Simulated data structures

This section outlines in a unified fashion how we will simulate data without the specifications of each simulation context.

We will simulated $n = n_A + n_B$ images, where n_A is the number of images from group A and n_B is the number of images from group B. Let y_i be the ith image structured as an $s \times 1$ vector corresponding with the s pixels/voxels of the image. Let z_i be 1 if image i is in group A and 0 if image i is in group B. We specify two covariance matrices $\Sigma_g = Q_g \Lambda^g Q_g^T$ for $g = A, B$ with matrix of eigenvectors Q_g and eigenvalues $\{\Lambda_{ii}^g\}_{i=1}^s$, and a group effect $\beta \in \mathbb{R}$. Then we simulate images independently from the model

$$y_i = \beta z_i + \epsilon_i \qquad\qquad \epsilon_i \sim \mathrm{MVN}(0, \Sigma_{g_i}),$$

equivalently

$$y_i \sim \mathrm{MVN}(\beta z_i, \Sigma).$$

A particular simulation context will be defined through specification of the following:

(a) An 'image space.' Unless otherwise stated, we'll be talking about images of the same dimension as the handwritten digit data. Thus, $s = 256$ pixels arranged in a 16×16 lattice.

(b) Desired number of observations per group, n_A and n_B

(c) Desired covariances Σ_g, either by providing a model (e.g. exponential) or by providing eigenvectors Q_g and eigenvalues Λ_g. Unless otherwise specified, we will assume $\Sigma_A = \Sigma_B$.

(d) The group effect β. We might choose β in a couple of ways, such as by a function of pixel location (e.g. 1 for all pixels in the top half of the image and -1 for all pixels in the bottom half), as a linear combination of ESF/GSP eigenvectors, or empirically (using the empirical covariance matrix, maybe estimated with regularization).

Models

(a) Predicting images y_i from group z_i/Inferring group effect

   (1) VBM

   (2) spVBM

      i. Only positive eigenvalue eigenvectors?

      ii. Only a subset of eigenvectors?

      iii. Knots or exact computation?

(b) Predicting group z_i from image $\tilde{y}_i$ using sparse logistic regression where $\tilde{y}_i$ is a transformation of the image y_i

   (1) Voxels as covariates $\tilde{y}_i = y_i$

   (2) functional PCs as covariates

(3) Frequency intensities as covariates (images after application of ESF or GSP transformation, e.g. $\tilde{y}_i = Q_{ESF} y_i$)

i. Only positive eigenvalue eigenvectors?

ii. Only a subset of eigenvectors?

iii. Knots or exact computation?

(4) One other method?

(c) Inferring network

(a) Do we have a specific methodology picked out to look at?

Specific simulations of interest

The intention for this section is that the first simulation is ready to get started on, while remaining simulations are currently being designed.

1. First simulation

Data Let $n_A = n_B = 1000$ where β is 1 for pixels in the center 8×8 pixel square and 0 elsewhere, $\Sigma = \Sigma_A = \Sigma_B$ is an exponential correlation matrix with rate 1.

PredictImage Fit VBM and spVBM models predicting images y_i from group z_i using all eigenvectors from 16 knots. Report the same performance metrics as in Sarah's paper. Use an exponential network.

PredictGroup Train sparse logistic regression models on 800 observations (400 per group) predicting group z_i from image covariates $\tilde{y}_i$ for the following transformations:

i. Voxels as covariates (no transformation)

ii. all exact ESF frequencies as covariates (transformation from exponential network)

iii. all exact GSP frequencies as covariates (transformation from exponential network, unnormalized Laplacian)

iv. functional PCs as covariates (talk with Yue for details)

Report test AUC, sensitivity, and specificity using remaining 200 observations. Report also which covariates were selected in the voxel, ESF frequency, and GSP frequency models.

2. Effects:

(a) Sparse-in-voxel (like a circle effect) vs sparse-in-frequency effects.

i. For sparse-in-voxel effects, ESF/GSP in PredictImage should outperform voxel-based while ESF/GSP in PredictGroup should underperform voxel-based

    ii. For sparse-in-frequency effects, voxel-based should underperform. If effects are on a small scale relative to resolution (negative eigenvalues), frequency approaches should vastly outperform everything, hopefully even under misspecification. If effects are on a large scale relative to resolution, we expect fPCA to still do fine while voxel-based analyses will still suffer for moderate strength effects.
    iii. Sharp boundaries vs soft boundaries; thinking about bias, eigenvector approximations, and which eigenvectors are incorporated into models
  (b) For sparse-in-frequency effects, negative eigenvalue vs positive eigenvalue effects

3. What if networks differ by group or aren't quite spatial? In neuroimaging applications, we might reasonably expect networks to differ between healthy and disease cohorts. In geostatistical applications, our interest is more in accounting substantially for correlation than perfectly representing it. This begs a simulation in which both groups essentially have spatial correlation structure, but one additionally has a couple of shorts/wormholes in that structure. We would want the true effect β to interact with that short in a meaningful way, and to see how ESF/GSP methods perform as they likely can't/won't account for the short. How many shorts and how strong until we've got a problem?

4. Inferring network simulations: Yue and William discussed a block spatial structure for these simulations representing our understanding of ROIs in the brain while also leveraging intuition available in spatial contexts. Brains tend to be well represented by ROI within which there is strong associations and among which there are weak associations. Thus, we will choose of a block network structure as a direct product of a strong spatial network (within ROI) and a weak complete network (among ROI). This structure may benefit from varying the number of voxels in a 'ROI.'

Alternative Simulations

Simulation 1

1. **Generate Covariate Matrix** X: Assume an image size of 16×16 pixels. X has a shape of 1000×256. Simulate each X_i from a multivariate normal distribution $X_i \sim \mathcal{N}(0, W)$, where $W(k, j) = \exp(-\text{dist}(k, j))$.

2. **Define Coefficient Vector** β: Set β as a vector of length 256, where $\beta_i = 0$ except for the center 8×8 region, where $\beta_i = 1$.

3. **Generate Response Variable** y: Generate $y \sim \text{Binomial}(1, p)$, where $p = 1/(1 + \exp(-\eta))$, $\eta = X\beta$.

4. **Evaluation**: Split the dataset into training (80%) and test (20%) sets. Fit a logistic LASSO model with cross-validation on the training set. Evaluate model performance on the test set using AUC. Visualize the identified pixels with the true non-zero coefficients in β to assess the model's accuracy in detecting the decisive region.

Simulation 2

Simulation 2 differs from Simulation 1 by transforming the covariate matrix into the frequency space using eigenvectors and assuming a sparse coefficient vector in this transformed space.

1. **Eigen Decomposition**: Assume the same correlation structure W. Perform eigen decomposition on W to obtain eigenvectors V, which is of shape 256×256. Use all eigenvectors for transformation.

2. **Generate Covariate Matrix** X: Generate X with shape 1000×256, where each X_i follows a multivariate normal distribution with a diagonal covariance matrix. Transform X using all eigenvectors to get $X_{\text{trans}} = XV$.

3. **Assume Sparse Coefficient Vector** b **in the Frequency Space**: Set b as a sparse vector of length 256, where most entries are 0, and a few are non-zero. Compute β in the pixel space: $\beta = V^T b$.

4. **Evaluation**: Generate y using X and β as in Simulation 1. Evaluate the model performance using AUC, following the same procedure as Simulation 1. Visualize the identified coefficients in b to assess the model's accuracy in detecting the decisive components in the frequency space.