Denoised Image Principal Components

Eleanor Wong

1 Introduction

Functional PCA extends principal component analysis to functional data of infinite dimensions. Compared with traditional PCA, this takes into account the order of the dimensions from one time point to next. The benefit over traditional principal component analysis is the flexibility of allowing constraints to get what are desirable properties for the extracted principal component functions. In image processing, images are functions in Sobolev space that can be denoised by penalizing the norm of L operators on the image, e.g. image gradient or Laplacian. For this project, the goal is to experiment with smoothness in the eigenfunctions from a set of images.

2 Functional PCA

Principal component regression is a regression technique that uses PCA to first reduce the dimensionality of the data prior to doing regression using the reduced components. Ramsay and Silverman[1] describes the procedure for functional PCA, and Jiang et al[2] applied 2D functional PCA on images. Here I added H1 and total variation penalties to impose smoothness on the image functional components.

In PCA, the goal is to find an orthonormal basis upon which to project the data onto with minimum reconstruction error. This is equivalent to maximizing the variance of the projected scores on the basis. Let X be the observed data centered at mean 0, with weight vector β_1 such that for $f_{i1} = \sum_j \beta_{j1} x_{ij}$, $i = 1 \dots N$, the variance $\frac{\sum_i f_{i1}^2}{N}$ is maximized, subject to $\sum \beta_{j1}^2 = \|\beta_1\|^2 = 1$ with the constraint that components are orthogonal aka $\sum_j \beta_{jk}\beta_{jm} = 0$. For the generalization to functional PCA, let $f_i = \int \beta_1(s)\, x_i(s)\, ds$ and we want to maximize $\frac{\sum_i f_{i1}^2}{N} = \frac{\sum_i \left(\int \beta_1 x_i\right)^2}{N}$, subject to $||\beta_1||^2 = \int \beta_1(s)^2\, ds = 1$ and the orthogonal constraint that $\int \beta_k \beta_m = 0$. Let $\bar{x} = (x - \mu_x)$ be the mean centered data, and $R = \frac{\sum \bar{x}\bar{x}^T}{N}$ be the sample covariance matrix. Luenberger[3] has shown that $\sup_{\|\beta\|=1} |F(\beta)| = \sup_{\beta \neq 0} \frac{|F(\beta)|}{\|\beta\|}$, the objective function to be maximized with the norm constraint

[1] Silverman, B. W., and J. O. Ramsay. Functional Data Analysis. Springer, 2005.

[2] Jiang, Junhai, et al. "Multiple functional linear model for association analysis of RNA-seq with imaging." Quantitative Biology 3.2 (2015): 90-102.

[3] Luenberger, David G. Optimization by vector space methods. John Wiley & Sons, 1997.

for each component is now $\max_\beta \frac{(\beta^T R\beta)}{\|\beta\|}$. This is the formula for the Rayleigh quotient, aka the eigenvalue of the principal component. Subsequent orthogonal principal components are found by subtracting out the projection to the components from the data, and repeating the process.

3 Regularization

Ramsay and Silverman proposed regularizing the sample variance by division, such that the objective function is $\max_\beta \frac{(\beta^T R\beta)}{\|\beta\|^2+\gamma J(\beta)}$, with $J(\beta)$ being the penalty function on β.

3.1 H1 Penalty

The H1 penalty is $J(\beta) = \gamma \int_\Omega |\nabla\beta|^2$, such that the functions with high amount of changes from one time point to next are penalized. γ is a parameter that controls how much smoothing is desired. In image denoising, this encourages smoothness between pixels. β is a function in L^2, such that it is a Hilbert space and inner products are defined satisfying the requirements of Holder's inequality. It is straighforward to do this optimization. The drawback of this penalty and others of L^p such that $p > 1$ on the gradient is that the norm tends to infinity for functions with large discontinuous jumps. Thus, this penalty does not preserve edges in images.

