Task: Write a function for simulating data. Make E one of the arguments and output data in pixel space whether or not they're generated there. Write another function for fitting models in either the frequency space or the pixel space where analysis data are always input in pixel space. Make E one of the arguments to facilitate fitting in frequency space.

What does it mean to use different values of E to generate the data and analyze the data? Why would we want that?

Spaces

Let X be an $n \times p$ matrix of n images with p pixels. Each row is comprised of a vectorized image. We refer to these images as being in pixel space.

Let E be a $p \times p$ orthogonal matrix. We refer to E as a transformation matrix as we will use it to transform data between pixel space and a frequency space. Typically, E is obtained through eigendecomposition of a symmetric matrix, in which case columns of E are eigenvectors, also referred to as frequencies.

Given transformation E and data X comprised of images in pixel space, we transform images to frequency space (X_{freq}) and back to pixel space as follows:

$$X_{freq} = XE; \qquad X = E^T E X = E^T X_{freq} \ .$$

Model

Suppose images X come in two groups, group 0 and 1. We might model the groups as a function of the images using a logistic model. Let y_i indicate the group to which image i belongs, let E denote a transformation, and let x_i and $x_{freq,i}$ be the ith image in pixel space and in frequency space (the ith rows of X and $X_{freq} = XE$). Let β_0 be a constant, β be a p-vector, and $\beta_{freq} = E^T\beta$. Then we model

$$y_i = \text{Bernoulli}(p_i); \qquad p_i = \frac{1}{1+\exp(-\beta_0 - X\beta)} = \frac{1}{1+\exp(-\beta_0 - X_{freq}\beta_{freq})} \ .$$

[I know we don't actually use an intercept β_0 right now, but we would specify a non-zero intercept in a future simulation if we wanted imbalanced groups.]

If we fit the model to data (X, y), we say we fit the model in pixel space. If we fit the model to data (X_{freq}, y), we say we fit the model in frequency space.

We fit models using LASSO type penalties in either pixel space or frequency space. When the model is fit in pixel space, the LASSO penalty is applied to coefficients β and the model is sparse in pixels. When the model is fit in frequency space, the LASSO penalty is applied to coefficients β_{freq} and the model is sparse in frequencies.

Simulated Data

Image data X are simulated with $n = 1000$ independent rows and $p = 256$ pixels, each row having multivariate normal distribution $\mathcal{N}(0, \Sigma)$ for a specified Σ. Group data y are then simulated from the model by specifying the intercept $\beta_0 = 0$ and specifying coefficients in either pixel space (β) or frequency space ($\beta_{freq}^{data}, E_{data}$). Here, we understand the 256 pixels to be arranged in a 16×16 grid. For each simulation context, we simulate 500 datasets.

Simulation 1

We set Σ to be an exponential correlation matrix with rate 1, meaning $\Sigma_{ij} = \exp(-d_{ij})$ where d_{ij} denotes the Euclidean distance between centers of the ith and jth pixels in the 16×16 grid. We specify coefficients β in pixel space as taking constant non-zero value over the central 8×8 pixel region of the image and 0 elsewhere. We select the constant non-zero value such that the resulting distribution of probabilities p_i is roughly uniform.

Simulation 2

We set $\Sigma = E_{data} D E_{data}^T$ where D is a diagonal matrix with ith entry $D_{ii} = p + 1 - i$ and transformation E_{data} obtained from eigendecomposition of $M(\Sigma - I)M$, using $\Sigma - I$ as a weighted adjacency matrix representing the 16×16 grid. This is achieved by simulating data $\tilde{X}$ with independent rows having distribution $\mathcal{N}(0, D)$ and setting $X = \tilde{X} E_{data}^T$. [In the case $E_{data} = E_{model}$, see below, we can say $\tilde{X} = X_{freq}$ unambiguously. If $E_{data} \neq E_{model}$, as will be the case for some future simulations, then we fit models to $X_{freq} = X E_{model}$, not $\tilde{X} = X E_{data}$.]

We specify coefficients β_{freq}^{data} in frequency space according to transformation E_{data}. Specifically, for a fixed random subset of 10% of frequencies, we set β_{freq}^{data} as taking constant non-zero value over the subset and 0 elsewhere. We select the constant non-zero value such that the resulting distribution of probabilities p_i is roughly uniform. Note, the random subset of frequencies is fixed over all replicate datasets.

[I think if we want to flatten out the bias, we might try the following: Suppose β_{freq}^{data} is non-zero at the jth frequency: $\beta_{j,freq}^{data} \neq 0$. Let $\beta_{j,freq}^{data} = D_{jj}^{-1}$ instead of a constant value.]

[We might consider taking D to be the eigenvalues of an exponential correlation matrix with rate 1 so that the simulations have the same correlation structure.]

Model fitting

LASSO models are fit in either pixel space or frequency space. For model fitting, frequency space is defined by transformation E_{model} obtained from eigendecomposition of MCM where $C + I$ is an exponential correlation matrix with rate 1 for the 16×16 grid. [Note: This means $E_{data} = E_{model}$. In future simulations, we will not equate E_{model} and E_{data} as in practice we would not know the true frequency space for the image data. We could, for example, pick C to be the unweighted 2-neighbor adjacency matrix.]

When fitting models, a dataset X is split into 80% training data and 20% test data. 10-fold cross-validation is used to select the LASSO regularization parameter λ (from what candidate values?) either producing the minimal cross-validated error (`lambda.min`) or the largest candidate parameter value producing cross-validated error within one standard error of the minimum cross-validated error (`lambda.1se`).

In total, we fit four models to each training dataset where a model fitting is characterized by the space in which it is fit (pixel or frequency) and the selection method for the tuning parameter (`lambda.min` or `lambda.1se`).

Fit models are then applied to the test data. We compute classification accuracy and ROC AUC (area under the curve for the received operating characteristic curve). We also

compute coefficient p-values using the `hdi` package.