference is, instead of splitting the runs into halves for training and test within each subject, we treated one subject as the test set and the other five subjects as the training set. This is consistent with the inter-subject classification in the decoding task. We report the normalized test R^2 for the three priors when treating each subject as the test set in Fig. 21. For most subjects, the brain kernel didn’t show a dominating performance over both the ridge covariance and the smooth RBF kernel.

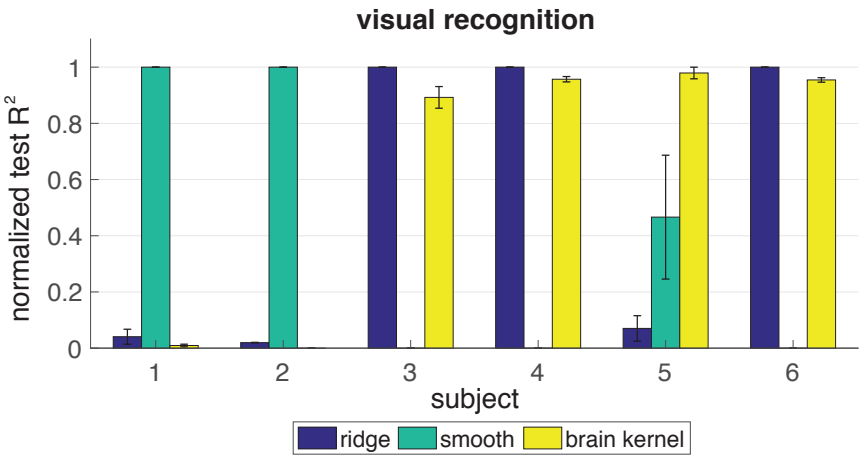

Fig 21. Normalized test R^2 performance on the visual recognition task.

C.2 BOLD5000 dataset

The BOLD5000 dataset consists of 4 subjects with 15 sessions per subject and 333 TRs per session. We extracted predefined functional ROIs [41] from the whole-brain data with 3804 voxels. When evaluating the performance using “co-smoothing", we split the 15 sessions into halves within each subject, one for training and one for inference and test as described in the main section. We report the normalized test R^2 for the three priors for each subject in Fig. 22. For most subjects, the brain kernel didn’t show a dominating performance over both the ridge covariance and the smooth RBF kernel. The smoothing prior had better performance than the brain kernel which was better than the ridge prior. This result is consistent with the decoding performance in Fig. 20.

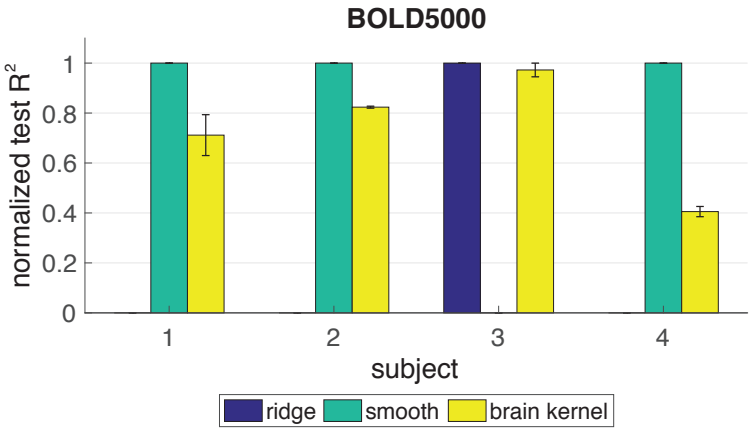

Fig 22. Normalized test R^2 performance on the BOLD5000 dataset.