We can use calculus of variations to maximize $\frac{(\beta^T R\beta)}{\|\beta\|^2+\gamma \int_\Omega |\nabla\beta^2}$. Since this is a functional integrating β and $\nabla\beta$ over the image domain Ω of $n \times m$ number of pixels, instead of going through Gateaux derivatives we can solve directly for the Euler-Lagrange equations. The satisfying conditions are:

$$\begin{aligned} \frac{2R\beta}{\|\beta\|^2+\gamma\|\nabla\beta\|^2} - \frac{\left(\beta^T R\beta\right)(2\beta)}{(\|\beta\|^2+\gamma\|\nabla\beta\|^2)^2} + \frac{\left(\beta^T R\beta\right)\gamma(\Delta\beta)}{(\|\beta\|^2+\gamma\|\nabla\beta\|^2)^2} &= 0 \text{ (on } \Omega) \\ \langle\nabla\beta, \nu\rangle &= 0 \text{ (on } \partial\Omega) \end{aligned}$$

where ν is the outward normal. The first condition is the derivative update to β, when the objective function is maximized the derivative is 0. The solution is by gradient ascent, with $x^{t+1} = x^t + \epsilon\left(\frac{2R\beta}{\|\beta\|^2+\gamma\|\nabla\beta\|^2} - \frac{(\beta^T R\beta)(2\beta)}{(\|\beta\|^2+\gamma\|\nabla\beta\|^2)^2} + \frac{(\beta^T R\beta)\gamma(\Delta\beta)}{(\|\beta\|^2+\gamma\|\nabla\beta\|^2)^2}\right)$ at each step. The boundary condition is maintained when ∇ and $-\Delta$ are properly implemented as adjoints of each other.

3.2 TV Penalty

The total variation penalty is $J(\beta) = \gamma \int_\Omega |\nabla\beta|$, which is tricky to optimize because it is not the dual of a Banach space. However, we can consider instead the space of all bounded variation where total variation is defined as $TV = \int_\Omega |\nabla\beta| = \sup_{w\in C_0^\infty, \|w\|\leq 1} \int_\Omega \beta\nabla\cdot w$ for a dual variable w.

Because the objective function is always positive, the parameters $\hat{\beta}$ that maximizes the log of the objective function also maximizes the original function. Therefore, the function to optimize is

$$\max_\beta = \log\left(\beta^T R\beta\right) - \log\left(\|\beta\|^2 + \gamma\|\nabla\beta\|\right)$$

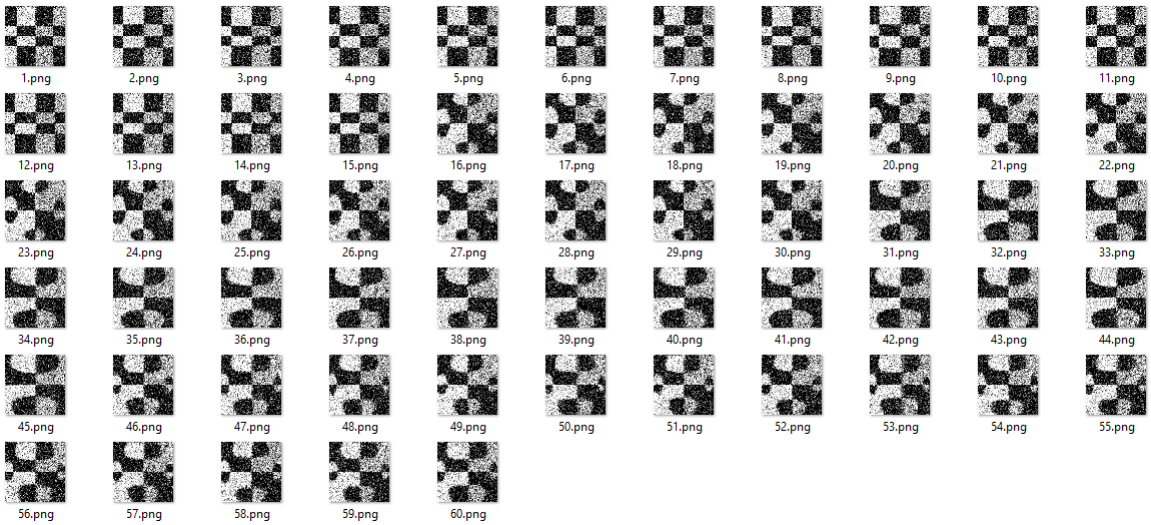
$$= \log\left(\beta^T R\beta\right) - \log\left(\|\beta\|^2 + \max_{\|w\|_\infty \leq 1} \int_\Omega -\beta\nabla\cdot w\right)$$

The optimization is by alternating updates on w and β until convergence. At every step of w, it is reprojected to a sup norm of 1. The gradient ascent updates are

$$\begin{aligned} w^{t+1} &= w^t + \epsilon\left(\gamma\nabla\cdot\beta\right) \\ \beta^{t+1} &= \beta^t + \epsilon\left(\frac{2R\beta}{\beta^T R\beta} - \frac{2\beta - \gamma\nabla\cdot\beta}{\left(\|\beta\|^2 + \gamma\left(-\beta\nabla\cdot w\right)\right)}\right) \end{aligned}$$

4 Experiments

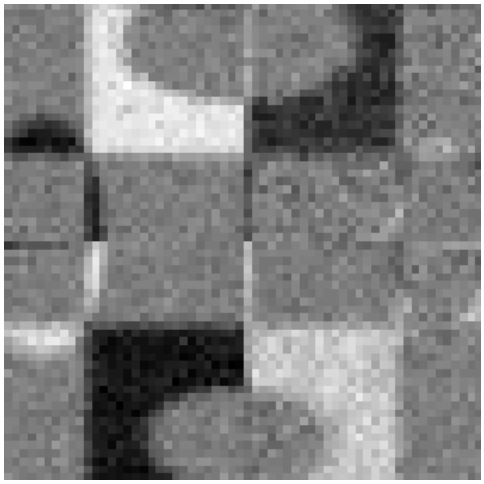
I generated a gallery of transformed images, starting with MATLAB's `checkerboard()` image generating function, and transforming with sinusoidal and pincushion transformations with randomized parameters, and have Gaussian noise added on top for a total of 60 images.

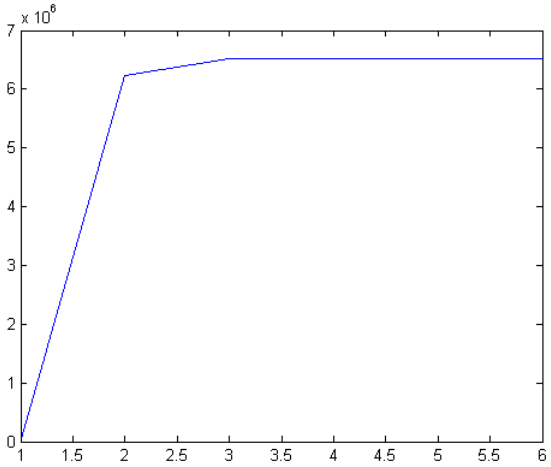


4.1 No penalty

With no penalty, the results are equivalent to doing regular PCA on the images. The true eigenvalue of the top component is $6.57179e6$, which is the energy converged to using both power iterations and gradient descent methods. Power iterations converges to this in less than 6 iterations, but gradient descent took closer to 300 iterations to convergence.

The top principal component resembles along the lines of what some of the original images look like, with quite a bit of noise.

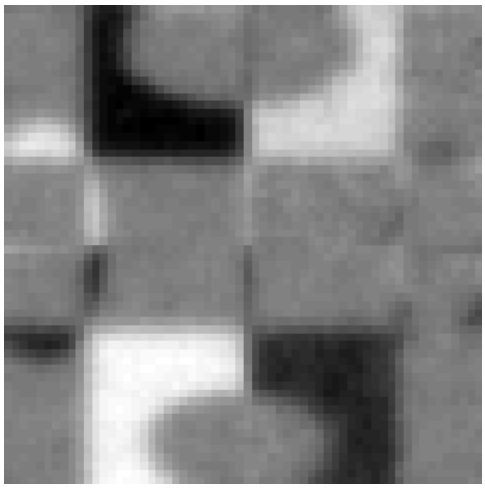


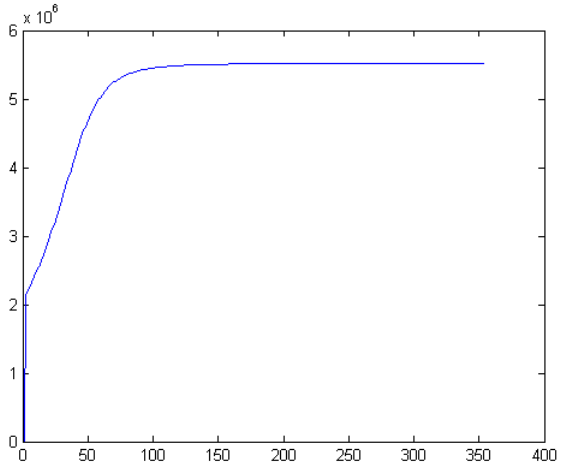


4.2 H1 penalty

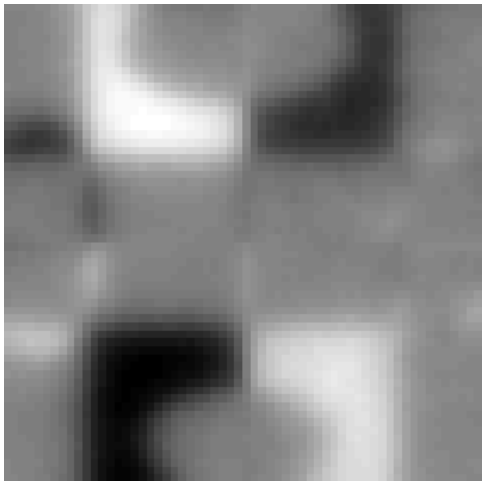
As expected, H1 penalty yields very smooth images that loses definition of the relevant the image features. The plot of the energy, aka the eigenvalue of the corresponding component, increases monotonically over iterations because the optimization is convex.

$\gamma = 2$:

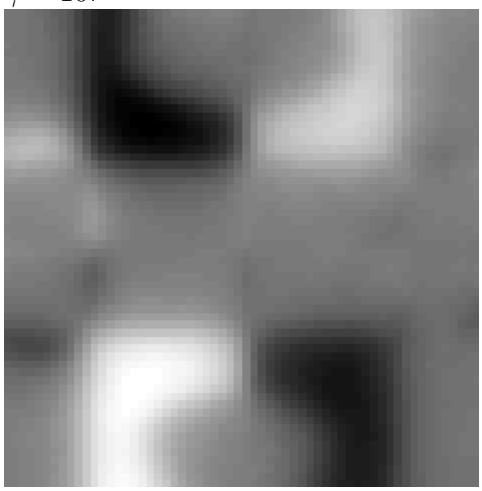




$\gamma = 5$:



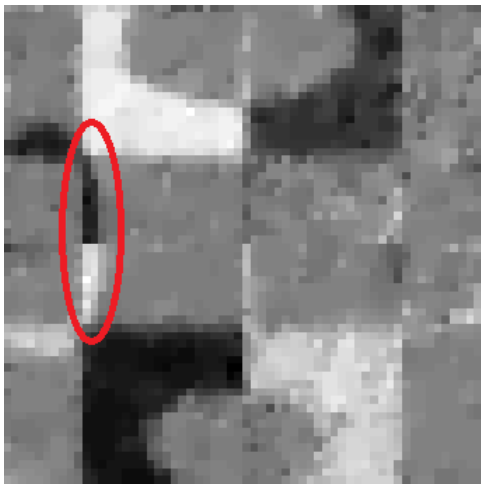
$\gamma = 10$:

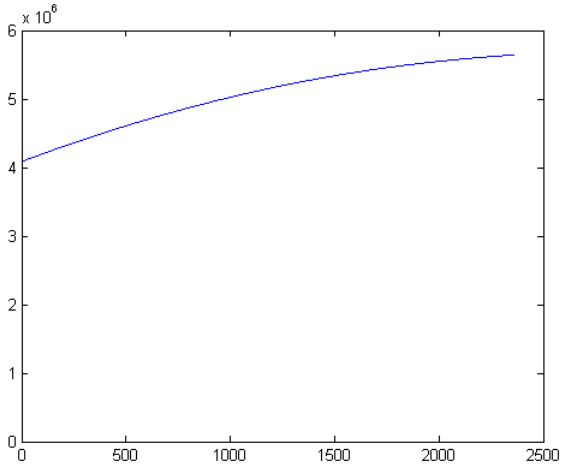


4.3 Total Variation

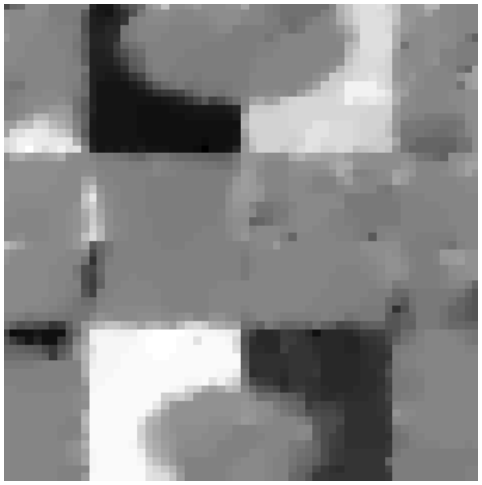
Increasing the penalty gives components that are more and more blocky and uniform. There are some areas of the image such as the below red circled part which distinctly shows the borderlines of where the images differed due to randomized parameters used when generating the images. Total variation took a long time to converge compared to either H1 or no penalty. For whatever reason, either due to a bug or something else which I hope to look into in the future, I could not optimize using the original functional and had to transform with a log on the functional instead. The graph of the energy below is the original functional, but I took derivatives based on its log.

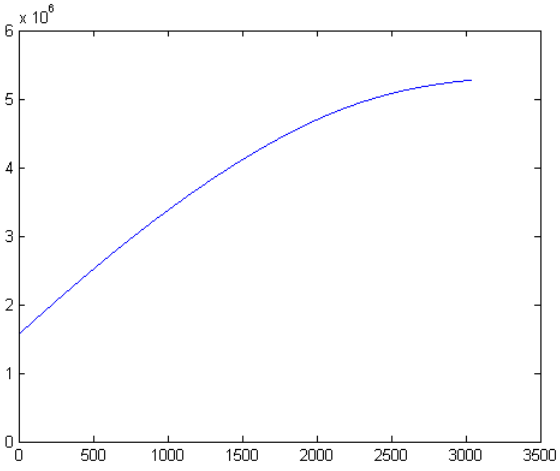
$\gamma = 50$





$\gamma = 100$





5 Discussion

For future work there are a few issues to look into. First, it'd be interesting to look at how good are the reconstruction of the original data are using the principal components, especially with H1 vs total variation penalties. My guess is that if the H1 components are too diffused, the reconstruction of the original would also be very diffused. However, the slowness of the optimization for the total variation case until convergence was an issue for me and made it difficult to get multiple principal components to try out the reconstruction. This is related to the second of issue of looking into ways of imposing the regularization. Here I used regularization by division, with the penalty in the denominator of the objective function. I first tried penalizing by subtraction but the results did not look as expected, though the energy did converge to an optimum. Third, it'd be interesting to study how to take care of the issue of a really big R matrix. Memory restriction means that R can't be too large. Some things to try would be to represent R by another basis instead, or there may be a way to use the PCA "transpose trick" when there are much fewer data samples than there are

dimensions in the images.